Level Set Applications for Visualization

Notes for IEEE Visualization 2007 Course #4
Sacramento, CA

Speakers

David Breen  Drexel University
Aaron Lefohn  Neoptica
Ken Museth  Linköping University
Mikael Rousson  Siemens Corporate Research
Course Schedule

Session 1

1:55  Welcome and Introduction – Breen

2:05  Level Set Approach for the Metamorphosis of Solid Models – Breen
   Level set morphing movements
   Controlling the morph
   Examples
   Morphing different types of solid models

2:25  Contour-Based Surface Reconstruction – Museth
   New morphing speed function
   Tracker particles
   Constructing the speed function
   Results

2:50  Level Set Volume Dataset Segmentation – Breen
   Segmentation framework
   Segmentation of diffusion tensor MRI data

3:15  Level Set Segmentation From Multiple Datasets – Museth
   Segmentation with Edges
   Longe range edge detector
   Partial derivatives with Moving Least Squares
   Results

3:40  Break

Session 2

Latest Developments in Level Set Segmentation – Rousson

4:15  Piecewise Smooth Intensity Model

4:50  Statistical Shape Models

5:15  Ordered Spatial Dependency for Coupled Segmentation

5:30  Interactive Segmentation using GPUs – Lefohn

6:00  Course Ends
Course Abstract

Level set methods, an important class of partial differential equation (PDE) methods, define dynamic surfaces implicitly as the level set (iso-surface) of a sampled, evolving nD function. This course is targeted for researchers interested in learning about the application of level set methods/models to visualization. The course material will be presented by recognized experts in the field, and will include extensive details on a variety of level set applications. The course will be taught at an intermediate level. Therefore attendees should have a working knowledge of calculus, linear algebra, computer graphics and geometric modeling. Some familiarity with differential geometry, differential equations, numerical computing and image processing is strongly recommended, but not required. The course will describe in detail level set methods for 3D morphing, contour-based surface reconstruction, a volume dataset segmentation framework, advanced segmentation techniques that utilize statistical shape models, piecewise smooth intensity models and ordered spatial dependencies. The course will close with a lecture on interactive segmentation with level set models on GPUs.

Speaker Biographies

David Breen is an Assistant Professor in the Computer Science Department at Drexel University. He has held research positions at the Center for Advanced Computing Research and the Computer Graphics Lab at the California Institute of Technology, the European Computer-Industry Research Centre, the Fraunhofer Institute for Computer Graphics, and the Rensselaer Design Research Center. His research interests include level set models for graphics and visualization, medical image analysis and segmentation, geometric modeling and computational biology. He has published over 60 research papers in these and other areas, as well as co-edited the book Cloth Modeling and Animation. Breen received a B.A. in Physics (Colgate University, 1982), and an M.S. and Ph.D. in Computer and Systems Engineering (Rensselaer Polytechnic Institute, 1985 &1993).
E-mail: david@cs.drexel.edu

Aaron Lefohn is a Principal Engineer at Neoptica, an interactive computer graphics startup company in San Francisco. Previously, Aaron was a researcher and graphics software engineer at Pixar Animation Studios, a Ph.D.
student at the University of California Davis, and an M.S. student at the University of Utah. His masters research focused on interactive GPU-based level-set segmentation and visualization, and his Ph.D. work focused on generic GPU data structures and GPGPU-based real-time rendering algorithms. Aaron has given a number of talks on general-purpose computation on graphics hardware (GPGPU) at ACM SIGGRAPH, IEEE Visualization, NVIDIA, ATI, Intel, Stanford, and elsewhere. Lefohn holds a B.A. from Whitman College, two M.S. degrees from the University of Utah, a Ph.D. in computer science from the University of California, Davis, and was an NSF graduate fellow in computer science. E-mail: lefohn@neoptica.com

Ken Museth is a professor of Computer Graphics at Linköping University and an Adjunct Professor at Aarhus University. He received his Ph.D. in computational quantum dynamics from the University of Copenhagen in 1997. From 1998 to 2003 he was a visiting faculty member in the Chemical Physics Department, then a research scientist in the Computer Science Department at the California Institute of Technology. He has also been a scientific consultant to Digital Domain, Rhythm & Hues Studios and NASA’s Jet Propulsion Laboratory. His research activities focus on the areas of deforming geometry and level set methods. E-mail: kenmu@itn.liu.se

Mikael Rousson is a research scientist at Siemens Corporate Research in Princeton, NJ. He received his Ph.D. in 2004 while working with Professor Rachid Deriche in the Odyssee Lab (Laboratory of Computer and Biological Vision of the INRIA/ENS/ENPC) at INRIA, Sophia Antipolis. His Ph.D. thesis research focused on Geometric and Statistical Methods for Image Segmentation. Rousson’s research interests include shape and image statistics, medical imaging and machine learning. E-mail: mikael.rousson@siemens.com

Level Set Web Sites

Level Set Applications for Visualization Course Page
http://www.cs.drexel.edu/~david/Viz07_LS_Apps

Breen - Geometric Modeling and Deformable Models
http://www.cs.drexel.edu/~david/geom_mod.html
http://www.cs.drexel.edu/~david/deform_mod.html
Lefohn Home Page
http://graphics.cs.ucdavis.edu/~lefohn

Museth Home Page
http://gg.itn.liu.se

Rousson Home Page
http://mikael.rousson.googlepages.com

Whitaker Home Page
http://www.cs.utah.edu/~whitaker

VISPack Web Site
http://www.cs.utah.edu/~whitaker/vispack

ITK Web Site
http://www.itk.org

Osher Home Page
http://www.math.ucla.edu/~sjo

UCLA CAM Technical Reports
http://www.math.ucla.edu/applied/cam

Level Set Systems, Inc.
http://www.levelset.com

Fedkiw Home Page
http://www.graphics.stanford.edu/~fedkiw

Sethian Home Page
http://www.math.berkeley.edu/~sethian

Rumpf Home Page
http://numerik.math.uni-duisburg.de/people/rumpf/rumpf.shtml

Strzodka Home Page
http://numerik.math.uni-duisburg.de/people/strzodka/strzodka.htm
Related Papers

• R.T. Whitaker, “A Level-Set Approach to 3D Reconstruction From Range Data,” 

**Related Books**

Table of Contents

Session 1

Level Set Approach for the Metamorphosis of Solid Models (Slides)
  D. Breen
Level Set Approach for the Metamorphosis of Solid Models (Slides)
  D. Breen and R. Whitaker
Surface Reconstruction Via Contour Metamorphosis: An Eulerian Approach With Lagrangian Particle Tracking (Slides)
  O. Nilsson and K. Museth
Surface Reconstruction Via Contour Metamorphosis: An Eulerian Approach With Lagrangian Particle Tracking
  O. Nilsson, et al.
Level Set Segmentation of Biological Volume Datasets (Slides)
  D. Breen, et al.
A Framework for Level Set Segmentation of Volume Datasets
  R. Whitaker, et al.
Level Set Modeling and Segmentation of DT-MRI Brain Data
  L. Zhukov, et al.
Level Set Segmentation From Multiple Non-uniform Volume Datasets (Slides)
  K. Museth
Level Set Segmentation From Multiple Non-uniform Volume Datasets
  K. Museth, et al.

Session 2

Latest Developments in Level Set Segmentation (Slides)
  M. Rousson
Efficient Segmentation of Piecewise Smooth Images
  J. Piovano, et al.
Implicit Active Shape Models for 3D Segmentation in MR Imaging
  M. Rousson, et al.
Efficient Kernel Density Estimation of Shape and Intensity Priors for Level Set Segmentation
  M. Rousson and D. Cremers
DTI Segmentation by Statistical Surface Evolution
  C. Lenglet, et al.
Review of Statistical Approaches to Level Set Segmentation:
Integrating Color, Texture, Motion and Shape
  D. Cremers, et al.
A General Framework for Image Segmentation Using Ordered Spatial Dependency
  M. Rousson and C. Xu
Interactive Level-Set Deformation on the GPU (Slides)
A. Lefohn
GIST: An Interactive, GPU-Based Level-Set Segmentation Tool for 3D Medical Images
J. Cates, et al.
Session 1
Level Set Approach for the Metamorphosis of Solid Models

David E. Breen
Computer Science Department
Drexel University

Ross T. Whitaker
School of Computing
University of Utah

What is a Level Set Model?

• A deformable implicit model, $\phi(X,t) = 0$
• Sampled representation of dimension $n+1$
  – Images (2D) represent curves
  – Volumes (3D) represent surfaces
• Change level set by modifying samples
• Change sample values by solving a PDE
This is a Level Set Model!

of a curve!

Changing the image moves the curve

- Red curve is defined by $\phi(x,y) = 127$
What is a Level Set Model?

- \( \phi(X) \) is not defined by a specific equation
- \( \phi(X) \) is represented by a regular 3D sampling
  - Signed distance volume dataset
- Level Set model is deformed by evolving the Level Set equation on the sampling
  - Osher & Sethian 1988
- \( \partial \phi(X) / \partial t = |\nabla \phi(X)| F(X, ...) \)
- Connects changes in sample values to changes of the level set curve/surface

Level Set Speed Term

- LS model deformation is controlled by \( F() \)
- \( F(X, ...) \) defines the speed of the LS surface in direction of the surface normal \( N \) at each point \( X \) on the surface
- \( F() \) is defined by a user for each application in order to achieve a specific goal
- We have defined many different speed terms for our CG applications
Advantages of LS Models

- Always produce closed, non-self-intersecting (simple) surfaces

Deforming Mesh May Self-Intersect
No Self-Intersection with Level Set Deformations

Important for GCAD, CAM and Analysis

Figurine automatically manufactured from a level set model
Advantages of LS Models

- Easily change topological genus
  - Holes may close or open
  - Separate pieces come together/split apart
- Ideal for complex deformable models of unknown genus

Mug-to-Chain Morph
Advantages of LS Models

• Concise interface for control $\rightarrow F()$
• Free of mesh connectivity and quality issues
• No need to reparameterize during deformation

Disadvantages of LS Models

• No inherent parameterization
  – Problem for texture mapping & some deformations
• Computationally expensive (?)
• High memory requirements (?)
  – Narrow Band methods are $O(\text{surface area})$
    Adalsteinsson & Sethian 1995, Whitaker 1998,
    Peng et al. 1999
• Cannot control genus (?)
  – Han et al. 2001, 2003, Bischoff & Kobbelt 2003,
    Ségonne et al. 2005
• Cannot represent fine or sharp features (?)
  – Losasso et al. 2004, Houston et al. 2006,
    Nielsen & Museth 2006
Level Set Models Summary

- Sampled implicit function $\phi(\tilde{x})$ (a volume)
- Deform an iso-surface (i.e. a level set)
- Solve Level Set PDE on grid
  $$\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot F (\tilde{x}, D\phi, D^2\phi, \ldots)$$
- Moves iso-surface by changing voxel values
- Define $F(\ )$ to achieve goal

3D Volumetric Metamorphosis
**Volumetric 3D Morphing**

- Two-stage process
  - Global warping
    Make initial shape similar to the target shape
    - Rigid transformations
    - Non-rigid transformations based on correspondences
  - Blending → Interpolate voxels to fill in details

![Diagram showing the process of Volumetric 3D Morphing](image)

**Level Set Morphing**

- Previous work (Lerios et al. 1995, Cohen-Or et al. 1998) primarily focused on warping
- Level set models provide a superior blending capability
  - Active, deformable implicit surfaces
  - Requires no user input (warping)
  - User input (warping) incrementally applied, as needed
  - Guarantees that source shape becomes target shape
  - Sub-voxel accuracy
  - No ghosting from voxel interpolation
Level Set Morphing Movements

- Each point on surface moves in the direction of local normal. Step-size proportional to signed distance to target $\gamma_B$

\[
\frac{\partial \phi(x)}{\partial t} = \nabla \phi(x) \left| \gamma_B(x) \right|
\]

- Regions inside expand
- Regions outside contract
- Guaranteed convergence
- Not moving points!
**Metamorphosis Steps**

- Distance volume A is initialization of level set evolution
- $\gamma_B$ is derived from distance volume B
- Solve LS equation on A’s grid (volume) until surface motion stops (A becomes B)
  - Track zero level set to sub-voxel accuracy
  - Compute only in narrow band $\Rightarrow O(n^2)$
- At each step extract iso-surface from volume
- Incrementally apply warping & render

**Simple Example**

- Morph a ball into a ‘C’
- Initial overlap
Interpolate Distance Volumes

- Pieces “pop out of nowhere”

Level Set Deformation

- Ball grows into the C
Controlling the Morph

- How initial object overlaps with final object determines how morph will proceed
- User interactively specifies initial overlap
- Automatic initial model alignment is an option
  - Matches center-of-mass and principal axes
- Correspondences may be used to specify a warping
  - Provides additional user control

Different Alignments Produce Different Morphs
**Mug-to-Chain Morph**

**Initial Conditions**

Level Set models easily change genus
Mug-to-Chain Morph

Color-Shading Level Set Models with Distance, Color and Closest-Point Volumes
Color-Shaded Dart-to-Jet Morph

- Colors at the closest points on original models are interpolated over time

Morphing Challenge

- How to morph between these three models?

 Polygonal Mesh

 CSG Model

 MRI Scan
Morphing Between Different Types of Models

- Combining
  - A variety of scan conversion algorithms
  - A flexible metamorphosis (morphing) technique based on level set models

- Produces
  - A technique for morphing between different types of geometric models

Results - MRI Scan to Polygonal Surface
Complete Morphing Sequence
1 Minute of Fame

Tar Monster Morphing Sequence
Scooby-Doo 2, 2004

Summary

• Surface moves in direction of local normal with a speed proportional to signed distance to target object
• Source is guaranteed to become target
  – Given some initial overlap
• Provides superior blending operation
• Requires no user input, but may be applied incrementally to produce desired result
Acknowledgments

• Thanks to Al Barr, Sean Mauch, Jia Mao, Ben Houston, Frantic Films
• Supported by NSF, NIH, ONR, Caltech
• Datasets provided by Caltech Multires Modeling Group, Utah’s SCI Institute

Related Publications

A Level-Set Approach for the Metamorphosis of Solid Models

David E. Breen, Member, IEEE Computer Society, and Ross T. Whitaker, Member, IEEE

Abstract—This paper presents a new approach to 3D shape metamorphosis. We express the interpolation of two shapes as a process where one shape deforms to maximize its similarity with another shape. The process incrementally optimizes an objective function while deforming an implicit surface model. We represent the deformable surface as a level set (iso-surface) of a densely sampled scalar function of three dimensions. Such level-set models have been shown to mimic conventional parametric deformable surface models by encoding surface movements as changes in the grayscale values of a volume data set. Thus, a well-founded mathematical structure leads to a set of procedures that describes how voxel values can be manipulated to create deformations that are represented as a sequence of volumes. The result is a 3D morphing method that offers several advantages over previous methods, including minimal need for user input, no model parameterization, flexible topology, and subvoxel accuracy.

Index Terms—Level set method, morphing, solid model, distance function, animation, volume graphics, optimization, deformable model.

1 INTRODUCTION

Shape metamorphosis is a process where an object continuously changes its own shape into the shape of another object. Within the computer graphics community, metamorphosis (the vernacular for metamorphosis) has been used frequently for special effects in movies, advertising, and entertainment. Image morphing takes a 2D image of an object and transforms the appearance of that object into the appearance of another object, with the goal of producing natural-appearing, or at least sensible, intermediate images. Three-dimensional morphing or shape morphing involves smoothly changing the model of one object into the model of another object. Shape morphing algorithms have been developed for both surface models and volumetric models. The surface model algorithms transform the surface patches (usually polygons) of the source model into the surface patches of the target model. The volume-based morphing algorithms represent 3D objects as volumes and manipulate the voxel values of volumes in order to make one object become another. Volume data sets needed for this process may be acquired directly from 3D scanning devices, such as MRI or CT, or they may be generated via 3D scan conversion of solid geometric models.

Despite the increased complexity and computation time associated with morphing 3D shapes rather than 2D images, shape morphing does have some distinct advantages. First, image morphing is directly tied to the views of the two input images. Any morphing effect must be based only on the information available in the two images. This prevents the animator from making camera, lighting, or shading changes during the morph. If the morph is performed with 3D techniques, the animator is provided with much more flexibility in presenting the results of the morph. Once a sequence of transforming models has been generated, the animator may experiment with a variety of camera angles and motions for viewing the models. The scene’s global lighting and shading parameters may also change during the morph. A second advantage is that a 3D morph ensures that the resulting rendered images represent a continuous sequence of 3D shapes. In image morphing, maintaining the 3D feasibility of the intermediate images is the responsibility of the user, i.e., care must be taken to make sure intermediate images represent pictures of feasible objects rather than a fuzzy mixture of ghost-like objects.

One disadvantage of 3D morphing is that it requires two 3D models. It is not always possible to acquire accurate models of the objects that one wishes to morph, e.g., a sequence of familiar faces. At present, digital photographs of real objects tend to capture more visual detail than 3D models of the same. Despite this, there is a class of morphing problems for which 3D models exist or are easily obtainable which lend themselves to the application of shape morphing.

In order to compare the adequacy of different approaches to 3D shape morphing, we have identified several desirable aspects of a 3D morphing technology. These properties are:

1. The transition process should begin with a source surface and end with a specified target surface.
2. Intermediate surfaces should undergo continuous 3D transitions (rather than continuity only in the image space).
3. The morphing algorithm should apply to a wide range of shapes and topologies.
global, geometric warping techniques [11], [26], is based on the image morphing
process. In image morphing and volumetric shape morphing, the blending step is usually achieved by a simple linear interpolation, or cross-dissolve, of the two images/volumes. The warping stage of image/volume morphing has been studied extensively by other researchers and our work primarily focuses on developing a superior method for the blending stage, one which reduces the amount of user input required to produce an acceptable morphing result. Indeed, the proposed level-set approach to shape metamorphosis will produce a reasonable morph with virtually no user input. Given this technology, user input can be incorporated in an incremental fashion, starting with a minimal amount and proceeding with progressively more until a satisfactory result is achieved.

Our morphing approach consists of several stages. First, the source and target objects are defined, with Constructive Solid Geometry (CSG) models in our examples. The models are scan converted into 3D distance volumes (a signed distance transform), where the shortest distance to the solid model is stored at each voxel. We use the sign convention that distance is positive inside the object and negative outside the object. A level-set model is fit to the zero level set of the source model distance volume. A coordinate transformation provides a mapping from every point in the domain of the level-set model into the distance volume of the target model. This is the warping step. The level-set model is then allowed to deform using the signed distance transform of the target model. Each point on the source surface moves in the direction normal to the surface at that point with a velocity proportional to the signed distance transform of the target object at that point in 3-space. This process is the blending step of the 3D surface morph. At user-defined time intervals, the level-set model is converted into a polygonal model. This model is expressed in the coordinate system of the source model. Therefore, the coordinate transformation is incrementally applied, as a function of time, to the sequence of polygonal models, which are then rendered to produce the frames of the morphing animation.

In general, an animator controls the morph by defining how the models overlap, using a variety of coordinate transformations. This can include rigid transformations, scalings, and nonrigid transformations determined by sets of user-defined fiducials. Our current implementation also provides an automated mechanism for determining rotation, translation, and nonuniform scaling and thereby offers the option of a totally automatic morphing algorithm. The system accomplishes this by overlaying the centroids of the two models, aligning their principal axes, and nonuniformly scaling along these axes. The surface of the level-set model only moves in those areas where the target and the source surfaces are not brought together perfectly by the coordinate transformation. The portion of the level-set model which is outside of the target model will shrink, while the portion inside the target model will expand to become the final shape.

The level-set approach to 3D model morphing provides several advantages over other 3D morphing algorithms. Level-set models are active; their underlying motion is

![Fig. 1. Shape morphing, like image morphing, can be described as a warping combined with a blending.](image-url)
defined algorithmically rather than interactively, guaranteeing that the source shape will become the target shape, given that both models initially overlap. The amount of user input required to produce a morph is directly proportional to the amount of control the animator wishes to impose on the process. The animator may allow the system to automatically generate the morph or he/she can employ a full 3D warping to describe the morph, with the level sets simply providing a small fine-tuning of the surface. Any intermediate amount of user input will produce a reasonable morph. Because level-set models do not rely on any kind of parameterization, they do not suffer the problems of parametric surfaces, e.g., a limited set of possible shapes and the need for reparameterization after undergoing significant changes in shape. This lack of parameterization, along with no direct representation of topological structure, allow level-set models to easily change topology while morphing. The model can “split” into pieces to form multiple objects. Conversely, several disjoint objects may come together to make a single object. Finally, both the scan conversion and deformation stages of our morphing approach produce models with user-defined subvoxel accuracy. This provides a flexible time-quality trade-off and generates superior results at low volume resolutions as compared to previous work.

The remainder of the paper proceeds as follows: Section 2 describes related morphing work. Section 3 describes the novel aspect of our method—shape interpolation based on the optimization of a similarity function using level-set models. It is followed by a discussion of general properties and numerical methods. Section 6 describes all stages of our morphing approach. The paper closes with three morphing examples, conclusions, and some thoughts on future work.

2 RELATED WORK

In recent years, numerous 3D morphing algorithms have been reported in the literature. These algorithms generally fall into two categories, surface-based approaches and volume-based approaches. The former primarily consists of those methods that continuously change one polygonal surface into another. Volumetric methods modify the voxel values of a volume data set in order to smoothly transform the object defined by the volume from a source shape into a target shape. A third class of algorithms involves morphing implicit models, either by transforming the underlying structure of “blobby” models [14] or by creating higher dimensional interpolating implicit functions [39].

Kent et al. [23] described the fundamentals for morphing 3D polygonal surfaces. The first step is to create a single topological description of vertices, edges, and faces which contains the combined topological structure of both the source and target surfaces (the correspondence phase). The vertices of the structure are then interpolated between their position on the initial surface to their position on the target surface to create the morphing result. Parent [31] presents an improved technique for establishing vertex and edge correspondences when creating the combined topological structure. Chen et al. [9] applied the same approach to polygonal surfaces defined in cylindrical coordinates. Kanai et al. [21] use harmonic maps to define the correspondences in the combined topological structure. Lazarus and Verroust [24] do not build a common adjacency graph, but instead create a common parameterized mesh during a sampling process. Gregory et al. [17] present a method to assist the user in defining these correspondences. Lee et al. [25] apply many of these surface-based methods to morphing multiresolution meshes. Kaufl and Rossignac [22], [34] introduce the notion of an interpolating polyhedron, an application of mathematical morphology to face sets. They also describe a user control methodology based on a Bezier curve paradigm. DeCarlo and Gallier [12] demonstrate that it is possible (with significant effort) to morph polygonal models of differing genus. Surface-based methods are important because of the wealth of polygonal models available to animators, but they are burdened with the difficult task of creating a single topological structure which can represent the source and target surfaces. While it has been shown that changing the genus of a morphing polygonal surface is possible, for example, from a sphere to a torus, it can be a difficult and tedious process which requires significant user input. Another troubling feature of surface-based methods is the problem of self-intersection. These methods cannot guarantee that polygonal surfaces will not pass through themselves, creating physically nonsensical intermediate results.

Embedding the morphing surfaces in volumes alleviates these problems. Hughes [19] demonstrates how volumes can be used to create a morph between objects of different genus. The approach involves linearly interpolating the Fourier transforms of the volumes. A schedule is used during interpolation which filters out the high frequencies of the initial volume while interpolating the low frequencies of both volumes and gradually adds the high frequencies of the target volume. He et al. [18] propose a similar approach using wavelets to control the high-frequency artifacts. They also developed methods that allow the user some control of the interpolation by establishing explicit correspondences between the volumes. Lerios et al. [26] describe a morphing method which is a 3D extension of Beier and Neely’s [4] 2D (image) morphing technique. The first step is to apply a user-defined geometric warp to the source object in order to deform it into approximately the same shape as the target object. This step allows the user to specify corresponding features on the two objects. The second step interpolates the values between matching voxels in the (warped) source and target volumes. Chen et al. [10] propose disk fields as a superior instrument for specifying the warps needed for this kind of 3D morphing. Payne and Toga [32] describe a method for changing one volumetric model into another by interpolating the distance fields generated from the two volumes. A distance field, or distance volume, is a volume data set where the value stored at each voxel is the shortest distance to the surface of the object being represented by the volume. Cohen-Or et al. [11] improve upon this approach by including a two-part warping step that calculates a rigid transformation and an elastic warping based upon user-supplied anchor points. This extra step approximately aligns the target and final objects and provides significant user control to the overall morphing process.
The proposed volume-based morphing technique shares many of the advantages of previous volume-based methods. It can easily morph objects of different genus. It is not burdened with the difficult vertex/edge/face bookkeeping of surface methods and surfaces cannot pass through each other. Additionally, our method provides a novel blending/interpolation mechanism that will not produce the ghosting or spontaneous generation of new objects which is possible when simply interpolating voxel intensity or distance values (for example, see Figs. 4, 5, and 6). Our method does not require user input to produce a reasonable morph, but can easily incorporate previously published warping techniques to provide a wide range of animator control. Finally, because we calculate the distance transform to subvoxel accuracy and effectively track a deforming surface within voxels, our results do not produce the aliasing artifacts commonly found in other distance-based, volumetric approaches.

3 Metamorphosis as a Goal-Driven Process

Morphing strategies may be built upon any number of underlying principles. For instance, surface metamorphosis could consist of a sequence of time slices from a four-dimensional manifold which optimizes some space-time criteria [39]. The proposed strategy for object metamorphosis is based on yet another principle: Shape metamorphosis is the process by which one object seeks to resemble another. This philosophy raises two questions. First, what is the metric by which we can quantify the similarity of two objects? Second, what is the process by which one object seeks to optimize that metric? This paper shows that even very simple answers to these two questions produce very powerful algorithms for shape metamorphosis, with a great deal of opportunity for future enhancement and refinement of the resulting algorithms.

In order to create a very general algorithm for metamorphosis, we work with a very general notion of a surface. Consider an open set $\Omega_B \subset \mathbb{R}^3$, which is the source object, and a target $\Omega_T \subset \mathbb{R}^3$. The source, $\Omega_B$, is enclosed by a surface $S_B = \partial \Omega_B$ and, likewise, $S_T = \partial \Omega_T$. We require that the objects be compact and lie in some finite domain $U \subset \mathbb{R}^3$. Notice that we do not require any specific connectivity or topology, which means that each object could consist of a set of disconnected pieces (each with any number of holes) all sitting in $U$.

We propose a very simple metric for comparing two shapes that maximizes the volume shared by the interiors of the two objects. First, define an inside-outside function for the target, $\gamma_B : \mathbb{R}^3 \rightarrow \mathbb{R}$ for $\Omega_B$ such that

$$
\begin{align*}
\gamma_B(x) &= 0 \quad \forall x \in S_B \\
\gamma_B(x) &> 0 \quad \forall x \in \Omega_B \\
\gamma_B(x) &< 0 \quad \text{otherwise}.
\end{align*}
$$

The inside-outside function $\gamma_B$ can be used to quantify the extent to which an intermediate object $\Omega_i$ overlaps with the target, $\Omega_B$, with the volume integral

$$
\mathcal{M}_{\Omega_B, \Omega_i} = \int_{\Omega_i} \gamma_B(x) \, dx.
$$

Notice, this integral achieves its maximum,

$$
\Omega_m = \arg \max_{\Omega_i} \left( \int_{\Omega_i} \gamma_B(x) \, dx \right),
$$

when $\Omega_m = \Omega_B$ because, in that case, the integral includes all of the positive parts of $\gamma_B$ and none of the negative parts.

We can compute the first variation of this metric with respect to $\Omega_i$ by noting that incremental changes in the object shape can be expressed in terms of the surface that encloses it:

$$
\int_{\Omega_i+dt} \gamma_B(x) \, dx = \int_{\Omega_i} \gamma_B(x) \, dx + \int_{S'_t} \gamma_B(x) \epsilon(x) \cdot N(x) \, dx,
$$

where $N : S_t \rightarrow S^3$ is the surface normal, which is a mapping from every point on the surface to the unit sphere. Differentiating with respect to $\epsilon$ gives the first variation [13] with respect to surface position:

$$
dM = \gamma_B(x)N(x).
$$

Using the first variation from (5) and a hill climbing strategy, we obtain the surface motion, defined for each surface point, that minimizes our similarity metric:

$$
\frac{ds}{dt} = \gamma_B(s(t))N(s(t)) \quad \forall s(t) \in S_t.
$$

This equation states that each point on the surface $S_t$ moves at each time step in the direction of the surface normal $N(s(t))$ with a step size proportional to the value of the inside-outside function $\gamma_B(s(t))$ at the point location.

4 Properties

This section describes the properties of the solutions of the partial differential equation given in (6). For this paper, we examine the case of simple, closed, surfaces, but the results easily generalize to more complicated surfaces. In this section, we systematically show that if the initial and target objects overlap, the final solution of the metamorphosis will be identical to the target.

4.1 Steady-State Solution

We begin by examining properties of the steady-state solution of (6), which is the surface $S_t$ such that $\partial s(t)/\partial t = 0$ for all $s \in S_t$. Thus, for steady state, which we denote $S_\infty$, $\gamma_B(x)N(s) = 0 \Rightarrow \gamma_B(s) = 0 \quad \forall s \in S_\infty.

The inside-outside function for the target, $\gamma_B$, has the property that

$$
\gamma_B(x) = 0 \iff x \in S_B.
$$

Thus, we can conclude $S_\infty \subseteq S_B$. From this, it follows that if $S_\infty$ is compact and closed, it must have one of two forms:

1. $S_\infty = \emptyset$
2. $S_\infty = S_B$.  


The first case is trivial because there are no points on $S_\infty$ to violate the condition given in (7). The second case follows from the fact that simple, closed, connected surfaces have no proper subsets that are also closed surfaces. Thus,

$$S_\infty \subseteq S_B \text{ and } S_\infty \text{ closed } \Rightarrow S_\infty = S_B. \quad (9)$$

Also, because the first variation of the metric $\mathcal{M}_{\Omega_t, \Omega}$ is zero only where $S_t = S_B$, it achieves its maximum for that solution with $\Omega_t = \Omega_B$.

Another important aspect of the algorithm is that a component of the source that overlaps with a component of the target will deform in such a way that it takes on the shape of that target component.\(^1\) This follows from the fact that simple, closed, connected surfaces have no proper subsets that are also closed surfaces. Thus, from the fact that simple, closed, connected surfaces have no proper subsets that are also closed surfaces. Thus, any intermediate surfaces are far from the target. If $f(a) = a$, then $S_t$ contracts or expands with a magnitude that depends on the signed distance to the target.

4.2 Strategy

This formulation suggests a strategy for shape metamorphosis: Construct a family of objects $\Omega_t$ (with a corresponding $S_t$), with $\Omega_{\mid t=0} = \Omega_s$ that evolve according to a hill-climbing strategy, and maximize $\mathcal{M}$. If properly initialized (i.e., all of the components of $\Omega_B$ have some overlap with $\Omega_s$), then the deformation will seek the global maximum, which is the target $\Omega_B$.

The intermediate shapes contained in $\Omega_t$ depend, of course, on the choice of $\gamma_B$. In order to avoid numerical difficulties and to avoid discontinuities in the solution, $\gamma_B$ should be continuous.\(^2\) Furthermore, $\gamma_B$ should reward shapes that are similar to the target but offset by some small distance. That is, $\gamma_B$ should carry information about the shape of the surface into 3D so that shapes tend to “look like” the target as they get nearer. This suggests that a natural choice of $\gamma_B$ is the signed distance transform $[2], [5]$ of the target surface $S_B$ or some monotonic function thereof.

Thus, if we let $D_B : U \rightarrow \mathbb{R}$ be the signed distance transform, $\gamma_B(x) = f(D_B(x))$, where $f(0) = 0$ and $f'(a) > 0$. By tuning $f$, one can control the way the metamorphosis behaves when intermediate surfaces are far from the target. If $f(a) = a$, then $S_t$ contracts or expands with a magnitude that depends on the signed distance to the target.

As it stands, this process is a “low-level” approach to shape metamorphosis, which can form continuous deformations for shapes that are somehow “close” in their initial shapes. As described in the introduction, this low-level process is meant to address the blending stage of 3D metamorphosis—it is meant to be combined with a higher-level process which accounts for “semantic” aspects of shape correspondence. Incorporating user input is important for any shape morphing technique because, in many cases, finding the best set of transition shapes depends on context. Only users can apply semantic considerations to the transformation of one object to another. Our assertion, however, is that this underlying coordinate transformation can achieve only some finite

\(^1\) A target object may consist of several disjoint components.

\(^2\) In general, signed distance transforms are C0 continuous, but not C1 continuous.
similarity between the “warped” source model and the
target and even this may require a great deal of user input.
In the event that a user is not able or willing to define every
important correspondence between two objects, some other
method must “fill in” the gaps remaining between the
source and target surface. We propose the use of deformable
level-set models to achieve a continuous transition (blending)
between the shapes that result from the underly-
ing coordinate transformation.

This higher-level information affects the blending pro-
cess through a 3D coordinate transformation, \( T \). \( T \) maps
a point \( x \) in the coordinate system of the source \( \Omega_A \) into
the coordinate system of the target \( \Omega_B \). As described in
the introduction, this transformation is meant to be quite
general; it can accommodate a wide range of global
deformations. However, if the blending is sufficiently
powerful and can adequately deal with significant shape
discrepancies (after the coordinate transformation), only a
crude alignment of the source and target is necessary. The
coordinate transformation enters into the deformation
process through the term \( \gamma_B \), which quantifies the proximity
of one surface to another by means of the distance
transform. If we express the distance transform for the
target in coordinates of the target, then we have
\( \gamma_B(x) = f(D_B(T(x))) \). Alternatively, one could resample
the distance transform of \( \Omega_B \) in the coordinate system of \( \Omega_A \),
i.e., \( \gamma_B'(x) = \gamma_B(T(x)) \).

5 LEVEL-SET MODELS

In practice, the strategy of shape metamorphosis by surface
deformations described in the previous section must be
computed using some specific surface representation.
Ideally, we would like the surface representation to be as
general as the underlying theory. For this we use the method
of level-set models [37], which is a technique for
modeling surfaces as iso-values of a densely sampled scalar
function over the domain \( U \). Surface movements are
encoded as changes in the grayscale values of the voxels.
Using level-set models, we can compute goal-driven
deformations on surfaces without any explicit surface
representation.

5.1 Theory

The strategy of level-set models represents a set of surface
points \( S \) as an iso-surface of \( \phi \), which we will call the
embedding:

\[
S = \{x | \phi(x) = k\},
\]

where the value of \( k \) is arbitrary and will fall out in
subsequent calculations.\(^3\) We can also represent a family of
surfaces and a corresponding family of embeddings:

\[
S_t = \{x | \phi(x, t) = k\}.
\]

Consider a point \( s(t) \) on the surface that moves through
space as a function of \( t \). Because \( s(t) \) remains the \( t \)th level-
surface of \( \phi \) over time, the total derivative of \( \phi \) with respect
time must be zero. Thus,

\[
\frac{\partial \phi(s(t), t)}{\partial t} + \nabla \phi(s(t), t) \cdot \frac{ds(t)}{dt} = 0,
\]

which establishes the connection between the way points on
\( S \) move and the way the grayscale values (at positions on
the surface) of the embedding change. We can rewrite (14)
as follows:

\[
\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{ds(t)}{dt} = |\nabla \phi| \frac{ds(t)}{dt} \cdot N(s),
\]

using the fact that \( \nabla \phi = |\nabla \phi| N \).

We can now “plug” into (15) any surface motion we wish
to compute. Thus, inserting \( ds(t)/dt \) from (6), which
describes the motion of the surface model as it becomes
more like the target, yields

\[
\frac{\partial \phi(s)}{\partial t} = |\nabla \phi| \gamma_B(s(t)),
\]

where we have used the fact that surface normal is unit
length, i.e., \( N \cdot N = 1 \). Notice, we have not chosen a
particular \( k \) and, therefore, this analysis applies to every
level set of \( \phi \). Thus, we are describing the deformation of an
embedded family of surface models each of which evolves
according to the same equation:

\[
\frac{\partial \phi(x)}{\partial t} = |\nabla \phi(x)| \gamma_B(x).
\]

The particular level set of interest is the one that we choose,
by construction of the initial conditions, such that
\( \phi(x, 0) = k \ \forall \ x \in S_A \).

The deformation process, defined by (6) and (17), is
further detailed in Fig. 3 with a two-dimensional example.
Given the inputs \( \Omega_A \) and \( \Omega_B \), surface \( S_t \) will begin on the
surface of \( S_A \). Each point on the surface will move in the
direction of the normal at that point with a velocity
proportional to the signed distance at that point in 3-space
from \( S_B \). Those parts of \( S_t \) that are outside of \( \Omega_B \) will
contract because \( D_B \) is negative in those regions. The parts
of \( S_t \) inside of \( \Omega_B \) will move in the direction of the surface
normal and will expand. Segments of the surface farther
away from \( S_B \) will move faster toward \( S_B \) than the
segments closer to the surface. As segments of \( S_t \) reach
the surface of \( \Omega_B \), they will no longer move because \( D_B \) goes
to zero in these regions.

A three-dimensional morphing example is presented in
Figs. 4 and 5. Fig. 4 presents the initial configuration for the
morphing sequence. A small sphere will morph into the
larger “C” shape. The two objects are first rendered as
translucent objects to give some indication of their initial

\(3\) For the remainder of this paper, we use the convention that \( k = 0 \) is the
level set of interest.
overlap. The ball is mostly contained inside the “C” shape, with the light blue regions of the sphere protruding outside the red “C.” The morphing sequence in Fig. 5 demonstrates that the regions outside of the “C” contract slightly to fit to the surface of the “C.” The surface regions of the sphere inside of the “C” expand to fill the remainder of the object. This also demonstrates that having a surface expand in the direction of its local normal will allow it to deform to fit concave objects, a fact also shown by Miller et al. [29]. Fig. 6 contrasts our method of blending with the method used in several other volume-based morphing techniques. Here, the voxel values of the two initial distance volumes are simply interpolated. It can be seen that voxel interpolation produces undesirable artifacts, namely pieces of the “C” shape “pop out of thin air.” This is a typical problem when using voxel interpolation to blend volumetric objects. In order to overcome this problem in this example, the sphere would have to be geometrically warped into approximately the same shape as the “C.” Our method requires no warping. The user specifies the initial overlap of the two objects and the level-set deformation completes the morph.

The complete deformation strategy is as follows: First, initialize a volume so that the ith level set is, approximately, aligned with $S_B$. For all of our work, we will use the zero-set as the level-set model. This initialization can be done by using the discrete distance transform of $S_B$, which we compute from CSG models using the method of Breen et al. [7], [8]. We use this initialization to solve the initial value problem given by (17), using the distance transform of the target as the $\gamma_B$. We solve this equation using finite forward differences, as described in the next section. When the model is sufficiently close to the target (a threshold on the RMS distance to the target), the process stops and the metamorphosis is complete.

### 5.2 Implementation

#### 5.2.1 Numerical Issues

The solutions to the partial differential equations described in Section 5.1 are computed using finite differences on a discrete grid. The use of a grid and discrete time steps raises a number of numerical and computational issues that are important to the implementation.

The first issue is the discrete approximation of the derivatives in (17). Let $u^n$ be a discrete approximation to $\phi(x,t)$ at the nth discrete time step. The equation can be solved using finite forward differences if one uses the up-wind scheme, proposed by Osher and Sethian [30], to compute the spatial derivatives. The update equation is

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \Delta u^n_{i,j,k},$$

where $\Delta t$ is a constant that is chosen to ensure stability and $\Delta u^n_{i,j,k}$ is the discrete approximation to $\partial \phi / \partial t$. We assume, without a loss in generality, that the grid spacing is unity. The initial conditions $u^n$ are established by the algorithm and the boundary conditions are such that the derivatives toward the outside of the grid are zero (Neumann type).

The up-wind scheme relies on one-sided derivatives:

$$\delta^+_x u^n_{i,j,k} = u^n_{i+1,j,k} - u^n_{i,j,k},$$

$$\delta^-_x u^n_{i,j,k} = u^n_{i,j,k} - u^n_{i-1,j,k},$$

$$\delta^+_y u^n_{i,j,k} = u^n_{i,j+1,k} - u^n_{i,j,k},$$

$$\delta^-_y u^n_{i,j,k} = u^n_{i,j,k} - u^n_{i,j-1,k},$$

and so forth. The partials in (17) are computed using only those derivatives that are up-wind relative to the movement of the level set. Thus, the update becomes

$$\Delta u_{i,j,k} = \gamma_B(i,j,k) \left\{ \begin{array}{ll}
\sum_{x,y,z} \min(\delta^+_x u^n_{i,j,k}, 0)^2 & \text{for } \gamma_B(i,j,k) \geq 0 \\
+ \sum_{x,y,z} \max(\delta^-_x u^n_{i,j,k}, 0)^2 & \text{for } \gamma_B(i,j,k) < 0
\end{array} \right\}^{1/2}$$

The time steps, $\Delta t$, are limited by the speed of the fastest moving wavefront, which can move only one grid unit per iteration, i.e.,

$$\Delta t \leq \frac{1}{3 \max_{i,j,k} \gamma_B(i,j,k)}.$$
Fig. 5. A 3D level-set morphing example.

Fig. 6. Interpolation of two distance volumes.
1. Initialize model volume \( v^0 \) by sampling the inside-outside function of the source.
2. Construct the volume \( v \) by sampling the inside-outside function of the target.
3. \( v_{\text{max}} = 0 \).
4. For each voxel \((i, j, k)\):
   a. Find \( v_{i,j,k} \).
   b. \( v_{\text{max}} = \text{MAX}(|v_{i,j,k}|, v_{\text{max}}) \).
   c. Calculate derivatives and the total change at \((i, j, k)\) using nearest neighbors according to the up-wind scheme given in (23).
   d. Save \( \Delta u_{i,j,k} \) in a separate volume.
5. Compute \( \Delta t \) according to (24).
6. For each voxel \((i, j, k)\):
   a. Update \( u_{i,j,k}^{n+1} \) according to (18).
   b. Compute the stopping criterion, either RMS change or RMS distance to target.
   c. If the stopping criterion is met, finish; otherwise, go to Step 4.

5.2.2 Sparse-Field Solutions

The up-wind solutions to the equations described in the previous section produce the motion of level-set models over the entire range of the embedding, i.e., for all values of \( k \) in (13). However, this method requires updating every voxel in the volume for each iteration, which means that the computation time increases as a function of the volume, rather than the surface area, of the model. Because the application of this paper, surface metamorphosis, requires only a single model, the calculation of solutions over the entire range of iso-values is an unnecessary computational burden.

The literature has shown this situation can be improved by the use of narrow-band methods, which compute solutions only in a narrow band of voxels that surround the level set of interest [1], [28]. In previous work [40], [42], we described an alternative numerical algorithm, called the sparse-field method, that computes the geometry of only a small subset of points in the range and requires a fraction of the computation time required by previous algorithms. We have shown two advantages to this method. The first is a significant improvement in computation times. The second is increased accuracy when fitting models to forcing functions that are defined to subvoxel accuracy.

The sparse-field algorithm takes advantage of the fact that a \( k \)-level surface, \( S \), of a discrete image \( u \) (of any dimension) has a set of cells through which it passes, as shown in Fig. 7. The set of grid points adjacent to the level set is called the active set and the individual elements of this set are called active points. As a first-order approximation, the distance of the level set from the center of any active point is proportional to the value of \( u \) divided by the gradient magnitude at that point. We compute the evolution given by (17) on the active set and then update the neighborhood around the active set using a fast approximation to the distance transform, which simply adds the “city-block” distance to values of the active set. Because active points must be adjacent to the level-set model, their positions lie within a fixed distance to the model. Therefore, the values of \( u \) for elements in the active set must lie within a certain range of grayscale values. When active-point values move out of this active range, they are no longer adjacent to the model. They must be removed from the set and other grid points, whose values are moving into the active range, must be added to take their place. The precise ordering and execution of these operations is important to the operation of the algorithm.

The values of the points in the active set can be updated using the up-wind scheme described in the previous section. In order to maintain stability, one must update the neighborhoods of active grid points in a way that allows grid points to enter and leave the active set without those changes in status affecting their values. Grid points should be removed from the active set when they are no longer the nearest grid point to the zero crossing. If we assume that the embedding \( u \) is a discrete approximation to the distance transform of the model, then the distance of a particular grid point, \((i, j, k)\), to the level set is given by the value of \( u \) at that grid point. If the distance between grid points is defined to be unity, then we should remove a point from the active set when the value of \( u \) at that point no longer lies in the interval \([-\frac{1}{2}, \frac{1}{2}]\). If the neighbors of that point maintain their distance of 1, then those neighbors will move into the active range just as \((i, j, k)\) is ready to be removed.

There are two operations that are significant to the evolution of the active set. First, the values of \( u \) at active points change from one iteration to the next. Second, as the values of active points pass out of the active range, they are removed from the active set and other neighboring grid points are added to the active set to take their place. Formal definitions of active sets and the operations that affect them are detailed in [42] and it is shown that active sets will always form a boundary between positive and negative regions in the image, even as control of the level set passes from one set of active points to another.

Because grid points that are near the active set are kept at a fixed value difference from the active points, active points serve to control the behavior of adjacent nonactive grid
points. The neighborhoods of the active set are defined in layers, \( L_{1}, \ldots, L_{N} \) and \( L_{-1}, \ldots, L_{-N} \), where the \( \ell \) indicates the distance (city block distance) from the nearest active grid point and negative numbers are used for the outside layers. For notational convenience, the active set is denoted \( L_{0} \). The number of layers should coincide with the size of the footprint or neighborhood used to calculate derivatives. In this way, the inside and outside grid points undergo no changes in their values that affect or distort the evolution of the zero set. The work in this paper uses only first-order derivatives of \( \phi \), which are calculated using nearest neighbors (six connected). Therefore only three layers are necessary (one inside layer, one outside layer, and the active set). These layers are denoted \( L_{1}, L_{-1}, \) and \( L_{0} \). The active set has grid point values in the range \( [-\frac{1}{2}, \frac{1}{2}] \). The values of the grid points in each neighborhood layer are kept one unit from the next layer closest to the active set, as shown in Fig. 8. Thus, the values of layer \( L_{\ell} \) fall in the interval \( [\ell - \frac{1}{2}, \ell + \frac{1}{2}] \). For \( 2N + 1 \) layers, the values of the grid points that are totally inside and outside are \( N + \frac{1}{2} \) and \( -N - \frac{1}{2} \), respectively.

This algorithm can be implemented efficiently using linked-list data structures combined with arrays to store the values of the grid points and their states, as shown in Fig. 9. This requires only those grid points whose values are changing, the active points and their neighbors, to be visited at each time step. The computation time grows as \( m^{2} \), where \( m \) is the number of grid points along one dimension of \( U \) (sometimes called the resolution of the discrete sampling). The \( m^{2} \) growth in computation time for the sparse-field models is consistent with conventional (parameterized) models for which computation times increase with surface area rather than volume.

Another advantage of the sparse-field approach is resolution. Equation (17) describes a process whereby all of the level sets of \( \phi \) are pushed toward the zero-set of \( \gamma_{H} \). The result is a shock, a discontinuity in \( \phi \). In discrete volumes, these shocks take the form of high-contrast areas which cause aliasing in the resulting models. This results in surface models that are unacceptable for many computer graphics applications and which do not resemble the target

![Diagram showing grid points and values in layers](image-url)
in the final stages of the morph (violating criterion 3 in Section 1).

When using the sparse-field method, the active points serve as a set of control points on the level set. Changing the values of these voxels changes the position of the level set. The forcing function is sampled not at the grid point, but at the location of the nearest level set, which generally lies between grid points. Using a first-order approximation to produces results that avoid the aliasing problem associated with the shocks that typically occur with level-set models. Previous work has shown significant increases in the accuracy of fitting level-set models using the first-order modification to the sparse-field method [42], which is essential to the shape metamorphosis application in this paper.

With the first-order modification, the procedure for updating the image and the active set based on surface movements is as follows:

1. For each active grid point \((i,j,k)\):
   a. Use first-order derivatives and Newton’s method to calculate the position \((i',j',k')\) of the nearest zero-crossing to \((i,j,k)\).
   b. Calculate the \(\gamma_B(i',j',k')\) using trilinear interpolation, if necessary.
   c. Compute the net change of \(u_{i,j,k}^{n+1}\) based on \(\gamma_B(i',j',k')\) and the values of its derivatives using the up-wind scheme (23).

2. For each active grid point \((i,j,k)\), add the change to the grid point value and determine if the new value \(u_{i,j,k}^{n+1}\) falls outside the \([-\frac{1}{2}, \frac{1}{2}]\) interval. If so, put \((i,j,k)\) on lists of grid points that are changing status, called the status list; \(S_1\) or \(S_{-1}\), for \(u_{i,j,k}^{n+1} > \frac{1}{2}\) or \(u_{i,j,k}^{n+1} < -\frac{1}{2}\), respectively.

3. Visit the grid points in the \(2N\) layers \(L_{\ell}\) in the order \(\ell = \pm 1, \ldots, \pm N\) and update the grid point values based on the values (by adding or subtracting one unit) of the next inner layer, \(L_{\ell+1}\). If more than one \(L_{\ell+1}\) neighbor exists, then use the neighbor that indicates a level set closest to that grid point, i.e., use the maximum for the outside layers and minimum for the inside layers. If a grid point in layer \(L_{\ell}\) has no \(L_{\ell+1}\) neighbors, then it is demoted to \(L_{\ell+1}\), the next level away from the active set.

4. For each status list \(S_{\pm 1}, S_{\pm 2}, \ldots, S_{\pm N}\):
   a. For each element \((i,j,k)\) on the status list \(S_{\ell}\), remove \((i,j,k)\) from the list \(L_{\ell+1}\) and add it to the \(L_{\ell}\) list or, in the case of \(\ell = \pm (N + 1)\), remove it from all lists.
   b. Add all \(L_{\ell+1}\) neighbors to the \(S_{\ell+1}\) list.

More details on sparse-field method and its properties can be found in [40], [42].

6 Summary of the Level-Set Metamorphosis Approach

This section describes the complete approach, based on level-set models, to 3D shape metamorphosis which meets the criteria listed in the introduction. The specific steps of our level-set morphing approach are

1. 3D scan conversion of source and target objects,
2. application of coordinate transformations,
3. level-set deformation, and
4. polygonization and rendering.

6.1 3D Scan Conversion

The essential input to the deformation stage of our morphing approach is two 3D models represented as distance volumes. A distance volume (or distance transform) is a volume data set where the value stored at each voxel is the shortest distance to the surface of the object being represented by the volume. In our examples, distance volumes are generated by scan converting CSG models, but any technique that converts a solid model into a distance volume may be used. Several methods have been developed for converting polygonal, swept, and volumetric models into distance volumes [11], [16], [20], [32], [35], [37].

We have developed a 3D scan conversion technique that produces a distance volume from a CSG model consisting of superellipsoids [3] and calculates distance to subvoxel accuracy [7], [8]. The distance volume is generated in a two step process. The first step calculates the shortest distance to the CSG model at a set of points within a narrow band around the evaluated surface. Additionally, a second set of points, labeled the zero set, which lies on the CSG model’s surface is computed. A point in the zero set is associated with each point in the narrow band. Once the narrow band and zero set are calculated, a fast marching method [36], [38] is employed to propagate the shortest distance and closest point information out to the remaining voxels in the volume.

6.2 Controlling the Morph with Coordinate Transformations

In order for our active level-set model to deform from one surface into another, the source and target objects must overlap. The objects may be automatically or interactively positioned, as well as interactively warped in order to produce a particular model alignment. The user may choose any of these methods depending upon the level of control and final output desired. The source object will shrink in those areas where it is outside the target object and will expand in those areas inside the target model. Thus, the user controls the morph by defining the regions of overlap between the source and the target. This is accomplished by applying a coordinate transformation which maps the voxel locations of the source object into new locations in the target object’s distance volume. The transformation is given by

$$x' = T(x, \alpha),$$

where $0 \leq \alpha \leq 1$ parameterizes a continuous family of transformations that begins with identity, i.e., $x = T(x, 0)$, and smoothly becomes the user-defined transformation at $T(x, 1)$. The parameterization is utilized during the polygonization stage and is explained further on in this section.

For this work, we have developed a software tool that allows a user to interactively position, rotate, and scale the source and target objects in order to produce the transformation $T$. The coordinate systems of the two objects are aligned and the user is able to manipulate the objects until they are properly overlapped. We have also developed a technique for automatically positioning, orienting, and scaling objects, using 3D moments, in order to achieve a significant correspondence between two objects without any user input. This method is detailed in the Appendix.

A generalized warping, as defined in [11], [26], may also be employed to provide even more detailed control of the process. Here, the user specifies the numerous geometric features which correspond in the source and target objects. The morph may be predominantly controlled by the geometric warp which has been interactively defined by the user and which has a number of degrees of freedom that are proportional to the number of fiducials. In this case, the level-set model would simply fine-tune the surface model as the source object is incrementally transformed by the warping algorithm into the target object. Because the goal of our work is to demonstrate a more powerful blending mechanism that performs well without extensive user input, we do not utilize such generalized warps for the morphing results presented in this paper.

6.3 Level-Set Deformation

Once the overlap of the source and target objects has been defined and any generalized warping has been applied to the source, the level-set deformation process, as described in Sections 3, 4, and 5, is initiated. The process produces a sequence of volume data sets that represent the morphing object. The user defines how often the level-set volume is written to disk during the deformation process.

6.4 Polygonization and Rendering

In order to view the morph, we extract a polygonal iso-surface (with the Marching Cubes algorithm [27]) from each volume produced by the level-set deformation process. The polygons are rendered to produce a series of images which are then combined to produce an animation. Once the level-set models have been converted into polygons, any number of conventional rendering and animation techniques may be used to shade and view the morphing object. We have developed a color shading method, based on scan-converted closest-point and color information, in order to define the colors on the resulting unparameterized polygonal models. Using this method, the color at any point in space is defined as the color at the closest point on the associated CSG model. We interpolate the color values computed from the source and target models to produce the surface colors for the intermediate shapes. The color shading method is beyond the scope of this paper and is described in [6], [8].

If a shape-changing transformation $T(x, \alpha)$ (e.g., a scaling or a generalized warp) has been utilized during the deformation process, the transformation must be interpolated and incrementally applied to the resulting polygonal models generated at each time step. Applying such a transformation implies that the morph is a combination of the user-defined transformation and the level-set deformation. The total time of the morph is scaled down to a range of $[0, 1]$, which matches the parameterization of $T(x, \alpha)$. For example, a particular morph may produce $N$ time steps, and therefore $N$ individual volume data sets. Before rendering the polygonal model produced from step $n$, the polygons should be transformed by $T(x, n/(N-1))$ before being rendered. The number $(N-1)$ results from starting the count at zero.
At the final frame, the transformation is $T(x,1) = T$, the same transformation that is used to generate $\gamma_B$. Thus, the level-set deformation ensures that the source evolves into the target, to within a coordinate transformation, $T$, which is accounted for by transforming all of the points in the polygonal model.

### 7 Results

Fig. 12 presents a morphing sequence of a dart becoming an X-29 jet. The X-29 and dart models were constructed and scan converted into distance volumes of resolution $96 \times 192 \times 240$ with The Clockworks [15], a CSG modeling system. Lerios et al. [26] demonstrate a similar transition with a jet and a dart, which required the specification of 37 different user-defined correspondence elements on both models, roughly 200 user-defined parameters. Our morph required only a few minutes of user time to interactively overlay the source and target models. The jet and dart have been rendered semitransparently in their initial configurations and presented in Fig. 10 in order to demonstrate how they were overlaid before initiating the deformation process. The jet was rendered in light transparent red and the dart was rendered in light transparent blue.
transparent blue. The areas of dark red or dark blue indicate regions where the models overlap. Fig. 13 presents the morphing sequence of Fig. 12 with interpolating surface colors generated with our color-shading algorithm [6], [8].

Fig. 14 uses the same input models with slightly different initial conditions to produce a different morph of the dart turning into the X-29. In this morph, the animator wanted the back fins of the dart to morph into the wings of the jet. This was achieved with only a few minutes of user input, the time needed to specify a scale and translation applied to the dart. The level-set deformation stage for this (and the previous) morphing sequence required approximately 9 CPU-minutes on an SGI R10000 Onyx2. Fig. 11 presents the initial conditions of the jet and dart model for the second morph. The dart has been scaled by 0.75 and translated by \([11.9, 24.0, -7.2]\) so that its fins will overlap with the wings of the jet. Before rendering each frame of the morphing sequence, each vertex \((P)\) of the polygonal model produced by the Marching Cubes algorithm is transformed by

\[
P' = (1.0 - ((n/(N - 1))(1.0 - 0.75))P + (n/(N - 1))[11.9, 24.0, -7.2].
\]

where \(n\) is the frame number and \(N\) is the total number of frames. An additional global rotation has been applied to the models in Figs. 12, 13, 14, and 17 to highlight the three-dimensional structure of the morphing model. Applying the transformation in (26) ensures that the size and location of the morphing model remains approximately constant while it is changing shape. Fig. 15 presents the morph from Fig. 14 without applying the transformation. It can be seen that, without the transformation, the morphing model changes shape and size, with the tail end of the dart growing into the rear section of the X-29. As seen in Fig. 10, the dart and X-29 initially are approximately the same size.

Fig. 16 presents the initial configuration of a mug-to-chain morph. The mug and chain were originally defined
with CSG models and then scan converted into distance volumes of resolution $120 \times 148 \times 184$. Fig. 17 presents the morphing result. The level-set deformation stage of this sequence required approximately 20 CPU-minutes on an SGI R10000 Onyx2. This sequence demonstrates that the level-sets approach easily copes with changes of topology during morphing. The results from all three morphing sequences also highlight the advantage of calculating to subvoxel accuracy. The volume resolutions are somewhat low, but the deforming surfaces extracted from them are mostly free of aliasing artifacts.

8 Conclusions and Future Work

This paper has presented a volume-based method for achieving 3D shape metamorphosis. The method relies on a novel approach to the blending stage of the morphing process. This stage is formulated as the optimization, via a hill-climbing strategy, of a similarity measure between the deforming surface and the target, utilizing level-set models for the incremental shape changes. These models take advantage of a volume-based representation to calculate surface deformations to subvoxel accuracy. The movements of these deformable models are driven by the signed distance transform of the target, which is also computed to subvoxel accuracy. The result is a 3D morphing technique that demonstrates outstanding fidelity, level of detail, flexibility, and degree of automation, comparing favorably with other methods in the literature.

As with all volumetric morphing methods, our method has its limitations. The basic volumetric representation of the objects can produce aliasing artifacts on objects that have regions of high curvature. Since a distance volume is a sampled representation, the accuracy of individual object features is restricted by the sampling resolution of the volume. Additionally, our method is only useful for solid (i.e., closed) objects. Our method cannot be used to morph open shell-like surfaces.

Even given our current results, more work remains. We will continue to work on our volume-based technique for including texture maps or surface coloring in the 3D morph. We have also developed methods for creating distance...
transforms from 3D polygonal meshes and MR/CAT-generated volume data sets. We will use these capabilities to generate morphing sequences between different types of models. Future work will also focus on better computational schemes, including parallel processing, in order to achieve 3D morphs at interactive rates.

**APPENDIX**

**AUTOMATIC OBJECT ALIGNMENT**

The user may allow the system to automatically calculate the transformation needed to align the source and target objects. While this won't guarantee that the objects will overlap and produce a reasonable morph, it does provide a way for a user to easily create an initial configuration for the morphing sequence. The alignment is accomplished by calculating two affine transformations (consisting of rotation, translation, and scaling) from gross geometric measures, that map a point in the global coordinate system of each of the objects into each of their intrinsic coordinate systems. We use the moments of the objects, calculated on point samples, to construct the transformation for each object. The centroid and principal axes of the objects define local coordinate systems for those objects, which we assume are aligned with each other.
The centroid, \( \mathbf{p} \), of an object is the average position of its internal points. The object is sampled on a regular grid within its bounding box. Each grid point is tested to determine if it is inside or outside the object. If \( V \) is the set of coordinates of internal points and \( n \) is the number of points in that set, then

\[
\mathbf{p} = \frac{1}{n} \sum_{\mathbf{p} \in V} \mathbf{p}.
\]  

The principal axes of the objects are the eigenvectors of the covariance matrix associated with each object. The covariance matrix is defined as:

\[
\mathbf{C} = \begin{pmatrix}
    c_{xx} & c_{xy} & c_{xz} \\
    c_{yx} & c_{yy} & c_{yz} \\
    c_{zx} & c_{zy} & c_{zz}
\end{pmatrix},
\]

where

Fig. 15. The morph from Fig. 14 without applying the incremental transformation.
and $i, j \in \{x, y, z\}$. The eigenvectors $(e_1, e_2, e_3)$, which are orthogonal, and eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$, which are positive and real, of the covariance matrix $C$ are calculated with a QL algorithm utilizing implicit shifts [33].

The eigenvalues provide some information about the relative size of the objects along each local axis. The square root of the eigenvalues gives a rough measure of the object’s dimensions (exact dimensions if the object is an ellipsoid). The normalized eigenvectors from the two objects are ordered and matched up according to the value of their corresponding eigenvalues. We assume that the principal axis with the greatest eigenvalue is the Z-axis, the second greatest is the Y-axis, and the principal axis with the smallest eigenvalue is the X-axis. The ordered eigenvectors may be "lined up" to construct a rotation matrix which maps a vector in the global coordinate system of the object into the local coordinate system defined by the object’s principal axes. Assuming that the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ are ordered in increasing magnitude, the associated rotation matrix is defined as:

\[
R = \begin{pmatrix}
e_1^T & e_2^T & e_3^T
\end{pmatrix}.
\]

The inverse rotation, which maps a vector in an object’s local intrinsic coordinate system back into the global coordinate system, is simply the transpose of (30),

\[
R^{-1} = \begin{pmatrix}
e_1 & e_2 & e_3
\end{pmatrix}.
\]

The scaling matrix $S$ is used to scale the source object $\Omega_A$ into the approximate size of the target object $\Omega_B$. It is defined as:

\[
S = \begin{pmatrix}
S_x & 0 & 0 \\
0 & S_y & 0 \\
0 & 0 & S_z
\end{pmatrix},
\]

where

\[
S_x = \sqrt{\frac{\lambda_1}{\lambda_A}}, \\
S_y = \sqrt{\frac{\lambda_2}{\lambda_A}}, \\
S_z = \sqrt{\frac{\lambda_3}{\lambda_A}},
\]

$\lambda_A, \lambda_B, \lambda_A^B, \lambda_B^A, \lambda_A^A, \lambda_B^B$ are the eigenvalues associated with the eigenvectors $e_1, e_2, e_3$ of object $\Omega_A$ and object $\Omega_B$.

A point $p$ in the global coordinate system of object $\Omega_A$ may be mapped into the local intrinsic coordinate system of object $\Omega_B$ ($p^B$) embedded in B’s global coordinate system by first subtracting the centroid of A, $p^A$, then applying the rotation matrix $R^A$. This places point $p$ into the intrinsic local coordinate system of A, which is assumed to be aligned with B’s. The scaling matrix $S$ is then applied so that the general dimensions of object $\Omega_A$ are approximately the same as object $\Omega_B$’s. The rotation matrix $R^B$ is applied and the point is shifted by $p^B$, the centroid of B, which maps the point into the global coordinate system of B. The transformation steps may be summarized as:

\[
p' = (p - p^A)R^A S(R^B)^{-1} + p^B.
\]

ACKNOWLEDGMENTS

The authors would like to thank Dr. Alan Barr and the other members of the Caltech Computer Graphics Group for their support and assistance. Sean Mauch developed the Fast Marching algorithm and software that was used for scan converting the models in the morphing examples. Timothy Doyle created the dart model. This work was financially supported by the US National Science Foundation (NSF)
Fig. 17. A mug morphing into a chain. This demonstrates that level set models easily cope with changes in topology.

(ASC-89-20219) as part of the STC for Computer Graphics and Scientific Visualization, the US National Institute on Drug Abuse, the US National Institute of Mental Health and the NSF, as part of the Human Brain Project; and the Volume Visualization Program of the US Office of Naval Research (N000140110033). Additional equipment grants were provided by SGI, Hewlett-Packard, IBM, and Digital Equipment Corporation. This work was initially funded by the former shareholders of the European Computer-Industry Research Centre: Bull SA, ICL PLC, and Siemens AG.

REFERENCES


Problem statement

- **Surface reconstruction**
  - Creating surfaces from volumetric data-sets
    - CT, MRI, ET scanners
    - Physical slicing and imaging
  - Non-uniform resolution
  - Important structures identified in 2D by expert anatomists and outlined with a stylus

- **Traditional surface reconstruction problems**
  - Topology changes difficult for explicit algorithms
  - Balance between smoothness and accuracy (fine details)

- **Propose to use robust and accurate 2D morphing to sweep out a smooth 3D surface**
Sweeping reconstruction

- Sweep out the surface that connects the contours
  - Use 2D level set morphing
  - Constrain the speed of the sweeping motion
  - Use a combination of Lagrangian and Eulerian techniques

![12 contours from a human model](image)

Level set method

- Level set method (Osher and Sethian 1988)
  - Robust, accurate and fast
  - Represent the interface as a time-dependent Euclidian distance function embedded in a Cartesian space that has co-dimension one
  - Arbitrary deformation problems may be recast into a framework that solves the equation:

\[
\frac{\partial \phi(x, t)}{\partial t} = \mathbf{V}(x, t) \cdot \nabla \phi(x, t) = \mathcal{F}(x, \phi(x, t), t, \ldots) \| \nabla \phi(x, t) \|
\]
The approach by Breen and Whitaker (2001), minimizes area difference. Our variational formulation:

\[
\begin{align*}
\text{Cost function:} & \quad \max(\phi) \int_{\Omega} H(\phi)H(\phi_{\text{target}})d\Omega - \int_{\Omega} H(\phi)H(-\phi_{\text{target}})d\Omega \\
\text{Formulate as} & \quad \delta(\phi) [H(\phi_{\text{target}}) - H(-\phi_{\text{target}})] = 0 \\
\text{Gradient descent} & \quad |\nabla \phi| [H(\phi_{\text{target}}) - H(-\phi_{\text{target}})] = \frac{\partial \phi}{\partial t} \\
\text{Global steady state} & \quad \frac{\partial \phi}{\partial t} = (\phi_{\text{target}}) |\nabla \phi| \\
\text{Morphing speed function} & \quad \frac{\partial \phi}{\partial t} = (\phi_{\text{target}} - \phi) |\nabla \phi|
\end{align*}
\]

Multiple morphing to sweep out a surface
- Using previous method
- Jerky movement
- Discontinuous speed when crossing contours
Novel method

- **Sweeping reconstruction**
  - Introduce time as a variable, give input a "time-of-arrival"
  - Speed is calculated by differentiating distance to target wrt time
    - Use particle tracing to find distances between contours
  - Constrain the speed of the sweeping level set

![Example of contours and swept surface](image)

Input/Preprocessing

- **Input** is manually outlined contours
- **Has assigned time-of-arrivals (height)**
- **Convert to signed distance transform**

\[ |\nabla \phi(x)| = 1, \quad x \in \mathbb{R}^n \]

- Fast sweeping, iterative PDE-based methods

![Contour](image) ![Signed distance transform](image)
Estimating distances

- *In order to get speed we need distance information*
- *Could use the signed distance transform itself but...*
  - The shortest distance is not necessarily the correct distance
  - Creates oscillations

- *Need accurate measure of distance*
  - Augment the interface with tracker particles
  - Do an intermediate morph between the current and the target to get the distance

Tracker particles

- **Seeding strategies**
  - Brute force uniform static seeding
    - A fixed number of particles per initial interface cell
  - Adaptive seeding
    - A fixed or dynamic number of particles per cell

- **Advection**
  - Opposite to the Particle level set by Fedkiw et al.
Construct speed function

- Distances and the time-of-arrivals known for particle trajectories
  - Copy onto interface grid cells
- For every interface grid cell
  - Use info to construct a 1D function of distance with respect to time
    - Differentiate to get speed

Wanted properties for speed function
- Smooth (continuous derivatives)
- Avoid over-shooting and oscillations (monotonicity)
- Use piecewise continuous polynomials (spline)

The speed function is only known for the interface grid cells (red)
- Needed throughout the narrow band

Use velocity extension (transport eqn.)

\[
\frac{\partial f(\vec{x}, t)}{\partial t} = S[\vec{x}, t] \nabla f(\vec{x}, t) \cdot \nabla \phi(\vec{x}, t)
\]

\[
S[\vec{x}, t] = \frac{\phi(\vec{x}, t)}{\sqrt{\phi(\vec{x}, t)^2 + |\nabla \phi(\vec{x}, t)|^2}}
\]

- At steady state the speed function is constant along the normal direction
Propagate the level set

Improved narrow band level set implementation (see paper)
- Time dependent PDE
- 3rd order Runge-Kutta
- Magnitude of the gradient
  - HJ upwind schemes
  - 3rd-5th order WENO

\[
\frac{\partial \phi(x, t)}{\partial t} = V(x, t) \cdot \nabla \phi(x, t)
\]

\[
= F(x, \phi(x, t), t, \ldots) |\nabla \phi(x, t)|
\]

“Boundary conditions”

The swept surface is an open “tube”

How should the open shape be terminated?
- Ill defined problem (extrapolation)

Solutions
- Flat cap
  - Simply cap the ends with a plane
- Extrapolation of the speed function (possibly with constraints) beyond boundary contours
  - Works best with terminating structures
- Manual insertion of contours
  - Can create a terminating structure
Many input contours...

- How does the method scale?

**Performance**
- All computations done in 2D, in the narrow band
- Most distances are static (max 2 traces)
- ... i.e scales in 2D with a factor depending on the number of input contours and the speed interpolation scheme

**Memory**
- Never holding the full volume in memory
- Can in principle reconstruct extremely large data-sets

- Possible to directly sweep out the mesh as well

Video

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

Ola Nilsson, David Breen, Ken Museth
Results

12 contours, $155 \times 522 \times 270$, 220 sec
Sparse input, non-uniform spacing
Boundary condition: Extra inserted flat cap

Results contd.

5 contours, $276 \times 276 \times 97$, 260 sec
Very fine detail
Boundary condition: Speed extrapolation
Results contd.

35 contours, 420 × 300 × 347, 1100 sec
Dense, complex input
Boundary condition: Flat cap

Conclusions

• Improvements
  • Performance on large data sets with many contours
    • Parallelizing and optimizing level set code
    • Particle advection/distance estimate
    • Optimize interpolation

• In summary, we proposed a method for robust surface reconstruction
  • Robust and stable with respect to topology changes
  • Accurate with respect to input constraints
  • Handles sparse input well
  • Scales in 2D, with the size of the interface
Future work

- Reconstruction of breast cancer segmentations
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.


Keywords: 3D reconstruction, contours, level sets

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-
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 [23], 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.
2.1 Novel Eulerian Approach: Level Set Model

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, \( \phi \), 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),

\[
\frac{\partial \phi(x(t), t)}{\partial t} = \frac{d \mathbf{x}(t)}{dt} \cdot \nabla \phi(x(t), t) = F(x(t), \phi(x(t), t), \ldots) \mid \nabla \phi(x(t), t),
\]

where \( d \mathbf{x}(t)/dt \) denotes the velocity vector of the deforming interface and \( F() \) is the speed function that may depend on a variety of arguments. The geometric interpretation of \( F() \) defines it as the magnitude of the velocity \( d \mathbf{x}(t)/dt \) in the direction normal to the interface at \( x, i.e. \ F \equiv n \cdot d \mathbf{x}(t)/dt. \) Also note that the local interface normal, \( n \), is given by \( \nabla \phi / |\nabla \phi| \) and that the mean curvature is \( \nabla \cdot n, \) 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, \( \phi(x, 0) = \phi_{source}(x) \), to a final level set, \( \phi_{target}(x) \), can be formulated and solved with the following PDE,

\[
\frac{\partial \phi(x(t), t)}{\partial t} = [\phi(x(t), t) - \phi_{target}(x)] |\nabla \phi(x(t), t)|,
\]

which is evolved to a steady-state where \( \phi \) and \( \phi_{target} \) 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 \( \Phi_{\text{target}} \), and requires that \( \Phi_{\text{source}} \) and \( \Phi_{\text{target}} \) overlap; otherwise \( \Phi_{\text{source}} \) will collapse to a point. Note that the presence of \( \phi \) in the speed function, \( \mathcal{F} = \phi - \Phi_{\text{target}} \), 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 \( \mathcal{F} = -\Phi_{\text{target}} \), 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).](image)

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 \( \phi_A \) to \( \phi_B \) 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 \( \phi_A \).
2. **Adveect** the particles in the following vector field:
   \[
   (\phi_B(x) - \phi_A(x)) \nabla \phi_A(x) / |\nabla \phi_A(x)|
   \]
3. **Propagate** \( \phi_A(x) \) with \( \mathcal{F} = \phi_B(x) - \phi_A(x) \).
4. **Back-project** particles into A using the vector field:
   \[
   -\phi_A(x) \nabla \phi_A(x) / |\nabla \phi_A(x)|
   \]
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 \( \mathcal{F}(\cdot) \), the fact that the local normal field of \( \phi_A \) is \( \nabla \phi_A(x) / |\nabla \phi_A(x)| \) 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 \( A \rightarrow B \rightarrow \ldots \rightarrow N \) and a particular grid point on the zero-crossing of the current level set. Using the particle tracking 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.
However, the speed-function of Eq. (1) must be defined in the full narrow band of \( \phi \). Therefore, to extend \( \mathcal{F}() \) off of the interface we solve the following transport equation [33, 39]:

\[
\frac{\partial \mathcal{F}(x, t)}{\partial t} = S[\phi(x, t)] \nabla \mathcal{F}(x, t) \cdot \nabla \phi(x(t), t)
\]  

(3)

where

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

(4)

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

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:

\[
\frac{\partial}{\partial t} |\nabla \phi|^2 = 2 \nabla \phi \cdot \nabla \frac{\partial \phi}{\partial t} = 2 \nabla \phi \cdot (\nabla \mathcal{F} |\nabla \phi| + \mathcal{F} \nabla |\nabla \phi|) = 0
\]  

(5)

which makes use of the fact that \( \phi \) is initialized as a Euclidian distance function (i.e. \( \nabla \phi = 1 \Rightarrow \nabla |\nabla \phi| = 0 \)), and \( \nabla \phi \cdot \nabla \mathcal{F} = 0 \), 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 \( \phi \) to a Euclidian distance function in order to ensure numerical stability of the morph. For this renormalization we solve

\[
\frac{\partial \phi(x, t)}{\partial t} = S[\phi(x, t)] (|\nabla \phi(x, t)| - 1)
\]  

(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.

3 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 careful construction of 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.

```plaintext
int dim = 1, X[m], Y[m], Mask[m][m];
foreach pixel (i, j) do
    if |φ(i, j)| < γ then
        Mask[i][j] = 0; /* outside both tubes */
        if |φ(i, j)| = 2 then /* 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, ..., 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]] ≥ 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 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 $(i, j)$ ∈ $N_{old}$ do
    if $|\phi(i, j)| < \gamma$ then
        add $(i, j)$ to $T_{new}$ and $N_{new}$;
    else
        foreach $(i, j)$'s neighbors $(p, q)$ ∈ $N_{new}$ do
            if $|\phi(p, q)| < \gamma$ then add $(i, j)$ to $N_{new}$;
        end
    end
end

foreach $(i, j)$ ∈ $T_{old}$ but $(i, j)$ ∈ $T_{new}$ then
    foreach $(i, j)$'s neighbors $(p, q)$ ∈ $N_{new}$ do
        add $(p, q)$ to $N_{new}$
    end
end
```

References


**Level Set Segmentation of Biological Volume Datasets**

David Breen  
Ken Museth  

Ross Whitaker  
Leonid Zhukov

**Department of Computer Science**  
**Drexel University**

**School of Computing**  
**University of Utah**

**Department of Science and Technology**  
**Linköping University**

**Department of Computer Science**  
**California Institute of Technology**

---

**Two Topics**

- Framework for Level Set Segmentation of Volume Datasets
- Level Set Segmentation of Diffusion Tensor MRI Brain Data
A Framework for Level Set Segmentation of Volume Datasets

LS Framework Conclusion

- Level set methods can be used to solve real problems.
  - Segmentation of a variety of noisy biological volume datasets
The Main Point

• Level set methods are not sufficient by themselves for segmentation
• For successful segmentations
  – They should be accompanied by powerful initialization techniques
  – Combined in a flexible framework
• General approach
  – Fit to (noisy, incomplete) data
  – Apply smoothing
  – Find the right balance between the two

LS Segmentation Framework

• Extract closed, smooth models from volumes
• Rough estimate + Fine tuning (sub-voxel)
• Initialization + Level Set Deformation
• Initial conditions + Evolve Level Set PDE
**Computational Initialization**

- Gaussian low-pass filter
  - Blurs data and removes noise
- Voxel classification
  - Low and high thresholding
- Topological/logical operations
  - Flood fill (image & gradient magnitude values)
- Morphological filtering
  - Opening and closing

**Interactive Initialization**

- Interactively fit primitives to input data
- Scan convert CSG model into initial volume
Level Set Deformation (1)

- Balance between fitting to (noisy) data and curvature-based smoothing
- \( F(X) = \alpha \cdot I(X) + (1-\alpha) \cdot H(X) \)
- \( I \)- image-based measures
- \( H \)- curvature
- \( \alpha \)- user-defined parameter

Level Set Deformation (2)

- Define components of velocity term \( F(\cdot) \)
- Smooth surface based on mean curvature \( H \)

\[
\tilde{F}(\tilde{X}) = H\tilde{N}(\tilde{X}) = \left( \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right) \nabla \phi
\]

- Smooths regions of high curvature
Level Set Deformation (3)

Image-based components

- Given distance to Canny edges $D_c(\vec{x})$, define velocity term

$$\vec{F}(\vec{x}) = \nabla D_c(\vec{x})$$

- Pulls level set from a far distance toward Canny edges in data

Level Set Deformation (4)

- Local greyscale features
  - Gradient of gradient magnitude
    $$\vec{F}(\vec{x}) = \pm \nabla |\nabla I(\vec{x})|$$
  - With smoothing, can move level set to edges with subvoxel accuracy

- Distance to isosurface $\sim \frac{(I(\vec{x}) - k)}{|\nabla I(\vec{x})|}$

- Smooth distance

$$\vec{F}(\vec{x}) = \frac{\nabla \phi}{|\nabla \phi|} g \left( \frac{I(\vec{x}) - k}{|\nabla I(\vec{x})|} \right)$$
Results - MRI Head

- Goal - single, smooth outer surface

Results - MRI Head

- Initialization
  - Gaussian blur
  - Threshold
  - Flood fill
  - Closing
Results - MRI Head

Fit to isosurface with smoothing

Results - MRI Mouse Liver

- Goal - extract structures from an MRI scan of a mouse embryo
- Liver especially difficult
  - surrounded by brighter and darker regions
- Gaussian blur, threshold, flood fill, opening followed by closing
- Move to Canny edges
- Fit to gradient magnitude
Results - MRI Mouse Liver

Liver

Results - MRI Mouse Liver

initialization  edge fitting  gradient mag
Mouse Embryo Segmentation

- Ventricles
- Eye
- Liver

Results - ET Dendrite

- Goal - extract spiny dendrite structure from an electron tomogram
Results - ET Dendrite

- Initialization
  - Gaussian blur
  - Threshold
  - Flood fill
  - Opening followed by closing
  - Move to Canny edges
- Fit to gradient magnitude with smoothing

Results - ET Dendrite

Initialization

LS Deformation
Results - ET Dendrite

- LS
- Manual

8/20/07

Results - MRI Frog Embryo

- Interactive initialization
- Gradient magnitude
- Smoothing

8/20/07
LS Framework Summary

- Level set methods can be used to successfully segment noisy volume datasets
- Level set methods (iso-surface deformation) are not sufficient by themselves
- For successful segmentations
  - They should be accompanied by powerful initialization techniques
  - Combined in a flexible framework

Software Available

- Many features implemented as modules in Iris Explorer (NAG) on SGI
- VISPACK
  - www.cs.utah.edu/~whitaker/vispack
- ITK
  - www.itk.org
Level Set Segmentation of Diffusion Tensor MRI Brain Data

Goal

• Extract 3D structures from diffusion tensor MRI volume datasets
• Regions with similar diffusion properties
• Visualization
• Modeling and analysis
  – Surface area
  – Curvature
  – Volume
MRI

- T1, T2, spin density
- Intensity image
- Voxel - scalars

Diffusion Tensor MRI

- Voxel - rank 2 tensor

- Tensor values:
  
  | 0.359 | 0.023 | 0.076 |
  | 0.023 | 0.432 | 0.354 |
  | 0.076 | 0.354 | 0.251 |
Diffusion Tensor MRI

- “raw data”
- $3 \times 3$ matrix

\[
\begin{bmatrix}
0.359 & 0.023 & 0.076 \\
0.023 & 0.432 & 0.354 \\
0.076 & 0.354 & 0.251
\end{bmatrix}
\]

Diffusion Tensor

- Diffusion equation
  \[
  \frac{\partial C}{\partial t} = \nabla \cdot (D \nabla C)
  \]
- Isotropic/anisotropic
  \[
  D = \begin{pmatrix}
  D_{xx} & D_{xy} & D_{xz} \\
  D_{yx} & D_{yy} & D_{yz} \\
  D_{zx} & D_{zy} & D_{zz}
  \end{pmatrix}
  \]
- $D$ dependent on orientation
How to Visualize?

- Direct volume rendering (Kindlmann)
- Streamlines based on diffusion properties (Zhukov)
- Probability ellipsoids (Laidlaw)
- Calculate scalar diffusion invariants
  - Create “invariant” volumes
  - Visualize and segment invariants

Invariants

Diffusivity

Anisotropy

DT MRI
**Eigenvalues/vectors**

- Eigenvalues/eigenvectors basis
  \[ D \mathbf{e} = \lambda \mathbf{e} \quad (D - \lambda I) \mathbf{e} = 0 \]
- Eigenvalue computations
  \[ \det (D - \lambda I) = 0 \]
  \[ \lambda^3 - \lambda_1 \cdot \lambda^2 + \lambda_2 \cdot \lambda - \lambda_3 = 0 \]
- Eigenvalue/eigenvector diagonalization, LAPACK

**Diffusion Ellipsoids**

- Isotropic
  \[ \lambda_1 \approx \lambda_2 \approx \lambda_3 \]
- Anisotropic
  \[ \lambda_1 \gg \lambda_2 \gg \lambda_3 \]
  \[ \lambda_1 \gg \lambda_2 \approx \lambda_3 \]
Spinal Cord Diffusion Tensor Imaging: Laidlaw et al.

Original Measures

- Mean diffusivity:
  \[ \lambda_m = \frac{1}{3} (\lambda_1 + \lambda_2 + \lambda_3) \]

- Fractional anisotropy:
  \[ f' \lambda = \frac{\sqrt{3}}{\sqrt{2}} \sqrt[4]{\frac{(\lambda_1 - \lambda_m)^2 + (\lambda_2 - \lambda_m)^2 + (\lambda_3 - \lambda_m)^2}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \]

  Basser 96, Westin 97
Invariants

\[ \lambda^3 - I_1 \cdot \lambda^2 + I_2 \cdot \lambda - I_3 = 0 \]

\[
I_1 = D_{xx} + D_{yy} + D_{zz} \\
I_2 = \begin{vmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{vmatrix} + \begin{vmatrix} D_{xx} & D_{xz} \\ D_{zx} & D_{zz} \end{vmatrix} + \begin{vmatrix} D_{xq} & D_{yq} \\ D_{zq} & D_{zz} \end{vmatrix} \\
I_3 = \begin{vmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{vmatrix} \]

New Invariants

• Instead of using eigenvalues, use coefficients of characteristic eq. \((I_1, I_2, I_3)\)
• Less computation
• Rotationally invariant
• Combinations of coefficients are invariant
• Less sensitive to noise
In eigen-frame-of-reference

- \( I_1 = \lambda_1 + \lambda_2 + \lambda_3 \)
- \( I_2 = \lambda_1 \lambda_2 + \lambda_1 \lambda_3 + \lambda_2 \lambda_3 \)
- \( I_3 = \lambda_1 \lambda_2 \lambda_3 \)
- Proportional to
  - sum of radii
  - surface area
  - volume
  of “diffusion” ellipsoid

New Invariants

- Diffusivity:
  \[ D_i = \frac{1}{3} I_1 \Rightarrow \frac{1}{3} (\lambda_1 + \lambda_2 + \lambda_3) \]

- Anisotropy:
  \[ D_{\alpha} = \frac{1}{6} \left[ \frac{I_1 I_2}{I_3} - 3 \right] \Rightarrow \frac{1}{3} \left[ \frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} \right] \]
New Invariants

Diffusivity - $D_i$

$\text{trace (T)} = \text{trace (}$

$\text{)} =$

Invariants – 2D slice

$\lambda_{i\text{ii}}$  $\text{FA}$  $D_i$  $D_{\alpha}$
**Isotropic**

- Isosurface
- Initialization
- LS Deformation

**Level Set Segmentation**

- Isosurface
- Initialization
- LS Deformation
Corpus Callosum

Same Dataset - Different Models
Exploded View

Different Dataset
Same Parameters

Isosurface  Initialization  LS Deformation
Acknowledgments

• Thanks to
  – Alan Barr, Jason Wood, Mark Ellisman, Maryanne Martone, Russ Jacobs, Cyrus Papan, Eric Ahrens, Chris Johnson, David Weinstein, Gordon Kindlmann, Mike Tyszka, Miriam Scadeng, John Wood, Cici Koenig

• Datasets provided by
  – Caltech Biological Imaging Center, Utah SCI Institute, NCMIR, Children’s Hospital Los Angeles, University of Edinburgh

• Supported by
  – NSF, NIH, ONR, NLM, Caltech

Related Papers


A Framework for Level Set Segmentation of Volume Datasets

Ross Whitaker
School of Computing
University of Utah

David Breen, Ken Museth, and Neha Soni
Computer Science Department
California Institute of Technology

Abstract

This paper presents a framework for extracting surface models from a broad variety of volumetric datasets. These datasets are produced from standard 3D imaging devices, and are all noisy samplings of complex biological structures with boundaries that have low and often varying contrasts. The level set segmentation method, which is well documented in the literature, creates a new volume from the input data by solving an initial-value partial differential equation (PDE) with user-defined feature-extracting terms. Given the local/global nature of these terms, proper initialization of the level set algorithm is extremely important. Thus, level set deformations alone are not sufficient, they must be combined with powerful initialization techniques in order to produce successful segmentations. Our level set segmentation approach consists of defining a set of suitable pre-processing techniques for initialization and selecting/tuning different feature-extracting terms in the level set algorithm. This collection of techniques forms a toolkit that can be applied, under the guidance of a user, to segment a variety of volumetric data. Users can combine these methods in different ways and thereby access a wide range of functionalities, several of which are described in this paper and demonstrated on noisy volume data.

1 Introduction

As visualization tasks grow in size and complexity, the problem of presenting data effectively is accompanied by another, potentially more difficult problem—how to extract presentable data from the flood of raw information produced by large simulations and high resolution instruments. Thus, the field of data visualization is intimately tied to more traditional studies of data analysis such as signal and image processing, pattern recognition, artificial intelligence, and computer vision. However, in contrast to more conventional areas of data analysis, the field of visualization explicitly includes the user in the process of filtering, extracting, and rendering meaningful data.

This paper deals with a specific visualization problem—that is, how to build meaningful 3D models of complex structures from noisy datasets generated from 3D imaging devices. In certain circumstances such data can be visualized directly [1, 2, 3, 4]. While direct techniques can provide useful insights into volume data, they are insufficient for many problems. For instance, direct volume rendering techniques typically do not remove occluding structures, i.e., they do not allow one to “peel back” the various layers of the data to expose the inner structures that might be of interest. They also do not generate the models needed for quantitative study/analysis of the visualized structures. Furthermore, direct visualization techniques typically do not perform well when applied directly to noisy data, unless one filters the data first. Techniques for filtering noisy data are abundant in the literature, but there is a fundamental limitation—filtering that reduces noise tends to distort the shapes of the objects in the data. The challenge is to find methods which present the best tradeoff between fidelity and noise.

Level set segmentation relies on a surface-fitting strategy, which is effective for dealing with both small-scale noise and smoother intensity fluctuations in volume data. The level set segmentation method, which is well documented in the literature [5, 6, 7, 8], creates a new volume from the input data by solving an initial-value partial differential equation (PDE) with user-defined feature-extracting terms. Given the local/global nature of these terms, proper initialization of the level set algorithm is extremely important. Thus, level set deformations alone are not sufficient, they must be combined with powerful initialization techniques in order to produce successful segmentations. Our level set segmentation approach consists of defining a set of suitable pre-processing techniques for initialization and selecting/tuning different feature-extracting terms in the level set algorithm. We demonstrate that combining several pre-processing steps with level set deformations produces a powerful toolkit that can be applied, under the guidance of a user, to segment a wide variety of volumetric data.

There are more sophisticated strategies for isolating meaningful 3D structures in volume data. Indeed, the so called segmentation problem constitutes a significant fraction of the literature in image processing, computer vision, and medical image analysis. For instance, statistical approaches [9, 10, 11, 12] typically attempt to identify tissue types, voxel by voxel, using a collection of measurements at each voxel. Such strategies are best suited to problems where the data is inherently multi-valued or where there is sufficient prior knowledge [13] about the shape or intensity characteristics of the relevant anatomy. Alternatively, anatomical structures can be isolated by grouping voxels based on local image properties. Traditionally, image processing has relied on collections of edges, i.e. high-contrast boundaries, to distinguish regions of different types [14, 15, 16]. Furthermore deformable models, incorporating different degrees of domain-specific knowledge, can be fitted to the 3D input data [17, 18]. The work of this paper demonstrates a mathematical and computational framework which effectively combines or unifies classification, filtering, and surface-fitting approaches to modeling and visualizing 3D data.

2 Example Datasets

Our work is largely motivated by the desire to produce a semi-automatic segmentation approach which can partly or fully replace the tedious and extremely time-consuming process of manual data segmentation—a solution which to our initial surprise is widely used by colleagues in biology and medicine. Thus, to scientists working in these fields even an approximate scheme which can segment out approximately 90% of the model is immensely useful because it reduces the manual labor needed to produce a final result. We stress that there exists no fully automatic solution to the segmentation problem typically encounter in 3D imaging. For example, Figure 1(a) shows one of 270 slices of an electron tomography (ET) volume of a spiny dendrite provided by the National Center for Microscopy and Imaging Research, at UC San Diego. The complex structure of the dendrite and the noisy nature of the data make the rendering of such volume data difficult. Figure 1(b) shows the results of attempting to isolate the relevant structures in this dataset by extracting isosurfaces at greyscale value of 129. For this example we have blurred the data with a small Gaussian ker-
Figure 1: a) One slice of a 154 × 586 × 270 ET scan of a spiny dendrite shows low contrast and high noise content in a relatively complex data set. b) An isosurface rendering, with prefiltering, shows how noise and inhomogeneities in density interfere with visualizing the 3D structure of the dendrite.

Figure 2: a) One slice of a 130 × 128 × 128 magnetic resonance (MR) volume of a human head shows high-contrast, relatively noise-free data with numerous internal structures. b) An isosurface rendering, with a small wedge removed for visualization, shows aliasing and internal structures that are not appropriate for the application.

Channel (σ = 1.0) to try to improve the appearance of the isosurfaces. Despite the smoothing the isosurfaces are quite noisy, and contain many small, disconnected pieces that are not indicative of the structure of the dendrite. Furthermore, fluctuations in the tissue density both within and outside of the dendrite create a large number of distortions which prevent the isosurface from accurately representing the underlying shape of that structure.

Note that the image shown in Figure 1(b) is produced in two stages: First, we compute the isosurface with the Marching Cubes algorithm [19] for a given isovalue. Next, the polygonal mesh is displayed using conventional graphics hardware. Alternatively we could visualize structures within the volume data using a one-stage direct method such as volume-rendering (e.g. ray casting with transfer functions or maximum intensity projection). Our choice of Marching Cubes for rendering isosurfaces of this and other datasets in this paper is not essential to the proposed method. The problems of noise and aliasing, present in the examples in this paper, would exist even if we used a direct volume rendering technique.

A second example, shown in Figure 2(a), is a magnetic resonance (MR) scan of a human head. Here the problem is not so much the quality of the data—isosurfaces can be used to visual-

Figure 3: a) One slice of a 130 × 128 × 128 magnetic resonance (MR) volume of a human head shows high-contrast, relatively noise-free data with numerous internal structures. b) An isosurface rendering, with a small wedge removed for visualization, shows aliasing and internal structures that are not appropriate for the application.

Figure 4: a) One slice of a 44 × 45 × 43 MR scan of a frog embryo. b) A Marching Cubes isosurface from the frog embryo volume. Iso-value = 60.

Figure 5: a) Level set models represent curves and surfaces implicitly using greyscale images. For example an ellipse is represented as the level set of an image shown here. b) To change the shape of the ellipse we modify the greyscale values of the image by solving a certain PDE.
ize the skin or skull. In this case the particular application [20] requires a relatively smooth, simple, closed surface, and will not tolerate significant aliasing. The application also requires that the fairly complicated structure of the inner head (usually unseen) be removed. Figure 2(b) shows a isosurface rendering, at a greyscale value of 30, which demonstrates the aliasing in the data. A small wedge has been removed to show the complex internal structures in this volume.

The third example, shown in Figure 3(a), is a $256 \times 128 \times 128$ MR scan of a 12-day-old mouse embryo. Colleagues in the Caltech Biological Imaging Center (BIC) are using such images to develop a detailed atlas for the gestational development of these organisms. For this paper we will consider the specific task of isolating the liver, which is the dark, kidney-shaped area on the right. The liver, however, is not a single grey-scale value, and it is bordered by both more dense and less dense regions. Furthermore, the data contains noise. Therefore, the liver is not easily isolated by simple greyscale classification or isovalue schemes. Figure 3(b) shows an isosurface rendering which accommodates high and low thresholds associated with the liver, i.e., the zero crossings of $I = \min(I - t_{hi}, t_{lo} - I)$, where $I$ is the input volume, and $t_{hi}$/$t_{lo}$ are the thresholds. The model constructed from the isosurface shows significant artifacts from noise and low-frequency fluctuations in the tissue. It also shows artifacts from the greyscale classification, which captures a large number of voxels in the transition between the skin and the surrounding regions. Smoothing further aggravates this problem.

The final example, shown in Figure 4(a), is a $44 \times 45 \times 43$ in vivo MR scan of a frog embryo. This is one slice from one scan of a sequence of 22 volumes taken over a 24-hour period. Colleagues at the Caltech BIC are acquiring time-lapsed MR volume sequences in order to generate the first 3D staging sequence of a developing frog embryo. They require models of the dynamic structures that appear, move, change shape, merge and/or disappear over time within the embryo, as well as the outside shell. The individual structures do not necessarily have distinct signals in the MR scans, thus making it difficult to computationally isolate them. Figure 4(b) presents a polygonal isosurface (isovalue = 60) generated with the Marching Cubes algorithm [19]. At this isovalue two internal structures are produced, as well as a significant part of the outer shell.

### 3 Level Set Surface Models

When considering deformable models for segmenting 3D volume data, one is faced with a choice from a variety of surface representations, including triangle meshes [20, 21], superquadrics [22, 23, 24], and many others [18, 25, 26, 27, 28, 29, 30]. Another option is an implicit level set model, i.e., specifying the surface as a level set of a scalar volumetric function, $\phi: U \mapsto \mathbb{R}$, where $U \subset \mathbb{R}^3$ is the range of the surface model. Thus, a surface $S$ is

$$S = \{s|\phi(s) = k\}, \quad (1)$$

and the choice of the isovalue, $k$, is arbitrary. In other words, $S$ is the set of points $s$ in $\mathbb{R}^3$ that composes the $k$ isosurface of $\phi$. The embedding $\phi$ can be specified as a regular sampling on a rectilinear grid.

Our overall scheme for segmentation is largely based on the ideas of Osher and Sethian [31] that model propagating surfaces with (time-varying) curvature-dependent speeds. The surfaces are viewed as a specific level set of a higher-dimensional function $\phi$ – hence the name level set methods. These methods provide the mathematical and numerical mechanisms for computing surface deformations as isosurfaces of $\phi$ by solving a partial differential equation on the 3D grid. That is, the level set formulation provides a set of numerical methods that describes how to manipulate the greyscale values in a volume, so that the isosurfaces of $\phi$ move in a prescribed manner (shown in Figure 5). This paper does not present a comprehensive review of level set methods, but merely introduces the basic concepts and the notation used in successive sections. See [7] for more details.

There are two different approaches to defining deformable surface from a level set of a volumetric function as described in Equation 1. Either one can think of $\phi(s)$ as a static function and change the isovalue $k(t)$ or alternatively fix $k$ and let the volumetric function dynamically change in time, i.e. $\phi(s, t)$. Thus, we can mathematically express the static and dynamic model respectively as

$$\phi(s) = k(t) \quad (2a)$$

or

$$\phi(s, t) = k. \quad (2b)$$

To transform these definitions into partial differential equations which can easily be solved by standard numerical techniques, we differentiate both sides of Equation 2 with respect to time $t$, and apply the chain rule:

$$\nabla \phi(s) \cdot \frac{ds}{dt} - \frac{dk(t)}{dt} \quad (3a)$$

$$\frac{\partial \phi(s, t)}{\partial t} + \nabla \phi(s, t) \cdot \frac{ds}{dt} = 0 \quad (3b)$$

The static Equation 3a is often referred to as the “Eikonal” equation and defines a boundary value problem for the time-independent volumetric function $\phi$. This static level set approach has been solved [32, 33] using a “Fast Marching Method”. However, it has some inherent limitations following the simple definition in Equation 2a. Because $\phi$ is a function (i.e. single-valued), isosurfaces cannot self intersect over time, i.e. shapes defined in the static model are strictly expanding or contracting over time. The dynamic level set approach of Equation 3b is much more flexible and shall serve as the basis of the segmentation scheme in this paper. Equation 3b is sometimes referred to as a “Hamilton-Jacobi-type” equation and defines an initial-value problem for the time-dependent $\phi$. Throughout the remainder of this paper we shall for simplicity refer to this dynamical approach as the level set method – and completely ignore the static alternative.

Thus, to summarize the essence of the (dynamic) level set approach, let $ds/dt$ be the movement of a point on a surface as it deforms, such that it can be expressed in terms of the position of $s \in U$ and the geometry of the surface at that point, which is, in turn, a differential expression of the implicit function, $\phi$. This gives a partial differential equation on $\phi$: $s \equiv s(t)$

$$\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{ds}{dt} \equiv -\nabla \phi \cdot F(s, D\phi, D^2\phi, \ldots), \quad (4)$$

where $F$ is a user-defined “speed” term which depends on a set of order-$n$ derivatives of $\phi$, $D^n\phi$, evaluated at $s$, as well as other functions of $s$. Because this relationship applies to every level set of $\phi$, i.e. all values of $k$; this equation can be applied to all of $U$, and therefore the movements of all the level set surfaces embedded in $\phi$ can be calculated from Equation 4. Such level set methods are well documented in the literature for applications such as computational physics [34], image processing [35, 36], computer vision [6, 37], medical image analysis [6, 38], shape morphing[39], and 3D reconstruction [40].

The level set representation has a number of practical and theoretical advantages over conventional surface models, especially in the context of deformation and segmentation. First, level set models are topologically flexible, they easily represent complicated surface shapes that can, form holes, split to form multiple objects, or merge with other objects to form a single structure. These models can incorporate many (millions) of degrees of freedom, and therefore they can accommodate complex shapes such as the dendrite in Figure 1.
Figure 6: Level set segmentation stages – initialization and surface deformation.

Indeed, the shapes formed by the level sets of $\phi$ are restricted only by the resolution of the sampling. Thus, there is no need to parameterize the model as it undergoes significant changes in shape.

Level set methods have been shown to be effective in extracting surface structures from biological and medical data. For instance Malladi et al. [6] propose a method in which the level sets form an expanding or contracting contour which tends to “cling” to interesting features in 2D angiograms. At the same time the contour is also influenced by its own curvature, and therefore remains smooth. Whitaker has shown [38, 41, 42] that level sets can be used to simulate conventional deformable surface models, and demonstrated this by extracting skin and tumors from thick-sliced (e.g. clinical) MR data, and by reconstructing a fetal face from 3D ultrasound. A variety of authors [36, 43, 44] have presented variations on the method with results for 2D and 3D data. Sethian [7] gives several examples of level set curves and surface for segmenting CT and MR data.

The purpose of this paper is to present a collection of initialization and level set mechanisms which form a “toolbox” for volume dataset segmentation. We also show how these methods can be combined to solve the problems presented in Figures 1–4. These tools provide a set of techniques that are not as direct as simple thresholding or volume rendering but are more powerful than the “hand-contouring” that is currently the state-of-the-art in many applications, such as the dendrite example in Figure 1.

4 Segmentation Stages

Our level set segmentation process has two major stages, initialization and level set surface deformation, as seen in Figure 6. Each stage is equally important for generating a correct segmentation. Within our framework a variety of operations are available in each stage. A user must “mix-and-match” these operations in order to produce the desired result.

4.1 Initialization

Because the deformable models move using gradient descent, they seek local solutions, and therefore the results are strongly dependent on the initialization, i.e., the starting position of the surface. Thus, one controls the nature of the solution by specifying an initial model from which the surface deformation process proceeds. We have implemented both computational (i.e. “semi-automated”) and manual/interactive initialization schemes; each offers distinct advantages in different situations.

4.1.1 Computational Initialization

Because the level set modeling technology is based on the isosurfaces of volumes, we can, for many different types of problems, computationally construct reasonable initial estimates directly from the input data. We do this by combining a variety of techniques.

Linear filtering: We can filter the input data with a low-pass filter (e.g. Gaussian kernel) to blur the data and thereby reduce noise. This tends to distort shapes, but the initialization need only be approximate.

Voxel classification: We can classify pixels based on the filtered values of the input data. For grayscale images, such as those used in this paper, the classification is equivalent to high and low thresholding operations. These operations are usually accurate to only voxel resolution (see [12] for alternatives), but the deformation process will achieve sub-voxel results.

Topological/logical operations: This is the set of basic voxel operations that takes into account position and connectivity. It includes unions or intersections of voxel sets to create better initializations. These logical operations can also incorporate user-defined primitives. Topological operations consist of connected-component analyses (e.g. flood fill) to remove small pieces or holes from objects.

Morphological filtering: This includes binary and grayscale morphological operators on the initial voxel set. For the results in the paper we implement openings and closings using morphological propagators [45, 46] implemented with level set surface models. This involves defining offset surfaces of $\phi$ by expanding/contracting a surface according to the following PDE,

$$\frac{\partial \phi}{\partial t} = \pm |\nabla \phi|,$$  \hspace{1cm} (5)

up to a certain time $t$. The value of $t$ controls the offset distance from the original surface of $\phi$ ($t = 0$). A dilation of size $\alpha$, $D_\alpha$, corresponds to the solution of Equation 5 at $t = \alpha$ using the positive sign, and likewise erosion, $E_\alpha$, uses the negative sign. One can now define a morphological opening operator $O_\alpha$ by first applying an erosion followed by a dilation of $\phi$, i.e. $O_\alpha \phi = D_\alpha \circ E_\alpha \phi$, which removes small pieces or thin appendages. A closing is defined as $C_\alpha \phi = E_\alpha \circ D_\alpha \phi$, and closes small gaps or holes within objects. Both operations have the qualitative effect of low-pass filtering the isosurfaces in $\phi$—an opening by removing material and a closing by adding material. Both operations tend to distort the shapes of the surfaces on which they operate, which is acceptable for the initialization because it will be followed by a surface deformation.

4.1.2 Interactive Initialization

Computational initialization may not always produce a reasonable starting model that deforms into an acceptable final result. Such is the case with the frog-embryo data shown in Figure 4. For volumes that do not allow one to automatically generate an initial model, it...
is desirable and easier for the user to interactively specify the initial model which is then deformed to fit to the input data. The interactive initialization process has four steps and is presented in Figures 4(b), and 7(a-c). First, the user generates a Marching Cubes mesh from the input volume. This gives some indication of the structures present in the data (Figure 4(b)). The user then creates a Constructive Solid Geometry (CSG) model which defines the shape of the initial surface. The CSG model in blue is interactively positioned relative to the Marching Cubes mesh (Figure 7(a)). The CSG model is scan-converted into a binary volume, with voxels simply marked as inside (1) or outside (0), using standard CSG evaluation techniques [47] within our modeling system [48]. An isosurface of the initialization volume dataset generated from the torus and sphere in Figure 7(a) is presented in Figure 7(b), isovalue = 0.5. This volume dataset is then used as the starting model for the level set deformation stage, which produces the final result seen in Figure 7(c).

4.2 Level Set Surface Deformation

The initialization should position the model near the desired solution while retaining certain properties such as smoothness, connectivity, etc. Given a rough initial estimate, the surface deformation process moves the surface model toward specific features in the data. One must choose those properties of the input data to which the model will be attracted and what role the shape of the model will have in the deformation process. Typically, the deformation process combines a data term with a smoothing term, which prevents the solution from fitting too closely to noise-corrupted data. There are a variety of surface-motion terms that can be used in succession or simultaneously, in a linear combination to form \( F(x) \) in Equation 4.

**Curvature:** This is the smoothing term. For the work presented here we use the mean curvature of the isosurface \( H \) to form a vector in the direction of the surface normal \( n \) given by

\[
F(x) = Hn = \left( \nabla \phi, \frac{\nabla \phi}{|\nabla \phi|} \right) \frac{\nabla \phi}{|\nabla \phi|}. \tag{6}
\]

The mean curvature is also the normal variation of the surface area (i.e., minimal surface area). There are a variety of options for second-order smoothing terms [41], and the question of efficient, effective higher-order smoothing terms is the subject of on-going research [7]. For the work in this paper, we combine mean curvature with one of the following three terms, weighting it by a factor \( \beta \), which is tuned to each specific application.

**Edges:** Conventional edge detectors from the image processing literature produce sets of “edge” voxels that are associated with areas of high contrast. For this work we use a gradient magnitude threshold combined with non-maximal suppression, which is a 3D generalization of the method of Canny [16]. The edge operator typically requires a scale parameter and a gradient threshold. For the scale, we use small, Gaussian kernels with standard deviation \( \sigma = [0.5 - 1.0] \) voxel units. The threshold depends on the contrast of the volume. The distance transform on this edge map produces a volume that has minima at those edges. The gradient of this volume produces a field that attracts the model to those edges. The edges are limited to voxel resolution because of the mechanism by which they are detected. Although this fitting is not sub-voxel accurate, it has the advantage that it can pull models toward edges from significant distances, and thus inaccurate initial estimates can be brought into close alignment with high-contrast regions, i.e., edges, in the input data. If \( \delta \) is the set of edges, and \( D_\phi(x) \) is the distance transform to those edges, then the movement of the surface model is given by

\[
F(x) = \nabla D_\phi(x). \tag{7}
\]

**Greyscale features—gradient magnitude:** Surface models can also be attracted to certain greyscale features in the input data. For instance, the gradient magnitude indicates areas of high contrast in volumes. By following the gradient of such greyscale features, surface models are drawn to minimum or maximum values of that feature. Typically greyscale features, such as the gradient magnitude are computed with a scale operator, e.g., a derivative-of-Gaussian kernel. If models are properly initialized, they can move according to the gradient of the gradient magnitude and settle onto the edges of an object at a resolution that is finer than the original volume.

If \( G(x) \) is some greyscale feature, for instance \( G(x) = |\nabla I(x)| \), where \( I(x) \) is the input data (appropriately filtered—we use Gaussian kernels with \( \sigma \approx 0.5 \)), then

\[
F(x) = \pm \nabla G(x), \tag{8}
\]

where a positive sign moves surfaces towards maxima and the negative sign towards minima.

**Isosurface:** Surface models can also expand or contract to conform to isosurfaces in the input data. To a first order approximation, the distance from a point \( x \in U \) to the \( k \)-level surface of \( I \) is given by \( (I(x) - k)/|\nabla I| \). If we let \( g(\alpha) \) be a fuzzy threshold, e.g., \( g(\alpha) = \alpha / \sqrt{1 + \alpha^2} \), then

\[
F(x) = \frac{\nabla \phi}{|\nabla \phi|} g \left( \frac{I(x) - k}{|\nabla I|} \right). \tag{9}
\]

causes the surfaces of \( \phi \) to expand or contract to match the \( k \) isosurface of \( I \). This term combined with curvature or one of the other fitting terms can create “quasi-isosurfaces” that also include other considerations, such as smoothness or edge strength.

5 Results

This section describes how our approach may be used to extract structures from the data described in Section 2. We present surface renderings of the resulting models and detail the specific methods needed to construct each model.

Figure 10 shows 3D renderings of the sequence of steps performed on the ET dendrite data from Figure 1. The first two are the initialization steps, generating a smooth isosurface and filling gaps with topological and morphological operations. The second two are surface deformation steps, first fitting to discrete edges and then to the gradient magnitude. Figure 8 shows a slice with the boundary of the solution drawn in red, that confirms the accuracy of the results—the red boundary is only an indicator of the solution because it is limited to voxel resolution while the level set model has sub-voxel resolution. This figure also shows the same result for a smoothed isosurface—which is significantly affected by density fluctuations in the data. Figures 9 and 11 show the results of the proposed method compared to the results of a manual segmentation, which took approximately 10 hours of slice-by-slice hand contouring. The manual method suffers from slice-wise artifacts, and, because of the size and complexity of the dataset, the manual segmentation is unable to capture the level of detail that we obtain with the surface-fitting results. Manual segmentation can, however, form connections that are not well supported by the data in order to complete the “spines” that cover this dendrite. These types of
“judgments” that humans make when they perform such tasks by hand are a mixed blessing. Humans can use high-level knowledge about the problem to fill in where the data is weak, but the expectations of a trained operator can interfere with seeing unexpected or unusual features in the data. Our future work will attempt to incorporate user input to guide the surface-fitting results to obtain a better blend of user expectations and data-driven modeling.

Figure 12 shows the results of fitting a surface model to the MR head data shown in Figure 2. Figure 12(a) is a rendering of the initial model which is the result of smoothing the data, using a flood fill on the exterior to remove isolated holes or bubbles within the head, and treating the model with a closing, C5. Figure 12(b) shows the results of fitting to the isosurface with a curvature term to ensure smoothness. Some detail is lost around the lips and ears, but overall the fidelity is good and the smoother, simpler surface model suites our application quite well [20].

Figure 13 presents 3D renderings of the sequence of steps performed on the mouse MR data from Figure 3. The first step is the initialization, and the second two are the surface deformation, first fitting to discrete edges and then to the gradient magnitude. This is a significant improvement over the result in Figure 3(b) which suffers from noise and misclassifications. Figure 13(d) presents several other structures that were segmented from the mouse embryo dataset. The skin (grey) and the liver (blue) were isolated using computational initialization. The brain ventricles (red) and the eyes (green) were segmented with interactive initialization.

Figure 14 presents models from four samples of the MR series of the developing frog embryo. The top left image (Hour 9) shows the first evident structure, the blastocoel, in blue, surrounded by the outside casing of the embryo in grey. The top right image (Hour 16) demonstrates the expansion of the blastocoel and the development of the blastoporal lip in red. In the bottom left image (Hour 20) the blastoporal lip has collapsed, the blastocoel has contracted, and the archenteron in green has developed. In the bottom right image (Hour 30) the blastocoel has collapsed and only the archenteron is present. As can been seen from Figure 4(b) that it may be difficult to isolate structures using only their voxel values. We therefore used our interactive techniques to isolate (during initialization) most of the structures in the frog embryo samples.

Table 1 describes for each dataset the specific techniques and parameters we used for the results in this paper. These parameters were obtained by first making a sensible guess based on the contrasts and sizes of features in the data and then using trial and error to obtain acceptable results. Each dataset was processed between 4 and 8 times to achieve these results. More tuning could improve things further, and once these parameters are set, they work moderately well for similar modalities with similar subjects. The method is iterative, but the update times are proportional to the surface area. On an SGI 180MHz MIPS 10000 machine, the smaller mouse MR dataset required approximately 10 minutes of CPU time, and the dendrite dataset ran for approximately 45 minutes. Most of this time was spent in the initialization (which requires several complete passes through the data) and in the edge detection. The frog
Figure 10: a) The steps in the surface fitting process: An isosurface of smoothed data. b) Morphological operators fill in gaps and remove smaller, disconnected pieces. c) Fitting to edges brings the model closer to high-contrast regions in the data. d) Fitting to maximal gradient magnitude gives more detail.

Figure 11: a) Close-up view of the final result of the dendrite rendering using our scheme – note the level of details. b) Close-up view of the manual segmentation – note the lack of detail compared to the proposed method.

Figure 12: a) This image shows the rough initialization surface used for a level set segmentation of an MR scanned head. A small section of the surface has been removed to show that it does not contain internal structures. b) The final result is smoothed (almost no aliasing from the scanning slices), but with good fidelity.

Embryo datasets needed only a few minutes of processing time, because they did not require computational initialization and are significantly smaller than the other example datasets.

6 Conclusions

This paper describes a system that uses level set surface models in conjunction with a suite of initialization techniques to segment structures in volume data. Level set surface modeling is a technology that allows one to manipulate or deform the isosurfaces of a volume toward interesting features in the input data. Because the technology is volumetric, it provides opportunities to combine voxel-based techniques, such as filtering, classification, and morphology with surface-fitting methods based on deformable models.

We have shown that combining level set methods with a variety of initialization techniques produces a powerful framework capable of segmenting many different types of volume datasets. In the case of the ET dendrite data, our approach offers significant advantages in both time and quality over hand-contoured segmentations, which are currently the state-of-the-art.

Currently there are two significant drawbacks of the proposed method. First is the choice of parameters. There are a number of parameters that must be tuned, and their settings affect the final solution. The second drawback is the computation time, which is quite long for large datasets. The second problem aggravates the first, because exploring the parameter space by trial and error is a potentially lengthy process. Future work will focus on increasing the update rates by parallelizing the computation. This is feasible because the numerical methods lend themselves to a spatial decomposition of the model domain. If the updates were sufficiently fast, users can explore the parameter space interactively by turning various knobs and evaluating the quality of the results. This would greatly increase the effectiveness of the method.

Acknowledgements

We would like to thank Dr. Alan Barr for his support of this project, and Dr. Cyrus Papan, Dr. Russ Jacobs, Dr. Eric Ahrens, Dr. Mark Ellisman, Dr. Maryanne Martone and Mr. David Weinstein for their assistance and cooperation. This work was supported
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Initialization</th>
<th>Surface Fitting</th>
</tr>
</thead>
</table>
| Dendrite | 1. Gaussian blur $\sigma = 0.5$  
2. Threshold: $I < 127$  
3. Fill isolated holes  
4. Morphology: $O_{0.5} \circ C_{1.5}$ | 1. Edge fitting: $\sigma = 0.75$, threshold = 6, $\beta = 0.1$  
2. Gradient magnitude fitting: $\sigma = 0.5$, $\beta = 1.0$ |
| Head | 1. Gaussian blur $\sigma = 1.0$  
2. Threshold: $I > 30$  
3. Fill isolated holes  
4. Morphology: $C_{5.0}$ | 1. Isosurface fitting: $\beta = 5.0$, $k = 30$. |
| Mouse | 1. Gaussian blur $\sigma = 0.5$  
2. Threshold: $I > 3, I < 60$  
3. Fill isolated holes  
4. Morphology: $O_{2.0} \circ C_{3.0}$ | 1. Edge fitting: $\sigma = 0.75$, threshold = 20, $\beta = 2$  
2. Gradient magnitude fitting: $\sigma = 0.5$, $\beta = 16.0$ |
| Frog | 1. Interactive | 1. Gradient magnitude fitting: $\sigma = 1.25$, $\beta = 1.0$ |

Table 1: Parameters for processing example datasets.

Figure 13: a) The initialization of a mouse liver dataset using morphology to remove small pieces and holes. b) Surface fitting to discrete edges. c) The final fit to maxima of gradient magnitude. d) Final mouse embryo model with skin (grey), liver (blue), brain ventricles (red), and eyes (green).

Figure 14: Geometric structures extracted from MRI scans of a developing frog embryo, with blastocoel (blue), blastoporal lip (red), and archenteron (green). Hour 9 (top left). Hour 16 (top right). Hour 20 (bottom left). Hour 30 (bottom right).
References


Level Set Modeling and Segmentation of DT-MRI Brain Data

Leonid Zhukov, Ken Museth, David Breen, Ross Whitaker†, Alan H. Barr

Department of Computer Science, California Institute of Technology
Mail Code 350-74, Pasadena, CA 91125-7400
†School of Computing, University of Utah
3190 MEB, Salt Lake City, UT 84112-9205

ABSTRACT

Segmentation of anatomical regions of the brain is one of the fundamental problems in medical image analysis. It is traditionally solved by iso-surfacing or through the use of active contours/deformable models on a gray-scale MRI data. In this paper we develop a technique that uses anisotropic diffusion properties of brain tissue available from DT-MRI to segment out brain structures. We develop a computational pipeline starting from raw diffusion tensor data, through computation of invariant anisotropy measures to construction of geometric models of the brain structures. This provides an environment for user-controlled 3D segmentation of DT-MRI datasets. We use a level set approach to remove noise from the data and to produce smooth, geometric models. We apply our technique to DT-MRI data of a human subject and build models of the isotropic and strongly anisotropic regions of the brain. Once geometric models have been constructed they may be combined to study spatial relationships and quantitatively analyzed to produce the volume and surface area of the segmented regions.

Keywords: DT-MRI, diffusion tensor, eigenvalues, invariants, iso-surface, geometric model, level set methods.

1. INTRODUCTION

Diffusion tensor magnetic resonance imaging\(^1,2\) (DT-MRI) is a technique used to measure the diffusion properties of water molecules in tissues. Anisotropic diffusion can be described by the equation

\[
\frac{\partial C}{\partial t} = \nabla : (\mathbf{D} \nabla C)
\]

where \(C\) is the concentration of water molecules and \(\mathbf{D}\) is a diffusion coefficient, which is a symmetric second order tensor

\[
\mathbf{D} = \begin{pmatrix}
D_{xx} & D_{xy} & D_{xz} \\
D_{yx} & D_{yy} & D_{yz} \\
D_{zx} & D_{zy} & D_{zz}
\end{pmatrix}.
\]

Figure 1 presents a “slice” of the diffusion tensor volume data of human brain used in our study. Each sub-image presents the scalar values of the associated diffusion tensor component for one slice of the dataset.

Tissue segmentation and classification based on DT-MRI offers several advantages over conventional MRI, since diffusion data contains additional physical information about the internal structure of the tissue being scanned. However, segmentation and visualization using diffusion data is not entirely straightforward. First of all, the diffusion matrix itself is not invariant with respect to rotations, and the elements that form the matrix will be different for different orientations of the sample or field gradient and therefore cannot themselves be used for classification purposes. Moreover, 3D visualization and segmentation techniques available today are predominantly designed for scalar and sometimes vector fields. Thus, there are two fundamental problems in tensor imaging: a) finding an invariant representation of a tensor that is independent of a frame of reference and constructing a mapping from the tensor field to a scalar or vector field, and b) visualization and classification of tissue using the derived scalar fields.

The traditional approaches to diffusion tensor imaging involve converting the tensors into an eigenvalue/eigenvector representation, which is rotationally invariant. Every tensor may then be interpreted as an ellipsoid with principal...
Figure 1. a: Slice of a tensor volume where every “element” of the image matrix corresponds to one component of the tensor $D$.

axes oriented along the eigenvectors and radii equal to the corresponding eigenvalues. This ellipsoid describes the probabilistic distribution of a water molecule after a fixed diffusion time.

Using eigenvalues/eigenvectors one can compute different anisotropy measures\cite{1,3,5} that map tensor data onto scalars and can be used for further visualization and segmentation. Although eigenvalue/vector computation of the 3x3 matrix is not expensive, it must be repeatedly performed for every voxel in the volume. This calculation easily becomes a bottleneck for large datasets. For example, computing eigenvalues and eigenvectors for a $512^3$ volume requires over 20 CPU-minutes on a powerful workstation. Another problem associated with eigenvalue computation is stability - a small amount of noise will not only change the values but also the ordering of the eigenvalues.\cite{7} Since many anisotropy measures depend on the ordering of the eigenvalues, the calculated direction of diffusion and classification of tissue will be significantly altered by the noise normally found in diffusion tensor datasets. Thus it is desirable to have an anisotropy measure which is rotationally invariant, does not require eigenvalue computations and is stable with respect to noise. The tensor invariants with these characteristics were first proposed by Ulug and Zijl.\cite{8} In Section 2 of this paper we formulate a new anisotropy measure for tensor field based on these invariants.

Visualization and model extraction from the invariant 3D scalar fields is the second issue addressed in this paper. One of the popular approaches to tensor visualization represents a tensor field by drawing ellipsoids associated with the eigenvectors/values.\cite{9} This method was developed for 2D slices and creates visual cluttering when used in 3D. Other standard CFD visualization techniques like tensor-lines do not provide meaningful results for the MRI data due to rapidly changing directions and magnitudes of eigenvector/values and also amount of noise present in the data. Recently Kindlmann\cite{10} developed volume rendering approach to tensor field visualization using eigenvalue-based anisotropy measures to construct transfer function and color maps, that highlight some brain structures and diffusion patterns.

In our work we perform iso-surfacing on the 3D scalar fields derived from our tensor invariants to visualize and segment the data. An advantage of iso-surfacing over other approaches is that it can provide the shape information needed for constructing geometric models, and computing internal volumes and external surface areas of the extracted regions. A detailed discussion of the modeling method is given in Section 3. Section 4 presents the results of tensor invariant calculations and model segmentation technique with examples from a DT-MRI scan of a human head. Section 5 then describes the quantitative analysis of obtained geometric models.
Finally, there has been a number of recent publications\textsuperscript{11,12} devoted to brain fiber tracking. This is a different and more complex task than the one addressed in this paper and requires data with a much higher resolution and better signal-to-noise ratio than the data used in our study.

\section{2. Tensor Invariants}

Tensor invariants (rotational invariants) are combinations of tensor elements that do not change after the rotation of the tensor's frame of reference, and thus do not depend on the orientation of the patient with respect to the scanner when performing DT imaging. The well-known invariants are the eigenvalues of diffusion tensor (matrix) $D$, which are the roots of corresponding characteristic equation
\begin{equation}
\lambda^3 - C_1 \lambda^2 + C_2 \lambda - C_3 = 0,
\end{equation}
with coefficients
\begin{align*}
C_1 &= D_{xx} + D_{yy} + D_{zz} \\
C_2 &= D_{xx}D_{yy} - D_{xy}D_{yx} + D_{xx}D_{zz} - D_{xz}D_{zx} + D_{yy}D_{zz} - D_{yz}D_{zy} \\
C_3 &= D_{xx}(D_{yy}D_{zz} - D_{yz}D_{zy}) - D_{xy}(D_{yz}D_{zy} - D_{xz}D_{yx}).
\end{align*}

Since the roots of Equation (3) are rotational invariants, the coefficients $C_1$, $C_2$ and $C_3$ are also invariant. In the eigen-frame of reference they can be easily expressed through the eigenvalues
\begin{align}
C_1 &= \lambda_1 + \lambda_2 + \lambda_3 \\
C_2 &= \lambda_1 \lambda_2 + \lambda_1 \lambda_3 + \lambda_2 \lambda_3 \\
C_3 &= \lambda_1 \lambda_2 \lambda_3,
\end{align}
and are proportional to the sum of the radii, surface area and the volume of the "diffusion" ellipsoid. Then instead of using $(\lambda_1, \lambda_2, \lambda_3)$ to describe the dataset, we can use $(C_1, C_2, C_3)$. Moreover, since $C_i$ are the coefficients of characteristic equation, they are less sensitive to noise, then roots $\lambda_i$ of the same equation.

Any combination of the above invariants is, in turn, an invariant. We consider the following dimensionless combination: $C_1C_2/C_3$. In the eigenvector frame of reference it becomes
\begin{equation}
\frac{C_1C_2}{C_3} = 3 + \frac{\lambda_2 + \lambda_3}{\lambda_1} + \frac{\lambda_1 + \lambda_3}{\lambda_2} + \frac{\lambda_1 + \lambda_2}{\lambda_3},
\end{equation}
and we can define a new dimensionless anisotropy measure
\begin{equation}
C_a = \frac{1}{6} \left[ \frac{C_1C_2}{C_3} - 3 \right].
\end{equation}

It is easy to show that for isotropic diffusion, when $\lambda_1 = \lambda_2 = \lambda_3$, the coefficient $C_a = 1$. In the anisotropic case, this measure is identical for both linear, directional diffusion ($\lambda_1 \gg \lambda_2 \approx \lambda_3$) and planar diffusion ($\lambda_1 \approx \lambda_2 \gg \lambda_3$) and is equal to
\begin{equation}
C_a^{\text{IM}} \approx \frac{1}{3} \left[ 1 + \frac{\lambda_1}{\lambda_3} + \frac{\lambda_3}{\lambda_1} \right].
\end{equation}

Thus $C_a$ is always $\sim \lambda_{\text{max}}/\lambda_{\text{min}}$ and measures the magnitude of the diffusion anisotropy. We again want to emphasize, that we use eigenvalue representation here only to analyze the behavior of the coefficient $C_a$, but we use invariants $(C_1, C_2, C_3)$ to compute it using Equations (5) and (7).
3. GEOMETRIC MODELING

Two options are usually available for viewing the scalar volume datasets, direct volume rendering\textsuperscript{14,15} and volume segmentation\textsuperscript{16} combined with conventional surface rendering. The first option, direct volume rendering, is only capable of supplying images of the data. While this method may provide useful views of the data, it is well-known that it is difficult to construct the exact transfer function that highlights the desired structures in the volume dataset.\textsuperscript{17} Our approach instead focuses on extracting geometric models of the structures embedded in the volume datasets. The extracted models may be used for interactive viewing, but the segmentation of geometric models from the volume datasets provides a wealth of additional benefits and possibilities. The models may be used for quantitative analysis of the segmented structures, for example the calculation of surface area and volume; quantities that are important when studying how these structures change over time. The models may be used to provide the shape information necessary for anatomical studies and computational simulation, for example EEG/MEG modeling within the brain.\textsuperscript{18} Creating separate geometric models for each structure allows for the straightforward study of the relationship between the structures, even though they come from different datasets. The models may also be used within a surgical planning/simulation/VR environment,\textsuperscript{19} providing the shape information needed for collision detection and force calculations. The geometric models may even be used for manufacturing real physical models of the structures.\textsuperscript{20} It is clear that there are numerous reasons to develop techniques for extracting geometric models from diffusion tensor volume datasets.

The most widely used technique for extracting polygonal models from volume datasets is the Marching Cubes algorithm.\textsuperscript{21} This technique creates a polygonal model that approximates the iso-surface embedded in a scalar volume dataset for a particular iso-value. The surface represents all the points within the volume that have the same scalar value. The polygonal surface is created by examining every “cube” of eight volume grid points and defining a set of triangles that approximates the piece of the iso-surface within the space bounded by the eight points. While the Marching Cubes algorithm is easy to understand and straightforward to implement, applying it directly to raw volume data from scanners can produce undesirable results, as seen in top row images in Figures 4, 7. The algorithm is susceptible to noise and can produce many unwanted triangles that mask the central structures in the data. In order to alleviate this problem, we utilize a deformable model approach to smooth the data and remove the noise-related artifacts. Many types of deformable models have been proposed for extracting structures from volumes.\textsuperscript{16,22} We utilize level set models as they have been shown to be flexible and effective for segmentation.\textsuperscript{23,24,26-28} Level set methods produce active deformable surfaces that may be directed to conform to features in a volume dataset while simultaneously applying a smoothing operation based on local surface curvature.\textsuperscript{28} Most importantly, they easily change topology during deformation and have no fixed parameterization, allowing them to represent complex shapes.

3.1. Level Set Models

A level set model\textsuperscript{20,29} specifies a surface as a level set (iso-surface) of a scalar volumetric function, $\phi: U \mapsto \mathbb{R}$, where $U \subset \mathbb{R}^3$ is the range of the surface model. Thus, a surface $S$ is

$$ S = \{ s \mid \phi(s) = k \}, $$

(9)
and \( k \) is the isovalue. In other words, \( S \) is the set of points \( s \) in \( \mathbb{R}^3 \) that composes the \( k \)th iso-surface of \( \phi \). The embedding \( \phi \) can be specified as a regular sampling on a rectilinear grid. The surfaces may propagate with (time-varying) curvature-dependent speeds. Level set methods provide the mathematical and numerical mechanisms for computing surface deformations as iso-values of \( \phi \) by solving a partial differential equation on the 3D grid \((U)\). That is, the level set formulation provides a set of numerical methods that describes how to manipulate the grey-scale values in a volume, so that the iso-surfaces of \( \phi \) move in a prescribed manner. See Figure 3.

**Figure 3.** Level set models represent curves and surfaces implicitly using grey scale images. For example an ellipse is represented as the level set of an image (left). To change the shape of the ellipse we modify the grey scale values of the image by solving a PDE (right).

There are two different approaches to defining a deformable surface from a level set of a volumetric function as described in Equation (9). Either one can think of \( \phi(s) \) as a static function and change the isovalue \( k(t) \) or alternatively fix \( k \) and let the volumetric function dynamically change in time, i.e. \( \phi(s,t) \). Following the second approach, we can mathematically express the dynamic model as

\[
\phi(s,t) = k.
\] (10)

To transform this definition into partial differential equation that can easily be solved by standard numerical techniques, we differentiate both sides of Equation (10) with respect to time \( t \), and apply the chain rule:

\[
\frac{\partial \phi(s,t)}{\partial t} + \nabla \phi(s,t) \cdot \frac{\text{d}s}{\text{d}t} = 0.
\] (11)

Equation (11) is sometimes referred to as a “Hamilton-Jacobi-type” equation and defines an initial value problem for the time-dependent \( \phi \). Let \( \text{d}s/\text{d}t \) be the movement of a point on a surface as it deforms, such that it can be expressed in terms of the position of \( s \in U \) and the geometry of the surface at that point, which is, in turn, a differential expression of the implicit function, \( \phi \). This gives a partial differential equation (PDE) on \( \phi: s \equiv s(t) \)

\[
\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{\text{d}s}{\text{d}t} = -\nabla \phi \cdot F(s,D\phi,D^2\phi,\ldots),
\] (12)

where \( F \) is a user-defined “speed” term which generally depends on a set of order-\( n \) derivatives of \( \phi \), \( D^n\phi \), evaluated at \( s \), as well as other functions of \( s \). Typically \( F(x) \) combines a data term with a smoothing term, which prevents the solution from fitting too closely to noise-corrupted data. There are a variety of surface-motion terms that can be used in succession or simultaneously in a linear combination to form \( F(x) \). For the work presented in this paper, we combine a feature attraction term and a smoothing term weighted by a factor \( \beta \),

\[
F = F_{\text{attr}} + \beta F_{\text{curv}}.
\] (13)

The first term \( F_{\text{attr}} \) is due to the attraction to the edges in the volume. It attracts the surface models to certain grey scale features in the input data. For instance, the gradient magnitude indicates areas of high contrast in volumes. By following the gradient of such grey scale features, surface models are drawn to minimum or maximum values of that feature. Typically grey scale features, such as the gradient magnitude are computed with a scale operator, e.g., a derivative-of-Gaussian kernel. If models are properly initialized, they can move according to the gradient magnitude and settle onto the edges of an object at a resolution that is finer than the original volume. For this work we used the attraction force

\[
F_{\text{attr}} = \nabla |(\nabla (G*I(x)))|,
\] (14)
where the volume data $I(x)$ is convolved with a Gaussian kernel $G$ with $\sigma \approx 0.5$, such that a positive sign moves surfaces towards maxima and the negative sign towards minima.

There are a variety of options for the curvature smoothing terms in Equation (13), and the question of efficient, effective higher-order smoothing terms is the subject of ongoing research.\textsuperscript{20} For the work presented in this paper the smoothing term uses the mean curvature $K_M$ of the level set $S$ to form a vector in the direction of the surface normal $n$:

$$ F_{\text{curv}} = K_M n = (\nabla \cdot n) n = \nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right) \frac{\nabla \phi}{|\nabla \phi|}. $$

(15)

It is weighted by a factor $\beta$, allowing the user to control the amount of smoothing, and is tuned for each dataset. The level set propagation stops when the $F_{\text{attr}}$ and $\beta F_{\text{curv}}$ terms cancel each other, or when the number of computational iterations reaches a user-specified value.

Level set models have a number of practical and theoretical advantages over conventional surface models, especially in the context of deformation and segmentation. Level set models are topologically flexible; they easily represent complicated surface shapes that can, form holes, split to form multiple objects, or merge with other objects to form a single structure. These models can incorporate many (millions) of degrees of freedom, and therefore they can accommodate complex shapes. Indeed, the shapes formed by the level sets of $\phi$ are restricted only by the resolution of the sampling. Thus, there is no need to re-parameterize the model as it undergoes significant changes in shape.

The solutions to the partial differential equations described above are computed using finite differences on a discrete grid. The use of a grid and discrete time steps raises a number of numerical and computational issues that are important to the implementation. However, it is outside of the scope of this paper to give a detailed mathematical description of such a numerical implementation. Rather we shall give a short outline below and refer to the actual source code which is publicly available\textsuperscript{*}.

Equation (12) to (15) can be solved using finite forward differences if one uses the up-wind scheme, proposed by Osher and Sethian,\textsuperscript{20} to compute the spatial derivatives. This up-wind scheme produces the motion of level-set models over the entire range of the embedding, i.e., for all values of $k$ in Equation (10). However, this method requires updating every voxel in the volume for each iteration, which means that the computation time increases as a function of the volume, rather than the surface area, of the model. Because segmentation requires only a single model, the calculation of solutions over the entire range of iso-values is an unnecessary computational burden.

This problem can be avoided by the use of narrow-band methods, which compute solutions only in a narrow band of voxels that surround the level set of interest.\textsuperscript{24} In previous work\textsuperscript{25} we described an alternative numerical algorithm, called the sparse-field method, that computes the geometry of only a small subset of points in the range and requires a fraction of the computation time required by previous algorithms. We have shown two advantages to this method. The first is a significant improvement in computation times. The second is increased accuracy when fitting models to forcing functions that are defined to sub-voxel accuracy.

4. SEGMENTATION

In this section we demonstrate the application of our methods to the segmentation of DT-MRI data of the human head. We use a high resolution data set from a normal volunteer which contains $60$ slices each of $128 \times 128$ pixels resolution. The raw data is sampled on a regular uniform grid.

We begin by generating two scalar volume datasets based on the invariants described in Section 2. The first scalar volume dataset ($V_1$) is formed by calculating the trace ($C_1$) of the tensor matrix for each voxel of the diffusion tensor volume. It provides a single number that characterizes the total diffusivity at each voxel within the sample. Higher values signify greater total diffusion irrespective of directionality in the region represented by a particular voxel. A slice from this volume can be seen in Figure 2 (left). The second scalar volume dataset ($V_2$) is formed by calculating ($C_1, C_2, C_3$) invariants for each voxel and combining them into $C_a$. It provides a measure of the magnitude of the anisotropy within the volume. Higher values identify regions of greater spatial anisotropy in the diffusion properties. A slice from the second scalar volume is presented in Figure 2 (right). The measure $C_a$ does not by definition

\textsuperscript{*}The level-set software used to produce the morphing results in this paper is available for public use in the VISPACK libraries at http://www.cs.utah.edu/~whitaker/vispack.
distinguish between linear and planar anisotropy. This is sufficient for our current study since the brain does not contain measurable regions with planar diffusion anisotropy. We therefore only need two scalar volumes in order to segment the DT dataset.

We then utilize level set methods to extract smoothed models from the two derived scalar volumes. Our level set segmentation approach consists of defining a set of suitable pre-processing techniques for initialization and selecting/tuning different feature-extracting terms in the level set equation to produce a surface deformation. Within our segmentation framework a variety of operations are available in each stage. A user must “mix-and-match” these operations in order to produce the desired result. We only describe those operations needed to produce the models in this paper. A more detailed description of our segmentation methods may be found in.

Because level set models move using gradient descent, they seek local solutions, and therefore the results are strongly dependent on the initialization, i.e., the starting position of the surface. Thus, one controls the nature of the solution by specifying an initial model from which the surface deformation process proceeds. We are able to computationally construct reasonable initial estimates directly from the input data by combining a variety of techniques.

The first step involves filtering the input data with a low-pass Gaussian filter ($\sigma \approx 0.5$) to blur the data and thereby reduce noise. This tends to distort shapes, but the initialization need only be approximate. Next, the volume voxels are classified for inclusion/exclusion in the initialization based on the filtered values of the input data ($k \approx 7.0$ for $V_1$ and $k \approx 1.3$ for $V_2$). For grey scale images, such as those used in this paper, the classification is equivalent to high and low thresholding operations. These operations are usually accurate to only voxel resolution, but the deformation process will achieve sub-voxel results. The final step before the actual level set deformation consist of performing a set of topological or logical operations on the voxels to “clean up” the initialization surface. This allows for the removal of undesired internal and external structures, which is extremely helpful to obtain simple models. It includes unions or intersections of voxel sets to create the better initializations. Specifically the topological operations consist of connected-component analyses (e.g. flood fill) to remove small pieces or holes from objects.

The initialization described above positions the model near the desired solution while retaining certain properties such as consistent geometry, connectivity, etc. Given this rough initial estimate, the level set surface deformation process, as described in Section 3.1, moves the surface model toward specific features in the data.

Figures 4 and 5 present two models that we extracted from DT-MRI volume datasets using our techniques. Figure 4 contains segmentations from volume $V_1$, the measure of total diffusivity. The image in the first row shows a marching cubes iso-surface using an iso-value of 7.5. In the bottom we have extracted just the ventricles from $V_1$. This is accomplished by creating an initial model with a flood-fill operation inside the ventricle structure shown in the middle image. This identified the connected voxels with value of 7.0 or greater. The initial model was then refined and smoothed with the level set method described in Section 3, using a $\beta$ value of 0.2.

Figure 5 again provides the comparison between direct iso-surfacing and and level set model, but on the volume $V_2$. The image in the top-left corner is a marching cubes iso-surface using an iso-value of 1.3. There is significant high-frequency noise and features in this dataset. The challenge here was to isolate coherent regions of high anisotropic diffusion. We applied our segmentation approach to the dataset and worked with neuroscientists from LA Childrens Hospital, City of Hope Hospital and Caltech to identify meaningful anatomical structures. We applied our approach using a variety of parameter values, and presented our results to them, asking them to pick the model that they felt best represented the structures of the brain. Figure 5 contains three models extracted from $V_2$ at different values of smoothing parameter $\beta$ used during segmentation. Since we were not looking for a single connected structure in this volume, we did not use a seeded flood-fill for initialization. Instead we initialized the deformation process with an iso-surface of value 1.3. This was followed by a level set deformation using a $\beta$ value of 0.2. The result of this segmentation is presented on the bottom-left side of Figure 5. The top-right side of this figure presents a model extracted from $V_2$ using an initial iso-surface of value 1.4 and a $\beta$ value of 0.5. The result chosen as the “best” by our scientific/medical collaborators is presented on the bottom-right side of Figure 5. This model is produced with an initial iso-surface of 1.3 and a $\beta$ value of 0.4. Our collaborators were able to identify structures of high diffusivity in this model, for example the corpus callosum, the internal capsule, the optical nerve tracks, and other white matter regions.

We can also bring together the two models extracted from datasets $V_1$ and $V_2$ into a single image. Figure 6 demonstrates that we are able to isolate different structures in the brain and show their proper spatial inter-
Table 1. Total polygon count in the models $N_{poly}$, surface areas $A$ and volumes $V$ and before/after application of the level set smoothing to datasets $V_1$ and $V_2$.

<table>
<thead>
<tr>
<th>Data set</th>
<th>$N_{poly}$</th>
<th>$A$ (cm$^2$)</th>
<th>$V$ (cm$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
<td>36,620/13,096</td>
<td>188/83</td>
<td>25/22</td>
</tr>
<tr>
<td>$V_2$</td>
<td>142,212/81,488</td>
<td>760/743</td>
<td>98/87</td>
</tr>
</tbody>
</table>

relationship. For example, it can be seen that the corpus callosum lies directly on top of the ventricles, and that the white matter fans out from both sides of the ventricles.

Finally, to verify the validity of our approach we applied it to the second data set of a different volunteer. This data set has 20 slices of the 256x256 resolution. We generated the anisotropy measure volume $V_2^p$ and performed the level-set model extraction using the same iso-values and smoothing parameters as for $V_2$. The results are shown in Figure 7.

5. MODEL PROPERTIES

Once a user has produced a satisfactory model of the desired segmented structures s/he may perform a number of quantitative geometric calculations on the resulting polygonal model, e.g., total area, volume, and average curvature. Though most of these measures are interesting from the modeling point of view, the volume of the ventricles, for example, can have clinical applications for disorder diagnosis and population comparison.

The models generated in the previous section are represented by triangle meshes consisting of vertices $v_i$, connectivities and associated normal vectors. The total surface area of the model can be easily computed by adding the areas $A_i$ of each triangle

$$A = \sum_{i=1}^{N_{poly}} A_i = \sum_{i=1}^{N_{poly}} \frac{1}{2} |(v_i^1 - v_i^2) \times (v_i^1 - v_i^3)|,$$

where $v_i^k$ is the $k$'th vertex of triangle $i$. Assuming that all of the extracted models are composed of closed polygonal surfaces, we can compute enclosed volume as a signed sum of the pyramids with a base composed of the $i$'th triangle and a top vertex places at the origin of the dataset. Then

$$V \approx \frac{1}{6} \sum_{i=1}^{N_{poly}} A_i \cdot \frac{1}{3} (v_i^1 + v_i^2 + v_i^3) \cdot N_i.$$

Table 1 lists values of polygon count, surface area and total volume, for the models extracted from scalar volume datasets ($V_1$ and $V_2$), before and after the level set algorithm is applied to the volumes. We note that the polygon count drops, because of the simplified form of the final extracted triangular mesh. The total surface area decrease is also due to smoothing imposed by the level set model. Volume decrease is partially caused by the removal (i.e. collapse) of small high frequency fragments cluttering the model and partially due to deformations of the model.

6. CONCLUSIONS

In this paper we have developed a computational pipeline for DT-MRI level set modeling and segmentation. We proposed a new rotationally invariant anisotropy measure, which does not require eigenvalue computations. We used the invariants to generate scalar volumes that characterize the total diffusivity and diffusion anisotropy of a DT-MRI scan of a human brain. Applying level set modeling and segmentation techniques to the derived scalar volumes we created geometric models of specific brain structures, e.g the ventricles, corpus callosum, and the internal capsul.

The geometric models were then used for quantitative analysis, including volume and surface area calculations.
We would like to thank Dr. J. Michael Tyszka, Dr. Miriam Scadeng and Dr. David Dubowitz for helping us to identify the 3-D structures extracted from the DT dataset. Dr. Jason Wood developed the Iris Explorer modules used to produce part of the results in this paper. This work was supported by National Science Foundation grants #ACI-9982273 and #ASC-89-20219, the National Institute on Drug Abuse, the National Institute of Mental Health and the NSF, as part of the Human Brain Project, Office of Naval Research Volume Visualization grant #N000140101003, and the National Library of Medicine “Insight” Project #N01-LM-0-3503. The first DT-MRI dataset is courtesy of the University of Utah SCI Institute, the second dataset is courtesy of Dr. Mark Bastin, University of Edinburgh, UK. Finally, we would like to thank our reviewers for a very detailed review and multiple valuable suggestions.

REFERENCES
Figure 4. Segmentation using isotropic measure $V_1$ for the first DT-MRI dataset. The first row is the marching cubes iso-surface with 7.5 iso-value. The second row is the result of flood-fill algorithm applied to the same volume and used for level set initialization. The third row is the final level set model.
Figure 5. Model segmentation from volume $\mathcal{V}_2$. Top left image is an iso-surface of value 1.3, used for initialization of the level set. Clockwise, are the results of level set development with corresponding $\beta$ values of 0.2, 0.4 and 0.5.
Figure 6. Combined model of ventricles and (semi-transparent) anisotropic regions: rear, exploded view (left), bottom view (right), side view (bottom). Note how model of ventricles extracted from isotropic measure dataset $\mathcal{V}_1$ fits into model extracted from anisotropic measure dataset $\mathcal{V}_2$. 
Figure 7. Segmentation using anisotropic measure $\nu_2$ from the second DT-MRI dataset. The first row is the marching cubes iso-surface with iso-value 1.3. The second row is the result of flood-fill algorithm applied to the volume and used for level set initialization. The third is the final level set model.
Level-Set Segmentation From Multiple Non-Uniform Volume Datasets
(K. Museth, D. Breen, L. Zhukov, R. Whitaker)
[IEEE Visualization '02]

Problem Statement

Assumptions And Limitations
- Registration is known!
  - Some are perfectly aligned
  - Some are done by hand
- Simple Merging or standard interpolation not feasible
  - Highly non-uniform samplings – regular reconstruction filters won’t work
  - Possible attenuation effects - gain-corrections needed
  - Noisy input – no direct iso-surfacing

Segmentation With Edges

\[ E(VI) = \int_\Omega E(VI) H(\phi(x)) dx \]
\[ E(VI) = \frac{1}{1 + |G_\epsilon * I|^2} \]
\[ \frac{\partial \phi}{\partial t} = [E(VI) + \nabla E(VI) \cdot \nabla] \nabla \phi \]

- Depends on derivatives of intensities
  - No need for a global intensity reconstruction
  - Only local computation needed
  - 3D edge-detector is short-ranges
Long-Range Edge Detector

Canny Edges = \( \{ \vec{x} \in \mathbb{R}^3 \mid D_N(\vec{x}) = \epsilon \land \nabla^2 D_N(\vec{x}) = 0 \} \)

Directional derivatives: \( \hat{n}_i = \nabla [n|n] \)

\( \nabla = \hat{n}_1 \cdot \nabla |\hat{n}_1 \cdot \nabla | = \hat{n}_1 \cdot \nabla |\nabla | \)

Manhattan distance transform

\( \Gamma(\vec{n}) = \vec{n} \cdot (- \nabla ) \)

Velocity field

Normal of LS surface

Partial Derivatives by MLS

\( \nabla \hat{n}_I = \frac{\hat{n}_I \cdot \hat{\nabla}[\vec{H}(\vec{x})]}{|\nabla |} \)

\( \hat{\nabla}[\vec{H}(\vec{x})] = \begin{bmatrix} \frac{\partial H(x)}{\partial x} & \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial z} \\ \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial x} & \frac{\partial H(x)}{\partial z} \\ \frac{\partial H(x)}{\partial z} & \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial x} \end{bmatrix} \)

\( \hat{\nabla}[\vec{H}(\vec{x})] = \begin{bmatrix} \frac{\partial H(x)}{\partial x} & \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial z} \\ \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial x} & \frac{\partial H(x)}{\partial z} \\ \frac{\partial H(x)}{\partial z} & \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial x} \end{bmatrix} = \begin{bmatrix} C_x(0) & C_y(0) & C_z(0) \\ C_x(0) & C_y(0) & C_z(0) \\ C_x(0) & C_y(0) & C_z(0) \end{bmatrix} \)

\( \hat{\nabla}[\vec{H}(\vec{x})] = \begin{bmatrix} \frac{\partial H(x)}{\partial x} & \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial z} \\ \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial x} & \frac{\partial H(x)}{\partial z} \\ \frac{\partial H(x)}{\partial z} & \frac{\partial H(x)}{\partial y} & \frac{\partial H(x)}{\partial x} \end{bmatrix} = \begin{bmatrix} 2C_x(0) & C_y(0) & C_z(0) \\ C_x(0) & 2C_y(0) & C_z(0) \\ C_x(0) & C_y(0) & 2C_z(0) \end{bmatrix} \)

Moving Least Square

Expansion coefficients

\( j_N^d(x - \vec{x}_0) = \sum_{i = 0}^{D} \sum_{j = 0}^{D} j_N^d(x - \vec{x}_i) \sum_{k = 0}^{D} \sum_{l = 0}^{D} \vec{x}_j - \vec{x}_i \) \( l_d^d(\vec{x}_j - \vec{x}_0) \)

Expansion point

\( E(\vec{x}_0) = \sum_{d = 1}^{D} \sum_{j = 1}^{D} \sum_{k = 1}^{D} \sum_{l = 1}^{D} \vec{x}_j - \vec{x}_0 \) \( l_d^d(\vec{x}_j - \vec{x}_0) \)

Sampling points for volume #d

Input volume #d

Algorithm Overview

Approximate initial surface

– Low-order global uniform re-sampling of input volumes
– Compute union to form single uniform volume.
Pull surface close to edges

– High-order local approximation of derivatives by Moving Least Squares. (in narrow band only!)
– Attraction to the long-range binary Canny edges.
Final surface deformation

– Attraction to short-range 3D directional edges.
– Regularize with curvature based flow
After Canny edge refinement

Final level set model

Composite of mouse segmentation

Three Non-Uniform CT Scans Of A (Real) Teapot

Laser Scan Reconstruction Of Griffin

Three MRI Scans Of A Zucchini
Three MRI Scans Of A Zucchini

Summary

- Relatively fast
  - Global tri-linear re-sampling of individual low-resolution input volumes.
  - Local N-order polynomial approximation of derivatives in a narrow band of the CSG union.
  - Complexity scales with the area of the model – not volume
- Flexible
  - Deforming LS model can change topology
  - Uses distance-weighting of irregular samplings
  - Allow for local gain-correction
- Robust
  - Using robust numerical algorithms like MLS and SVD
  - Relatively stable with respect to noise and imperfect registration
Level Set Segmentation From Multiple Non-Uniform Volume Datasets

Ken Museth†∗ David E. Breen† Leonid Zhukov† Ross T. Whitaker†
†Computer Science Department
California Institute of Technology
‡School of Computing
University of Utah

Figure 1: 3D level set model of a griffin derived from two non-uniform laser scan reconstruction volume datasets. The two input models with resolution 294 × 312 × 24 and 294 × 52 × 144 show severe aliasing effects from insufficient sampling, especially in the wings. Our method merges information from both of the datasets to produce a high resolution 294 × 312 × 144 level set model. All models throughout this paper are flat-shaded to highlight details.

Abstract

Typically 3-D MR and CT scans have a relatively high resolution in the scanning X – Y plane, but much lower resolution in the axial Z direction. This non-uniform sampling of an object can miss small or thin structures. One way to address this problem is to scan the same object from multiple directions. In this paper we describe a method for deforming a level set model using velocity information derived from multiple volume datasets with non-uniform resolution in order to produce a single high-resolution 3D model. The method locally approximates the values of the multiple datasets by fitting a distance-weighted polynomial using moving least-squares. The proposed method has several advantageous properties: its computational cost is proportional to the object surface area, it is stable with respect to noise, imperfect registrations and abrupt changes in the data, it provides gain-correction, and it employs a distance-based weighting to ensures that the contributions from each scan are properly merged into the final result. We have demonstrated the effectiveness of our approach on four multi-scan datasets, a griffin laser scan reconstruction, a CT scan of a teapot and MR scans of a mouse embryo and a zucchini.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Surface and object representations;

Keywords: Segmentation, visualization, level set models, 3D reconstruction.

*kmu@caltech.edu

1 Introduction

Many of today’s volumetric datasets are generated by medical MR, CT and other scanners. A typical 3-D scan has a relatively high resolution in the scanning X – Y plane, but much lower resolution in the axial Z direction. The difference in resolution between the in-plane and out-of-plane samplings can easily range between a factor of 5 to 10, see figure 1. This occurs both because of physical constraints on the thickness of the tissue to be excited during scanning (MR), total tissue irradiation (CT), and scanning time restrictions. Even when time is not an issue, most scanners are by design incapable of sampling with high resolution in the out-of-plane direction, producing anisotropic “brick-like” voxels.

The non-uniform sampling of an object or a patient can create certain problems. The inadequate resolution in the Z direction implies that small or thin structures will not be properly sampled, making it difficult to capture them during surface reconstruction and object segmentation. One way to address this problem is to scan the same object from multiple directions, with the hope that the small structures will be adequately sampled in one of the scans. Generating several scans of the same object then raises the question of how to properly combine the information contained in these multiple datasets. Simply merging the individual scans does not necessarily assemble enough samples to produce a high resolution volumetric model. Normally a technique for filling in between samples is needed.

We have previously developed a framework for extracting 3D models from volume datasets [23] based on level set methods [13]. In this paper we describe a method for deforming a level set model using velocity information derived from multiple volume datasets with non-uniform resolution in order to produce a single high-resolution 3D model. The method locally approximates the values of the multiple datasets by fitting a distance-weighted polynomial
using moving least-squares (MLS) [19, 8]. Directional 3D edge information that may be used during a level set segmentation process is readily derived from MLS.

The proposed method has several advantageous properties. Instead of merging all of the input volumes by global resampling (interpolation), we locally approximate the derivatives of the intensity values by MLS. This local versus global approach is feasible because the segmentation process is implemented with a deformable level set model that requires only edge information in a narrow band around the surface. Consequently the MLS calculation is only performed in a small region of the volume, rather than throughout the whole volume, making the computational cost proportional to the object surface area [25]. As opposed to many interpolation schemes the MLS method is stable with respect to noise and imperfect registrations [5]. Our implementation also allows for small intensity attenuation artifacts between the multiple scans thereby providing gain-correction. The distance-based weighting employed in our method ensures that the contributions from each scan is properly merged into the final result. If a slice of data from one scan is closer to a point of interest on the model, the information from this scan will contribute more heavily to determining the location of the point.

To the best of our knowledge there is no previous work on creating deformable models directly from multiple volume datasets. While there has been previous work on 3D level set segmentation and reconstruction[24, 10, 9, 20, 26], it has not been based on multiple volume datasets. However, 3D models have been generated from multiple range maps [22, 4, 25, 16], but the 2D nature of these approaches is significantly different from the 3D problem being addressed in this paper. The most relevant related projects involve merging multiple volumes to produce a single high-resolution volume dataset [21, 6], and extracting edge information from a single non-uniform volume [2]. Our work does not attempt to produce a high-resolution merging of the input data. Instead, our contribution stands apart from previous work because it deforms a model based on local edge information derived from multiple non-uniform volume datasets.

We have demonstrated the effectiveness of our approach on four multi-scan datasets. The first three examples are derived from single high resolution volume datasets that have been sub-sampled in the X, Y and Z directions respectively. Since the non-uniform scans are extracted from a single dataset they are therefore perfectly aligned. The first example is a volumetric laser scan reconstruction of a griffin model. The second example is a high resolution MR scan of a 12-day-old mouse embryo, which has already had its outer skin isolated with a previous segmentation process. The third example is a preprocessed high resolution CT scan of a teapot, that also only contains an outer surface. The final example consists of multiple MR scans of a zucchini that have been imperfectly aligned by hand. The first three examples show that our method is able to perform level set segmentation from multiple non-uniform scans of an object, picking up and merging features only found in one of the scans. The final example demonstrates that our method generates satisfactory results, even when there are misalignments in the registration.

The remainder of the paper has the following structure. In Section 2 we outline the details of our method, and in section 3 we present the results obtained with this method. We close with conclusions and an appendix describing the moving least-squares method.

2 Method Description

We formulate our approach to 3D reconstruction of geometric models from multiple non-uniform volumetric datasets within a level-set segmentation framework [23]. The level set models utilized within this framework are deformable implicit surfaces whose deformation is controlled by a speed function in the level set partial differential equation (PDE). The speed function describes the velocity at each point on the evolving surface in the direction of the local surface normal. All of the information needed to deform a surface is encapsulated in the speed function, providing a simple, unified computational framework. In this section we briefly describe our level set segmentation framework, review the fundamental level set PDE, and define speed functions that allow us to solve the multiple-data segmentation problem. The key to constructing suitable speed terms is 3D directional edge information derived from the multiple datasets. This problem is solved using a moving least-squares scheme that extracts edge information by locally fitting sampling points to high-order polynomials. This section concludes by outlining the overall algorithm of the method.

2.1 Level Set Segmentation Framework

Level set segmentation relies on a surface-fitting strategy that creates a new volume from the input data by solving a partial differential equation (PDE) with user-defined feature-extracting terms. Because the deformable models move using gradient descent, they seek local solutions, and therefore the results are strongly dependent on the starting position of the surface. Thus, level set deformations alone are not sufficient, they must be combined with powerful initialization techniques in order to produce successful segmentations. Our level set segmentation framework consists of a set of suitable pre-processing techniques for initialization, which are then followed by the selection and tuning of different feature-extracting terms in the level set algorithm, as seen in Figure 2 [23]. Once these terms are defined the level set deformation proceeds to produce the final result.

Each stage in this two-step process is equally important for generating a correct segmentation. A user must “mix-and-match” these operations in order to produce the desired result. The operators available for creating the initial model include high and low thresholding, flood-filling, as well as CSG and morphological (opening and closing) operators. These operators provide a rough initial estimate of the desired model. The level set surface deformation process then moves the model toward specific features in the data while balancing this movement with a regularizing smoothing term, in order to prevent the surface from fitting too closely to noise-corrupted data. The smoothing term utilizes a local mean curvature measure in order to remove regions of high curvature. The inclusion of such a regularization term is also a well known technique in most implicit snake algorithms. Concurrently, the level set model may be attracted to “Canny” edges [3], iso-surfaces and regions of maximum gradient magnitude in the input data.

2.2 The Level Set Method

The Level Set Method [13] is a mathematical tool for modeling surface deformations. A deformable (i.e. time-dependent) surface, \(S(t)\), is implicitly represented as an iso-surface of a time-varying scalar function \(\phi(x, t)\) embedded in 3D, i.e.,

\[
S(t) = \{ x(t) | \phi(x(t), t) = k \},
\]

where \(k \in \mathbb{R}\) is the iso-value, \(t \in \mathbb{R}^+\) is time, and \(x(t) \in \mathbb{R}^3\) is a point in space on the iso-surface. It might seem inefficient to implicitly represent a surface with a 3D scalar function; however the higher dimensionality of the representation provides one

\[\text{Our work uses the dynamic level set equation, which is more flexible than the corresponding stationary equation, } \phi(x) = k(t), \text{ see [18] for more details.}\]
of the major advantages of the LS method: the flexible handling of changes in the topological genus of the deformable surface. This implies that LS surfaces can easily represent complicated surface shapes that can form holes, split to form multiple objects, or merge with other objects to form a single structure. This is an important property when segmenting complex models with an unknown topological genus.

The fundamental level set equation of motion for \( \phi(x(t), t) \) is derived by differentiating Eq. (1) with respect to time \( t \), and applying the chain rule giving:

\[
\frac{\partial \phi}{\partial t} = - \nabla \phi \cdot \frac{dx}{dt} = \|\nabla \phi\| \mathcal{F}(x, n, \phi) \tag{2a}
\]

\[
\mathcal{F}(x, n, \phi) \equiv n \cdot \frac{dx}{dt}, \tag{2b}
\]

where \( dx/dt \) and \( n \equiv -\nabla \phi/\|\nabla \phi\| \) are the velocity and normal vectors at \( x \) on the surface. We assume a positive-inside/negative-outside sign convention for \( \phi(x, t) \), i.e. \( n \) points outward. Eq. (2b) introduces the speed function \( \mathcal{F} \), which is a user-defined scalar function that can depend on any number of variables including \( x \), \( n \), \( \phi \) and its derivatives evaluated at \( x \), as well as a variety of input data. \( \mathcal{F}() \) is a signed scalar function that defines the motion (i.e. speed) of the level set surface in the direction of the local normal \( n \) at \( x \).

A number of numerical techniques [13, 1] make the initial value problem of Eq. (2) computationally feasible. A complete discussion of the details of the level set method is beyond the scope of this paper. We instead refer the interested reader to [18, 12, 17]. However, we will briefly mention two of the most important techniques: the first is the so-called “up-wind scheme” which addresses the problem of overshooting when trying to integrate Eq. (2) in time by finite differences. Specifically the upwind scheme is used to compute first order partial derivatives by a single-sided finite difference which is up-wind with respect to the motion of the level set surface. The second important technique is related to the fact that one is typically only interested in a single solution to Eq. (2), say the \( k = 0 \) level set. This implies that the evaluation of \( \phi \) is important only in the vicinity of a particular level set. This forms the basis for “narrow-band” schemes [1, 25, 14] that solve Eq. (2) in a narrow band of voxels near the surface. The “up-wind scheme” makes the level set method numerically robust, and the “narrow-band scheme” makes its computational complexity proportional to the level set’s surface area rather than the size of the volume in which it is embedded.

### 2.3 Level Set Speed Function for Segmentation

Many different speed functions have been proposed over the years for segmentation of a single volume dataset [24, 10, 9, 20]. Typically such speed functions consist of a (3D) image-based feature attraction term and a smoothing term which serves as a regularization term that lowers the curvature and suppresses noise in the input data. From computer vision it is well known that features, i.e. significant changes in the intensity function, are conveniently described by an edge-detector [7]. There exists a very large body of work devoted to the problem of designing optimal edge detectors for 2D images [11, 3], most of which are readily generalized to 3D. For the work presented in this paper we found it convenient to use speed functions with the 3D directional edge term

\[
\mathcal{F}_{edge}(x, n, \phi) = \alpha n \cdot \nabla \|\nabla V_0\| \tag{3}
\]

where \( \alpha \) is a scaling factor for the image-based feature attraction term \( \nabla \|\nabla V_0\| \). \( V_0 \) symbolizes some global uniform merging of the multiple non-uniform input volumes. This feature term is effectively a 3D directional edge-detector of \( V_0 \). However there are two problems associated with using this speed function exclusively. The first is related to the fact that we cannot expect to compute reliable 3D directional edge information in all regions of space simply because of the nature of the non-uniform multiple volumes that serves as input for our segmentation process. In other words \( V_0 \) cannot be interpolated reliably in regions of space where there are no nearby sampling points. Hence the level set surface will not experience any image-based forces in these regions. In other words the surface fitting is an ill-posed problem in regions of space with no image-based information. The solution is to use a regularization term that imposes constraints on the mean curvature of the deforming level-set surface. We include the following smoothing term in the speed function in order to smooth the regions where no edge information exists as well as suppress noise in the remaining regions thereby preventing excessive aliasing:

\[
\mathcal{F}_{smooth}(x, n, \phi) = \beta \nabla \cdot [\nabla \phi/\|\nabla \phi\|] \tag{4}
\]

where \( \beta \) is a scaling factor for the mean curvature, \( \nabla \cdot [\nabla \phi/\|\nabla \phi\|] \), on the level set surface defined from \( \phi \).

However, one problem remains. Normally the feature attraction term, \( \nabla \|\nabla V_0\| \), creates only a narrow range of influence. In other words, this feature attraction term will only reliably move the portion of the level set surface that is in close proximity to the actual edges in \( V_0 \). Thus, a good initialization of the level set surface is needed before solving Eq. (3). A reasonable initialization of the level set surface may be obtained by computing the CSG union of the multiple input volumes, which are first tri-linearly resampled to give a uniform sampling. However, if the input volumes are strongly non-uniform their union produces a poor initial model. This occurs when the input volumes are severely undersampled in one or more directions, as seen in Figure 3. Consequently we attract the CSG union surface to the distance transform of the Canny edges [3] computed from \( V_0 \). Canny edges are non-directional edges defined from the zero-crossing of the second derivative of the image...
in the direction of the local normal. In 3D this is
\[
\frac{\partial^2}{\partial n^2} V_g = 0 \tag{5}
\]
where \( n_g \equiv \nabla V_g / \| \nabla V_g \| \) is the local normal vector of \( V_g \). Using the expressions \( \partial / \partial n = n_g \cdot \nabla \) we can rewrite Eq. (5) as
\[
\frac{\partial^2}{\partial n^2} V_g = n_g \cdot \nabla [n_g \cdot \nabla V_g] = n_g \cdot \nabla \| \nabla V_g \|. \tag{6}
\]
This expression highlights the relationship between the Canny edge detector and the 3D directional edge detector defined in Eq. (3).

The next section focuses on the methods needed to reliably compute the vectors \( n_g \) and \( \| \nabla V_g \| \). In preparation, the latter may be explicitly expressed in terms of the derivatives of the merged volume \( V_g \)
\[
\nabla \| \nabla V_g \| = \frac{\nabla V_g \hat{H} V_g}{\| \nabla V_g \|} \tag{7}
\]
where we have defined the gradient vector and the Hessian matrix,
\[
\nabla V_g = \left( \frac{\partial V_g}{\partial x}, \frac{\partial V_g}{\partial y}, \frac{\partial V_g}{\partial z} \right) \tag{8a}
\]
\[
\hat{H} V_g = \begin{pmatrix}
\frac{\partial^2 V_g}{\partial x^2} & \frac{\partial^2 V_g}{\partial y^2} & \frac{\partial^2 V_g}{\partial z^2} \\
\frac{\partial^2 V_g}{\partial x \partial y} & \frac{\partial^2 V_g}{\partial y \partial x} & \frac{\partial^2 V_g}{\partial x \partial z} \\
\frac{\partial^2 V_g}{\partial x \partial z} & \frac{\partial^2 V_g}{\partial z \partial x} & \frac{\partial^2 V_g}{\partial z^2}
\end{pmatrix}. \tag{8b}
\]

Thus, in closing we note that the level set propagation needed for segmentation only needs information about the first and second order partial derivatives of the input volumes, not the interpolated intensity values themselves.

### 2.4 Computing Partial Derivatives

As outlined above the speed function \( \mathcal{F} \) in the level-set equation, Eq. (2), is based on edge information derived from the input volumes. This requires estimating first and second order partial derivatives from the multiple non-uniform input volumes. We do this by means of moving least-squares (MLS), which is an effective and well-established numerical technique for computing derivatives of functions whose values are known only on irregularly spaced points [19, 8, 5].

Let us assume we are given the input volumes \( \hat{V}_d, d = 1, 2, \ldots, D \) which are volumetric samplings of an object on the non-uniform grids \( \{ \hat{x}_d \} \). We shall also assume that the local coordinate frames of \( \{ \hat{x}_d \} \) are scaled, rotated and translated with respect to each other. Hence, we define a world coordinate frame (typically one of the local frames) in which we solve the level set equation. Now, let us define the world sampling points \( \{ x_d \} \) as
\[
x_d \equiv \mathbf{T}^{(d)}[\hat{x}_d] \tag{9}
\]
where \( \mathbf{T}^{(d)} \) is the coordinate transformation from a local frame \( d \) to the world frame. Next we locally approximate the intensity values from the input volumes \( \hat{V}_d \) with a 3D polynomial expansion. Thus, we define the \( N \)-order polynomials
\[
V_N^{(d)}(x) = C^{(d)}_{000} + \sum_{i+j+k=1}^{N} C^{(d)}_{ijk} x^i y^j z^k, \quad d = 1, 2, \ldots, D \tag{10}
\]
where the \( C \) coefficients are unknown. Note that these local approximations to the intensity values share coefficients \( C^{(d)}_{ijk} \) of order higher than zero, i.e. all of the functions \( V_N^{(d)} \), \( d = 1, 2, \ldots, D \) have the same edges. The fact that the zero-order term in Eq. (10) is input volume dependent means we allow for local constant offsets between the input volumes \( \hat{V}_d \). This effectively provides built-in gain-correction in the scheme, since it can handle small intensity attenuation artifacts between the multiple scans. The details of deriving a set of linear equations for the coefficients \( C \) by means of the moving least-squares method is described in the Appendix. The resulting system of linear equations can be solved using standard techniques from numerical analysis. Summarizing the results from the Appendix, Eq. (18a) and Eq. (18b) can be conveniently expressed as
\[
\sum_{q} A_p q_c q = b_p \tag{11}
\]
where \( A \) is a diagonal matrix, and \( b, c \) are vectors. In this equation we have also introduced the compact index notations \( p \equiv (i, j, k, r) \) and \( q \equiv (l, m, n, s) \) defined as
\[
p \in \left\{ i, j, k, r \in \mathbb{N}^+ \mid i = j = k = 0, 1 \leq r \leq D \right\}
\cup \left\{ i, j, k, r \in \mathbb{N}^+ \mid 1 \leq i+j+k \leq N, \ r = 0 \right\} \tag{12a}
\]
\[
q \in \left\{ l, m, n, s \in \mathbb{N}^+ \mid l = m = n = 0, 1 \leq s \leq D \right\}
\cup \left\{ l, m, n, s \in \mathbb{N}^+ \mid 1 \leq l+m+n \leq N, \ s = 0 \right\}. \tag{12b}
\]

The diagonal matrix \( A \), and the vectors \( b, c \) in Eq. (11) are defined as
\[
A_p q \equiv \sum_{d} (\delta_{r,d} + \delta_{r,0}) (\delta_{s,d} + \delta_{s,0}) \sum_{x_d} w_d(x_d-x_0) \times (x_d - x_0)^i (y_d - y_0)^j (z_d - z_0)^k \tag{13a}
\]
\[
\times (x_d - x_0)^i (y_d - y_0)^j (z_d - z_0)^k \tag{13b}
\]
\[
b_p \equiv \sum_{d} (\delta_{r,d} + \delta_{r,0}) w_d(x_d-x_0) \hat{V}_d(x_d) \tag{13c}
\]
Next the matrix equation \( A c = b \) must be solved for the vector \( c \) of dimension \( (N + 1)^3 + D - 1 \), where \( N \) is the order of the expansion in Eq. (10) and \( D \) is the number of non-uniform input volumes. As is well known for many moving least-squares problems it is possible for the condition number of the matrix \( A \) to become very large.
Any matrix is singular if its condition number is infinite and can be defined as ill-conditioned if the reciprocal of its condition number approaches the computer's floating-point precision. This can occur if the problem is over-determined (number of sampling points, \( x_d \) greater than number of coefficients \( C \)) and under-determined (ambiguous combinations of the coefficients \( C \) work equally well or equally bad). To avoid such numerical problems, a singular value decomposition (SVD) linear equation solver is recommended for use in combination with the moving least-squares method. The SVD solver identifies equations in the matrix \( A \) that are, within a specified tolerance, redundant (i.e., linear combinations of the remaining equations) and eliminates them thereby improving the condition number of the matrix. We refer the reader to reference [15] for a helpful discussion of SVD pertinent to linear least-squares problems.

Once we have the expansion coefficients \( c \) we can readily express the Hessian matrix and the gradient vector of the combined input volumes as

\[
\nabla V = (C^{(0)}_{100}, C^{(0)}_{010}, C^{(0)}_{001}) \\
HV = \begin{pmatrix}
2C^{(0)}_{200} & C^{(0)}_{110} & C^{(0)}_{101} \\
C^{(0)}_{120} & 2C^{(0)}_{200} & C^{(0)}_{011} \\
C^{(0)}_{110} & C^{(0)}_{101} & 2C^{(0)}_{002}
\end{pmatrix}
\]

(14a)

(14b)

evaluated at the moving expansion point \( x_0 \). This in turn is used in Eq. (7) to compute the edge information needed to drive the level set surface.

### 2.5 Algorithm Overview

The level set segmentation algorithm used in this paper is outlined below. Algorithm 2.1 describes the main steps of our approach. The initialization routine, Algorithm 2.2, is called for all of the multiple non-uniform input volumes, \( V_d \). Each non-uniform input dataset is uniformly resampled using tri-linear interpolation. Edge information and the union, \( V_0 \), of the \( V_d \)'s is then computed. Algorithm 2.2 calculates Canny and 3D directional edge information using moving least-squares in a narrow band in each of the resampled input volumes, \( V_d \). This in turn is used in Eq. (7) to compute the edge information needed to drive the level set surface.

Algorithm 2.1: **M**ain\( (V_1, \ldots, V_D) \)

**comment:** \( V_1, \ldots, V_D \) are non-uniform samplings of object \( V \)

**global** \( V_{edge}, V_{edge} \)

\( V_0 \leftarrow \) uniform sampling of empty space

for \( d = 1 \) to \( D \)

\( V_d \leftarrow V_0 \cup \text{INITIALIZATION}(V_d) \)

\( V_F \leftarrow \nabla [\text{distance transform}[\text{zero-crossing}[V_{edge}]]] \)

\( V_0 \leftarrow \text{SOLVELEVELSETEQ}(V_0, V_F, \alpha, 0) \)

\( V_0 \leftarrow \text{SOLVELEVELSETEQ}(V_0, V_{edge}, \alpha, \beta) \)

return (Marching Cubes mesh of \( V_0 \))

### 3 Results

We have applied our segmentation method to several multi-scan non-uniform datasets to produce high resolution level set models. The parameters used for these segmentations are listed in Table 1. \( \alpha \) and \( \beta \) are weights that the user adjusts to balance attraction to edges with curvature-based smoothing during the level set deformation process.

Table 1: Maximum in-plane to out-of-plane sampling ratios of non-uniform input datasets, and parameters for the two level set speed terms defined in Eq. (3) and Eq. (4).

<table>
<thead>
<tr>
<th>Model</th>
<th>Origin</th>
<th>Ratio</th>
<th>( \alpha )</th>
<th>( \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Griffin</td>
<td>Laser scan</td>
<td>10:1</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Mouse</td>
<td>MR scan</td>
<td>10:1</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Teapot</td>
<td>CT scan</td>
<td>9:1</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Zucchini</td>
<td>MR scan</td>
<td>10:1</td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### 3.1 Griffin Dataset

The griffin dataset was created with a volumetric laser scan reconstruction algorithm [4]. This algorithm creates a high resolution volumetric representation of an object by merging multiple depth maps produced via a laser scan. The original griffin dataset has a resolution of \( 312 \times 294 \times 144 \). We have extracted three non-uniform datasets from this high resolution representation by copying every sixth plane of data in the \( X \) and \( Y \) directions and every tenth plane of data in the \( Z \) direction. The three derived non-uniform griffin datasets have the following resolution: \( 52 \times 294 \times 144, 312 \times 30 \times 144 \) and \( 312 \times 294 \times 24 \). Iso-surfaces have been extracted from these datasets, appropriately scaled in the low resolution direction, and are presented in the first three images in Figure 4. Each low resolution scan inadequately captures some important geometric feature of the griffin. In the first scan the wing on the right contains numerous holes. In the second scan the horns on the head are not properly represented, and in the third image the wing on the left contains significant notches. Additionally, all three scans are severely...
Figure 4: Three non-uniform samplings of a high resolution laser scan reconstruction of a griffin figurine, followed by a level set model derived from the first three scans. Each input model is missing a particular feature - first: holes in right wing, second: jagged edges of both wings, third: right horn not connected to wing (as it should be). The level set reconstruction contains all of these missing features.

Figure 5: Three non-uniform samplings of a high resolution MR scan of a mouse embryo, followed by a level set model derived from the first three scans.

Figure 6: Three non-uniform samplings of a high resolution CT scan of a teapot, followed by a level set model derived from the first three scans.
aliased. We have performed two reconstructions from the under-sampled non-uniform scans. In Figure 1 a reconstruction produced from just the first two scans is presented. The final image in Figure 4 presents the results of applying our segmentation method to all three low resolution scans. The method produces high resolution (312 × 294 × 144) level set models that contain all of the features mentioned above and do not exhibit the aliasing seen in the low resolution scans. Adding the third scan provides more information around the edges of the wings. It should also be noted that the wing on the right is connected to the right horn in the initial high resolution dataset.

3.2 Mouse Embryo Dataset

The first three scans in Figure 5 are derived from a high resolution MR scan of a mouse embryo. They are subsampled versions of a 256 × 128 × 128 volume dataset, and have the following resolutions: 26 × 128 × 128, 256 × 16 × 128 and 256 × 128 × 13. The last image in Figure 5 presents the result produced by our multi-scan segmentation method. The information in the first three scans has been successfully used to create a level set model of the embryo with a resolution of 256 × 128 × 130. The finer features of the mouse embryo, namely its hands and feet, have been reconstructed.

3.3 Teapot Dataset

The first three scans in Figure 6 are derived from a CT scan of a teapot. They are subsampled versions of a 244 × 218 × 188 volume dataset, and have the following resolutions: 28 × 218 × 188, 244 × 25 × 188 and 244 × 218 × 21. The last image in Figure 6 presents the result produced by our multi-scan segmentation method. The information in the first three scans has been successfully used to create a level set model of the original teapot with a resolution of 244 × 218 × 189. The finer features of the teapot, namely the handle and the spout, have been reconstructed.

3.4 Zucchini Dataset

The zucchini dataset consists of three individual MRI scans of an actual zucchini. The separate scans have been registered manually and are presented on the left side of Figure 7, each with a different color to demonstrate their imperfect alignment. The resolutions of the individual scans are 28 × 218 × 188, 244 × 25 × 188 and 244 × 218 × 21. This image highlights the rough alignment of the scans. The right side of Figure 7 presents the result of our level set segmentation. It demonstrates that our approach is able to extract a reasonable model from multiple datasets that are imperfectly aligned.

4 Conclusions

In this paper we have proposed a method that uses multiple volume datasets with non-uniform resolution acquired in different local coordinate frames, but with known relative transformations, to deform a level set model on a uniform grid. As described in section 2.4, the contribution from each of the datasets to the velocity of the evolving level set model is weighted according to the quality (resolution) of the dataset near the propagating front. We obtain this result by employing a moving least-squares (MLS) method. Our method only performs the MLS calculation in the neighborhood of the propagating front and thus has $O(N^2)$ computational complexity. Additionally, it is stable with respect to noise, imperfect registration and abrupt changes in the data, it provides gain-correction, and employs a distance-based weighting to ensure that the contributions from each scan are properly merged into the final result. We have demonstrated the effectiveness of our approach on four multi-scan datasets, a griffin laser scan reconstruction, a CT scan of a teapot and MR scans of a mouse embryo and a zucchini. As future work we plan to explore other integration techniques in the moving least-squares method that might allow us to reduce the amount of smoothing needed during segmentation.

5 Acknowledgment

We would like to thank Dr. Alan Barr for his technical assistance, Dr. Jason Wood of the University of Leeds for creating several useful visualization tools, and Ms. Cici Koenig for helping us with our figures. We would also like to thank Dr. John Wood of Childrens Hospital Los Angeles for providing the zucchini dataset, Dr. J. Michael Tyszka of the Division of Diagnostic Radiology at the City of Hope National Medical Center for providing the teapot dataset, Dr. Russell Jacobs of the Caltech Biological Imaging Center for providing the mouse embryo dataset, and the Caltech Multi-Res Modeling Group and the Stanford Computer Graphics Laboratory for providing the griffin dataset. This work was supported by National Science Foundation grants #ASC-89-20219, #ACI-9982273 and #ACI-0089915, and the National Institute on Drug Abuse and the National Institute of Mental Health, as part of the Human Brain Project.

References


Session 2
Latest Developments in Level Set Segmentation:
Piecewise smooth image model, Statistical shape models and Spatial dependency

Mikael Rousson
Siemens Corporate Research
(Princeton, NJ)

Outline

1. The level set representation for image segmentation
   a) Edge-based and region-based formulation
   b) A new image term for segmenting piecewise smooth images

2. Modeling and introducing prior shape knowledge

3. Coupled segmentations: Ordered spatial dependency
The Level Set Representation

- A contour is represented as the zero-crossing of a level set function (Osher & Sethian)
  \[ \phi : \Omega \rightarrow \mathbb{R} : \]
  \[ C = \{ x \in \Omega | \phi(x) = 0 \}. \]

- This representation has several advantages compared to explicit representations:
  - Independent of any parameterization,
  - Surfaces of any dimension can be represented,
  - Topological changes are possible,
  - Geometrical quantities like normal vector or curvature can be easily expressed.

Level Set Contour Evolution

- Any contour evolution can be expressed as a level set evolution:
  \[ \frac{\partial C}{\partial t} = F \nabla \phi \quad \longrightarrow \quad \frac{\partial \phi}{\partial t} = -|\nabla \phi| F \]

- Alternatively, similar flows can be obtained from the minimization of an energy that depends on the level set representation of \( C \):
  \[ E(\phi) = E_{\text{image}}(\phi) + \alpha E_{\text{prior}}(\phi) \quad \longrightarrow \quad \frac{\partial \phi}{\partial t} = \frac{\partial E}{\partial \phi} \]

  **Image term**
  - Edge-based: favors contours overlapping with high image gradient,
  - Region-based: separates regions that are piecewise constant, piecewise smooth, or regions with specific intensity distributions.

  **Prior term**
  - Regularization: favors smooth contours,
  - Shape prior: constrains the contour to a shape family learned from known solutions.
Geometric Quantities & Common Terms

- **Geometric quantities**
  - Normal vector: \( \mathbf{N} = \nabla \phi / |\nabla \phi| \)
  - Curvature: \( \kappa = \nabla \cdot (\nabla \phi / |\nabla \phi|) \)

- **Common terms**
  - Let \( H \) be Heaviside function and \( \delta \) its derivative (Dirac)
  - Integral of a function along the curve:
    \[
    E_{\text{norm}} = \int_{\Omega} |\nabla H(\phi)| f(x, \mathcal{C}) \, dx = \int_{\Omega} \delta(\phi) |\nabla \phi| f(x, \mathcal{C}) \, dx.
    \]
  - Domain integrals of a function:
    \[
    E_{\text{in}} = \int_{\Omega} H(\phi(x)) f(x, \mathcal{C}) \, dx, \quad E_{\text{out}} = \int_{\Omega} (1 - H(\phi(x))) f(x, \mathcal{C}) \, dx.
    \]

Edge-Based Image Term

- **Geodesic Active Contours** (Caselles et al., Kichenassamy et al. 1995)
  - Fit a smooth contour to high gradients of the image
  - Edge detection function:
    \[
    g\left(\frac{|\nabla I|}{|\nabla I|^n}\right) = \frac{1}{1 + |\nabla I|^n}, \quad n = 1 \text{ or } 2
    \]
  - Estimate the contour with minimal length according to this metric:
    \[
    E(\mathcal{C}) = \int_0^1 g(|\nabla I(\mathcal{C}(s))|) |\mathcal{C}'(s)| ds = \int_0^1 g(|\nabla I(\mathcal{C}(s))|) ds
    \]
  - Corresponding contour and level set evolutions:
    \[
    \frac{\partial \mathcal{C}}{\partial t} = g() \kappa \mathbf{N} - (\nabla g() \cdot \mathbf{N}) \mathbf{N} \quad \Rightarrow \quad \frac{\partial \phi}{\partial t} = g() |\nabla \phi| \text{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right) + \nabla g() \cdot \nabla \phi
    \]
Piecewise Smooth Model (Mumford-Shah 1985)

- Model the image as piecewise smooth regions (MS functional or weak membrane model)
  - Estimate simultaneously the boundary \( \partial \) between these regions and the ideal piecewise smooth image \( u \) by minimizing:
    \[
    E_{MMS}(u, \partial) = \int_\Omega |u(x) - I(x)|^2 \, dx + \mu \int_{\Omega \setminus \partial} |\nabla u(x)|^2 \, dx + \nu |\partial|
    \]

- Level set formulation proposed by Chan-Vese and Tsai et al. 2001
  - A system of coupled partial differential equations is obtained for minimization (2 regions):
    \[
    \begin{align*}
    u^+ &= I + \mu \Delta u^+ \text{ on } \{ \phi > 0 \}, \quad \frac{\partial u^+}{\partial N} = 0 \text{ on } \{ \phi = 0 \} \cup \partial;
    u^- &= I + \mu \Delta u^- \text{ on } \{ \phi < 0 \}, \quad \frac{\partial u^-}{\partial N} = 0 \text{ on } \{ \phi = 0 \} \cup \partial;
    \frac{\partial \phi}{\partial t} &= \delta(\phi) \left( \nu \div \left( \frac{\nabla \phi}{|\nabla \phi|} \right) \right) - |u^+ - I|^2 + |u^- - I|^2 - \mu |\nabla u^+|^2 + \mu |\nabla u^-|^2
    \end{align*}
    \]

Piecewise Constant Model (Chan-Vese 2001)

- Approximation of the image by a piecewise constant model (cartoon model)
  - The boundary and each region representation are estimated by minimizing:
    \[
    E_{PCS}(u, \phi) = \sum_{i=1}^N \int_{\Omega_i} |c_i - I(x)|^2 \, dx + \nu |\partial|
    \]

- Level set formulation introduced by Chan and Vese (2 regions):
  \[
  E_{PCS}(c^+, c^-, \phi) = \int_{\Omega^+} |c^+ - I(x)|^2 H(\phi(x)) \, dx + \int_{\Omega^-} |c^- - I(x)|^2 (1 - H(\phi(x))) \, dx
  + \nu \int_\Omega |\nabla H(\phi(x))| \, dx
  \]

- Minimization:
  \[
  \frac{\partial \phi}{\partial t} = \delta(\phi) \left( \nu \div \left( \frac{\nabla \phi}{|\nabla \phi|} \right) \right) - |c^+ - I|^2 + |c^- - I|^2
  \]
  with
  \[
  \begin{align*}
  c^+ &= \int_{\Omega^+} I \, H(\phi) / \int_{\Omega^+} H(\phi)
  c^- &= \int_{\Omega^-} I (1 - H(\phi)) / \int_{\Omega^-} (1 - H(\phi))
  \end{align*}
  \]
Statistical Region Model (Zhu-Yuille 1995, Paragios-Deriche 2001)

- Pixel's values inside each region are assumed to be independent and identically distributed
  - Maximum likelihood criterion, equivalent to minimize:
    \[ E(C) = - \sum_{i=1}^{n} \int_{\Omega_i} \log p_i(I(x)) \, dx + \nu |C| \]
  - The region distribution can be either estimated a priori or added as unknown parameters
  - Gaussian model:
    \[ E(\phi; \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \int_{\Omega_i} H(\phi) \left( \frac{1}{2 \pi \sigma_1^2} \right)^{n \phi} \frac{1}{2 \sigma_2^2} \left( \frac{I(x) - \mu_1}{\sigma_1^2} \right)^2 + \int_{\Omega_j} \left( \frac{1 - H(\phi)}{2 \sigma_2^2} \right)^{n \phi} \frac{1}{2 \sigma_2^2} \left( \frac{I(x) - \mu_2}{\sigma_2^2} \right)^2 \, dx + \nu |C| \]
  - Corresponding evolution:
    \[ \frac{\partial \phi}{\partial t} = \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) - \frac{(I - \mu_1)^2}{2 \sigma_1^2} + \frac{(I - \mu_2)^2}{2 \sigma_2^2} - \log \frac{\sigma_1^2}{\sigma_2^2} \right) \]

Application to Color, Texture and DT Image Segmentation

- **Color images** (Zhu and Yuille, Chan and Vese, Rousson and Deriche): Assumption of homogeneous color distribution inside each region.

- **Texture images** (Rousson, Brox et al): Extraction of texture features based on the nonlinear structure tensor.

- **Diffusion tensor images** (Lenglet et al 05): Riemannian metric on sets of covariance matrices.
Efficient Piecewise Smooth Model

- **Piecewise constant**: Each region is approximated by a mean intensity, equivalent to minimizing the intensity variance inside each region,
- **Region distributions**: More general than piecewise constant (Gaussian), but does not incorporate spatial correlation,
- **Piecewise smooth**: Most general, can deal with intensity bias, but many local minima and computationally expensive.

Recent publications propose an efficient implementation of the piecewise smooth model.

Piecewise smooth model

- The Mumford-Shah piecewise smooth model:
  \[ E_{MS}(u, C) = \int_{\Omega} (1 - u)^2 \, dx + \mu \int_{\Omega \setminus C} |\nabla u|^2 \, dx + \nu |C| \]
  
- We propose to replace the second term by an isotropic smoothing of the intensity inside each region:
  \[ u_{\sigma}(x, y, \xi) = \frac{\int_{\mathbb{R}^2} g_{\sigma}(x-y)I(y) \, dy}{\int_{\mathbb{R}^2} g_{\sigma}(x-y) \, dy}, \quad \text{where} \quad g_{\sigma}(\nu) = \frac{1}{\sqrt{2\pi\nu}} \exp\left(\frac{-\nu^2}{2\nu^2}\right). \]
  
- These smoothed images depend only on the boundary.
  
- Therefore, the corresponding new cost function depends only on \( C \):
  \[ E'(C) = \int_{\Omega} (1 - u_{\sigma}(C))^2 \, dx + \nu |C| - \sum_i \int_{\Omega_i} (1 - u_{\sigma}(C_i))^2 \, dx + \nu |C| \]
Piecewise smooth model

- As this energy includes nested domain integrals, we use the shape derivative theory for derivation.
- Level set evolution (2 regions):

  \[
  \frac{\partial \phi}{\partial t} = |\nabla \phi| \left( \left( I - u_\phi(\phi) \right)^2 - \left( I - u_\phi(\phi) \right)^2 - q_{ii}(\phi) + q_{ij}^T(\phi) + \nu \kappa \right)
  \]

  with

  \[
  \begin{align*}
  q_{ii}(\phi) &= I \left( g_{\sigma \tau} \frac{2 H(I - u_\phi^+) \sigma}{q_{\sigma \tau} + H} \right) - g_{\sigma \tau} \frac{2 H(I - u_\phi^+) \nu_{\sigma \tau}}{q_{\sigma \tau} + H} \\
  q_{ij}^T(\phi) &= I \left( g_{\sigma \tau} \frac{(1 - H)(I - u_\phi)}{g_{\sigma \tau} (1 - H)} \right) + g_{\sigma \tau} \frac{(1 - H)(I - u_\phi \nu_{\sigma \tau})}{g_{\sigma \tau} (1 - H)}
  \end{align*}
  \]

- Alternate optimization:
  1. Estimate the smooth images
  2. Evolve the contour (level set) with a PDE minimizing the energy

→ Since the smooth image can be obtained with recursive Gaussian filter (linear complexity), the overall complexity is of the same order as the piecewise constant model!
Results

- Liver segmentation in MR images

More Results

- Comparison with Piecewise-Constant model

Piecewise Constant Model    Efficient Piecewise Smooth Model
More Results

- Robustness to high spatial intensity bias

![Original Image](image1.png) ![Image with Synthetic Bias](image2.png)

Outline

1. The level set representation for image segmentation
   a) Segmentation of gray, color, texture and diffusion images
   b) A new image term for segmenting piecewise smooth images

2. Modeling and introducing prior shape knowledge

3. Coupled segmentations: Ordered spatial dependency
Prior Shape Knowledge

- In many applications, the shape of the object to be recovered remains “consistent” from one image to another (ex: organs in medical imaging, people silhouette, ...)

- A shape space or distribution can be learned from a set of training data

- Two steps:
  1) Learning the shape model
  2) Constraining the segmentation process

PCA LS Shape Model (Leventon et al. 2001)

- Let \( \{C_1, \ldots, C_n\} \) be a set of training shapes rigidly registered and \( \{\phi_1, \ldots, \phi_n\} \) their distance transforms
- Leventon et al. proposed to apply a PCA on \( \{\phi_1, \ldots, \phi_n\} \)
- A new level set of the same class can expressed as
  \[ \phi = \phi_s + \sum_{j=1}^{m} \lambda_j U_j, \]
  - \( U_j \) stands for the eigenmodes,
  - The shape is represented by \( \lambda \)

- Assumption of Gaussian distribution:
  \[ p(\lambda) = \frac{1}{\sqrt{(2\pi)^k |\Sigma_k|}} \exp \left( -\frac{1}{2} \lambda^T \Sigma_k^{-1} \lambda \right) \]
PCA LS Shape Model (Leventon et al. 2001)

- Integration of the prior in the surface evolution by alternating two steps
  - Bayesian estimation of the best shape given the model, the current level set, and the image:
    \[ \phi^\text{MAP} = \arg\max_{\phi} P(\phi^*|\phi, \nabla I) \]
  - Constrained surface evolution:
    \[ \phi(t+1) = \phi(t) - \lambda_1 \frac{\partial E_{\text{data}}(\phi)}{\partial \phi} + \lambda_2 (\phi^*(t) - \phi(t)) \]

\[ \rightarrow \]
Several issues:
- Linear combinations of level sets
- Not a minimization of a single cost function

Implicit Active Shape Models (Rousson et al. 2004)

- PCA on level set representation as Leventon et al. but minimization of a single cost function:
  \[ E(\phi, \lambda, \mathcal{A}) = E_{\text{data}}(\phi, \mathcal{A}) + \alpha E_{\text{prior}}(\phi, \lambda) \]

- \( E_{\text{prior}}(\phi, \lambda) \) is defined as the distance between \( \phi \) and its projection to the PCA subspace:
  \[ E_{\text{prior}}(\phi, \lambda) = \int_{\Omega} \delta(\phi) \left( \phi(x) - \left( \phi_s(x) \mid \sum_{j=1}^{m} \lambda_j U_j(x) \right) \right)^2 dx \]

- Minimization
  - Alternate gradient descents w.r.t. \( \phi \) and the transformation \( \mathcal{A} \)
  - \( \lambda \) is updated by solving a simple linear system
Weak constraint to a PCA shape model

Efficient Kernel Density Estimation of Shape Prior

• Principal issues with related works:
  – Dimensionality (proportional to the image size)
  – Accuracy of the shape modeling (Gaussian or Uniform distrib.)

• Tsai et al. (2003) proposed to restrict new shapes to the space spanned by the training shapes

• Each training shape $\phi_i$ can be represented by its corresponding shape vector $\lambda_i$ in the PCA subspace

• The goal is to infer the distribution $p(\lambda|\{\lambda_i\})$ from these shapes
An Efficient Nonparametric Statistical Shape Model

• Tsai et al considered a uniform distribution
• We consider a kernel density estimation:

\[ \nu(\lambda | \lambda_i) = \frac{1}{N\sigma} \sum_{i=1}^{N} K\left( \frac{\lambda - \lambda_i}{\sigma} \right), \]
where \( K(u) = \frac{1}{\sqrt{2\pi}} \exp \left( \frac{-u^2}{2} \right). \)

• The nonparametric estimation has two advantages:
  – Better approximates the true distribution for large sample size
  – Favors shapes close to the samples vectors \( \lambda_i \) (signed distance functions)
→ In the limit of infinite sample size, the distribution inferred by the kernel density estimator converges towards a distribution on the manifold of signed distance functions

Nonparametric Density Estimate

Gaussian Density

Kernel Density
Energy Formulation & Minimization

• Applying the negative logarithm, the most probable segmentation is obtained by minimizing:

\[ E(\lambda, A) = - \log p(I|\lambda, A) - \log p(\lambda|\lambda_i) \]

• A coupled gradient descent is employed for its minimization.

• Implementation details:
  – The mean shape of the training set is used as initialization ($\lambda = 0$).
  – The Dirac function appears as factor of the gradient descent equations: A narrow-band implementation is possible (this is not an approximation!),
  – Standard finite difference scheme are used (the gradient descent is faster and more stable than a direct optimization on the level set).

Cardiac Ultrasound Segmentation without Shape Prior

Provided by D. Cremers
Cardiac Ultrasound Segmentation with Uniform Shape Prior

Provided by D. Cremers

Cardiac Ultrasound Segmentation with Nonparametric Shape Prior

Provided by D. Cremers
Principal Component Analysis on Distance Transforms

Prostate Segmentation without Bladder
Prostate Segmentation without Bladder

Outline

1. The level set representation for image segmentation
   a) Segmentation of gray, color, texture and diffusion images
   b) A new image term for segmenting piecewise smooth images

2. Modeling and introducing prior shape knowledge

3. Coupled segmentations: Ordered spatial dependency
Multiple Organ Segmentation

- General framework for image segmentation using ordered spatial dependency.

- Given several structures to be extracted in an image, we learn the spatial dependency between them, as well as the “best” ordering.

LEARNING: Modeling spatial dependency between structures

- **Atlas**: Set of $N$ annotated images
- **Reference anchor structure**
- **Probability map of another structure w.r.t. the reference one**
- **Warping of other structures based on anchor structures registrations**
- **Shape-based registration**
SEGMENTATION: Integrating spatial prior in the segmentation process

1- Segmentation of anchor structure
2- Registration of anchor structure to reference
3- Apply $\psi^{-1}$ to probability maps of other structures
4- Use the registered probability map to help segment the secondary structure in the current image by providing:
   - Automatic initialization,
   - Location prior for the segmentation algorithm.

Learning on Brain MR Images

- We learn the spatial dependency on a set of training images where the skull, the ventricles, the caudate nucleus and the thalamus have be manually segmented.
Learning on Brain MR Images

- This gives us a set of probability maps for each structure given a set of anchor structures.
- Probability maps obtained for the optimal ordering:

Experiments

- The method provides an automatic initialization for all organs but the first one.
- Since the first structure is the skull, it can be segmented without user interaction.
  → Fully automatic algorithm using level set (implicit contours) evolutions.
Discussion

- Novel image segmentation framework that learns ordered spatial dependencies among structures of interest
- Hierarchical modeling and segmentation
- Brings automatic initializations
- Improves performance and robustness of any individual segmentation algorithm (not only level set)
- Still very preliminary results:
  → Requires more experiments and validations.
- Spatial dependency can be improved by modeling conditional shape variations

Conclusion

1. The level set representation for image segmentation
   a) Segmentation of gray, color, texture and diffusion images
   b) A new image term for segmenting piecewise smooth images

2. Modeling and introducing prior shape knowledge

3. Coupled segmentations: Ordered spatial dependency
Thank you!

References:

Efficient Segmentation of Piecewise Smooth Images

Jérôme Piovano¹, Mikael Rousson², and Théodore Papadopoulo¹

¹ Odyssée team, INRIA/ENPC/ENS, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis, France

² Siemens Corporate Research, 755 College Road East, Princeton, NJ 08540, USA
Mikael.Rousson@siemens.com

Abstract. We propose a fast and robust segmentation model for piecewise smooth images. Rather than modeling each region with global statistics, we introduce local statistics in an energy formulation. The shape gradient of this new functional gives a contour evolution controlled by local averaging of image intensities inside and outside the contour. To avoid the computational burden of a direct estimation, we express these terms as the result of convolutions. This makes an efficient implementation via recursive filters possible, and gives a complexity of the same order as methods based on global statistics. This approach leads to results similar to the general Mumford-Shah model but in a faster way, without solving a Poisson partial differential equation at each iteration. We apply it to synthetic and real data, and compare the results with the piecewise smooth and piecewise constant Mumford-Shah models.

1 Introduction

The extraction of piecewise smooth regions from an image is of great interest in different domains, and still remains a challenging task. For example, this is very useful in medical imaging where organs or structures of interest are often characterized by smooth intensity regions. This problem has been formulated as the minimization of an energy by Mumford and Shah in [9]:

\[ E_{MS}(u, \Gamma) = \mu \int_{\Omega} (u_0 - u)^2 \, dx + \int_{\Omega \setminus \Gamma} |\nabla u|^2 \, dx + \nu |\Gamma| \]  

where \( u \) is the piecewise smooth function, \( \Gamma \) the interface between smooth regions, and \( u_0 \) the original image. The interpretation of the three term is straightforward: the first one is the usual mean-square data term; the second one means that we want to extract smooth regions; the third one means that we want to extract regions with smooth boundaries.

The minimizer of this so-called Mumford-Shah functional gives a boundary that separates the image domain in smooth regions. A very interesting property of this approach is that it solves two common image-processing tasks simultaneously: image denoising and image segmentation. However, finding the minimizer

© Springer-Verlag Berlin Heidelberg 2007
is not straightforward and remains an issue. Being non-convex, the functional is most of time minimized using gradient descent techniques which are subject to local minima. For example, in [16][14], the optimization process alternates between the evolution of one or two level set functions [10][35] and the resolution of Poison partial differential equations. This process is computational expensive and requires a very good initialization to avoid being stuck into a local minima.

To relax this problem, one can consider a restriction of $E_{MS}$ to piecewise constant functions. Let $\Omega_i$ be the open subsets delimited by $\Gamma$, the piecewise constant Mumford-Shah functional writes:

$$E_{MS}^{0}(\Gamma) = \sum_i \int_{\Omega_i} (u_0 - \text{mean}_{\Omega_i}(u_0))^2 dx + \nu|\Gamma|. \quad (2)$$

This functional was shown in [9] to be a limit functional of (1) as $\mu \to 0$. A level set implementation of this functional known as the Chan-Vese model was proposed in [16]. While this simplified functional is easier to minimize, it also makes a very strong assumption on the image by assuming implicitly a Gaussian intensity distribution for each region $\Omega_i$ [12]. Other papers model this distribution with Gaussian mixtures [11,6] or with nonparametric distributions [8] but they all make the assumption of a global distribution over each region. In many real images, these global intensity models are not valid. This is often the case in medical imaging, especially in MR images where an intensity bias can be observed.

Several approaches are available to overcome the limitation of global techniques. One is to consider image gradients by fitting the contour to image discontinuities. This is generally referred to as edge-based methods, and it is the basis of the Geodesic Active Contours [2,7]. Edge-based methods are also well-known for their high sensitivity to noise and for the presence of local minima in the optimization [3]. Another alternative was briefly discussed in [15] where the function $u$ of the Mumford-Shah functional is restricted to a linear function of the spatial location $x$: $u(x) = a.x + b$. Even though this last one is promising, it is still restricted to very particular spatial distributions of the intensity.

In this paper, a general approach for extracting piecewise smooth regions is proposed. Instead of minimizing the distance between the intensity and the average intensity of the region like in (2), the distance between the intensity and local averaging inside the region is minimized. This gives a model able to approximate piecewise smooth functions like the original Mumford-Shah functional [1], but with a complexity closer to that of the piecewise constant model.

Section 2 explains in detail our model and how it can be linked to the Mumford-Shah model. The minimized energy as well as its derivative using the shape gradient are expressed. In Section 2.2, the level set method is used to compute the evolution of the interface, and each term of the derivative is expressed as the result of a convolution. The importance of the fast recursive filter is

---

1 A third functional was also introduced in the seminal work of Mumford and Shah [9]. This functional is the integral along $\Gamma$ of a *generalized Finsler metric* and leads indeed to the first geodesic active contour (before 2[7]).
briefly explained. In Section 3, some results on synthetic and real data are shown and compared with the piecewise smooth and piecewise constant Mumford-Shah Model.

2 Piecewise Smooth Approximation Through Gaussian Convolutions

The Mumford-Shah model approximates the image by a piecewise smooth function by penalizing the high gradient of an “ideal” cartoon image (u in (1)). The estimation of this image at the same time as the segmentation makes the functional difficult to minimize, and computationally expensive, as it’s the solution of a poisson equation.

Here we approach the problem differently by fixing the cartoon image to a smoothing of the image intensity inside each subset $\Omega_i$. for the 2 Dimensional case, the smooth function in $\Omega_i$ is then defined as,

$$
u_\sigma(x, \Omega_i) = \int_{\Omega_i} g_\sigma(x-y)u_0(y)dy / \int_{\Omega_i} g_\sigma(x-y)dy,$$

where $g_\sigma(v) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{v^2}{2\sigma^2}\right)$.

The denominator of this expression is a normalization factor which is important for voxels that are close to the border of $\Omega_i$, i.e. when the Gaussian kernel does not overlap completely with $\Omega_i$. This can be also interpreted as a local weighted averaging of the image intensity around the voxel $x$ inside the region $\Omega_i$ (Figure 1).

![Fig. 1. Importance of the denominator in the formulation of smooth regions](image)

Let $\chi_i$ be a characteristic function of $\Omega_i$ such that $\chi_i(x) = 1$ if $x \in \Omega_i$ and 0 otherwise. We can express the overall piecewise smooth approximation of the image as

$$u_\sigma(x, \Gamma) = \sum_i \chi_i \nu_\sigma(x, \Omega_i).$$
With this approximation, \( u_\sigma \) is a piecewise smooth function that is given analytically with respect to the boundary \( \Gamma \), and we do not need anymore the regularization term on \( u \) present in (1). This leads us to a new functional:

\[
E(\Gamma) = \mu^2 \int_S (u_0 - u_\sigma(\Gamma))^2 dx + \nu |\Gamma| = \mu^2 \sum_i \int_{\Omega_i} (u_0 - u_\sigma(\Omega_i))^2 dx + \nu |\Gamma| \tag{5}
\]

Interestingly, when the variance \( \sigma \) of the Gaussian goes to infinity, this functional becomes equivalent to the piecewise constant model. This limit model has become very popular in its level set formulation (Chan-Vese model[16]) because it performs very well for regions that are characterized by quite different global means. However it is not able to discriminate regions with nearly the same global intensity distributions (Figure 2). With a different choice of \( \sigma \), our model becomes more local and can segment a wider set of images where image regions differ only in their local intensity distributions. Hence, tuning the parameter \( \sigma \) permits to control the locality of intensity statistics inside each region.

![Fig. 2. Example of image that does not suit the Chan-Vese model](image)

Although our model has no restriction on the number of regions to segment, in the following we focus on the bi-partitioning case to make the explanations simpler. In particular, this allows us to represent the boundary \( \Gamma \) with a single level set function.

### 2.1 Energy Minimization

In the case of bi-partitioning, the contour \( \Gamma \) separates a region \( \Omega \) from its complementary \( \Omega^c \), and the energy (5) becomes:

\[
E(\Gamma) = \mu^2 \sum_{D=\{\Omega, \Omega^c\}} \int_D \left( u_0(x) - \frac{\int_D g_\sigma(x-y)u_0(y)dy}{\int_D g_\sigma(x-y)dy} \right) dx + \nu |\Gamma| \tag{6}
\]

\[\text{Several extensions using multiple level set functions have been proposed to segment an arbitrary number of regions [16].}\]
To minimize this energy, we use the shape gradient tools developed in [1]. The detailed derivation is presented in appendix A. It leads to the following evolution of the boundary \( \Gamma \):

\[
\frac{\partial \Gamma}{\partial t}(x) = \left( (u_0(x) - u_\sigma(x, \Omega))^2 + q_\sigma(x, \Omega) - (u_0(x) - u_\sigma(x, \Omega))^2 - q_\sigma(x, \Omega) \right) \mathbf{N}(x),
\]

with \( q_\sigma(x, \Omega) = \int_{\Omega} \frac{2 (u_0(y) - u_\sigma(y, \Omega))(u_0(x) - u_\sigma(x, \Omega))}{\int_{\Omega} g_\sigma(y - z) \, dz} \, dy. \tag{7} \]

Where \( \mathbf{N}(x) \), the outward normal vector to \( \Gamma \) at the point \( x \). The first two terms of this evolution equation are similar to the ones that can be found in the usual piecewise smooth and constant Mumford-Shah cases. Their interpretation is quite straightforward: the contour will locally move to include the current image voxel in the region it is the most similar to. The other terms are unique to our formulation and come from the analytical expression of the piecewise smooth image as a function of the boundary.

### 2.2 Level Set Implementation

Any curve representation can be used to implement the evolution described in (7). Here we present how to do it with a level set representation. In particular, this allows us to give an implementation that is valid in any dimension.

Let \( \phi \) be the signed distance function to the boundary \( \Gamma \), positive in \( \Omega \) and negative in \( \Omega \). We introduce \( H_\alpha \), regularized versions of the Heaviside functions. Equation (7) becomes:

\[
\frac{\partial \phi}{\partial t}(x) = \left( (u_0 - \overline{\pi}_\sigma(\phi))^2 - (u_0 - u_\sigma(\phi))^2 - \overline{\nu}_\sigma(\phi) + q_\sigma(\phi) \right) |\nabla \phi|, \tag{8} \]

All four different terms \( u_\sigma(\phi), \overline{\pi}_\sigma(\phi), q_\sigma(\phi) \) and \( \overline{\nu}_\sigma(\phi) \) can be computed with convolutions by the Gaussian kernel \( g_\sigma \):

\[
\begin{align*}
    u_\sigma(\phi) &= g_\sigma * H_\alpha(\phi) \, u_0 \\
    \overline{\pi}_\sigma(\phi) &= g_\sigma * (1 - H_\alpha(\phi)) \, u_0 \\
    q_\sigma(\phi) &= u_0 \left( g_\sigma * \frac{2(u_0 - u_\sigma)H_\alpha}{g_\sigma * H_\alpha(\phi)} \right) - g_\sigma * \frac{2(u_0 - u_\sigma)u_\sigma H_\alpha(\phi)}{g_\sigma * H_\alpha(\phi)} \\
    \overline{\nu}_\sigma(\phi) &= u_0 \left( g_\sigma * \frac{2(u_0 - \overline{\pi}_\sigma)H_\alpha(1 - H_\alpha(\phi))}{g_\sigma * (1 - H_\alpha(\phi))} \right) - g_\sigma * \frac{2(u_0 - \overline{\pi}_\sigma)\overline{\pi}_\sigma(1 - H_\alpha(\phi))}{g_\sigma * (1 - H_\alpha(\phi))} \tag{9} \end{align*}
\]

Each one of these terms need to be updated at each evolution of the level set. Even though these expressions seem complicated, their estimation is quite straightforward since they are the results of convolutions by a Gaussian kernel (more details are given in appendix B). This is a good advantage because it can
be implemented very effectively with a recursive filter \[4\]. Hence, for a \(d\) dimensional image with \(N\) voxels, the complexity of each convolution \(O(d.N)\), and only six convolutions are needed to compute the four terms. If we compare this complexity to the piecewise constant case, where the means inside each region also need to be recomputed at each iteration, it is of the same order.

3 Results and Comparisons

We applied our model on several images:

In Figure 3 we can see the role of the variance parameter \(\sigma\). As pointed out in section 2, when \(\sigma\) goes to infinity, the functional become equivalent to the Chan-Vese model. However, for small variances, the initial contour has to be close to edges to evolve, but is able to extract much thinner details. Actually, the model behave like the geodesic active contour model without any balloon force: it drives the front toward edges in the image, and makes it evolve via a mean curvature flow in homogeneous regions. More generally, the evolution of the front “follow” the edges that cross the initial contour. Our model is thus very dependent to the initialization.
Efficient Segmentation of Piecewise Smooth Images

(a) Initialization

Fig. 4. Limitation of the Chan-Vese model. first row : Chan-Vese model; second row : our model.

In Figure 4 we applied our model in the case of two distinct regions characterized by same global statistic. In the first row, we pointed out the limitations of the Chan-Vese model, that separate the white regions from the black, and do not extract accurately the leaf in the image. We can see in the second row our

Fig. 5. Extraction of the liver from 2D real data with the Chan-Vese model
Fig. 6. Extraction of the liver from 2D real data with our model ($\sigma = 16$)

model, that behave quite well. We can clearly see that the front "follows" the edges that crossed it in its initial shape.

In Figure 5 and 6 we applied both our model and the Chan-Vese model in order to extract a 2D liver from a biased anatomical MRI. As the Chan-Vese model is not robust to bias, the liver is not correctly extracted, and the front "leak" in the part of the image where intensity is close to the global mean inside the front. We can see that our model behave quite well, as the liver is correctly delimited at the local level. We just have to make the initial front cross the edges of the liver, and make the front evolve, thus following the edges of the liver, and finally extract it almost completely.

4 Conclusion and Future Works

In this paper, we presented a new model for extracting smooth regions from image data. This model is based on the Mumford-Shah functional, but is formulated in a simpler and more efficient way. We introduced a new functional, and showed that it was linked to the Chan-Vese model, by representing regions as local average instead of global mean. One of the most interesting point is that the minimization of this functional can be computed in a very fast way, thanks to the Deriche recursive filter. Finally, we showed quite promising results on 2D synthetic and real data.

In the future, we will apply this method to 3D medical images, in order to segment organs from anatomical MRI. We also started some works to estimate a different variance in each points, thus modeling regions as a space-varying convolution. The main goal is to remove the "sigma parameter" by estimating a varying optimal one, and thus to relax the initialization dependency of the model.
References

A Derivation of the Energy

In section 2, we defined an energy that is composed of two domain integrals. Here, we present a detailed derivation using shape gradients. We consider the following energy:

\[ E(\Omega, \Omega) = \int_{\Omega} f(x, \Omega) \, dx + \int_{\Omega} f(x, \overline{\Omega}) \, dx, \]

where

\[ f(x, \Omega) = (u_0(x) - \sigma(x, \Omega))^2 = \left( u_0(x) - \frac{G_1(x, \Omega)}{G_2(x, \Omega)} \right)^2, \quad (10) \]

and

\[
\begin{align*}
G_1(x, \Omega) &= \int_{\Omega} H_1(x, y) \, dy \quad \text{and} \quad H_1(x, y) = u_0(y) g(x, y) \\
G_2(x, \Omega) &= \int_{\Omega} H_2(x, y) \, dy \quad \text{and} \quad H_2(x, y) = g(x, y)
\end{align*}
\]

(11)

We recall the main theorem of shape gradient method presented in [1]:

**Theorem 1.** The Gâteaux derivative of a functional \( J(\Omega) = \int_{\Omega} g(x, \Omega) \, dx \) in the direction of a vector field \( \mathbf{V} \) is:

\[
\langle J'(\Omega), \mathbf{V} \rangle = \int_{\Omega} g_s(x, \Omega, \mathbf{V}) \, dx - \int_{\partial\Omega} g(x, \Omega) (\mathbf{V}(x) \cdot \mathbf{N}(x)) \, d\mathbf{a}(x)
\]

Where \( g_s(x, \Omega, \mathbf{V}) \) is the shape derivative of \( g(x, \Omega) \) in the direction of \( \mathbf{V} \), \( \partial\Omega \) is the boundary of \( \Omega \), \( \mathbf{N} \) is the unit inward normal to \( \partial\Omega \) and \( d\mathbf{a} \) is its area element.

In our case, for the bi-partitioning problem, we have,

\[
\langle E'(\Omega, \overline{\Omega}), \mathbf{V} \rangle = \int_{\Omega} f_s(x, \Omega, \mathbf{V}) \, dx + \int_{\Omega} f_s(x, \overline{\Omega}, \mathbf{V}) \, dx
\]

\[
- \int_{\Gamma} (f(x, \Omega) - f(x, \overline{\Omega})) (\mathbf{V}(x) \cdot \mathbf{N}(x)) \, d\mathbf{a}(x)
\]

with

\[ f_s(x, \Omega, \mathbf{V}) = f_{G_1}(x, \Omega, G_1, G_2) (G_1'(x, \Omega), \mathbf{V}) + f_{G_2}(x, \Omega, G_1, G_2) (G_2'(x, \Omega), \mathbf{V}), \]

where \( f_{G_1} \) and \( f_{G_2} \) denote the partial derivative of (10) with respect to \( G_1 \) and \( G_2 \). They can be expressed as:

\[
\begin{align*}
f_{G_1}(x, \Omega, G_1, G_2) &= -\frac{2}{G_2(x, \Omega)} \left( I(x) - \frac{G_1(x, \Omega)}{G_2(x, \Omega)} \right) \\
f_{G_2}(x, \Omega, G_1, G_2) &= \frac{2G_1(x, \Omega)}{G_2(x, \Omega)^2} \left( I(x) - \frac{G_1(x, \Omega)}{G_2(x, \Omega)} \right)
\end{align*}
\]

(12)
and by using Theorem 1

\begin{align}
\langle G'_1(x, \Omega) \cdot V \rangle &= \int_{\Omega} H_{1s}(x, y, V) \, dy - \int_{\Gamma} H_1(x, y)(V(y) \cdot N(y)) \, da(y) \\
\langle G'_2(x, \Omega) \cdot V \rangle &= \int_{\Omega} H_{2s}(x, y, V) \, dy - \int_{\Gamma} H_2(x, y)(V(y) \cdot N(y)) \, da(y)
\end{align}

(13)

Since \( H_1 \) and \( H_2 \) do not depend on \( \Omega \), we obtain \( H_{1s} = 0 \) and \( H_{2s} = 0 \).

Putting all the term together we find

\begin{align}
f_s(x, \Omega, V) &= \langle G'_1 \cdot V \rangle + \langle G'_2 \cdot V \rangle \\
&= \frac{2}{f_{G_1}} \int_{\Omega} g_s(x - y)(u_0(y) - u_\sigma(x, \Omega))(V(y) \cdot N(y)) \, da(y)
\end{align}

(14)

and at last we obtain, by changing the order of integration,

\begin{align}
\langle E'(\Omega, \Omega), V \rangle &= \int_{\Omega} f_s(x, \Omega, V) \, dx + \int_{\Omega} f_s(x, \Omega, V) \, dx \\
&\quad - \int_{\Gamma} (f(y, \Omega) - f(y, \Omega))(V(y) \cdot N(y)) \, da(y) \\
&= \int_{\Gamma} \left( q(y, \Omega) - q(y, \Omega) - (u_0(y) - u_\sigma(y, \Omega))^2 + (u_0(y) - u_\sigma(y, \Omega))^2 \right) (V(y) \cdot N(y)) \, da(y)
\end{align}

(15)

with

\[ q(y, \Omega) = \int_{\Omega} \frac{2}{f_{G_1}} \int_{\Omega} g_s(x - y)(u_0(y) - u_\sigma(x, \Omega)) \, dx \]

(16)

We finally get the following gradient descent:

\[ \frac{\partial \Gamma}{\partial \tau}(x) = \left[ (q(x, \Omega) - q(x, \Omega) - (u_0(x) - u_\sigma(x, \Omega))^2 + (u_0(x) - u_\sigma(x, \Omega))^2 \right] N(x) \]

(17)

### B Implementation

Each integral term in the gradient descent can be seen as a convolution by a kernel of variance \( \sigma \). Therefore, the computation of the evolution can be done in a very fast way in two separate steps:

1. first we make several convolutions via a fast recursive filter (ref deriche)
then we compute the speed of each points in the narrow-band by using the previously computed blurred images.

We have,

\[
    u_\sigma(x, \Omega) = \frac{\int_{\Omega} g_\sigma(x-y) u_0(y) dy}{\int_{\Omega} g_\sigma(x-y) dy} = (g_\sigma * u_0) |_{\Omega}(x)
\]

(18)

and

\[
    q(y, \Omega) = \int_{\Omega} \frac{2(u_0(x) - u_\sigma(x, \Omega)) g_\sigma(x-y) (u_0(y) - u_\sigma(x, \Omega))}{\int_{\Omega} g_\sigma(x-z) dz} dx
\]

\[
    = u_0(y) \int_{\Omega} \frac{2(u_0(x) - u_\sigma(x, \Omega))}{(g_\sigma * 1) |_{\Omega}(x)} g_\sigma(x-y) dx
\]

\[
    - \int_{\Omega} \frac{2(u_0(x) - u_\sigma(x, \Omega)) u_\sigma(x, \Omega)}{(g_\sigma * 1) |_{\Omega}(x)} g_\sigma(x-y) dx
\]

\[
    = u_0(y) (g_\sigma * q_1) |_{\Omega}(y) - (g_\sigma * q_2) |_{\Omega}(y)
\]

with

\[
    q_1(x, \Omega) = \frac{2(u_0(x) - u_\sigma(x, \Omega))}{(g_\sigma * 1) |_{\Omega}(x)}
\]

and

\[
    q_2(x, \Omega) = \frac{2(u_0(x) - u_\sigma(x, \Omega)) u_\sigma(x, \Omega)}{(g_\sigma * 1) |_{\Omega}(x)}
\]

(19)

We can compute the domain-convolution by using the Heaviside $H_\sigma$ of the level-set function :

\[
    \left\{ \begin{array}{l}
    (g_\sigma * f) |_{\Omega}(x) = (g_\sigma * H_\sigma f)(x) \\
    (g_\sigma * f) |_{\Omega}(x) = (g_\sigma * (1 - H_\sigma) f)(x)
    \end{array} \right.
\]

(20)
Implicit Active Shape Models for 3D Segmentation in MR Imaging

Mikaël Rousson¹, Nikos Paragios², and Rachid Deriche¹

¹ I.N.R.I.A. Sophia Antipolis, France
E-mail: {Mikael.Rousson,Rachid.Deriche}@sophia.inria.fr
² Ecole Nationale des Ponts et Chaussées, Paris, France
E-mail: Nikos.Paragios@cermics.enpc.fr

Abstract. Extraction of structures of interest in medical images is often an arduous task because of noisy or incomplete data. However, hand-segmented data are often available and most of the structures to be extracted have a similar shape from one subject to another. Then, the possibility of modeling a family of shapes and restricting the new structure to be extracted within this class is of particular interest. This approach is commonly implemented using active shape models [2] and the definition of the image term is the most challenging component of such an approach. In parallel, level set methods [8] define a powerful optimization framework, that can be used to recover objects of interest by the propagation of curves or surfaces. They can support complex topologies, considered in higher dimensions, are implicit, intrinsic and parameter free. In this paper we re-visit active shape models and introduce a level set variant of them. Such an approach can account for prior shape knowledge quite efficiently as well as use data/image terms of various form and complexity. Promising results on the extraction of brain ventricles in MR images demonstrate the potential of our approach.

1 Introduction

Object extraction is one of the first steps in medical imaging. Further analysis will highly depend on the quality of the segmented structures. However, medical images often suffer from noise, occlusions and incomplete data. Therefore, regularization constraints and prior knowledge are usually of good use. In this paper, we address this application with objective to recover a structure of particular geometric form.

B-splines deformable models as well as point distribution models are mathematical formulations introduced to the snake framework [4] to account for shape consistency. Active shape models [2] were a major breakthrough in object extraction and image segmentation. Such a framework consists of two stages; (i) the modeling and (ii) the segmentation phase.

During modeling the objective is to recover a compact representation for the geometric form of the structure of interest. Using a set of registered training examples, one can either represent prior knowledge using simple or more
complicated density functions. Gaussian distribution [2], mixture models [1] or non-parametric function [3] were considered in the past. The segmentation/object extraction stage aims at recovering a geometric structure in the image plane that accounts for the desired image characteristics while being in the family of shapes generated by the model. To this end, a mechanism for recovering the most probable object location in the image was considered. Then, one can iterate and move closer to the target by updating the position of the model such that it gets closer to the desired image characteristics.

Level set representations [8] is an established technique for tracking moving interfaces in imaging, vision, and graphics [7]. One can see numerous advantages for considering a level set variant of the active shape model. Such a formulation could account for various forms (boundary or regional) of data/image terms of various nature (edges, intensity properties, texture, motion, etc.), an important limitation of the active shape model. Furthermore, one can maintain the implicit and intrinsic property of the level set method as well as the ability to account for topological changes while being able to introduce prior shape knowledge, a task partially addressed up to now [6, 12, 10].

In this paper we propose a level set variant of active shape models that consists of various terms. Quite critical is the term that refers to the prior knowledge with objective to constrain the evolving surface to belong to a compact family of shapes - the one recovered through the training set. Such a term couples two unknown variables: (i) the evolving contour, (ii) the optimal projection parameters of this contour to the model space and imposes the active shape model behavior on the process. Furthermore, various image-driven terms - a major advantage/characteristics of the method - could be considered to guide the evolving contour towards the desired image characteristics.

The most closely related work with our approach, the active shape model can be found in [2]. In [6, 12, 10] substantial efforts to integrate prior knowledge within level set representations were considered. Worth mentioning is [6, 12] where modeling of prior knowledge is done in a consistent active shape model manner. Contrary to [6], where two optimization processes alternate, we propose a variational integration of data and prior terms. Moreover, the evolving surface is not restricted within the modeled space like in [12], but only attracted to this space, allowing more flexibility.

The reminder of the paper is organized as follows: in Section 2 we briefly introduce the level set representations, while in Section 3 we address the construction of the prior model in the space of level set functions. The main contribution of the paper, the level set variant of the active shape model is presented in Section 4, while in Section 5 we demonstrate the efficiency and the flexibility of our approach through the integration of a region-based data term for 3D segmentation in MR images.

2 Level Set Representations

Level set representations [8] are a useful mathematical formulation for implementing efficiently curve/surface propagation. One can also consider the level
let space as an optimization framework. Let \( \phi: \Omega \times \mathbb{R}^+ \to \mathbb{R}^+ \) be a Lipschitz function with the following properties,

\[
\phi((x, y); t) = \begin{cases} 
0 & (x, y) \in C(t) \\
+D((x, y), C(t)) & (x, y) \in C_{in}(t) \\
-D((x, y), C(t)) & (x, y) \in C_{out}(t) = [\Omega - C_{in}(t)]
\end{cases}
\]

where \((x, y) = p, C_{in}(t)\) is the area enclosed by the curve \(C, D((x, y), C(t))\) the minimum Euclidean distance between the pixel \((x, y)\) and \(C(t)\) at time \(t\).

Let us also introduce the approximations of Dirac and Heaviside distributions as defined in [10]. Then one can define terms along \(C\) as well as interior and exterior to the curve using the Dirac and Heaviside functions:

\[
(x, y) \in \Omega: \{\lim_{\alpha \to 0^+} [\delta_\alpha(\phi(x, y))] = 1\} = C
\]

\[
(x, y) \in \Omega: \{\lim_{\alpha \to 0^+} [H_\alpha(\phi(x, y))] = 1\} = C_{in}
\]

Such terms will be used later to introduce the active shape prior term as well as data/image-driven terms that guides the contour \((C)\) towards the object of interest. The extension to higher dimensions is straightforward and in the following parts, we use this representation for an an evolving surface \((S)\) in \(\mathbb{R}^3\).

### 3 Modeling Prior Knowledge in the Level Set Space

Learning the distribution of geometric/image structures is a common problem in computer vision with applications to segmentation, tracking, recognition, etc. It is clear that the selection of the representation is important. Given the selected optimization framework, level set functions is a natural selection to account for prior knowledge with numerous earlier described advantages. Let us consider a training set \(\phi_i\) of \(N\) registered curves or surfaces. Then, a distance transform can be used to represent \(\phi_i\).

The next step is the construction of the shape model, using the aligned contours. In order to create an invariant representation, one should first normalize the training set \(\phi_i\). Subtraction of the mean (that can be recovered by averaging \(\phi_i\)’s) is a common selection to this end. However, a simple averaging over the training will not give a distance function. To overcome this limitation, we consider a more rigorous approach [10], seeking to estimate the distance function \((\phi_M)\) that minimizes:

\[
E(\phi_M) = \sum_{i=1}^{n} \int_\Omega (\phi_i - \phi_M)^2 d\Omega, \quad \text{subject to}: |\nabla \phi_M|^2 = 1
\]

One can optimize such a term though a gradient descent method:

\[
\frac{d}{dt} \phi_M = \sum_{i=1}^{n} (\phi_i - \phi_M)
\]

while \(\phi_M\) is projected to the space of distance functions following [11]. The two steps alternate until the system reaches a steady-state solution. Then, we consider the modeling approach introduced in [6, 12]. Once the samples \(\phi_i\) centered
with respect to $\phi_M$, 

$$[\psi_i = \phi_i - \phi_M],$$

the most important modes of variations can be recovered through Principal Component Analysis:

$$\phi = \phi_M + \sum_{j=1}^{m} \lambda_j U_j$$

where $m$ is the number of retained modes of variation, $U_j$ are these modes (eigenvectors), and $\lambda_j$ are linear weight factors within the allowable range defined by the eigenvalues.

An example of such an analysis is shown in [fig. (1)] for the 3D modeling of lateral brain ventricles. The model was built using 8 surfaces from different subjects. This example includes a difficult issue for classical parametric approaches because of different surface topologies within the training set. For example, the fourth surface in [fig. (1)] shows a separation between left and right ventricles. Our approach can deal naturally with this type of data. The obtained model gives a compact representation of the shape family: the first two modes of variation represent the major part of the class (80%), while the third one (9%) accounts for non-symmetric properties of the ventricles that can be observed in some of the training samples. Moreover, the implicit representation of the surfaces make the modeling phase entirely automatic.

### 4 Introducing Prior Knowledge in the Level Set Space

Let us now consider an interface represented by a level-set function $\phi(x)$ as described in Section 2 (where $x$ is in $\mathbb{R}^2$ or $\mathbb{R}^3$). We would like to evolve it while respecting some shape properties $\phi_P(x)$ modulo a transformation $A$ belonging to a predefined family. Assuming a rigid transformation $A(x) = Rx + T$, the evolving interface and the transformation should satisfy the conditions:

$$\begin{cases}
    x \rightarrow A(x) \\
    \phi(x) \approx \phi_P(A(x)), \quad \forall x \in \Omega
\end{cases}$$
In that case, the optimal transformation $A$ should minimize:

$$E(\phi, A) = \int_\Omega \rho(\phi, \phi_P(A)) d\Omega$$

where $\rho$ is a dissimilarity measure. For the sake of simplicity, we will use the sum of squared differences. Scale variation can be added to the rigid transformation $A$, leading to a similarity one $A(x) = S \mathbf{R} x + T$ (for 3D images, we obtain 7 parameters: $S, R(\theta_1, \theta_2, \theta_3), T = (T_x, T_y, T_z)^T$). In that case, the objective function should be slightly modified (refer to [10] for further details). Furthermore, one can assume that estimating and imposing the prior within the vicinity of the zero-crossing of the level set representation is more meaningful. Within distance transforms, shape information is better captured when close to the origin of the transformation. The prior can be thus rewritten:

$$E(\phi, A) = \int_\Omega \delta_\epsilon(\phi) (S\phi - \phi_P(A))^2 d\Omega \quad \text{where } \epsilon \gg \alpha$$

During the model construction, we have analyzed the principal modes of variation within the training set. Including this information, the ideal transformation will map each value of current representation to the ”best” level set representation belonging to the class of the training shapes. If a shape representation $\phi_P$ belongs to this class, then it can be derived from the principal modes:

$$\phi_P = \phi_M + \sum_{j=1}^m \lambda_j U_j$$

Hence, we define a new objective function by introducing the modes weights $\lambda = (\lambda_1, \ldots, \lambda_m)$ as additional free parameters:

$$E(\phi, A, \lambda) = \int_\Omega \delta_\epsilon(\phi) \left( S\phi - \left( \phi_M(A) + \sum_{j=1}^m \lambda_j U_j(A) \right) \right)^2 d\Omega$$
In order to minimize the above functional with respect to the evolving level set representation, the global linear transformation and the modes weights $\lambda_j$, we use the calculus of variations. The equation of evolution for $\phi$ is given by the calculus of its variations:

$$\frac{d}{dt}\phi = -2\delta(\phi)\mathcal{S}(S\phi - \phi_M(A)) - \frac{d}{d\phi}\delta(\phi)(S\phi - \phi_M(A))^2$$

The differentiation with respect to the modes weights gives us a close form of the optimal parameters by solving the linear system $\bar{U}\lambda = b$ with:

$$\begin{align*}
\bar{U}_{i,j} &= \int_{\Omega} \delta_i(\phi)U_i(A)U_j(A) \\
b_i &= \int_{\Omega} \delta_i(\phi)(S\phi - \phi_M(A))U_i(A)
\end{align*}$$

where $\bar{U}$ is a $m \times m$ positive definite matrix. Finally, the minimization of the energy with respect to the pose parameters is done by considering the gradient of each parameter:

$$\begin{align*}
\frac{d}{dt}S &= 2\int_{\Omega} \delta_i(\phi)\mathcal{S}(S\phi - \phi_P(A))(-\phi + \nabla\phi_P(A) \cdot \frac{\partial}{\partial S}A)d\Omega \\
\frac{d}{dt}a_i &= 2\int_{\Omega} \delta_i(\phi)\mathcal{S}(S\phi - \phi_P(A))\mathcal{S}(\nabla\phi_P(A) \cdot \frac{\partial}{\partial a_i}A)d\Omega \quad \text{with } a_i \in \{\theta_1, \theta_2, \theta_3, T_x, T_y, T_z\}
\end{align*}$$

5 Active Shapes, Level Sets & Object Extraction

In this section, we integrate the proposed level set variant of the active shape model to the Geodesic Active Region model [9], that on top of salient features uses global region statistics.
5.1 Geodesic Active Region

Introducing global region properties is a common technique to improve segmentation performance. To this end, one can assume a two-class partition problem where the object and the background follow different intensity distribution. Let $p_{C_{in}}$ and $p_{\Omega-C_{in}}$ be the densities of $I(x)$ in $C_{in}$ and $\Omega - C_{in}$. Then according to the Geodesic Active Region model [9] one can recover the object through the optimization of the following function:

$$E(\phi, p_{C_{in}}, p_{\Omega-C_{in}}) = (1 - a) \int_{\Omega} \delta_a(\phi) g(|\nabla I|)|\nabla \phi|d\Omega$$

$$- a \int_{\Omega} [H_a(\phi) \log(p_{C_{in}}(I)) + (1 - H_a(\phi)) \log(p_{\Omega-C_{in}}(I))] d\Omega$$

One can consider either parametric approximation [9] or a non-parametric density [5] functions to describe intensity properties. In both cases the new term will result in a local balloon force that moves the contour in the direction that maximizes the posterior segmentation probability as shown in [9].

5.2 Object Extraction

The Geodesic Active Region module is used jointly with the shape prior constraint. This data-specific information make the contour evolve toward the object of interest while keeping a global shape consistent with the prior shape family. For this purpose a variational formulation incorporating two terms is used:

$$E(\phi, A, \lambda) = b E_{shape}(\phi, A, \lambda) + (1 - b) E_{data}(\phi)$$

where $E_{shape}$ is the shape prior and $E_{data}$ is the Geodesic Active Region module.

This framework has been tested on the extraction of the lateral brain ventricles. [Fig. (2)] show the robustness to noise brought by the prior shape knowledge (the image is one of the training images but with additional Riccian noise). In [fig. (3)], we show the ability of our approach to extract objects from new images (not used for building the model). The active shape model is able to approximate the surface with a similar one from the modeled class while the object extraction allows small local variations with respect to the model. Finally, in [fig. (4)], we show the influence of the shape prior term by changing its weight. While prior knowledge improves the quality of the object extraction, overweighting shape prior will make object details to be missed. The possibility of tuning this parameter is an important advantage of our approach compared to [12].
6 Conclusion

We have proposed a level set variant of active shape models to deal with object extraction in medical MR images. Our approach exhibits numerous advantages. It can deal with noisy, incomplete and occluded data because of its active shape nature. It is intrinsic, implicit parameter and topology free, a natural property of the level set space. Examples on the brain ventricles extraction demonstrate the potential of our method. The nature of the sub-space of plausible solutions is a limitation of the proposed framework. Quite often the projection to this space does not correspond to a level set distance function. To account for this limitation, we currently explore prior modeling directly on the Euclidean space, and then conversion to the implicit space during the object extraction.

References

Efficient Kernel Density Estimation of Shape and Intensity Priors for Level Set Segmentation

Mikael Rousson and Daniel Cremers

Department of Imaging and Visualization
Siemens Corporate Research, Princeton, NJ

{mikael.rousson,daniel.cremers}@siemens.com

Abstract. We propose a nonlinear statistical shape model for level set segmentation which can be efficiently implemented. Given a set of training shapes, we perform a kernel density estimation in the low dimensional subspace spanned by the training shapes. In this way, we are able to combine an accurate model of the statistical shape distribution with efficient optimization in a finite-dimensional subspace. In a Bayesian inference framework, we integrate the nonlinear shape model with a nonparametric intensity model and a set of pose parameters which are estimated in a more direct data-driven manner than in previously proposed level set methods. Quantitative results show superior performance (regarding runtime and segmentation accuracy) of the proposed nonparametric shape prior over existing approaches.

1 Introduction

Originally proposed in [5, 11] as a means to propagate interfaces in time, the level set method has become increasingly popular as a framework for image segmentation. The key idea is to represent an interface \( \Gamma \subset \Omega \) in the image domain \( \Omega \subset \mathbb{R}^3 \) implicitly as the zero level set of an embedding function \( \phi : \mathbb{R}^3 \rightarrow \Omega \):

\[
\Gamma = \{ x \in \Omega \mid \phi(x) = 0 \},
\]

and to evolve \( \Gamma \) by propagating the embedding function \( \phi \) according to an appropriate partial differential equation. The first applications of this level set formalism for the purpose of image segmentation were proposed in [1, 10, 7]. Two key advantages over explicit interface propagation are the independence of a particular parameterization and the fact that the implicitly represented boundary \( \Gamma \) can undergo topological changes such as splitting or merging. This makes the framework well-suited for the segmentation of several objects or multiply-connected objects.

When segmenting medical images, one commonly has to deal with noise, missing or misleading image information. For certain imaging modalities such as ultrasound or CT, the structures of interest do not differ much from their background in terms of their intensity distribution — see Figure 1. Therefore they can no longer be accurately segmented based on the image information.
Fig. 1. Segmentation challenges and estimated intensity distributions.
The two curves on the right correspond to the empirical probability of intensities
inside and outside the left ventricle (for the ultrasound image) and the prostate (for
the CT image). The region-based segmentation of these structures is a challenging
problem, because objects and background have similar histograms. Our segmenta-
tion scheme optimally exploits the estimated probabilistic intensity models.

alone. In recent years, researchers have therefore proposed to enhance the level
set method with statistical shape priors. Given a set of training shapes, one can
impose information about which segmentations are a priori more or less likely.
Such prior shape information was shown to drastically improve segmentation
results in the presence of noise or occlusion [9, 16, 3, 14, 4, 6]. Most of these ap-
proaches are based on the assumption that the training shapes, encoded by their
signed distance function, form a Gaussian distribution. This has two drawbacks:
Firstly, the space of signed distance functions is not a linear space, therefore,
the mean shape and linear combinations of eigenmodes are typically no longer
signed distance functions. Secondly, even if the space were a linear space, it is
not clear why the given set of sample shapes should be distributed according to
a Gaussian density. In fact, as we will demonstrate in this work, they are gen-
erally not Gaussian distributed. Recently, it was proposed to use nonparametric
density estimation in the space of level set functions [3] in order to model non-
linear\(^1\) distributions of training shapes. While this resolves the above problems,
one sacrifices the efficiency of working in a low-dimensional subspace (formed by
the first few eigenmodes) to a problem of infinite-dimensional optimization.

In the present paper, we propose a framework for knowledge-driven level set
segmentation which integrates three contributions: Firstly, we propose a statis-
tical shape prior which combines the efficiency of low-dimensional PCA-based
methods with the accuracy of nonparametric statistical shape models. The key
idea is to perform kernel density estimation in a linear subspace which is suffi-
ciently large to embed all training data. Secondly, we propose to estimate pose
and translation parameters in a more data-driven manner. Thirdly, we optimally
exploit the intensity information in the image by using probabilistic intensity
models given by kernel density estimates of previously observed intensity distri-
butions.

\(^1\) The term nonlinear refers to the fact that the manifold of permissible shapes is not
merely a linear subspace.
2 Level Set Segmentation as Bayesian Inference

The goal of level set segmentation can be formulated as the estimation of the optimal embedding function \( \phi : \Omega \rightarrow \mathbb{R} \) given an image \( I : \Omega \rightarrow \mathbb{R} \). In the Bayesian framework, this can be computed by maximizing the posterior distribution

\[
P(\phi | I) \propto P(I | \phi) P(\phi). \tag{2}
\]

The maximization of (2) results in a problem of infinite-dimensional optimization. Given a set of training shapes encoded by their signed distance functions \( \{\phi_i\}_{i=1..N} \), Tsai et al. [16] proposed to reduce the segmentation problem to one of finite-dimensional optimization by constraining the optimization problem to the finite-dimensional subspace spanned by the training shapes.

In this paper, we make use of this compact representation of the embedding function. Given the distance \( d \) on the space of signed distance functions defined by:

\[
d^2(\phi_1, \phi_2) = \int_{\Omega} (\phi_1(x) - \phi_2(x))^2 \, dx,
\]

we align the set of training shapes with respect to translation and rotation. Subsequently, we constrain the level set function \( \phi \) to a parametric representation of the form:

\[
\phi_{\alpha, h, \theta}(x) = \phi_0(R_\theta x + h) + \sum_{i=1}^{n} \alpha_i \psi_i(R_\theta x + h), \tag{3}
\]

where \( \phi_0(x) = \frac{1}{N} \sum_{i=1}^{N} \phi_i(x) \) represents the mean shape, \( \{\psi_i(x)\}_{i=1..n} \) are the eigenmodes of the distribution, and \( n < N \) is the dimension of the subspace spanned by the \( N \) training shapes. The parameter vector \( \alpha = (\alpha_1, \ldots, \alpha_n) \) models shape deformations, while the parameters \( h \in \mathbb{R}^3 \) and \( \theta \in [0, 2\pi]^3 \) model translation and rotation of the respective shape.\(^2\)

The infinite-dimensional Bayesian inference problem in (2) is therefore reduced to a finite-dimensional one where the conditional probability

\[
P(\alpha, h, \theta | I) \propto P(I | \alpha, h, \theta) P(\alpha, h, \theta), \tag{4}
\]

is optimized with respect to the shape parameters \( \alpha \), and the transformation parameters \( h \) and \( \theta \). In the following, we will assume a uniform prior on these transformation parameters, i.e. \( P(\alpha, h, \theta) = P(\alpha) \). In the next section, we will discuss three solutions to model this shape prior.

3 An Efficient Nonparametric Statistical Shape Model

Given a set of aligned training shapes \( \{\phi_i\}_{i=1..N} \), we can represent each of them by their corresponding shape vector \( \{\alpha_i\}_{i=1..N} \). In this notation, the goal of statistical shape learning is to infer a statistical distribution \( P(\alpha) \) from these sample shapes. Two solutions which have been proposed are based on the assumptions

\(^{2}\) In our applications, where the scale of objects is known, a generalization to larger transformations groups (e.g. similarity or affine) did not appear useful.
that the training shapes can be approximated by a uniform distribution \([16, 14]\): 
\[ \mathcal{P}(\alpha) = \text{const.} \]

or by a Gaussian distribution \([9]\):
\[
\mathcal{P}(\alpha) \propto \exp\left(-\alpha^\top \Sigma^{-1} \alpha\right), \quad \text{where} \quad \Sigma = \frac{1}{N} \sum_i \alpha_i \alpha_i^\top.
\] (5)

In the present paper, we propose to make use of nonparametric density estimation \([13]\) to approximate the shape distribution within the linear subspace. We model the shape distribution by the kernel density estimate:
\[
\mathcal{P}(\alpha) = \frac{1}{N \sigma} \sum_{i=1}^N K\left(\frac{\alpha - \alpha_i}{\sigma}\right), \quad \text{where} \quad K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right).
\] (6)

There exist various methods to automatically estimate appropriate values for the width \(\sigma\) of the kernel function, ranging from \(k\)-th nearest neighbor estimates to cross-validation and bootstrapping. In this work, we simply set \(\sigma\) to be the average nearest neighbor distance:
\[
\sigma^2 = \frac{1}{N} \sum_{i=1}^N \min_{j \neq i} |\alpha_i - \alpha_j|^2.
\]

In the context of level set based image segmentation, the kernel density estimator (6) has two advantages over the uniform and Gaussian distributions:

- The assumptions of uniform distribution or Gaussian distribution are generally not fulfilled. The kernel density estimator, on the other hand, is known to approximate arbitrary distributions. Under mild assumptions, it was shown to converge to the true distribution in the limit of infinite sample size. We refer to \([15]\) for a proof.

- The space of signed distance functions is known to not be a linear space. Therefore, neither the mean shape \(\phi_0\) nor a linear combination of eigenmodes as in (3) will in general be a signed distance function. As a consequence, the functions \(\phi(x)\) favored by the uniform or the Gaussian distribution cannot be expected to be signed distance functions. The kernel density estimator (6), on the other hand, favors shape vector \(\alpha\) which are in the vicinity of the sample shape vectors \(\alpha_i\). By construction, these vector correspond to signed distance functions. In fact: In the limit of infinite sample size, the distribution inferred by the kernel density estimator (6) converges towards a distribution on the manifold of signed distance functions.
Figure 2 shows schematic plots of the three methods for a set of sample data spanning a two-dimensional subspace in $\mathbb{R}^3$. The kernel density estimator clearly captures the distribution most accurately.

In analogy to the shape learning, we make use of kernel density estimation to learn the conditional probability for the intensity function $I$ in (4) from examples. A similar precomputation of intensity distributions by means of mixture models was proposed in [12]. Given a set of presegmented training images, the kernel density estimate of the intensity distributions $p_{in}$ and $p_{out}$ of object and background are given by the corresponding smoothed intensity histograms. This has two advantages: Firstly, the kernel density estimator does not rely on specific assumptions about the shape of the distribution. Figure 1 shows that the intensity distributions for ultrasound and CT images are not well approximated by Gaussian or Laplacian models. Secondly, in contrast to the joint estimation of intensity distributions (cf. [2, 8]), this simplifies the segmentation process which no longer requires an updating of intensity models. Moreover, we found the segmentation process to be more robust to initialization in numerous experiments.

4 Energy Formulation and Minimization

Maximizing the posterior probability in (2), or equivalently minimizing its negative logarithm, will generate the most probable segmentation of a given image. With the nonparametric models for shape and intensity introduced above, this leads to an energy of the form

$$E(\alpha, h, \theta) = -\log P(I|\alpha, h, \theta) - \log P(\alpha),$$

(7)

The nonparametric intensity model permits to express the first term and equation (6) gives exactly the second one. With the Heaviside step function $H$ and the short hand

$$K_i = K \left( \frac{\alpha - \alpha_i}{\sigma} \right),$$

we end up with:

$$E(\alpha, h, \theta) = -\int_{\Omega} H \log p_{in}(I) + (1-H) \log p_{out}(I) \, dx - \log \left( \frac{1}{N} \sum_{i=1}^{N} K \right),$$

With $e(x) = [\log \frac{p_{out}(I(x))}{p_{in}(I(x))}]$, $K_i = K \left( \frac{\alpha - \alpha_i}{\sigma} \right)$, and $\psi = (\psi_1, \ldots, \psi_n)$, we obtain the following system of coupled gradient descent equations:

$$\begin{align*}
\frac{d\alpha}{dt} &= \int_{\Omega} \delta(\phi_{\alpha, h, \theta}(x)) \psi(R_{\theta}x + h) e(x) \, dx + \frac{1}{\sigma^2} \sum_{i=1}^{N} (\alpha_i - \alpha) K_i, \\
\frac{dh}{dt} &= \int_{\Omega} \delta(\phi_{\alpha, h, \theta}(x)) \nabla \phi_{\alpha, h, \theta}(x) e(x) \, dx, \\
\frac{d\theta}{dt} &= \int_{\Omega} \delta(\phi_{\alpha, h, \theta}(x)) (\nabla \phi_{\alpha, h, \theta}(x) \cdot \nabla_{\theta} Rx) e(x) \, dx.
\end{align*}$$

(8)

In applications, we solve these equations by initializing the shape $\alpha$ with the mean shape ($\alpha = 0$) and the transformation parameters $h$ and $\theta$ with some...
reasonable estimates. Subsequently, we discretize the above partial differential equations by a standard finite difference scheme.

Note that in all equations, the Dirac delta function $\delta$ appears as factor inside the integrals over the image domain $\Omega$. This allows to restrict all computations to a narrow band around the zero crossing of $\phi$. While the evolution of translation and pose parameters $h$ and $\theta$ are merely driven by the data term $e(x)$, the shape vector $\alpha$ is additionally drawn towards each training shape with a strength that decays exponentially with the distance to the respective shape.

5 Experimental Results and Validation

Heart segmentation from ultrasound images

Figure 3 shows experimental results obtained for the segmentation of the left ventricle in 2D cardiac ultrasound sequences, using shape priors constructed from a set of 21 manually segmented training images. In contrast to the segmentation with uniform prior (top row), the nonparametric statistical shape prior allows to accurately constrain the segmentation (bottom row). This becomes particularly apparent in areas where the data term is too weak. As a quantitative evaluation we computed the percentage of correctly classified object pixels and that of misclassified ones. During energy minimization, the percentage of correctly classified pixels increases from 56% to 90% while the percentage of false positives decreases from 27% to 2.7% by using the kernel prior. Using the uniform prior, we attain 92% correctly classified, yet the percentage of false positives increases to 42%: Merely constraining the boundary evolution to the linear subspace spanned by the training shapes is insufficient to provide for accurate segmentation results.

Prostate segmentation from 3D CT images

We built a nonparametric 3D shape model of the prostate using 12 manually extracted prostates (with seminal vesicles) collected from two different patients. In contrast to existing work, we subsequently used a single shape model for the segmentation of images from different patients.

We employed a leave-one-out strategy by removing the image of interest from the training phase. Figure 5 shows 2D cuts of a few results obtained using this
strategy. With a one-click initialization inside the organ, the algorithm lead to a steady-state solution in less than 20 seconds. We obtained 86% successfully classified organ voxels and 11% mis-classified organ voxels. This compares favorably to the intra-patients results reported in [6]. Figure 4 provides qualitative comparisons to the manual segmentation, as well as to the segmentations obtained with uniform and Gaussian approximations of the shape distribution.

6 Conclusion

We proposed an efficient and accurate statistical shape prior for level set segmentation which is based on nonparametric density estimation in the linear subspace spanned by the level set surfaces of a set of training shapes. In addition, our segmentation scheme integrates nonparametric estimates of intensity distributions and efficient optimization of pose and translation parameters. We reported quantitative evaluation of segmentation accuracy and speed for cardiac ultrasound images and for 3D CT images of the prostate. These indicate that the proposed nonparametric shape prior outperforms previously proposed shape priors for level set segmentation.
Acknowledgments

We thank Christophe Chefd’hotel for fruitful discussions. We thank Marie-Pierre Jolly for providing us with image and training data for the ultrasound sequences.

References

Abstract—We address the problem of the segmentation of cerebral white matter structures from diffusion tensor images (DTI). A DTI produces, from a set of diffusion-weighted MR images, tensor-valued images where each voxel is assigned with a $3 \times 3$ symmetric, positive-definite matrix. This second order tensor is simply the covariance matrix of a local Gaussian process, with zero-mean, modeling the average motion of water molecules. As we will show in this paper, the definition of a dissimilarity measure and statistics between such quantities is a nontrivial task which must be tackled carefully. We claim and demonstrate that, by using the theoretically well-founded differential geometrical properties of the manifold of multivariate normal distributions, it is possible to improve the quality of the segmentation results obtained with other dissimilarity measures such as the Euclidean distance or the Kullback-Leibler divergence. The main goal of this paper is to prove that the choice of the probability metric, i.e., the dissimilarity measure, has a deep impact on the tensor statistics and, hence, on the achieved results. We introduce a variational formulation, in the level-set framework, to estimate the optimal segmentation of a DTI according to the following hypothesis: Diffusion tensors exhibit a Gaussian distribution in the different partitions. We must also respect the geometric constraints imposed by the interfaces existing among the cerebral structures and detected by the gradient of the DTI. We show how to express all the statistical quantities for the different probability metrics. We validate and compare the results obtained on various synthetic data-sets, a biological rat spinal cord phantom and human brain DTIs.

Index Terms—Diffusion tensor MRI, Fisher information matrix, information geometry, Kullback-Leibler divergence, level-set, probability metric, Riemannian geometry, segmentation.

I. INTRODUCTION

DIFFUSION magnetic resonance imaging is a relatively new modality [7], [42] able to quantify the anisotropic diffusion of water molecules in highly structured biological tissues. As of today, it is the only noninvasive method that allows to distinguish the anatomical structures of the cerebral white matter such as the corpus callosum, the superior longitudinal fasciculus or the corticospinal tract. These are examples of commissural, association and projection neural pathways, the three main types of fiber bundles, respectively connecting the two hemispheres, regions of a given hemisphere, or the cerebral cortex with the basal ganglia and the cerebellum.

In 1994, Basser et al. [5] proposed to model the local probability density function of the three-dimensional (3-D) molecular motion by a Gaussian distribution whose covariance matrix is given by the diffusion tensor. The estimation of these tensors requires the acquisition of diffusion weighted images in different sampling directions. Numerous algorithms have been proposed to perform a robust estimation and regularization of these tensors fields [8], [17], [19], [38], [41], [58], [60], [63]–[65].

Diffusion MRI is particularly relevant to a wide range of clinical investigations related, for example, to brain ischemia detection [56], stroke, Alzheimer disease, or schizophrenia [1]. It is also extremely useful in order to identify the neural connectivity patterns of the human brain [6], [12], [35], [44].

Most of the existing techniques addressing this last issue work on a fiber-wise basis. In other words, they do not take into account the global coherence that exists among fibers of a given tract. Recent work by Corouge et al. [18] has proposed to cluster and align fibers by local shape parameterization so that a statistical analysis of the tract geometrical and physiological properties can be carried out. This paper relies on the extraction of a set of streamlines from diffusion tensor images (DTI) by the method proposed in [44] which is known to be sensible to noise and unreliable in areas of fiber crossings.

For these reasons, we propose to directly perform the segmentation of DTI in order to extract neural fiber bundles. While many techniques have been proposed to classify the gray matter, white matter and cephalo-spinal fluid from T1-weighted MR images (see [69] for example), the literature addressing the segmentation of white matter structures from DTI is still new. We hereafter draw a quick state of the art of the DTI segmentation problem:

Zhukov et al. [71] defined an invariant anisotropy measure in order to drive the evolution of a level-set and isolate strongly anisotropic regions of the brain. The reduction of the full tensor to a single scalar value can result in a relatively low discrimination capability, potentially yielding the segmentation of mixed structures. Alternatively, Wiegell et al. [66], Feddern et al. [24], [25], Rousson et al. [54], Wang et al. [61], and [62], Lenglet et al. [36], and Jonasson et al. [31] use or propose different measures of dissimilarity between diffusion tensors. In [61], [66], and [54], the authors use the Frobenius norm of the difference of tensors (i.e., the Euclidean distance). A $k$-means algorithm with a spatial coherence constraint and an active contour model with a regularity term were respectively used by the first two methods ([61] and [66]) to perform the segmentation of different cerebral structures such as the thalamus nuclei or the corpus callosum. The third method [54] used a region-based surface propagation. In [61], a generalization of the region-based active contours to matrix-valued images is proposed. However, it is restricted to the two-dimensional (2-D) case and obviously limited when it comes to 3-D brain data.
ential equations based on mean curvature motion, self-snakes and geodesic active contour models are extended to 2-D and 3-D tensor-valued images by generalizing the notion of structure tensor to matrix-valued data. This method still relies on the Euclidean metric between tensors. The authors apply this framework to the regularization and segmentation of DTI. In [31], the authors introduce a geometric measure of dissimilarity by computing the normalized tensor ‘scalar product’ of two tensors, which can be interpreted as a measure of overlap. Finally, the methods exposed in [62] and [36] rely on the symmetrized Kullback-Leibler divergence to derive an affine invariant dissimilarity measure between diffusion tensors.

Contributions: Our contributions are threefold: First, we recast the DTI segmentation problem into a unified statistical surface evolution framework. We also make use of the tensor field gradient to detect boundaries between various structures of the white matter. This framework can be implemented with different probability metrics. This is done for the Euclidean distance, Kullback-Leibler divergence and geodesic distance on the manifold of multivariate normal distributions. The second contribution is related to the development of a rigorous differential geometrical framework, as presented in [39], rooted in the information geometry and used to express a Gaussian law between diffusion tensors. We overcome the classical hypothesis considering covariance matrices as a linear space and define relevant statistics to model the distribution of diffusion tensors. To that end, we also extend the methods proposed in [62] and [36] by showing how to compute the covariance matrix, associated to the Kullback-Leibler divergence, of a set of tensors. Finally, we demonstrate that the properties of the geodesic distance lead to its superiority, for our segmentation task, over the other two dissimilarity measures. This is achieved by presenting results on both synthetic and real data-sets as well as on a biological phantom, for which only this method succeeds by comparison with the ground truth or neuroanatomical knowledge.

Organization of the Paper: Section II describes how to approximate a Gaussian distribution between diffusion tensors, in other words how to compute a mean tensor and a $6 \times 6$ covariance matrix. It also presents how to evaluate the norm of a diffusion tensor belonging to a $6$-dimensional submanifold of $\mathbb{R}^6$ which can be endowed with a Riemannian metric. This general characterization of $S^+(3)$ will be very useful to derive statistics on diffusion tensors based on different probability metrics.

A. Riemannian Geometry Basics

1) Metric, Geodesics, Distance: For an $n$-dimensional manifold $\mathcal{M}$, a Riemannian metric is a collection of inner products $\langle \cdot, \cdot \rangle_p$ defined for every point $p$ of $\mathcal{M}$. These inner products are defined on the tangent space $T_p \mathcal{M}$ of $\mathcal{M}$ at $p$ and provide a natural way to measure the lengths of vectors tangent to $\mathcal{M}$ at location $p$. We call tangent vector an element of a tangent space which is simply a vector space (a copy of $\mathbb{R}^n$) attached to each point $p \in \mathcal{M}$.

A good example of tangent vector at $p$ is the case of the derivative $\dot{\gamma}(t_0) = \frac{d\gamma(t_0)}{dt}$ of a curve $\gamma: I = [t_0, t_1] \subset \mathbb{R} \mapsto \mathcal{M}$ passing through $\gamma(t_0) = p$ (Fig. 1).

It is possible to introduce a map $\phi: \mathcal{M} \mapsto U \subset \mathbb{R}^n$, known as a coordinate chart, that defines a local coordinate system $\phi(p) = x = (x^1, \ldots, x^n)^T$ and a basis of the tangent space $T_p \mathcal{M}$ denoted by $(\partial/\partial x_1, \ldots, \partial/\partial x_n)$. Any element of the tangent space can, hence, be expressed in the form $\sum_i x^i \partial/\partial x_i$ and the inner products $(\partial/\partial x_i, \partial/\partial x_j)$ define an $n \times n$ symmetric, bilinear and positive-definite matrix $G = g_{ij}$ known as the local representation of the Riemannian metric. The inner product of two tangent vectors $u$ and $v$ is then expressed as

$$\langle u, v \rangle_p = u^T G v$$

(the reference to the location $p$ is usually discarded in notation $g_{ij}$).

Equipped with these notions we can now define the concept of geodesic on a Riemannian manifold $\mathcal{M}$. It is the equivalent of straight line in Euclidean spaces and defined as the locally length-minimizing curve $\gamma: I \subset \mathbb{R} \mapsto \mathcal{M}$. The tangent vector $\dot{\gamma}(t)$ defines the instantaneous speed of the curve and its norm $|\dot{\gamma}(t)| = \langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_p^{1/2}$ is the instantaneous velocity. Integrating $|\dot{\gamma}(t)|$ along $\gamma$ yields its length which is also the geodesic distance between the two endpoints $p_1$ and $p_2$ of the curve

$$D(p_1, p_2) = \int_{t_1}^{t_2} |\dot{\gamma}(t)| \, dt.$$

Finally, taking $I = [0, 1]$ for simplicity, it is possible to show, under certain assumptions that will be met in the following, that a geodesic $\gamma: [0, 1] \mapsto \mathcal{M}$ is uniquely defined by its starting point $\gamma(0)$ and its initial velocity $\gamma(0) \in T_{\gamma(0)} \mathcal{M}$. The endpoint
\(\gamma(1)\) can be easily computed by applying the exponential map at \(\gamma(0)\) to \(\gamma'(0)\): \(\gamma(1) = \text{exp}_{\gamma(0)}(\gamma'(0))\). A detailed presentation of this map can be found in [22]. The inverse map, known as the logarithm map of \(\gamma(1)\) at \(\gamma(0)\) : \(\text{log}_{\gamma(0)}(\gamma(1))\), yields the unique tangent vector \(\gamma'(0)\) if we know the two endpoints of the curve. Moreover, it can be proved that

\[
D(\gamma(0), \gamma(1)) = \langle \gamma'(0), \gamma'(0) \rangle_{\gamma(0)^{\frac{1}{2}}}. 
\]

In this paper, we will use the fact that the velocity \(\gamma'(0)\) can be computed from the gradient of the squared geodesic distance with respect to \(\gamma(0)\). In other words, we have

\[
\gamma'(0) = -\nabla_{\gamma(0)} D^2(\gamma(0), \gamma(1)). 
\]

Using this definition, we can now define the notions of mean and covariance matrix on a Riemannian manifold. They will play a central role in the variational formulation [see (19) and (21)] of the segmentation problem to be detailed in the Section III. We also show how to compute the norm of the spatial gradient of a tensor field which will be useful to introduce a boundary term in our segmentation energy [see (20)].

*Statistics:* As defined by Fréchet in [28] and used by Pennec in [49], the empirical mean of a set of \(N\) random elements \(\{p_i\}_i\), \(i = 1, \ldots, N\) of \(M\), such as diffusion tensors, is defined as the minimizer \(p = \bar{p}\) of the variance \(\sigma^2_{\bar{p}}(\{p_i\})\) of the \(p_i\) with respect to \(p\)

\[
\sigma^2_{\bar{p}}(\{p_i\}) = \mathbb{E} [D^2(p, p_i)] = \frac{1}{N} \sum_{i=1}^{N} D^2(p, p_i). \quad (1)
\]

The empirical covariance matrix of the set \(\{p_i\}\), with respect to the mean \(\bar{p}\) is defined as the expected value of the quantity \(\beta_i^T T^T \beta_i\) and denoted by \(\Lambda_{\bar{p}}\). As depicted in Fig. 2, \(\beta_i\) is the initial velocity \(\gamma'(0)\) of the \(i\)th geodesic joining \(\gamma'(0) = \bar{p}\) to \(p_i\) and expressed in local coordinates, i.e., it is taken to be the \(n\)-dimensional vector of coordinates \(\varphi(\gamma'(0))\) and not the tangent vector \(\varphi'(0)\) itself. The dot product then boils down to a simple Euclidean dot product and we have

\[
\Lambda_{\bar{p}} = \frac{1}{N} \sum_{i=1}^{N} \varphi(\beta_i) \varphi(\beta_i)^T \text{ with } \beta_i = -\nabla_{\bar{p}} D^2(p_i, p_i) \quad (2)
\]

where \(\varphi\) is the coordinate chart introduced in Section II-A1 and also used in Sections II-B1, II-B2, and II-B3. In Section II-B, we will apply these definitions to the Euclidean, Kullback-Leibler and geodesic probability metrics in order to approximate Gaussian distributions of diffusion tensors based on these dissimilarity measures. In particular, we will show how the gradient of the squared distances can be computed and used to estimate the associated covariance matrices [see (2)] as well as the empirical mean tensor. We will then evaluate and compare their respective virtue for our segmentation purpose.

Spatial Gradient: We recall that we are interested in images \(\Sigma\) associating to each location of a regular sampling \(\Omega\) of \(\mathbb{R}^3\) an element of \(S^+(3)\). The spatial gradient of \(\Sigma\) can be estimated from the gradient of the squared distance as

\[
\nabla_{\Sigma(x)} \Sigma(x) = -\frac{1}{|\epsilon_k|} \left( \nabla_{\Sigma(x)} D^2(\Sigma(x), \Sigma(x + \epsilon_k)) \right)
\]

where the \(\epsilon_k\), \(k = 1, 2, 3\) denote the canonical basis of \(\mathbb{R}^3\) and are used to access the neighbors of \(\Sigma(x)\) on the grid \(\Omega\). If \(\epsilon_k\) is either \(+1\) or \(-1\) it denotes the forward and backward approximation of the gradient. \(\nabla_{\Sigma(x)} \Sigma(x)\) is, for example, the initial tangent vector of the geodesic joining \(\Sigma(x)\) and \(\Sigma(x - \epsilon_k [1, 0, 0]^T)\).

It is then straightforward to compute the squared norm of the gradient at location \(x\) as

\[
|\nabla_{\Sigma(x)} \Sigma(x)|^2 = \frac{1}{2} \sum_{k=1}^{3} \sum_{\epsilon \in \{-1, 1\}} |\nabla_{\Sigma(x)} \Sigma(x + \epsilon \epsilon_k)|^2
\]

where the \(\frac{1}{2}\) factor arises from the fact that we use \(3 \times 3 \times 3\) neighborhoods.

We now use the fact that statistics and gradient norm can be computed from the distance \(D\) and its gradient. We endow the space \(S^+(3)\) with different probability metrics (i.e., distances) and derive the associated statistics and gradient norms which will be used in Section III respectively to model the distribution of tensors within a subset of a DTI and to detect the interface between white matter structures.

### B. Derivations of Statistics and Gradient Norms

As we will see shortly, the manifold \(M\) of 3-D normal distributions with zero-mean can be identified with the manifold \(S^+(3)\) of \(3 \times 3\) real, symmetric, positive-definite matrices which provides a natural means of parameterizing those probability density functions. Ultimately, we will use the fact that the Fisher information matrix corresponds to the Riemannian metric on this manifold (see [23] for example) and induces a geodesic distance \(D_{\text{g}}\). However, other distances between parameterized normal distributions (i.e., between covariance matrices and, hence, diffusion tensors) have been introduced. We will first use the Euclidean distance \(D_{\text{e}}\), then exploit the properties of the symmetrized Kullback-Leibler divergence \(D_{\text{f}}\).
also known as the $J$-divergence [29], and finally describe the geometry of $S^+(3)$ equipped with a metric derived from the Fisher information matrix.

1) Euclidean Probability Metric: We consider $S^+(3)$ with the simple Euclidean metric. In this case, the dissimilarity measure between diffusion tensors is given by the Frobenius norm of the difference such that for all $A, B \in S^+(3)$, we have

$$D_e(A, B) = |A - B|_F = \sqrt{\text{tr}((A - B)(A - B)^T)}$$

(3)

where $\text{tr}$ denotes the trace operator. Using the fact that $\nabla_X \text{tr}(XY) = Y^T$ for $X, Y \in GL(n)$, it is easy to see that

$$\nabla_A D^2_e(A, B) = A - B.$$  

(4)

In other words, we find that the gradient of the squared Euclidean distance corresponds to the usual difference tangent vector. This is a symmetric matrix which can be used to compute the $6 \times 6$ covariance matrix (2) of a set of $N$ diffusion tensors.

Plugging (3) into (1), the empirical mean diffusion tensor is estimated as

$$\Sigma_e = \frac{1}{N} \sum_{i=1}^{N} \Sigma_i$$

where we denote by $\Sigma_i$ the tensor located at voxel $x_i$ in $\Omega$. The associated covariance matrix is obtained as

$$\Lambda_e = \frac{1}{N} \sum_{i=1}^{N} \varphi(\Sigma_i - \Sigma_e)\varphi(\Sigma_i - \Sigma_e)^T$$

The map $\Phi : S^+(3) \mapsto \mathcal{G}^6$ associates to each symmetric matrix $\varphi = \Sigma_i - \Sigma_e$ its 6 independent components. In this Euclidean setting, we can define a Gaussian distribution between diffusion tensors with the probability function

$$p_e(\Sigma, \Sigma_e, \Lambda_e) = \frac{1}{\sqrt{(2\pi)^6|\Lambda_e|}} \exp\left(-\frac{\varphi(\beta)^T \Lambda_e^{-1} \varphi(\beta)}{2}\right)$$

(5)

with $\beta = \Sigma - \Sigma_e$. We will use this expression, in the Euclidean case, for the probability distributions $p_{\text{cov/ord}}$ in (19) of Section III-C. Finally, the squared norm of the spatial gradient of a DTI $\Sigma : \Omega \mapsto S^+(3)$ is given by

$$|\nabla\Sigma_e|^2 = \frac{1}{2} \sum_{k=1}^{3} \sum_{s=-1}^{1} \text{tr}\left((\Sigma(x) - \Sigma(x + se_k))^T \Sigma(x + se_k)\right)$$

(6)

and is used in the distribution $p_{\text{g}}$ defined by (20) for the Euclidean case.

2) J-Divergence Probability Metric: We now adopt a more information-theoretic point of view and consider another dissimilarity measure between Gaussian probability densities known as the Kullback-Leibler divergence $D_k$ or relative entropy. This probability metric has the desirable property of being invariant under affine transformation of the density parameters, hence, it is invariant under congruence transformations such that

$$D_k(A, B) = D_k(XAX^T, XBX^T), \quad \forall A, B \in S^+(3), X \in GL(3).$$  

(7)

This property does not hold for the Euclidean distance previously introduced. The Kullback-Leibler divergence is defined for parametric as well as nonparametric densities. In (7), $A$ and $B$ actually stand for the covariance matrices of 3-D normal distributions $\mathcal{N}(r|A)$ and $\mathcal{N}(r|B)$ with zero-mean and we have

$$D_k(A, B) = \int_{\mathbb{R}^3} \mathcal{N}(r|A) \log \frac{\mathcal{N}(r|A)}{\mathcal{N}(r|B)} dr.$$  

We recall that diffusion tensors are indeed the parameters of Gaussian distributions $\mathcal{P}$ modeling the local displacement $r$ of water molecules.

It turns out, however, that the Kullback-Leibler divergence is not symmetric and, hence, not a true metric. We will use, as in [62], its symmetrized version, or J-divergence

$$\frac{1}{2} \int_{\mathbb{R}^3} \mathcal{P}(r|A) \log \frac{\mathcal{P}(r|A)}{\mathcal{P}(r|B)} + \mathcal{P}(r|B) \log \frac{\mathcal{P}(r|B)}{\mathcal{P}(r|A)} dr.$$  

As we will see in the next section, the $J$-divergence is closely related to the squared geodesic distance on $S^+(3)$ induced by the Fisher information matrix but only coincides with the latter for special probability densities. Hence, it is natural to define

$$D_j(A, B) = \sqrt{\frac{1}{2} (D_k(A, B) + D_k(B, A))}$$

As stated in [67] and used in [62], the expression of this distance is particularly simple when $\mathcal{P}$ is a 3-D Gaussian density

$$D_j(A, B) = \sqrt{\frac{1}{4} \text{tr}(A^{-1}B + B^{-1}A) - 6}.$$  

(8)

We have the following proposition.

Proposition 2.1: The gradient of the squared distance $D^2_j$ between 3-D normal distributions parameterized by their covariance matrix $A, B \in S^+(3)$ is

$$\nabla_A D^2_j(A, B) = \frac{1}{4}(B^{-1} - A^{-1}BA^{-1}).$$  

(9)

Proof: This comes from the fact that $\nabla_A \text{tr}(B^{-1}A) = \nabla_A \text{tr}(A^TB^{-T}) = B^{-T} = B^{-1}$ and that $\nabla_A \text{tr}(A^{-1}B) = -(A^{-1}BA^{-1})^{-T} = -A^{-1}BA^{-1}$.
From this result, we are able to compute the covariance matrix (2) of a set of diffusion tensors. We just need to define the empirical mean diffusion tensor (1) associated to the distance $D_j$ (8). This was already proposed in [62] as the following theorem.

**Theorem 2.1:** The empirical mean diffusion tensor of a set of $N$ tensors $\{\Sigma_i\}, i = 1, \ldots, N$ is given by

$$\Sigma_j = \arg\min_{\Sigma \in S^+(3)} \frac{1}{N} \sum_{i=1}^{N} D_j^2(\Sigma, \Sigma_i) = V^{-\frac{1}{2}} \left(U^T \Sigma U \right)^{\frac{1}{2}} V^{-\frac{1}{2}}$$

with $U = (1/N) \sum_{i=1}^{N} \Sigma_i$ and $V = (1/N) \sum_{i=1}^{N} \Sigma_i^{-1}$

The associated covariance matrix is obtained as

$$A_j = \frac{1}{N} \sum_{i=1}^{N} \varphi(\beta_i) \varphi(\beta_i)^T$$

where, once again, the map $\varphi$ associates to each symmetric matrix $\beta_i = -(1/4)(\Sigma_i^{-1} - \Sigma_i^{-1} \Sigma_j \Sigma_i^{-1})$ its 6 independent components. In this information-theoretic setting, we now define a Gaussian distribution between diffusion tensors with the probability function

$$p_j(\Sigma, \Sigma_j, A_j) = \frac{1}{\sqrt{(2\pi)^6 |A_j|}} \exp \left( - \frac{\varphi(\beta_j)^T A_j^{-1} \varphi(\beta_j)}{2} \right)$$

(10)

with $\beta_i = -(1/4)(\Sigma_i^{-1} - \Sigma_i^{-1} \Sigma_j \Sigma_i^{-1})$. We will use this expression, in the $J$-divergence case, for the probability distributions $p_{in/out}$ in (19) of Section III-C. Finally, we can easily obtain the snared norm of the spatial gradient of a DTI $\Sigma$ as

$$|\nabla \Sigma(x)|_j^2 = \frac{1}{2} \sum_{k=1}^{3} \sum_{s=\pm 1} \left( \frac{1}{4} \text{tr} \left( \Sigma(x)^{-1} \Sigma(x + s \delta_k) \right) + 2 \text{tr} \left( \Sigma(x)(\Sigma(x + s \delta_k)^{-1} - I_3) \right) \right)$$

(11)

and use it in the distribution $p_N$ of (20) for the J-divergence case.

3) Geodesic Probability Metric: We introduce, as in [39], a last dissimilarity measure between diffusion tensors, which we claim to be more natural and powerful for the comparison of 3-D normal distributions. Its superiority will be demonstrated through the numerical experiments presented in Section IV.

Following [51] and [10], it is possible to define a Riemannian metric on $S^+(3)$ in terms of the Fisher information matrix. The Fisher information is a popular measure of the amount of information carried by the realizations of a random variable about the unknown parameters of the underlying probability density. This is classically used to derive maximum likelihood estimators of density parameters. Once again, we use the natural chart $\varphi$ of $S^+(3)$ such that for all $A \in S^+(3)$, we have $\varphi(A) = (x_1, x_2, x_3, x_4, x_5, x_6)^T$. The tangent space $T_A S^+(3)$ at $A \in S^+(3)$ coincides with $S(3)$, the space of $3 \times 3$ real, symmetric matrices. Its basis is denoted by $(\partial_1, \ldots, \partial_6)$.

We now detail the fundamental properties of $S^+(3)$ and propose an original formulation for a Gaussian law on this manifold. The fundamental tools needed to derive our numerical schemes were detailed in [9], [11], [23], [27], [37], [43], [46], [55], and [40]. Other recent works, such as [50] and [26] do not employ the information geometry associated with the Fisher information matrix but rather consider $S^+(3)$ as the quotient space $GL^+(3)/SO(3)$ to derive statistical or filtering tools on tensor fields.

The Fisher information matrix

$$g_{ij} = \int \frac{\partial \log \mathcal{F}(r|A)}{\partial \xi^i} \frac{\partial \log \mathcal{F}(r|A)}{\partial \xi^j} \mathcal{F}(r|A) dr$$

takes the following form for $S^+(3)$

**Theorem 2.2:** The Riemannian metric for the space of 3-D normal distributions with zero-mean, $S^+(3)$ is given, for all $A \in S^+(3)$ by

$$g_{ij} = \langle \partial_i, \partial_j \rangle_A = \frac{1}{2} \text{tr} (A^{-1} \partial_i A^{-1} \partial_j) \quad i, j = 1, \ldots, 6$$

In practice, this means that for any tangent vectors $V_1, V_2 \in S(3)$, their inner product at $A$ is given by

$$\langle V_1, V_2 \rangle_A = \frac{1}{2} \text{tr} (A^{-1} V_1 A^{-1} V_2).$$

Below are two examples of the metric tensor $G$, respectively computed for $A_1 = \mathcal{F}$ and $A_2 = \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2)$ with $\mathcal{F}$ and $\text{diag}(\cdot)$ denoting the identity and diagonal matrices. They correspond to a locally isotropic diffusion process and to the more general case of an anisotropic diffusion, with variances $\sigma_1^2$, $\sigma_2^2$ and $\sigma_3^2$, whose principal axes coincide with the coordinate frame of the image

$$G_{A_1} = \begin{pmatrix}
\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2}
\end{pmatrix}$$

and

$$G_{A_2} = \begin{pmatrix}
\frac{1}{2\sigma_1^2} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2\sigma_2^2} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2\sigma_3^2} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2\sigma_1^2} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2\sigma_2^2} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2\sigma_3^2}
\end{pmatrix}.$$
As stated for example in [43], the geodesic starting from \( \gamma(0) = A \in S^+(3) \) in the direction \( \dot{\gamma}(0) \in S(3) \) is given by

\[
\gamma(t) = A^\frac{1}{2} \exp \left( tA^{-\frac{1}{2}} \dot{\gamma}(0) A^{-\frac{1}{2}} \right) A^\frac{1}{2} \quad \forall t \in [0, 1]. \tag{12}
\]

We recall that the geodesic distance \( D_g \) between any two elements \( A \) and \( B \) is the length of the minimizing geodesic between these points

\[
D_g(A, B) = \inf_{\gamma} \{ L(\gamma) : A = \gamma(0), B = \gamma(1) \}
\]

It is given by the following theorem, whose original proof is available in an appendix of [2] but different versions can also be found in [55] and [27].

**Theorem 2.3 (S.T. Jensen, 1976):** Consider the family of multivariate normal distributions with common mean vector but different covariance matrices. The geodesic distance between two members of the family with covariance matrices \( A, B \in S^+(3) \) is given by

\[
D_g(A, B) = \sqrt{\frac{1}{2} \text{tr} \left( \log^2 \left( A^{-\frac{1}{2}} B A^{-\frac{1}{2}} \right) \right)} = \sqrt{\frac{1}{2} \sum_{i=1}^{3} \log^2(\eta_i)} \tag{13}
\]

where \( \eta_i \) denote the three eigenvalues of the matrix \( A^{-1/2} B A^{-1/2} \).

Apart from being a true distance, hence, being positive, symmetric and verifying the triangle inequality (see [27] although no complete proof of the triangle inequality was provided by the authors), this distance is also invariant under congruence transformation (i.e., affine invariant) as well as under inversion.

It is interesting, at this stage, to study the relationship between this geodesic distance and the J-divergence. As summarized in [3], given suitable technical conditions on two nearby densities \( \mathcal{P}(r|A) \) and \( \mathcal{P}(r|A + dA) \), the zeroth and first order terms of a Taylor expansion of the Kullback-Leibler divergence around \( \mathcal{P}(r|A) \) vanish. Assuming second-order differentiability of \( D_{kl} \), a second order expansion of \( D_{kl}(A, A + dA) \) yields

\[
\frac{1}{2} \int_{\mathcal{B}^3} \left( \frac{1}{\mathcal{P}(r|A)} \frac{\partial \mathcal{P}(r|A)}{\partial x_i} \right) \frac{\partial \mathcal{P}(r|A)}{\partial x_j} \mathcal{P}(r|A) dx^i dx^j dr
\]

\[
- \frac{1}{\mathcal{P}(r|A)} \frac{\partial \mathcal{P}(r|A)}{\partial x_i \partial x_j} \mathcal{P}(r|A) dx^i dx^j
\]

which can be shown to reduce to

\[
D_{kl}(A, A + dA) = \frac{1}{2} \delta \left[ \frac{\partial \log \mathcal{P}(r|A)}{\partial x^i} \frac{\partial \log \mathcal{P}(r|A)}{\partial x^j} \right] dx^i dx^j
\]

(if the partial derivatives commute with the integral) and which is precisely half of the squared geodesic distance between \( A \) and \( A + dA \). Consequently, it is easy to see that the J-divergence coincides, up to the second order, with half of the squared geodesic distance between two nearby diffusion tensors. Whenever the tensors are not infinitesimally close, the two distances become inconsistent. This is another reason supporting our claim that diffusion tensors statistics based on the geodesic distance should improve the quality of DTI segmentation results.

It was shown in [43] that the gradient of the squared geodesic distance writes

\[
\nabla_A D_g^2(A, B) = A \log(B^{-1} A). \tag{14}
\]

Based on this result and on the following method for the computation of the mean tensor in our Riemannian setting, we will be able to estimate the covariance matrix (2) of a set of diffusion tensors \( \{\Sigma_i\}, i = 1, \ldots, N \), and, finally, approximate a Gaussian distribution on \( S^+(3) \). As presented in [40], a closed-form expression for the empirical mean (1) cannot be obtained but a gradient descent algorithm was proposed. It estimates a quantity, known as the Riemannian barycenter, which exists and is unique for manifolds of nonpositive sectional curvature (see [32]) like \( S^+(3) \). The algorithm is based on the minimization of the variance of the \( \Sigma_i \). It can be shown that this boils down to evolving an initial guess of the mean (like the identity matrix \( J \)) along the geodesics of \( S^+(3) \) [see (12)] with a velocity given by the gradient of the variance, i.e., a tangent vector \( V \) such as

\[
V = -\frac{1}{N} \sum_{i=1}^{N} \nabla_M D_g^2(M, \Sigma_i) = -\frac{1}{N} M \sum_{i=1}^{N} \log(\Sigma_i^{-1} M)
\]

where \( M \) denotes the evolving mean tensor. After a few iterations of this procedure, \( M \) converges toward the mean tensor \( \Sigma_g \). We describe this procedure in the Algorithm I.

**Algorithm I Riemannian estimation of the mean diffusion tensor**

Require: \( \{\Sigma_i\} \in S^+(3), i = 1, \ldots, N \) and \( n \) it, the number of iterations

Ensure: \( \Sigma_g \), the mean tensor

1: \( M \leftarrow J \)
2: for \( k = 1 \) to \( n \) it do
3: \( V \leftarrow (3 \times 3 \) zero matrix)
4: for \( i = 1 \) to \( N \) do
5: \( V \leftarrow \log(\Sigma_i^{-1} M) \)
6: end for
7: \( V \leftarrow (1/N) M V \)
8: \( M \leftarrow M^{1/2} \exp\left(-M^{-1/2} V M^{-1/2}\right) M^{1/2} \)
9: end for
10: Return \( M \)

The associated covariance matrix is obtained as

\[
\Lambda_g = \frac{1}{N} \sum_{i=1}^{N} \varphi(\beta_i) \varphi(\beta_i)^T
\]

where \( \beta_i = -\Sigma_i \log(\Sigma_i^{-1} \Sigma_g) \) and \( \varphi \) associates to each \( \beta_i \) its six independent components. The notion of Gaussian distribution was generalized to random samples of primitives belonging to a Riemannian manifold in [49] where more details can be found regarding this particular point. From this work, we have proposed in [40] a definition of the Gaussian law between diffusion tensors which can be approximated as follows for a covariance matrix \( \Lambda_g \) of small variance \( \sigma^2 = \text{tr}(\Lambda_g) \):

\[
p_g(\Sigma_i | \Sigma_g, \Lambda_g) \approx \frac{1}{\sqrt{(2\pi)^6 |\Lambda_g|}} \exp \left( -\varphi(\beta_i)^T \Lambda_g \varphi(\beta_i) \right) \tag{15}
\]
where $\beta_i$ is defined as $\beta_i = -\frac{1}{N} \sum_j \log(\Sigma_{ij})$ and the concentration matrix is $\Gamma \simeq \Lambda_{\Sigma}^{-1} - R/3$, with $\Lambda_{\Sigma}$ the Ricci tensor at the mean $\Sigma_{\bar{x}}$. The computation of the Ricci tensor $\mathcal{R}$ can be performed on the basis of closed-form expressions for the metric and the Riemann tensor provided in [55] and simply involving traces of matrix products. As we will point out in Section IV, our numerical experiments have shown that the Ricci tensor exhibits a difference of at least 2 orders of magnitude with the inverse of the covariance matrix. Hence, we can approximate $\Gamma$ by $\Lambda_{\Sigma}^{-1}$.

We will use $p_g$, in the geodesic case, for the probability distributions $p_{in/out}$ in (19) of Section III-C. Finally, the squared norm of the spatial gradient of a DTI can be estimated as follows:

$$|\nabla \Sigma(x)|^2_g = \frac{1}{2} \sum_{k=1}^{3} \sum_{x \in \pm 1} \left( \frac{1}{2} \text{tr} \left( \log^2 \left( \Sigma(x) - \frac{1}{N} \sum_j \Sigma_{ij} \right) \right) \right)$$

(16)

and subsequently used in the distribution $p_b$ of (20) in the geodesic case.

4) Summary and Numerical Examples: We summarize, in Table I, the expressions of the squared distance, its gradient, and of the mean tensor for the Euclidean, $J$-divergence and geodesic cases. The evaluation of the squared distance and its gradient for the matrices $A_1$ and $B_1$ respectively given below shows a good coherence (although the Euclidean distance is quite larger than the other two) and, more importantly, illustrates the fact that the $J$-divergence accurately approximates half of the squared geodesic distance when the tensors are relatively close:

$$\begin{pmatrix} 0.0878 & -0.0027 & 0.0050 \\ -0.0027 & 1.0112 & -0.0372 \\ 0.0050 & -0.0372 & 1.0391 \end{pmatrix}, \begin{pmatrix} 1.0384 & -0.0012 & 0.0107 \\ -0.0012 & 1.0051 & 0.0000 \\ 0.0107 & 0.0000 & 1.0233 \end{pmatrix}$$

$D_g^2(A_1, B_1) = 0.010158$

$$\begin{pmatrix} -0.0006 & -0.0015 & -0.0057 \\ -0.0015 & 0.0006 & -0.0312 \\ -0.0057 & -0.0312 & 0.0158 \end{pmatrix}, \begin{pmatrix} -0.0274 & -0.0037 & 0.0040 \\ -0.0037 & 0.0002 & -0.0447 \\ 0.0040 & -0.0447 & 0.0066 \end{pmatrix}$$

$D_g^2(A_1, B_1) = 0.004256$

$$\begin{pmatrix} -0.0024 & -0.0066 & -0.0040 \\ -0.0066 & 0.0002 & -0.0447 \\ -0.0040 & -0.0447 & 0.0066 \end{pmatrix}, \begin{pmatrix} 0.0000 & 0.0015 & -0.0033 \\ 0.0015 & 0.0003 & -0.0048 \\ -0.0033 & -0.0048 & 0.0084 \end{pmatrix}$$

$D_g^2(A_1, B_1) = 0.000950$

$$\begin{pmatrix} -0.0003 & 0.0074 & -0.0034 \\ 0.0003 & -0.0034 & 0.0084 \end{pmatrix}, \begin{pmatrix} -0.0048 & -0.0034 & 0.0084 \end{pmatrix}$$

$D_g^2(A_1, B_1) = 0.000050$

On the contrary, if we consider the matrices $A_2$ and $B_2$, which are much more different than $A_1$ and $B_1$, we find out that the $J$-divergence becomes sensibly different from half of the squared geodesic distance

$$\begin{pmatrix} 1.0096 & -0.0053 & 0.0405 \\ -0.0053 & 0.0621 & 0.1068 \\ 0.0405 & 0.1068 & 1.4056 \end{pmatrix}, \begin{pmatrix} 0.0327 & 0.1965 & 0.0932 \\ 0.1965 & 0.0327 & 0.0932 \end{pmatrix}$$

$D_g^2(A_2, B_2) = 1.111446$

$$\begin{pmatrix} -0.2117 & -0.2883 & 0.3708 \\ -0.2883 & 0.7190 & -0.0897 \\ 0.3708 & -0.0897 & 0.4095 \end{pmatrix}, \begin{pmatrix} -0.2079 & -0.2875 & 0.4483 \\ -0.2875 & 0.4483 & -0.0799 \\ 0.4483 & -0.0799 & 0.0295 \end{pmatrix}$$

$D_g^2(A_2, B_2) = 0.329119$

$$\begin{pmatrix} -0.2079 & -0.2875 & 0.4483 \\ -0.2875 & 0.4483 & -0.0799 \\ 0.4483 & -0.0799 & 0.0295 \end{pmatrix}, \begin{pmatrix} 0.1765 & 0.0783 & 0.0880 \\ 0.0783 & 0.1765 & 0.0880 \end{pmatrix}$$

$D_g^2(A_2, B_2) = 0.062150$

$$\begin{pmatrix} -0.2079 & -0.2875 & 0.4483 \\ -0.2875 & 0.4483 & -0.0799 \\ 0.4483 & -0.0799 & 0.0295 \end{pmatrix}, \begin{pmatrix} -0.1508 & -0.4424 & -0.0799 \\ -0.4424 & 0.1508 & 0.4483 \\ -0.0799 & 0.4483 & -0.1508 \end{pmatrix}$$

$D_g^2(A_2, B_2) = 0.4483$

Now, in order to compare the statistics derived from each distance, we have manually segmented the genu of the corpus callosum on a DTI data-set used in the last section of this paper. This is a well-known region of the brain white matter (Fig. 3) where fibers are essentially aligned in a right-left fashion, i.e., along the $x_3$ axis on an axial slice. Consequently, the tensors in this region are very anisotropic with a major eigenvector close to $(1, 0, 0)^T$. This resulted in a set $\{\Sigma_i\}$ of $N = 614$ tensors. The ellipsoids presented in the bottom-right corner of Fig. 3 represent the mean tensor respectively computed, from left to right, with the Euclidean distance, $J$-divergence and geodesic distance. Visually, we can see that the Euclidean mean is somehow more oblate than the other two ellipsoids. This can be explained.

![Fig. 3. Statistics in the genu of the corpus callosum.](image-url)
by the fact that Euclidean averaging is blind to the spectral
components of the tensors (eigenvalues and eigenvectors) and has a
tendency to mix them. We now present the estimated statistics
for each distance (We scaled by a factor 2 the values obtained
for the \( J \)-divergence to make the comparisons easier).

1) Euclidean probability metric

\[
\sum_e = \begin{pmatrix}
0.0293 & -0.1334 & 0.0347 \\
0.0434 & 1.2047 & 0.0526 \\
2.8085 & -0.2261 & 0.0580 \\
-0.2261 & 0.178 & -0.0016 & -0.0527 & -0.0070 & -0.0161 \\
0.0293 & -0.0046 & 0.0012 & 0.0017 & 0.0018 & 0.0042 \\
0.0090 & -0.0527 & 0.137 & 0.1560 & 0.2008 & 0.0485 \\
0.0092 & -0.0700 & 0.0018 & 0.0298 & 0.0027 & 0.0064 \\
0.2083 & -0.0164 & 0.0042 & 0.0485 & 0.0064 & 0.0151
\end{pmatrix}, \quad \text{tr}(\Lambda_e) = 3.0615
\]

2) \( J \)-divergence probability metric

\[
\sum_j = \begin{pmatrix}
2.2091 & -0.1063 & 0.0296 \\
-0.1063 & 1.0833 & 0.0455 \\
0.0296 & 0.0455 & 0.8776 \\
0.0369 & 0.0075 & -0.0015 & 0.0339 & -0.0105 & 0.0816 \\
0.0075 & 0.0022 & 0.0035 & 0.0137 & 0.0051 & 0.0226 \\
-0.0015 & 0.0035 & 0.0113 & -0.0335 & 0.0019 & -0.0142 \\
0.0139 & 0.0031 & -0.0135 & -0.0458 & 0.0465 & 0.5147 \\
-0.0015 & -0.0001 & 0.0010 & -0.0040 & 0.0432 & -0.0552 \\
0.0816 & 0.0236 & -0.0142 & 0.5147 & -0.0052 & 0.8256
\end{pmatrix}, \quad \text{tr}(\Lambda_j) = 1.5161
\]

3) Geodesic probability metric

\[
\sum_g = \begin{pmatrix}
2.3296 & -0.1088 & 0.0312 \\
-0.1088 & 1.1102 & 0.0523 \\
0.0312 & 0.0523 & 0.8812 \\
0.7706 & -0.0207 & 0.0207 & -0.0003 & -0.0207 & 0.1570 \\
-0.0207 & 0.3516 & 0.0431 & -0.0061 & 0.0016 & -0.0041 \\
0.0207 & 0.0431 & 0.1332 & -0.0013 & 0.0086 & 0.0073 \\
-0.0003 & 0.0461 & -0.0113 & 0.1492 & 0.0004 & 0.3010 \\
-0.0257 & -0.0036 & -0.0096 & 0.0094 & 0.0276 & 0.0068 \\
0.1970 & 0.0041 & 0.0073 & 0.3010 & 0.0068 & 0.3306
\end{pmatrix}, \quad \text{tr}(\Lambda_g) = 2.0370
\]

It is clear that there are important differences between these
three approaches. They are hard to interpret though on such a
simple example but their effect on the segmentation results will
be outlined in the Section IV.

In the next section, we set up a unified Bayesian formulation
of the segmentation problem that will be used throughout this
paper. It relies on the different possible estimates of the mean
\( \Sigma \) and covariance matrix \( \Lambda \) to evaluate the likelihood of a
diffusion tensor to belong to a given subset of the DTI data-set.
This will be used in (19). We recall that we will consider three
different cases associated to the Euclidean distance (3), the \( J \)-
divergence (8), and the geodesic distance (13). Within these three
different frameworks, we have shown how to approximate a
Gaussian distribution between diffusion tensors [see (5), (10),
and (15)] by using the information provided by the gradient of the
square of geodesic distance [see (4), (9), and (14)]. We will also exploit the information provided by the norm of the
tensor field spatial gradient [see (6), (11), and (16)] to localize the boundaries between structures of the brain white matter and
avoid mixing them through the boundary term (20) in our en-
ergy (21).

III. STATISTICAL SEGMENTATION BY SURFACE EVOLUTION

We recall that our goal is to compute the optimal 3-D surface
separating an anatomical structure of interest from the rest of a
DTI data-set. The statistical surface evolution, as developed in
[52], is a well-suited framework for our segmentation problem.
We hereafter summarize the important notions of this technique.

A. Bayesian Formulation for Image Partitioning

Following general works on image segmentation [4], [34],
[48], [70], we seek the optimal partition \( \mathcal{P} \) of the image domain
\( \Omega \) by maximizing the \emph{a posteriori} frame partition probability
\( p(\mathcal{P}(\Omega) | \Sigma) \) for the observed DTI \( \Sigma \). The Bayes rule allows to express this probability as

\[ p(\mathcal{P}(\Omega) | \Sigma) \propto \Psi (\Sigma | \mathcal{P}(\Omega)) p(\mathcal{P}(\Omega)). \]  

This formulation yields a separation of the image-based cues
from the geometric properties of the boundary given by \( \mathcal{P}(\Omega) \).
While being valid for any number of regions, we restrict this
formulation to binary partitions: the structure of interest and the
background. The image partition can be represented as the zero-crossing of a level-set function \( \phi \) [16], [20], [21], [47].
Noting \( B \) the interface between the two regions \( \Omega_{\text{in}} \) and \( \Omega_{\text{out}} \),
\( \phi \) is constructed as the signed distance function to \( B \)

\[
\begin{cases}
\phi(x) = 0, & \text{if } x \in \Omega_{\text{in}} \\
\phi(x) = \varphi (x, B), & \text{if } x \in \Omega_{\text{out}} \\
\phi(x) = -\varphi (x, B), & \text{if } x \in \Omega_{\text{out}}
\end{cases}
\]

where \( \varphi (x, B) \) stands for the Euclidean distance between \( x \)
and \( B \). Hence, the optimal partition is obtained by maximizing:
\( p(\phi | \Sigma) \propto \Psi (\Sigma | \phi) p(\phi) \). At this stage, these two terms still need to
be defined. For this purpose, several assumptions on the structure
of interest need to be introduced. In the following, a smoothness
constraint is imposed with the term \( p(\phi) \) while \( \Psi (\Sigma | \phi) \)
expresses the likelihood of the diffusion tensors to be inside, out-
side or on the boundary of the structure. This yields an optimization
criterion similar to the \emph{Geodesic Active Regions} presented in
[48].

B. Smoothness Constraint

The second term of (17) expresses the probability of the
interface to represent the structure of interest and can be used to
introduce prior shape knowledge. For the segmentation of DTI,
we have no high level prior information but we can use this
term to impose shape regularity. Such a constraint can be ob-
tained by favoring structures with a smaller surface \( |B| \) with
\( p(\phi) \propto \exp (-\nu |B|) \). This can be expressed with \( \phi \) by intro-
ducing the Dirac function \[68]

\[ p(\phi) \propto \exp \left( -\nu \int_{\Omega} \delta(\phi) \left| \nabla \phi(x) \right| dx \right) , \]  

(18)
C. Data Term

To further specify the image term \( p(\Sigma|\phi) \), we introduce some hypothesis. First, for a given level-set \( \phi \), we can classify the voxels into three classes: inside, outside or on the boundary. Then, we can define the probability density functions of a diffusion tensor for each class; \( p_{\text{in}}, p_{\text{out}} \) and \( p_{\phi} \). Assuming the diffusion tensors to be independent and identically distributed realizations of the corresponding random process, the data term is given by

\[
p(\Sigma|\phi) = \prod_{x \in \Omega_{\text{in}}} p_{\text{in}}(\Sigma(x)) \cdot \prod_{x \in \Omega_{\text{out}}} p_{\text{out}}(\Sigma(x)) \cdot \prod_{x \in \partial \Omega} p_{\phi}(\Sigma(x)).
\]  

(19)

This gives two different types of probability distributions: region-based with \( p_{\text{in/out}} \) and boundary-based with \( p_{\phi} \). \( p_{\text{in}} \) and \( p_{\text{out}} \) are given by the Gaussian distributions on tensors introduced in Section II-B \( p_{\phi} \) [see (5)], \( p_{\text{in}} \) [see (10)] and \( p_{\text{out}} \) [see (15)]. The parameters of these laws may be known a priori but in the absence of such information, they are introduced as unknown parameters.

Regarding \( p_{\phi} \), the probability should be close to one for high gradients of the diffusion tensors field and around zero for small variations. This leads to

\[
p_{\phi}(\Sigma(x)) \propto \exp(-g_{\alpha}(\nabla \Sigma(x)))
\]  

(20)

with \( g_{\alpha}(u) = 1/(1 + u^2) \). This type of boundary term is the basis of several works referred to as active contours [15] and, often, \( \alpha = 1 \) or 2 is chosen. For the sake of readability, we will use the short notation \( g_{\alpha}(\Sigma(x)) \cdot |\nabla \Sigma(x)| \) will be computed by using (6) for the Euclidean case, (11) for the \( J \)-divergence case, or (16) for the geodesic case.

D. Energy Formulation

Maximizing the \textit{a posteriori} segmentation probability is equivalent to minimizing its negative logarithm. Integrating the regularity constraint (18) and the image term (19), we end up with the following energy:

\[
E(\phi) \propto \int_{\Omega} \left[ \delta(\phi)|\nabla \phi|d\sigma + \int_{\Omega} \delta(\phi)|\nabla \phi|g_{\alpha}(\Sigma(x))d\sigma - \int_{\Omega_{\text{in}}} \log p(\Sigma(x)|\Omega_{\text{in}}, \Lambda_{\text{in}})d\sigma - \int_{\Omega_{\text{out}}} \log p(\Sigma(x)|\Omega_{\text{out}}, \Lambda_{\text{out}})d\sigma \right]
\]  

(21)

The boundary term of this energy corresponds to the Geodesic Active Contours [15] and naturally includes a regularization\(^1\) on the interface. Following [33] and [53], an alternate minimization is employed to perform the optimization for the two types of unknown parameters. For given statistical parameters, the Euler-Lagrange equations are computed to derive the implicit front evolution

\[
\frac{\partial \phi}{\partial t} = \delta(\phi) \left( (\nu + g_{\alpha}(\Sigma)) \nabla \phi + g_{\alpha}(\Sigma) \right)
\]  

(22)

while the statistics can be updated after each evolution of \( \phi \) from their empirical estimates, as described in Section II-B. More details on this level-set based optimization can be found in [16], [53], where different applications were considered.

The right-hand side of (22), between parenthesis, corresponds to the magnitude of the velocity used to deform each point of the evolving surface \( \partial \Omega \) along its normal at that point. The purpose of the next section will be to evaluate the influence of the choice of the density function \( p_{\phi} \) which can be taken in its Euclidean version \( p_{\phi} \) [see (5)], \( J \)-divergence version \( p_{\phi} \) [see (10)], or geodesic version \( p_{\phi} \) [see (15)]. We will describe several numerical experiments in order to evaluate the respective performances of each probability metrics for our DTI segmentation task. We demonstrate that the Riemannian statistical tools presented in Section II-B achieve the best results.

IV. RESULTS AND VALIDATION

We begin our numerical experiments with three different synthetic data-sets of increasing complexity in order to emphasize the respective virtue of the Euclidean, Kullback-Leibler and geodesic probability metrics. We then apply our algorithm to a biological rat spinal cord phantom. Finally, we consider real DTI data-sets on which we perform the segmentation of the corpus callosum.

In practice, there are a few important points that must be carefully addressed when implementing and running our segmentation algorithm: When dealing with real DTI data, we use a mask of the brain so that tensors statistics of \( \Omega_{\text{out}} \) are not corrupted by the signal from the outside of the brain. Regarding the initialization, we noticed and will demonstrate that our method is very robust. We will show that the geodesic distance is indeed the only metric capable of representing, through the associated Gaussian distribution, a smoothly varying tensor field with relatively high variability. Next, there are two parameters that have to be chosen: The first one is the value of \( \nu \) in (18). It constrains the smoothness of the surface and is usually set in the range 1 to 10. The second parameter arises from the very definition of the Gaussian distribution on \( S^{+}(3) \) presented in Section II-B3. The main hypothesis for this definition to be valid is that the trace of the covariance matrix \( \Lambda_{\phi} \) should be small and this means that we restrict ourselves to concentrated distributions. Hence, we set a threshold for the variance which, whenever reached, induces the end of the update for the statistical parameters. We then let the surface evolve while using a fixed mean and covariance matrix to model the distributions of the tensors in \( \Omega_{\text{in}}/\Omega_{\text{out}} \). The threshold is chosen in the range [0.01, 0.1] for tensors with components around 1.0. We noticed that the variance, after a few iterations of increase at the very beginning of the algorithm, keeps decreasing as the segmentation process converges. Consequently, a careful selection of this parameter is
not critical. Finally, we improved the computational efficiency of the method using the geodesic distance by noticing and experimentally verifying that, in (15), the term involving the $6 \times 6$ Ricci tensor $\mathcal{R}/3$ can be neglected since we have found, in our numerical experiments, a difference of at least 2 orders of magnitude between $\Lambda^{-1}$ and $\mathcal{R}/3$.

Regarding the computational cost of the method, we should point out that it is fairly efficient since the results presented in Figs. 20 and 21 were respectively obtained, on $128 \times 128 \times 64$ images, in 5 and 10 min on a 1.7-GHz Pentium M processor with 1 Gb of RAM.

### A. Synthetic Examples

Each of the three synthetic data-sets consists of a $40 \times 40 \times 40$ 3-D tensor field with a main pattern and a background. The tensors follow the shape of the pattern so that, as the shape becomes more twisted, the tensors variability increases and makes it more difficult for the algorithm to recover the entire shape. The regularity factor $\varphi$ is set to 1 for all the experiments. The initialization is done by the means of one or two spheres (see figures). Finally, the mean tensor and covariance matrix used for the generation of noise in all the experiments are

$$\mathbf{\Sigma} = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix},$$

$$\lambda = \begin{pmatrix}
0.0135 & -0.0412 & 0.005 \\
-0.0412 & 0.005 & 0.00013 \\
0.005 & 0.00013 & 0.0160
\end{pmatrix}.$$

1) **On the Generation of Gaussian Noise in $S^+(3)$**:

#### Algorithm 2 Generation of Gaussian noise in $S^+(3)$

**Require:** $\mathbf{\Sigma}$ and $\lambda$, mean tensor and covariance matrix

**Ensure:** $\mathbf{\Sigma}_1$, $N$ normally distributed elements of $S^+(3)$

1. for $i = 1$ to $N$ do
2. $\lambda = H H^T$ \{Cholesky decomposition of the covariance matrix\}
3. Create a random vector $Z \in \mathbb{S}^6$, with zero-mean and unit variance
4. Form $\beta_i = \varphi^{-1}(HZ) \in S^+(3)$
5. $\mathbf{\Sigma}_1 = (\exp(-\mathbf{\Sigma}^{-1}\beta_i \mathbf{\Sigma}^{-1}))^{-1}$
6. end for

The main drawback of this approach is that it leaves no grasp on the actual distribution of tensors. We proposed in [40], to use (15) to generate random tensors with a known mean $\mathbf{\Sigma}$ and covariance matrix $\Lambda$. The method, described in the algorithm 2, is fairly simple since all we need to do is to randomly choose the initial velocities $\{\beta_i\}$, $i = 1, \ldots, N$ of the geodesics in $S^+(3)$ joining the imposed mean tensor $\mathbf{\Sigma}$ to the random elements $\Sigma_1$.

In practice, this operation is performed in local coordinates so that we only need to draw random samples of the $\varphi(\beta_i) \in \mathbb{S}$ with zero-mean and covariance matrix $\Lambda$. The $\Sigma_1$ are easily obtained by using the expression $\beta_i = -\mathbf{\Sigma}_1 \log(\Sigma_1^2 \Sigma_1)$ [see (2)].

2) **The $Y$ Tensor Field**:

We start with a simple example composed by a diverging tensor field and a background of isotropic tensors (Fig. 4). Within the $Y$ shape, tensors fractional anisotropy decreases as we get away from the center-line. This example is relatively simple since the tensors variability stays low and the segmentation procedure succeeds with the three probability metrics. One important difference must be noted though: By comparison with the Euclidean distance, which requires 45 iterations to segment the $Y$ structure, the process converges faster when the $J$-divergence is used (30 iterations), and relatively faster with the geodesic distance (28 iterations).

This is easily explained by the fact that the velocity of the evolving surface, at location $\mathbf{x}$ of the image $\Sigma$, is directly related to the likelihood of tensor $\Sigma(x)$ to belong to $\Omega_{\text{in}}$ or $\Omega_{\text{out}}$. It is, hence, a first argument in favor of our claim that the geodesic probability metrics yields more adequate tensor statistics.

3) **The Torus Tensor Field**:

Next, we consider another example where the tensors follow the tangent of the center-line of a torus (Fig. 5) and share the same eigenvalues. This yields a higher orientational variability of the tensors. A direct consequence of this increased variability is the failure of the segmentation process when we use the Euclidean probability metric. The evolution is presented in Fig. 7. The initial sphere is setup so that it covers half of the torus and contains the part of the background situated “inside” the torus. The surface evolution falls into a local minimum and is unable to recover the desired shape. On the contrary, the $J$-divergence and geodesic distance behave consistently and succeed to segment the complete torus (Fig. 8). We notice, as in the previous example, that the segmentation using the geodesic distance converges faster (20 iterations) than the one relying on the $J$-divergence (27 iterations). The result presented for the Euclidean metric in Fig. 7 is the final state after 600 iterations.

4) **The Helix Tensor Field**:

The last synthetic data-set that we consider is the helix tensor field presented in Fig. 6. It is composed of a background with anisotropic tensors aligned on the

---

**Fig. 4.** Segmentation of the $Y$ tensor field. From left to right: axial slice of the original and noisy data-set, evolution of the segmentation.
$x_1$ axis of the 3-D field and an helix containing tensors oriented along the tangent of its center-line. The fractional anisotropy of the helix tensors varies around each spire. Moreover, the tensors orientation spans a broader range of possibilities than in the torus case since it changes along the $x_1$, $x_2$, and $x_3$ axes. This is certainly an example on which it is desirable for our segmentation algorithm to succeed since this tensors variation pattern is fairly realistic and may be found in real DTI. As a matter of fact, only the statistics computed with the geodesic distance enable our segmentation framework to achieve a correct extraction of the helix. The initialization consists of 2 small spheres overlapping the helix and the background. As we can see in Figs. 9 and 10, the surface evolution quickly stops when it uses the Euclidean distance or the $J$-divergence, even though the latter propagates further than the former as we could have expected. The local minima are respectively reached after 130 and 80 iterations. Using the geodesic distance, the complete helix is recovered (Fig. 11) after 300 iterations.

This last example undoubtedly demonstrates the superiority of the Riemannian framework over the statistics derived from the Euclidean or Kullback-Leibler dissimilarity measures.

B. Biological Phantom Data-Set

We tested our algorithm on a biological phantom produced by J. Campbell et al. at the McConnell Brain Imaging Center and Montreal Neurological Institute [13], [14] and was created from two excised Sprague-Dawley rat spinal cords embedded in 2% agar. Diffusion weighted images (DWI) were acquired at the Center for Magnetic Resonance Research, University of Minnesota, on a 3 Tesla Siemens Magnetom Trio whole-body clinical scanner. Measurements were made along 12 gradient directions. Acquisition parameters were: $b$ value $= 1000$ s$\cdot$mm$^{-2}$, TE $= 92$ ms, and TR $= 1.7$ s. The images were obtained on 64 evenly spaced axial planes with $128 \times 128$ pixels per slice. The voxel size is $2 \times 2 \times 2$ mm. As for the biological rat spinal cord phantom, diffusion tensors are computed by using the method proposed in [38]. An example of the resulting DTI is presented in Fig. 15. Following [59] and [30], we indicate the names of major fiber bundles.

C. Real DTI Data-Sets

1) Method and Tensors Estimation: Diffusion weighted images were acquired at the Center for Magnetic Resonance Research, University of Minnesota, on a 3 Tesla Siemens Magnetom Trio whole-body clinical scanner. Measurements were made along 12 gradient directions. Acquisition parameters were: $b$ value $= 1000$ s$\cdot$mm$^{-2}$, TE $= 92$ ms, and TR $= 1.7$ s. The images were obtained on 64 evenly spaced axial planes with $128 \times 128$ pixels per slice. The voxel size is $2 \times 2 \times 2$ mm. As for the biological rat spinal cord phantom, diffusion tensors are computed by using the method proposed in [38]. An example of the resulting DTI is presented in Fig. 15. Following [59] and [30], we indicate the names of major fiber bundles.

2) Performance of the Probability Metrics: In order to further compare the performance of the three probability metrics, within our segmentation framework, we have experimented with the extraction of the corpus callosum from a given DTI data-set. This important structure corresponds to the so-called callosal radiations which connect homologous areas of each hemisphere. It can be roughly divided into three main parts known as the genu (gcc), body (bcc), and splenium (scc). The genu radiates into the prefrontal, orbital and inferior premotor areas to form the forceps minor. The body of the corpus callosum radiates into the premotor, motor and supplementary motor cortical areas. Finally, the splenium radiates into the inferior/superior temporal, occipital and posterior parietal regions to form the forceps major.

It turns out that, near the midsagittal plane, all the fibers follow the same right-left orientation pattern making it quite easy to extract this structure from anatomical MRI [see

\[\text{LENGLET et al.: DTI SEGMENTATION BY STATISTICAL SURFACE EVOLUTION} 695\]

\[\text{Fig. 5. Axial slice of the original and noisy torus tensor field.}\]

\[\text{Fig. 6. Axial, coronal and sagittal slices of the original and noisy helix tensor field.}\]

\[\text{relies on the minimization of an energy functional derived from the linearized Stejskal-Tanner equation [57] while ensuring to remain in } S^+(3). \text{ An axial slice of the resulting DTI is presented in Fig. 12 together with a 3-D surface modeling the spinal cords. This data-set is well suited to evaluate the robustness to the initialization of our segmentation framework as well as to demonstrate the importance of the Riemannian framework to achieve good segmentation results.}\]

\[\text{Fig. 13 illustrates the evolution of the segmentation process, using the geodesic distance, for three very different initializations: One large sphere and one small sphere centered at the cord crossing, and one small sphere placed at one end of a cord. These three examples yield the same final result, thus experimentally showing the nondependence of our method on the initialization. Finally, Fig. 14 displays, on top of the Apparent Diffusion Coefficient image, the three final segmentation results obtained by using the Euclidean distance (black line), $J$-divergence (light gray line), and geodesic distance (gray line). We can see that the most accurate result is obtained with the latter. Especially, it is interesting to note that, in the upper right part of the image where the two cords are very close to each other, only the geodesic distance is able to distinguish between the two structures. This is another example of the better properties of the Riemannian statistics to model the distribution of the diffusion tensors.}\]

\[\text{In the next section, we will show that the Riemannian statistical approach also performs better on human brain DTI.}\]
Fig. 7. Failure of the torus segmentation with the Euclidean distance (right: final state after 600 iterations).

Fig. 8. Successful segmentations of the torus with the $J$-divergence and geodesic distances (right: final state after 27 iterations with the $J$-divergence or 20 iterations with the geodesic distance. The evolutions are similar.).

Fig. 9. Failure of the helix segmentation with the Euclidean distance.

Fig. 10. Failure of the helix segmentation with the $J$-divergence.

Fig. 16 (left). This has been used in group studies [45] to investigate architectural variability of the corpus callosum in relation with pathologies like schizophrenia. However, as we can see in Fig. 16 (right), once we get away from the midsagittal planes the callosal radiations quickly merge within the white matter and cannot be segmented anymore. We show that our Riemannian segmentation framework is able to provide more accurate segmentations of the corpus callosum.

The initialization is obtained either by a quick and approximate delineation of the genu and splenium on only two axial
Fig. 11. Successful segmentation of the helix with the geodesic distance.

Fig. 12. Segmentation of the rat spinal cords phantom—axial slice of the data-set (left) and final segmentation using the geodesic distance (right).

Fig. 13. Segmentation of the rat spinal cords phantom with the geodesic distance and a large sphere initialization (1st row), a small sphere initialization (2nd row) and initialization at one end of a cord (3rd row).

slices (Fig. 17) or by a simple sphere of radius eight voxels centered in the middle of the body of the corpus callosum. In both cases, results are identical and presented in Figs. 19 and 20. It is obvious that there is a clear improvement of the segmentation quality (especially in the region of the splenium, Fig. 19) when moving from the Euclidean distance to the $J$-divergence and it is much better when the statistics are computed with the geodesic distance.

The splenium of the corpus callosum is almost entirely recovered by the Riemannian approach while it is barely visible with the Euclidean method and only partially extracted when using the $J$-divergence. We noticed moreover than the Euclidean approach has a tendency to misclassify some tensors from the ventricles. This means that the statistics are not enough discriminant and even take over the boundary term at some locations. The geodesic distance definitely yields the best results.

3) Multiple Fiber Bundles Segmentation: We conclude our numerical experiments on human brain DTI by trying to also recover fibers from the corona radiata, which is known to merge with the corpus callosum. The initialization is presented in Fig. 18 and is meant to include some tensors from the superior part of the corona radiata (scr). To that end we simply added tensors of the scr on 2 coronal slices. It turns out that, with the Euclidean distance and $J$-divergence, these new tensors quickly disappear from the segmentation and the final results are the same as those presented in Fig. 20. This is not surprising and proves that the associated statistics do not constitute accurate descriptors of the tensors distribution. On the other side, the statistics computed with the geodesic distance make it possible to perform the desired segmentation, as presented in Fig. 21. This is a very interesting result since the superior part of the corona radiata is partially recovered. But more importantly, fiber tracts which are known to mingle with the callosal radiations are also segmented. It is indeed well-known that the corpus callosum merges with association and projection fibers as its gets toward the cortex. We can see in Fig. 21 that the tapetum, the posterior region of the corona radiata and a part of the superior longitudinal fasciculus are extracted since they fuse with the splenium of the corpus callosum. The posterior limb of the internal capsule (essentially the corticospinal tract) is equally segmented since it intersects with the corpus callosum.
Fig. 15. Axial (A,B) and coronal (C,D) DTI fractional anisotropy (FA) maps and major fiber bundles. acr = anterior region of the corona radiata, alic = anterior limb of the internal capsule, bcc = body of the corpus callosum, ec = external capsule, gec = genu of the corpus callosum, iff = inferior longitudinal fasciculus, ifo = inferior fronto-occipital fasciculus, pcr = posterior region of the corona radiata, plic = posterior limb of the internal capsule, scc = splenium of the corpus callosum, scr = superior region of the corona radiata, sfo = superior fronto-occipital fasciculus, slf = superior longitudinal fasciculus, and tpt = tapetum.

Fig. 16. Corpus callosum on a midsagittal (left) and sagittal (right) slice from a T1 MRI.

Fig. 17. Initialization of the corpus callosum segmentation by an approximate delineation of the genu and splenium.

and with the superior longitudinal fasciculus in the region of the centrum semiovale. All these results contribute to clearly validate our claim that the proposed Riemannian framework achieves the best segmentation results.

Fig. 18. Initialization of the corpus callosum segmentation with added tensors from the superior region of the corona radiata.

Fig. 19. Segmentation results in the region of the splenium (black line: Euclidean distance, light gray line: J-divergence, gray line: geodesic distance).

V. CONCLUSION

We have presented a unified statistical surface evolution framework for the segmentation of DTI. Since a diffusion tensor can be understood as the covariance matrix of a 3-D normal distribution with zero-mean, we have introduced various probability metrics (Euclidean distance, J-divergence and geodesic distance), i.e., dissimilarity measures between probability density functions, to derive statistics on DTI. These statistical parameters (mean and covariance matrix) allowed us to define a notion of Gaussian density for diffusion tensors, depending on the probability metric, which was used to model the distribution of a set of tensors. Finally, we have shown how to estimate the norm of the spatial gradient of a DTI in order to detect boundaries between structures of the white matter. By fusing these statistical and geometrical information within a variational framework, we derived a powerful level-set based DTI segmentation technique. At this point, our claim was that the special properties of the space of $3 \times 3$ diffusion tensors (symmetry and positivity) were naturally handled by working in the Riemannian framework. It must consequently yield more adequate tools to deal with tensors than the Euclidean or J-divergence approaches. The former, by seeing $S^+(3)$ as a linear space is completely blind to its curvature. The latter was shown to be equivalent to the geodesic distance only for infinitesimally close tensors. The Riemannian framework was proposed to derive the proper tools to work within the space of $3 \times 3$ diffusion tensors while taking into account its special properties. We proved that the choice of the probability metric, i.e., the dissimilarity measure, has a deep impact on the tensor statistics and, hence, on the segmentation results.
Through numerical experiments on synthetic data-sets, a biological rat spinal cord phantom, as well as on human brain DTI, we could experimentally demonstrate the superiority of the geodesic probability metric over the \( J \)-divergence which, in turn, performed better than the Euclidean distance. This order was found on synthetic data-sets with increasing complexity and for which, ground truth being known, it was very easy to undoubtedly evaluate the quality of the segmentations. The biological phantom data-set, because of its known and relatively simple geometry, allowed to test the robustness to the initialization of our algorithm and, again, to demonstrate on a single realistic data-set that the best results were obtained with the geodesic distance. Finally, on human brain DTI data-sets, the Riemannian approach was the only one capable of correctly segmenting highly variable tensor fields. It achieved better results than the other metrics (Euclidean distance and \( J \)-divergence), by comparison with neuroanatomical knowledge, for the segmentation of the corpus callosum or the corticospinal tract.

ACKNOWLEDGMENT

The authors would like to thank S. Lehéricy and K. Ugurbil (Center for Magnetic Resonance Research, University of Minnesota) for the human brain DTI data-sets used in this paper. They also gratefully acknowledge J. Campbell, K. Siddiqi, V.V. Rymar, and B. Pike (McConnell Brain Imaging Center, McGill University) as well as M. Descoteaux for the rat data-set.

REFERENCES

Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape

Daniel Cremers and Mikael Rousson

Siemens Corporate Research
755 College Road East, Princeton, NJ 08540
{daniel.cremers,mikael.rousson}@siemens.com

Rachid Deriche

INRIA, Sophia-Antipolis, France
{rachid.deriche}@sophia.inria.fr

Abstract. Since their introduction as a means of front propagation and their first application to edge-based segmentation in the early 90’s, level set methods have become increasingly popular as a general framework for image segmentation. In this paper, we present a survey of a specific class of region-based level set segmentation methods and clarify how they can all be derived from a common statistical framework.

Region-based segmentation schemes aim at partitioning the image domain by progressively fitting statistical models to the intensity, color, texture or motion in each of a set of regions. In contrast to edge-based schemes such as the classical Snakes, region-based methods tend to be less sensitive to noise. For typical images, the respective cost functionals tend to have less local minima which makes them particularly well-suited for local optimization methods such as the level set method.

We detail a general statistical formulation for level set segmentation. Subsequently, we clarify how the integration of various low level criteria leads to a set of cost functionals and point out relations between the different segmentation schemes. In experimental results, we demonstrate how the level set function is driven to partition the image plane into domains of coherent color, texture, dynamic texture or motion. Moreover, the Bayesian formulation allows to introduce prior shape knowledge into the level set method. We briefly review a number of advances in this domain.

Keywords: Image segmentation, level set methods, Bayesian inference, color, texture, motion

1. Introduction

The goal of image segmentation is to partition the image plane into meaningful areas, where meaningful typically refers to a separation of areas corresponding to different objects in the observed scene from the area corresponding to the background.

A large variety of segmentation algorithms have been proposed over the last few decades. While earlier approaches were often based on a set of rather heuristic processing steps (cf. [69]), optimization methods...
have become established as more principled and transparent methods: Segmentations of a given image are obtained by minimizing appropriate cost functionals. Among optimization methods, one can distinguish between spatially discrete and spatially continuous representations.

In spatially discrete approaches, the pixels of the image are usually considered as the nodes of a graph, and the aim of segmentation is to find cuts of this graph which have a minimal cost. Optimization algorithms for these problems include greedy approaches such as the Iterated Conditional Modes (ICM) [2] and continuation methods such as Simulated Annealing [35] or Graduated Non-convexity [5]. Specific classes of graph cut approaches gained in popularity with the re-discovery of efficient global optimization methods, which are based on concepts of dynamic programming [6], on spectral methods [82, 56] or on semidefinite programming techniques [45].

In spatially continuous approaches, the segmentation of the image plane \( \Omega \subset \mathbb{R}^2 \) is considered as a problem of infinite-dimensional optimization. Using variational methods, one computes segmentations of a given image \( I : \Omega \rightarrow \mathbb{R} \) by evolving contours in the direction of the negative energy gradient using appropriate partial differential equations (pdes). Such pde-based segmentation methods became popular with the seminal paper on Snakes by Kass et al. [44]. In this paper, the contour is implemented by an explicit (parametric) curve \( C : [0, 1] \rightarrow \Omega \) which is evolved by locally minimizing the functional

\[
E(C) = -\int |\nabla I(C)|^2 ds + \nu_1 \int |C_s|^2 ds + \nu_2 \int |C_{ss}|^2 ds, \quad (1)
\]

where \( C_s \) and \( C_{ss} \) denote the first and second derivative with respect to the curve parameter \( s \). The first term in (1) is the external energy which accounts for the image information, in the sense that the minimizing contour will favor locations of large image gradient. The last two terms – weighted by nonnegative parameters \( \nu_1 \) and \( \nu_2 \) – can be interpreted as an internal energy of the contour, measuring the length of the contour and its stiffness or rigidity.

The Snakes approach had an enormous impact in the segmentation community (with over 3000 citations to date). Yet, it suffers from several drawbacks:

1. The implementation of contour evolutions based on an explicit parameterization requires a delicate regridding (or reparameterization) process to avoid self-intersection and overlap of control or marker points.

\footnote{From a survey of a number of related publications and from our personal experience, it appears that the rigidity term is not particularly important, such that one commonly sets \( \nu_2 = 0 \).}
2. The explicit representation by default does not allow the evolving contour to undergo topological changes such that the segmentation of several objects or multiply-connected objects is not straight-forward.\(^2\)

3. The segmentations obtained by a local optimization method are bound to depend on the initialization. The Snake algorithm is known to be quite sensitive to the initialization. For many realistic images, the segmentation algorithm tends to get stuck in undesired local minima – in particular in the presence of noise.

4. The Snakes approach lacks a meaningful probabilistic interpretation. Extensions to other segmentation criteria – such as color, texture or motion – are not straight-forward.

In the present paper, we will review recent developments in the segmentation community which aim at resolving the above problems. We will review the level set method for front propagation as a means to handle topological changes of evolving interfaces and to remove the issues of contour parameterization and control point regridding. Among the level set methods, we will focus on statistical region-based methods, where the contour is not evolved by fitting to local edge information (as in the Snakes) but rather by fitting statistical models to intensity, color, texture or motion within each of the separated regions. The respective cost functionals tend to have less local minima for most realistic images. As a consequence, the segmentation schemes are far less sensitive to noise and to varying initialization.

The outline of the paper is as follows: In Section 2, we will review the general idea of level set based boundary propagation and its first applications to image segmentation. In Section 3, we will then review a probabilistic formulation of region-based segmentation. In particular, we will make very explicit what are the assumptions underlying the derivation of appropriate cost functionals. In the subsequent sections, we then detail how to adapt the probabilistic level set framework to different segmentation criteria: In Section 4, we present probabilistic models which drive the segmentation process to group regions of homogeneous intensity, color or texture. In Section 5, we briefly present extensions of this framework to Diffusion Tensor Images. In Section 6, we discuss a further extension which allows to exploit spatio-temporal dynamics to drive a segmentation process, given an entire sequence of images. In particular, this approach allows to separate textures which

\(^2\) It should be pointed out that based on various heuristics, one can successfully incorporate regridding mechanisms and topological changes into explicit representations – cf. [62, 48, 28, 25].
have identical spatial characteristics but differ in their temporal dynamics. In Section 7 we detail how to integrate motion information as a criterion for segmentation, leading to a partitioning of the image plane into areas of piecewise parametric motion. Finally, in Section 8, we briefly discuss numerous efforts to introduce statistical shape knowledge in level set based image segmentation in order to cope with missing or misleading low-level information.

2. Level Set Methods for Image Segmentation

In the variational framework, a segmentation of the image plane $\Omega$ is computed by locally minimizing an appropriate energy functional, such as the functional (1). The key idea is to evolve the boundary $C$ from some initialization in direction of the negative energy gradient, which is done by implementing the gradient descent equation:

$$\frac{\partial C}{\partial t} = -\frac{\partial E(C)}{\partial C} = F \cdot n,$$

modeling an evolution along the normal $n$ with a speed function $F$.

In general, one can distinguish between explicit (parametric) and implicit representations of contours. In explicit representations – such as splines or polygons – a contour is defined as a mapping from an interval to the image domain: $C : [0,1] \rightarrow \Omega$. The propagation of an explicit contour is typically implemented by a set of ordinary differential equations acting on the control or marker points. In order to guarantee stability of the contour evolution (i.e. preserve well-defined normal vectors), one needs to introduce certain regridding mechanisms to avoid overlap of control points, for example by numerically resampling the marker points every few iterations, by imposing in the variational formulation a rubber-band like attraction between neighboring points [25], or by introducing electrostatic repulsion [91]. Moreover, in order to segment several objects or multiply connected objects, one needs to introduce numerical tests to enable splitting and remerging of contours during the evolution. Successful advances in this direction were proposed among others by [50, 62, 48, 28].

In implicit contour representations, contours are represented as the (zero) level line of some embedding function $\phi : \Omega \rightarrow \mathbb{R}$:

$$C = \{ x \in \Omega \mid \phi(x) = 0 \}.$$  

$^3$ Most meaningful contour evolutions do not contain any tangential component as the latter does not affect the contour, but only the parameterization.
There are various methods to evolve implicitly represented contours. The most popular among these is the level set method [29, 30, 65], in which a contour is propagated by evolving a time-dependent embedding function \( \phi(x, t) \) according to an appropriate partial differential equation. In the following, we will briefly sketch two alternative methods to derive a level set evolution implementing the minimization of the energy \( E(C) \).

For a contour which evolves along the normal \( n \) with a speed \( F \) – see equation (2) – one can derive a corresponding partial differential equation for the embedding function \( \phi \) in the following way. Since \( \phi(C(t), t) = 0 \) at all times, the total time derivative of \( \phi \) at locations of the contour must vanish:

\[
\frac{d}{dt} \phi(C(t), t) = \nabla \phi \frac{\partial C}{\partial t} + \frac{\partial \phi}{\partial t} = \nabla \phi \cdot F \cdot n + \frac{\partial \phi}{\partial t} = 0. \tag{4}
\]

Inserting the definition of the normal \( n = \frac{\nabla \phi}{|\nabla \phi|} \), we get the evolution equation for \( \phi \):

\[
\frac{\partial \phi}{\partial t} = -|\nabla \phi| F. \tag{5}
\]

By derivation, this equation only specifies the evolution of \( \phi \) (and the values of the speed function \( F \)) at the location of the contour. For a numerical implementation one needs to extend the right-hand side of (5) to the image domain away from the contour.

Alternatively to the above derivation, one can obtain a level set equation from a variational formulation (cf. [98, 12]): Rather than deriving an appropriate partial differential equation for \( \phi \) which implements the contour evolution equation (2), one can embed a variational principle \( E(C) \) defined on the space of contours by a variational principle \( E(\phi) \) defined on the space of level set functions:

\[
E(C) \longrightarrow E(\phi)
\]

Subsequently, one can derive the Euler-Lagrange equation which minimizes \( E(\phi) \):

\[
\frac{\partial \phi}{\partial t} = -\frac{\partial E(\phi)}{\partial \phi}. \tag{6}
\]

In both cases, the embedding is not uniquely defined. Depending on the chosen embedding, one can obtain slightly different evolution equations for \( \phi(x, t) \).

The first applications of this level set formalism for the purpose of image segmentation were proposed in [10, 58, 57]. Independently, Caselles et al. [11] and Kichenassamy et al. [46] proposed a level set formulation
for the Snake energy (1) given by:

\[
\frac{\partial \phi}{\partial t} = |\nabla \phi| \text{div} \left( g(I) \frac{\nabla \phi}{|\nabla \phi|} \right) = g(I)|\nabla \phi| \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + \nabla g(I) \cdot \nabla \phi, \quad (7)
\]

where the gradient $|\nabla I|$ in functional (1) was replaced by a more general edge function $g(I)$. This approach is known as Geodesic Active Contours, because the underlying energy can be interpreted as the length of a contour in a Riemannian space with a metric induced by the image intensity. See [11, 46] for details.

Local optimization methods such as the Snakes have been heavily criticized because the computed segmentations depend on the initialization and because algorithms are easily trapped in undesired local minima for many realistic images. In particular in the presence of noise, numerous local minima of the cost functional (1) are created by local maxima of the image gradient. To overcome these local minima and to drive the contour toward the boundaries of objects of interest, researchers have introduced an additional balloon force [16] which leads to either a shrinking or an expansion of contours. Unfortunately this requires prior knowledge about whether the object of interest is inside or outside the initial contour. Moreover, the final segmentation will be biased toward smaller or larger segmentations.

In the following, we will review a probabilistic formulation of the segmentation problem which leads to region-based functionals rather than edge-based functionals such as the Snakes. Moreover, we will provide numerous experiments which demonstrate that such probabilistic region-based segmentation schemes do not suffer from the above drawbacks. While optimization is still done in a local manner, the respective functionals tend to have few local minima and segmentation results tend to be very robust to noise and varying initialization.

3. Statistical Formulation of Region-Based Segmentation

3.1. Image Segmentation as Bayesian Inference

Statistical approaches to image segmentation have a long tradition, they can be traced back to models of magnetism in physics, such as the Ising model [41], pioneering works in the field of image processing include spatially discrete formulations such as those of Geman and Geman [35] and Besag [2], and spatially continuous formulations such as the ones of Mumford and Shah [63] and Zhu and Yuille [100].

The probabilistic formulation of the segmentation problem presented in the following extends the statistical approaches pioneered in [49, 100,
In particular, this extension allows the probabilistic framework to be applied to segmentation criteria such as texture and motion, which will be detailed in subsequent sections. In [49], a segmentation functional is obtained from a Minimum Description Length (MDL) criterion. The link with the Mumford-Shah functional and the equivalence to Bayesian maximum a posteriori (MAP) estimation is provided in [100]. Following [66], an optimal partition $\mathcal{P}(\Omega)$ of the image plane $\Omega$ (i.e. a partition of the image plane into pairwise disjoint regions) can be computed by maximizing the a posteriori probability $p(\mathcal{P}(\Omega) \mid I)$ for a given image $I$.\footnote{In the following, $I$ can refer to a single image or to an entire image sequence.} The Bayes rule permits to express this conditional probability as

$$p(\mathcal{P}(\Omega) \mid I) \propto p(I \mid \mathcal{P}(\Omega)) p(\mathcal{P}(\Omega)), \quad (8)$$

thereby separating image-based cues (first term) from geometric properties of the partition (second term). The Bayesian framework has become increasingly popular to tackle many ill-posed problems in computer vision. Firstly the conditional probability $p(I \mid \mathcal{P}(\Omega))$ of an observation given a model state is often easier to model than the posterior distribution, it typically follows from a generative model of the image formation process. Secondly, the term $p(\mathcal{P}(\Omega))$ in (8) allows to introduce prior knowledge stating which interpretations of the data are a priori more or less likely. Wherever available, such a priori knowledge may help to cope with missing low-level information.

One can distinguish between generic priors and object specific priors. Object specific priors can be computed from a set of sample segmentations of an object of interest. In Section 8, we will briefly review a number of recent advances regarding the incorporation of statistically learnt priors into the level set framework.

In this section, we will focus on generic (often called geometric) priors. The most commonly used regularization constraint is a prior which favors a short length $C$ of the partition boundary:

$$p(\mathcal{P}(\Omega)) \propto e^{-\nu |C|}, \quad \nu > 0. \quad (9)$$

Higher-order constraints may be of interest for specific applications such as the segmentation of thin elongated structures [71, 64].

To further specify the image term $p(I \mid \mathcal{P}(\Omega))$ in (8), we make the following hypotheses. Following [66], we assume the image partition to be composed of $N$ regions without correlation between the labellings. This gives the simplified expression:

$$p(I \mid \mathcal{P}(\Omega)) = p(I \mid \{\Omega_1, \ldots, \Omega_N\}) = \prod_{i=1}^{N} p(I \mid \Omega_i), \quad (10)$$
where \( p(I | \Omega_i) \) denotes the probability of observing an image \( I \) when \( \Omega_i \) is a region of interest. Let us assume that regions of interest are characterized by a given feature \( f(x) \) associated with each image location. This feature may be a scalar quantity (such as the image intensity), a vector quantity (such as color or the spatio-temporal image gradient), or a tensor (such as a structure tensor or a diffusion tensor).

For the features presented in this paper, we make the assumption that the values of \( f \) at different locations of the same region can be modeled as independent and identically distributed realizations of the same random process.\(^5\) Let \( p_i \) be the probability density function (pdf) of this random process in \( \Omega_i \). Expression (10) then reads

\[
p(I | \mathcal{P}(|\Omega)) = \prod_{i=1}^{N} \prod_{x \in \Omega_i} \left( p_i(f(x)) \right)^{dx},
\]

where the bin volume \( dx \) is introduced to guarantee the correct continuum limit. Approximation (11) is not valid in general since image features (such as spatial gradients) are computed on a neighborhood structure and may therefore exhibit local spatial correlations. More importantly, one should expect to find spatial correlations of features when modeling textured regions. However, one can capture certain spatial correlations in the above model by computing appropriate features such as the structure tensor.

Maximization of the \textit{a posteriori} probability (8) is equivalent to minimizing its negative logarithm. Integrating the regularity constraint (9) and the region-based image term (11), we end up with following energy:

\[
E(\{\Omega_1, \ldots, \Omega_N\}) = -\sum_i \int_{\Omega_i} \log p_i(f(x)) \, dx + \nu |C|.
\]

In the context of intensity segmentation (i.e. \( f = I \)), this energy is the basis of several works [49, 100, 80, 66]. The region statistics are typically computed interlaced with the estimation of the boundary \( C \) [100], yet one can also compute appropriate intensity histograms beforehand [66]. In this paper, we will focus on the case that distributions and segmentation are computed jointly. Distributions can be either modeled as parametric or non-parametric ones. Upon insertion of parametric representations for \( p_i \) with parameters \( \theta_i \), the energy (12) takes on the

\(^5\) In Section 7, we will consider a generalization in which the underlying random processes are assumed to be space-varying. The distributions \( p_i \) in (11) then contain an explicit space dependency \( p_i(f(x), x) \) which allows to model spatially varying statistical distributions of features.
form
\[
E(\{\Omega_i, \theta_i\}_{i=1..N}) = - \sum_i \int_{\Omega_i} \log p(f(x) | \theta_i) \, dx + \nu |C|. \tag{13}
\]

For particular choices of parametric densities, the optimal parameters can be expressed as functions of the corresponding domains and only the regions remain as unknowns of the new energy
\[
\hat{E}(\{\Omega_i\}) \equiv \min_{\{\theta_i\}} E(\{\Omega_i, \theta_i\}) = - \sum_i \int_{\Omega_i} \log p(f(x) | \hat{\theta}_i) \, dx + \nu |C|, \tag{14}
\]
such that
\[
\hat{\theta}_i = \arg \min_{\theta} \left( - \int_{\Omega_i} \log p(f(x) | \theta) \, dx \right). \tag{15}
\]
In this case, the optimal model parameters \( \hat{\theta}_i \) typically depend on the regions \( \Omega_i \). As pointed out by several authors [85, 81, 1], this region-dependence can be taken into account in the computation of accurate shape gradients. Exact shape gradients can also be applied with non-parametric density estimation techniques like the Parzen window method [47, 73, 40]. In [75], it is shown that no additional terms arise in the shape gradient if the distributions \( p_i \) are assumed to be Gaussian. And in [39], the authors point out that the additional terms are negligible in the case of Laplacian distributions.\(^6\) We will therefore neglect higher-order terms in the computation of shape gradients and simply perform an alternating minimization of the energy (13) with respect to region boundaries and region models.

### 3.2. Two-Phase Level Set Formulation

Let us for the moment assume that the solution to (13) is in the class of binary (two-phase) segmentations, i.e. a partitioning of the domain \( \Omega \) such that each pixel is ascribed to one of two possible phases. Extending the approach of Chan and Vese [12], one can implement the functional (13) by:
\[
E(\phi, \{\theta_i\}) = - \int_{\Omega} H\phi \log p(f|\theta_1) - (1-H\phi) \log p(f|\theta_2) + \nu |\nabla H\phi| \, dx, \tag{16}
\]
where \( H\phi \) denotes the heaviside step function defined as:
\[
H\phi \equiv H(\phi) = \begin{cases} 
1 & \text{if } \phi \geq 0 \\
0 & \text{else}
\end{cases}. \tag{17}
\]
\(^6\) For a recent study of various noise models on level set segmentation see [61].
The first two terms in (16) model the areas inside and outside the contour while the last term represents the length of the separating interface.

Minimization is done by alternating a gradient descent for the embedding function $\phi$ (for fixed parameters $\theta_i$):

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + \log \frac{p(f(x)|\theta_2)}{p(f(x)|\theta_1)} \right).$$  \hspace{1cm} (18)

with an update of the parameters $\theta_i$ according to (15). In practice, the delta function $\delta$ is implemented by a smooth approximation – cf. [12].

3.3. Multiphase Level Set Formulation

Several authors have proposed level set formulations which can handle a larger number of phases [98, 96, 66, 8]. These methods use a separate level set function for each region. This clearly increases the computational complexity. Moreover, numerical implementations are somewhat involved since the formation of overlap and vacuum regions needs to be suppressed. By interpreting these overlap regions as separate regions, Chan and Vese derived an elegant formulation which only requires $\log_2(n)$ level set functions to model $n$ regions. Each of the $n$ regions is characterized by the various level set functions being either positive or negative. See [93] for details.

3.4. Scalar, Vector and Tensor-valued images

3.4.1. Scalar images

Let us consider a scalar image made up of two regions, the intensities of which are drawn from a Gaussian distribution:

$$p(I | \mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(I-\mu_i)^2}{2\sigma_i^2}}, \quad i = \{1, 2\}. \hspace{1cm} (19)$$

This distribution can be injected in the general bi-partitioning energy (16). Given a partition of the image plane according to a level set function $\phi$, optimal estimates for the mean $\mu_i$ and the variance $\sigma_i$ can be computed analytically:

$$\begin{cases} 
\mu_1 = \frac{1}{a_1} \int H(\phi)I(x)dx, \\
\sigma_1^2 = \frac{1}{a_1} \int H(\phi)(I(x)-\mu_1)^2dx,
\end{cases} \quad 
\begin{cases} 
\mu_2 = \frac{1}{a_2} \int (1-H(\phi))I(x)dx, \\
\sigma_2^2 = \frac{1}{a_2} \int (1-H(\phi))(I(x)-\mu_2)^2dx.
\end{cases}$$

where $a_1 = \int H(\phi)dx$ and $a_2 = \int (1-H(\phi))dx$ are the areas of the inside and outside region. For fixed model parameters, the gradient descent
equation for the level set function $\phi$ – see (18) – reads
\[
\frac{\partial \phi}{\partial t} = \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + \frac{(I-\mu_2)^2}{2\sigma_2^2} - \frac{(I-\mu_1)^2}{2\sigma_1^2} + \log \frac{\sigma_1}{\sigma_2} \right). \tag{20}
\]

More details on this derivation when the parameters $(\mu_i, \sigma_i)$ are taken as functions of $\Omega_i$ can be found in [76]. We end up with an algorithm that alternates the estimation of the empirical intensity means and variances inside each region and the level set evolution described by equation (20). Regarding the complexity, each iteration of the level set evolution is applied only inside a narrow band around the zero-crossing because the Dirac function is equal to zero at other locations. More interesting is that the statistical parameters can also be updated with a similar complexity: new updates are functions of their previous values and of the pixels where the sign of $\phi$ changes. Assuming the evolving interface to visit each pixel only once, the total complexity is thus linear in the size of the image.

3.4.2. Vector-valued images

A direct extension to vector-valued images is to use multivariate Gaussian densities as region models. Region pdfs are then parameterized by a vector mean and a covariance matrix. Similarly to the scalar case, the optimal statistical parameters are their empirical estimates in the corresponding region. The 2-phase segmentation of an image $I$ of any dimension can thus be obtained through the following level set evolution (cf. equation 18):
\[
\frac{\partial \phi}{\partial t} = \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + \log \frac{p(I(x)|\mu_2, \Sigma_2^2)}{p(I(x)|\mu_1, \Sigma_1^2)} \right), \tag{21}
\]
with:
\[
\begin{align*}
\mu_i &= \frac{1}{|\Omega_i|} \int_{\Omega_i} I(x) \, dx, \\
\Sigma_i &= \frac{1}{|\Omega_i|} \int_{\Omega_i} (I(x) - \mu_i)(I(x) - \mu_i)^\top \, dx \quad \text{for} \quad i = 1, 2. \tag{22}
\end{align*}
\]

Like in the scalar case, the estimation of the statistical parameters can be optimized to avoid a full computation over the whole image domain at each iteration. Here, it becomes a bit more technical since cross-components products appear in the covariance matrices but the final complexity is identical to the one obtained in the scalar case [72].
3.4.3. Tensor-valued images

In order to apply the above statistical level set framework to the segmentation of tensor images, one needs to define appropriate distances on the space of tensors. Several approaches have been proposed to define distances from an information theoretic point of view by interpreting the tensors as parameterizations of 0-mean multivariate normal laws. The definition of a distance between tensors is then translated to one of a dissimilarity measure between probability distributions.

The symmetric KL divergence

Wang and Vemuri [94] applied the symmetrized Kulback-Leibler (SKL) divergence – also called J-divergence – to define the region term of the front evolution. For multivariate 0-mean normal laws with covariance matrices $J_1$ and $J_2$, the SKL divergence is given by:

$$D_{SKL}(J_1, J_2) = \frac{1}{2} \sqrt{\text{trace} \left[ J_1^{-1} J_2 + J_2^{-1} J_1 \right] - 2n},$$  \hspace{1cm} (23)

where $n$ is the dimension of the tensors. This measure has the advantage of being affine invariant and closed form expressions are available for the mean tensors which is particularly interesting to estimate region statistics. Region confidence were also incorporated in [77]. These works present several promising segmentation experiments on 2D [94] and 3D [77] real diffusion tensor images.

The Rao Distance

Another distance has been proposed in [52, 51] with the same idea of considering tensors as covariance matrices of multivariate normal distributions. Following [84], a Riemannian metric is introduced and the geodesic distance between two members of this family is given by

$$D_G(J_1, J_2) = \sqrt{\frac{1}{2} \sum_{i=1}^{2} \log^2(\lambda_i)},$$  \hspace{1cm} (24)

where $\lambda_i$ denote the eigenvalues of the matrix $J_1^{-1/2} J_2 J_1^{-1/2}$. The same metric was proposed in [68] from a different viewpoint. It verifies the basic properties of a distance (positivity, symmetry and triangle inequality) and it is invariant to inversions: $D(J_1, J_2) = D(J_1^{-1}, J_2^{-1})$. The above metrics permit to define statistics on sets of SPD matrices which can be used to define the region term of the segmentation. It was also shown in [51] that the (asymmetric) Kullback Leibler divergence (23) is a Taylor approximation of the geodesic distance (24).
In the following sections, we will exploit the statistical level set framework introduced above to construct segmentation schemes for color, texture, dynamic texture and motion. To this end, we will consider different choices regarding the features $f$ – namely intensity values, color values, spatial structure tensors, spatio-temporal image gradients, or features modeling the local spatio-temporal dynamics – and respective sets of model parameters $\theta_i$, modeling color or texture distributions or parametric motion in the separated regions. Moreover, we will consider different choices for the distributions $p_i$ of these model parameters.

4. Intensity, Color and Texture

In the previous section, we considered Gaussian approximations for scalar and vector-values images. These models can be used to segment gray, color and texture images [12, 75, 73]. In the following, these models are applied to the segmentation of natural images. Curve evolutions are presented to illustrate the gradient descent driving to the segmentation.

4.1. Gray & color images

In Figure 1, we present the curve evolution obtained with the gray-value level set scheme of Section 3.4.1. The curve is initialized with a set of small circles and it successfully evolves toward the expected segmentation. Other initializations may be considered but using tiny circles provides a fast convergence speed and helps to detect small parts and holes. Note that changes of topology during the evolution are naturally handled by the implicit formulation.

We previously argued that the region-based formulation exhibits less local minima than approaches which solely rely on gradient information along the curve. To support this claim, we plotted in Figure 2 the

Figure 1. Curve evolution for the segmentation of a gray-level image using Gaussian intensity distributions to approximate region information.
Figure 2. Comparison of edge- and region-based segmentation methods in 1D. For a 1D intensity profile of the coin image (taken along the line indicated in white), we computed the energy associated with a split of the interval at different locations. While the region-based energy exhibits a broad basin of attraction around a single minimum located at the boundary of the coin (thick black line), the energy of the edge-based approach is characterized by numerous local minima (red line). A gradient descent on the latter energy would not lead to the desired segmentation.

Figure 3. Binary segmentation of a color image using multivariate Gaussian distributions as region descriptor (initialization and final segmentation) and multiphase color segmentation obtained with the algorithm developed in [8].

The region-based approach can directly be extended to color images by applying the vector-valued formulation of Section 3.4.2. The only point to be careful about is the choice of color space for the multivariate Gaussian model to make sense. The RGB space is definitely not the best one since, as can be seen from the MacAdam ellipse, the perception of color difference is nonlinear in this space. The CIE-lab space has been designed to approximate this nonlinearity by trying to mimic the logarithmic response of the eye. Figure 3 shows a two-phase and a multiphase example of vector-valued segmentation obtained on natural color images using this color space (the algorithm proposed in [8] was used for the multiphase implementation).
4.2. Texture

In gray and color image segmentation, pixel values are assumed to be spatially independent. This is not the case for textured images which are characterized by local correlations of pixel values. In the following, we will review a set of basic features which allow to capture these local correlations. More sophisticated features are conceivable as well [53].

4.2.1. The nonlinear structure tensor as texture feature

While texture analysis can rely on texture samples to learn accurate models [38, 26, 59, 83, 99], unsupervised image segmentation should learn these parameters on-line. Since high-order texture models introduce too many unknown parameters to be estimated in an unsupervised approach, more compact features are usually favored. Bigger et al. in [4] addressed this problem with the introduction of the structure tensor (also called second order moment matrix) which yields three different feature channels per scale. It has mainly been used to determine the intrinsic dimensionality of images in [3, 34] by providing a continuous measure to detect critical points like edges or corners. Yet, the structure tensor does not only give a scalar value reflecting the probability of an edge but it also includes the texture orientation. All these properties make this matrix a good descriptor for textures. The structure tensor [34, 4, 70, 55, 36] is given by the matrix of partial derivatives smoothed by a Gaussian kernel $K_\sigma$ with standard deviation $\sigma$:

$$J_\sigma = K_\sigma * (\nabla I \nabla I^T) = \begin{pmatrix} K_\sigma * I_{x1}^2 & K_\sigma * I_{x1} I_{x2} \\ K_\sigma * I_{x2} I_{x1} & K_\sigma * I_{x2}^2 \end{pmatrix}. \quad (25)$$

For color images, all channels can be taken into account by summing the tensors of the individual channels [97].

Despite its good properties for texture discrimination, the structure tensor is invariant to intensity changes. In order to segment images with and without texture, a feature vector including the square root of
Figure 5. Curve evolution for the segmentation of a zebra image using the nonlinear structure tensor and the smoothed intensity (here, a rectangle is used as initialization but small circles also lead to a similar result).

The structure tensor and the intensity was defined in [73]:

\[ f(x) = \left( I, \frac{I_x}{|\nabla I|}, \frac{I_y}{|\nabla I|}, 2I_x I_y \right)^T. \]  \quad (26)

The major problem of the classic structure tensor is the dislocation of edges due to the smoothing with Gaussian kernels as shown in figure 4. To address this problem, Weickert and Brox proposed in [95] to replace the Gaussian smoothing by nonlinear diffusion, applying nonlinear matrix-valued diffusion schemes introduced in [89, 90]. Applied on the feature vector \( f \), this nonlinear diffusion couples all channels by a joint diffusivity, the information of all channels is used to decide whether an edge is worth to be enhanced or not, leading to the simplification of the data, the removal of outliers, and the closing of structures. Figure 4 shows the features obtained on the zebra image.

The feature vectors resulting from the nonlinear diffusion form a vector-valued image which can be segmented using the vector-valued formulation presented in Section 3.4.2. Figure 5 shows a segmentation of the zebra image obtained with this method. For results on a wider range of texture images, we refer to [73].

The structure tensor is undoubtedly pertinent for texture discrimination but the approach developed so far still allows for further improvements. We shortly mention two recent extensions of this work in the following two paragraphs.

4.2.2. Scale introduction via TV flow

With the nonlinear structure tensor, one mainly considers a single scale for the whole image. Yet textures often differ from one another with respect to their intrinsic scale. In order to account for varying scale, a straightforward extension is to combine texture features at different scales. While this modification may integrate information at different scales, it also increases dramatically the number of channels and redundant information is introduced, making the second phase – the image
Figure 6. Segmentation results obtained by running a level set segmentation process on the 5-dimensional feature space given by the features of the structure tensor, the image intensity and a local scale measure computed from the speed of a total variation flow. Images are courtesy of Brox and Weickert [9].

Figure 7. Segmentation of a textured image with different dissimilarity measures between tensors. The left result was obtained using the Frobenius norm while the right segmentation is based on the Rao distance (see text for details). Images courtesy of de Luís García and Deriche [27].

partitioning – more difficult. In order to work with a reduced feature space, Brox and Weickert [9] proposed an elegant and efficient extension of the above framework, which combines similar texture features as above with a local scale measure. By exploiting the linear contrast reduction property of the TV (total variation) flow:

\[
\begin{align*}
\frac{\partial f}{\partial t} &= \text{div} \left( \frac{\nabla f}{\sqrt{|\nabla f|^2 + \epsilon^2}} \right) \\
 f(t = 0) &= I
\end{align*}
\]  

(27)

the authors are able to extract a local scale measure computed from the speed of the diffusion process. Upon combining this scale with the intensity and orientation features in (26), one can perform segmentation in a 5-dimensional feature space. Figure 6 shows three representative segmentation results.
4.2.3. **Metric between tensors**

While the previous approaches construct a feature vector from the components of the structure tensor and apply a vector-valued segmentation scheme, one can directly define metrics on the space of structure tensors [27], for example the metrics defined in Section 3.4.3. The comparison in Figure 7 shows that appropriate tensor distances lead to drastic improvements in the segmentation.

5. **Diffusion tensor images**

The problem of segmenting tensor-valued data also appears in medical imaging with the relatively new modality of diffusion tensor magnetic resonance images. In these images, a diffusion tensor is measured at each voxel. This tensor captures the local motion of water molecules as approximated by a Gaussian law. The metric between tensors described in Section 3.4.3 can be used to segment these images. Figure 8 shows a segmentation of the corpus callosum obtained with the geodesic distance.\footnote{This result was obtained in [51] with data provided by J. F. Mangin and J. B. Poline.}
6. Dynamic Texture

The texture segmentation framework detailed earlier is based on assigning local texture signatures to each image location. The subsequent integration into a level set framework aims at optimally grouping regions of similar signatures while imposing a length constraint on the separating boundary.

Given a video sequence of temporally varying textures – such as smoke on water – one can extend this concept to the space-time domain and group regions of similar spatio-temporal statistics. The first work addressing this problem was proposed in [32] where the authors made use of recent developments in the modeling of dynamic textures [31]. Due to the scope of this paper, we will merely review the key ideas.

Dynamic textures are models of temporally varying textures which assume the image sequence to be generated by a second-order stationary process. Experiments have demonstrated that numerous realistic image sequences, such as water waves, fluttering foliage, smoke and steam can be well synthesized by such Gauss-Markov processes [32].

More specifically, it is assumed that the temporally varying pixel intensities \( \{ I_i(t) \}_{i=1..m} \) can be approximated by a model \( \{ y_i(t) \}_{i=1..m} \) which is driven by a random process \( r(t) \in \mathbb{R}^n \) as follows [31]:

\[
\begin{cases}
    r(t + 1) = A r(t) + \sqrt{Q} v(t); & r(0) = r_0 \\
    y(t) = C r(t) + \sqrt{R} w(t)
\end{cases}
\]

(28)

Here \( y(t) \in \mathbb{R}^m \) represents the vector of intensities of all \( m \) pixels at time \( t \), \( v(t) \in \mathbb{R}^n \) and \( w(t) \in \mathbb{R}^m \) are white zero-mean Gaussian processes, \( A \in \mathbb{R}^{n \times n} \), \( C \in \mathbb{R}^{m \times n} \) are the model parameters, and \( Q \in \mathbb{R}^{n \times n} \), \( R \in \mathbb{R}^{m \times m} \) are the noise covariance matrices. The model parameters \( A \) and \( C \) in (28) can be estimated from an image sequence \( I(x, t) \) [31].

As suggested in [32], we can associate with each image location \( x \in \Omega \) a local signature \( \xi(x) \) characterizing the spatio-temporal dynamics at this location based on the model parameters \( A \) and \( C \) computed in a small spatial window. A meaningful signature cannot be directly defined on these model parameters because – as can be seen from the definition of (28) – there exists an entire equivalence class of model parameters which lead to the same dynamic texture\(^8\). Instead we define a local signature \( \xi(x) \) by:

\[
\xi(x) = (\cos \theta_1(x), \ldots, \cos \theta_n(x)),
\]

(29)

\(^8\) Substituting in (28) \( A \) with \( T A T^{-1} \), \( C \) with \( C T^{-1} \), \( Q \) with \( T Q T^{-1} \), and choosing the initial condition \( T r(0) \), where \( T \in GL(n) \) is any invertible \( n \times n \) matrix generates the same output covariance sequence.
where \( \{\theta_i\}_{i=1..n} \) are the subspace angles associated with the equivalence classes of the model at location \( x \) and some reference model. More precisely, if \( A \) and \( B \) are two measurement matrices, then \( \{\theta_i\}_{i=1..n} \) are given by the principal angles [15] between \( \text{range}(A) \) and \( \text{range}(B) \). For details on the computation of these angles, we refer to [32].

Assuming that the spatio-temporal signatures defined in (29) correspond to two Gaussian distributions, one can apply the vector-valued segmentation scheme introduced in Section 3.4.2 to group areas of similar spatio-temporal dynamics.

Due to the scope of this survey paper, we will merely show two complementary results obtained by the above segmentation scheme. Figure 6 shows the separation based on spatial orientation of a moving texture. The image data shows a water sequence for which we simply rotated two areas by 90 degrees. By construction, intensity characteristics and dynamics of the separated regions are identical, yet due to the different orientation they can be separated. Figure 6 shows a segmentation result which is complementary to the previous one. We generated a sequence containing regions which only differ with respect to their dynamics (but have identical spatial texture) by overlapping the ocean sequence in the regions corresponding to the disc and square over an ocean sequence slowed down by a factor of 2.
7. Motion

7.1. Motion as a criterion for segmentation

The central question underlying the construction of segmentation methods is to identify what properties characterize objects and distinguish them from other objects and from the background. In the previous sections, we reviewed level set methods which exploit low level properties such as color, texture or even dynamical texture. The respective image segmentation algorithms essentially group regions of similar low level properties.

Many objects in our environment are characterized by the fact that they move in a coherent manner. Figure 11, top row, shows the intensity-based segmentation of a single frame taken from an image sequence of two cars driving down the street.9 The two cars and the background are moving in different directions. Clearly the individual cars are not homogeneous regarding their intensity or texture. A purely intensity-based segmentation therefore fails to separate the objects from the background.

In the following, we will detail how the statistical segmentation scheme presented above can be adapted to incorporate motion information given two consecutive frames from an image sequence. Minimization of the resulting cost functional leads to a segmentation of the scene in terms of piecewise parametric motion.10 The present formulation was proposed in [17, 23] with an earlier (explicit contour) formulation in [21]. Related approaches were also proposed in [60, 67]. The central idea is that we do not precompute local motion vectors. Instead we jointly estimate the segmentation and the motion models for each of a set of regions by minimizing the proposed functional. In the notation introduced in Section 3, this means that – in contrast to the texture schemes of the previous Sections – the model parameters \( \theta_i \) (the motion models of the separate regions) will not correspond to simple aggregates of the local feature vectors \( f(x) \) (the space time gradients), but rather they will be derived quantities. Due to the scope of this article, we will constrain the presentation to the key ideas. For further details and a discussion of related approaches we refer the reader to [23]. For an extension of the proposed framework to the segmentation of space-time volumes given an entire video sequence, we refer to [22].

9 http://i21www.ira.uka.de/image_sequences/
10 In this paper, we are only concerned with 2D motion models. Such 2D motion models allow in particular to separate the different depth layers of a static scene filmed by a moving camera (cf. [22]). In terms of 3D motion, such a static scene instead corresponds to a single motion model.
Figure 11. Intensity versus motion segmentation. Since cars and background are not well-defined in terms of homogeneous intensity, color or texture, unsupervised low-level segmentation schemes based on a single frame are unable to separate objects and background (top row). By minimizing the motion competition functional (38) with $\nu = 1.5$, one obtains a fairly accurate segmentation of the two cars and an estimate of the motion of cars and background.

7.2. Motion Competition

Let $I : \Omega \times \mathbb{R} \rightarrow \mathbb{R}$ be a gray value image sequence. Denote the spatio-temporal image gradient of $I(x, t)$ by

$$\nabla_3 I = \left( \frac{\partial I}{\partial x_1}, \frac{\partial I}{\partial x_2}, \frac{\partial I}{\partial t} \right)^\top.$$  \hfill (30)

Let $v : \Omega \rightarrow \mathbb{R}^3$, $v(x) = (u(x), w(x), 1)^\top$ be the velocity vector at a point $x$ in homogeneous coordinates.\(^{11}\)

Let us assume that the intensity of a moving point remains constant throughout time.\(^{12}\) Expressed in differential form, this gives a relation between the spatio-temporal image gradient and the homogeneous velocity vector, known as the optic flow constraint:

$$\frac{dI}{dt} = \frac{\partial I}{\partial t} + \frac{\partial I}{\partial x_1} \frac{dx_1}{dt} + \frac{\partial I}{\partial x_2} \frac{dx_2}{dt} = v^\top \nabla_3 I = 0.$$  \hfill (31)

\(^{11}\) Since we are only concerned with two consecutive frames from a sequence, we will drop the time coordinate in the notation of the velocity field.

\(^{12}\) To allow for variation of the global illumination, one can alternatively assume constancy of higher-order derivatives (cf. [7]).
For the sake of segmentation, we will assume that the velocity in each of a set of regions can be modeled by a parametric motion of the form

\[ v(x) = S(x) \cdot q, \quad (32) \]

with a space dependent matrix \( S \) and a parameter vector \( q \). In particular, this includes the case of translational motion where \( S \) is the 3 \( \times \) 3 unit matrix and \( q = (u, w, 1) \) the vector of constant velocity in homogeneous coordinates. The parametric formulation (32) also includes the more general *affine motion model* with:

\[
S(x) = \begin{pmatrix} x_1 & x_2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_1 & x_2 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad q = (a, b, c, d, e, f, 1)^\top \quad (33)
\]

In the context of segmentation in space and time, this parametric formulation can be extended to incorporate temporally varying motion (such as acceleration and deceleration) – for details we refer to [22]. Inserting the parametric model (32) into the optic flow constraint (31) leads to a constraint on the relation between the parameter vector \( q \) and the space-time gradient \( \nabla_3 I \) at a specific location:

\[
\nabla_3 I^\top S(x) q = 0. \tag{34}
\]

Neglecting the case that the space-time gradient vanishes, this constraint states that the two vectors \( q \) and \( S(x)^\top \nabla_3 I(x) \) must be orthogonal. We therefore model the conditional probability to encounter a certain gradient measurement given a velocity model as a function of the angle \( \alpha \) between the two vectors:

\[
P(\nabla_3 I \mid q; x) \propto \exp\left(-\frac{(q^\top S^\top \nabla_3 I)^2}{|q|^2 |S^\top \nabla_3 I|^2}\right). \quad (35)
\]

This expression is maximal if the two vectors are indeed orthogonal, it is minimal if the two vectors are parallel. Yet it does not depend on the length of the two vectors. Note that due to the introduction of a spatially parametric model, this conditional probability becomes space-dependent – this is in contrast to the space-independent conditional probabilities considered in the Sections 4, 4.2 and 6 on color and texture segmentation. Analogous parametric generalizations of intensity-based level set segmentation approaches have been proposed by Vese [92]. And corresponding extensions of the above texture segmentation schemes are certainly conceivable.

Based on the Bayesian formulation introduced in Section 3, we can integrate the conditional probability for a measurement given certain model parameters into a variational framework for segmentation.
Inserting equation (35) into energy (13), we obtain the functional

\[ E(C, \{q_i\}) = \sum_{i=1}^{n} \int_{\Omega_i} \frac{q_i^\top T(x) q_i}{|q_i|^2} \, dx + \nu |C|, \tag{36} \]

where, for notational simplification, we have introduced the matrix

\[ T(x) = \frac{\nabla_3 I S^\top S \nabla_3 I^\top}{|S^\top \nabla_3 I|^2}, \tag{37} \]

The corresponding two-phase level set implementation – cf. equation (16) – is given by

\[ E(q_1, q_2, \phi) = \int_{\Omega} \frac{q_1^\top T q_1}{|q_1|^2} H\phi + \frac{q_2^\top T q_2}{|q_2|^2} (1 - H\phi) + \nu |\nabla H\phi| \, dx, \tag{38} \]

The first two terms in (38) enforce a homogeneity of the estimated motion in the two phases, while the last term enforces a minimal length of the region boundary given by the zero level set of \( \phi \). As discussed in 3.2, the functional (38) is optimized by alternating the estimate of the motion models \( q_1 \) and \( q_2 \) and an update of the level set function \( \phi \) defining the motion boundaries.

For fixed level set function \( \phi \), i.e. fixed regions \( \Omega_i \), minimizing this functional with respect to the motion parameters \( \{q_i\} \) results in a set of eigenvalue problems of the form:

\[ q_i = \arg \min_{q} \frac{q^\top T_i q}{q^\top q}, \quad \text{with} \quad T_i = \int_{\Omega_i} T(x) \, dx. \tag{39} \]

The parametric motion model \( q_i \) for each region \( \Omega_i \) is therefore given by the eigenvector corresponding to the smallest eigenvalue of \( T_i \). It is normalized, such that the third component is 1. Similar eigenvalue problems arise in motion estimation due to normalization with respect to the velocity magnitude (cf. [4, 43]).

Conversely, for fixed motion models \( q_i \), a gradient descent on the energy (38) for the boundary \( C \) results an evolution equation – cf. (18) – of the form:

\[ \frac{\partial \phi}{\partial t} = \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + e_2 - e_1 \right), \tag{40} \]

where

\[ e_i = -\log P(\nabla_3 I \mid q_i; x) = \frac{q_i^\top T q_i}{q_i^\top q_i} = \frac{q_i^\top \nabla_3 I S^\top S \nabla_3 I^\top q_i}{|q_i|^2 |S^\top \nabla_3 I|^2} \tag{41} \]
are the motion energy densities associated with the respective regions.

Note that – as in the previous sections – we have neglected in the evolution equation (40) higher-order terms which account for the dependence of the motion parameters \( q_i \) on the level set function \( \phi \). An Eulerian accurate shape optimization scheme as presented for example in [42] is the focus of ongoing research.

The two terms in the contour evolution (40) have the following intuitive interpretation: The first term aims at minimizing the length of the separating motion boundary. The second term is proportional to the difference of the energy densities \( e_i \) in the regions adjoining the boundary: The neighboring regions compete for the boundary in terms of their motion energy density, thereby maximizing the motion homogeneity. For this reason we refer to this process as Motion Competition.

7.3. EXPERIMENTAL RESULTS

All image segmentation models are based on a number of more or less explicitly stated assumptions about the properties which define the objects of interest. The motion competition model is based on the assumption that objects are defined in terms of homogeneously moving regions. It extends the Mumford-Shah functional of piecewise constant intensity to a model of piecewise parametric motion. Despite this formal similarity, the segmentations generated by the motion competition framework are very different from those of its gray value analogue. Figure 11, bottom row, shows the boundary evolution obtained by minimizing the motion segmentation functional (38) and the corresponding motion estimates superimposed on the first frame. In contrast to its gray value analogue, the energy minimization simultaneously generates a fairly accurate segmentation of the two cars and an estimate of the motion of cars and background.
Figure 13. Multiphase motion segmentation. Contour evolution for a multiphase implementation of motion competition on two consecutive frames from the flower garden sequence. A static scene filmed by a moving camera is partitioned into layers of different depth. See [23] for details.

Figure 14. Piecewise affine motion segmentation. Segmentations obtained by minimizing functional (38) with $\nu = 8 \cdot 10^{-5}$ for two image pairs showing a hand rotating (top) and moving toward the camera (bottom).

Figure 12 shows segmentation the contour evolution generated by minimizing functional (38) for two wall paper images with the text region (right image) moving to the right and the remainder of the image plane moving to the left. Even for human observers the differently moving regions are difficult to detect – similar to a camouflaged lizard moving on a similarly-textured ground. The gradient descent evolution superimposed on one of the two frames gradually separates the two motion regions without requiring salient features such as edges or Harris corner points. Figure 13 shows results obtained with a multiphase implementation of the motion competition functional. The static scene filmed by a moving camera is segmented into layers of different depth.

The functional (38) allows to segment piecewise affine motion fields. In particular, this class of motion models includes rotation and expansion/contraction. Figure 14 shows segmentations obtained for a hand in a cluttered background rotating (in the camera plane) and moving toward the camera. In this example the object of interest can be extracted from a fairly complex background based exclusively on their motion.
In the previous sections, we reviewed a number of approaches which allow to drive the level set segmentation based on various low-level assumptions regarding the intensity, color, texture or motion of objects and background. In numerous real-world applications, these approaches may fail to generate the desired segmentations, because the respective assumptions about the low-level properties are either insufficient or even violated. In certain medical images for example, object and background may exhibit very similar intensity characteristics. Moreover, the observed intensity or color of a 3D object may not be uniform due to directional lighting and cast shadows. And finally, misleading low-level information may arise due to noise or partial occlusion of the objects of interest. While the generic constraint (9) on the length of the segmenting boundary helps to cope with a certain amount of noise, it does introduce a bias toward contours of smaller length, thereby rounding corners or suppressing small scale details.

Beyond simple geometric regularity, the Bayesian formulation of the image segmentation problem allows to introduce higher-level prior knowledge about the shape of expected objects. This idea was pioneered by Grenander and coworkers [37]. In the following, we will briefly list some of the key contributions in the field of shape priors for level set segmentation.

The first application of shape priors for level set segmentation was developed by Leventon et al. [54] who propose to perform principal component analysis on a set of signed distance function embedding a set of sample shapes. The distance functions are sampled on a regular grid to obtain a vector representation. A term is added to the contour evolution equation to drive the embedding function to the most likely shape of the estimated distribution. Tsai et al. [86, 88] proposed a very efficient implementation of shape-driven level set segmentation by directly optimizing in the linear subspace spanned by the principal components. A detailed analysis of various shape distances and statistical shape analysis in the level set formulation can be found in [13]. Figure 15 shows the effect of variation along the first principal component on the embedding function and the implicitly represented contour.

The use of principal component analysis to model level set based shape distributions has two limitations: Firstly, the space of signed distance functions is not a linear space, i.e. arbitrary linear combinations of signed distance functions will in general not correspond to a signed distance function. Secondly, while the first few principal components capture (by definition) the most variation on the space of embedding functions, they will not necessarily capture the variation on
the space of the embedded contours. As a consequence, one may need to include a larger number of eigenmodes (compared to PCA on explicit contours) in order to capture certain details of the modeled shape. Nevertheless, we found the PCA representation to work fairly well in practical applications. An alternative linear shape representation on the basis of harmonic embedding has been studied in [33]. Chen et al. [14] proposed to impose shape information on the zero crossing (rather than on the level set function). Rousson et al. proposed variational integrations of the shape prior [78, 79] based on the assumption of a Gaussian distribution. The use of nonparametric density estimation to model larger classes of level set based shape distributions was developed in [20, 74]. This approach allows to model distributions of shape which are not Gaussian – such as the various views of a 3D object [19] or the silhouettes of a walking person [20]. Moreover, in the limit of large sample size, the nonparametric estimator constrains the distribution to the vicinity of the training shapes, such that the distribution favors shapes which are signed distance functions. A method to simultaneously impose shape information about several objects into level set based segmentation and to induce a recognition-driven segmentation through the competition of shape priors was developed in [24]. Dynamical statistical shape priors for implicit shape representations were proposed in [18]. The latter approach takes into account that in the context of image sequence segmentation, the probability of a contour
Figure 16. Sample segmentations using statistical shape priors. From left to right, the shape priors are static (a single shape), uniformly distributed in the PCA subspace, automatically selected from multiple shape instances [24] and dynamical [18].

will depend on which contours have been observed in previous frames. The respective shape models capture the temporal correlations among silhouettes which characterize many deforming shapes.

In Figure 16, we show a selection of segmentations obtained with some of the above methods. For further details we refer the reader to the respective publications.

9. Conclusion

We presented a survey of the class of region-based level set segmentation methods and detailed how they can be derived from a common statistical framework. The common goal of these approaches is to identify boundaries such that the color, texture, dynamic texture or motion in each of the separated regions is optimally approximated by simple statistical models.

Given a set of features or measurements $f(x)$ at each image location, minimization of the respective cost functionals leads to an estimation of a boundary $C$ and a set of parameter vectors $\{\theta_i\}$ associated with each of the separated regions. Depending on the chosen segmentation criterion, the features $f$ may be the pixel colors, the local structure tensors or the spatio-temporal intensity gradients, while the parameter vectors $\{\theta_i\}$ model distributions of intensity, color, texture or motion. The model parameters $\{\theta_i\}$ can be either simple aggregates of the features (as in the cases of color, texture or dynamic texture presented here) or derived quantities – as in the case of motion which is computed from the aggregated space-time gradients. The boundary $C \subset \Omega$ is implemented as the zero-crossing of an embedding function $\phi : \Omega \rightarrow \mathbb{R}$. Energy minimization leads to a gradient descent evolution of the embedding function interlaced with an update of the parameter vectors $\{\theta_i\}$ modeling the statistical distributions in the separated regions.
In numerous experimental results, we demonstrate that this class of level set methods allows to partition images into domains of coherent color, texture, dynamic texture or motion. In particular, we show that – in contrast to the traditional edge-based segmentation schemes, these region-based approaches are quite robust to noise and to varying initialization, making them well-suited for local optimization methods such as the level set method. We ended by reviewing some recent advances regarding the introduction of statistical shape knowledge into level set based segmentation schemes.

Acknowledgments

We thank Christoph Schnörr for helpful comments on the manuscript. We also thank Zhizhou Wang for fruitful discussions, and Thomas Brox, Rodrigo de Luis García, Christophe Lenglet, Gianfranco Doretto, Paolo Favaro, Stefano Soatto and Anthony Yezzi for providing image examples of their methods.

References


A General Framework for Image Segmentation Using Ordered Spatial Dependency

Mikaël Rousson and Chenyang Xu

Department of Imaging and Visualization
Siemens Corporate Research, Princeton, NJ

Abstract. The segmentation problem appears in most medical imaging applications. Many research groups are pushing toward a whole body segmentation based on atlases. With a similar objective, we propose a general framework to segment several structures. Rather than inventing yet another segmentation algorithm, we introduce inter-structure spatial dependencies to work with existing segmentation algorithms. Ranking the structures according to their dependencies, we end up with a hierarchical approach that improves each individual segmentation and provides automatic initializations. The best ordering of the structures can be learned off-line. We apply this framework to the segmentation of several structures in brain MR images.

1 Introduction

Different anatomical structures often have strong spatial dependency among each other. This spatial dependency is usually present in a hierarchical manner, i.e., the shape and pose variations of one structure is fully or partially bounded by those of other more stable structures. We refer to this type of spatial dependency as ordered spatial dependency due to its ordered nature. Radiologists routinely rely on ordered spatial dependency to help them locating and identifying structures that have large variations in shape, pose, and appearance by searching its presence relative to other structures that are much easier to identify. In this paper, we would like to take benefit from this inter-structure ordered spatial dependency in an explicit manner by proposing a novel general image segmentation framework. The proposed framework learns the ordered spatial dependency from pre-segmented training images and applies the learned model to improve both the performance and robustness of individual segmentation algorithms utilized.

A key benefit of the proposed framework is that it is not another new segmentation algorithm but a new general framework that could integrate any existing segmentation algorithms. The motivation of this work arises from the fact that many powerful and effective segmentation algorithms such as seeded region growing[1], watershed[16], active contours[9, 3], and graph cuts[2, 12] have been proposed and used widely in particular for medical image segmentation applications. The topic of devising a segmentation framework that combines existing segmentation algorithms to achieve better results has recently emerged as a new promising research direction [13]. The work [13] proposed a framework that computes an improved segmentation result based on optimizing parameter values of multiple segmentation algorithms. Similarly, our proposed framework also improves the segmentation results over the individual algorithms utilized,
but it achieves this in a different way by focusing on segmenting objects in a
hierarchical manner using the ordered spatial dependency.

Observing that structures often have a strong spatial dependency, we define a
new spatial prior based on neighboring structures. This dependency is introduced
by registering the structure of interest to a common reference coordinate system
based on neighboring structures. This can be achieved by computing the elastic
matching of the neighboring structures from one image to a reference one, and
then applying it to the structure of interest. This modeling can be implemented
for each structure, based on the ones already segmented. This leads us to the
definition of a hierarchical segmentation framework.

The proposed work has two important contributions: 1) the explicit modeling
and utilization of ordered spatial dependency for segmentation; 2) the estimation
of the optimal segmentation sequence for segmenting multiple structures. Our
work is closely related to atlas-based segmentation (cf. [5, 11, 8]), which treats
segmentation as a registration problem by elastically matching a pre-segmented
atlas to the target image. Atlas-based segmentation approaches are generally
better-suited for segmenting structures that are stable over the population of
study. Our proposed framework uses elastic matching to enforce the spatial de-
pendency and restricts the plausible segmentation space rather than using it to
obtain the final segmentation. The actual segmentation of each structure is still
performed using a pre-selected segmentation algorithm.

Other closely related work are the active shape and appearance models [6,
7, 10], which assume a statistical correlation between the shape or appearance
of the organs over population. Our proposed segmentation framework uses a
weaker assumption by modeling the relative locations of the structures between
one and another. It is also worth noting that recent works on joint segmentation
and registration [17, 15, 14], that also use segmentation and registration in an
iterative manner, are primarily used for segmenting two or more images simulta-
neously and do not use ordered dependency. In fact, one could incorporate
active shape and appearance models within our framework as its building blocks
like any other segmentation algorithms.

In the following sections, we detail the proposed segmentation framework and
demonstrate its utility in segmenting multiple structures from MR brain images.

2 Modeling inter-structure spatial dependency

Let \( \{S_1, \ldots, S_N\} \) be the set of structures of interest in an image. We assume a
dataset of \( M \) annotated images to be available. We note \( \{s_{ij}; i = 1, \ldots, N, j = 1, \ldots, M\} \) the complete set of structures. Given a manual segmentation \( s \) of a
structure \( S \in \{S_1, \ldots, S_N\} \), we propose a smooth approximation of the condi-
tional probability of an image location \( x \) to be inside the structure \( s \):

\[
p_S(x|s) \propto \exp(H_{\epsilon}(\phi(x)) - 1),
\]

where \( \phi \) is the distance transform of \( s \) and \( H_{\epsilon} \) is a regularized Heaviside func-
tion with \( \epsilon \) controlling the level of smoothness [3]. This distribution gives high
probability to voxels inside \( s \) and low probability to the ones outside, and the
smoothness of the transition is related to the distance to the interface. Also, we note the conditional probability of a voxel \( x \) to belong to the background of \( s \) as \( p_S(x|s) \):

\[
p_S(x|s) \propto \exp(-H_s(\phi(x))).
\]

Now, if we want to combine all the annotated instances of a structure \( S_i \) to define the spatial prior probability of the structure \( S_i \), we need to place the manual segmentations in a common reference\(^1\). It is at this point that we consider the ordered spatial dependency, i.e., \( S_i \)'s dependency on known neighboring structures. The principle is to align all the instances \( s_{ij} \) of \( S_i \) to a common coordinate system using the known structures as anchors. This is done by estimating a warping between each instance of the anchor structure(s) to a chosen reference. These warpings are obtained using an image based registration algorithm \([4]\) applied on level set representations of each structure instance. If several anchor structures are available, they are merged to form a single shape composed of several components. This allows us to constrain even more the deformation field between the structures.

Then, these warpings are applied to the corresponding structures \( s_{ij} \). Let \( \tilde{s}_{ij} \) be the segmentation transformed by the warping \( \psi_{ij} \) and \( \tilde{\phi}_{ij} \) be its level set representation, the spatial prior probability of \( S_i \) and its background \( \bar{S}_i \) are defined in the reference image as the geometric mean of each individual prior:

\[
\begin{align*}
p_{S_i}(x) &\propto \left( \prod_{j=1}^{M} \exp \left( H_s(\tilde{\phi}_{ij}(x)) - 1 \right) \right)^{\frac{1}{M}}, \\
p_{\bar{S}_i}(x) &\propto \left( \prod_{j=1}^{M} \exp \left( -H_s(\tilde{\phi}_{ij}(x)) \right) \right)^{\frac{1}{M}}.
\end{align*}
\]

Up to now, we did not give any detail about which anchor structures were considered to estimate the warping. A priori, we do not know which structure \( S_i \) is spatially dependent on. We propose to learn these dependencies by defining the spatial probability of \( S_i \) with respect to other structures \( \{S_k, k \neq i\} \). We denote \( \mathcal{V}_i \) all possible subsets of \( \{S_k, k \neq i\} \), \( v_i \in \mathcal{V}_i \) a subset of segmented structures, and \( v_{ij} \) the corresponding annotated structures in the training image \( j \). With these notations, all structure \( s_{ij} \) are registered to a reference image by estimating the warpings that align \( v_{ij} \) to a reference set \( v_{ir} \). Therefore, for each choice of subset \( v_i \), we end up with different registrations and hence, a different spatial prior probability for \( S_i \). Since this probability is subject to the selected subset \( v_i \), we will use the notations \( p_{S_i}(x|v_i) \) and \( p_{\bar{S}_i}(x|v_i) \) to denote respectively the prior probabilities of a voxel to be inside and outside the structure \( S_i \), given a set of known segmentations \( v_i \). The choice of the optimal subset of reference for each structure is studied in Section 3.2.

In the following section, we incorporate this spatial prior in a general segmentation framework and we detail the complete framework obtained when a level set based approach is considered.

---

1 The common reference is chosen arbitrary in this work. It would be interesting to study how important this choice is.
3 Integrating spatial prior in the segmentation process

Given an image $I$ and a set of segmented structures $v$, we want to extract another structure $s$ of the class $S$ (the class $S$ is one of the classes $S_i$ considered previously). We consider a statistical formulation of this segmentation problem using a maximum a posteriori estimation. This consists in maximizing the posterior conditional distribution $p(s|I, v)$. Making the assumption that $I$ and $v$ are not correlated, the optimal structure is the one maximizing

$$p(s|I, v) = p(s|I) p(s|v).$$

The first term can be expressed with any statistically defined segmentation algorithm whereas the other one can integrate the spatial prior learned in the previous section. To incorporate this prior knowledge, we make the assumption that the prior probabilities of the locations $x$ are independent and identically distributed. This allows us to incorporate the spatial prior probability term introduced in the previous section:

$$p(s|v) = \prod_{x \in s_{in}} p_S(x|v) \prod_{x \in s_{out}} p_{\bar{S}}(x|v),$$

where $s_{in}$ and $s_{out}$ are respectively the parts of the image inside and outside the structure $s$. This formulation is very general and any efficient segmentation approaches like graph-cuts [2, 12] and surface evolutions [9, 3] can be considered. In the following, we develop our system using level set based surface evolutions but this should not be seen as the only possibility.

3.1 Level set based segmentation

In the level set framework, the structure of interest is represented as the zero crossing of an embedding function $\phi : \Omega \rightarrow \mathbb{R}$:

$$s = \{ x \in \Omega | \phi(x) = 0 \}.$$ 

Hence, the problem of finding the surface $s$ becomes the one of finding a real function $\phi$ that maximizes: $p(s|I, v) \rightarrow p(\phi|I, v)$. Equivalently, the optimal solution can be obtained from the minimization of the energy:

$$E(\phi) = - \log p(\phi|I, v) = - \log p(\phi|I) - \log p(\phi|v)$$

We follow [3] to define the first term with a region-based criteria and a regularity constraint. To use the spatial prior, we first need to register the anchor structures from the current image to the reference ones used for modeling. Let $\psi$ be the obtained warping, the whole energy can be written as follows:

$$E(\phi) = - \int_{\Omega} (H_\phi \log p_{in}(I(x)) + (1 - H_\phi) \log p_{out}(I(x)) + \nu |\nabla H_\phi|) \, dx$$

$$- \lambda \int_{\Omega} (H_\phi \log p_S(\psi(x)|v) + (1 - H_\phi) \log p_{\bar{S}}(\psi(x)|v)) \, dx,$$

where $p_{in}$ and $p_{out}$ are the intensity distributions inside and outside the structures. They can be estimated on-line or a priori from the learning set. We minimize this energy using a gradient descent obtained from the Euler-Lagrange
equations. This gives the following curve evolution:
\[
\phi_t = \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + \log \frac{p_{\text{in}}(I)}{p_{\text{out}}(I)} + \lambda \log \frac{p_S(\psi(x)|v)}{p_S(\psi(x)|v)} \right) \\
= \delta(\phi) \left( \nu \text{div} \left( \frac{\nabla \phi}{|\nabla \phi|} \right) + \log \frac{p_{\text{in}}(I)}{p_{\text{out}}(I)} + \frac{\lambda}{M} \sum_{j=1}^{M} \left( 2H_\epsilon(\tilde{\phi}_{c_j}(\psi(x)) - 1) \right) \right)
\]

where \(\tilde{\phi}_{c_j}\) stand for the warpings estimated during the modeling phase for the current shape. The segmentation of \(s\) is obtained by evolving \(\phi\) according to this equation until convergence (the initialization is discussed in the next paragraph).

For an efficient implementation, \(\sum_{j=1}^{M} \left( 2H_\epsilon(\tilde{\phi}_{c_j}(\psi(x)) - 1) \right)\) can be estimated off-line, and then warped to the current image domain using \(\psi\).

### 3.2 Hierarchical segmentation

For a given ordering, we can run the segmentation algorithm on each structure successively. This process can be initialized automatically if we are able to segment the first structure without any spatial prior. In most medical images, this can be done easily by starting with the envelope of the body. Then, to segment each structure, we also need to initialize the associated level set. A straightforward solution is to place seeds inside each structure. Obviously, this would give a good initialization but it requires user interaction. The spatial prior can be used to make these initializations automatic by selecting the voxels with a prior probability greater than a threshold \(\tau\). More precisely, the initial level set \(\phi_0^i\) used to extract the structure \(S_i\) is set as follow:

\[
\begin{aligned}
\phi_0^i(x) &= +1, & \text{if } \log \frac{p_S(\psi(x)|v)}{p_{\text{out}}(I)} \geq \tau, \\
\phi_0^i(x) &= -1, & \text{otherwise}.
\end{aligned}
\]

With this technique, the segmentation of all \(N\) structures can be obtained automatically. Only the weights \(\epsilon, \nu, \lambda\) and \(\tau\) must be set before starting the process.

### 4 Estimation of the optimal segmentation sequence

To learn the subset \(v_i\) that helps the segmentation of \(S_i\) the best, we propose to apply the segmentation on a second set of annotated images. For each \(v_i\), we can measure the quality of the segmentation according to a chosen similarity measure \(\mathcal{M}\) between the automatic and “true” segmentation. Assuming that, if \(S_j\) depends on \(S_k\), \(S_k\) cannot depend on \(S_j\), the objective is to estimate the best ordering of the structures such that structures classified higher can be used to extract lower-classified ones. Once all the segmentations obtained for a given

\[\text{Once initialized with the spatial prior, the level set is projected to a signed distance functions. This is repeated after each iteration of the level set evolution.}\]
ordering, we measure the overall quality of the process by comparing the results with the manual segmentations according to a similarity measure $\mathcal{M}$. The optimal ordering is then given by:

$$\hat{O} = \arg \max_{O \in \mathcal{O}} \sum_{i=1}^{N} \sum_{j=1}^{M} \mathcal{M}(s_{ij}, \hat{s}_{ij}(O)),$$

where $\mathcal{O}$ is set of all permissible orderings, and $\hat{s}_{ij}(O)$ is the segmentation obtained automatically in the image $j$ for the structure $S_i$ using the ordering $O$. As for the similarity measure, we use the Dice coefficient. In general the number of structures to extract is relatively small ($<10$) and all combinations can be tested. If we fix the first structure, the number of combinations is equal $(N−1)!$. Even though, this number gets high for $N = 10$, this is an off-line process and the user can introduce heuristics to reduce the possible orderings. For example if choosing a given structure at a high level conducts to a bad segmentation of the next one, a whole set of possible orderings can be discarded. We are conducting further investigations to reduce efficiently the possible orderings in a more theoretical way.

5 **Hierarchical segmentation of brain structures in MR images**

We validate this segmentation framework on several structures of the brain in MR images: the lateral ventricles, the caudate nucleus, the thalamus and the skull. These structures were annotated manually in 13 different sagittal slices. The first step is the learning of the optimal ordering. To start this process, we must be able to segment automatically the first anchor structure. For the brain image shown in Figure 1, this is relatively simple if we consider the skull. Initializing the level set with a seed in the background is sufficient to get its segmentation. Then, each structure can be initialized automatically by following Section 3.2. We set the threshold $\tau$ to the maximum value of the location map $J$ minus 0.1. This guarantees to give a seed inside the structure. Having 4

---

**Fig. 1. Location priors corresponding to the optimal ordering** - From left to right are shown the reference image, and the spatial priors of (1) the lateral ventricle given the skull, (2) the thalamus given the skull and the lateral ventricle, and (3) the caudate nucleus given all other structures.
structures, the number of possible orderings is 6. We have tested our algorithm for each possible choice. Figure 1 shows the sequence that maximizes the overall Dice coefficient between the obtained segmentations and the manual ones for the whole training set. Figure 2 shows the location maps estimated for each ordering.

Once the optimal ordering known, we can validate the approach using a leave-one-out strategy on the 13 available images. A few results are presented in Figure 3. As quantitative validation, we computed the Dice coefficient for each of the 52 automatically computed segmentation, giving an average above 0.8.

6 Conclusion

We have presented a novel image segmentation framework that learns the ordered spatial dependency among structures to be segmented and applies it in a hierarchical manner to both provide automatic initializations and improve each individual segmentation algorithm’s performance. We demonstrated the efficacy of the proposed framework by applying it to the MR brain image segmentation with level set algorithm as its segmentation algorithm. Future work includes applying this framework to more applications with more types of segmentation algorithms. We believe that the paradigm of “boosting” segmentation performance by combining existing segmentation algorithms into a systematic framework is a promising research direction and the work presented in this paper is one step along this direction.
Acknowledgments

We thank Christophe Chefd’Hotel for helpful discussions and for making available his non-rigid registration algorithm, and support from Ali Khamene and Frank Sauer. We also thank Mariappan Nadar for providing brain MR images. This work is funded by Siemens Corporate Research.

References

Problem Statement

Goal
- Interactive system for deformable surface manipulation
- Level-sets

Challenges
- Deformation is slow
- Deformation is hard to control

Solution
- Accelerate level-set computation with GPU
- Visualize computation in real-time
Collaborators

University of Utah
Joe Kniss
Joshua Cates
Charles Hansen
Ross Whitaker

Deformable Surfaces

Applications of Level-Sets
- Fluid simulation
- Surface reconstruction for 3D scanning
- Surface processing
- Image / Volume segmentation
**Level-Set Method**

- Implicit surface
  \[ S_t = \{ x | \phi(x, t) = k \} \]

- Distance transform
  \( \phi \) denotes inside/outside

- Surface motion
  \[ \phi(x, t + \Delta t) = \phi(x, t) + \Delta t F | \nabla \phi | \]
  \( F \) = Signed speed in direction of normal

**CPU Level-Set Acceleration**

- Narrow-Band/Sparse-Grid
  - Compute PDE only near the surface
    - Adalsteinson et al. 1995
    - Whitaker et al. 1998
    - Peng et al. 1999
    - Houston et al., 2004
    - Museth et al., 2004 - 2006

- Time-dependent, sparse-grid solver

Diagram:
- Initialize Domain
- Compute
- Update Domain
GPU Level-Set History

- Strzodka et al. 2001
  - 2D level-set solver on NVIDIA GeForce 2
  - No narrow-band optimization

- Lefohn et al. 2002
  - Brute force 3D implementation on ATI Radeon 8500
  - No faster than CPU, but ~10x more computations
  - No narrow-band optimization

- Lefohn et al. 2003 / 2004
  - Narrow band GPU 3D level set solver

- Crane et al. 2007
  - 3D level set solver as part of fluid simulation in NVIDIA G80 launch demo
  - Mask unused grid cells

- Kolb et al. 2007
  - GPU particle level sets
**GPU Narrow-Band Solver**

- Sparse Volume Computation
  - CPU algorithm: Traverse list of active voxels
  - GPU algorithm: Compute all active voxels in parallel

- Data structures change after each PDE time step

**A Dynamic, Sparse GPU Solver**

- GPU: Computes PDE
- CPU: Manages GPU memory
**Level-Set Segmentation**

- Surface velocity attracts level set to desired feature

\[
\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[ \alpha D(\bar{x}) + (1 - \alpha) \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right]
\]

- Segmentation Parameters
  1) Intensity value of interest (center)
  2) Width of intensity interval (variance)
  3) Percentage of data vs. smoothing

**Data speed term**

- Attract level set to range of voxel intensities

\[
D(I) = \epsilon - |I - T|
\]
Curvature speed term

- Enforce surface smoothness
- Prevent segmentation “leaks”
- Smooth noisy solution

Movie
Interactive 3D Level Set Visualization

- Use GPU to perform interactive volume rendering of the level set solution while it evolves
  - Render with original data
  - Directly render level set data without reformatting data
  - 3D user interface to guide evolving level set surface

A Dynamic, Sparse GPU Data Structure

- Multi-Dimensional Virtual Memory
  - 3D virtual memory
  - 2D physical memory
  - 16 x 16 pixel pages
Direct Volume Rendering of Level Set

- Reconstruct 2D Slice of Virtual Memory Space
  - On-the-fly decompression on GPU
  - Use 2D geometry and texture coordinates

Deferred Filtering: Volume Rendering Compressed Data

- 2D slice-based rendering: No data duplication
- Tri-linear interpolation
- Full transfer function and lighting capabilities
Level-Set Segmentation Application

Idea: Segment Surface from 3D Image
- Begin with “seed” surface
- Deform surface into target segmentation

Demo
- Segmentation of MRI volumes
  - 128³ scalar volume

Hardware Details
- ATI Radeon 9800 Pro
- 1.7 GHz Intel Pentium 4
- 1 GB of RAM
Movie

Region-of-Interest Volume Rendering

- Limit extent of volume rendering
  - Use level-set segmentation to specify region
  - Add level-set value to transfer function
Evaluation User Study

Goal
- Can a user quickly find parameter settings to create an accurate, precise 3D segmentation?
- Relative to hand contouring

User Study Methodology

- Six users and nine data sets
  - Harvard Brigham and Women’s Hospital Brain Tumor Database
  - 256 x 256 x 124 MRI
- No pre-processing of data & no hidden parameters
- Ground truth
  - Expert hand contouring
  - STAPLE method (Warfield et al. MICCAI 2002)
User Study Results

- **Efficiency**
  - 6 ± 3 minutes per segmentation (vs multiple hours)
  - Solver idle 90% - 95% of time

- **Precision**
  - Intersubject similarity significantly better
  - 94.04% ± 0.04% vs. 82.65% ± 0.07%

- **Accuracy**
  - Within error bounds of expert hand segmentations
  - Compares well with other semi-automatic techniques
    - Kaus et al., Radiology, 2001

Summary

- **Interactive Level-Set System**
  - 10x – 15x speedup over optimized CPU implementation
  - Intuitive parameter tuning
  - User study evaluation

- But…
That was three+ years ago…

- GPUs are 6-7x faster!
- New GPU capabilities make building dynamic data structures easier and more efficient
- GPU data structures better understood (Glift, etc.)
- New, faster CPU level-set methods (RLE, etc.)

- Tremendous opportunity for new research

Future Directions

- Other Level-Set Applications
  - Surface processing, surface reconstruction, physical simulation

- Better User Interface for Level Sets
  - Add more user control of evolving level set solver
  - More powerful editing of level set solution

- “Interactive Visulation”
  - User-controllable PDE solvers
  - Combine automatic and by-hand methods
  - New visualization and computation challenges
Acknowledgements

- Joe Kniss – Volume rendering
- Josh Cates – Tumor user study
- Gordon Kindlmann – “Teem” raster-data toolkit
- Milan Ikits – “GLEW” OpenGL extension wrangler
- Ross Whitaker, Charles Hansen, Steven Parker and John Owens
- ATI: Evan Hart, Mark Segal, Jeff Royle, and Jason Mitchell
- Brigham and Women’s Hospital
- National Science Foundation Graduate Fellowship
- Office of Naval Research grant #N000140110033
- National Science Foundation grant #ACI008915 and #CCR0092065

Questions?

For More Information
Google “Lefohn level set”
http://graphics.cs.ucdavis.edu/~lefohn/

Journal Papers Based on this Work

GIST: an interactive, GPU-based level set segmentation tool for 3D medical images

Joshua E. Cates *, Aaron E. Lefohn, Ross T. Whitaker

Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112-9205, USA

Available online 5 August 2004

Abstract

While level sets have demonstrated a great potential for 3D medical image segmentation, their usefulness has been limited by two problems. First, 3D level sets are relatively slow to compute. Second, their formulation usually entails several free parameters which can be very difficult to correctly tune for specific applications. The second problem is compounded by the first. This paper describes a new tool for 3D segmentation that addresses these problems by computing level-set surface models at interactive rates. This tool employs two important, novel technologies. First is the mapping of a 3D level-set solver onto a commodity graphics card (GPU). This mapping relies on a novel mechanism for GPU memory management. The interactive rates level-set PDE solver give the user immediate feedback on the parameter settings, and thus users can tune free parameters and control the shape of the model in real time. The second technology is the use of intensity-based speed functions, which allow a user to quickly and intuitively specify the behavior of the deformable model. We have found that the combination of these interactive tools enables users to produce good, reliable segmentations. To support this observation, this paper presents qualitative results from several different datasets as well as a quantitative evaluation from a study of brain tumor segmentations.

© 2004 Elsevier B.V. All rights reserved.

1. Introduction

Image segmentation is arguably the most widely studied problem in image processing, and the literature shows a plethora of image segmentation algorithms that rely on a diverse range of strategies such as statistics, differential geometry, heuristics, graph theory, and algebra. No one segmentation technique has emerged as being superior to all others in all circumstances, and thus it seems that the field of medical image processing will evolve to a state where researchers and clinicians have access to a set of segmentation tools, i.e. a toolbox, from which they can choose the particular tool that is best suited for their particular application.

A complete segmentation toolbox will include a set of general purpose tools as well as various specialized segmentation tools. General purpose tools are those that can be quickly launched and used as the need arises in a wide range of applications. Specialized tools rely on stronger assumptions about a specific modality, anatomy, or application. When properly trained, tuned, and applied we would expect specialized tools to perform better than general purpose tools – when all other factors, such as operator time and compute time, are equal. Among general tools, the most popular example, and the goal standard for many applications, is hand contouring, which entails a knowledgeable user (e.g. a medical doctor) creating a 2D curve, drawn by manipulating a mouse, on a sequence of slices to delineate the object of interest.

This paper describes a new, general-purpose segmentation tool that relies on interactive deformable models implemented as level sets. While level sets have demonstrated a great potential for 3D medical image segmentation, their usefulness has been limited by two
problems. First, 3D level sets are relatively slow to compute. Second, their formulation usually entails several free parameters, which can be very difficult to correctly tune for specific applications. The second problem is compounded by the first. That is, users find it impractical to explore the space of possible parameter settings when an example result from a point in that space requires minutes or hours to generate. The tool presented in this paper alleviates these two problems by marrying a very fast solver with an intuitive speed function, a combination that allows a user to interactively guide a level-set based segmentation.

The software application described in this paper is called GPU-based interactive segmentation tool (GIST). GIST updates a level-set surface model at interactive rates on commodity graphics cards (GPUs), such as those that are commonly found on consumer-level personal computers. It can be applied to a general set of medical and biological applications by tuning several free parameters. Despite its general nature, we demonstrate the effectiveness of GIST by a quantitative comparison to a specialized tool and the associated gold standard for a specific problem: brain tumor segmentation (Kaus et al., 2001; Warfield et al., 2000). This paper makes the following contributions:

- A 3D segmentation tool that uses a new level-set deformation solver to achieve interactive rates (approximately 10–15x faster than previous solutions).
- An interactive mechanism for defining a level-set speed function that works on both scalar and multi-valued (i.e. spectral) data.
- Quantitative and qualitative evidence that this interactive level-set approach is effective for brain tumor segmentation.

The remainder of the paper, which is an extended version of (Lefohn et al., 2003), is organized as follows. The next section gives some technical background and related work on level sets, GPUs, and segmentation evaluation methods. Section 3 describes the formulation of the level-set equations and the solution on the GPU. Section 5.2 presents qualitative results on various datasets and a quantitative analysis of the performance of the method for brain tumor segmentation. Section 6 summarizes this work.

2. Background and related work

2.1. Level sets

This paper relies on an implicit representation of deformable surface models called the method of level sets, proposed by Osher and Sethian (1988). The level-set method (see also Section 3) computes the motion of a moving interface by solving a partial differential equation (PDE) on a volume. The use of level sets has been widely documented in the medical imaging literature, and several works give more comprehensive reviews of the method and the associated numerical techniques (Sethian, 1999; Fedkiw and Osher, 2002).

For certain classes of applications level sets have several advantages over parametric models. Because they are implicit, level sets can change topology and deform far from their initial conditions without reparameterization. This means that during a deformation a user need not worry about surfaces colliding or pinching off. Deformable and topologically adaptable meshes typically require the insertion or deletion of triangles under such circumstances (Miller et al., 1991; Lachaud and Montanvert, 1999; McInerney and Terzopoulos, 1995). Finally, level sets allow for geometric surface deformations, which means that the results of a deformation process depend on the shape of the surface and the input data and not on some underlying parameterization. The level-set method is a general framework that must be tuned to specific applications.

As with the original work on image segmentation by parametric deformable models (Kass et al., 1987), the level-set approach to segmentation typically combines a data-fitting term with a smoothing term. However, there are alternatives. For instance, Whitaker (1994) proposes a formulation that mimics parametric deformable models, in which level surfaces move toward edges (high gradient magnitude) in volumes. In that formulation the model must be within a somewhat narrow band of attraction (defined by the second derivative) in order to lock onto such edges, and therefore the author proposes a multiscale computational method to improve convergence. Malladi et al. (1995) describe a formulation in which level curves/surfaces expand (or contract) with a motion that slows at image edges. Because of the monotonic expansion/contraction, convergence to local minima is less of a problem, but the results tend to be biased either inward or outward. Caselles et al. (1995) propose an alternative that minimizes an edge-weighted area metric. In that case the data term is weighted more heavily as the model approaches its target. These methods (and many others) focus on image edges, but the literature documents several other strategies for fitting level sets to image data. For instance, several authors propose using the statistics of the greyscale interior of the model to control the motion (Chesnaud et al., 1999; Radeva and Vitria, 2001). Alternatively, the motion of the level set can depend on a variational formulation that positions the interface to create discontinuities that best model the discontinuities in the input data (Tsai et al., 2000; Chan and Vese, 2001; Whitaker and Elangovan, 2002). In this paper we use a supervised, statistical classifier to drive the motion of the level-set model.
Virtually all of these methods include a form of mean curvature to keep the level-set smooth as it converges on a solution. Whitaker (1994) proposes a weighted sum of principle curvatures to preserve cylindrical structures. Lorigo et al. (2000) propose the minimum curvature in the context of segmenting blood vessels, which is equivalent to a space-curvature shortening for very thin objects. Siddiqi et al. (1998) propose a linear combination of standard curvature flow with a weighted area minimizing flow for general shape modeling. Recently, Tasdizen et al. (2003, 2004) propose the diffusion of normals in order to approximate higher-order geometric flows.

Solving level-set PDEs on a volume requires proper numerical schemes (Osher and Sethian, 1988) and entails a significant computational burden. Stability requires that the surface can progress at most a distance of one voxel at each iteration, and thus a large number of iterations are required to compute significant deformations. There is a special case of the level-set PDEs in which the surface motion is strictly inward or outward. Such equations can be solved somewhat efficiently using the fast marching method (Sethian, 1999) and variations thereof (Droske et al., 2001). However, this case covers only a very small subset of interesting speed functions, and such speed functions are inconsistent with interactive parameter tuning. In general we are concerned with problems that include a surface curvature term and simultaneously require the model to expand and contract to match the data.

Efficient algorithms for solving the more general level-set problem rely on the observation that at any one time step the only parts of the solution that are important are those adjacent to the moving surface. In light of this observation, several authors propose numerical schemes that compute solutions for only those voxels that lie in a small number of layers adjacent to the surface as shown in Fig. 1(b). Adalsteinson and Sethian (1995) propose the narrow band method, which updates the embedding on a band of 10–20 pixels around the model, and reinitializes that band whenever the model approaches the edge. Whitaker (1998) proposes the sparse-field method, which introduces a scheme in which updates are calculated only on the wavefront, and several layers around that wavefront are updated via a distance transform at each iteration. Peng et al. (1999) present a similar local method. Even with this very narrow band of computation, update rates using conventional processors on typical medical data sets (e.g. 256³ voxels) are not interactive. This is the motivation behind the GPU-based solver in GIST.

2.2. Graphics processing units for scientific computation

Graphics processing units have been developed primarily for the computer gaming industry, but over the last several years researchers have come to recognize them as low cost, high performance computing platforms. Two important trends in GPU development, increased programmability and higher precision arithmetic processing, have helped to foster new non-gaming applications.

Graphics processors outperform central processing units (CPUs) – often by more than an order of magnitude – because of their streaming architecture (Buck et al., 2004; Owens, 2002) and dedicated high-speed memory. In the streaming model of computation, arrays of input data are processed identically by the same computation kernel to produce output data streams. The GPU takes advantage of the data-level parallelism inherent in this model by having many identical processors execute the computation in parallel. This computation model has been used by a number of researchers to map a wide variety of computationally demanding problems to GPUs. Examples include matrix multiplication, finite element methods, and multi-grid solvers.
(Goodnight et al., 2003; Larsen and McAllister, 2001; Strzodka and Rumpf, 2001). All of these examples demonstrate a homogeneous sequence of operations over a densely populated grid structure.

Rumpf and Strzodka (2001) were the first to show that the level-set equations could be solved using a graphics processor. Their solver implements the 2D level-set method using a time-invariant speed function for flood-fill-like image segmentation, without the associated curvature. Their solver does not take advantage of the sparse nature of the level-set PDEs and therefore performs only marginally better than a highly optimized sparse-field CPU implementation. The work in this paper relies on a 3D generalization of Rumpf and Strzodka (2001), which includes a second-order curvature computation, and a significantly improved GPU solver that implements a narrow-band strategy. Also related is the work of Sherbondy et al. (2003), in which the authors identify regions of interest to solve a diffusion equation for volume segmentation.

This paper describes a GPU computational model that supports time-dependent, sparse grid problems. Such problems are difficult to solve efficiently with GPUs for two reasons. The first is that in order to take advantage of the GPU’s parallelism, the streams being processed must be large, contiguous blocks of data, and thus grid points near the level-set surface model must be packed into blocks of GPU texture memory, often referred to as textures. The second difficulty is that the level set moves with each time step, and thus the packed representation must readily adapt to the changing position of the model. This requirement is in contrast to the recent sparse matrix solvers (Bolz et al., 2003; Krüger and Westermann, 2003) and previous work on rendering with compressed data (Beers et al., 1996; Kraus and Ertl, 2002). In the two sparse-matrix solvers (Bolz et al., 2003; Krüger and Westermann, 2003), a packed texture memory scheme is used to efficiently compute sparse matrix-vector multiplications as well as compute values of the sparse matrix elements on the GPU. The scheme is static, however, in the sense that the nonzero matrix elements must be identified before the computation begins.

2.3. Segmentation evaluation

This paper includes a systematic evaluation of the performance of GIST. The role of segmentation evaluation is to understand the strengths, limitations, and potential applications of a particular segmentation algorithm. There are two strategies for evaluating segmentation algorithms. One strategy is to study segmentation performance in the context of a particular clinical or scientific question (Malpica et al., 1997; Sijbers et al., 1997). For instance, one might examine the effectiveness of the algorithm within a study that monitors the volumes or sizes of tumors. The second approach is to study to evaluate segmentation in the absence of a specific clinical application by quantifying the general behavior of the algorithm relative to an ideal solution. This paper takes the second approach, and uses general shape metrics to compare watershed segmentation results with the defacto gold standard for clinical applications, which is hand contouring one slice at a time (which we will also call manual segmentation) by expert observers.

Segmentation evaluation is difficult because of the lack of standard metrics and the difficulty of establishing ground truth in clinical data. Our evaluation methodology is derived from ideas developed by Yoo et al. (2000) and others (Udupa et al., 2002; Chalana and Yongmin, 1997; Gerig et al., 2001), who emphasize the importance of quantitative evaluation and statistical metrics. The study in this paper concerns a user-assisted segmentation technique, which requires a user-based evaluation to capture variations in the individual decision-making process. Experimental trials across a number of users and images (Zijdenbos et al., 1994; 1997) can generate data appropriate for statistical analysis that account for user variability.

A combination of different factors determines the effectiveness of a segmentation. For instance Udupa et al. (2002) propose a quantification of performance based on validity of the results (accuracy), reproducibility of the results (precision) and efficiency of the segmentation method (time). Other researchers have studied the sensitivity of the technique to various disruptive factors such as data artifacts, pathology, or individual anatomical variation (robustness) (Jannin et al., 2002). Accuracy metrics typically rely on a ground truth segmentation – segmentations that are somehow close to this ground truth are considered better than those that are not. Studies with digital or physical phantoms provide a ready definition of ground truth. For biological or clinical data sets, however, ground truth is usually unknown. In such a case, researchers typically rely on experts to delineate the ground truth by hand (Jannin et al., 2002; Prastawa et al., 2003). Experts seldom all agree, but a statistical combination (averaging) of several expert segmentations can account for expert variability. Averaging of multiple nonparametric shapes, however, is itself a difficult problem. One technique for combining multiple segmentations is Simultaneous Truth and Performance Level Estimation (STAPLE) (Warfield et al., 2002). This technique treats segmentation as a pixelwise classification, which leads to an averaging scheme that accounts for systematic biases in the behavior of experts.

The accuracy of an individual experimental segmentation is usually given through some measure of a region’s overlap and its distance from the ground truth. Common distance metrics include the Hausdorff distance (Huttenlocher et al., 1993) and the root mean
squared distance between selected boundary points (Chalana and Yongmin, 1997; Gerig et al., 2001). Often overlap is characterized by a similarity measure between experimental and ground truth volumes. One common similarity measure is the cardinality of the intersection (in pixels or voxels) of positive classifications in two volumes over the union of the positive classifications (Zijdenbos et al., 1994; Leemput et al., 1999). Another overlap metric is the total correct fraction, which is simply the percentage of correctly classified pixels in the image volume (negative and positive) (Kaus et al., 2001).

Another strategy for evaluating a single-object segmentation is to view each pixel as an instance of a detection task, which gives rise to metrics for sensitivity and specificity. Sensitivity, \( p \), is the true positive fraction of the segmentation, the percentage of pixels in an image correctly classified as lying inside the object boundary. Specificity, \( q \), is the true negative fraction, the percentage of pixels in a segmentation correctly classified as lying outside the object boundary. Because there is an explicit trade-off between sensitivity and specificity, researchers have proposed using receiver operator characterizations (ROC), which monitor the behavior of this trade-off for different segmentation algorithms or parameter settings (Whitaker, 1993; Udupa et al., 2002).

The precision of a segmentation method is an indicator of how repeatable the results are using that technique and, simultaneously, an indicator of the degree of randomness inherent to the method. Precision does not rely on a knowledge of ground truth and can be estimated by applying a similarity measure within a set of experimental segmentations (Udupa et al., 2002). The mean similarity value from these comparison is a characterization of the precision of the method.

The efficiency of a segmentation technique is a measure of the time involved in producing a segmentation. Efficiency measurements can include user interaction time and compute times. These two factors are usually considered individually because each has a separate cost and, depending on the specific application, may affect the practicability of the method in different ways.

3. Level-set formulation and algorithms

We begin this section with a brief review of the notation and mathematics of level-set methods and describe the particular formulation that is relevant to this paper. Comprehensive reviews of level-set methods are given in the literature (Sethian, 1999; Fedkiw and Osher, 2002).

An implicit model is a surface representation in which the surface consists of all points \( \mathcal{S} = \{ \mathbf{x} | \phi(\mathbf{x}) = 0 \} \), where \( \phi : \mathbb{R}^n \rightarrow \mathbb{R} \). Level-set methods relate the motion of that surface to a PDE on the volume, i.e. \( \frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \mathbf{v}(t) \), where \( \mathbf{v}(t) \) describes the motion of the surface. Within this framework one can implement a wide range of deformations by defining an appropriate \( \mathbf{v} \). For segmentation, the velocity often consists of a combination of two terms (Whitaker, 1994; Malladi et al., 1995)

\[
\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[ z D(\mathbf{x}) + (1 - z) \nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right) \right],
\]

where \( D \) is a data term that forces the model toward desirable features in the input data, the term \( \nabla \cdot (\nabla \phi / |\nabla \phi|) \) is the mean curvature of the surface, which forces the surface to have less area (and remain smooth) and \( z \in [0,1] \) is a free parameter that controls the degree of smoothness in the solution. There are several variations on this framework in the literature (e.g. Caselles et al., 1995).

The behavior of the model is mostly characterized by the data term and how it relates to the image. Invariably, the data term introduces free parameters, and the proper tuning of those parameters, along with \( z \), is critical to making the model behave in a desirable manner.

3.1. An intensity-based speed function

Our strategy is to construct a speed function \( D \) that causes the model to grow in regions where the data is consistent with the desired segmentation and to contract in regions where it is not. We can achieve this by letting \( D \) have positive or negative values depending on whether or not the model is within a specified range of data values. In this case the speed function at any one point is based solely on input value \( I \) at the point \( x \) in the image, i.e. \( D(x) = D(I(x)) \).

Such a simple scalar speed function is given by

\[
D(I) = \epsilon - |I - T|,
\]

where \( T \) controls the dominant intensity of the region to be segmented and \( \epsilon \) controls the range of greyscale values around \( T \) that could be considered inside the object. Thus when the model lies on a voxel with a greyscale level between \( T - \epsilon \) and \( T + \epsilon \), the model expands and otherwise it contracts. The speed term is gradual, and thus the effects of the \( D \) diminish as the model approaches the boundaries of regions whose greyscale levels lie within the \( T \pm \epsilon \) range. Even with this simple scheme a user would have to specify three free parameters, \( T, \epsilon \), and \( \alpha \), as well as an initialization. Fig. 1 shows a graph of \( D \) defined in this manner.

3.2. A statistical classifier

The speed term in (2) represents a simple 1D, two-class statistical classifier. If we let \( P(A|I) \) be the probability that a pixel lies in the object conditional on the pixel/voxel value \( I \) and \( P(B|I) \) be the probability that the pixel
is not in the object, then the Bayesian decision variable is

\[ \mathcal{D} = \frac{P(A|I)}{P(B|I)} = \frac{P(I|A)P(A)}{P(I|B)P(B)}, \] (3)

which should be compared to unity in order to decide on either A or B. If all intensities in the background, B, are equally likely (the goal here is simplicity), the denominator is constant, and the log of \( \mathcal{D} \) is

\[ \log \mathcal{D} = \log P(I|A) + \log P(A) - \log P(I|B) - \log P(B). \] (4)

If we let the statistics of the inside of the object be Gaussian \( P(A|I) = \exp[-(I - \mu)^2/2\sigma^2] \), we have the following decision rule for a pixel \( \tilde{x} \) with intensity \( I \):

\[ \tilde{x} \in \begin{cases} A & |I - T| \leq \epsilon, \\ B & \text{otherwise}, \end{cases} \] (5)

where \( \epsilon = [(2\sigma^2(\log P(A) - \log P(I|B) - \log P(B)))]^{1/2} \) and \( T = \mu \). Thus, we see that this simple three parameter model allows a user to explore the possibilities of this simple statistical classifier and combine it with the geometric information embodied in the curvature of the level set.

This analysis sheds some light on the proper interface for these parameters. For instance, we can help the user choose these parameters by providing an interactive tool that allows a user to select a set of points (e.g. by holding down a mouse button and moving the cursor over the area of interest) that generate a mean and a variance, and use these values initialize \( T \) and \( \epsilon \).

3.3. A speed function for spectral volumes

This statistical classifier also extends to image/volumes with multiple values, i.e. spectral images, with values denoted \( \tilde{I}(\tilde{x}) \in \mathbb{R}^m \). In this case the in-object condition probability is

\[ P(A|\tilde{I}) = \exp \left[-\frac{1}{2} (\tilde{I} - \tilde{\mu})^T \Sigma^{-1} (\tilde{I} - \tilde{\mu}) \right], \] (6)

where \( \Sigma \) is the covariance. We can express the classifier in terms of Mahalanobis distance. That is

\[ \tilde{x} \in \begin{cases} A & D \leq \epsilon, \\ B & \text{otherwise}, \end{cases} \] (7)

and \( D = [(\tilde{I} - \tilde{\mu})^T \Sigma^{-1} (\tilde{I} - \tilde{\mu})]^{1/2} \).

The graph of the speed function given in (7) is an ellipsoidal hypercone that crosses the zero axis of the independent variable to form an ellipsoid, centered at \( \tilde{\mu} \), with a shape and orientation given by the covariance. Fig. 2 depicts this for \( m = 2 \). The free parameter \( \epsilon \) defines the width of the resulting ellipsoidal classifier in units of standard deviation. The model expands when it lies on a pixel whose value is within the classifier range and contracts elsewhere. This Mahalanobis classifier is a natural extension of the scalar speed function that accounts for the correlation between different features of \( \tilde{I} \) and allows for curved decision boundaries. In the case where all features are uncorrelated Mahalanobis distance is equivalent to Euclidean distance.

For three-channel data such as color RGB images, the covariance matrix and mean vector comprise a total of 12 free parameters. The design of an effective user interface to allow simultaneous, interactive control of all 12 of these free parameters is a potentially difficult research question, and not one we try to answer with GIST. Instead, we allow the user to extract the mean and variance from a region of interest and provide the user with a single free parameter \( \epsilon \), which controls the size of the ellipsoidal region in the feature space and

![Diagram](image-url)
corresponds to the relative prior probabilities between the object and background.

### 3.4. The role of surface curvature

If a user were to initialize a model in a volume and use the speed terms in Eqs. (2)–(7) without any curvature the results would be virtually the same as a simple flood fill over the region bounded by the upper and lower thresholds (or the ellipsoid in the nD case). The inclusion of the curvature term, however, alleviates the critical leaking problem that can arise with flood filling when attempting to delineate object boundaries that show poor contrast to their background. This leaking effect is often particularly acute in 3D segmentations and is easily demonstrated on a brain tumor data set, as shown in Fig. 3. The degree to which leaking can be controlled with higher curvature depends on the level of detail required for the segmentation relative to the length of the weak boundary. Higher curvature produces smoother results, alleviating both the distorting effects of noise in the data and potential leaks due to weak boundary sections. Oversmoothing with curvature, however, can significantly distort the shapes of segmented objects, and if the weight of the curvature is too large with respect to the speed term, the model will pull away from the data.

### 3.5. Rescaling the distance function

When solving the PDE associated with Eq. (1), the different level sets of the function \( \phi \) tend to spread out in some regions of the volume (due to the curvature term and numerical diffusion) and aggregate in other areas (due to the speed term). These phenomena are characterized by a decreasing or increasing of \( |\nabla \phi| \) over time, respectively. Both of these tendencies will undermine the effectiveness of narrow-band algorithms, and therefore the literature describes mechanisms for maintaining \( \phi \) with a relatively constant gradient magnitude. For instance, in (Adalsteinson and Sethian, 1995), the authors stop the evolution of \( \phi \) at regular intervals and establish a new \( \phi \) that corresponds to the signed distance transform of the zero level set. To further alleviate problems of level-set aggregation, the speed term values in the narrow band may also be reinitialized at intervals by an extension of the values at the evolving interface (Sethian, 1996). In (Whitaker, 1998), the author updates the values of grid points around the zero-set of \( \phi \) in layers that maintain an approximation to the signed distance.

For the GPU-based solver, none of these strategies are appropriate because they entail pointer-based, dynamic data structures that cannot be easily implemented in the streaming architecture (Buck et al., 2004). Instead we maintain \( |\nabla \phi| \), by the addition of an extra term to the update equation (PDE) that governs the evolution of \( \phi \). This term will force the level sets of \( \phi \) to spread out if the gradient is too large and to move together if the gradient is too low. This rescaling term, \( G_\epsilon \), is of the form

\[
G_\epsilon = \phi (g_\phi - |\nabla \phi|) ,
\]

where \( g_\phi \) is the target magnitude for the gradient.

This rescaling term has several properties that are important to the implementation of GIST. First, the distance transform of the level set, scaled by \( g_\phi \), is formally (i.e. ignore points where \( \nabla \phi \) is undefined) a fixed point of (8). Second, because \( G_\epsilon \) is proportional to \( \phi \), it does not affect the values of \( \phi \) near the zero set, and therefore should not impact the evolution of the surface model. Finally, when \( G_\epsilon \) is implemented with the upwind scheme, it will maintain monotonicity, and therefore the fixed point of this term applied to updates of grid representing \( \phi \) will be a clamped distance transform with extreme values limited by those in the initial conditions. Thus, \( G_\epsilon \) will maintain the narrow-band property, which is to say that \( \phi \) will have \( |\nabla \phi| \approx g_\phi \) within a narrow band around the zero set and \( |\nabla \phi| \approx 0 \) elsewhere.

### 4. Software application design

This section describes GIST, an interactive level-set segmentation tool, and the GPU implementation that makes it possible. It begins with a brief overview of the GPU-based level-set solver, describes the visualization of the volume data and surface models, and then describes the user interface to GIST. The complete
description of the GPU level-set algorithm, including technical details of its implementation, is given in (Lefohn et al., 2004).

4.1. GPU level-set solver

The efficient solution of the level-set PDEs relies on only updating voxels that are on or near the isosurface. The sparse GPU level-set solver achieves this by decomposing the volume into a set of small 2D tiles (e.g. 16 x 16 pixels each). Only those tiles with non-zero derivatives are stored on the GPU (Fig. 4(b)). These active tiles are packed, in an arbitrary order, into a large 2D texture on the GPU. The 3D level-set PDE is computed directly on this compressed format. Because active tiles are identified by non-zero gradients, it is crucial that the volume in which the level-set surface is embedded, resemble a clamped distance transform. In this way regions on or near the model will have finite derivatives, while tiles outside this narrow band will be flat, with derivative values of zero. Thus, the rescaling term given in Eq. (8) is particularly important.

For each PDE time step update, the 3D neighbourhoods of all pixels in the active tiles must be sampled from the compressed 2D compressed format. For each active tile, the CPU sends texture coordinates, i.e. memory addresses, to the GPU for each of the tiles that share a side or an edge in the 3D volume. These texture coordinates are generated and maintained on the CPU. Using these texture coordinates, the GPU can perform neighbourhood lookups to produce the complete set of partial derivatives (finite differences) used for the gradient and curvature calculations, which are in turn used to update values of \( \phi \).

After the level-set embedding is updated, the GPU uses built-in, hardware accelerated, mipmapping capabilities to create a bit vector image that summarizes the status of each tile. Each pixel in this coarse texture contains a bit code that identifies if that tile, as well as any of its six cardinal neighbours, need to be active for the next time step. This small image (\( \leq 64 \) kB) is read back by the CPU and used to update the data structures that track the active volume regions. The texture coordinates are updated based on these structures and the next time step is computed.

This GPU-based level-set solver achieves a speedup of 10–15 times over a highly optimized, sparse-field, CPU-based solver. All benchmarks were run on an Intel Xeon 1.7 GHz processor with 1 GB of RAM and an ATI Radeon 9700 Pro GPU. For the tumor segmentations performed in the user study, the GPU-based solver ran at 60–70 steps per second while the CPU version ran at 7–8 steps per second. The final steps of the cerebral cortex segmentation shown in Fig. 10 ran at 4 steps per second on the GPU and 0.25 steps per second on the CPU.

4.2. Interactive visualization

An important aspect of GIST is the interactive visualization of the level-set surface, the volume data, and the speed function. This interactivity includes 2D slice-by-slice visualization of data, model and speed, as well as 3D volume/surface rendering with user-controlled clipping planes to visualize and query the volume data.

GIST provides a simultaneous volume visualization of the input data with the evolving level-set model. The volume renderer associated with GIST performs a full 3D (transfer-function based) volume rendering of the greyscale data. For rendering the original volume, the input data and its gradient vectors are kept on the GPU as 3D textures. This GPU-based volume rendering incorporates multi-dimensional transfer functions as described in (Kniss et al., 2002). The current implementation of GIST renders only scalar volume data, and thus for spectral data it renders only a derived scalar quantity (e.g. one component or magnitude). Future work will include the use of multi-dimensional transfer functions to directly render spectral data.

For rendering the evolving level-set model, we use a modification of the conventional 2D sliced approach to texture-based volume rendering (Cabral et al., 1994). The modification to the conventional approach is the rendering of the level-set solution directly from the packed tiles, which are stored as a single 2D texture. The level-set data and tile configuration is dynamic, and therefore does not require separate precomputed
versions of the data (e.g. sliced along cardinal views) as is typically done with 2D texture approaches. Instead the renderer reconstructs these views, as needed, each time the volume is rendered. For efficiency, the renderer reuses data wherever possible. For instance, lighting for the level-set surface uses gradient vectors computed during the level-set update stage. The rendering of the source data relies on precomputed gradient data – the gradient magnitude is used by the transfer function and the gradient direction is used in the lighting model. More details on this design are given in (Lefohn et al., 2004).

4.3. Interface and usage

GIST combines a graphical user interface (GUI), which controls the underlying GPU-based level-set solver, with a volume renderer. The GUI presents the user with two volume slices, a 3D rendering window, and a control panel. The first slice window displays the current segmentation as a yellow line overlaid on top of the target data. The second slice viewing window displays a visualization of the speed function that uses color to clearly delineate the positive and negative regions. The GUI has controls for scrolling through image slices, starting and stopping the solver, and saving the 3D segmentation to file. The user can also query data values in the slice viewer and create spherical surface models to use as initializations to the level-set solver. A screen capture of the slice-based interface is shown in Fig. 5.

To set the free parameters of the speed function, the user samples image values by clicking and dragging the mouse in regions of interest through the 2D slice view window (center window of 5). As the user gathers statistical samples, GIST simultaneously updates the mean value and the variance or covariance that defines the shape of the classifier. A user may probe the object of interest across a range of volume slices for a better representative sampling than can be obtained in just one slice. The remaining speed function parameter $\epsilon$ is set manually in the GUI. The speed function is updated and displayed in real time as parameters are modified to guide the process.

The volume renderer window displays a 3D rendering of the source data and a surface rendering of the evolving level-set model. The opacity of each rendering can be controlled by the user. A clipping plane with the original

Fig. 5. The main user interface of software application, called GIST. The center window shows a slice of an MRI volume overlaid by a brain tumor segmentation in progress. The right window displays the sign of the speed function.

Fig. 6. Two views of the volume rendering window from GIST. A brain cortex segmentation is shown at left with a cutting plane applied to the rendering on the right. The intersection of the level-set surface with the cutting plane is shown as a yellow band.
data can also be applied to the rendering in any orientation and position. All of the interactions available in the 2D slice view are also available on the clipping plane, e.g. the user can probe data to set the speed term parameters and draw spheres for initializing the model directly into the 3D view. The intersection of the level-set solution with the clipping plane is shown as a yellow band. Fig. 6 shows two views from the volume rendering window.

In a typical session with GIST, a user scrolls through slices until they find the location of the target object and then queries values with the mouse to set the speed function parameters. Next, the user creates an initial model by drawing one or more spheres within the object and then starts the solver. The user scrolls through slices as the model begins to deform, observing its behavior and modifying curvature (model smoothness) and classifier width as needed. The user may also stop the solver and resample the data to either refine or replace the current statistical speed function parameters. Using the immediate feedback they get on the behavior of the model, the user continues to modify parameters until the model boundaries appear to align with those of the tumor. In a typical 5-min session, a user may modify the model parameters between 10 and 30x.

5. Results

This section presents results from the application of our GPU-based level-set segmentation tool to a range of scalar and spectral data. The evaluations in this section include qualitative and quantitative comparisons with hand contouring as well as two other user-assisted methods. We choose hand contouring as the main focus of the comparison for several reasons. First, it is, like the proposed method, a general purpose segmentation method. Second, the field at large considers hand contouring by experts to be the de facto gold standard. Third, hand contouring is, in many cases, the state of the art. That is, a large number of clinical applications that require image segmentation still rely on hand contouring as their primary segmentation technique.

Section 5.1 gives a qualitative analysis of several anatomical segmentations from MRI and color cryosection data. A more rigorous, quantitative evaluation is presented in Section 5.2, which describes a user study of our software and compares results of brain tumor segmentations with ground truth obtained from experts.

5.1. Qualitative evaluation

As a preliminary evaluation of our segmentation tool, we segment a variety of anatomical structures in several imaging modalities: scalar and spectral MRI, and color cryosection data from the Visible Human Female (VHF) (Ackerman et al., 2000). This section presents results and discussion of those segmentations.

Fig. 7(a) is a rendering of a cortical brain surface segmentation from a 256 × 256 × 175 MRI volume. The complete segmentation required no preprocessing (e.g. no filtering) of the data and took 5 min using the 1D classifier speed function with a small, spherical surface (placed by the user) as the initial model. This type of segmentation is impractical to compute on ordinary, CPU-based solvers because of the size and complexity of the solution. In our experience with state-of-the-art CPU-based solvers (e.g. see the Insight Toolkit, www.itk.org) the same cortical segmentation typically takes more than an hour.

For MRI spectral data, we use volumes consisting of co-registered T1, T2, and proton density data. This combination of image modalities requires the 3D classifier, given in Eq. (7), to take full advantage of the wider spectrum of information. Fig. 7(b) shows a rendering of a segmentation of the white matter of the brain. As with the cortical segmentation, the results are encouraging because they can be obtained in only a few minutes (no preprocessing) with a simple spherical seed point initialization. We have seen similarly promising results.
with our tool segmenting skin and skull tissue from spectral MRI data.

For the VHF color cryosection data, we use a region of interest (cropped volume) from the head that contains two interesting structures: the right lateral rectus muscles, and anterior portions of the right and left optic nerves. The texture information in this data set posed a significant challenge, and therefore we preprocessed the data by smoothing with 10 iterations of modified-curvature diffusion (Whitaker and Xue, 2001). This diffusion step blurs the more homogeneous regions of the data while preserving object boundaries. This nonlinear diffusion is relatively computationally expensive, especially on spectral data, and it required approximately 20 min of computation on a two-processor Pentium IV desktop machine. The equations governing nonlinear diffusion are solved similarly to those governing level-set surface evolution, and work is in progress on a GPU-based implementation of nonlinear diffusion to reduce computation time.

Fig. 8 presents the results of our VHF anatomical segmentations and compares them with results obtained using other general-purpose segmentation methods. Column (a) shows a single slice of the original data with the target object for segmentation highlighted. Column (b) is a surface rendering of the results using our level-set tool. The results from (b) are overlaid on the slice in column (a). Expert segmentations of the same structures are shown in 8(b). The expert segmentations were obtained from multiple operators at Harvard Brigham and Women's Hospital and at the University of Utah using the Slicer Tool (MIT, 2004). The renderings shown in (b) are of composite ground-truth volumes created with the STAPLE method described in Section 5.2. Column (c) shows results obtained using another general, interactive segmentation method based on morphological watersheds segmentation method (for details see (Cates et al., 2004)).

Visual inspection of the GPU level-set results show them to be of similar quality as the hand-contour and watershed results. Anterior and posterior sections optic nerves in the area of the optic chiasm are segmented separately in this example and combined prior to rendering. The current speed function implementation in GIST is limited to a single statistical feature profile, and therefore distinct structures in color space such as the optic chiasm must be segmented separately.

Because of the curvature term in (1), segmentations created using our tool are naturally anti-aliased. The level-set technique also tends to produce a smoother boundary in the axial direction than the hand-contour and watershed methods, which tend to more resemble a stack of 2D slices with poor continuity of the boundary in the axial direction. The greatest advantage to the GPU level-set segmentation is its relative efficiency. The time taken for the VHF segmentations are up to 20× faster than hand-contouring (several minutes versus up to several hours) and were up to 6× faster than using the watersheds method (full processing time). As the following section will demonstrate, the level-set segmentation tool can generally produce acceptable results on the raw image data, which is not possible with many other algorithms, such as watershed segmentation; therefore the level-set segmentation tool is particularly useful for fast, impromptu segmentations of 3D data sets.

5.2. User study

5.2.1. Motivation

The purpose of this study is to determine if our level-set tool can produce volumetric delineations of brain tumor boundaries comparable to those done by experts (e.g. radiologists or neurosurgeons) using traditional hand-contouring. We apply our method to the problem of brain tumor segmentation using data from the Brain Tumor Segmentation Database, which is made available

![Fig. 8. Visual comparison of surface renderings of GPU level-set (b) and manual (c) segmentations of the Visible Human Female color cryosection anatomy. The targeted anatomical structure is highlighted in column (a), which shows the segmentation from (b) superimposed over a transverse slice through the original color data. Column (d) is a comparison with the user-assisted watershed technique.](image-url)
by the Harvard Medical School at the Brigham and Women’s Hospital (HBW) (Kaus et al., 2001; Warfield et al., 2000). The HBW database consists of 10 3D 1.5T MRI brain tumor patient datasets selected by a neurosurgeon as a representative sampling of a larger clinical database. For each of the 10 cases, there are also four independent expert hand segmentations of one randomly selected 2D slice in the region of the tumor.

We use 9 cases for our study: three meningioma (cases 1–3) and 6 low grade glioma (4–6, 8–10). One case, number 7, is omitted because a quick inspection shows that its intensity structure is too complicated to be segmented by the proposed tool – such a problem remains as future work, as we will discuss in Section 6. For this study, there is no preprocessing on the data and there are no hidden parameters in this study – all parameters in our system are set by the users in real time, as they interact with the data and the models.

The subjects consist of five people from among the staff and students in our group who have each been given a brief introduction on how to use the application. During the study, each user is asked to delineate the full, 3D boundaries of the tumor in each of the nine selected cases. We set no time limit on the users and record their time to complete each tumor. None of our users are experts in reading radiological data. It is not our intention to test for tumor recognition (tissue classification), but rather to test whether parameters can be selected for our algorithm to produce a segmentation which mimics those done by the experts. To control for tumor recognition, we allow each user to refer to a single slice from an expert segmentation. Users are told to treat this hand segmentation slice as a guide for understanding the difference between tumor and non-tumor tissue. Our assumption is that an expert would not need such an example.

5.2.2. Aggregation of expert segmentation data

The expert data serves two purposes in this study. First, it provides a mechanism for establishing a ground truth, against which we can compare the level-set segmentation. Second, the set of expert segmentations establish a performance benchmark for the accuracy, precision, and efficiency of hand contouring.

Ground truth is established from manual segmentations by the experts using the STAPLE algorithm (Warfield et al., 2000), an iterative EM algorithm that accounts for systematic biases in the behavior of experts. The STAPLE algorithm generates a fuzzy ground truth as well as sensitivity and specificity parameters for each expert and each case.

We denote a single subject within a population with the subscript $j$ and the pixels within the image/volume as $i$. An image of binary values $D_{ij}$ represents a segmentation for a particular subject. Given sensitivities $p_j$ and specificities $q_j$ for each subject, the degree of confidence that a particular pixel is in the target object is

$$W_i = \frac{g_i a_i}{g_i a_i + (1 - g_i) b_i},$$

where $g_i$ is the prior probability that any pixel would be classified as inside the target object (usually taken to be the fraction of the image that is filled by the object). The values of $\alpha$ and $\beta$ are

$$\alpha = \left[ \prod_j p_j D_{ij} \right] \left[ \prod_j (1 - p_j) (1 - D_{ij}) \right]$$

and

$$\beta = \left[ \prod_j q_j (1 - D_{ij}) \right] \left[ \prod_j (1 - q_j) D_{ij} \right].$$

Given a probability image $W_i$, the sensitivity/specificity for each subject can be updated as

$$p_j = \frac{\sum_i W_i D_{ij}}{\sum_i W_i} \quad \text{and} \quad q_j = \frac{\sum_i (1 - W_i) (1 - D_{ij})}{\sum_i (1 - W_i)}. \quad (11)$$

The full STAPLE algorithm entails iterating on these updates, back and forth between $(p, q)$ and $W$, until the process converges.

Accuracy is evaluated against aggregate volumes created for each segmented object by applying the STAPLE algorithm to the expert hand-contours. These aggregate (STAPLE) volumes consist of a graded membership function (zero to one). We analyze the accuracy of the experimental, level-set results by evaluating the sensitivity and specificity of each experimental subject, using Eq. (11), relative to these aggregate volumes. We can then make comparisons by computing average sensitivity and specificity for the two groups – subjects using hand contouring and subjects using the level-set GUI. Additionally, we can combine values of $p_j$ and $q_j$ to compute a total correct fraction for a subject:

$$c_j = \frac{\sum_i W_i D_{ij} + \sum_i (1 - W_i) (1 - D_{ij})}{\sum_i 1}. \quad (12)$$

Ideally we would compute accuracy of hand-contour segmentations using aggregate data from an independent group of expert segmenters. A characterization of the accuracy of a small group of manual segmentations using ground truth generated as a complete aggregate of those same segmentations contains a clear bias that over estimates the accuracy of the expert segmentations. A second, less conservative measurement that produces a more unbiased estimate of the manual segmentation accuracy is a round-robin leave-one-out strategy (Tou and Gonzalez, 1974), where $p$, $q$ and $c$ values for each $D_{ij}$ are computed using $W_k$ generated by all segmentations $k \neq j$.

Accuracy metrics must be interpreted carefully. Note that where a segmentation technique shows high sensitivity, there is a high confidence level in the results it
produces for negatively classified pixels, and where a technique shows high specificity, there is a high confidence level for positively classified pixels. The magnitudes of \( p \) and \( q \) are incommensurate because they are percentages of different populations of pixels. Total correct fraction is particularly difficult to interpret because it is biased by the ratio of the size of the image volume to the size of the target object. Where this ratio is high, \( c \) approaches \( q \). Where the ratio is low, \( c \) approaches \( p \). Total correct fraction is used in this study only as a way to compare our results with other published results on the same data.

We quantify precision in this study using the similarity \( s_{jk} \) of results from subjects \( j \) and \( k \),

\[
s_{jk} = \frac{2 \sum_i D_{ij} D_{ik}}{\sum_i D_{ij} + D_{ik}},
\]

and average similarity across all pairs of subjects \( j \neq k \). Accuracy, precision, and efficiency metrics were also applied across subjects. Given the limited resources for this study and the scarcity of manually segmented data, we were not able to make intra-subject comparisons, which require multiple segmentations from the same subject.

5.2.3. Discussion and analysis

Fig. 9 shows graphs of average \( p \), \( q \) and \( c \) values for the experts and the users in our study. Error bars represent the standard deviations of the associated values. This figure shows the average accuracy across all experts using round-robin ground truth.

The performance of the experts and our users varies case by case, but in almost all cases the performance of our users was within the range of performances of the experts. A comparison with expert-biased ground truth shows similar results. The average correct fraction of our users was better than the experts in 6 out of 9 cases. A general trend is that our users tended to underestimate the tumor relative to the experts, as indicated by lower values of \( p \) and higher values of \( q \), especially when compared to the round-robin expert averages. This is consistent with our experiences with hand segmentations and level set models – with hand contouring users tend to overestimate structures, and with level sets the curvature term tends to reduce the size of convex structures.

The segmentations in our study show a much higher degree of precision than the expert hand segmentations. Mean precision (Udupa et al., 2002) across all users and cases was 94.04 ± 0.04% while the mean precision across all experts and cases was 82.65 ± 0.07%. Regarding efficiency, the average time to complete a segmentation (all users, all cases) was 6 ± 3 min. Only 5–10% of this time is spent processing the level-set surface. This compares favorably with the 3–5 h required for a typical 3D segmentation done by hand.

The accuracy and precision of subjects using our tool also compares well with the automated brain tumor segmentation results of Kaus et al. (2001), who use a superset of the same data used in our study. They report an average correct volume fraction of 99.68 ± 0.29% (using...
the expert-biased ground truth), while the average correct volume fraction of our users was $99.78 \pm 0.13\%$. Their method required similar average operator times (5–10 min), but unlike the proposed method their classification approach required subsequent processing times of approximately 75 min. That method, like many other segmentation methods discussed in the literature, includes a number of hidden parameters, which were not part of their analysis of timing or performance.

These quantitative comparisons with experts pertain to a only single 2D slice that was extracted from the 3D segmentations. This is a limitation due to the scarcity of expert data. Our experience is that computer-aided segmentation tools perform relatively better for 3D segmentations because the hand contours typically show signs of interslice inconsistencies and fatigue.

Figs. 10(a)-(b) show a segmentation by an expert with hand contouring compared with a segmentation done by one of our subjects.

6. Summary and conclusions

A careful implementation of real-time visualization and a sparse level-set solver on a GPU provides new tool, called GIST, for interactive 3D segmentation. Users can manipulate several parameters simultaneously in order to find a set of values that are appropriate for a particular segmentation task. The quantitative results of using this tool for brain tumor segmentation suggest that it compares well with hand contouring and state-of-the-art automated methods. However, the tool as built and tested is quite general, and it has no hidden parameters. Thus, the same tool can be used to segment a variety of anatomy as was shown in Section 5.1.

The current limitations are mostly in the speed function and the interface. The speed function used in this paper is quite simple and easily extended, within the current framework, to include image edges and more complicated statistical classifiers. Future work will include development of a more intuitive 3D interface that could potentially improve user interaction times and accuracy.

References


