

[Department of Computer Science](#)
[Drexel University College of Engineering](#)

The following item is made available as a courtesy to scholars by the author(s) and Drexel University Library and may contain materials and content, including computer code and tags, artwork, text, graphics, images, and illustrations (Material) which may be protected by copyright law. Unless otherwise noted, the Material is made available for non profit and educational purposes, such as research, teaching and private study. For these limited purposes, you may reproduce (print, download or make copies) the Material without prior permission. All copies must include any copyright notice originally included with the Material. **You must seek permission from the authors or copyright owners for all uses that are not allowed by fair use and other provisions of the U.S. Copyright Law.** The responsibility for making an independent legal assessment and securing any necessary permission rests with persons desiring to reproduce or use the Material.

Please direct questions to archives@drexel.edu

Drexel University Libraries
www.library.drexel.edu

University Archives and Special Collections:
<http://www.library.drexel.edu/archives/>



<http://www.drexel.edu/>

Level Set and PDE Methods for Visualization

Notes for IEEE Visualization 2005 Course #6
Minneapolis, MN
October 25, 2005

Organizer

David Breen

Drexel University

Speakers

Mike Kirby

University of Utah

Aaron Lefohn

University of California, Davis

Ken Museth

Linköping University

Tobias Preusser

University of Bremen

Guillermo Sapiro

University of Minnesota

Ross Whitaker

University of Utah

Course Schedule

Session 1 - PDE and Level Set Fundamentals

8:30 Welcome - Breen

8:40 Introduction to PDEs with Applications - Sapiro

9:50 Introduction to Level Set Methods - Whitaker

10:00 Break

Session 2 – Level Set Software and Numerical Methods

10:30 Open Source Level Set Software (ITK) - Whitaker

10:45 Level Set Applications: Segmentation, Surface Processing and Reconstruction - Whitaker

11:30 Numerical Methods and Algorithms for PDEs - Museth

12:15 Lunch Break

Session 3 – Implementation Details

1:45 Level Set Applications: Segmentation and Reconstruction From Sparse Data - Museth

2:50 Introduction to GPU Computation - Lefohn

3:20 Level Set Application: Interactive Segmentation - Lefohn

3:45 Break

Session 4 – PDE Applications

4:15 Level Set Method for 3D Anisotropic Geometric Diffusion - Preusser

4:45 PDE Methods in Flow Field Post-Processing - Preusser

5:15 Algorithms for Solving Reaction-Diffusion Equations – Kirby

5:35 Reaction-Diffusion Models for Vector Visualization - Kirby

6:00 Course Ends

Table of Contents

Session 1 - PDE and Level Set Fundamentals

Introduction to PDE's in Image Processing, Visualization, Computer Vision and Computer Graphics (Slides)

G. Sapiro

Isosurfaces, Level Sets and PDEs (Slides)

R. Whitaker

Session 2 - Level Set Software and Numerical Methods

Fundamental Properties of Level Sets (Slides)

K. Museth

Session 3 – Implementation Details

Segmentation with Level Sets (Slides)

K. Museth

Introduction to GPU Computation (Slides)

A. Lefohn

Interactive Level-Set Deformation On The GPU (Slides)

A. Lefohn

Session 4 – PDE Applications

PDE Methods in Flow Field Post-Processing & Anisotropic Levelset Diffusion (Slides)

T. Preusser

Anisotropic Diffusion in Vector Field Visualization on Euclidean Domains and Surfaces

U. Diewald, T. Preusser, and M. Rumpf

Reaction-Diffusion Models for Vector Visualization: Algorithms and Implementations (Slides)

R.M. Kirby

Display of Vector Fields Using a Reaction-Diffusion Model

A. Sanderson, C. Johnson, and R.M. Kirby

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 level set and other PDE-based methods, and their application to visualization. The course material will be presented by several of the recognized experts in the field, and will include introductory concepts, practical considerations and extensive details on a variety of level set/PDE applications.

The course will begin with preparatory material that introduces the concept of using partial differential equations to solve problems in visualization. This will include the structure and behavior of several different types of differential equations, e.g. the level set, heat and reaction-diffusion equations, as well as a general approach to developing PDE-based applications. The second stage of the course will describe the numerical methods and algorithms needed to implement the mathematics and methods presented in the first stage, including information on implementing the algorithms on GPUs. Throughout the course the technical material will be tied to applications, e.g. image processing, geometric modeling, dataset segmentation, model processing, surface reconstruction, anisotropic geometric diffusion, flow field post-processing and vector visualization.

Prerequisites

Knowledge of calculus, linear algebra, computer graphics, visualization, geometric modeling and computer vision. Some familiarity with differential geometry, differential equations, numerical computing and image processing is strongly recommended, but not required.

Organizer Biography

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 50 research papers in these and other areas, as well as the book *Cloth Modeling and Animation*. Breen received a B.A. in Physics (Colgate University, 1982), and a Ph.D. in Computer and Systems Engineering (Rensselaer Polytechnic Institute, 1993).
E-mail: david@cs.drexel.edu

Speaker Biographies

Mike Kirby is an Assistant Professor in the School of Computing and is Director of the Computational Engineering and Science (CES) Program at the University of Utah. He is also affiliated with the Institute for Scientific Computing and Imaging. He received ScM degrees in computer science and applied mathematics and the PhD degree in applied mathematics from Brown University. His teaching and research focuses on large-scale scientific computation and visualization, with an emphasis on the scientific cycle of mathematical modeling, high-performance computation and parallelization, visualization, evaluation, and understanding. He has co-authored a textbook on parallel scientific computing, and numerous journal articles and book chapters spanning both theory and applications in computational engineering and science.
E-mail: kirby@cs.utah.edu

Aaron Lefohn is a Ph.D. student in computer science at the University of California at Davis and a graphics software engineer at Pixar Animation Studios. His research interests include general-purpose computation on graphics hardware and high-quality interactive rendering. His current research focuses on data structure abstractions for graphics hardware, and his M.S. thesis describes an interactive GPU-based level-set

segmentation and visualization system. Lefohn completed an M.S. in computer science at the University of Utah in 2003, an M.S. in theoretical chemistry from the University of Utah in 2001, and B.A. in chemistry from Whitman College in 1997. He is an NSF graduate fellow in computer science. E-mail: lefohn@cs.ucdavis.edu

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 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

Tobias Preusser studied Mathematics at the University of Bonn, Germany, and at New York University. In 1999 he finished his Diploma thesis on "Adaptive Methods in Large-Scale Image Processing" at the University of Bonn. During his PhD studies he worked on diffusion methods in scientific visualization and geometric image processing. In 2000 and 2001 he was a visiting researcher at the Max-Planck-Institute for Mathematics in the Sciences in Leipzig, Germany, and at Bologna University, Italy, respectively. Preusser obtained the PhD degree from the Department of Mathematics of the University of Duisburg, Germany. His thesis investigated anisotropic geometric diffusion in image and image-sequence processing. He is now a postdoctoral fellow at the Center for Complex Systems and Visualization at the University of Bremen, Germany, where he is continuing his research on PDE methods in image processing and scientific visualization, as well as image-based computing in the context of medical simulations. E-mail: tp@mevis.de

Guillermo Sapiro is a Distinguished McKnight University Professor with the Department of Electrical and Computer Engineering at the University of Minnesota. He received a B.Sc. (summa cum laude) and Ph.D. in Electrical Engineering from the Technion, Israel Institute of Technology, in 1989 and 1993 respectively. He was a Member of Technical Staff at HP Labs in Palo Alto, CA. He works on differential geometry and geometric partial differential equations, both in theory and applications in computer vision, computer graphics, medical imaging and image analysis. Sapiro

recently co-edited special issues of IEEE Image Processing and the Journal of Visual Communication and Image Representation. He has authored and co-authored numerous papers in this area and has written a book published by Cambridge University Press, January 2001. He has received the Gutwirth Scholarship for Special Excellence in Graduate Studies, the Ollendorff Fellowship for Excellence in Vision and Image Understanding, the Rothschild Fellowship for Post-Doctoral Studies, the ONR Young Investigator Award, the PECASE Award, and the NSF Career Award. E-mail: guille@ece.umn.edu

Ross Whitaker is an Associate Professor in the School of Computing at the University of Utah and is affiliated with the Institute for Scientific Computing and Imaging. He received a B.S. in Electrical Engineering and Computer Science (Princeton University, 1986) and a Ph.D. in Computer Science (University of North Carolina, Chapel Hill, 1993). He was a research staff member at the European Computer-Industry Research Centre, and an Assistant Professor at the University of Tennessee. He teaches and conducts research in computer vision, image processing, medical imaging, and computer graphics/visualization. He has published numerous papers and book chapters on PDE/level set methods for image processing and computer graphics. E-mail: whitaker@cs.utah.edu

Level Set / PDE Web Sites

Sapiro Home Page

<http://www.ece.umn.edu/users/guille>

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>

Museth Home Page

<http://gg.itn.liu.se>

Lefohn Home Page

<http://graphics.cs.ucdavis.edu/~lefohn>

Preusser Home Page

<http://www.mevis.de/~tp>

Kirby Home Page

<http://www.cs.utah.edu/~kirby>

Breen - Geometric Modeling and Deformable Models

http://www.cs.drexel.edu/~david/geom_mod.html

http://www.cs.drexel.edu/~david/deform_mod.html

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

- J. Becker, T. Preusser, and M. Rumpf, "PDE Methods in Flow Simulation Post Processing," *Computing and Visualization in Science*, 2000.
- D. Breen, R. Whitaker, K. Museth and L. Zhukov, "Level Set Segmentation of Biological Volume Datasets," *Handbook of Medical Image Analysis*, 2005.
- J. Cates, A. Lefohn and R. Whitaker, "GIST: An Interactive, GPU-Based Level-Set Segmentation Tool for 3D Medical Images," *Medical Image Analysis*, 2004.
- U. Diewald, T. Preusser, and M. Rumpf, "Anisotropic Diffusion in Vector Field Visualization on Euclidian Domains and Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, 2000
- A. Lefohn, J. Kniss, C. Hansen and R. Whitaker, "A Streaming Narrow-Band Algorithm: Interactive Computation and Visualization of Level Sets," *IEEE Transactions on Visualization and Computer Graphics*, 2004.
- A. Lefohn, J. Kniss, R. Strzodka, S. Sengupta and J. Owens, "Glif : An Abstraction for Generic, Efficient GPU Data Structures," *ACM Transactions on Graphics*, 2005.
- F. Memoli and G. Sapiro, "Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-Surfaces," *Journal of Computational Physics*, 2001.
- K. Museth, D.E. Breen, R.T. Whitaker and A.H. Barr, "Level Set Surface Editing Operators," *Proc. SIGGRAPH 2002 Conference*.
- K. Museth, D.E. Breen, R.T. Whitaker, S. Mauch and D. Johnson, "Algorithms for Interactive Editing of Level Set Model," *Computer Graphics Forum*, 2005.
- M. Nielsen and K. Museth, "Dynamic Tubular Grid: An Efficient Data Structure and Algorithms for High Resolution Level Sets," to be published in *Journal of Scientific Computing*, 2005.
- T. Preusser and M. Rumpf, "A Level Set Method for Anisotropic Geometric Diffusion in 3D Image Processing," *SIAM Journal of Applied Mathematics*, 2002.
- A. Sanderson, C. Johnson, R.M. Kirby, "Display of Vector Fields Using a Reaction-Diffusion Model," *Proc. IEEE Visualization 2004*.
- T. Tasdizen, R. Whitaker, P. Burchard and S. Osher, "Geometric Surface Processing via Normal Maps," *ACM Transactions on Graphics*, 2003.
- R.T. Whitaker, "A Level-Set Approach to 3D Reconstruction From Range Data," *International Journal of Computer Vision*, 1998.
- R.T. Whitaker, "Isosurfaces and Level-Sets," *The Visualization Handbook*, Eds. C. Hansen and C. Johnson, Elsevier, 2005

Related Books

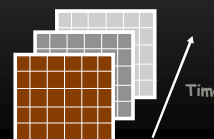
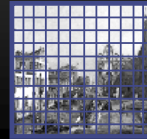
- S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York, 2002.
- S. Osher and N. Paragios (eds.), *Geometric Level Set Methods in Imaging, Vision and Graphics*, Springer, New York, 2003.
- G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, 2001.
- J. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 1999

Introduction to PDE's in image processing, visualization, computer vision, and computer graphics

Guillermo Sapiro

Electrical and Computer Engineering
University of Minnesota
guille@ece.umn.edu

What is a **discrete** computer image?



Consequences of discrete image representations

- Classical image processing and computer vision is based on discrete mathematics (most of it)
 - Sums instead of integrals
 - Re-definition of classical continuous operators as a Laplacian, Minkowsky addition, etc

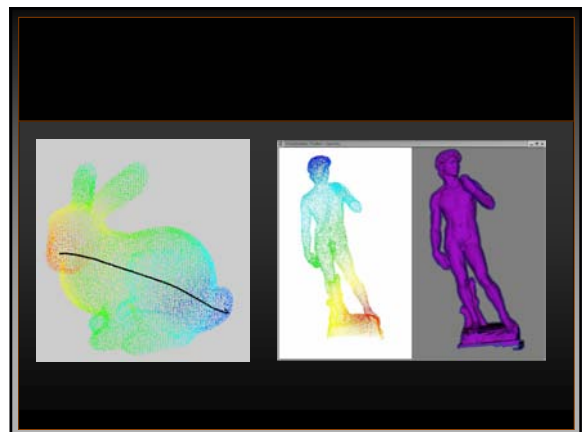
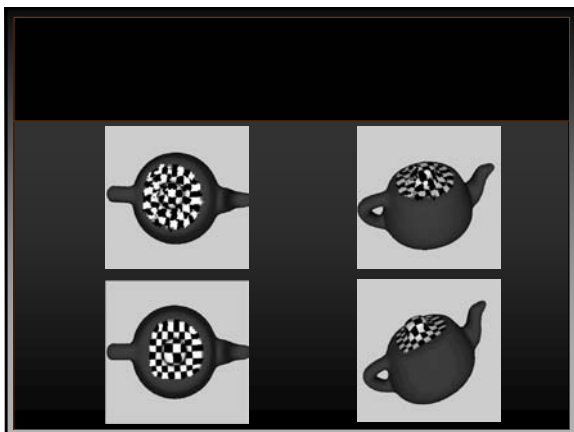
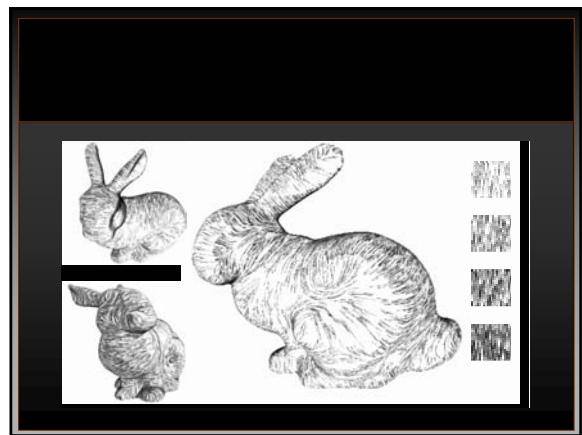
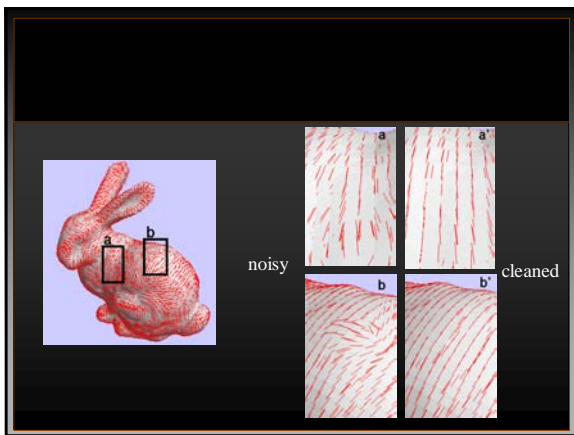
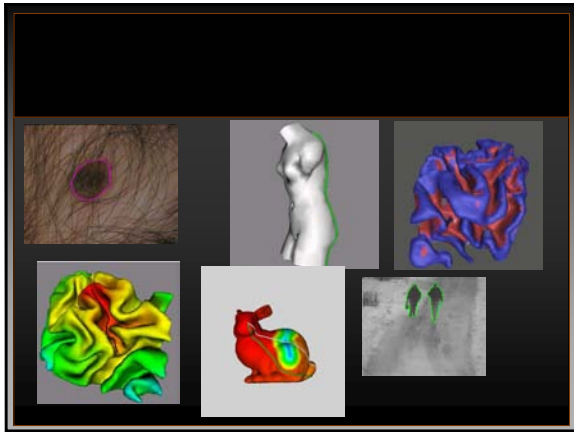
The PDE's approach

- Images are **continuous** objects
- Image processing is the results of **iteration of infinitesimal operations: PDE's**
- **Differential geometry** on images
- **Computer** image processing is based on **numerical analysis**

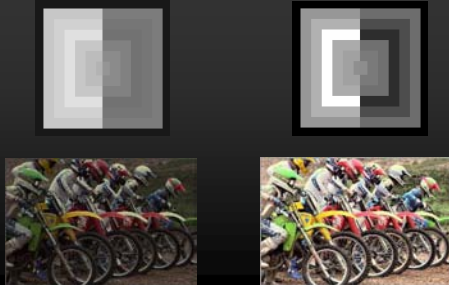
Why? Why Now? Who?

- **Why now:**
 - Computers!!!
 - People
- **Why:**
 - New concepts
 - Accuracy
 - Formal analysis (existence, uniqueness, etc)
- **Consequences:**
 - Many state of the art results

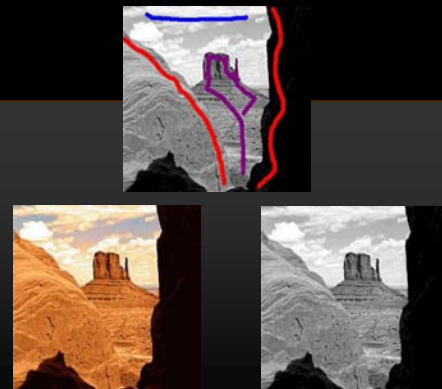
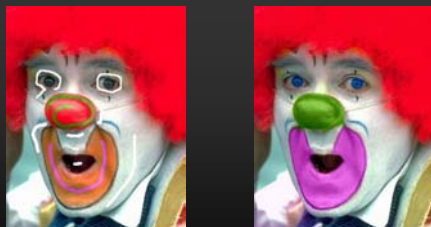
**By this afternoon,
we will be able to ...**

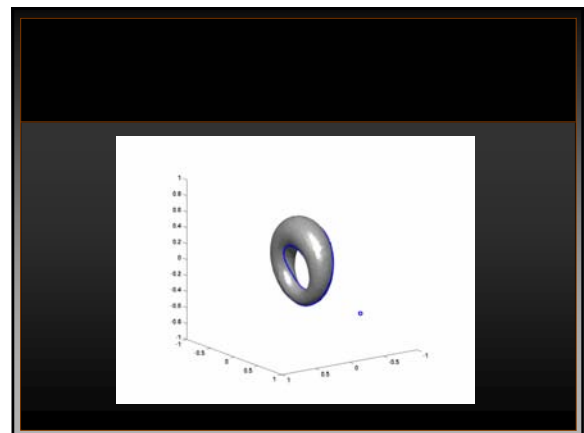
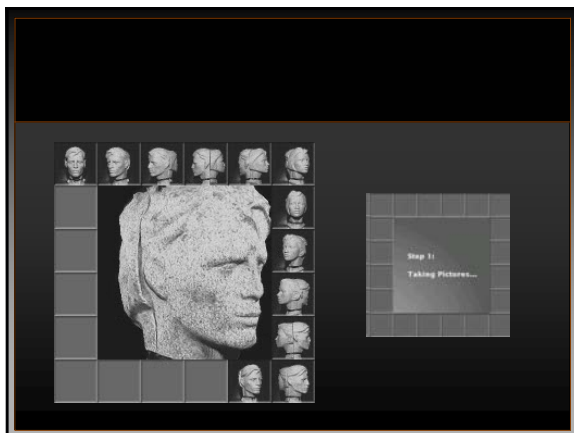
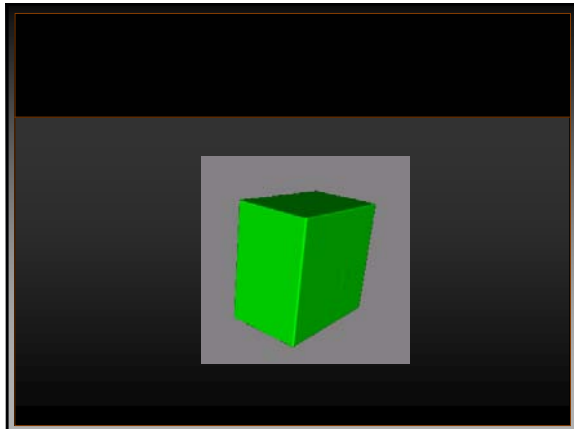


Shape preserving contrast enhancement



Examples





So, please stay...
(or buy my book 😊)

Publisher: Cambridge
U. Press
ISBN 0521790751

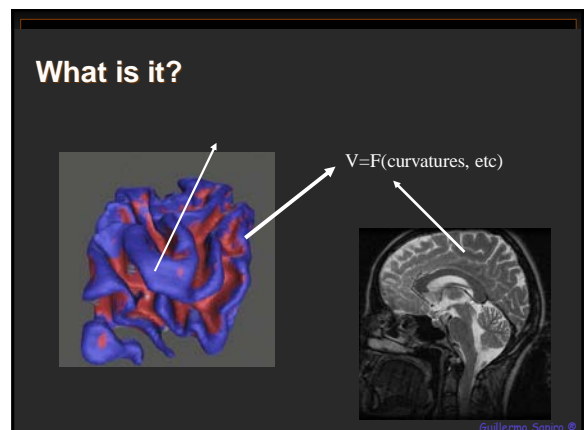


Image Inpainting: An Overview

mountains.ece.umn.edu/~guille/inpainting.htm

Overview

- Goal and background
 - Art, biology, math, and engineering come together
- Related work
- Inpainting
- Filling-in
- Inpainting and image decomposition
- 3D surface filling-in

What is inpainting?

- Modifying an image in a non-detectable form



"Cornelia, Mother of the Gracchi" by J. Suvee (Louvre). Emile-Male "The Restorer's Handbook of easel painting".

Another example



From Geary Gallery

Real world example: Photo restoration



• Restorations courtesy of Photo Imaging Studio, Image Enigma, Alleycat Designs

Real world example: Object removal



• From D. King, "The Commissar vanishes".

Real world example: Object removal



- From D. King, "The Commissar vanishes".

Real world example: Object removal



Lenin and friend Trotsky

Where is Trotsky?

- From www.newseum.org

The goal



Related work: Films

- e.g. Kokaram et al., Geman et al.



- Doesn't work for stills or static objects

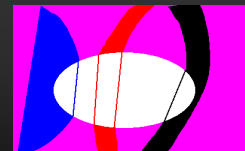
Related work: Texture synthesis



- Hirani, Efros, Heeger, DeBonet, Simoncelli, Zhu, etc.
- Not practical for rich regions
- Not (originally) designed for structured regions
- "Copy" information instead of "see and interpolate"

Related work: Disocclusion

- Masnou-Morel, Nitzberg-Mumford, etc.



- Limitations: Topology, angles

See also Jacobs, Basri, Zucker, etc, and Chan-Shen '00, Zhu-Mumford

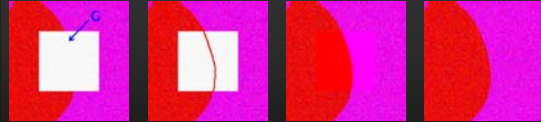
Our Contribution

- User only selects region to inpaint
- Rich background and topology not an issue
- Less than 1 minute on a PC



How conservators inpaint

- Minneapolis Institute of Art



Approach 1: PDEs

*Bertalmio, Sapiro, Caselles, Ballester,
SIGGRAPH 2000*

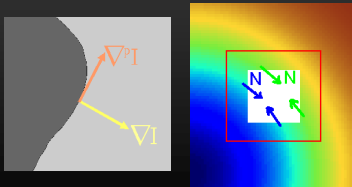
Automatic digital inpainting



- Propagate information $\nabla L \cdot \vec{N} = 0$
- Evolutionary form $\frac{\partial I}{\partial t} = \nabla L \cdot \vec{N}$

Digital inpainting (cont'd)

- L = smoothness estimator (Laplacian)
- N = isophote direction (time variant)



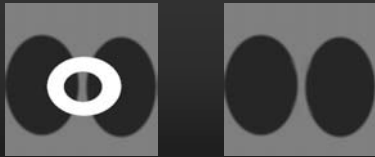
The equation

$$\frac{\partial I}{\partial t} = \nabla(\Delta I) \cdot \nabla^\perp I$$

- Plus numerical schemes (Osher-Marquina)
- Boundary conditions
 - Gray values (in a band)
 - Directions (in a band)



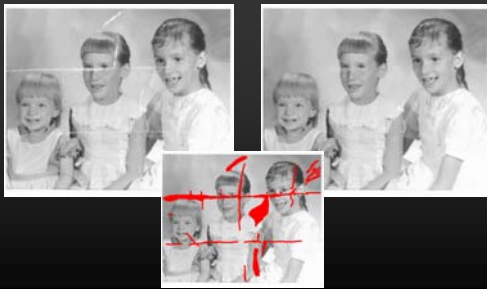
Example



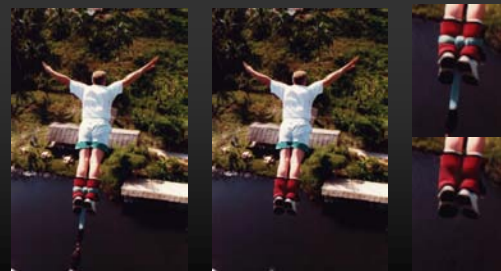
Example: Text removal



Example: Photo restoration



Example: Special effects



Example: Special effects



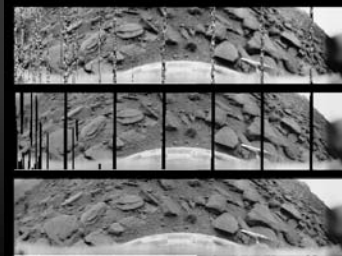
Example: Special effects



Example: Scratch removal

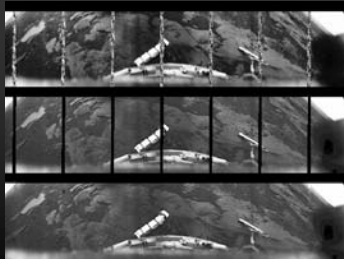


Russian Venus Mission Venera 9



From Don Mitchell

Russian Venus Mission Venera 10



From Don Mitchell

Automatic image inpainting/interpolation for compression and wireless transmission (Rane-Sapiro-Bertalmio) JPEG and/or JPEG-2000 compatible

Transmitted



Automatic image inpainting/interpolation for compression and wireless transmission (Rane-Sapiro-Bertalmio) JPEG and/or JPEG-2000 compatible

Automatic
reconstruction



Approach 1: Concluding remarks

- **Technique imitates professionals**
- **Key concepts**
 - Information propagation
 - Both gray values and directions are needed
 - Use a band surrounding the region
- **Sharp results**
- **Low complexity**
- **Texture is not (yet) reproduced**

Concluding remarks (cont.)

- Connected to fluid dynamics (see Bertalmio-Bertozzi-Sapiro CVPR 2001)

$$\frac{\partial(\Delta I)}{\partial t} = \nabla(\Delta I) \cdot \nabla I^\perp$$

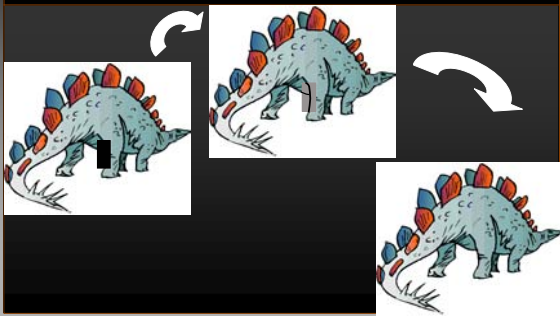
- Opens then door to high order PDE's
- Extended to a variational formulation: Approach 2...

Approach 2: Variational

*C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera,
IMA Report 2000, IEEE Trans. IP 2001*

How conservators fill-in

(Minneapolis Institute of Art)

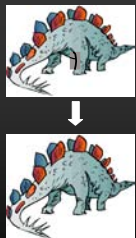


Our approach

- Jointly continue/interpolate level-lines (geometry) and gray values (photometry) in a smooth fashion



Interpolate the gray values given the edges



$$\begin{aligned} \theta &= \text{normalized gradient} \Rightarrow \theta \cdot \nabla I = \|\nabla I\| \\ \min(I) \int_{\Omega \setminus \text{Band}} \|\nabla I\| - \theta \cdot \nabla I \, d\Omega \\ \frac{\partial I}{\partial t} &= \text{div} \left(\frac{\nabla I}{\|\nabla I\|} \right) - \text{div}(\theta) \end{aligned}$$

Theorem: The minimizer exists in BV space

Example

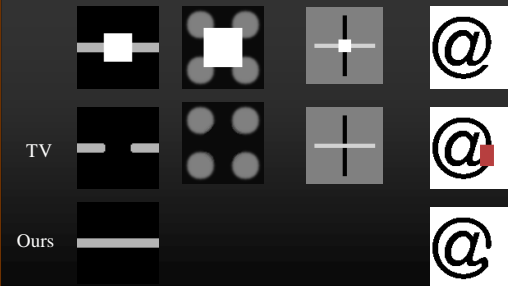


The full functional

$$\min(I, \theta) \int_{\Omega \setminus \text{Band}} \text{div}(\theta)^p (a + b \|\nabla G * I\|) + c(\|\nabla I\| - \theta \bullet \nabla I)$$

- Solved via E-L: Coupled 2nd order PDE's
- Implicit discretization used
- Connected to Euler's elastica (Mumford)
- **Theorem:** For $p > 1$ the minimizer exists

Examples



Examples



Examples



Examples



Approach 2: Concluding remarks

- Technique imitates professionals
- Key concepts
 - Information propagation
 - Both gray values and directions are needed
 - Use a band surrounding the region
- Sharp results
- Low complexity
- Texture is not (yet) reproduced

Inpainting and Image Decomposition: PDEs + Variational

Bertalmio, Vese, Sapiro, Osher, July 2002
IEEE Trans. IP, 2003

Basic Idea

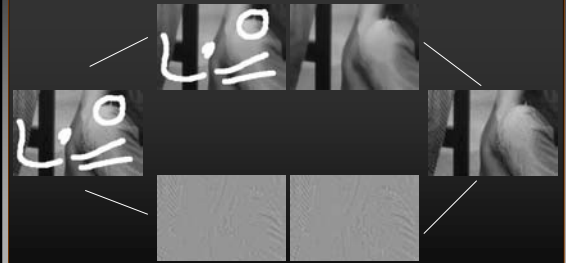


Image decomposition

Definition (Meyer) G = Banach space of generalized functions $v(x, y)$:

$$v(x, y) = \partial_x g_1(x, y) + \partial_y g_2(x, y), \quad g_1, g_2 \in L^\infty(R^2),$$

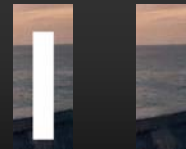
$$g := (g_1, g_2), |g(x, y)| = \sqrt{g_1(x, y)^2 + g_2(x, y)^2},$$

and use the infimum over all possible decompositions.

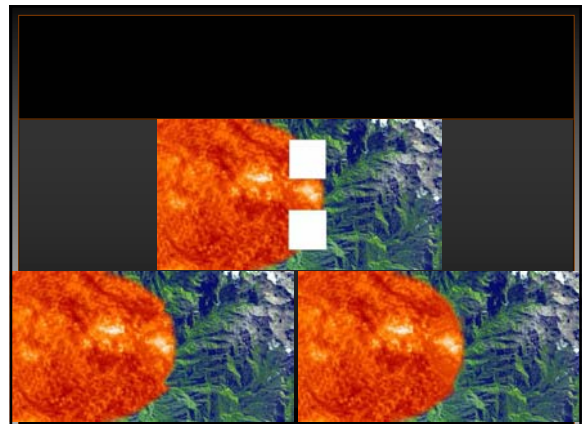
Definition (Vese-Osher) The image I is decomposed in *structure* (u) and *texture* (v) solving via gradient descent the variational problem

$$\inf_{u, g_1, g_2} \left\{ G_p(u, g_1, g_2) = \int |\nabla u| + \lambda \int |I - u - \partial_x g_1 - \partial_y g_2|^2 dx dy \right. \\ \left. + \mu \left[\int \left(\sqrt{g_1^2 + g_2^2} \right)^p dx dy \right]^{\frac{1}{p}} \right\}$$

Example



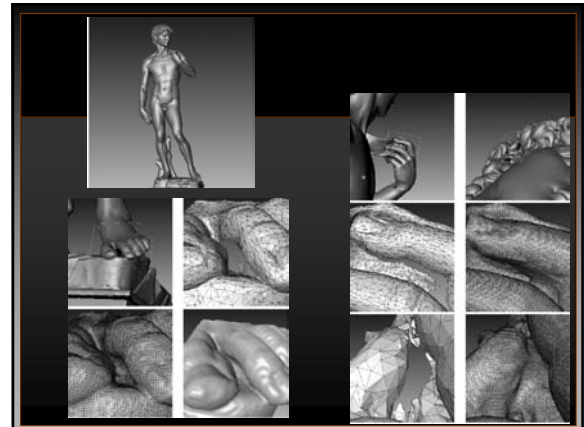
Example



Filling surface holes

Verdera, Bertalmio, Caselles, Sapiro,
IEEE ICIP 2003

Data and inspiration from Levoy and the
Michelangelo Project



Inpainting from Sensor Arrays

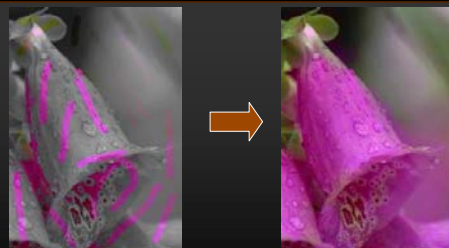
Yatziv, Sapiro, Levoy
IEEE ICIP 2004

Example



Inpainting the Colors

Colorization



Inspired by work of Levin et al.

Basic Idea

- **Inpaint with**

- Edges from monochromatic image (Chung-Sapiro, Caselles et al, Kimmel)
- Boundary conditions from given color strokes

$$\min_{Cb} \int_{\Omega} \rho(\|\nabla Y - \nabla Cb\|) d\Omega$$

$$\Delta Cb = \Delta Y$$

See also Caselles et al., Kenney et. al., Perez et al.

Basic idea (cont.)

$$\min_{Cb} \int_{\Omega} \rho \left(\frac{\nabla Y}{\|\nabla Y\|} \cdot \nabla Cb - \|\nabla Cb\| \right) d\Omega$$

$$\operatorname{div} \left(\frac{\nabla Cb}{\|\nabla Cb\|} \right) = \operatorname{div} \left(\frac{\nabla Y}{\|\nabla Y\|} \right)$$

Example



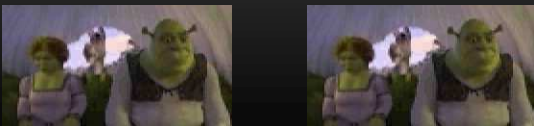
Example



Movies

- **Use 2D+time gradients**

- Color constancy
- All channels “sharing” spatial gradients
- All channels having same motion vectors



Conclusion

- Inpainting 2D and 3D via PDEs (flows)
- Inpainting in a decomposition space
- Inpainting light-fields
- Inpainting the colors

- See also recent works such as Tensor Voting (CVPR'03), Edge directed Eiros (CVPR'03), Global inpainting (ICCV'03).

Isosurfaces, Level Sets, and PDEs

Ross Whitaker
SCI Institute, School of Computing
University of Utah



The Next Several Talks

1. Implicit surfaces and level-set geometry
2. Level sets, numerical schemes, and software
3. Applications of level sets and PDEs to surface/volume processing



Overview

- Introduce implicit surfaces/level sets
- Geometry of level sets
- Application of level-set geometry



Isosurfaces

- Implicit representation

$$F : U \mapsto \mathbb{R} \\ x, y, z \mapsto k$$

- Domain of volume - where surface lives

$$U \subset \mathbb{R}^3$$

- Surface S is set of points

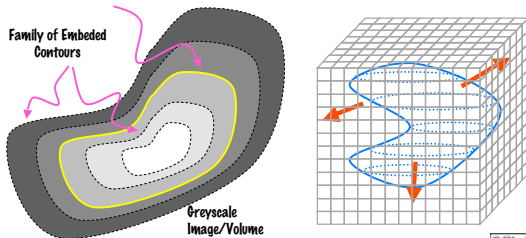
$$S = \{\bar{x} | F(\bar{x}) = k\}$$



Isosurfaces

- Implicit formulation $F(\bar{x}) = k$

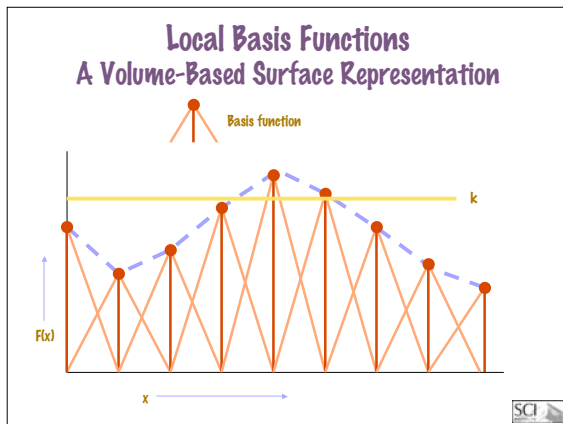
Curve/Surface Model



How Do We Represent $F(x)$?

- Linear combination of global basis functions
 - "Blobby" models [Blinn 82]
 - Deformation by modifying size, position, number, etc. [Muraki 91]
- Linear combination of local basis functions
 - Local deformations defined by neighborhood
 - Many degrees of freedom-arranged on grid
 - Well defined relationship between surface motion and grid values





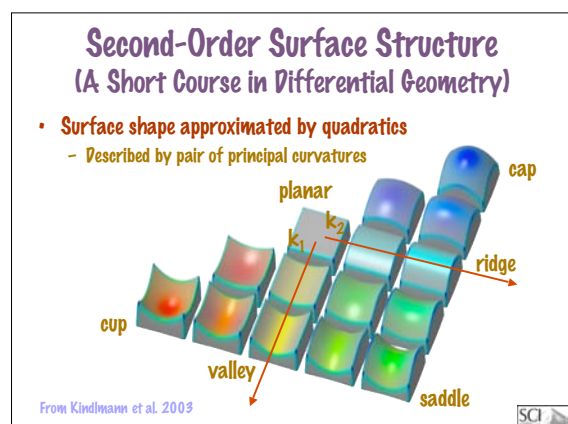
- ### Local Basis Functions
- Geometry defined by *local* operations
 - Continuous mathematics on $F(x)$
 - Grid value (voxel) manipulations determined by well-defined numerical methods
 - Level-set method [Osher & Sethian 88]

- ### Visualizing Level Sets
- Direct volume rendering methods
 - E.g. cast rays from viewpoint
 - Transfer functions, root finding
 - Extract surface primitives
 - E.g. marching cubes [Lorensen & Cline, 87]
 - Others
 - *Not* subject of this talk

- ### Geometry of Isosurfaces
- Surface normals
 - Curvature
 - Goal: express surface geometry in terms of derivatives of $F()$

- ### Surface Normals
- Exist for every point in U

$$\bar{n}(\bar{x}) = \frac{\nabla F(\bar{x})}{|\nabla F(\bar{x})|} \text{ where } \bar{x} \in U.$$
 - Gives normal to level-set passing through that point
 - Convention - inside or out (be consistent)
 - How to compute? (e.g. central differences)



Surface Curvature

- Principle curvatures k_1, k_2
- Principle directions \bar{e}_1, \bar{e}_2
- Invariants
 - Gaussian curvature $K = k_1 \times k_2$
 - Mean curvature $H = \frac{k_1 + k_2}{2}$
 - Deviation from flatness $D = \sqrt{k_1^2 + k_2^2}$
 - total curvature



Curvature of Isosurfaces

- Projection operator (tangent plane)

$$P = I - \bar{n} \otimes \bar{n} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} n_x n_x & n_x n_y & n_x n_z \\ n_y n_x & n_y n_y & n_y n_z \\ n_z n_x & n_z n_y & n_z n_z \end{pmatrix}$$

- Hessian of FO $D^2 F = \begin{pmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{xy} & F_{yy} & F_{yz} \\ F_{xz} & F_{yz} & F_{zz} \end{pmatrix}$

- Curvature (matrix) given by projected, normalized Hessian of FO

$$W = \frac{P(D^2 F)P}{|\nabla F|}$$

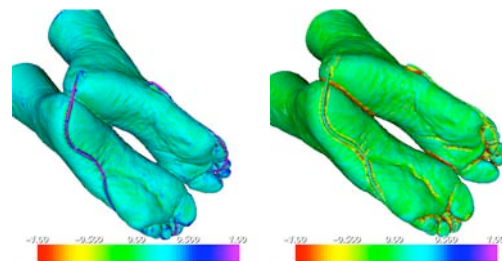


Curvature of Isosurfaces

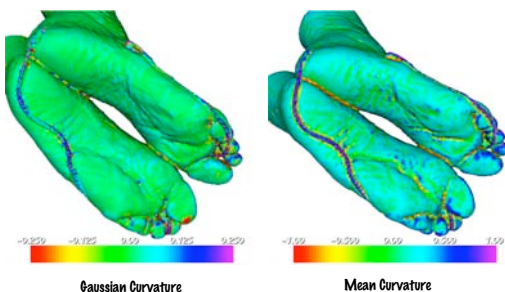
- Eigenvalues of $W \rightarrow k_1, k_2, 0$
- Eigenvectors of $W \rightarrow e_1, e_2, n$
- Trace of $W \rightarrow 2H$
- Norm of $W \rightarrow D$
- $K = 4H^2 - D^2$
- Note: derivatives must be taken using appropriately smooth basis functions



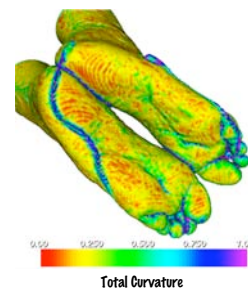
Example – Principle Curvatures



Example

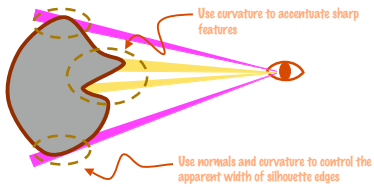


Example

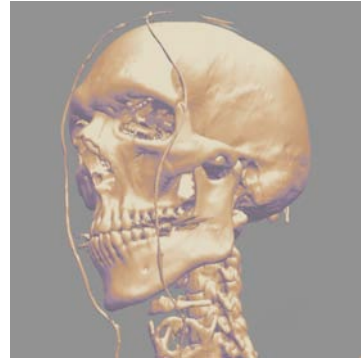


Application of Isosurface Curvature

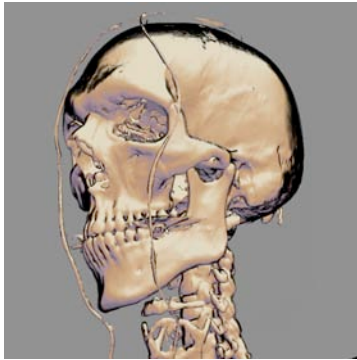
- Kindlemann et al., Vis 2003
- Use curvature for NPR volume rendering
 - Controlling thickness of silhouette edges
 - Accentuating ridges/valleys



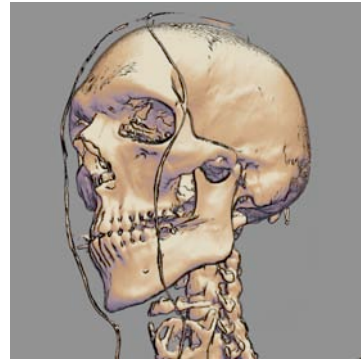
Volume NPR: results



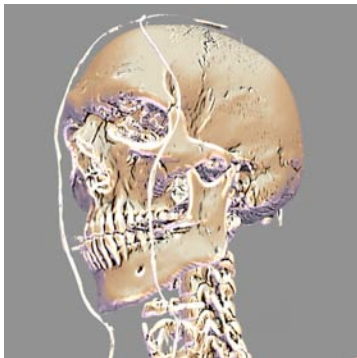
Volume NPR: results



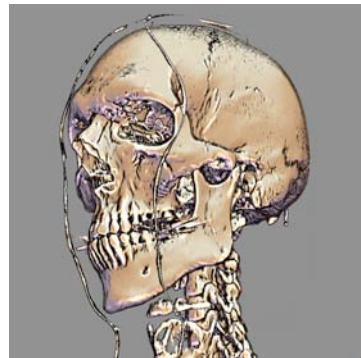
Volume NPR: results



Volume NPR: results



Volume NPR: results



Summary

- Implicit surfaces
- Volumes
- Geometry of level sets



The Next Several Talks

1. Implicit surfaces and level-set geometry
2. Level sets, numerical schemes, and software
3. Applications of level sets and PDEs to surface/volume processing



Level Set Introduction



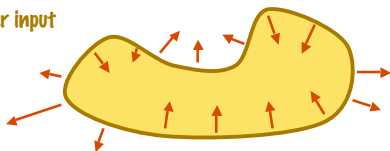
Overview

- Deformable models
- Level-set equations
- Numerical/computational techniques
- Software



Freeform Deformable Surfaces

- E.g. "Snakes" [Kass et al. 86]
- Velocity $v()$ at each point x on S
- Where does $v()$ come from?
 - Data (e.g. attraction to edges in images)
 - Geometry (e.g. curvature, smoothness)
 - User input



Level Sets – Moving Isosurfaces

- Osher and Sethian 1988
 - Method for modeling moving wave fronts
 - Formulation and numerical scheme
- Strategy
 - Function $F()$ encodes the motion of the moving interface
 - Allows for a great deal of flexibility of shapes and topologies



Static vs Dynamic Formulation

- **Static**

- Single F , k varies
- Motion strictly inward or outward
- Fast marching method (O(N lg M)) - Sethian 95

$$F(\bar{x}) = k(t)$$

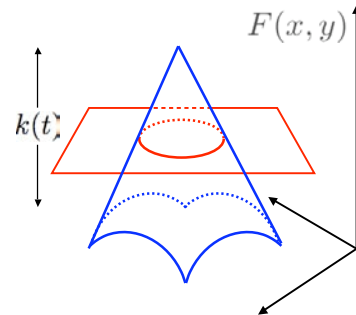
- **Dynamic**

- Evolving F , k fixed
- General motions
- Front tracking schemes

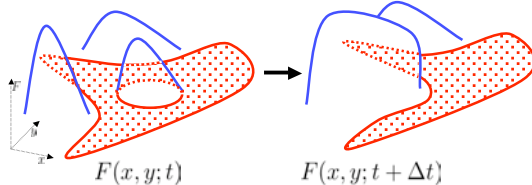
$$F(\bar{x}, t) = k$$



Static Formulation



Dynamic Formulation



Level-Set Equation Derivation

Total derivative is zero

$$\frac{d}{dt} F(\bar{x}(t), t) = \frac{dk}{dt} = 0$$

Chain rule

$$\frac{d}{dt} F(\bar{x}(t), t) = \frac{\partial F(\bar{x}, t)}{\partial \bar{x}} \frac{d\bar{x}}{dt} + \frac{\partial F(\bar{x}, t)}{\partial t}$$

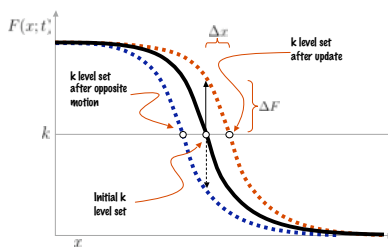
Level set equation (PDE)

$$\frac{\partial F}{\partial t} = -\nabla F \cdot \bar{v}$$



Level-Set Equation

$$\frac{\partial F}{\partial t} = -\frac{\partial F}{\partial x} v \quad \frac{\partial F}{\partial t} = -\nabla F \cdot v$$



Numerical Issues

- Analytical expressions approximated on discrete grid u_{ij}^n
 - Finite forward differences in time

$$u_{ij}^{n+1} = u_{ij}^n + \Delta t \Delta u_{ij}^n$$

- Spatial derivatives approximated using kernels or stencils

$$\frac{\partial F}{\partial x} \approx \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2} \rightarrow \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

- Special care must be taken in the first-order derivatives in the LS equation (up-wind scheme)

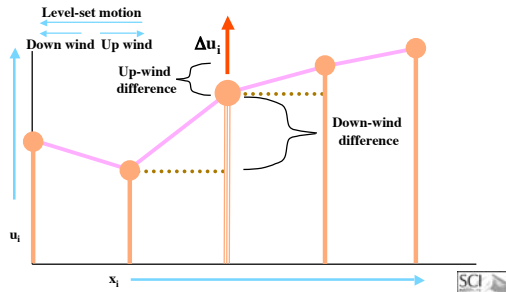
- Must maintain monotonicity -> one-sided derivatives

$$\begin{bmatrix} 0 & 1/2 & -1/2 \\ 1/2 & -1/2 & 0 \end{bmatrix}$$

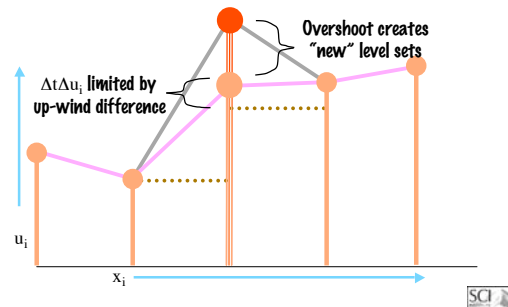


Updates Proportional to 1st Derivatives

- Avoid creation of new level sets

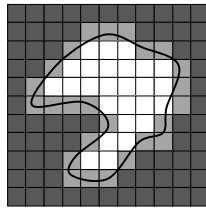


Update Moves Value in Direction of (But No Farther Than) Up-Wind Neighbor



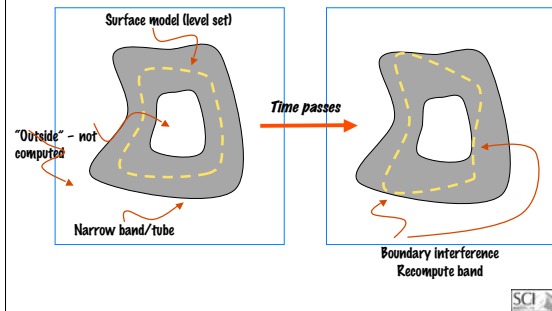
Efficient Computational Schemes: Front Tracking

- Solution important only in proximity to ls (wf) of interest
- Maintain computational domain that moves with wf
- Computational cost grows with surface area not volume



Narrow-Band Method

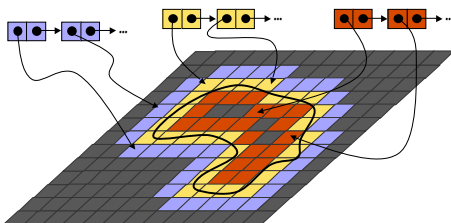
Adalsteinson and Sethian 1995



Sparse-Field Method

Whitaker 1998

- Update computational domain in layers



See also: Peng et al. 1999

Public Domain Software Insight Toolkit (ITK)

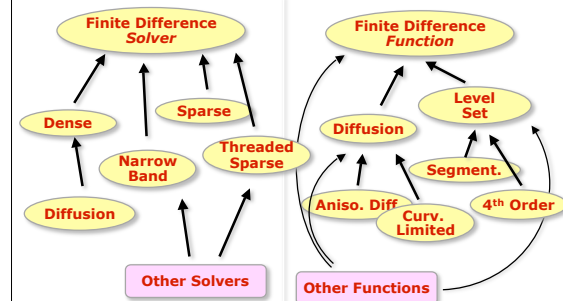
- Large toolkit for multidimensional image processing
- NIH sponsored consortium—industry and academics
- Includes API (algorithms) and applications
- www.itk.org
- Includes a framework for PDE-based image processing in N dimensions
- Others:
 - Vispack
 - Ian Mitchel <http://www.cs.ubc.ca/~mitchell/ToolboxLS>

ITK PDE Solvers

- **Purpose**
 - Nonlinear image processing - e.g. anisotropic diffusion
 - Moving wave fronts - level set models
- **Generic framework**
 - Separate solvers from equations (plug & play)
 - Include instances/examples in toolkit



ITK PDE Solver Hierarchy



Constructing a PDE Filter

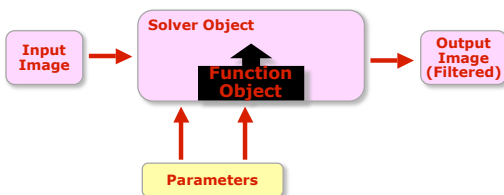
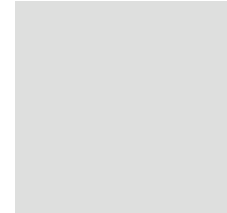


Image Processing Nonlinear PDE Filtering

- N-dimensional implementations
- Scalar and Vector based methods (e.g. color, MRI)
- Anisotropic (P&M) and curvature-based variations



Anisotropic Diffusion Framework

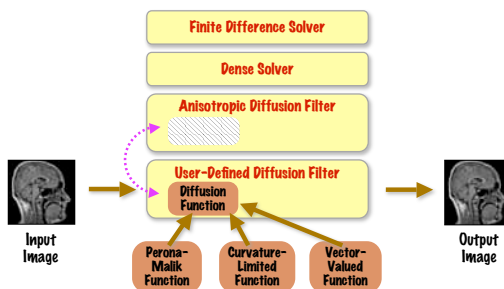
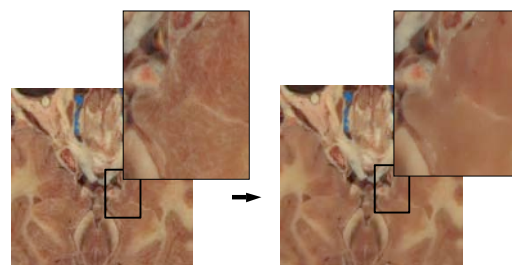
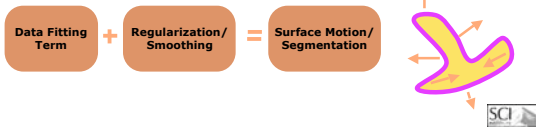


Image Processing Vector-Valued Anisotropic Diffusion Filtering

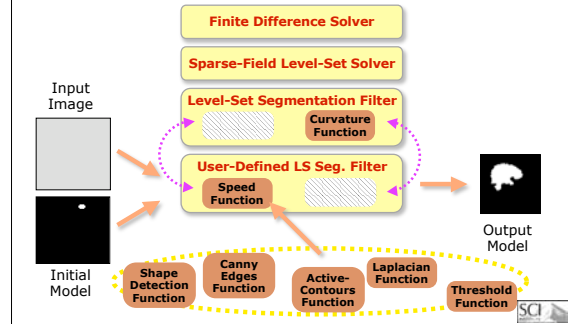


Level-Set Segmentation Framework

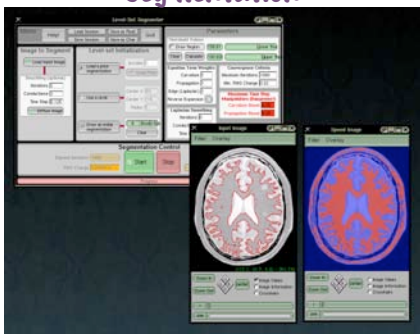
- Sparse or Narrow-band solver
- Speed function – plugs into a generic level-set solver
- Surface motion/speed based on image features or intensity
- Combine different terms in a *plug and play* manner



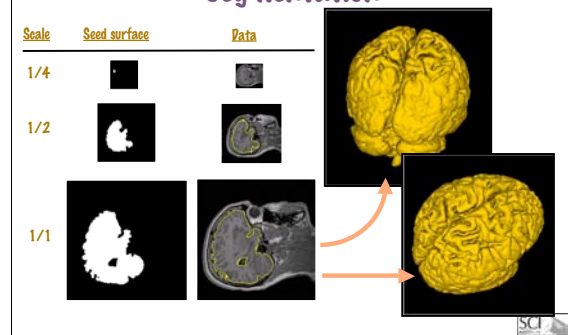
Level-Set Segmentation Framework



Example: Interactive 2D Segmentation



Example: Multiscale 3D Segmentation



Applications of PDEs

Overview

- Surface Processing
- Surface Reconstruction
- Processing Seismic Data

Surface Processing

- **Goals**
 - Rich set of tools paralleling those available for images
 - Scientific approach that does not depend on content
 - Free from arbitrary decisions (user input)
- **Why level sets?**
 - Topological changes are part of smoothing
 - Shape based-free from parameterization
 - Good for data-driven applications
 - Not as useful when parameterization is part of model



Generalizing IP to Surfaces E.g. Feature-Preserving Flows

- **Image Processing**
 - Anisotropic smoothing (Perona & Malik)
 - Markov Random Fields

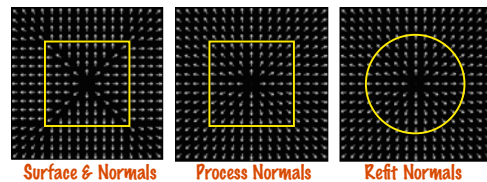


Strategy

- **Variational formulations of curvature**
 - Generalization of grad mag from IP
 - Gradient of normal map
 - Allowance for outliers
- **Decouple the normals from surface**
 - Process normals
 - Refit surface to normal map
 - System of 2nd-order equations
- Tasdizen, Whitaker, Burckard, Osher, *IEEE Vis 2002*, 10th October 2003.



Surface Filtering Strategy



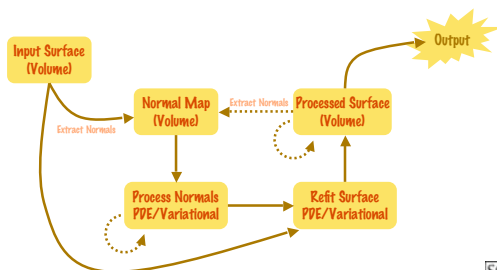
Surface & Normals

Process Normals

Refit Normals



Normal-Based Surface Filtering Strategy



"Gaussian Smoothing"



Original model

Isotropic 4th order smoothing



Anisotropic Diffusion

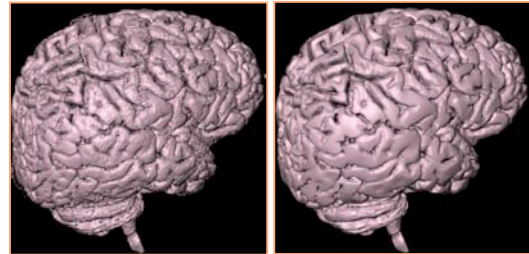


Original model

Anisotropic 4th order smoothing



Anisotropic Diffusion

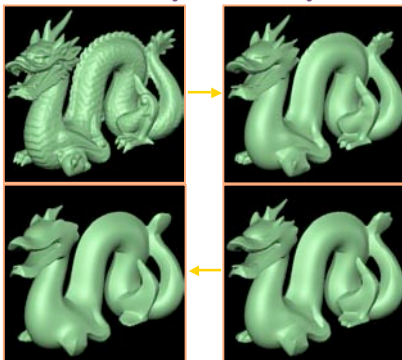


Original model

Anisotropic 4th order smoothing



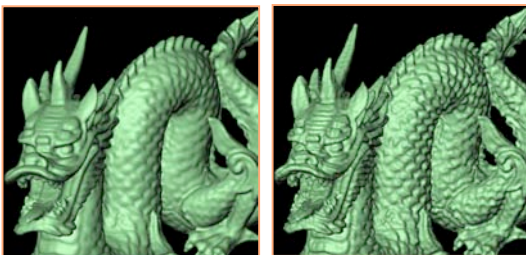
Anisotropic Scale Space



More General Image Processing Unsharp Masking/High Boost Filtering



Unsharp Masking of Surface Normals



High Boost

More High Boost



High Boost Filtering of Surfaces



Original

High Boost



Surface Reconstruction

- **Surface Measurement Technologies**
 - LADAR, structured light, vision
 - Tomography
 - Sonar/ultrasound
 - Radar
- **New computational capabilities**
- **Applications**
 - Visualization
 - Analysis



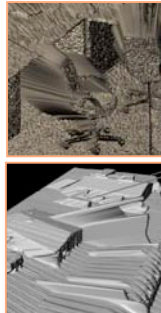
Goals

- **Foundations**
 - Statistical framework
- **Implementations**
 - Numerical
 - Computational - parallelism
- **Systems**
 - Practical-real data
 - Computation time



3D Surface Reconstruction

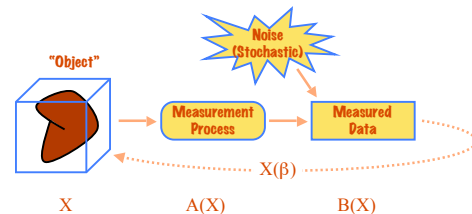
- Noise
- Occlusions
- Geometric Distortion (Calibration)
- Registration



Data Courtesy U.S. Naval Air Warfare Center, China Lake, CA



Processing Measured Data A Systems Approach

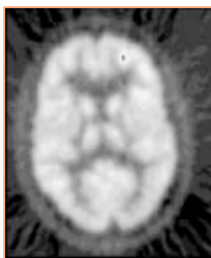


Bayesian reconstruction: maximizing the posterior - likelihood (data) + prior -> MAP

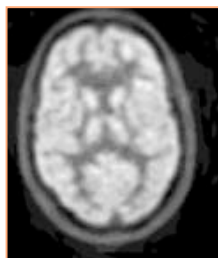


MAP PET Reconstruction

(R. Leahy, USC)



Linear Reconstruction

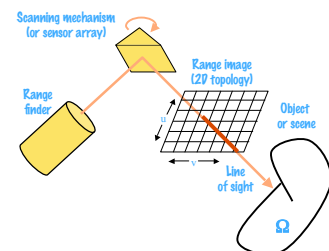


MAP Reconstruction



Maximum Likelihood Estimates of Surface Parameters

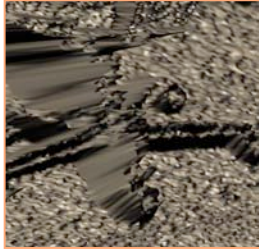
- **Scanner Model**
 - Geometry
 - Lines of sight
 - Pose
- **Sensor Model**
 - Noise
 - Gaussian w/outliers
- **Shape Optimization**
 - Parameters
 - Free-form



Progress Example: Ladar Data From Indoor Scene



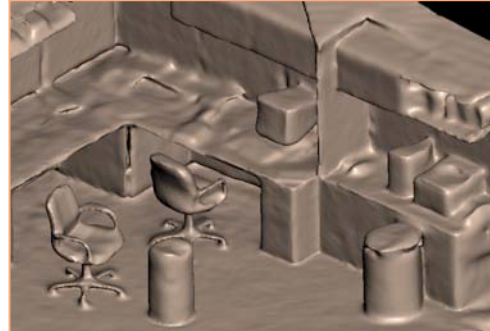
Surface Rendering of Single Scan



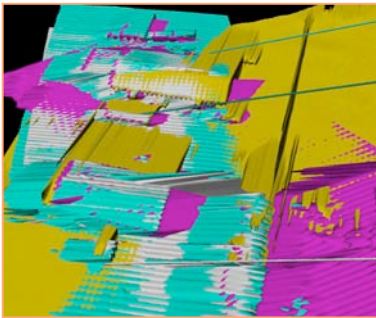
Close Up View



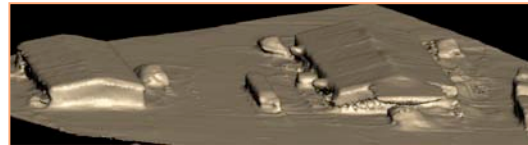
Data Fitting + 4th Order Prior



Four Ladar Views



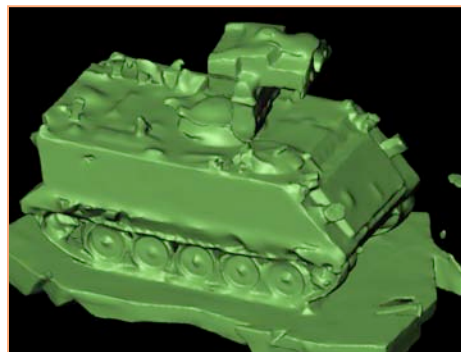
Terrain Reconstruction



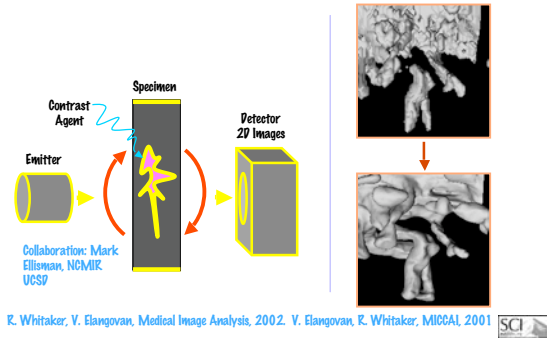
Deltasphere LADAR Scan



Volumetric LADAR Reconstruction



Tomographic Surface Reconstruction



Summary

- PDEs and geometric signal processing
- New modeling processing capabilities
- Real applications (real data) running at interactive rates



Thanks!

- **Sponsors:**
 - ONR, NSF, NIH, Exxon-Mobil
- **Students and Colleagues:**
 - Ernesto Juarez-Valdes, Stan Osher, Tolga Tasdizen, Vidya Elangovan, Paul Burchard, Suyash Awate, Won-Ki Jeong




Fundamental Properties of Level Sets

Ken Museth
Graphics Group
Linköping University, Sweden
<http://www.gg.itn.liu.se>

VIS 05
MINNEAPOLIS, MN USA

LINKÖPING UNIVERSITY



[Model from Frantic Films]

Outline:

- Properties of the Level Set function
- Numerical Implementations
- Medical Segmentation
- Extremely high resolution Level Sets

2

The Fundamental Level Set Equation

Iso-surface of co-dim 1 Iso-value

$S(t) = \{\vec{x}(t) \in \mathbb{R}^3 \mid \phi(\vec{x}(t), t) = k\}$ [Osher & Sethian 1988]

Lipschitz function $\phi : \mathbb{R}^3 \times \mathbb{R}^+ \rightarrow \mathbb{R}$

$$\frac{d\phi}{dt} = 0 = \frac{\partial \phi}{\partial t} + \frac{dx}{dt} \frac{\partial \phi}{\partial x} + \frac{dy}{dt} \frac{\partial \phi}{\partial y} + \frac{dz}{dt} \frac{\partial \phi}{\partial z} = \frac{\partial \phi}{\partial t} + \frac{d\vec{x}}{dt} \cdot \nabla \phi$$

Speed Function

$$\frac{\partial \phi}{\partial t} = - \left[\frac{d\vec{x}}{dt} \cdot \frac{\nabla \phi}{|\nabla \phi|} \right] |\nabla \phi| \equiv \Gamma(\vec{x}, \phi, \dots) |\nabla \phi|$$


$$\frac{\partial \phi}{\partial t} = \begin{cases} \vec{V} \cdot \nabla \phi \\ \Gamma |\nabla \phi| \end{cases}$$

Ken Museth, Graphics Group, Linköping University

3

Euclidian Distance Function

The level set function can in principle be any scalar function, but numerical stability requires Euclidian distance functions!



$$\phi(\vec{x}) = |\vec{x} - \vec{x}_0| = \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

$$\nabla \phi = \begin{bmatrix} \partial \phi / \partial x \\ \partial \phi / \partial y \end{bmatrix} = \frac{1}{\sqrt{(x - x_0)^2 + (y - y_0)^2}} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

$$|\nabla \phi| = \sqrt{\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2} = 1 \quad \text{Eikonal EQ}$$

$|\nabla \phi| = 1 \Rightarrow \phi$ is a distance function

$|\nabla \phi| \neq 1 \nRightarrow \phi$ is a distance function

4

Normal of a Level Set

$$\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right] \equiv \vec{e}_x \frac{\partial}{\partial x} + \vec{e}_y \frac{\partial}{\partial y} + \vec{e}_z \frac{\partial}{\partial z}$$

$$= \vec{e}_1 \frac{\partial}{\partial e_1} + \vec{e}_2 \frac{\partial}{\partial e_2} + \vec{n} \frac{\partial}{\partial n} = \vec{e}_1 (\vec{e}_1 \cdot \nabla) + \vec{e}_2 (\vec{e}_2 \cdot \nabla) + \vec{n} (\vec{n} \cdot \nabla)$$

$$\frac{\partial \phi}{\partial e_1} = \frac{\partial \phi}{\partial e_2} = 0 \Rightarrow \nabla \phi = \vec{n} (\vec{n} \cdot \nabla \phi) \Rightarrow \vec{n} \parallel \nabla \phi$$


$$\Rightarrow \vec{n} = \pm \frac{\nabla \phi}{|\nabla \phi|} \quad |\nabla \phi| = 1 \Rightarrow \vec{n} = \pm \nabla \phi$$

$$\vec{n} \cdot \nabla \phi = \pm 1$$

Ken Museth, Graphics Group, Linköping University

5

Closest-Point-Transform of a Level Sets



$\vec{x}_1 = \vec{x}_0 + h \vec{n}$

$\vec{n} = \nabla \phi(\vec{x}_0)$

i.e. $\nabla \phi$ is constant along


$$\phi(\vec{x}_1) = \phi(\vec{x}_0) + h \frac{d\phi}{dn} \Big|_{\vec{x}_0} + \frac{h^2}{2} \frac{d^2 \phi}{dn^2} \Big|_{\vec{x}_0} + \dots$$

$$= \phi(\vec{x}_0) + h (\vec{n} \cdot \nabla \phi) \Big|_{\vec{x}_0} + \frac{h^2}{2} (\vec{n} \cdot \nabla [\vec{n} \cdot \nabla \phi]) \Big|_{\vec{x}_0} + \dots$$

$$\vec{n} \cdot \nabla \phi = 1 \quad \phi(\vec{x}_1) = \phi(\vec{x}_0) + h \quad \nabla \phi(\vec{x}_1) = \nabla \phi(\vec{x}_0)$$

$$\phi(\vec{x}_0) = 0 \Rightarrow \phi(\vec{x}_1) \equiv h \quad CPT(\vec{x}_1) \equiv \vec{x}_0 = \vec{x}_1 - h \vec{n} = \vec{x}_1 - \phi(\vec{x}_1) \nabla \phi(\vec{x}_1)$$

6



OS

GG

Mean Curvature of Level Sets

$$K = \frac{1}{2}(K_1 + K_2) = \frac{1}{2}(\text{Div}_{\vec{e}}[\vec{n}] + \text{Div}_{\vec{e}}[\vec{n}])$$

$$= \frac{1}{2}(\vec{e}_1(\vec{e}_1 \cdot \nabla) \cdot \vec{n} + \vec{e}_2(\vec{e}_2 \cdot \nabla) \cdot \vec{n})$$

$$\nabla = \vec{e}_1(\vec{e}_1 \cdot \nabla) + \vec{e}_2(\vec{e}_2 \cdot \nabla) + \vec{n}(\vec{n} \cdot \nabla)$$


$$K = \frac{1}{2}(\nabla \cdot \vec{n}(\vec{n} \cdot \nabla)) \cdot \vec{n}$$

$$\vec{n}(\vec{n} \cdot \nabla) \cdot \vec{n} = 0$$

$$K = \frac{1}{2} \nabla \cdot \vec{n} = \frac{1}{2} \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$$

Ken Museth, Graphics Group, Linköping University

7



OS

GG


Other Properties of Level Sets

- Consider a closed region Ω with a boundary Γ
- Size of Ω (area in 2D and volume in 3D):
$$|\Omega| = \int_{\Omega} H(-\phi) d\vec{x} \quad H(\phi) = \begin{cases} 0 & \text{if } \phi < 0 \\ 1 & \text{if } \phi \geq 0 \end{cases} \leftarrow \text{Heaviside function}$$
- Size of Γ (arc-length in 2D and area in 3D):
$$|\Gamma| = \int_{\Omega} |\nabla H(\phi)| d\vec{x} = \int_{\Omega} \delta(\phi) |\nabla \phi| d\vec{x} \quad \delta(\phi) = \frac{d}{d\phi} [H(\phi)]$$

$\leftarrow \text{Dirac's Delta function}$

Ken Museth, Graphics Group, Linköping University

8



OS

GG

Numerical Implementation


- Smeared out 1st order approximations (band-limited)

$$H(\phi) = \begin{cases} 0 & \text{if } \phi < -\varepsilon \\ \frac{1}{2} + \frac{\phi}{2\varepsilon} + \frac{1}{2\pi} \sin\left(\frac{\pi\phi}{\varepsilon}\right) & \text{if } -\varepsilon \leq \phi \leq \varepsilon \\ 1 & \text{if } \phi > \varepsilon \end{cases}$$

$$\delta(\phi) = \begin{cases} 0 & \text{if } \phi < -\varepsilon \\ \frac{1}{2\varepsilon} + \frac{1}{2\varepsilon} \cos\left(\frac{\pi\phi}{\varepsilon}\right) & \text{if } -\varepsilon \leq \phi \leq \varepsilon \\ 0 & \text{if } \phi > \varepsilon \end{cases}$$

Ken Museth, Graphics Group, Linköping University

9



OS

GG


Minimization of Area and Volume

- Minimization of area:
$$\delta \left[\int_{\Omega} \delta(\phi) |\nabla \phi| d\vec{x} \right] = 0 \Rightarrow \frac{\partial \phi}{\partial t} = \kappa |\nabla \phi|$$
- Minimization of volume:
$$\delta \left[\int_{\Omega} H(-\phi) d\vec{x} \right] = 0 \Rightarrow \frac{\partial \phi}{\partial t} = |\nabla \phi|$$

Gradient decent of Euler-Lagrange EQ (Calculus of Variations)

Ken Museth, Graphics Group, Linköping University

10



OS


GG

Summary of Properties

- Closest distance $\phi(\vec{x})$
- Interface normal $\vec{n} = \frac{\nabla \phi}{|\nabla \phi|} = \nabla \phi$
- Closest Distance Transform $\text{CPT}(\vec{x}) = \vec{x} - \phi(\vec{x}) \nabla \phi(\vec{x})$
- Mean curvature (3D) $K = \frac{1}{2} \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$
- Volume (3D) $|\Omega| = \int_{\Omega} H(-\phi) d\vec{x} \quad H(\phi) = \begin{cases} 0 & \text{if } \phi < 0 \\ 1 & \text{if } \phi \geq 0 \end{cases}$
- Area (3D) $|\Gamma| = \int_{\Omega} \delta(\phi) |\nabla \phi| d\vec{x} \quad \delta(\phi) = \frac{d}{d\phi} [H(\phi)]$

Ken Museth, Graphics Group, Linköping University

11



OS

GG

Level Set Implementations

- **Stability**
 - Explicit integration scheme: CFL time-step condition
 - Implicit integration scheme: unconditionally stable
 - Parabolic: infinite domain-of-dep. => central difference
 - Hyperbolic: finite domain-of dep. => upwind schemes
- **Accuracy**
 - 1st order forward Euler or higher order Runge-Kutta
 - Parabolic: 2nd or higher order central difference
 - Hyperbolic: Gudonov with 1st order FD or (W)ENO
- **Efficiency**
 - Computational: Narrow band
 - Storage: adaptive grids!

Ken Museth, Graphics Group, Linköping University

12

Two Types of Level Set Equations

- Diffusion level set equations:
 - Parabolic PDE's
 - Infinite domain of dependence
 - Information has no particular direction
 - Information is propagated at infinite velocity
 - Use central-difference finite difference schemes!
- Advection level set equations:
 - Hyperbolic PDE's
 - Finite domain of dependence
 - Information has a direction
 - Information is propagated at finite velocity
 - Use upwind finite difference schemes!

13

Advection example:

- Advection by vector field: $\frac{\partial \phi}{\partial t} = \vec{V} \cdot \nabla \phi$
- Numerical implementation
 - It's hyperbolic so use upwind finite difference!

$$\frac{\partial \phi}{\partial x} = \begin{cases} \phi_x^+ = (\phi_{i+1,j,k} - \phi_{i,j,k}) / \Delta x + O(\Delta x) & \text{if } V_x < 0 \\ \phi_x^- = (\phi_{i,j,k} - \phi_{i-1,j,k}) / \Delta x + O(\Delta x) & \text{if } V_x > 0 \end{cases}$$

- Or higher order schemes like WENO [Liu *et al.*, JCP 94]

14

Advection example:

- Advection in normal direction: $\frac{\partial \phi}{\partial t} = a |\nabla \phi|$
- Numerical implementation:
 - It's hyperbolic so use upwind finite difference!

$$\left(\frac{\partial \phi}{\partial x} \right)^2 = \text{Max}(-\text{Sign}(a)\phi_x^-, \text{Sign}(a)\phi_x^+, 0)^2$$

- This is called Godunov's Method

15

Diffusion example: Geometric Heat EQ

- Mean Curvature flow in normal direction:

$$\frac{\partial \phi}{\partial t} = \kappa |\nabla \phi| = \frac{1}{2} \nabla \cdot \left[\frac{\nabla \phi}{|\nabla \phi|} \right] |\nabla \phi| \approx \nabla \cdot \nabla \phi = \Delta \phi$$
- Numerical implementation
 - It's parabolic so use central finite difference!

$$\frac{\partial \phi}{\partial x} = \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x} + O(\Delta x^2)$$

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{\Delta x^2} + O(\Delta x^2)$$

16

Re-normalization

- Direct static approach – solve Eikonal EQ:

$$|\nabla \phi| = 1$$
 - Fast Marching Method - $O(N \log N)$
 - Fast Sweeping Method - $O(N)$
- Direct dynamic approach – solve LS EQ:

$$\frac{\partial \phi}{\partial t} = \text{Sign}(\phi)(1 - |\nabla \phi|)$$
- Indirect approach – velocity extension!

17

Fast Sweeping Method

- The key observations:
 - The Eikonal EQ is a hyperbolic PDE!
 - Information is propagated from the interface and out
 - Characteristics are straight lines along the normals of the interface!
 - Partition characteristics into groups according to their direction
- Strategy:
 - Compute distances in alternating sweeps of groups of characteristics

18

Fast Sweeping Method

- Initialization
 - Define boundary condition: $\phi(\vec{x}) = 0$ for $\vec{x} \in \Gamma$
 - Freeze boundary values
 - Set remaining points to large values
- Main Loop
 - Solve Godunov's upwind-scheme by Gauss-Seidel iterations
 - When updating use: $\phi(\vec{x}) = \text{Min}(\phi_{\text{new}}(\vec{x}), \phi_{\text{old}}(\vec{x}))$
 - Alternating sweeping orderings: 2^{dim}

Ken Museth, Graphics Group, Linköping University

O(N)

19

FSM: 1D example

$$1 = \text{Max}(V_i - V_{i-1}, V_i - V_{i+1}, 0)^2$$

Ken Museth, Graphics Group, Linköping University

20

FSM: 1D example

$$1 = \text{Max}(V_i - V_{i-1}, V_i - V_{i+1}, 0)^2$$

Ken Museth, Graphics Group, Linköping University

21

FSM: 2D example

$$1 = \text{Max}(V_{ij} - V_{i-1,j}, V_{ij} - V_{i+1,j}, 0)^2 + \text{Max}(V_{ij} - V_{i,j-1}, V_{ij} - V_{i,j+1}, 0)^2$$

Ken Museth, Graphics Group, Linköping University

22

FSM: 2D example

$$1 = \text{Max}(V_{ij} - V_{i-1,j}, V_{ij} - V_{i+1,j}, 0)^2 + \text{Max}(V_{ij} - V_{i,j-1}, V_{ij} - V_{i,j+1}, 0)^2$$

Ken Museth, Graphics Group, Linköping University

23

Direct dynamic approach

Hyperbolic PDE!

$$\frac{\partial \phi(\vec{x}, t)}{\partial t} = \text{Sign}(\vec{x})(1 - |\nabla \phi(\vec{x}, t)|)$$

$$\frac{\partial \phi}{\partial t} = 0 \Rightarrow |\nabla \phi| = 1$$

$$\text{Sign}(\vec{x}) = \frac{\phi_0(\vec{x})}{\sqrt{\phi_0^2(\vec{x}) + h^2}}$$

$$\text{Sign}(\vec{x}, t) = \frac{\phi(\vec{x}, t)}{\sqrt{\phi^2(\vec{x}, t) + (\nabla \phi(\vec{x}, t)h)^2}}$$

Can move interface!

24

Trick: Velocity Extension

$\Gamma(\bar{x}) = \Gamma(\text{CPT}(\bar{x}))$
 $|\phi(\bar{x})| = |\bar{x} - \text{CPT}(\bar{x})|$
 Characteristic of $|\nabla \phi| = 1$

$\nabla \phi(\bar{x}) = \nabla \phi(\text{CPT}(\bar{x}))$
 $\text{CPT}(\bar{x}) = \bar{x} - \phi(\bar{x}) \nabla \phi(\bar{x})$
 $\frac{d\Gamma(\bar{x})}{d\bar{n}} = 0 \Leftrightarrow \nabla \phi(\bar{x}) \bullet \nabla \Gamma(\bar{x}) = 0$
 $\frac{\partial}{\partial t} \Gamma = \text{Sign}(\phi) \nabla \phi \bullet \nabla \Gamma$
 $\frac{\partial}{\partial t} |\nabla \phi|^2 = 2 \nabla \phi \bullet \nabla \frac{\partial \phi}{\partial t} = 2 \nabla \phi \bullet \Gamma \nabla |\nabla \phi| + 2 \nabla \phi \bullet \nabla [\Gamma] \nabla \phi = 0$

25

Explicit Time Integration

$\frac{\partial \phi}{\partial t} = H(\phi)$

- Forward Euler: $\phi^{n+1} = \phi^n + \Delta t H(\phi^n) + O(\Delta t^2)$
- 2. Runge-Kutta: $\tilde{\phi}^{n+1} = \phi^n + \Delta t H(\phi^n)$
 $\phi^{n+1} = \phi^n + \frac{\Delta t}{2} [H(\phi^n) + H(\tilde{\phi}^{n+1})] + O(\Delta t^3)$
- 3. Runge-Kutta: $\tilde{\phi}^{n+1} = \phi^n + \Delta t H(\phi^n)$
 $\tilde{\phi}^{n+1} = \phi^n + \frac{\Delta t}{4} [H(\phi^n) + H(\tilde{\phi}^{n+1})]$
 $\phi^{n+1} = \phi^n + \frac{\Delta t}{6} [H(\phi^n) + 4H(\tilde{\phi}^{n+1}) + H(\tilde{\phi}^{n+1})] + O(\Delta t^4)$

26

CFL Condition

- Most explicit time integration schemes are only conditionally stable!
- Stability requires that the slowest numerical wave is faster than the fastest physical wave:

$\frac{\partial \phi}{\partial t} = \vec{V} \bullet \nabla \phi \quad \frac{\text{Min}\{\Delta x, \Delta y, \Delta z\}}{\Delta t} > \text{Max}\|\vec{V}\| \quad \Delta t < \frac{\Delta x}{\text{Max}\|\vec{V}\|}$

- Level set front can only move one grid cell per iteration!

Ken Museth, Graphics Group, Linköping University

27

Parabolic Stability condition

- CFL is a necessary but not sufficient stability condition for parabolic PDE's!
- Von Neumann stability analysis of explicit integration of parabolic curvature flow:

$\frac{\partial \phi}{\partial t} = \alpha \kappa |\nabla \phi| \quad \Delta t < \left(\frac{2\alpha}{(\Delta x)^2} + \frac{2\alpha}{(\Delta y)^2} + \frac{2\alpha}{(\Delta z)^2} \right)^{-1} \approx \frac{(\Delta x)^2}{6\alpha}$

Ken Museth, Graphics Group, Linköping University

28

Implicit Time Integration

$\frac{\partial \phi}{\partial t} = H(\phi)$

- Backward Euler: $\phi^{n+1} = \phi^n + \Delta t H(\phi^{n+1}) + O(\Delta t^2)$
- Crank-Nicolson: $\phi^{n+1} = \phi^n + \Delta t \left[\frac{H(\phi^n) + H(\phi^{n+1})}{2} \right] + O(\Delta t^3)$
- Implicit integrations schemes are unconditionally stable
- However still inaccurate for large time-steps!
- Complex to implement and often slow due to the necessity to solve a linear system

Ken Museth, Graphics Group, Linköping University

29

Semi-Lagrangian Integration

$\frac{\partial \phi}{\partial t} = \vec{V} \bullet \nabla \phi$

- This is a hyperbolic advection PDE
- Can be integrated by the method of characteristics!
- Let's look at a stream-line $\vec{p}(\vec{x}_0, t)$ going through \vec{x}_0 at $t = 0$

$\frac{d}{dt} \vec{p}(\vec{x}_0, t) = -\vec{V}(\vec{p}(\vec{x}_0, t)), \quad \vec{p}(\vec{x}_0, 0) = \vec{x}_0$

- Value is constant along such stream-lines (characteristics):

$\tilde{\phi}(\vec{x}_0, t) \equiv \phi(\vec{p}(\vec{x}_0, t), t)$

$\frac{d\tilde{\phi}}{dt} = \frac{\partial \phi}{\partial t} + \frac{d\vec{p}}{dt} \bullet \nabla \phi = \frac{\partial \phi}{\partial t} - \vec{V} \bullet \nabla \phi = 0$

Ken Museth, Graphics Group, Linköping University

30

Semi-Lagrangian Integration

- We simply track each point on the grid backward in time and use simple tri-linear interpolation!

$\vec{p}(\bar{x}, 0)$ $\vec{p}(\bar{x}, -\Delta t)$
 0 $-\Delta t$

Ken Museth, Graphics Group, Linköping University

31

Semi-Lagrangian Integration

- We simply track each point on the grid backward in time and use simple tri-linear interpolation!
- Since stream-lines correspond to tracking of (Lagrangian) mass-less particle this technique is call Semi-Lagrangian!

$$\phi(\bar{x}, t + \Delta t) = \phi(\vec{p}(\bar{x}, -\Delta t), t)$$

- Explicit and yet unconditionally stable!

32

The Narrow Band Methods

Solve PDE's in narrow band embedding the interface

$\Gamma(i, j, t)$
 $\phi(i, j, t)$

positive
 zero
 negative

33

Accurate Level Set Implementation

Tubes embedding zero-crossing

positive
 zero
 negative

$T = \{(i, j, k) : |\phi(i, j, k)| < \gamma\}$
 $N = \{(i, j, k) : |\phi(i \pm 1, j \pm 1, k \pm 1)| < \gamma\}$

Peng et al., J. of Comp. Phys., 155, pp. 410-438, 1999

34

Narrow Bands With Arrays

```

dim = 1;
for each grid point (i, j)
    mask(i, j) = 0; // outside both tubes
    if (|\phi(i, j)| < \gamma)
        mask(i, j) = 2; // inside both tubes
        index1(dim) = i;
        index2(dim++) = j;
    else if (|\phi(i \pm 1, j \pm 1)| < \gamma)
        mask(i, j) = 1; // inside N tube
        index1(dim) = i;
        index2(dim++) = j;
    
```

$O(N^2)$

35

Accessing Tube Elements

➤ Run over T tube:

```

for k = 1 to dim;
    i = index1(k); j = index2(k);
    if (mask(i, j) = 2)
        \phi(i, j) = ...;
    
```


➤ Run over N tube:

```


for k = 1 to dim;
    i = index1(k); j = index2(k);
    if (mask(i, j) \ge 1)
        \phi(i, j) = ...;
    
```

$O(N)$

36



Outline of Algorithm



➤ Update in T tube:


$$\frac{\partial \phi}{\partial t} = C(\phi) \Gamma |\nabla \phi| \quad C(\phi) = \begin{cases} 1 & \text{if } |\phi| \leq \beta \\ \frac{(\gamma - |\phi|)^2 (2|\phi| + \gamma - 3\beta)}{(\gamma - \beta)^3} & \text{if } \beta < |\phi| \leq \gamma \\ 0 & \text{if } |\phi| > \gamma \end{cases}$$

➤ Re-initialize in N tube:


$$\frac{\partial d}{\partial t} = \text{Sign}(\phi) (|\nabla d| - 1) \quad \text{Sign}(\phi) \approx \frac{\phi}{\sqrt{\phi^2 + \epsilon^2}}$$

$$\phi(\bar{x}) = \begin{cases} -\gamma & \text{if } d(\bar{x}) < -\gamma \\ d(\bar{x}) & \text{if } |d(\bar{x})| \leq \gamma \\ \gamma & \text{if } d(\bar{x}) > \gamma \end{cases}$$

37



Fast Rebuild of Tubes



```

for all grid point  $(i, j) \in N_{old}$ 
  if  $|\phi(i, j)| < \gamma$ 
    add  $(i, j)$  to  $T_{new}$  and  $N_{new}$ 
  else
    for all  $(i, j)$ 's neighbors  $(p, q) \in N_{new}$ 
      if  $|\phi(p, q)| < \gamma$ 
        add  $(i, j)$  to  $N_{new}$ 
    if  $(i, j) \notin T_{old}$  but  $(i, j) \in T_{new}$ 
      for all  $(i, j)$ 's neighbors  $(p, q) \in N_{new}$ 
        add  $(p, q)$  to  $N_{new}$ 

```



$O(N)$

[Nilsson, Breen and Museth, IEEE Vis05]


38

Segmentation With Level Sets


Ken Museth
Graphics Group
Linköping University, Sweden
<http://www.gg.itn.liu.se>

MINNEAPOLIS, MN USA




Segmentation




- Separate interesting regions from background (uninteresting region) !
- Extract geometric structures
 - E.g. interface between broken bone and tissue
- Segmentation with edges
 - Based on intensity changes, i.e. derivative information!
 - Edge detection
- Segmentation without edges
 - Separate regions with smoothly varying intensities
 - Mumford-Shan segmentation [Chan-Vese]

Ken Museth, Graphics Group, Linköping University

40



Segmentation With Edges




- Interface between boundary is located at edges – where the gradient has an extrema:
- Variational formulation:

$$F(\phi, u_0) = \int_{\Omega} E(\nabla u_0(\bar{x})) \nabla H(\phi(\bar{x})) d\bar{x}$$


$$E(\nabla u_0(\bar{x})) = \frac{1}{1 + |\nabla [G_e * u_0(\bar{x})]|}$$

$$\frac{\partial \phi}{\partial t} = [E(\nabla u_0) \kappa + \nabla E(\nabla u_0) \bullet \nabla \phi] \nabla \phi$$

41




Segmentation Without Edges





- Decompose image into segments with piece-wise constant intensities
- Variational formulation:

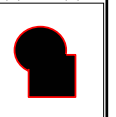
$$F_1(C) + F_2(C) = \int_{\text{inside}(C)} |u_0(x) - A|^2 + \int_{\text{outside}(C)} |u_0(x) - B|^2$$

- C is separating contour and u_0 is an image
- A is average if u_0 inside C and B outside


$F_1(C) > 0, F_2(C) = 0$


$F_1(C) = 0, F_2(C) > 0$


$F_1(C) > 0, F_2(C) > 0$



$F_1(C) = 0, F_2(C) = 0$


42



OS

Euler-Lagrange Equations



Length

Area


$$F(\phi, u_0) = \mu \int_{\Omega} |\nabla H(\phi)| d\Omega + \nu \int_{\Omega} H(\phi) d\Omega + \lambda_1 \int_{\text{inside}(\phi)} |\mu_0 - A|^2 + \lambda_2 \int_{\text{outside}(\phi)} |\mu_0 - B|^2$$

$$A = \frac{\int_{\Omega} u_0 H(\phi) d\Omega}{\int_{\Omega} H(\phi) d\Omega}$$

$$B = \frac{\int_{\Omega} u_0 (1 - H(\phi)) d\Omega}{\int_{\Omega} (1 - H(\phi)) d\Omega}$$


$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left[\mu \kappa - \nu - \lambda_1 |\mu_0 - A|^2 + \lambda_2 |\mu_0 - B|^2 \right]$$

43



OS

Framework For Semi-Automatic Segmentation



Two step procedure

Input Volume

Voxel-based Methods

- Linear filtering
- Classification
- Topology
- Morphology
- Interactive

Initialization


Level-Set Surface Models

- Curvature
- Discrete edges
- Grayscale features
- Iso-surface

Fitted Surface


[Whitaker, Breen, Museth and Soni, IEEE Vol. Graphics '01]

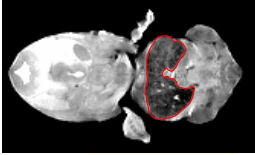

44

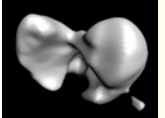
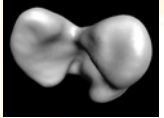
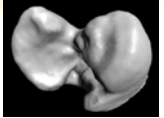


OS

Results - MRI Mouse Liver




Initialization

Canny edges

Gradient mag.


Ken Museth, Graphics Group, Linköping University

45



OS

Final Mouse Embryo Composite



Ventricles


Eyes

Liver

Skin


Ken Museth, Graphics Group, Linköping University

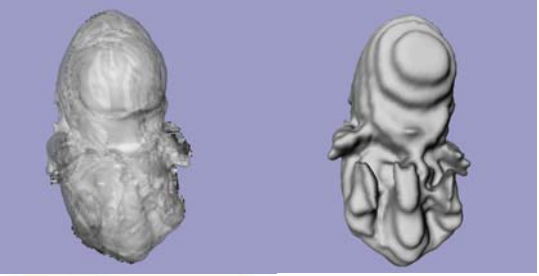
46



OS


Segmentation From Multiple Non-Uniform Volume Datasets






Ken Museth, Graphics Group, Linköping University

47

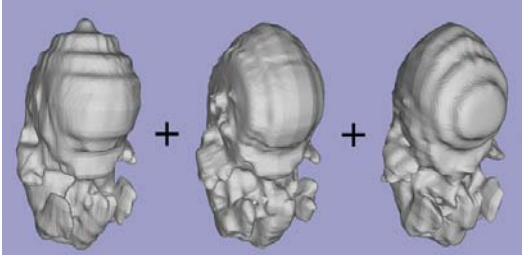


OS

Segmentation From Multiple Non-Uniform Volume Datasets



[Museth, Breen, Zhukov and Whitaker, IEEE Visualization '02]



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.
- Attraction to the discrete Canny edges.

Final surface deformation

- Attraction to 3D directional edges.
- Regularize with curvature based flow

Ken Museth, Graphics Group, Linköping University

49

Complete Segmentation Pipeline

26×128×128 256×16×128 256×128×13

50

[Model from Frantic Films]

Extremely High Resolution Level sets

- Previous work on compact narrow band methods
- DT-Grid: Dynamic Tubular Grid [J. Sci. Comp. 2005]
- H-RLE: Run-Length-Encoded Level Sets [TOG 2006]

Ken Museth, Graphics Group, Linköping University

51

Problem Statement

- Fast Narrow Band Methods [Whitaker '96 and Peng et al. '99]
 - Computational complexity linear in size of interface (area)
 - Memory usage linear in size of the embedding (volume)!
 - Efficient data structures – fast stencil access operations
- Adaptive Distance Fields [Friskien et al., SIGGRAPH '00]
 - Memory usage scales with geometric features of interface!
 - Relatively complex data structure - slow access operations
 - Level set upwind schemes are limited to uniform sampling!
- Uniformly sampled Octrees [Losasso et al., SIGGRAPH 04]
 - Computation and memory scales with size of interface
 - Relatively complex data structure – slow access operations
 - Doesn't utilize main advantage of hierarchical tree structure

Imbalance between efficiency of computation and memory usage!

52

Solutions 1: DT-Grid

[Nielsen and Museth, Journal of Scientific Computing, 2005]

- Dynamic Tubular Grid
 - Based on "Compressed Row Storage" (CRS)
 - Computational complexity linear in size of interface
 - Memory usage linear in size of interface - smaller than octrees!
 - Fast cache-coherent data structures – fast than octrees!
 - Can employ standard FD-HJ schemes in any dimensions
 - Out-Of-The-Box simulations!!!!

Ken Museth, Graphics Group, Linköping University

53

DT-Grid Data Structure

CRS format
Value-index
Narrow band value
Compressed row index
Value-index

54

DT-Grid of a 3D Sphere

X-coordinates

Y-coordinates

Z-coordinates

Values

Values

P-column, Connected Component

Ken Museth, Graphics Group, Linköping University

Random Access

Look up (x,y)

Time Complexity:
 $O(\log C_0 + \log C_x)$

Neighbor Access in Y-direction

Locator : uniquely defines a grid point in the recursive data structure.

Time Complexity: $O(1)$

Neighbor Access in X-direction

Assume we have a locator to the grid point.

Time Complexity:
 $O(\log C_{x+1})$

Sequential and Stencil Access

Time Complexity: $O(1)$

Rebuilding and Dilating the Narrow Band

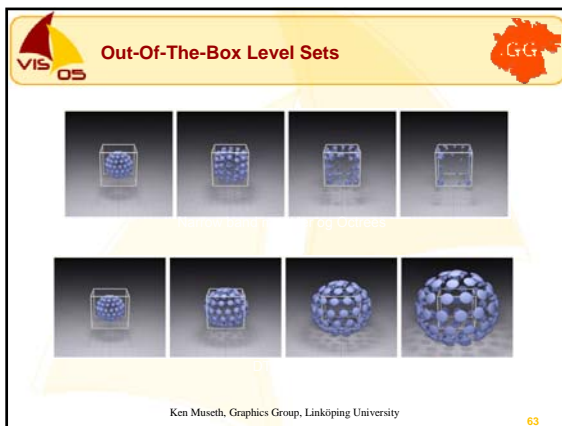
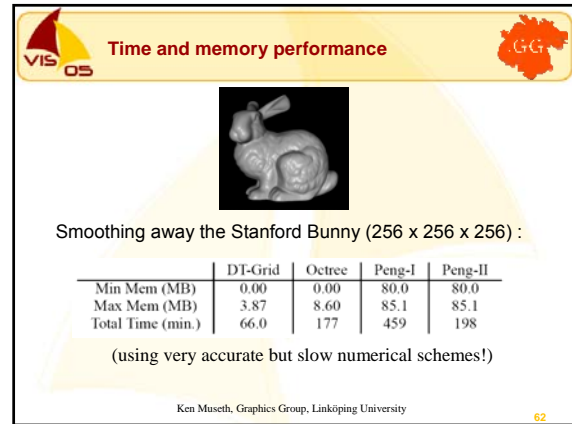
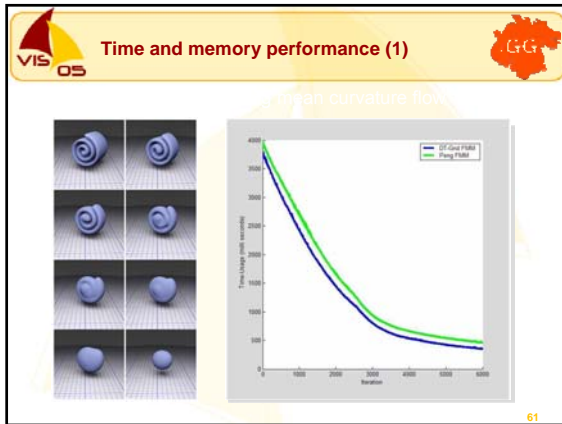
Original:

Dilated by 3x3 stencil:

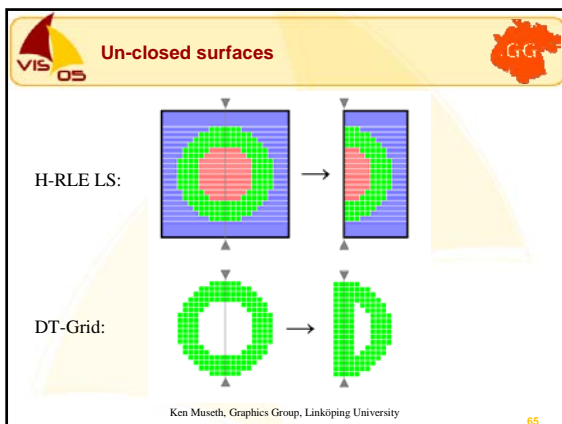
P-columns X-1,X,X+1

P-column X



Time complexity: $O(M_N)$




- Solutions 2: H-RLE Level Sets**
- Hierarchical RLE Level Sets
 - Combines benefits of DT-Grid with more flexibility for CG!
 - Based on Run-Length-Encoding in each scan-line (p-column)!
 - Allows for the representation of unclosed surfaces
 - Can be augmented with arbitrary data fields like textures etc.
 - Can encode non-symmetric narrow bands and arbitrary run-codes
 - Optimized for fluid simulations with only interior velocities
- [Houston, Nielsen, Batty, Nilsson and Museth, *ACM SIGGRAPH Sketch*, 2005]
- [Houston, Nielsen, Batty, Nilsson and Museth, *ACM TOG*, January, 2006]
- Ken Museth, Graphics Group, Linköping University
- 64



- Robust Mesh to Level Set Scan Converter**
-
- High-resolution compact RLE level set scan
 - Converted from 724 unclosed tri-mesh components
 - Effective Resolution: 1691x1223x839
 - Volume: 1.74 billion voxels
 - Total memory usage: 229.6 MB (as oppose to 7GB!)
- Ken Museth, Graphics Group, Linköping University
- 66






Gigantic Level Set with sharp features




229.6 MB as oppose to 7GB!

[TOG '06] 67

Extreme scan conversion on a laptop!

- Scan convert on a **laptop with 1GB**
 - Lucy Statue Model: 14 million vertices, 28 million triangles
 - Effective Resolution: $3000 \times 1726 \times 5144 = 26$ billion voxels
 - Total memory usage: 738 MB (as opposed to 128GB!)



Ken Museth, Graphics Group, Linköping University [TOG '06] 68






Details at 3000x1726x5144



738 MB vs 128GB


[TOG '06] 69



Mesh To Level Set Scan Conversion

- Linear time complexity and memory consumption.


Model	Bounding Box	Compact RLE	Ocree	Peng
Thai Statue	$1800 \times 3023 \times 1556$	402 MB	836 MB	41.0 GB
Lucy	$3000 \times 1726 \times 5144$	738 MB	1.60 GB	128 GB
Buddha	$1800 \times 4367 \times 1802$	816 MB	1.63 GB	68.0 GB
Asian Dragon	$5000 \times 2781 \times 3329$	750 MB	1.48 GB	222 GB



[TOG '06] 70






Shape Metamorphosis




- H-RLE level set morph at high resolution
 - Source: Human model with resolution $512 \times 797 \times 145$
 - Target: Thai statuette with resolution $1000 \times 1676 \times 865$
 - Conversion Volume: 26 billion voxels



[TOG '06] 71

Shape Metamorphosis



Ken Museth, Graphics Group, Linköping University [TOG '06] 72

Bunny torture ☺

$$V_x(x, y, z, t) = 2\sin(\pi)\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)$$

$$V_y(x, y, z, t) = -\sin(\pi)\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)$$

$$V_z(x, y, z, t) = -\sin(\pi)\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)$$

- Deformation the "Stanford Bunny" in the Enright vector field
 - Effective grid resolution: 1024x1024x1024
 - Maximum memory footprint: 96MB (vs. 5GB)
 - Direct ray tracing with 400 million random accesses pr. second


Ken Museth, Graphics Group, Linköping University



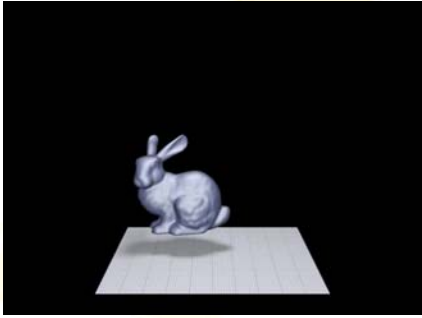

A New Level Set Benchmark ☺





[TOG '06]



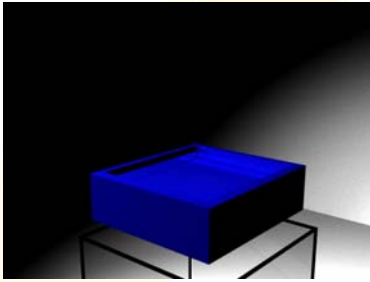

A New Level Set Benchmark ☺





[TOG '06]

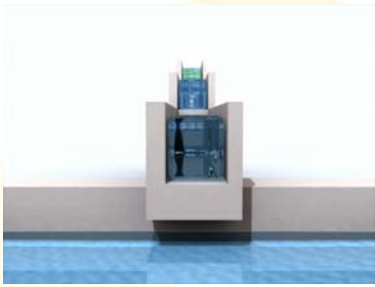
Compact Fluid Simulations



[TOG '06]

Fluid In Non-Convex Simulation "Box"



[TOG '06]




Hierarchical RLE in Hollywood

Level Set Surface Editing

The results of a non-interactive fluid simulation was converted to render time to a high-res RLE compressed level set. The basic features of the for instance represented by a vector RLE compressed level set were then interactively edited into the non-interactive fluid level set to produce the Te Mause.

1995, Oscar-nominated rendering of liquid for the film The Mause.

[Scooby Doo 2, Frantic Films]



[TOG '06]

**Acknowledgement**

- PhD students
 - Michael Nielsen
 - Ola Nilsson
 - Andreas Söderström
 - Anders Brodersen
- Co-workers
 - Ben Houston @ Frantic Films
 - Chris Batty @ Frantic Films
 - David Breen @ Drexel
 - Ross Whitaker @ Utah
- More info:
 - <http://www.gg.itn.liu.se/Teaching/Vis05>

Ken Museth, Graphics Group, Linköping University

79

**Thursday 10:30am**

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

Ola Nilsson David Breen Ken Museth

80

Introduction to GPU Computation

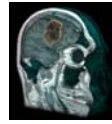


Aaron Lefohn

Institute for Data Analysis and Visualization
University of California, Davis



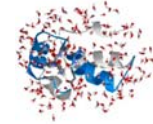
Motivation: GPU Compute Power



Level-Set Surface Deformation (Lefohn)



Cloud Simulation (Harris)



Molecular Dynamics (Buck)



Motion Estimation (Strzodka)



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

2



Motivation: GPU Compute Power

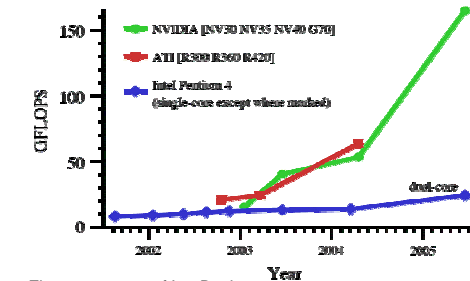


Figure courtesy of Ian Buck



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

3



(General-Purpose programming on
Graphics Processing Units)



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

4



What is GPGPU?

“Use GPU as alternate
parallel desktop compute platform”



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

5



Motivation: Why GPGPU?

- Beginning of desktop parallel computing age
 - GPUs are the first **commodity, desktop, parallel** architecture
 - NVIDIA and ATI GPUs: 24 processors
 - IBM Cell: 9 processors
 - Intel/AMD multicore: 2 processors
- Commodity
 - Inexpensive
 - Ubiquitous



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

6



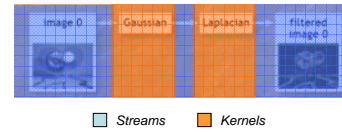
Data-Parallel Programming Basics

- What is a data-parallel program?
 - Explicitly expresses data dependencies
 - Exposes parallelism
- Stream programming is data-parallel model
 - Stream programs are dependency graphs
 - Kernels are graph nodes
 - Streams are edges flowing between kernels



Stream Program Basics

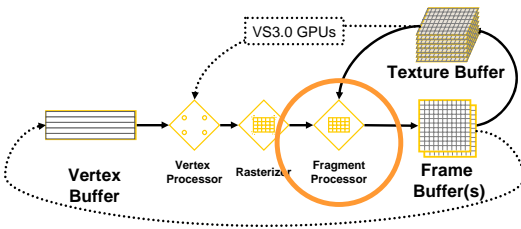
- The stream programming model
 - Split algorithm into kernels based on dependencies
 - Example: Image processing



Dally et al., "Stream Processors: Programmability with Efficiency,"
ACM Queue, March 2004, pp. 52-62
<http://cva.stanford.edu/pub/publications/spqueue.pdf>



Modern Graphics Pipeline

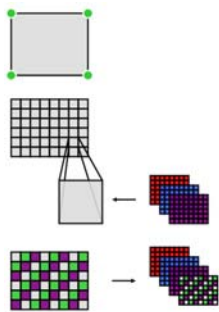


Programming a GPU for Graphics

- Application specifies geometry → rasterized
- Each fragment is shaded w/ SIMD program
- Shading can use values from texture memory
- Image can be used as texture on future passes



GPU as a Stream Processor

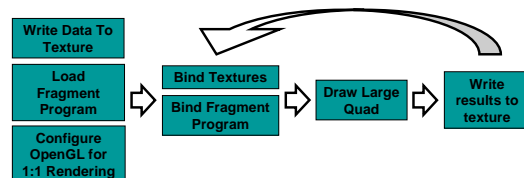


- Draw a screen-sized quad (creates stream of fragments)
- Run kernel (fragment program) over stream of fragments
- Read stream data from textures
- Write stream of results to frame buffer



GPU as a Stream Processor

- Streams → Textures
- Kernels → Fragment program
- map execution → Draw single large quad



“Hello World“ GPGPU Example

- 3 x 3 convolution
- CPU version

```
image = loadImage( WIDTH, HEIGHT );
blurImage = allocZeros( WIDTH, HEIGHT );

for (x=0; x < WIDTH; x++)
  for (y=0; y < HEIGHT; y++)
    for (i=-1; i <= 1; i++)
      for (j=-1; j <= 1; j++)
        float w = computeWeight(i,j);
        blurImage[x][y] += w * image[x+i, y+j];
```



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

13



“Hello World“ GPGPU Example

- GPU Version

- 1) Load **image** into texture

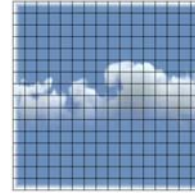


Figure courtesy of Mark Harris

- 2) Create **blurImage** texture to hold result



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

14



“Hello World“ GPGPU Example

- GPU Version

- 3) Load fragment program (kernel)
Example shown in Cg

```
float4 blurKernel( uniform samplerRECT image,
                  float2 winPos : WPOS,
                  out float4 blurImage )
{
    blurImage = float4(0,0,0,0);

    for (i=-1; i <= 1; i++) {
        for (j=-1; j <= 1; j++) {
            float2 texCoord = winPos + float2(i,j);
            float w = computeWeight(i,j);
            blurImage += w * texRECT( image, texCoord );
        }
    }
}
```



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

15



“Hello World“ GPGPU Example

- GPU Version

- 4) Configure OpenGL to draw 1:1
No projection or rescaling

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity();
gluOrtho2D(0, 1, 0, 1);
glViewport(0, 0, WIDTH, HEIGHT );
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
```

- 5) Bind **image** and **blurKernel** (texture and fragment program)
- 6) Bind **blurImage** as render target



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

16

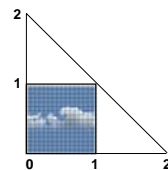


“Hello World“ GPGPU Example

- GPU Version

- 7) Execute kernel on each stream element
Draw quad of size [WIDTH x HEIGHT]

```
glBegin( GL_TRIANGLES );
glVertex2f(0, 0);
glVertex2f(2, 0);
glVertex2f(0, 2);
glEnd();
```



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

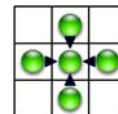
17



“Hello World“ GPGPU Example

- What happened?

- blurKernel** executed on each element of **image**
 - Rendering replaced outer two loops of CPU version
- blurKernel** performed *gather* operation at each element



Gather

- Result (**blurImage**) was written to framebuffer / texture



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

18



GPGPU Languages

● BrookGPU

● Stanford University

```
kernel void blur( float4 image[][],
                  out float blurImage<> ) {
    for (float i=-1; i <= 1; i++) {
        for (float j=-1; j <= 1; j++) {
            float2 coord = here.xy() + float2(i,j);
            float w = computeWeight(i,j);
            blurImage += w * image[coord.y][coord.x];
        }
    }
    float4 image<WIDTH * HEIGHT>;
    float4 blurImage<WIDTH * HEIGHT>;
    blur( image, blurImage );
}
```



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

19



GPGPU Languages

● Sh

● University of Waterloo

```
fsh = SH_BEGIN_PROGRAM("gpu:fragment") {
    shInputNormal3f nv; // normal (VCS)
    shInputVector3f lv; // light-vector (VCS)
    shInputVector3f vv; // view vector (VCS)
    shInputColor3f ec; // irradiance
    shInputTexCoord2f u; // texture coordinate

    shOutputColor3f fc; // fragment color

    vv = normalize(vv);
    lv = normalize(lv);
    nv = normalize(nv);
    shVector3f hv = normalize(lv + vv);
    fc = kd(u) * ec;
    fc += ks(u) * pow(pos(hv|nv), spec_exp);
} SH_END;
```

Slide courtesy of Ian Buck



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

20

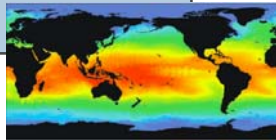


GPGPU Languages

● Scout

● Los Alamos National Labs

```
// Compute mean value
render with(shapeof(pt)) {
    // land and pt must have the same shape...
    where(land) // Don't color the continents...
        image = 0;
    else
        image = henv(240 - norm(pt) * 240, 1.0, 1.0, 1.0);
}
```



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

Slide courtesy of Patrick McCormick

21



GPGPU Data Structures

● Gltf

● University of California, Davis

```
typedef gltf::ArrayGpu<vec2i, vec4f> ArrayType;

vec2i size(WIDTH, HEIGHT);
ArrayType image( size );
ArrayType blurImage( size );

vec3i origin(0,0,0);
gpu_iterator it = blurImage.gpu_n_range(origin, size,
                                         vec2i(-1,-1), vec2i(1,1));

float4 main( NeighborIter2D it ) : COLOR {
    for (float i=-1; i <= 1; i++) {
        for (float j=-1; j <= 1; j++) {
            float w = computeWeight(i,j);
            blurImage += w * it.value( float2(i,j) );
        }
    }
}
```



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

22



For Further GPGPU Information

Vibrant research community

- Two major focuses
 - Application-specific results
 - New programming models for data-parallel computation
- <http://www.gpgpu.org/>
 - Paper, forums, source code examples
 - ACM SIGGRAPH and IEEE Visualization course notes
 - GPU Gems II
 - Eurographics 2005 STAR report



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

23



“Hello World” GPGPU Example

● Source code for GPGPU examples

- <http://www.gpgpu.org/developer/>
- <http://developer.nvidia.com>
- <http://www.ati.com/developer/>



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

24



Summary

- Many PDE solutions map very well to GPU
 - Local access patterns
 - Regular grids
 - Lots of math per memory access



Summary

- Desktop parallel computing is here to stay
 - Other architectures
 - Intel/AMD multi-core
 - IBM Cell
 - Next-gen GPUs



Interactive Level-Set Deformation On the GPU



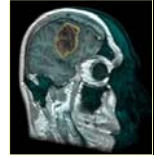
Aaron Lefohn

Institute for Data Analysis and Visualization
University of California, Davis



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



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

2



Collaborators



University of Utah
Joe Kniss
Joshua Cates
Charles Hansen
Ross Whitaker



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

3



Overview

- Why?
 - Motivation and previous work
- How?
 - Intro to GPU computation
 - Streaming level-set algorithm
 - Real-time visualization
 - Segmentation application
- Does it work?
 - Demonstration
 - User study



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

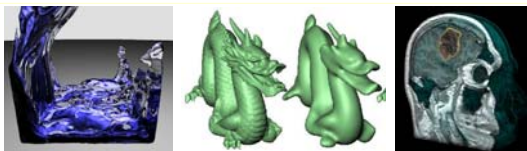
4



Deformable Surfaces

Introduction

- Applications of Level-Sets
 - Fluid simulation
 - Surface reconstruction for 3D scanning
 - Surface processing
 - Image / Volume segmentation



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

5

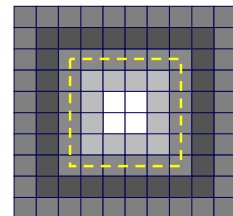


Level-Set Method

Introduction

- Implicit surface

$$S_t = \{x | \phi(x, t) = k\}$$
- Distance transform
 - ϕ 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



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

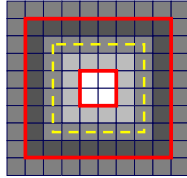
6



CPU Level-Set Acceleration

Introduction

- Narrow-Band/Sparse-Grid
 - Compute PDE *only* near the surface
 - Adalsteinson et al. 1995
 - Whitaker et al. 1998
 - Peng et al. 1999



- Time-dependent, sparse-grid solver



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

7



GPU Level-Set Acceleration

Introduction

- 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



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

8



Overview

- Why?
 - Motivation and previous work
- How?
 - Streaming narrow-band
 - Real-time visualization
 - Segmentation application
- Does it work?
 - Demonstration
 - User study



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

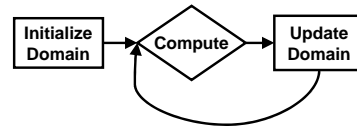
9



GPU Narrow-Band Solver

Algorithm

- 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



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

10



Algorithm Goals

Algorithm

GPU Narrow-Band Solver

- Goals
 1. Leverage GPU parallelism
 2. Perform sparse computation
 3. Minimize GPU memory usage
 4. Fast update of sparse data structures
 5. Interactive visualization



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

11



Algorithm Solutions

Algorithm

- Pack Active Voxels Into 2D Texture
 - Increase parallelism, reduce computation and memory use
- Efficient GPU-to-CPU Message Passing
 - Fast update of packed data structure
- On-The-Fly Decompression Volume Rendering
 - Interactive visualization without increasing memory use



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

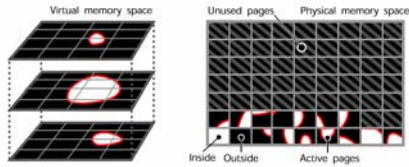
12



A Dynamic, Sparse GPU Data Structure

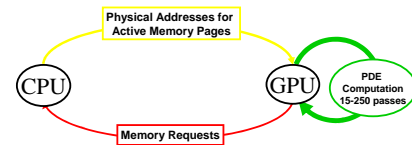
Multi-Dimensional Virtual Memory

- 3D virtual memory
- 2D physical memory
- 16 x 16 pixel pages



A Dynamic, Sparse GPU Data Structure

- GPU: Computes PDE
 - Level-set computation (2D physical memory)
 - Issues memory requests
- CPU: Manages memory
 - Memory manager
 - Page table (3D virtual memory)



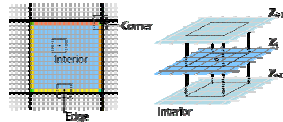
A Dynamic, Sparse GPU Data Structure

Problem

- Neighbor lookups across page boundaries
- Branching slow on GPU

Solution

- Substreams
 - Create homogeneous data streams
 - Resolve conditionals with geometry
 - Lefohn 2003, Goodnight 2003, Harris 2003
 - Optimizes cache and pre-fetch performance



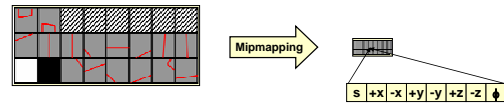
GPU-to-CPU Message Passing

Problem: Active Voxel Set is Time-Dependent

- GPU memory request mechanism
- Low bandwidth GPU-to-CPU communication

Solution

- Compress GPU memory request
- Use GPU computation to save GPU-to-CPU bandwidth



Overview

Why?

- Motivation and previous work

How?

- Streaming level-set algorithm
- Real-time visualization (with Joe Kniss)
- Segmentation application

Does it work?

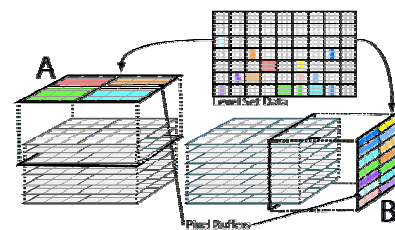
- Live demonstration
- User study



Direct Volume Rendering of Level Set

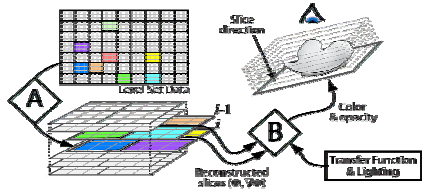
Reconstruct 2D Slice of Virtual Memory Space

- On-the-fly decompression on GPU
- Use 2D geometry and texture coordinates



Direct Volume Rendering of Level Set

- Deferred Filtering: Volume Rendering Compressed Data
 - 2D slice-based rendering: *No data duplication*
 - Tri-linear interpolation
 - Full transfer function and lighting capabilities



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

19



Overview

- Why?
 - Motivation and previous work
- How?
 - Streaming level-set algorithm
 - Real-time visualization
 - Segmentation application
- Does it work?
 - Demonstration
 - User study



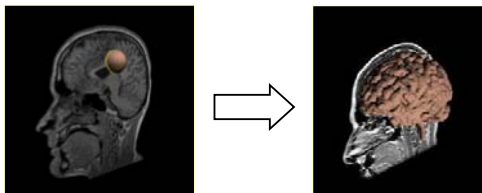
Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

20



Level-Set Segmentation Application

- Idea: Segment Surface from 3D Image
 - Begin with "seed" surface
 - Deform surface into target segmentation



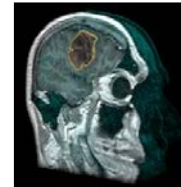
Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

21



Demo

- Segmentation of MRI volumes
 - 128³ scalar volume
- Hardware Details
 - ATI Radeon 9800 Pro
 - 1.7 GHz Intel Pentium 4
 - 1 GB of RAM

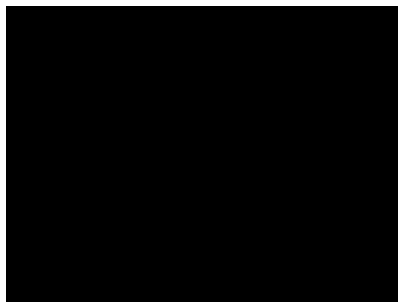


Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

22



Movie



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

23



GPU Narrow-Band Performance

- Performance
 - 10x – 15x faster than optimized CPU version (Insight Toolkit)
 - Linear dependence on size of narrow band
- Bottlenecks
 - Fragment processor (~80%)
 - Conservative time step
 - Need for global accumulation register (min, max, sum, etc.)



Level Set and PDE Methods for Visualization
Aaron Lefohn
University of California, Davis, USA

24



Overview

- Why?
 - Motivation and previous work
- How?
 - Streaming level-set algorithm
 - Real-time visualization
 - Segmentation application
- Does it work?
 - Demonstration
 - User study (with Josh Cates)



Evaluation User Study

- Goal
 - Can a user quickly find parameter settings to create an accurate, precise 3D segmentation?
- Relative to hand contouring



User Study Results

- Efficiency
 - 6 ± 3 minutes per segmentation (vs multiple hours)
 - Solver idle 90% - 95% of time
- Precision
 - Intersubject similarity significantly better
 - $94.04\% \pm 0.04\%$ vs. $82.65\% \pm 0.07\%$
- 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
- Algorithm Developments
 - Multi-dimensional virtual memory
 - GPU-to-CPU Message passing
 - Volume rendering packed data



That was two years ago...

- GPUs have quadrupled in speed
- New sparse-field CPU level-set methods
- Template library for generic GPU data structures
 - "Glift : Generic, Efficient, Random-Access GPU Data Structures."
 - ACM Transactions on Graphics (TOG)
 - Greatly simplifies implementing data structures like this
- Research on octree-based GPU PDE solver is in-progress...



Future Directions

- Other Level-Set Applications
 - Surface processing, surface reconstruction, physical simulation
- Integrate GPGPU Code Into Open Source Software
 - The Insight Toolkit (www.itk.org)?
- "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

Lefohn, Kniss, Hansen, Whitaker, “**A Streaming Narrow Band Algorithm: Interactive Computation and Visualization of Level Sets,**” IEEE Transactions on Visualization and Computer Graphics, 10 (40), Jul / Aug, pp. 422-433, 2004

Cates, Lefohn, Whitaker, “**GIST: An Interactive, GPU-Based Level-Set Segmentation Tool for 3D Medical Images,**” Medical Image Analysis, to appear 2004



PDE Methods in Flow Field Post-Processing & Anisotropic Levelset Diffusion

Tobias Preusser
CeVis/MeVis @ University of Bremen
Germany



Collaborators

- Jens Acker, Dortmund University
- Joachim Becker, Freiburg University
- David Bürkle, Freiburg University
- Udo Diewald, Bonn University
- Harald Garcke, Regensburg University
- Nadine Olischläger, Bonn University
- Martin Rumpf, Bonn University
- Alex Telea, Eindhoven University
- Stefan Turek, Dortmund University
- Ulrich Weikard, Shell AG, Hamburg
- J. van Wijk, Eindhoven University

2



Overview

- PDE methods in flow field post-processing
- Scale space concept of image processing
 - Anisotropic nonlinear diffusion for flow visualization
 - Implementation
 - Phase separation for flow clustering

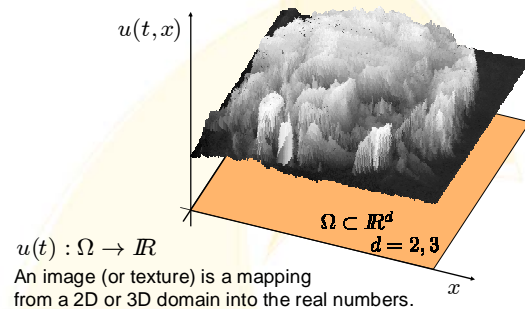
Anisotropic levelset diffusion

- Introduction
- Anisotropic levelset diffusion
- The importance of regularization
- Evaluation of curvature

3



Representation of images



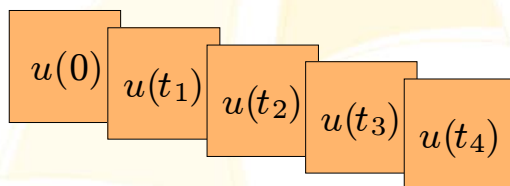
4



Scale space concept in image processing

- Filters generate a scale of images

Fine Representation (Original Version) $\xrightarrow{\{u(t)\}_{t \geq 0}}$ Coarse Representation (Simplified Version)



5



The diffusion equation

- The scale $\{u(t)\}_{t \geq 0}$ of filtered images can be generated by a **parabolic PDE**

$$\partial_t u - \operatorname{div}(A \nabla u) = 0$$

- We call t the **scale parameter** (or **scale**)
- The choice of the **diffusion tensor** A steers the behavior of the **evolution**

6

What is diffusion?

- (1) Conservation of Mass

$$\partial_t u = -\operatorname{div} J$$

The temporal change of a density u results from the flux J .
- (2) Fick's law

$$J = -A \nabla u$$

The flux J can be expressed by the gradient of the density u .
- Diffusion equation \rightarrow substituting (2) into (1)

$$\partial_t u = \operatorname{div}(A \nabla u)$$

Isotropic diffusion example
(The heat equation)

$$\partial_t u - \operatorname{div}(\nabla u) = 0$$

Better isotropic diffusion example
[Perona, Malik '87], [Catté, Lions, Morel, Coll '92]

$$\partial_t u - \operatorname{div}(A(|\nabla u|^{\sigma}) \nabla u) = 0$$

Isotropic vs. Anisotropic

- Diffusion/Flux is aligned with the image gradient vector
- Diffusion/Flux is aligned with arbitrary vector

Anisotropic diffusion example

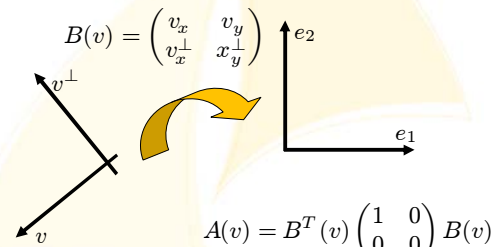
Anisotropic diffusion

The flux is determined by $J = -A \nabla u$

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad A = B^T \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} B$$

Flow aligned diffusion

➤ For flow visualization we need local transformations of the coordinate system:



$$B(v) = \begin{pmatrix} v_x & v_y \\ v_x^\perp & v_y^\perp \end{pmatrix}$$

$$A(v) = B^T(v) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} B(v)$$

The flow aligned diffusion equation

➤ Given a vector field $v : \Omega \rightarrow \mathbb{R}^d$ and a given initial image $u_0 : \Omega \rightarrow \mathbb{R}$ find a scale $\{u(t)\}_{t \geq 0}$ of representations which obey the following PDE

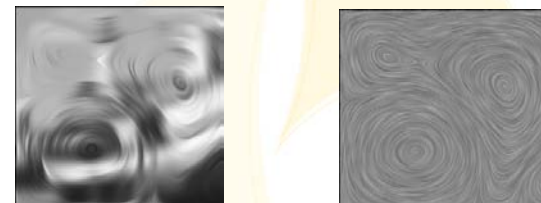
$$\partial_t u - \operatorname{div}(A(v) \nabla u) = 0$$

where the diffusion tensor is given by

$$A(v) = B^T(v) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} B(v)$$

Choice of initial data

➤ Arbitrary initial data can be used. But to avoid artifacts a white noise should be used.



Modeling multi-scale visualization

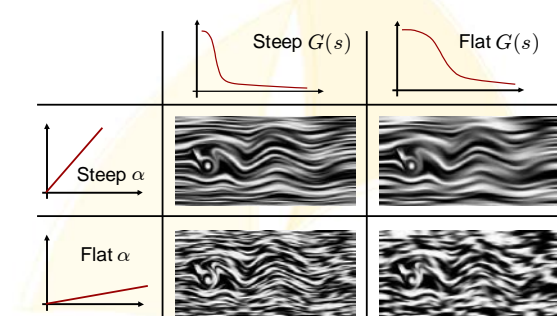
➤ Larger scale t shall give a coarser representation of the flow field.
➔ Incorporate a “clustering” of the resulting filaments.

➤ To this end use diffusion perpendicular to the flow field.

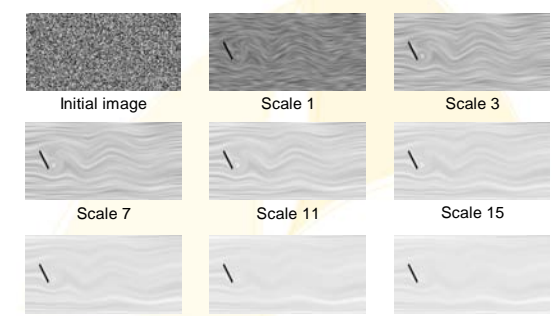
➤ The modified diffusion tensor is then

$$A(v, \nabla u) = B^T(v) \begin{pmatrix} \alpha(|v|) & 0 \\ 0 & G(|\nabla u|) \end{pmatrix} B(v)$$

Multiscale behavior

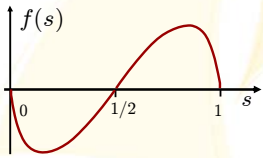


A first application example



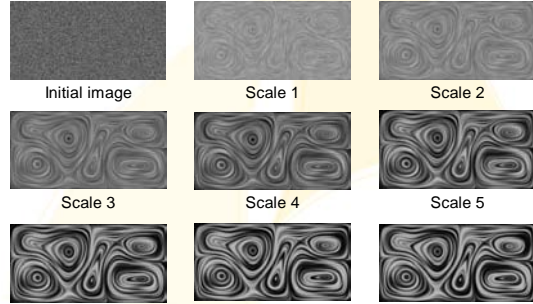
Contrast enhancement

- Contrast decreases due to the intrinsic behavior of diffusion
- Contrast enhancement can be modeled as a source term on the right hand side:

$$\partial_t u - \operatorname{div}(A(v, \nabla u) \nabla u) = f(u)$$


In the asymptotic limit $t \rightarrow \infty$ values will be either zero or one.

Example and Application



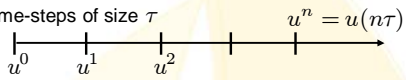
Initial image Scale 1 Scale 2
Scale 3 Scale 4 Scale 5
Scale 6 Scale 7 Scale 8

Characterization and comparison of the results

- Generates streamlines
- Incorporates a continuous scaling possibility
- Anisotropic diffusion is an asymptotic limit of LIC (Line integral convolution) (cf. [Cabral, Leedom '93], [Stalling, Hege '95] and many others ...)
- Anisotropic diffusion is a parallel version of Spot Noise (cf. [van Wijk '91], [de Leeuw, van Wijk '95] and others ...)

FEM-Discretization I

- Use time-steps of size τ



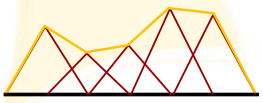
- Backward difference quotient** in scale variable:

$$\underbrace{\frac{u^n - u^{n-1}}{\tau}}_{\approx \partial_t u(n\tau)} - \operatorname{div}(A^{n-1} \nabla u^n) = f(u^{n-1})$$

- Spatial discretization: **Finite Element Methods** (9-pt stencil) can resolve anisotropic diffusion better than Finite Difference Methods (5-pt stencil)

FEM-Discretization II

- Mesh consisting of quadrilaterals or hexahedrals (node of mesh = pixel of output texture)
- Piecewise **bi- or trilinear function space** (corresponding to bi- or trilinear interpolation in 2D resp. 3D)



$$u^n(x) = \sum_{i=0}^N (\vec{u}^n)_i \phi_i(x)$$

- Computation of one step of the anisotropic diffusion means solving a system of equations for the nodal function values

$$(M + \tau L^{n-1}) \vec{u}^n = M(\vec{u}^{n-1} + \tau \vec{f}^{n-1})$$

Mass- and stiffness-matrices


- The shape functions ϕ_i determine the image space and the matrices involved into the system of equations.
- Mass matrix**

$$M_{ij} = \int_{\Omega} \phi_i(x) \phi_j(x) dx$$

- Stiffness matrix**

$$L_{ij}^n = \int_{\Omega} B^T(v) \begin{pmatrix} \alpha & 0 \\ 0 & G^n \end{pmatrix} B(v) \nabla \phi_i(x) \cdot \nabla \phi_j(x) dx$$


$$= \int_{\Omega} \begin{pmatrix} \alpha & 0 \\ 0 & G^n \end{pmatrix} B(v) \nabla \phi_i(x) \cdot B(v) \nabla \phi_j(x) dx$$



Algorithm

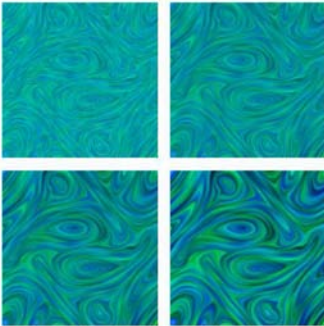
- Initialize $u_0 = \text{white noise}$
- Set $t = 0$
- Set $n = 1$
- While $t < T_{\max}$
 - Assemble M
 - Assemble L^{n-1}
 - Compute right hand side $M(\vec{u}^{n-1} + \tau \vec{f}^{n-1})$
 - Solve the system $(M + \tau L^{n-1})\vec{u}^n = M(\vec{u}^{n-1} + \tau \vec{f}^{n-1})$
 - $t = t + \tau$
 - $n = n + 1$
- End while


Performance for 256x256 texture on standard CPU is approx. 2fps.



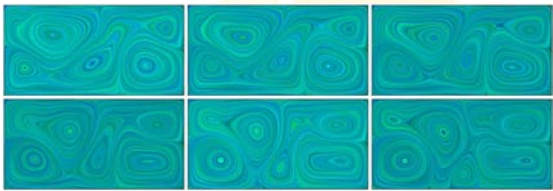
2D Application

Magneto-Hydro-Dynamics (MHD) Simulation






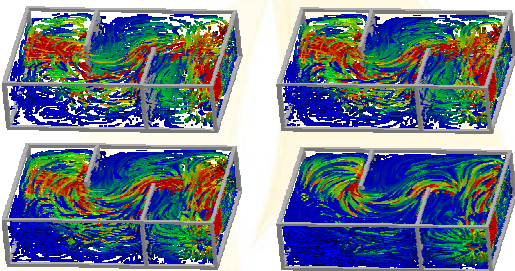
2D Application




Simulation of convective flow (Benard w. Boussinesq approximation)
Numerical data: E. Bänsch



3D Application



3D flow in a box with 2 interior walls.



Bibliography

- T. Preusser and M. Rumpf:
Anisotropic nonlinear diffusion in flow visualization
Proceedings IEEE Visualization 1999
- J. Becker, T. Preusser and M. Rumpf
PDE methods in flow simulation post processing
Computing and Visualization in Science 3(3):159-167,2000
- U. Diewald, T. Preusser and M. Rumpf
Anisotropic diffusion in vector field visualization on Euclidian domains and surfaces
IEEE Trans. Vis. Comp. Graphics 6(2):139-149, 2000

Time dependent flow visualization

- So far: static flow fields $v = v(x)$
- ➔ Flow visualization delivers a multiscale of streamline-type patterns
- Now: time-dependent flow fields $v = v(t, x)$
- ➔ Transport the resulting patterns with the flow

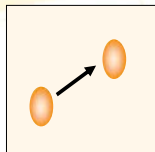
31

The material derivative

- Transport is formulated with the **material derivative**

$$\frac{D}{Dt}u(t, x) := \partial_t u(t, x) + v(t, x) \cdot \nabla u(t, x)$$

- Transport PDE moves the density along the vector field v :

$$\frac{D}{Dt}u(t, x) = 0$$


This equation is also known as **continuity equation**.

32

The Transport Diffusion PDE

- Combine the diffusion equation and the transport equation:

$$\frac{D}{Dt}u - \text{div}(A(v, \nabla u) \nabla u) = 0$$

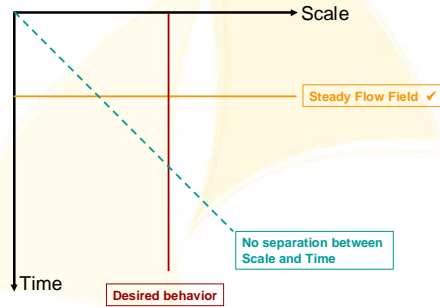
- For the diffusion equation t means the (artificial) scale
- For the transport equation t means the (real) time

➔

- There is an unwanted coupling between scale and time!
Multiscale-visualization is difficult!

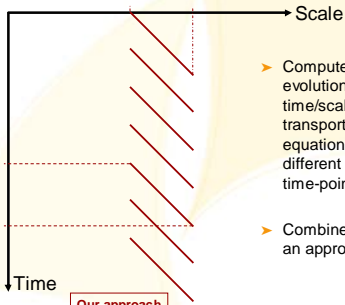
33

Scale versus Time



34


Scale versus Time



- ➔ Compute short evolution of the time/scale coupled transport diffusion equation starting at different successive time-points
- ➔ Combine all results by an appropriate blending


35

Examples and Applications

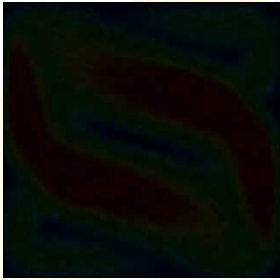


Flow of pollutant through a salt dome.
Numerical data: J. Geiser

36




Examples and Applications



Benard convective flow in a box with heating from below and cooling from above.


37



Remarks on implementation

- The proper discretization of the material derivative is not easy: The longer a density is transported the more it is blurred (**numerical diffusion**)
- Implementation for real-time visualization on CPUs is not feasible
- Realizations on graphics hardware are available for 2D and 3D flow fields
(cf. e.g. IBVF and IBVF 3D, J. v. Wijk, A. Telea)

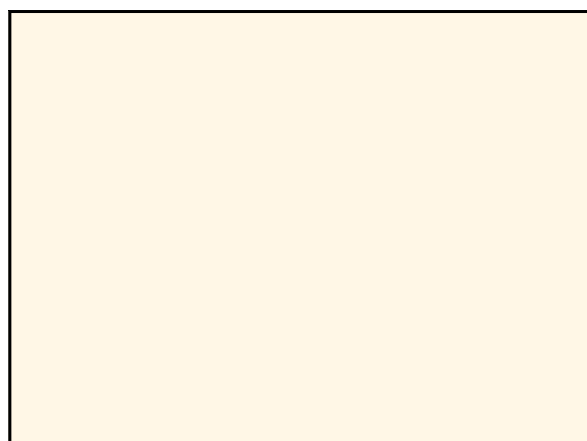
38




Bibliography

- D. Bürkle, T. Preusser, and M. Rumpf
Transport and diffusion in timedependent flow visualization
Proceedings IEEE Visualization 2001

39



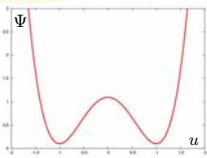


Clustering with Phase Separation (Cahn-Hilliard model)


- **Phase separation** and coarsening in (binary) metal alloys is described by minimizing the Energy

$$E(u) := \int_{\Omega} \left\{ \Psi(u) + \frac{\gamma}{2} |\nabla u|^2 \right\}$$

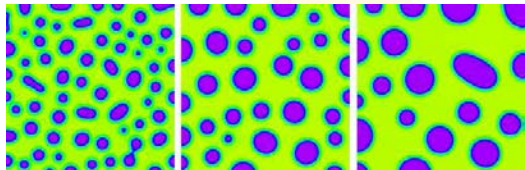
- Ψ describes the chemical energy (double-well potential)
- $|\nabla u|^2$ describes the interface energy between the phases



41



Results from phase separation

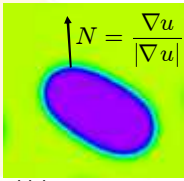


- Chemical energy is minimized → phases separate
- Interface energy is minimized → clusters grow bigger

42

Anisotropic interface energy

- The interface energy is given by $|\nabla u|^2$
- All interfaces are penalized equally
- Anisotropic energy penalizing interfaces which are not aligned with a given flow field: $A(v)\nabla u \cdot \nabla u$



$$A(v) = B(v)^T \begin{pmatrix} 1 & 0 \\ 0 & \beta \end{pmatrix} B(v) \quad \beta \geq 1$$

43

Anisotropic phase separation

- Minimize the **anisotropic energy**

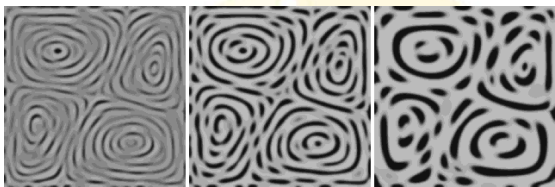
$$E(u) := \int_{\Omega} \left\{ \Psi(u) + \frac{\gamma}{2} A(v) \nabla u \cdot \nabla u \right\}$$

- The first variation of energy and a gradient descent approach leads to the PDE

$$\partial_t u - \Delta(\Psi'(u) - \gamma \operatorname{div}(A(v) \nabla u)) = 0$$

44

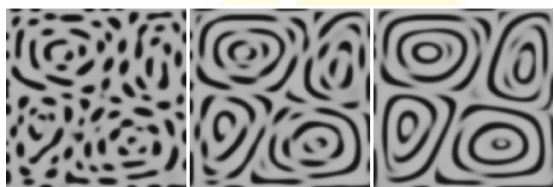
Examples and Applications



Convective flow in a 2D box. From left to right the scale increases.

45

Varying anisotropy



Convective flow in a 2D box. From left to right β increases.

46

Examples and Applications

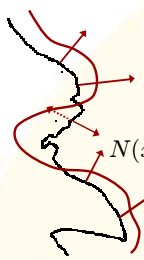
- H. Garcke, T. Preusser, M. Rumpf, A. Telea, U. Weikard and J. van Wijk
A Phase Field Model for Continuous Clustering on Vector Fields
 IEEE Trans. Vis. Comp. Graphics, 7, 230-241, 2001.

47



Levelset-Evolution

Consider motion of the level-set boundary of an image



The image u then obeys the level-set PDE

$$\partial_t u = f(x) |Du|$$

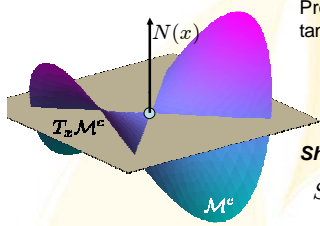
$$N(x) = \frac{\nabla u}{|\nabla u|}$$

If $f(x)$ depends only on the shape of the level-sets we call the evolution to be **geometric**.

49

Curvature of Level-Sets

Consider variation $DN(x)$ of the normal $N(x)$ on the levelset \mathcal{M}^c .



Projection onto tangent space $T_x \mathcal{M}^c$

$$P_{Du} = \text{Id} - \frac{Du \otimes Du}{|Du|^2}$$

Shape operator

$$S = DN|_{T_x \mathcal{M}^c}$$

$$= \frac{1}{|Du|} (P_{Du} D^2 u P_{Du})$$

50

The shape operator

Analysis of Eigenvalues and Eigenvectors of Shape operator leads to curvature notions

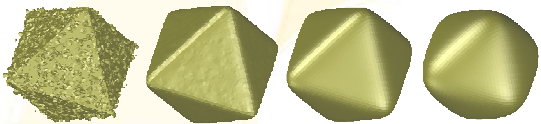
Mean curvature	$H = \text{trace } S$
Gaussian curvature	$K = \det S = k^1 k^2$
Principal curvatures	k^1, k^2

51

The mean curvature motion (MCM)

Speed of motion is equal to the mean curvature of levelsets

$$\partial_t u = f(x) |Du|$$

$$f(x) = \text{trace } S = \text{div} \left(\frac{\nabla u}{|\nabla u|} \right)$$


52

Anisotropic level-set equation

Instead of the mean curvature use the variation of an anisotropic normal to define the evolution speed

$$f(x) = \text{div} \left(A(S^\sigma) \frac{\nabla u}{|\nabla u|} \right)$$

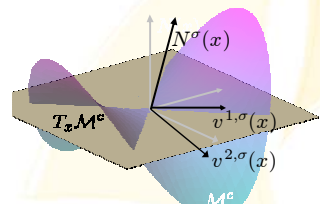

Base the definition of the anisotropic normal on a **regularized shape operator**

$$N_A := A(S^\sigma) \frac{\nabla u}{|\nabla u|}$$

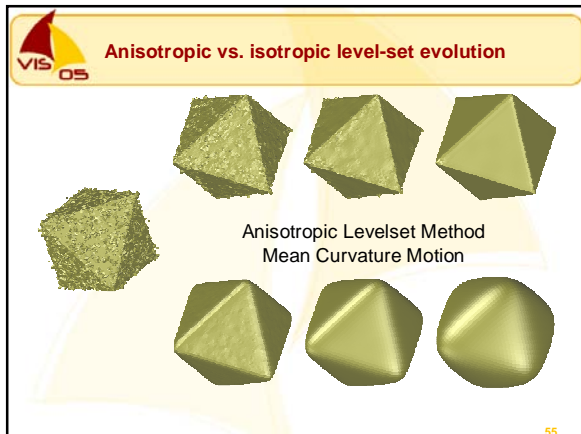
53

Curvature dependent diffusion tensor

$$f(x) = \text{div} \left(A(S^\sigma) \frac{\nabla u}{|\nabla u|} \right)$$

$$A(S^\sigma) = B_\sigma^T \begin{pmatrix} G(k^{1,\sigma}) & & \\ & G(k^{2,\sigma}) & \\ & & 0 \end{pmatrix} B_\sigma$$



54



The importance of regularization

- Regularization acts as an estimate of the true (non-noisy) shape
- One can show that

$$f(x) \sim (S^\sigma - S) + (N^\sigma - N)$$
- Levelsets are invariant under the evolution if the regularized shapes coincide with the non-regularized shapes

56

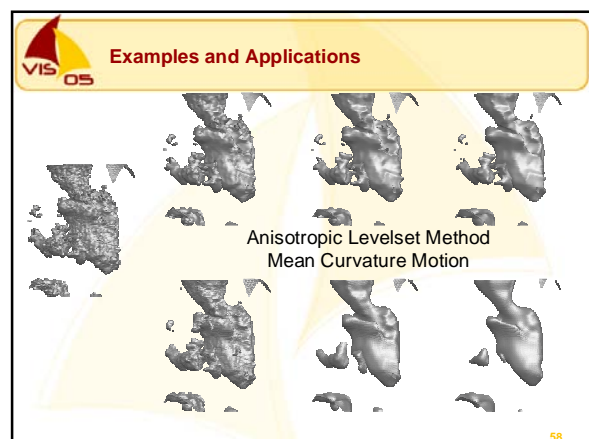
Evaluation of curvature

- Evaluation of curvature (= computation of shape operator) plays an important role

Convolution with compact smoothing kernel

Local least squares projection onto polynomials

57



Bibliography

- T. Preusser and M. Rumpf:
A level set method for anisotropic geometric diffusion in 3D image processing
SIAM J. Appl. Math., 62(5):1772-1793, 2002
- K. Mikula, T. Preusser and M. Rumpf:
Morphological Image Sequence Processing
Computing and Visualization in Science, 6(4):197-209, 2004
- T. Preusser:
Existence of viscosity solutions of nonlinear anisotropic level-set methods
submitted to SIAM J. Math. Anal.

59

Anisotropic Diffusion in Vector Field Visualization on Euclidean Domains and Surfaces

Udo Diewald, Tobias Preußner, and Martin Rumpf

Abstract—Vector field visualization is an important topic in scientific visualization. Its aim is to graphically represent field data on two and three-dimensional domains and on surfaces in an intuitively understandable way. Here, a new approach based on anisotropic nonlinear diffusion is introduced. It enables an easy perception of vector field data and serves as an appropriate scale space method for the visualization of complicated flow pattern. The approach is closely related to nonlinear diffusion methods in image analysis where images are smoothed while still retaining and enhancing edges. Here, an initial noisy image intensity is smoothed along integral lines, whereas the image is sharpened in the orthogonal direction. The method is based on a continuous model and requires the solution of a parabolic PDE problem. It is discretized only in the final implementational step. Therefore, many important qualitative aspects can already be discussed on a continuous level. Applications are shown for flow fields in 2D and 3D, as well as for principal directions of curvature on general triangulated surfaces. Furthermore, the provisions for flow segmentation are outlined.

Index Terms—Flow visualization, multiscale, nonlinear diffusion, segmentation.

1 INTRODUCTION

THE visualization of field data, especially of velocity fields from CFD computations, is one of the fundamental tasks in scientific visualization. A variety of different approaches has been presented. The simplest method of drawing vector plots at nodes of some overlaid regular grid in general produces visual clutter because of the typically different local scaling of the field in the spatial domain, which leads to disturbing multiple overlaps in certain regions, whereas, in other areas, small structures such as eddies cannot be resolved adequately. This gets even worse if tangential fields on highly curved surfaces are considered.

The central goal is to come up with intuitively better perceptible methods which give an overall, as well as a detailed, view on the flow patterns. Single particle lines only partially enlighten features of a complex flow field. Thus, we want to define a texture which represents the field globally on a 2D or 3D domain and on surfaces, respectively. Here, we confine ourselves to stationary fields. In the Euclidean case, we suppose $v: \Omega \rightarrow \mathbb{R}^n$ for some domain $\Omega \subset \mathbb{R}^n$, whereas, in the case of a manifold \mathcal{M} embedded in \mathbb{R}^3 , we consider a tangential vector field v . We ask for a method generating stretched streamline type patterns which are aligned to the vector field $v(x)$. Furthermore, the possibility of successively coarsening this pattern is obviously a desirable property. Methods which are based on such a scale of spaces and enhance certain structures of images are well-known in image processing analysis.

Actually, nonlinear diffusion allows the smoothing of gray or color images while retaining and enhancing edges [18]. Now, we set up a diffusion problem, with strong smoothing along integral lines and edge enhancement in the orthogonal directions. Applying this to some initial random noise image intensity, we generate a scale of successively coarser patterns which represent the vector field. Finite elements in space and a semi-implicit time stepping are applied to solve this diffusion problem numerically. Furthermore, a suitable modification of the approach allows the identification of topological regions.

Before we explain in detail the method, let us discuss related work on vector field visualization and image processing. Later on we will identify some of the well-known methods as equivalent to special cases or asymptotic limits of the presented new method, respectively.

2 RELATED WORK

The spot noise method proposed by van Wijk [25] introduces spot-like texture splats which are aligned by deformation to the velocity field in 2D or on surfaces in 3D. These splats are plotted in the fluid domain, showing strong alignment patterns in the flow direction. The original first order approximation to the flow was improved by de Leeuw and van Wijk in [6] by using higher order polynomial deformations of the spots in areas of significant vorticity. In an animated sequence, these spots can be moved along streamlines of the flow. Furthermore, in 3D, van Wijk [26] applies the integration to clouds of oriented particles and animates them by drawing similar moving transparent and illuminated splats.

The Line Integral Convolution (LIC) approach of Cabral and Leedom [4] integrates the fundamental ODE describing streamlines forward and backward in time at every

• The authors are with the Institute for Applied Mathematics, University of Bonn, Wegelstraße 6, 53115 Bonn, Germany.
E-mail: {diewald, tpreuss, rumpf}@iam.uni-bonn.de.

Manuscript received 15 Mar. 2000; accepted 3 Apr. 2000.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number 111480.

pixelized point in the domain, convolves a white noise along these particle paths with some Gaussian type filter kernel, and takes the resulting value as an intensity value for the corresponding pixel. According to the strong correlation of this intensity along the streamlines and the lack of any correlation in the orthogonal direction, the resulting texturing of the domain shows dense streamline filaments of varying intensity. Stalling and Hege [21] increased the performance of this method, especially by reusing portions of the convolution integral already computed on points along the streamline. Forssell [10] proposed a similar method on surfaces and Max et al. [17] discussed flow visualization by texturing on contour surfaces. Max and Becker [16] presented a method for visualizing 2D and 3D flows by animating textures.

Shen and Kao [20] applied an LIC type method to unsteady flow fields. Recently, a method [2] has been presented which generates streakline type patterns by numerical calculation of the transport of inlet coordinates and inlet position. Interrante and Grosch [12] generalized line integral convolution to 3D in terms of volume rendering of line filaments.

In [24], Turk and Banks discuss an approach which selects a certain number of streamlines. They are automatically equally distributed all over the computational domain to characterize, in a sketch-type representation, the significant aspects of the flow. An energy minimizing process is used to generate the actual distribution of streamlines.

Especially for 3D velocity fields, particle tracing is a very popular tool. But, a few particle integrations released by the user can hardly scope with the complexity of 3D vector fields. Stalling et al. [22] use pseudorandomly distributed, illuminated, and transparent streamlines to give a denser and more receptive representation, which shows the overall structure and enhances important details.

Van Wijk [27] proposed the implicit stream surface method. For a stationary flow field, the transport equations $v \cdot \nabla \phi = 0$ are solved for given v and certain inflow and outflow boundary conditions in a precomputing step. Then, isosurfaces of the resulting function ϕ are streamsurfaces and can be efficiently extracted with interactive frame rates, even for larger data sets.

Most of the methods presented so far have in common, that the generation of a coarser scale requires a recomputation. For instance, if we ask for a finer or coarser scale of the line integral convolution pattern, the computation has to be restarted with a coarser initial image intensity. In the case of spot noise, larger spots have to be selected and their stretching along the field has to be increased. The approach to be presented here will incorporate a successive coarsening as time proceeds in the underlying diffusion problem.

As already mentioned in the introduction, our method of anisotropic nonlinear diffusion to visualize vector fields is derived from well-known image processing methodology. Discrete diffusion type methods have been known for a long time. Perona and Malik [18] introduced a continuous diffusion model which allows the denoising of images together with the enhancing of edges. Alvarez et al. [1] established a rigorous axiomatic theory of diffusive scale

space methods. Kawohl and Kutev [14] investigate a qualitative analysis of the Perona and Malik model. The recovering of lower dimensional structures in images is analyzed by Weickert [28], who introduced an anisotropic nonlinear diffusion method, where the diffusion matrix depends on the so-called structure tensor of the image. A finite element discretization and its convergence properties have been studied by Kacur and Mikula [13].

Concerning the application of diffusion type methods on surfaces, a general introduction to differential calculus on manifolds can be found for instance in the book by do Carmo [7]. Dziuk [8] presented an algorithm for the solution of partial differential equations on surfaces and, in [9], he discussed a numerical method for geometric diffusion applied to the surface itself which coincides with the mean curvature motion.

3 THE NONLINEAR DIFFUSION PROBLEM

Let us now derive our method based on a suitable PDE problem. At first, we confine ourselves to the case of planar domains in 2D and 3D. Here, nonlinear anisotropic diffusion applied to some initial random noisy image will enable an intuitive and scalable visualization of complicated vector fields. Therefore, we pick up the idea of line integral convolution, where a strong correlation in the image intensity along integral lines is achieved by convolution of an initial white noise along these lines. As proposed already by Cabral and Leedom [4], a suitable choice for the convolution kernel is a Gaussian kernel. On the other hand, an appropriately scaled Gaussian kernel is known to be the fundamental solution of the heat equation. Thus, line integral convolution is nothing else than solving the heat equation in 1D on an integral line parameterized with respect to arc length. On pixels which are located on different integral lines, the resulting image intensities are not correlated. Hence, the thickness of the resulting image patterns in line integral convolution is of the size of the random initial patterns, in general, a single pixel. Increasing this size, as has been proposed by Kiu and Banks [15], leads to broader stripes and, unfortunately, less sharp transitions across streamline patterns. As described so far, line integral convolution is a discrete pixel-based method. If we ask for a well-posed continuous diffusion problem with similar properties, we are led to some anisotropic diffusion, now controlled by a suitable diffusion matrix.

To begin with, let us at first introduce a general nonlinear diffusion method from image processing and then discuss the selection of the appropriate diffusion tensor and the righthand side. Here, we consider first the case of an image in Euclidean space either in 2D or 3D. In Section 6, we then generalize this with respect to textures on surfaces. We consider a function $\rho : \mathbb{R}_0^+ \times \Omega \rightarrow \mathbb{R}$ which solves the parabolic problem

$$\begin{aligned} \frac{\partial}{\partial t} \rho - \operatorname{div}(A(\nabla \rho_\epsilon) \nabla \rho) &= f(\rho) & \text{in } \mathbb{R}^+ \times \Omega, \\ \rho(0, \cdot) &= \rho_0 & \text{on } \Omega, \\ \frac{\partial}{\partial \nu} \rho &= 0 & \text{on } \mathbb{R}^+ \times \partial \Omega. \end{aligned}$$

for given initial density $\rho_0 : \Omega \rightarrow [0, 1]$. Here, $\rho_\epsilon = \chi_\epsilon * \rho$ is a mollification of the current density, which will later on turn

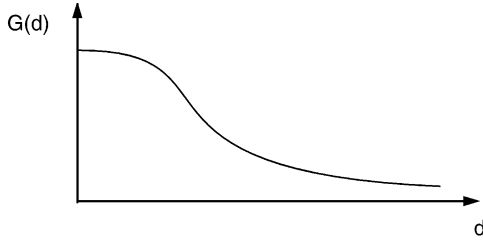


Fig. 1. The shape of $G(\cdot)$ which, applied to the gradient of the mollified image intensity, serves as a diffusion coefficient in image processing.

out to be necessary for the well-posedness of the above parabolic, boundary, and initial value problem. In our setting, we interpret the density as an image intensity, a scalar grayscale or—with a slight extension to the vector valued case—as a vector valued color. Thus, the solution $\rho(\cdot)$ can be regarded as a family of images $\{\rho(t)\}_{t \in \mathbb{R}_0^+}$, where the time t serves as a scaling parameter. Let us remark that, by the trivial choice $A = 1$ and $f(\rho) = 0$, we obtain the standard linear heat equation with its isotropic smoothing effect. In image processing, ρ_0 is a given noisy initial image. The diffusion is supposed to be controlled by the gradient of the image intensity. Large gradients mark edges in the image which should be enhanced, whereas small gradients indicate areas of approximately equal intensity. Here, denoising, i.e., intensity diffusion, is considered. For that purpose we prescribe a diffusion coefficient

$$A = G(\|\nabla \rho_\epsilon\|),$$

where $G: \mathbb{R}_0^+ \rightarrow \mathbb{R}^+$ is a monotone decreasing function with $\lim_{d \rightarrow \infty} G(d) = 0$ and $G(0) = \beta$, where $\beta \in \mathbb{R}^+$ is constant (cf. Fig. 1), e. g. $G(d) = \frac{\beta}{1+\|d\|^2}$. If we would replace the mollified gradient $\nabla \rho_\epsilon$ as argument of G by the true gradient $\nabla \rho$, which leads to the original Perona Malik model, we would, in general, obtain a backward parabolic problem in areas of high gradients which is no longer well-posed [14]. The invoked mollification avoids this shortcoming and comes along with a desirable presmoothing effect. Nevertheless, the enhancing of steep gradients and, thereby, edges in the image, known from backward diffusion, is retained if we adjust the mollification carefully. A suitable choice [13] for this mollification is a convolution with the heat equation kernel, i.e., we define $\rho_\epsilon = \tilde{\rho}(t = \epsilon^2/2)$ where $\tilde{\rho}$ is the solution of the heat equation with initial data ρ . Then, ϵ is the width of the corresponding Gaussian filter. Fig. 2 gives an example of such an image smoothing and edge enhancement by nonlinear diffusion. The function $f(\cdot)$ may serve as a penalty which forces the scale of images to stay close to the initial image, e.g., choosing $f(\rho) = \gamma(\rho_0 - \rho)$, where γ is a positive constant.

Now, we incorporate anisotropic diffusion. For a given vector field $v: \Omega \rightarrow \mathbb{R}^n$, we consider linear diffusion in the direction of the vector field and a Perona Malik type diffusion orthogonal to the field. Let us suppose that v is continuous and $v \neq 0$ on Ω . Then, there exists a family of continuous orthogonal mappings $B(v): \Omega \rightarrow SO(n)$ such that $B(v)v = \|v\|e_0$, where $\{e_i\}_{i=0,\dots,n-1}$ is the standard base



Fig. 2. The noisy image on the left is successively smoothed by nonlinear diffusion. On the right the resulting smoothed image with enhanced edges is shown.

in \mathbb{R}^n (cf. Fig. 3). We consider a diffusion matrix $A = A(v, \nabla \rho_\epsilon)$ and define

$$A(v, d) = B(v)^T \begin{pmatrix} \alpha(\|v\|) & \\ & G(\|d\|) \text{Id}_{n-1} \end{pmatrix} B(v),$$

where $\alpha: \mathbb{R}^+ \rightarrow \mathbb{R}^+$ controls the linear diffusion in the vector field direction, i.e., along streamlines, and the edge-enhancing diffusion coefficient $G(\cdot)$ introduced above acts in the orthogonal directions. Here, Id_{n-1} is the identity matrix in dimension $n-1$. We may either choose a linear function α or, in the case of a velocity field which spatially varies over several orders of magnitude, we select a monotone function α (cf. Fig. 4) with

$$\begin{aligned} \alpha(0) &> 0 \text{ and} \\ \lim_{s \rightarrow \infty} \alpha(s) &= \alpha_{\max}. \end{aligned}$$

In general, it does not make sense to consider a certain initial image. As initial data ρ_0 , we thus choose some random noise of an appropriate frequency range. This can, for instance, be generated by running a linear isotropic diffusion simulation on a discrete white noise for a short time. Hence, patterns will grow upstream and downstream, whereas the edges tangential to these patterns are successively enhanced. Still, there is some diffusion perpendicular to the field which supplies us for evolving time with a scale of progressively coarser representation of the flow field. If we run the evolution for vanishing righthand side f , the image contrast will, unfortunately, decrease due to the diffusion along streamlines. The asymptotic limit would turn out to be an averaged gray value. Therefore, we strengthen the image contrast during the evolution, selecting an appropriate function $f: [0, 1] \rightarrow \mathbb{R}$ (cf. Fig. 4) with

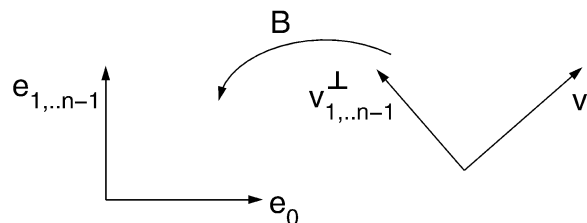


Fig. 3. The coordinate transformation $B(v)$.

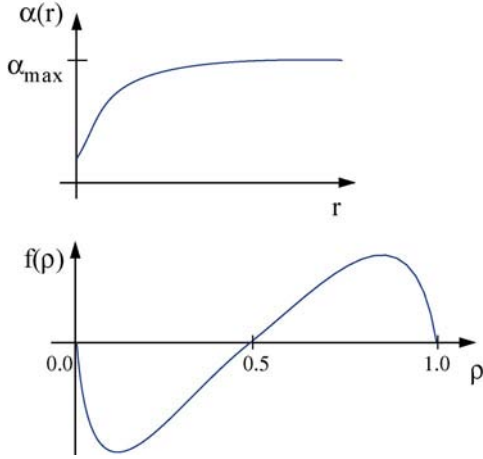


Fig. 4. The graphs of the velocity dependent linear diffusion $\alpha(\cdot)$, respectively, the scalar contrast enhancing right hand side $f(\cdot)$.

$$\begin{aligned} f(0) &= f(1) = 0 \\ f &> 0 \text{ on } (0.5, 1) \text{ and } f < 0 \text{ on } (0, 0.5). \end{aligned}$$

If we—at first glance—neglect the diffusive term in the equation, one realizes that perturbations below the average value 0.5 are pushed toward the zero value and, accordingly, values above 0.5 are pushed toward 1. Well-known maximum principles ensure that the interval of gray values $[0, 1]$ is not enlarged running the nonlinear diffusion. Here, the first property of f is of great importance. Finally, we end up with the method of nonlinear anisotropic diffusion to visualize complex vector fields. We thereby solve the nonlinear parabolic problem

$$\frac{\partial}{\partial t} \rho - \operatorname{div}(A(v, \nabla \rho) \nabla \rho) = f(\rho),$$

starting from some random initial image ρ_0 , and obtain a scale of images representing the vector field in an intuitive way (cf. Fig. 5).

The corresponding variational formulation is obviously given by

$$(\partial_t \rho, \theta) + (A(v, \nabla \rho) \nabla \rho, \nabla \theta) = (f, \theta),$$

for all $\theta \in C^\infty(\Omega)$, where (\cdot, \cdot) denotes the L^2 product on the domain Ω . Our later finite element implementation will be based on this formulation by restriction to finite dimensional function spaces.

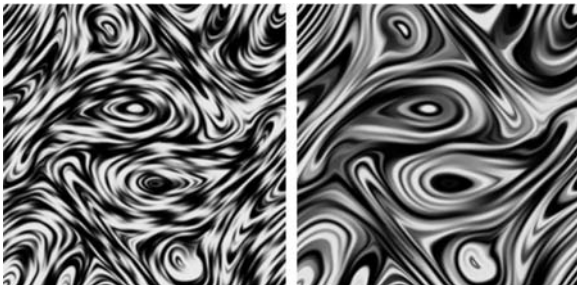


Fig. 5. A vector field from a 2D magneto-hydrodynamics simulation (MHD) is visualized by nonlinear diffusion. A discrete white noise is considered as initial data. We run the evolution on the left for a small and, on the right, for a large constant diffusion coefficient α .

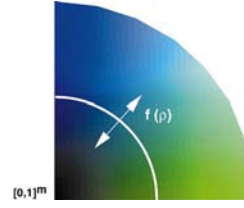


Fig. 6. A sketch of the vector valued contrast enhancing function f which leads to asymptotic states $\rho(\infty, \cdot) \in \{0\} \cup (S^{m-1} \cap [0, 1]^m)$. Here, the components of the density are interpreted as blue, respectively green, color values. The arrows indicate the direction of contrast enhancement.

4 COUPLED SYSTEM OF DIFFUSION EQUATIONS

If we ask for pointwise asymptotic limits of the evolution, we expect an almost everywhere convergence to $\rho(\infty, \cdot) \in \{0, 1\}$ due to the choice of the contrast enhancing function $f(\cdot)$. Analytically, 0.5 is a third, but unstable, fix point of the dynamics. Thus, numerically, it will not turn out to be locally dominant. The space of asymptotic limits significantly influences the richness of the developing vector field aligned structures. We may ask how to further enrich the pattern which is settled by anisotropic diffusion. This turns out to be possible by increasing the set of asymptotic states. We no longer restrict ourselves to a scalar density ρ , but consider a vector valued $\rho : \Omega \rightarrow [0, 1]^m$ for some $m \geq 1$, and a corresponding system of parabolic equations. The coupling is given by the nonlinear diffusion coefficient $G(\cdot)$ which now depends on the norm $\|\nabla \rho\|$ of the Jacobian of the vector valued density $\nabla \rho$ and the righthand side $f(\cdot)$. We define

$$f(\rho) = h(\|\rho\|)\rho$$

with $h(s) = \tilde{f}(s)/s$ for $s \neq 0$, where \tilde{f} is the old righthand side from the scalar case and $h(0) = 0$. Furthermore, we select an initial density which is now a discrete “white” noise with values in $B_1(0) \cap [0, 1]^m$. Thus, the contrast enhancing now pushes the pointwise vector density ρ either to the 0 or to some value on the sphere sector $S^{m-1} \cap [0, 1]^m$ in \mathbb{R}^m (cf. Fig. 6). Again, a straightforward application of the maximum principle ensures $\rho(t, x) \in S^{m-1} \cap [0, 1]^m$ for all t and $x \in \Omega$.

Fig. 7 shows an example for the application of the vector valued anisotropic diffusion method applied to a 2D flow field from a MHD simulation convective flow field. Furthermore, Fig. 8 shows results of this method applied to several time steps of a convective flow field. An incompressible Bénard convection is simulated in a rectangular box with heating from below and cooling from above. The formation of convection rolls will lead to an exchange of temperature. We recognize that the presented method is able to nicely depict the global structure of the flow field, including its saddle points, vortices, and stagnation points on the boundary. Fig. 9 shows results for the same data sets obtained by line integral convolution (here, we used the implementation of Stalling and Hege [21]). Finally, Fig. 10 shows a different application to a porous media flow field.

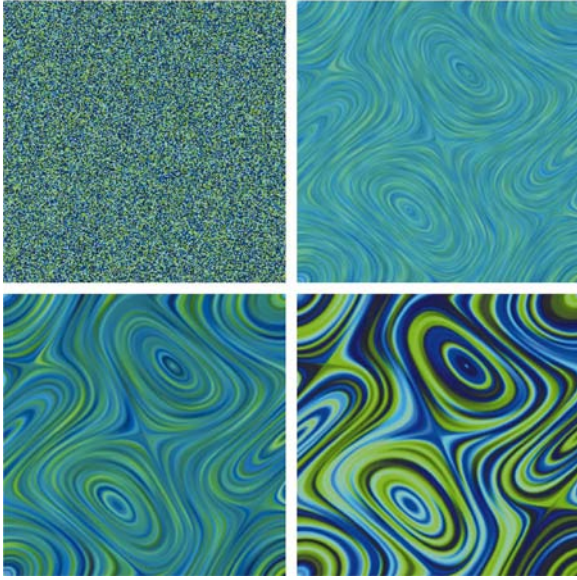


Fig. 7. Different snapshots from the multiscale based on anisotropic diffusion are depicted for a 2D MHD simulation vector field. Here, we consider a two-dimensional diffusion problem and interpret the resulting density as a color in a blue/green color space.

5 APPLICATION IN 3D

The anisotropic nonlinear diffusion problem has been formulated in Section 3 for arbitrary space dimension. It results in a scale of vector field aligned patterns which we then have to visualize. In 2D, this has already been done in a straightforward manner in the above figures. In 3D, we have somehow to break up the volume and open up the view to inner regions. Otherwise, we must confine ourselves with some pattern close to the boundary representing solely the shear flow.

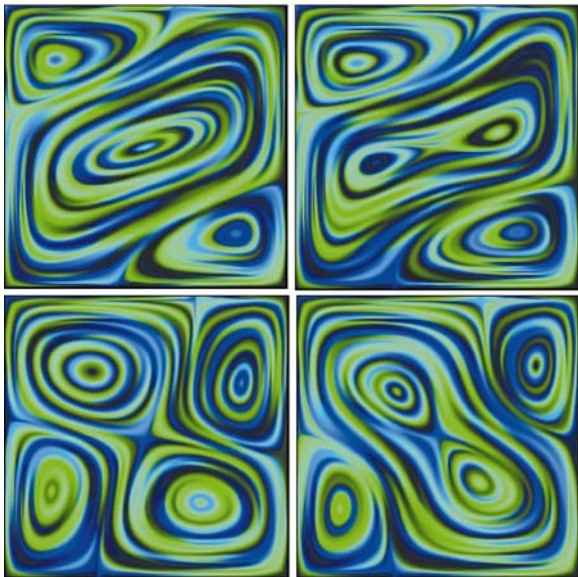


Fig. 8. Convective patterns in a 2D flow field are displayed and emphasized by the method of anisotropic nonlinear diffusion. The images show the velocity field of the flow at different time steps. The resulting alignment is thereby with respect to streamlines of this time dependent flow.

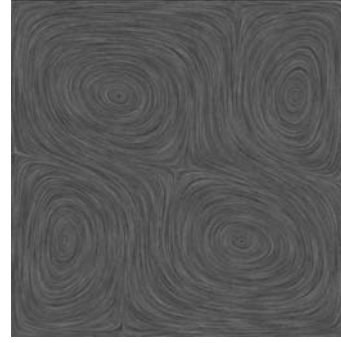


Fig. 9. LIC image generated for one of the data sets that have already been processed in Fig. 8 by nonlinear diffusion (cf. lower left image in Fig. 8).

Here, a further benefit of the vector valued diffusion comes into operation. We know that, for $m = 2$, the asymptotic limits—which differ from 0—are, in mean, equally distributed on $S^1 \cap [0, 1]^2$. Hence, we reduce the informational content and focus on a ball-shaped neighborhood $B_\delta(\omega)$ of a certain point $\omega \in S^1 \cap [0, 1]^2$. Now, we can either look at isosurfaces of the function

$$\sigma(x) = \|\rho(x) - \omega\|^2,$$

where the isolevel δ^2 allows us to depict the boundary of the preimage of $B_\delta(\omega)$ with respect to the mapping ρ (cf. Fig. 11 and Fig. 12). Alternatively, we might use volume rendering to visualize this type of subvolumes. A detailed discussion of the latter approach is beyond the scope of this paper.

6 ANISOTROPIC DIFFUSION ON SURFACES

In the above sections, we have discussed anisotropic diffusion in vector field visualization on domains which are subsets of two and three-dimensional Euclidean space.

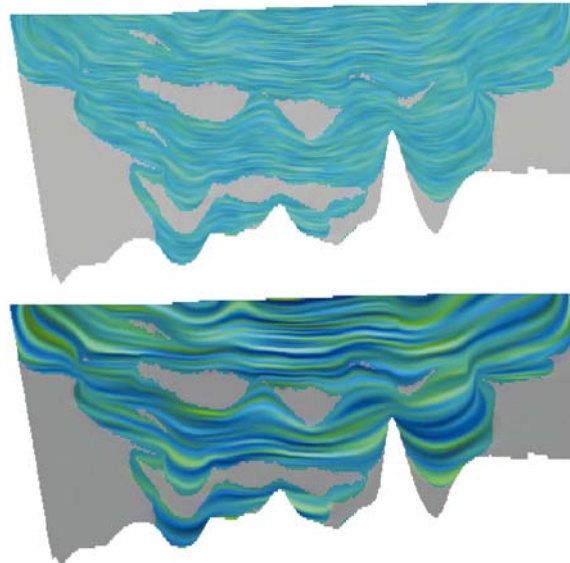


Fig. 10. Field aligned diffusion clearly outlines the principal features of a porous media flow in the vicinity of a salt dome. Lenses of lower permeability force the flow to pass through narrow bridges. We depict two time steps of the diffusion process.

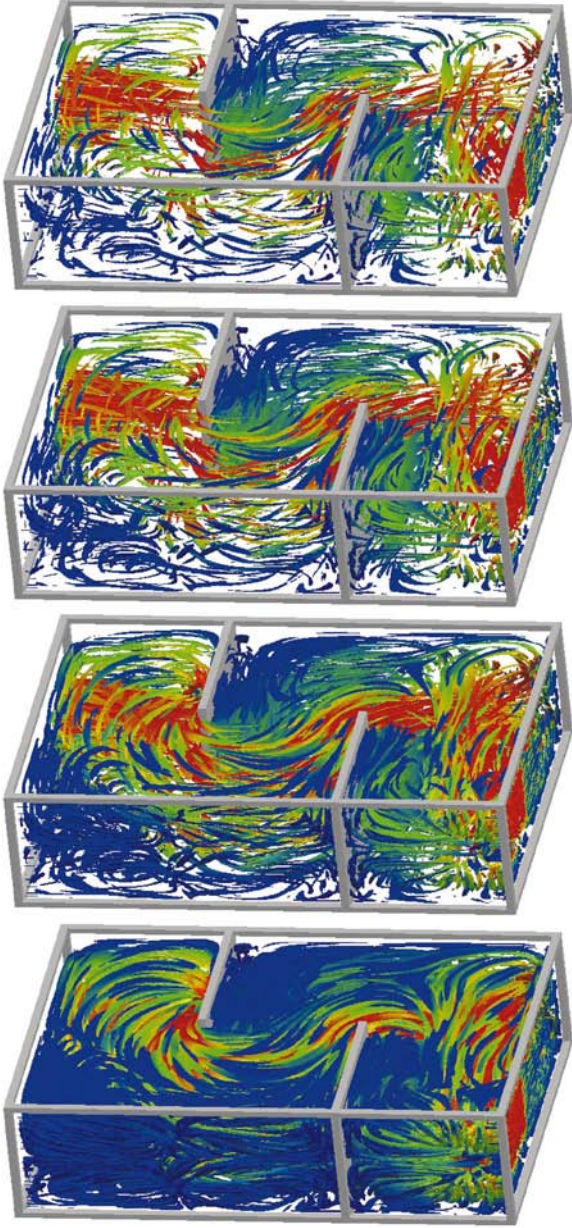


Fig. 11. The incompressible flow in a water basin with two interior walls and an inlet (on the left) and an outlet (on the right) is visualized by the anisotropic nonlinear diffusion method. Isosurfaces show the preimage of $\partial B_\delta(\omega)$ under the vector valued mapping ρ for some point ω on the sphere sector. From top to bottom, the radius δ is successively increased. A color ramp blue–green–red indicates an increasing absolute value of the velocity. The diffusion is applied to initial data, which is a relatively coarse grain random noise.

In what follows, we will outline how to carry over this methodology to display tangential vector fields on surfaces. Important examples are results from meteorological simulations, flow fields on streamsurfaces, or vector fields in differential geometry. The applications presented here will focus on the latter case and present multiscale textures on surfaces representing the principal directions of curvature. Based on the well-established intrinsic differential calculus on manifolds [7], we can pick up the same diffusion problems with an appropriate reinterpretation of the operators. Thus, let us first briefly review the basic notation

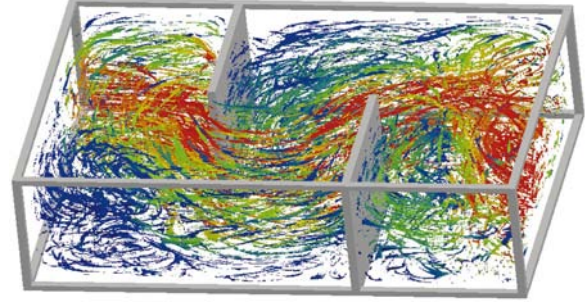


Fig. 12. Nonlinear anisotropic diffusion applied to the same 3D data set as in Fig. 11, but with a fine grain white noise as initial data.

of manifolds, differential calculus, and geometric diffusion. For a detailed introduction to geometry and differential calculus, we refer to [7] and [5, chapter 1]. For the sake of simplicity, we assume our surfaces to be compact embedded manifolds without boundary. Thus, we consider a smooth manifold \mathcal{M} , which we suppose to be embedded in \mathbb{R}^3 . Let $x : \Omega \rightarrow \mathcal{M}$; $\xi \mapsto x(\xi)$ be a coordinate map from an atlas of \mathcal{M} . For each point x on \mathcal{M} , the embedded tangent space $T_x \mathcal{M}$ is spanned by the basis $\{\frac{\partial x}{\partial \xi_1}, \frac{\partial x}{\partial \xi_2}\}$. By $T\mathcal{M}$, we denote the tangent bundle. On \mathcal{M} , the metric $g(\cdot, \cdot)$ as a bilinear form on $T\mathcal{M} \times T\mathcal{M}$ is prescribed by the metric tensor $g = (g_{ij})_{ij}$ with

$$g_{ij} = \frac{\partial x}{\partial \xi_i} \cdot \frac{\partial x}{\partial \xi_j},$$

where \cdot indicates the scalar product in \mathbb{R}^3 . The inverse of g is denoted by $g^{-1} = (g^{ij})_{ij}$. Based on the metric, we can define the integration of a function f on \mathcal{M} . We split up an integral over \mathcal{M} into separate integrals over subsets which are in the image $x(\Omega)$ of some coordinate map x and define

$$\int_{x(\Omega)} f := \int_{\Omega} f(x(\xi)) \sqrt{\det g} d\xi.$$

Integrating either a product of two functions f, g on \mathcal{M} or the product of two vector fields v, w on $T\mathcal{M}$, we obtain the following scalar products on $C^0(\mathcal{M})$ and $C^0(T\mathcal{M})$, respectively:

$$(f, g)_{\mathcal{M}} := \int_{\mathcal{M}} fg dx,$$

$$(v, w)_{T\mathcal{M}} := \int_{\mathcal{M}} g(v, w) dx.$$

Next, we have to introduce the fundamental intrinsic gradient and divergence operators on \mathcal{M} . The gradient $\nabla_{\mathcal{M}} f$ of f is defined as the representation of df with respect to the metric g . We obtain, in coordinates,

$$\nabla_{\mathcal{M}} f = \sum_{i,j} g^{ij} \frac{\partial(f \circ x)}{\partial \xi_j} \frac{\partial x}{\partial \xi_i}.$$

Furthermore, we define the divergence $\text{div}_{\mathcal{M}} v$ for a vector field $v \in T\mathcal{M}$ as the dual operator of the gradient by

$$\int_{\mathcal{M}} \text{div}_{\mathcal{M}} v \phi dx := - \int_{\mathcal{M}} g(v, \nabla_{\mathcal{M}} \phi) dx$$



Fig. 13. The principal directions of curvature are visualized by anisotropic diffusion on a minimal surface.

for all $\phi \in C_0^\infty(\mathcal{M})$.

Finally, with these differential operators at hand, we can discuss a general and intrinsic diffusion on a manifold in analogy to diffusion in Euclidean space: We ask for a solution $\rho : \mathbb{R}_0^+ \times \mathcal{M} \rightarrow \mathbb{R}$ of the parabolic equation

$$\frac{\partial}{\partial t} \rho - \operatorname{div}_{\mathcal{M}}(A \nabla_{\mathcal{M}} \rho) = f(\rho)$$

on $\mathbb{R}_0^+ \times \mathcal{M}$ for given initial data $\rho(0, \cdot) = \rho_0$ on \mathcal{M} . Here, we suppose A to be some positive definite symmetric endomorphism on $\mathcal{T}\mathcal{M}$. Testing with any function $\theta \in C^\infty(\mathcal{M}(t))$ and integrating over \mathcal{M} , we obtain the variational formulation

$$(\partial_t \rho, \theta)_{\mathcal{M}} + (A \nabla_{\mathcal{M}} \rho, \nabla_{\mathcal{M}} \theta)_{\mathcal{T}\mathcal{M}} = (f(\rho), \theta)_{\mathcal{M}}.$$

Now, we consider our actual goal, which is the generation of a texture by nonlinear anisotropic diffusion to represent a given vector field $v \in \mathcal{T}\mathcal{M}$ on the surface. Thus, we suppose A to depend on the vector field v and the norm of the gradient of a convoluted intensity ρ_ϵ :

$$A = A(v, \|\nabla_{\mathcal{M}} \rho_\epsilon\|).$$

For non vanishing v , let $w \in \mathcal{T}_x \mathcal{M}$ be some unit vector normal to v , i.e., $g(v, w) = 0$. Hence, $\{\frac{v}{\|v\|}, w\}$ is a basis of $\mathcal{T}_x \mathcal{M}$ and, with respect to this basis, we define, as before in the Euclidean case,

$$A(v, d) = \begin{pmatrix} \alpha(\|v\|) & \\ & G\|d\| \end{pmatrix}.$$

As righthand side $f(\cdot)$, we pick up the one already introduced in Section 3 and again assume ρ_0 to be a

random noise, either scalar or vector valued, but now prescribed on the surface \mathcal{M} . Furthermore, we have to give a suitable definition of the regularizing presmoothing to obtain ρ_ϵ from the original intensity ρ . Again, we proceed in analogy to the Euclidean case and define ρ_ϵ as the result of the above diffusion problem with $A = \operatorname{Id}$ at time $t = \frac{\epsilon^2}{2}$ and for initial data ρ .

Finally, the resulting family $\{\rho(t)\}_{t \geq 0}$ of intensities on \mathcal{M} gives a multiscale of representations of the given vector field v . Fig. 13 and Fig. 14 show results on different surfaces. We consider the principal directions of curvature as tangential vector fields on which we apply the anisotropic diffusion method. On the underlying triangular grids, the shape operator, whose eigenvalues are the principal curvatures, is approximated as follows: Locally, we regard a single triangle T and all the neighboring triangles which have a nonzero intersection with T as a graph over the plane containing T and calculate the L^2 projection of this piecewise linear graph onto the set of quadratic graphs which are tangential to the plane. Then, we evaluate the constant shape operator on this graph. Let us emphasize that the L^2 projection is always defined, although the local graph property of the triangular grid might not hold in certain degenerate cases.

7 DISCRETIZATION IN 2D AND 3D

In what follows, we discuss the discretization and implementation of the field aligned diffusion method. We will first focus on domains in 2D and 3D Euclidean space. For this purpose, a finite element discretization in space and a semi-implicit backward Euler or second order Crank



Fig. 14. For both principal directions of curvature, different timesteps of the anisotropic diffusion are displayed on the surface of a presmoothed Stanford bunny. In addition, the corresponding principle curvature values are color coded.

Nicolsson scheme in time are considered. Here, we have restricted ourselves to regular grids in 2D and 3D generated by recursive subdivision. On these grids, we consider bilinear, respectively, trilinear, finite element spaces. Numerical integration is based on the lumped masses product $(\cdot, \cdot)^h$ [23] for the L^2 product (\cdot, \cdot) in the variational formulation and a one point quadrature rule for the bilinear form $(A\nabla \cdot, \nabla \cdot)$. Semi-implicit means, for the schemes considered here, that the nonlinearity $A(\cdot)$ is evaluated at the old time. Finally, in each step of the discrete evolution, we have to solve a single system of linear equations. We obtain, for a backward Euler discretization,

$$(M^k + \tau L^k(A^k))\bar{\rho}^{k+1} = M^k \bar{\rho}^k + \tau M^k \bar{f}^k.$$

Here, $\bar{\rho}^k = (\bar{\rho}_i^k)_i$ is the vector of nodal intensity values at time $t^k = k\tau$, where τ is the selected time step size. Furthermore, if we denote the “hat shaped” multilinear basis functions by Φ_i and the diffusion tensor with respect to the discrete intensity at time t^k by A^k ,

$$M^k := ((\Phi_i, \Phi_j)^h)_{ij}$$

$$L^k(A^k) := ((A^k \nabla \Phi_i, \nabla \Phi_j))_{ij}$$

are the lumped mass matrix and nonlinear stiffness matrix, respectively. Finally, the components of the righthand side \bar{f}^k are evaluated by $(\bar{f}^k)_i = f(\bar{\rho}_i^k)$.

The global matrices M^k and $L^k(A^k)$ are assembled from local matrices m^E and l^E with respect to a single element. Their entries correspond to all pairings of local basis functions. Due to the applied lumped mass integration, we immediately verify

$$m_{ij}^E = \frac{1}{2n} \delta_{ij} |E|,$$

where $|E|$ is the volume of the rectangular element E and δ_{ij} the usual Kronecker symbol. For the nonlinear stiffness matrix we obtain

$$l_{ij}^E(A) = |E| \left[\alpha(\|V\|) \left(\nabla \Phi^i \cdot \frac{V}{\|V\|} \right) \left(\nabla \Phi^j \cdot \frac{V}{\|V\|} \right) + \right.$$

$$G(\|D\|) \left(\nabla \Phi^i - \nabla \Phi^i \cdot \frac{V}{\|V\|^2} V \right)$$

$$\left. \cdot \left(\nabla \Phi^j - \nabla \Phi^j \cdot \frac{V}{\|V\|^2} V \right) \right].$$

where $V = v(c_E)$ for the center of mass c_E of E , D the gradient of the presmoothed discrete intensity at c_E , and $\{\Phi^i\}_i$ the set of local basis functions.

In each time step, the computation of the prefiltered intensity vector $\bar{\rho}_\epsilon^n$ is based on a single implicit time step $\epsilon^2/2$ for the corresponding discrete heat equation scheme with respect to initial data $\bar{\rho}^n$.

In our implementation, the regular grids are procedurally interpreted as quadrees, respectively octrees [19]. Finally, no matrix is explicitly stored. The necessary matrix multiplications in the applied iterative CG solver are performed in successive tree traversals. Hierarchical BPX type [3] preconditioning is used to accelerate the convergence of the linear solver. The computation of a single time

step on a 257^2 grid performed on a Silicon Graphics workstation with an R10000 processor requires 1.2 seconds. Computing time in 3D is currently much more expensive. But, there is still a great potential to speed up the algorithm considerably, for instance, by taking into account better ordering strategies for the unknowns which correspond to the anisotropy. This will be exploited in the future. Furthermore, the code is prepared to incorporate spatial grid adaptivity if possible (cf. Fig. 17).

8 DISCRETIZATION ON SURFACES

The discretization of the proposed anisotropic diffusion method on surfaces is completely analogous to the above Euclidean case. We only have to replace the discrete differential operators and bilinear forms by their intrinsic geometric counterparts. We suppose the surface \mathcal{M} to be approximated with a sufficiently fine triangular grid \mathcal{M}_h consisting of nondegenerate triangles T with maximal diameter h . Thus, we only focus on the computation of the local mass matrix m^T and the local nonlinear stiffness matrix $l^T(A)$, respectively. We obtain again by lumped mass integration

$$m_{ij}^T = \frac{1}{3} \delta_{ij} |T|,$$

where $|T|$ is the area of the triangle T . Next, let us consider, for every triangle T , the reference triangle $\hat{T} \subset \mathbb{R}^2$ with independent variables ξ_1, ξ_2 and nodes $\xi^0 = (0, 0)$, $\xi^1 = (1, 0)$, and $\xi^2 = (0, 1)$. Then, an affine coordinate mapping X maps \hat{T} onto T and its nodes ξ^i onto the corresponding nodes P^i of T on the discrete surface in \mathbb{R}^3 . Hence, the corresponding metric tensor is as in the continuous case given by $g_{ij} = \frac{\partial X}{\partial \xi_i} \cdot \frac{\partial X}{\partial \xi_j}$, where $\frac{\partial X}{\partial \xi_i} = P_k^i - P_k^0$. Hence, we can evaluate gradients of the linear basis functions Φ^l corresponding to the nodes P^l by

$$\nabla_{\mathcal{M}_h} \Phi^l = \sum_{i,j} g^{ij} \frac{\partial \Phi^l}{\partial \xi_j} (P^i - P^0),$$

where the derivatives of Φ^l with respect to the reference coordinates ξ are

$$\begin{pmatrix} \frac{\partial \Phi^l}{\partial \xi_1} \\ \frac{\partial \Phi^l}{\partial \xi_2} \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Finally, we calculate the local nonlinear stiffness matrix

$$l_{ij}^T(A) =$$

$$|T| \left[\alpha(\|V\|) \left(\nabla_{\mathcal{M}_h} \Phi^i \cdot \frac{V}{\|V\|} \right) \left(\nabla_{\mathcal{M}_h} \Phi^j \cdot \frac{V}{\|V\|} \right) + \right.$$

$$G(\|D\|) \left(\nabla_{\mathcal{M}_h} \Phi^i - \nabla_{\mathcal{M}_h} \Phi^i \cdot \frac{V}{\|V\|^2} V \right)$$

$$\left. \cdot \left(\nabla_{\mathcal{M}_h} \Phi^j - \nabla_{\mathcal{M}_h} \Phi^j \cdot \frac{V}{\|V\|^2} V \right) \right].$$

where $V = v(c_T)$ for the center of mass c_T of T , D the geometric gradient of the presmoothed discrete intensity on T , and “ \cdot ” still indicates the scalar product in \mathbb{R}^3 .

9 COMPARISON TO OTHER METHODS

So far, we have introduced a novel approach which provides us with an intuitive understanding of complex vector fields. We have discussed a variety of important properties and advantages. Let us now rank this method among other visualization methods and compare it with different techniques. Here, we especially pick up the line integral convolution method and the spot noise approach.

For stationary vector fields, we obtain similar results by all methods. Thin field aligned patterns are generated. Line integral convolution leads to comparable results with the essential difference that the PDE-based method carries a nice scale space property, i.e., evolving a longer time in the anisotropic diffusion method, we obtain a successive coarsening of the resulting pattern representing the vector field.

Furthermore, in a restricted sense, line integral convolution (LIC) and spot noise can be regarded as special cases of the anisotropic nonlinear diffusion method. LIC with Gaussian filter kernel can be identified as the asymptotic limit of the latter method for a concentration of the edge enhancing function $G(\cdot)$ at 0. Other filter kernel shapes correspond to different, in general, nonlinear diffusion processes along streamlines. Further on, generating a single deformed spot on the computational domain, as proposed in [6], can be regarded as an early time step in the diffusion starting with initial data, that is, a characteristic function of a circular disk. If we release a bunch of such disks as initial data in such a way that the evolving patterns do not overlap, then the resulting image is comparable to spot noise. Thus, the original spot noise technique can be regarded as a parallel version of short time diffusive vector field visualization.

10 TOWARD FLOW SEGMENTATION

The above applications already show the capacity of the anisotropic nonlinear diffusion method to outline the flow structure not only locally. Indeed, especially for larger evolution times in the diffusion process, the topological skeleton of a vector field becomes clearly visible. We will now investigate a possible flow segmentation by means of the anisotropic diffusion. Let us restrict this to the two-dimensional case of an incompressible flow with vanishing velocity v at the domain boundary $\partial\Omega$. Then, topological regions are separated by homoclinic, respectively, heteroclinic, orbits connecting critical points in the interior of the domain and stagnation points on the boundary. Critical points, by definition, points with vanishing velocity $v = 0$, may either be saddle points or vortices. Furthermore, we assume critical points to be nondegenerate, i.e., ∇v is regular. Saddle points are characterized by two real eigenvalues of ∇v with opposite sign, whereas, at vortices, we obtain complex conjugate eigenvalues with vanishing real part. Stagnation points on $\partial\Omega$ are similar to saddles. For details we refer to [11]. In each topological region, there is a family of periodic orbits close to the heteroclinic, respectively, homoclinic, orbit. This observation gives reason for the following segmentation algorithm. At first, we search for critical points in Ω and stagnation points on $\partial\Omega$. We

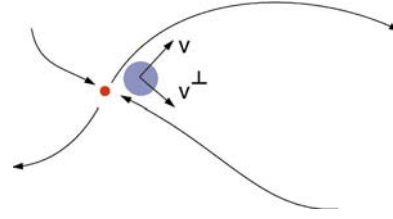


Fig. 15. A sketch of the four sectors at a critical point, the initial spot for the diffusion calculation and the oriented system $\{v, v^\perp\}$.

calculate the directions which separate the different topological regions. In the case of saddle points, these are the eigenvectors of ∇v . Next, we successively place an initial spot in each of the sectors and perform an appropriate field aligned anisotropic diffusion. Let us suppose that a single sector is spanned by vectors $\{s_+, s_-\}$, where the sign \pm indicates incoming and outgoing direction. The method presented in Section 3 would lead to a closed pattern along one of the above closed orbits for time t large enough. To fill out the interior region, we modify the diffusion as follows: Up to now, the Perona Malik diffusions enhance edges of the current image in both directions normal to the velocity. Henceforth, we select an orientation for a “one sided” diffusion (cf. Fig. 15), i.e., we

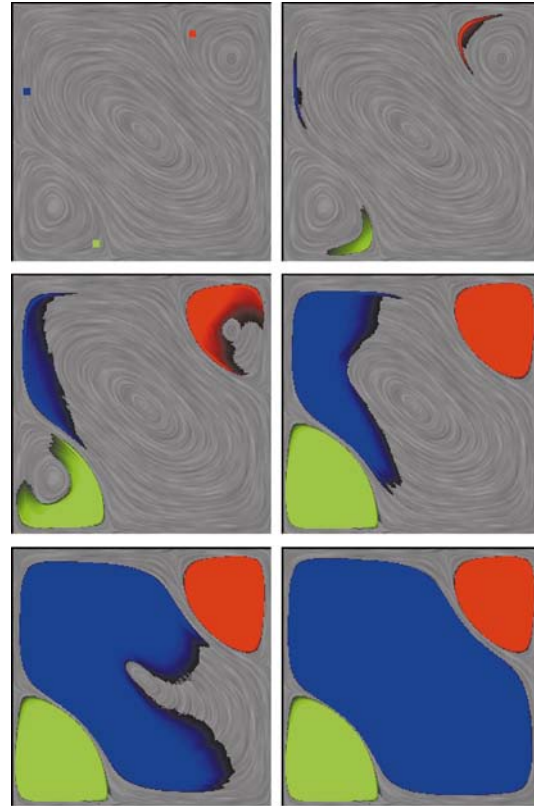


Fig. 16. Nonlinear diffusion segmentation is applied to a velocity field from a Bénard convection. Several time steps are shown starting from initial seed spots in critical point sectors. Here, we have placed these seeds as close as possible in terms of the grid size in the sectors spanned by the eigenvalues of the Jacobian of the velocity. Only to emphasize the evolution process, a single grayscale image from the diffusion calculation (cf. Fig. 8) is underlying the sequence of segmentation time steps.

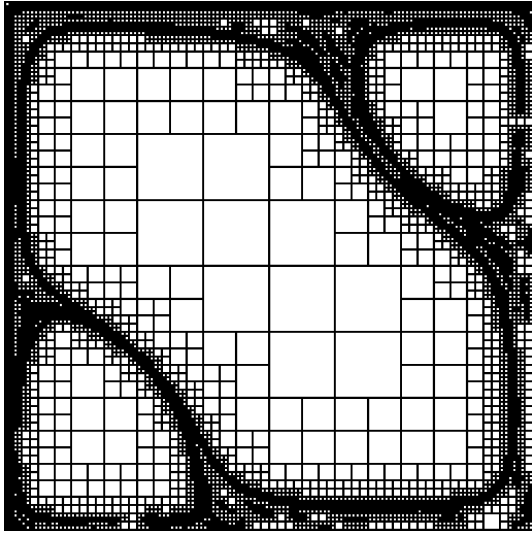


Fig. 17. The adaptive quadtree on which we approximate the segmentation function ρ at a certain time step.

select a unique normal v^\perp to v and consider the diffusion matrix

$$A(v, \nabla \rho_\epsilon) = B(v)^T \begin{pmatrix} \alpha & G((\nabla \rho_\epsilon \cdot v^\perp)_+) \end{pmatrix} B(v),$$

where α is a positive constant and $(s)_+ := \max\{s, 0\}$. Furthermore, we consider a nonnegative, concave function $f: \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$ with $f(0), f(1) = 0$ as a source term in the diffusion equation. If the orientation of $\{s_+, s_-\}$, coincides with that of $\{v, v^\perp\}$, then linear diffusion in the direction toward the interior will fill up the complete topological region. A segmentation of multiple topological regions at the same time is possible if we carefully select the sectors to release initial spots. Fig. 16 shows different time steps of the segmentation applied to a convective incompressible flow. So far, we have seen that anisotropic diffusion has strong provisions for flow segmentation as well. In a certain sense, we thereby identify the complement of what is usually extracted in topology recognition. An outstanding advantage of the new method is its numerical stability and its self-sharpening effect due to the edge enhancing strategy. We pay for this by a higher computational complexity. If we apply a standard implementation on a uniform grid of size n^2 , the segmentation cost is at least $O(n^2)$ compared to an $O(n)$ count of grid cells met by the direct ODE integration to compute the homoclinic and heteroclinic orbits corresponding to the critical points. Fig. 17 shows an adaptive quadtree which allows the same resolution quality for the segmentation function ρ as on a full grid, but now at a much lower cost. We thereby consider a piecewise linear and continuous finite element space on the adaptive quadtree.

11 CONCLUSIONS

We have introduced a new method based on the solution of a nonlinear anisotropic diffusion problem for the post processing of vector data. The method can be applied on 2D or 3D domains, as well as on two-dimensional surfaces embedded in \mathbb{R}^3 . From a mathematical point of view, one of

the major advantages is that it is based on a physically intuitive continuous model, i.e., streamline aligned diffusion. Most of the properties can be discussed on this level. Finally, it is discretized in an appropriate way making use of recent and efficient numerical algorithms.

From the authors' point of view, exciting future research directions are further investigations of flow visualization in 3D. The exploiting of adaptive finite element paradigms and ordering strategies for the unknowns will be especially key issues to reducing the computing costs.

Furthermore, a visualization approach based on anisotropic diffusion and applicable for time dependent vector fields is a challenging topic. Finally, the anisotropic diffusion flow segmentation also carries provisions for the identification of interesting flow regions in 3D, such as recirculation zones and vortex cores.

Further results and the algorithm running on an $n \times m$ 2D vector array is available as a source code at http://www.iam.uni-bonn.de/FktAna_NumMath/Num_Vis/projekte/flow_visualization/.

ACKNOWLEDGMENTS

The authors would like to acknowledge Karol Mikula and Jarke van Wijk for inspiring discussions and many useful comments on image processing and flow visualization. Furthermore, they thank Eberhard Bänsch from Bremen University, Klaus Johannsen from Stuttgart University, Konrad Polthier from the Technical University at Berlin, and Wolfram Rosenbaum from Bonn for providing the incompressible flow data sets, the porous media simulation data, the minimal surface data, and the MHD simulation data sets.

REFERENCES

- [1] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, "Axioms and Fundamental Equations of Image Processing," *Architecture Ration. Mechical Analysis*, vol. 123, no. 3, pp. 199-257, 1993.
- [2] J. Becker and M. Rumpf, "Visualization of Time-Dependent Velocity Fields by Texture Transport," *Proc. Eurographics Scientific Visualization Workshop '98*, 1998.
- [3] J. Bramble, J. Pasciak, and J. Xu, "Parallel Multilevel Preconditioners," *Math. of Computers*, vol. 55, pp. 1-22, 1990.
- [4] B. Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Computer Graphics (SIGGRAPH '93 Proc.)*, J.T. Kajiya, ed., vol. 27, pp. 263-272, Aug. 1993.
- [5] I. Chavel, *Eigenvalues in Riemannian Geometry*. Academic Press, 1984.
- [6] W.C. de Leeuw and J.J. van Wijk, "Enhanced Spot Noise for Vector Field Visualization," *Proc. Visualization '95*, 1995.
- [7] M.P. do Carmo, *Riemannian Geometry*. Boston, Basel, Berlin: Birkhäuser, 1993.
- [8] G. Dziuk, "Finite Elements for the Beltrami Operator on Arbitrary Surfaces," *Partial Differential Equations and Calculus of Variations*, 1988.
- [9] G. Dziuk, "An Algorithm for Evolutionary Surfaces," *Numerische Math.*, vol. 58, pp. 603-611, 1991.
- [10] L. Forsell, "Visualizing Flow over Curvilinear Grid Surfaces Using Line Integral Convolution," *Proc. IEEE Visualization '94*, pp. 240-246, 1994.
- [11] J.L. Helman and L. Hesselink, "Visualizing Vector Field Topology in Fluid Flows," *IEEE Computer Graphics and Applications*, vol. 11, no. 3, pp. 36-46, 1991.
- [12] V. Interrante and C. Grosch, "Strategies for Effectively Visualizing 3D Flow with Volume LIC," *Proc. Visualization '97*, pp. 285-292, 1997.

- [13] J. Kacur and K. Mikula, "Solution of Nonlinear Diffusion Appearing in Image Smoothing and Edge Detection," *Appl. Numer. Math.*, vol. 17, no. 1, pp. 47-59, 1995.
- [14] B. Kawohl and N. Kutev, "Maximum and Comparison Principle for One-Dimensional Anisotropic Diffusion," *Math. Annals*, vol. 311, no. 1, pp. 107-123, 1998.
- [15] M.-H. Kiu and D.C. Banks, "Multi-Frequency Noise for LIC," *Proc. Visualization '96*, 1996.
- [16] N. Max and B. Becker, "Flow Visualization Using Moving Textures," *Proc. ICASE/LaRC Sympo. Time Varying Data, NASA Conf. Publication 3321*, pp. 77-87, 1996.
- [17] N. Max, R. Crawfis, and C. Grant, "Visualizing 3D Velocity Fields Near Contour Surface," *Proc. IEEE Visualization '94*, pp. 248-254, 1994.
- [18] P. Perona and J. Malik, "Scale Space and Edge Detection Using Anisotropic Diffusion," *Proc. IEEE CS Workshop Computer Vision*, 1987.
- [19] T. Preußer and M. Rumpf, "Anisotropic Nonlinear Diffusion in Flow Visualization," *Proc. Visualization 1999*, 1999.
- [20] H.-W. Shen and D.L. Kao, "Uflic: A Line Integral Convolution Algorithm for Visualizing Unsteady Flows," *Proc. Visualization '97*, pp. 317-322, 1997.
- [21] D. Stalling and H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution," *SIGGRAPH 95 Conf. Proc.*, pp. 249-256, Aug. 1995.
- [22] D. Stalling, M. Zöckler, and H.-C. Hege, "Fast Display of Illuminated Field Lines," *IEEE Trans. Visualization and Computer Graphics*, vol. 3, no. 2, Apr.-June 1997.
- [23] V. Thomée, *Galerkin-Finite Element Methods for Parabolic Problems*. Springer, 1984.
- [24] G. Turk and D. Banks, "Image-Guided Streamline Placement," *Computer Graphics (SIGGRAPH '96 Proc.)*, 1996.
- [25] J.J. van Wijk, "Spot Noise-Texture Synthesis for Data Visualization," *Computer Graphics (SIGGRAPH '91 Proc.)*, T.W. Sederberg, ed., vol. 25, pp. 309-318, July 1991.
- [26] J.J. van Wijk, "Flow Visualization with Surface Particles," *IEEE Computer Graphics and Applications*, vol. 13, no. 4, pp. 18-24, July 1993.
- [27] J.J. van Wijk, "Implicit Stream Surfaces," *Proc. IEEE Visualization '93*, pp. 245-252, 1993.
- [28] J. Weickert, *Anisotropic Diffusion in Image Processing*. Teubner, 1998.



Udo Diewald studied mathematics and physics at the University at Kaiserslautern and at Bonn University. He spent one year at the University of Wales at Aberystwyth. Currently, he is working on his Diplom thesis. His interests are geometric evolution problems and non-linear diffusion methods in image and surface processing.



Tobias Preußer studied mathematics at the University of Bonn, Germany, and at the Courant Institute at New York. He received his Diploma degree in 1999. Currently, he is working on his PhD thesis in the field of anisotropic diffusion methods in multiscale feature extraction. While a student, he worked for the National Research Center for Information Technology (GMD). In the spring of 2000, he received a grant from the European Community for his work at the Italian CINECA supercomputing center.



Martin Rumpf received his Diplom degree and his PhD in mathematics from Bonn University in 1989 and 1992, respectively. He has been a professor of applied mathematics at Bonn University since 1996. His research interests are numerical methods for nonlinear partial differential equations, adaptive finite element methods, and computer vision. Furthermore, he is concerned with new flow visualization techniques and efficient multi-scale methods in scientific visualization.

Reaction-Diffusion Models for Vector Visualization: Algorithms and Implementations

Speaker: Mike Kirby

School of Computing and
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT, USA

Work Done In Collaboration With:
Allen Sanderson
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT, USA

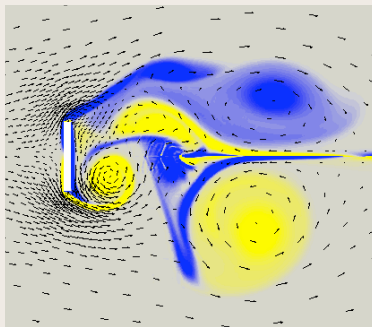


Outline

- I. Introduction
 - A. Motivation
 - B. Basic 2D Vector Visualization Discussion
- II. Reaction-Diffusion Equations
 - A. Uses in Engineering (Examples)
 - B. Mathematical Discussion
 - C. Use within Visualization
- III. Numerical Solution
 - A. Spatial Discretization
 - B. Time Discretization
 - C. Implementation Discussion
- IV. Visualization Examples
- V. Summary

Scientific Computing and Imaging Institute, University of Utah

Back to the Application Why are we interested?



Scientific Computing and Imaging Institute, University of Utah

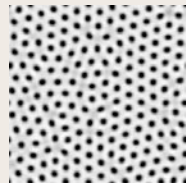
Motivation Visualization Goals

- Create a visualization of a vector field that combines the advantages of dense and sparse representations.
- Dense representations such as Spot Noise and LIC show global orientation but without augmentation lack local magnitude and direction.
- Sparse representations such as streamlines and glyphs give local magnitude and direction but due to occlusion can miss features.
- Represent other scalar values such as uncertainty within the visualization without overloading.

Scientific Computing and Imaging Institute, University of Utah

Motivation

Proposed Solution: Use a Reaction-Diffusion model to create an irregular spatio-temporal pattern.

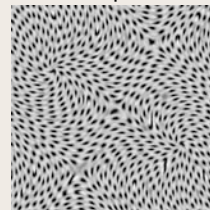


Scientific Computing and Imaging Institute, University of Utah

Motivation

Challenges

- Creating a pattern that represents a vector field.
- Computationally expensive.
- Difficult to form stable patterns.



Scientific Computing and Imaging Institute, University of Utah

Examples of Reaction - Diffusion Models In Engineering

Alan Turing (1952) - The chemical basis of morphogenesis.

$$\begin{aligned}\frac{\partial a}{\partial t} &= s(\alpha - ab) + d_a \nabla^2 a \\ \frac{\partial b}{\partial t} &= s(ab - b - \beta) + d_b \nabla^2 b\end{aligned}$$

Describes the chemical process between two morphogens, a and b.

Scientific Computing and Imaging Institute, University of Utah

Examples of Reaction - Diffusion Models In Engineering

Gray-Scott Equations: Mathematical Biology

$$\begin{aligned}\frac{\partial u}{\partial t} &= -uv^2 + F(1 - u) + d_u \nabla^2 u \\ \frac{\partial v}{\partial t} &= uv^2 - (F + k)v + d_v \nabla^2 v\end{aligned}$$

Scientific Computing and Imaging Institute, University of Utah

Examples of Reaction - Diffusion Models In Engineering

FitzHugh-Nagumo Equations: Electrocardiology

$$\begin{aligned}C_m \frac{\partial v}{\partial t} &= Av(v - \alpha)(1 - v) + d_v \nabla^2 v \\ \frac{\partial w}{\partial t} &= \epsilon(v - \gamma w) + d_w \nabla^2 w\end{aligned}$$

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Previous Research

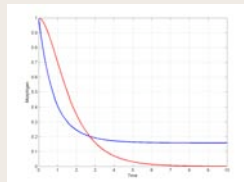
- Reaction - Diffusion for texture generation
 - Turk et al. 1991
 - Witkin et al. 1991
 - Fowler et al. 1992
 - Chambers et al. 1995
- Reaction - Diffusion for tensor visualization
 - Kintleman et al. 2000
- Diffusion for vector visualization
 - Preußner et al. 1999
 - Garcke et al. 2001
- Advection for vector visualization
 - Shen et al. 1996
 - Weiskopf et al. 2004

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Properties

What is "Reaction" Anyway?

$$\begin{aligned}\frac{\partial a}{\partial t} &= s(\alpha - ab) \\ \frac{\partial b}{\partial t} &= s(ab - b - \beta)\end{aligned}$$

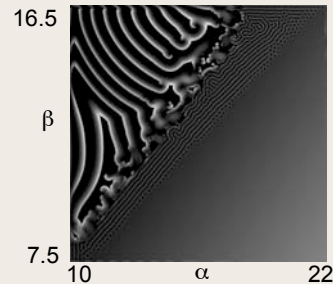


$$\begin{aligned}u(0) &= 1, v(0) = 1 \\ \alpha &= \beta = 0 \\ s &= 1.0\end{aligned}$$

Scientific Computing and Imaging Institute, University of Utah

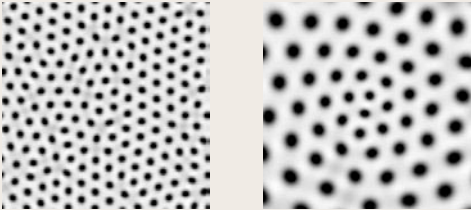
Reaction - Diffusion Properties

Turing Equations: Alpha and Beta control the pattern formed



Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Properties



Changes in the Spot Pattern
For Different Reaction Rates

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Properties

How do we use diffusion?

Embedding the vector field in the diffusion tensor:

$$\frac{\partial u}{\partial t} = H(u, v) + (\nabla \cdot \sigma_u \nabla) u$$

where σ_u is a symmetric positive definite diffusion tensor.

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Properties

Assume that we are working on a finite domain $[a_1, b_1] \times [a_2, b_2] \in \mathcal{R}^2$, and that we are given a regularly-spaced computational grid of size $N_x \times N_y$. At each point (x_i, y_j) , $i = 1, \dots, N_x, j = 1, \dots, N_y$ suppose that we are given an angle $\theta_{ij} \in [0, 2\pi]$.

$$R_{ij} = \begin{pmatrix} \cos \theta_{ij} & \sin \theta_{ij} \\ -\sin \theta_{ij} & \cos \theta_{ij} \end{pmatrix}$$

$$R_{ij}^{-1} = R_{ij}^T = \begin{pmatrix} \cos \theta_{ij} & -\sin \theta_{ij} \\ \sin \theta_{ij} & \cos \theta_{ij} \end{pmatrix}$$

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Properties

$$\Lambda_{ij} = \begin{pmatrix} (\lambda_1)_{ij} & 0 \\ 0 & (\lambda_2)_{ij} \end{pmatrix}$$

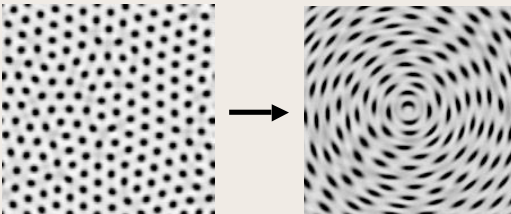
where $(\lambda_1)_{ij}$ is the diffusivity in the first principal direction and $(\lambda_2)_{ij}$ is the diffusivity in the second principal direction.

$$\sigma_{ij} = R_{ij}^T \Lambda_{ij} R_{ij}.$$

$$\sigma_{ij} = \begin{pmatrix} (\sigma_{11})_{ij} & (\sigma_{12})_{ij} \\ (\sigma_{21})_{ij} & (\sigma_{22})_{ij} \end{pmatrix}.$$

Scientific Computing and Imaging Institute, University of Utah

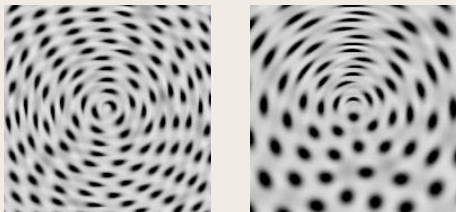
Reaction - Diffusion Properties



Adding Inhomogeneous Anisotropic Diffusion

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Properties



Different "Tricks" With The Diffusion

Scientific Computing and Imaging Institute, University of Utah

SC

Reaction - Diffusion Implementation

$$(\nabla \cdot \sigma(\vec{x}) \nabla) u = \frac{\partial}{\partial x} \sigma_{11}(\vec{x}) \frac{\partial u}{\partial x} + \frac{\partial}{\partial x} \sigma_{12}(\vec{x}) \frac{\partial u}{\partial y} + \frac{\partial}{\partial y} \sigma_{21}(\vec{x}) \frac{\partial u}{\partial x} + \frac{\partial}{\partial y} \sigma_{22}(\vec{x}) \frac{\partial u}{\partial y}$$

$$\begin{aligned} \delta_x^+ u_{ij} &= u_{i+1,j} - u_{i,j} \\ \delta_x^- u_{ij} &= u_{i,j} - u_{i-1,j} \\ \delta_x^+ u_{ij} &= u_{i,j+1} - u_{i,j} \\ \delta_y^- u_{ij} &= u_{i,j} - u_{i,j-1} \\ \delta_x u_{ij} &= u_{i+1/2,j} - u_{i-1/2,j} \\ \delta_y u_{ij} &= u_{i,j+1/2} - u_{i,j-1/2} \\ \bar{\delta}_x u_{ij} &= \frac{1}{2}(u_{i+1,j} - u_{i-1,j}) \\ \bar{\delta}_y u_{ij} &= \frac{1}{2}(u_{i,j+1} - u_{i,j-1}) \\ \mu_x u_{ij} &= \frac{1}{2}(u_{i+1/2,j} + u_{i-1/2,j}) \\ \mu_y u_{ij} &= \frac{1}{2}(u_{i,j+1/2} + u_{i,j-1/2}) \end{aligned}$$

**Basic Finite
Difference
Operators**

Scientific Computing and Imaging Institute, University of Utah

SC

Reaction - Diffusion Implementation

$$(\nabla \cdot \sigma(\vec{x}) \nabla) u = \underbrace{\left(\frac{\partial}{\partial x} \sigma_{11}(\vec{x}) \frac{\partial u}{\partial x} + \frac{\partial}{\partial y} \sigma_{22}(\vec{x}) \frac{\partial u}{\partial y} \right)}_{Term 1} + \underbrace{\left(\frac{\partial}{\partial x} \sigma_{12}(\vec{x}) \frac{\partial u}{\partial y} + \frac{\partial}{\partial y} \sigma_{21}(\vec{x}) \frac{\partial u}{\partial x} \right)}_{Term 2}$$

Term 1 = $\xi_1 \mathcal{F}_1 + \xi_2 \mathcal{F}_2$

and

Term 2 = $\xi_3 \mathcal{F}_3 + \xi_4 \mathcal{F}_4$

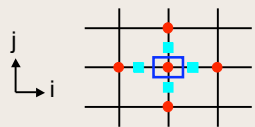
Both $\mathcal{O}(\Delta x^2, \Delta y^2)$ Approximations

Scientific Computing and Imaging Institute, University of Utah

SC

Reaction - Diffusion Implementation

$$\mathcal{F}_1 = \frac{1}{\Delta x^2} (\delta_x^- [\mu_x(\sigma_{11})_{i+1/2,j}] \delta_x^+ u_{ij}) + \frac{1}{\Delta y^2} (\delta_y^- [\mu_y(\sigma_{22})_{i,j+1/2}] \delta_y^+ u_{ij})$$



● u_{ij} variable information

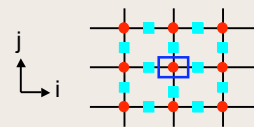
■ σ_{ij} variable information

Scientific Computing and Imaging Institute, University of Utah

SC

Reaction - Diffusion Implementation

$$\begin{aligned} \mathcal{F}_2 = & \frac{1}{4\Delta x^2} (\delta_x^- [\mu_x(\sigma_{11})_{i+1/2,j+1}] \delta_x^+ u_{i,j+1} \\ & + 2\delta_x^- [\mu_x(\sigma_{11})_{i+1/2,j}] \delta_x^+ u_{i,j} \\ & + \delta_x^- [\mu_x(\sigma_{11})_{i+1/2,j-1}] \delta_x^+ u_{i,j-1}) + \\ & \frac{1}{4\Delta y^2} (\delta_y^- [\mu_y(\sigma_{22})_{i,j+1/2}] \delta_y^+ u_{i+1,j} \\ & + 2\delta_y^- [\mu_y(\sigma_{22})_{i,j+1/2}] \delta_y^+ u_{i,j} \\ & + \delta_y^- [\mu_y(\sigma_{22})_{i,j-1/2}] \delta_y^+ u_{i,j-1}) \end{aligned}$$



● u_{ij} variable information

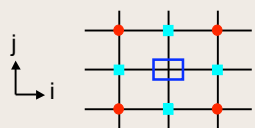
■ σ_{ij} variable information

Scientific Computing and Imaging Institute, University of Utah

SC

Reaction - Diffusion Implementation

$$\mathcal{F}_3 = \frac{1}{\Delta x \Delta y} (\bar{\delta}_x [(\sigma_{12})_{i,j}] \bar{\delta}_y u_{ij} + \bar{\delta}_y [(\sigma_{21})_{i,j}] \bar{\delta}_x u_{ij})$$



● u_{ij} variable information

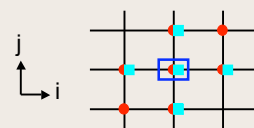
■ σ_{ij} variable information

Scientific Computing and Imaging Institute, University of Utah

SC

Reaction - Diffusion Implementation

$$\begin{aligned} \mathcal{F}_4 = & \frac{1}{2\Delta x \Delta y} ([\delta_x^+ (\sigma_{12})_{ij}] \delta_y^+ u_{ij} + \delta_x^- (\sigma_{12})_{ij} \delta_y^- u_{ij}) \\ & + \frac{1}{2\Delta x \Delta y} ([\delta_y^+ (\sigma_{21})_{ij}] \delta_x^+ u_{ij}) \end{aligned}$$



● u_{ij} variable information

■ σ_{ij} variable information

Scientific Computing and Imaging Institute, University of Utah

Reaction – Diffusion Implementation

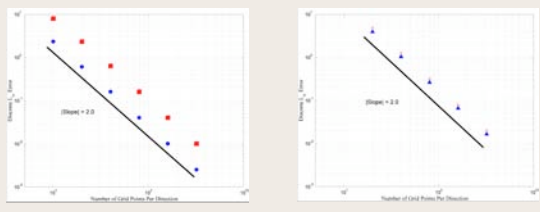
$$(\nabla \cdot \sigma(\vec{x}) \nabla) \vec{U} = \xi_1 \mathcal{F}_1 + \xi_2 \mathcal{F}_2 + \xi_3 \mathcal{F}_3 + \xi_4 \mathcal{F}_4$$

where $\xi_1 + \xi_2 = 1$ and $\xi_3 + \xi_4 = 1$.

$$\begin{aligned}
 (\delta_x [\mu_x(\sigma_{11})_{i+1/2,j}] \delta_x^+ u_{ij}) &= (\delta_x [\mu_x(\sigma_{11})_{i+1/2,j}]) (u_{i+1,j} - u_{i,j}) \\
 &= \delta_x \frac{1}{2} [(\sigma_{11})_{i+1,j} + (\sigma_{11})_{i,j}] (u_{i+1,j} - u_{i,j}) \\
 &= \frac{1}{2} [(\sigma_{11})_{i+1,j} + (\sigma_{11})_{i,j}] (u_{i+1,j} - u_{i,j}) \\
 &\quad - \frac{1}{2} [(\sigma_{11})_{i,j} + (\sigma_{11})_{i-1,j}] (u_{i,j} - u_{i-1,j}).
 \end{aligned}$$

Scientific Computing and Imaging Institute, University of Utah

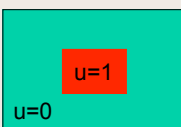
Reaction - Diffusion Implementation

$$\begin{aligned}
 u(x, y) &= \sin 2\pi x \sin 2\pi y \\
 \sigma_{11}(x, y) &= \sin 2\pi x \cos 2\pi y \\
 \sigma_{12}(x, y) &= \sin 2\pi x \cos 2\pi y \\
 \sigma_{21}(x, y) &= \cos 2\pi x \sin 2\pi y \\
 \sigma_{22}(x, y) &= \cos 2\pi x \sin 2\pi y
 \end{aligned}$$


Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

Initial Conditions:



Periodic Boundary Conditions:

$$\begin{aligned}
 u(0, y) &= u(1, y) & v(0, y) &= v(1, y) \\
 u(x, 0) &= u(x, 1) & v(x, 0) &= v(x, 1)
 \end{aligned}$$

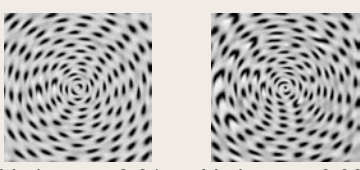
Boundary Conditions:

No-Flux Boundary Conditions:

$$\begin{aligned}
 \mathbf{n} \cdot \sigma \nabla u &= 0 \\
 \mathbf{n} \cdot \sigma \nabla v &= 0
 \end{aligned}$$

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation



Variance = 0.01 Variance = 0.001

Sensitivity to Initiation Condition Perturbation

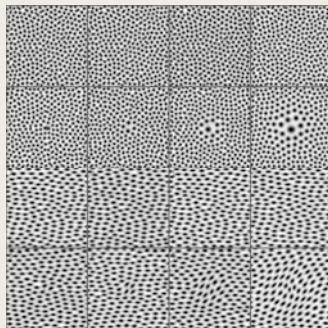
Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

Sensitivity to Resolution

Reaction Rate

Diffusion



Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

Let $A(\vec{u})$ denote a discretized finite difference operator which, when acting upon the \vec{u} returns a second-order centered approximation of the advection operator and let $D(\vec{u})$ denote a discretized finite difference operator which, when acting upon the \vec{u} returns a second-order centered approximation of the inhomogeneous anisotropic diffusion operator.

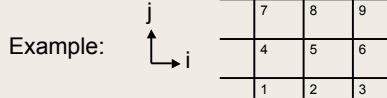
$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} \rightarrow 1/h^2 \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

Let us discretize our morphogens as follows:
 $u_{ij}^n = u(x_{ij}, t_n)$ (and similarly for v) where $t_{n+1} = (\Delta t)t_n$ denotes discretization in time.

Let us define \tilde{u}^n to be an $N^2 \times 1$ vector containing the values of the morphogens over the grid at time level n where the entries of the vector are given by $\tilde{u}_{i+jN}^n = u_{ij}^n$.



Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

Forward-Euler For Reaction/Diffusion

$$\frac{\tilde{u}^{n+1} - \tilde{u}^n}{\Delta t} = F(\tilde{u}^n, \tilde{v}^n) + D(\tilde{u}^n)$$

$$\frac{\tilde{v}^{n+1} - \tilde{v}^n}{\Delta t} = G(\tilde{u}^n, \tilde{v}^n) + D(\tilde{v}^n)$$

where the reaction terms are assumed to be explicit evaluations of the morphogens at the grid points. This scheme is first-order in time.

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

$$\frac{\tilde{u}^{n+1} - \tilde{u}^n}{\Delta t} = F(\tilde{u}^n, \tilde{v}^n) + D(\tilde{u}^{n+1})$$

$$\frac{\tilde{v}^{n+1} - \tilde{v}^n}{\Delta t} = G(\tilde{u}^n, \tilde{v}^n) + D(\tilde{v}^{n+1})$$

Forward-Euler for Reaction
Backward-Euler For Diffusion

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

Given the linearity of the discretized diffusion operator, we can thus re-write the system to be solved as:

$$(\mathbf{I} - \Delta t \mathbf{D}) \tilde{u}^{n+1} = \tilde{u}^n + (\Delta t) F(\tilde{u}^n, \tilde{v}^n)$$

$$(\mathbf{I} - \Delta t \mathbf{D}) \tilde{v}^{n+1} = \tilde{v}^n + (\Delta t) G(\tilde{u}^n, \tilde{v}^n)$$

where \mathbf{I} denotes the $N^2 \times N^2$ identity operator and \mathbf{D} denotes the $N^2 \times N^2$ matrix form of the linear finite difference diffusion operator.

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

$$\frac{\tilde{u}^{n+1} - \tilde{u}^n}{\Delta t} = F(\tilde{u}^n, \tilde{v}^n) + \frac{1}{2} (D(\tilde{u}^n) + D(\tilde{u}^{n+1}))$$

$$\frac{\tilde{v}^{n+1} - \tilde{v}^n}{\Delta t} = G(\tilde{u}^n, \tilde{v}^n) + \frac{1}{2} (D(\tilde{v}^n) + D(\tilde{v}^{n+1}))$$

Forward-Euler For Reaction
Crank-Nicholson For Diffusion

Scientific Computing and Imaging Institute, University of Utah

Reaction - Diffusion Implementation

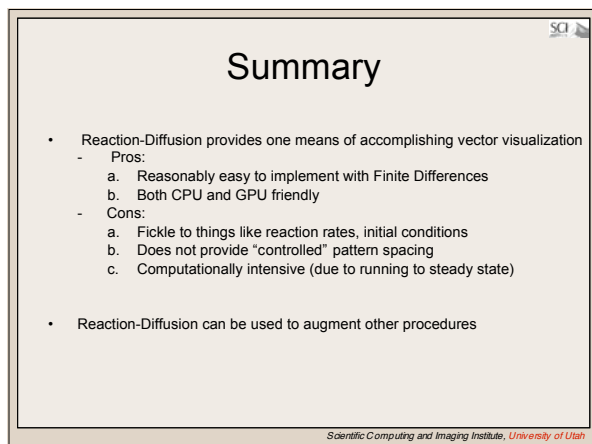
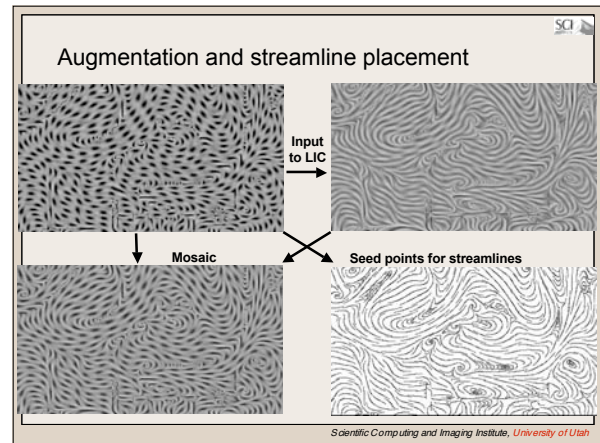
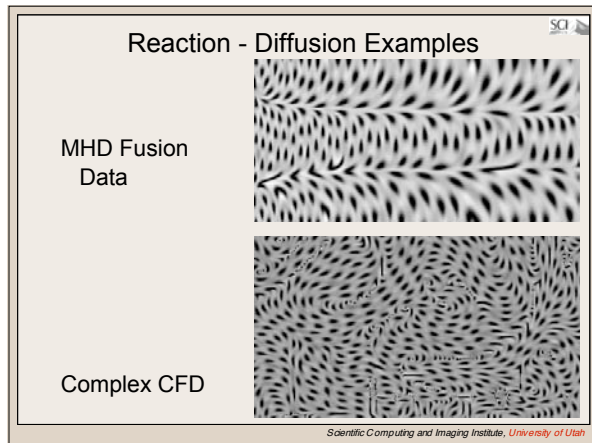
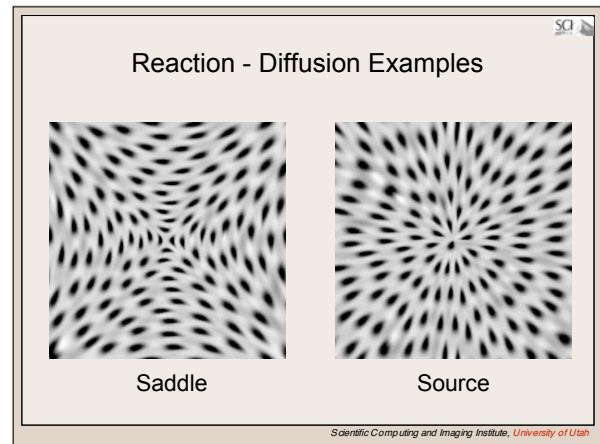
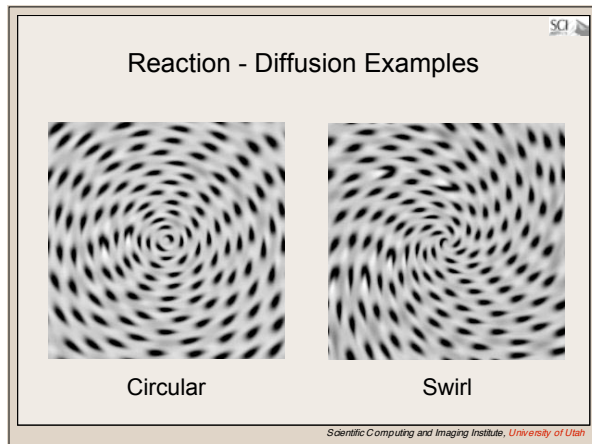
Given the linearity of the discretized diffusion operator, we can thus re-write the system to be solved as:

$$(\mathbf{I} - \frac{1}{2} \Delta t \mathbf{D}) \tilde{u}^{n+1} = \tilde{u}^n + \Delta t \left(F(\tilde{u}^n, \tilde{v}^n) + \frac{1}{2} D(\tilde{u}^n) \right)$$

$$(\mathbf{I} - \frac{1}{2} \Delta t \mathbf{D}) \tilde{v}^{n+1} = \tilde{v}^n + \Delta t \left(G(\tilde{u}^n, \tilde{v}^n) + \frac{1}{2} D(\tilde{v}^n) \right)$$

where \mathbf{I} denotes the $N^2 \times N^2$ identity operator and \mathbf{D} denotes the $N^2 \times N^2$ matrix form of the linear finite difference diffusion operator.

Scientific Computing and Imaging Institute, University of Utah



Display of Vector Fields Using a Reaction-Diffusion Model

Allen R. Sanderson, Chris R. Johnson, and Robert M. Kirby*

Scientific Computing and Imaging Institute, University of Utah

ABSTRACT

Effective visualization of vector fields relies on the ability to control the size and density of the underlying mapping to visual cues used to represent the field. In this paper we introduce the use of a reaction-diffusion model, already well known for its ability to form irregular spatio-temporal patterns, to control the size, density, and placement of the vector field representation. We demonstrate that it is possible to encode vector field information (orientation and magnitude) into the parameters governing a reaction-diffusion model to form a spot pattern with the correct orientation, size, and density, creating an effective visualization. To encode direction we texture the spots using a light to dark fading texture. We also show that it is possible to use the reaction-diffusion model to visualize an additional scalar value, such as the uncertainty in the orientation of the vector field.

An additional benefit of the reaction-diffusion visualization technique arises from its automatic density distribution. This benefit suggests using the technique to augment other vector visualization techniques. We demonstrate this utility by augmenting a LIC visualization with a reaction-diffusion visualization. Finally, the reaction-diffusion visualization method provides a technique that can be used for streamline and glyph placement.

Keywords: Vector Field Visualization, Flow Visualization, Reaction-Diffusion, Vector Fields

1 INTRODUCTION

Visualizing vector fields is important for many computational applications, including fluid dynamics, wind and water currents in climate modeling, bioelectric fields in neuroscience, and magnetic fields in nuclear fusion. To meet the needs arising from this diverse set of applications, many different techniques for visualizing vector fields have been developed [1, 3, 7, 8, 19, 22, 31, 33, 30]. Each technique has strengths and weaknesses in its ability to represent the magnitude, orientation, direction, uncertainty and topological structures of the associated vector field.

For instance, the simplest method for displaying a vector field is to place glyphs representing the vector direction and magnitude at regular intervals. However, because of scaling differences, overlap between the glyphs can occur. This can produce visual clutter and occlusion [25]. The problem is further compounded when data are displayed in three dimensions. Displaying normalized vector values can reduce the clutter but at a loss of information. Even when the visual clutter can be overcome, displaying vector fields using regular intervals may not be appropriate. This is because the grid spacing may not correspond to the underlying vector field.

More complicated techniques, such as streamlines, can provide powerful visual cues [10]. However, enough streamlines must be placed to provide cues without causing visual clutter. Streamlines

may be placed selectively to reduce the clutter, but such selective placement may cause critical areas to be missed [28].

With the exception of a glyph-based method, no technique is singularly able to visualize uncertainty in vector fields. In [17] Pang et al. demonstrated several different glyphs for characterizing uncertainty in vector fields. However, as a glyph based method it can also succumb to clutter and occlusion.

Given the various shortcomings in many of the current vector field visualization techniques, the main goal of this work was to develop an automated method that uses the vector magnitude, orientation, direction, and uncertainty to control the shape, size, orientation, direction, and density of the objects used to represent the vector field. At the same time, we wanted a method that would be mesh independent and produce a visualization that would be natural and pleasing to the eye. To achieve these goals, we have explored the use of a reaction-diffusion model for vector field visualization.

2 BACKGROUND AND PREVIOUS WORK

Because of its importance to scientific computing applications, creating effective techniques for visualizing vector fields is an active area of research. Given the large number of techniques that have been developed, it is not practical to review each technique; a very complete review can be found in [18]. Instead, we focus on three related areas for visualizing vector fields: the use of random patterns, selective placement, and reaction-diffusion.

The use of random patterns for visualizing a vector field has been explored by van Wijk [29], Cabral and Leedom [3], Shen et al. [23], and others using either spot or white noise to form a dense representation of the vector field by integrating an ODE that represents the basis for a streamline. By dense, we mean that there is value for each grid location. Others, such as Preußer, [20] have formulated the problem as a PDE describing nonlinear anisotropic diffusion. With the ODE and PDE formulations the resulting image has a brush-stroke appearance of variable thickness. While this type of image is useful for showing the vector orientation, it fails to provide information about magnitude and direction. These shortcomings have been addressed in various forms by adding directional [23, 30] and magnitude [5, 13] cues.

More recent work has focused on creating images that are less dense, but still contain useful information about the vector field, e.g. [12, 24, 28]. Turk and Banks [28] explored a method to bundle similar streamlines until an energy function is minimized. Once the function is minimized, the streamlines can be replaced with variable sized curved arrows to show direction and magnitude. Kirby et al. [12], were able to achieve similar results using a random placement of variable sized arrows. Once an arrow is placed, a Poisson distribution disk based on the vector magnitude is used to prevent other arrows from being placed near it. However, because the arrow represents just the value at a single location rather than representing the values in a neighborhood of the sample, it is possible to create the illusion that an area is homogeneous when it is not.

Computer graphics applications have also made use of a reaction-diffusion model to generate texture maps [27, 32]. These types of textures are useful for forming patterns that are natural looking and are typically applied to organic models such as ani-

*e-mail: {allen,crj,kirby}@sci.utah.edu

mals. Turk explored the use of different reaction models to produce a variety of patterns [27]. At the same time, Witkin and Kass [32] used anisotropic diffusion to form different patterns. These patterns can be classified as either spot or stripe patterns.

Rather than forming the texture and then applying it to a model using a traditional texture mapping, Turk exploited the fact that a reaction-diffusion model can be used on an irregular grid. This allowed him to create textures directly on a tessellated surface, avoiding any warping between model space and parameter space. It is possible to make use of this same property to texture isosurfaces, which is a very common visualization tool. In a similar vein, Chambers and Rockwood [4] employed a reaction-diffusion model to generate a solid texture, which is used on a surface and on a volume. Like Witkin and Kass, Chambers and Rockwood also used an anisotropic diffusion technique to form stripe patterns.

Although the reaction-diffusion model was initially mentioned by Cabral and Leedom as a possible model for visualizing data [3], the first researchers to use the method were Kindlmann et al. [11]. They created a solid texture using a reaction-diffusion model based on tensor values from diffusion-weighted magnetic resonance images. This anisotropy formed elliptical “blobs,” which were then volume rendered. Our approach is similar in that we also use a tensor but rather than having tensor data supplied we create the tensor from vector data. This gives several additional degrees of freedom that can be used to encode additional data.

Perhaps the most closely related work is that of Garcke et al. [6] where they use the vector field to define the amount anisotropy in the diffusion. Further, they are able to incorporate a shaped particle to show direction as well as orientation. They use both methods to create visualizations of clustered vector fields.

We now present a more detailed look at reaction-diffusion, and describe how it can be used to visualize vector field data.

3 REACTION DIFFUSION

In 1952 Turing [26] proposed a reaction-diffusion model for describing the chemical process between two morphogens within a series of cells. Due to the dynamics of the system, the morphogens both react and diffuse which changes their concentration within each cell. With time, the morphogens may form a stable pattern representing the dynamic equilibrium of the system. The pattern formation is not dependent on the initial state of the system; the dynamics of the system drives the concentrations toward an equilibrium state.

Turing described the reaction-diffusion of a two morphogen model as a set of nonlinear partial differential equations:

$$\frac{\partial a}{\partial t} = F(a, b) + d_a \nabla^2 a \quad (1)$$

$$\frac{\partial b}{\partial t} = G(a, b) + d_b \nabla^2 b \quad (2)$$

where a and b are the morphogen concentrations; F and G are the functions controlling the production rate of a and b , respectively; d_a and d_b are the diffusion rates, and $\nabla^2 a$ and $\nabla^2 b$ are the Laplacians of a and b , respectively. Turing further defined F and G as:

$$F(a, b) = s(a - ab) \quad (3)$$

$$G(a, b) = s(ab - b - b) \quad (4)$$

where a and b again are the morphogen concentrations, a and b are the formation and degradation rates of a and b respectively, and s is the reaction rate.

For the state to change, there must be some perturbation in the initial conditions which is a stable solution. This perturbation can

arise from a non-uniformity in either the initial concentrations, a and b , or the formation and degradation rates, a and b . A nonuniform formation and degradation rate can be interpreted as being the natural variation within each cell.

After the system is put into motion, the morphogen concentrations will change until a dynamic equilibrium is reached and thus a stable pattern is formed. Although the pattern is stable, the morphogen concentration in each cell will continue to change. However, the change is statistically very small.

Turing’s equations are just one specific instance of reaction-diffusion phenomena. Other similar variations can be found in the literature, including those in [16]. In this paper, we have focused on the use of Turing’s model because it provides results that are indicative of reaction-diffusion in general. We note that the techniques provided in this paper can be similarly applied to other reaction-diffusion systems.

3.1 Mapping the Reaction-Diffusion Kinetics

In order to use a reaction-diffusion model for visualization, a mapping must be established between the vector field and the reaction-diffusion model. There are three possibilities: a mapping between the vector field and the reaction kinetics, a mapping between the vector field and the diffusion kinetics, or a combination of both. Because of the instabilities associated with a reaction-diffusion model, we have focused on finding a mapping for the vector magnitude to the reaction kinetics and the vector orientation to the diffusion kinetics.

3.2 Reaction Kinetics

Within Turing’s reaction kinetics there are several free variables. We first mapped the patterns formed as a function of the two reaction values, a and b . The patterns formed can be described as being finger print or spot patterns. However, these formed only inside a very narrow band of values for a and b . On either side of this band a stable pattern did not occur. For this work we chose a and b to be $16 \pm 1\%$ and $12 \pm 1\%$ respectively, which produced spot like patterns.

The other free parameter in the reaction kinetics is the reaction rate s . Changing the reaction rate changes the size of the pattern formed. This provides an ideal mapping to a scalar value such as the vector magnitude. Thus, it is possible to create patterns of varying size with the size directly relating to the vector magnitude.

It should be noted that similar results can be obtained by varying the diffusion rates d_a and d_b . As such, it is neither the reaction nor the diffusion rates that changes the size, but rather their relative difference. For simplicity and clarity, we vary only the reaction rate for each cell.

3.3 Diffusion Kinetics

The diffusion kinetics as written in Equations (1) and (2) have one free parameter per equation, the diffusion rates d_a and d_b . As previously noted, changing the diffusion rate changes the size of the pattern formed, and for our purposes it is fixed. However, this is not the only free parameter in the diffusion equation. If we relax the isotropic diffusion condition and use anisotropic diffusion we are able to create a broader range of patterns. Assuming anisotropy, the reaction-diffusion equation can be generalized as:

$$\frac{\partial u}{\partial t} = H(u, v) + (\nabla \cdot s_u \nabla) u, \quad (5)$$

where s_u is a symmetric positive definite diffusion tensor into which we will encode the vector field of interest.

Assume that we are working on a finite domain $[a_1, b_1] \times [a_2, b_2] \in \mathbb{R}^2$, and that we are given a regularly-spaced computational grid of size $N_x \times N_y$. At each point (x_i, y_j) , $i = 1, \dots, N_x$, $j = 1, \dots, N_y$ suppose that we are given a vector $\vec{v}_{ij} = (u_{ij}, v_{ij})^T$. To embed this vector field into a tensor field, define $q_{ij} = \arctan \frac{v_{ij}}{u_{ij}} \in [0, 2\pi]$.

We can now define a rotation matrix and its inverse based upon the angle determined above:

$$\mathbf{R}_{ij} = \begin{pmatrix} \cos q_{ij} & \sin q_{ij} \\ -\sin q_{ij} & \cos q_{ij} \end{pmatrix} \quad (6)$$

$$\mathbf{R}_{ij}^{-1} = \mathbf{R}_{ij}^T = \begin{pmatrix} \cos q_{ij} & -\sin q_{ij} \\ \sin q_{ij} & \cos q_{ij} \end{pmatrix} \quad (7)$$

We now define a principal diffusivity matrix Λ which is a diagonal matrix and gives the diffusivity coefficients along the two principal axes of diffusion:

$$\Lambda_{ij} = \begin{pmatrix} (l_1)_{ij} & 0 \\ 0 & (l_2)_{ij} \end{pmatrix} \quad (8)$$

where $(l_1)_{ij}$ is the diffusivity in the first principal direction and $(l_2)_{ij}$ is the diffusivity in the second principal direction.

With the definitions above we can define a diffusivity tensor s_{ij} based on our vector field as

$$s_{ij} = \mathbf{R}_{ij}^T \Lambda_{ij} \mathbf{R}_{ij}. \quad (9)$$

We then combine the spatially nonuniform anisotropic diffusion matrix with a discrete finite difference Laplacian as outlined in [9] in a manner that maintains second-order convergence. With this control, we now have our desired mapping between the vector orientation and the diffusion kinetics. Witkin and Kass [32] took a similar approach for creating 2D texture patterns but assumed a spatially uniform anisotropic diffusion matrix.

3.4 Directional Texturing

The final step in the mapping process is to encode directional information in the pattern created. Up to this point we have described the use of a reaction-diffusion model for generating non-specific patterns. The most common pattern formed using a reaction-diffusion model is a spot pattern. The exact formation of the spot pattern will be discussed in the following section. Assuming an oriented elliptical spot pattern, we show the direction by texturing the spots with a contrasting light to dark fading texture. The texturing is local for each spot using the following steps:

Step 1: Assuming dark spots on a light background, normalize the image values to be between 0 and 255.

Step 2: Find the centroid of each spot by first thresholding to remove all pixels with a value greater than 64. Second, thin the remaining pixels into single pixels [21]. Label the remaining single pixels as the centroid.

Step 3: For each centroid find all connected pixels with a value less than 128.

Step 4: For each connected pixel, calculate the dot product between the vector from the centroid and the connected pixel and the underlying vector field value at the centroid.

Step 5: Normalize the dot product to be between 0 and 1 based on the minimum and maximum dot product values within the spot.

Step 6: If the normalized dot product is less than .9, interpolate between the minimum and maximum gray scales in the spot. This becomes the new gray scale value that gives the dark to light fading on the spot. Otherwise the gray scale is set to be 255, which produces a contrasting light tip on the spot.

The texturing does not change the size or orientation of the spots; merely highlights the direction using values that maintain their natural appearance in the image. A light to dark fading provides directional cues because the fading has a natural strong to weak association. Wegenkittl et al. [30], took a similar approach to create oriented streamlines. The texturing is fully demonstrated in the following section.

4 IMAGE FORMATION

A reaction-diffusion image is created using a forward Euler integration on the discrete version of Equation (5) for a and b until a dynamic equilibrium state is reached, at which time, a stable pattern will have formed. We have found that using a cell size of 1.0 and a step size of 0.5 provides a balance between numerical stability and the pattern formation.

Figure (1a) shows a spot pattern created with the Turing model. Analysis of the image shows that the spot placement is balanced. That is, there is a uniform density of spots with equal spacing around them. This balancing process can be observed during the integration process when a spot begins to form in an area of lower concentration. Other nearby spots adjust themselves so they are not too close to the newly formed spot. Sometimes this adjustment may come in the form of a change in the position of the spots or when one or more of the spots disappears and its concentration is absorbed by remaining spots. This natural organization is one of the properties of reaction-diffusion equations that makes them very useful for visualization purposes.

Circular spots alone do not show magnitude, orientation, or direction. As discussed previously, to show magnitude we scale the spots using the reaction rate, s to reflect the vector magnitude, Figure (1b). To show orientation we compress the spots into elliptical shapes by applying an anisotropic diffusion matrix where the values along the principal axis have a 3:1 ratio. Next, we rotate the diffusion matrix for each vector so that the ellipse's major axis is aligned with the vector field. Once the system comes to a dynamic equilibrium and the spots have formed, the light to dark texture is applied to each spot to show direction, Figure (2a). Figure (2b) shows the magnitude, orientation, and direction combined.

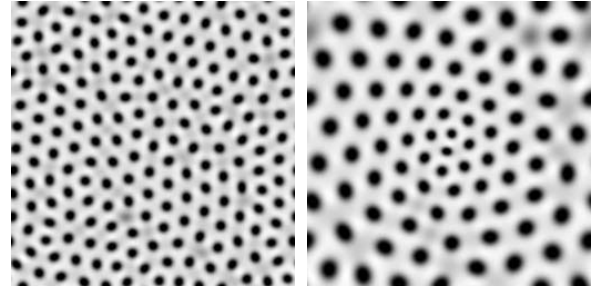


Figure 1: (a) Reaction-Diffusion visualization of circular spots of constant size and (b) variable size.

We now show the application of our reaction-diffusion model to visualize a set of idealized vector fields. Our goal was to see if it was possible to capture the nature of different types of vector fields. These fields include the electric field from a dipole and an electrostatic charge, Figure (3); and a vector field for a saddle and a sink, Figure (4). In each vector field the magnitude and orientation changes smoothly. The properties of vector field can easily be discerned as the ellipsoidal spots have the correct size and orientation. In the case of the circular, saddle, and sink vector fields, a change in the vector magnitude occurs as the vector field moves away from the center and is shown by a corresponding change in the spot size. However, in Figures (2-4), the images have several spots that did

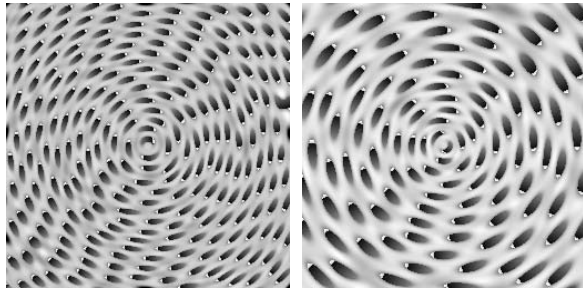


Figure 2: (a) Reaction-Diffusion visualization of a circular vector field showing orientation and direction and (b) magnitude, orientation, and direction.

not form very well, appearing to be smeared together. This is attributable to the variance of the formation and degradation factors, a and b which need to have enough variance to perturb the system but not so much as to cause irregularities in the patterns.

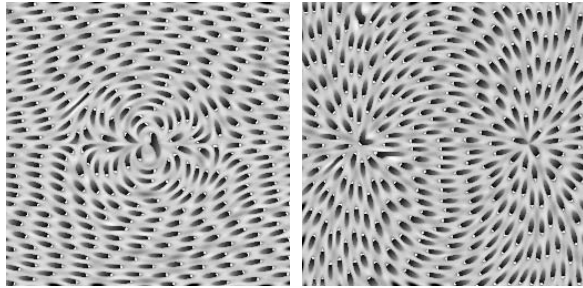


Figure 3: (a) Reaction-Diffusion visualization the electric field from a dipole and (b) an electrostatic charge.

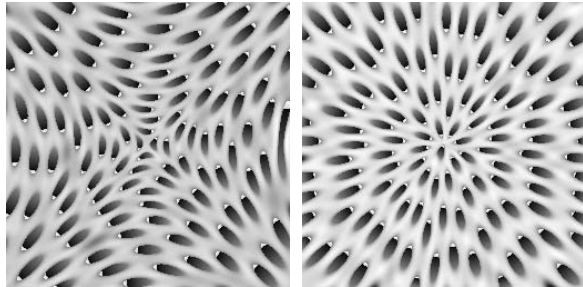


Figure 4: (a) Reaction-Diffusion visualization of a saddle vector field and (b) a sink vector field.

One of the unique features of using a reaction-diffusion model with anisotropic diffusion as we have done is that the spot, although appearing random, naturally align themselves into pseudo streamlines. Further, when the vector field is curved, the spots are not perfectly elliptical, but rather a bean-like shape. This is due to the spatially nonuniform anisotropy influencing the overall spot shape. Another feature of the reaction-diffusion model is that, due to the diffusion, faint streaks emanate from the ends of the spots. These streaks act to connect the spots, further aiding in visualizing the vector field.

This aligning, bending, and streaking all give the observer cues to the underlying vector field. But critical areas may also be of interest. This is another area where the reaction-diffusion model gives visual cues. For instance, at locations where the vector field is diverging, the spots are no longer elliptical but assume odd shapes. If the vector field is diverging equally in all directions the spots will be circular. As such, oddly shaped or circular spots could indicate,

critical areas, or as will be demonstrated in the next section, the location of a large uncertainty in orientation. These are locations the observer may want to inspect further. For instance, in Figures (2-4) the spots are elliptical and are aligned with the vector field throughout the image except at the critical areas.

We have shown that it is possible to view different vector fields using a reaction-diffusion model. However, when implementing the reaction-diffusion method, a question regarding resolution arises: what is the minimum resolution required for individual features to be seen? By its nature, the process of diffusion acts to smooth, lowering high concentrations and raising low concentrations. As such, it is possible to lose individual features that are significantly different than their neighbors.

To determine the minimum resolution at which features can be seen, we oversampled a vector field until it was possible to see the impact of a single vector that was significantly different in both magnitude and orientation than its surrounding, otherwise constant neighbors. This is demonstrated in Figures (5) and (6) for both magnitude and orientation, respectively. It is not until there is an oversampling of eight times the original sample that the magnitude will significantly impact its neighbors to be visually noticeable. Similarly, it takes an oversampling of eight times for the angle to impact its neighbors. Unfortunately, for large vector fields, oversampling is not always practical because it may require significantly more computational time. As such, when visualizing a vector field without oversampling features less than eight nodes in size may be smoothed out.

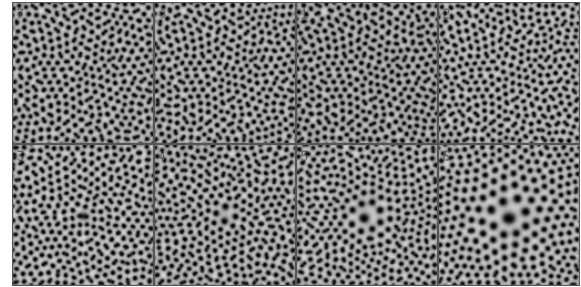


Figure 5: Effect of a single value on the spot size with an oversampling of 0, 1, 2, 4, 8, 16, 32, and 64 times.

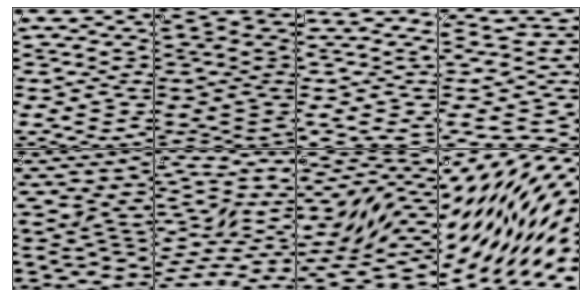


Figure 6: Effect of a single value on the spot orientation with an oversampling of 0, 1, 2, 4, 8, 16, 32, and 64 times.

4.1 Uncertainty Measurements

In the previous examples, we have fixed the amount of anisotropy in the diffusion matrix. However, this is not necessary. If we allow the anisotropy to vary, we can map and visually represent another variable. In this case, we define the amount of anisotropy to be the ratio of the values along the principal axes in the diffusion matrix. When the amount of anisotropy is small, the spot formed is circular. Where when the anisotropy is high, the spot formed is elliptical, at

times, almost to the point of being a thick line. This difference is very well suited to mapping an orientation uncertainty. When the orientation uncertainty is very small the spot maintains its elliptical shape, reflecting a precise orientation. When the uncertainty is very high, the spot is more circular, reflecting the uncertainty in the orientation. In the previous examples, the ratio of the values along the principal axes in the diffusion tensor was fixed at 3:1. We now allow it to vary between 5:3 and 7:1. This is demonstrated in Figure (7) where the uncertainty is a function of the angular position. In a subsequent example, instead of encoding the uncertainty, we encode the vorticity of the vector field,

$$w_{ij} = (\nabla \times u_{ij}), \quad (10)$$

which can be visualized as a scalar in 2D.

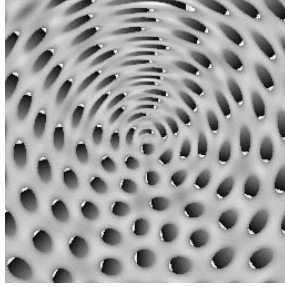


Figure 7: Reaction-Diffusion visualization of orientation uncertainty. The orientation uncertainty is a function of the angular position.

4.2 Augmentation and Automatic Streamline/Glyph Placement

Figure (8) shows our reaction-diffusion model for visualizing an idealized vector field that contains three saddles and two vortices. To augment the LIC image we have taken Kirby et al.'s [12] painter's approach by using a LIC image as an undercoat with the reaction-diffusion as the topcoat. In addition to the trending pattern formed by the spots, the brush stroke appearance from the LIC image enhances the ability of the observer to follow the vector field, while the spots provide magnitude and direction.

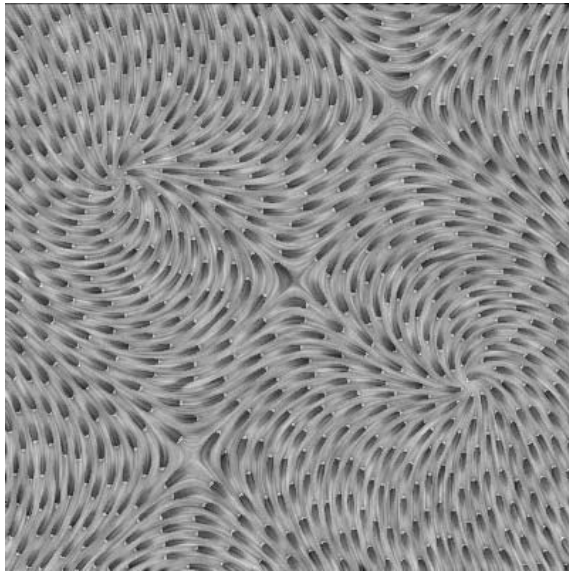


Figure 8: Reaction-Diffusion visualization of multiple vector fields using an underlying LIC image as a base coat.

One of the features of using a reaction-diffusion model is that it also provides a mechanism for automatic streamline or glyph placement. By finding the centroid of each spot using standard image processing techniques [21], it can serve as a seed point for placing a streamline, Figure (9) or a glyph, Figure (10). Although there is one streamline per spot in Figure (9), there appears to be fewer streamlines because of their alignment and slight overlap. Because the density of the spots is based on the magnitude of the underlying vector field in Figure (10) it was possible to scale the arrow glyphs to reflect the vector magnitude without causing occlusion. Further, because of the density relationship, the number of glyphs also reflects, albeit inversely, the magnitude of the vector field. If a uniform density is desired then all that needs to be done is to run the reaction-diffusion using a constant magnitude term which would then determine the density.

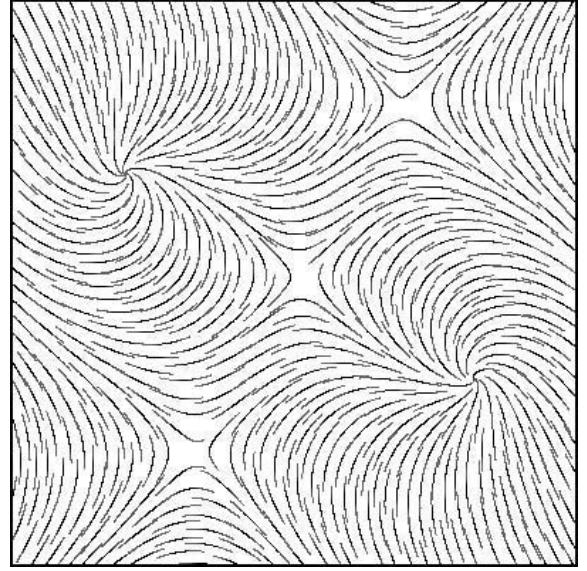


Figure 9: Automatic streamline placement based on the spot centroids in Figure (8).

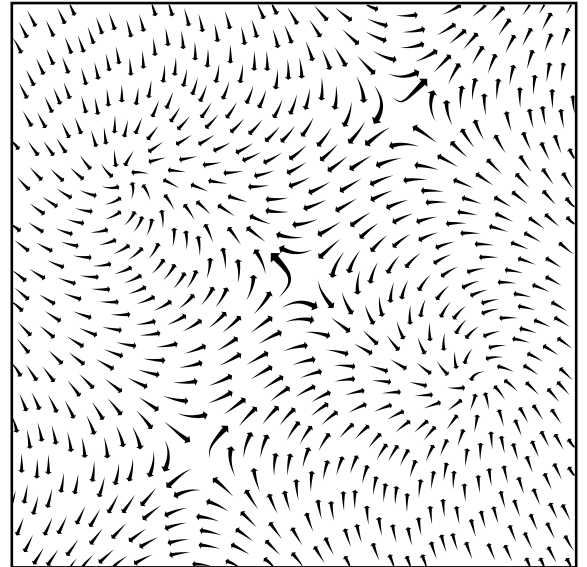


Figure 10: Automatic arrow glyphs placement based on the spot centroids in Figure (8).

5 RESULTS AND DISCUSSION

We now apply our reaction-diffusion model to a numerical simulation of the nonlinear magneto-hydrodynamics (MHD) that occur in the DIII-D tokamak nuclear fusion reactor. The vector field shown in Figures (11-13) is a reoriented two-dimensional slice of the magnetic field in the Tokamak reactor. In Figure (11), just the magnitude and vorticity of the vector field is visualized with no orientation information. The greater the vorticity, the more symmetric the spots become. This gives a good example of how this technique can be used for visualizing two scalar values. Figure (12) is the same vector field showing the orientation and direction. Finally, in Figure (13), the vector field is shown with magnitude, orientation, and direction. In Figure (14), we have encoded the magnitude, orientation, direction, and vorticity of the vector field. Although the spots in Figure (14) appear in different locations than in Figure (13) a comparison of the two images shows a significant difference in the spot shape in those areas with high vorticity.

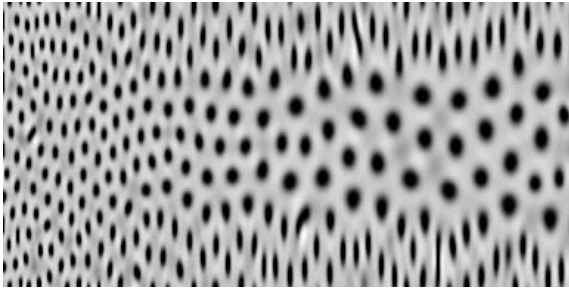


Figure 11: Reaction-Diffusion visualization of a MHD Magnetic vector field. Magnitude and Vorticity are shown.

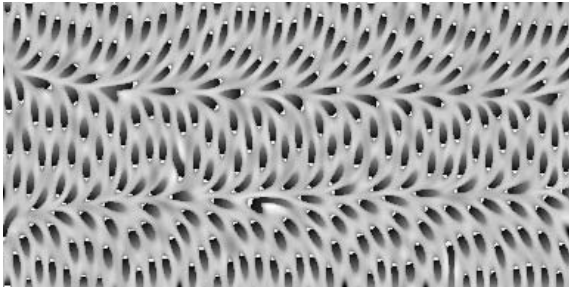


Figure 12: Reaction-Diffusion visualization of a MHD Magnetic vector field. Orientation and direction are shown.

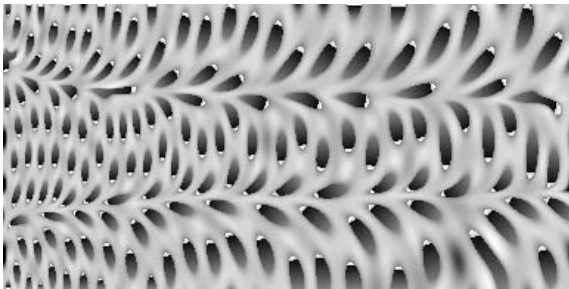


Figure 13: Reaction-Diffusion visualization of a MHD Magnetic vector field. Magnitude, orientation, and direction are shown.

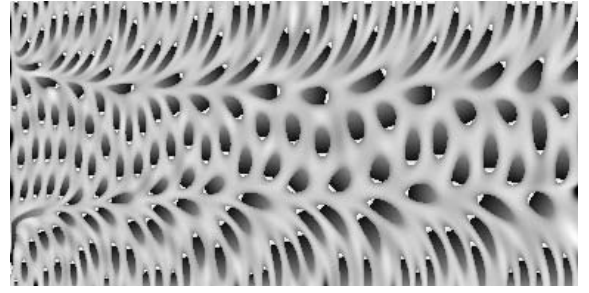


Figure 14: Reaction-Diffusion visualization of a MHD Magnetic vector field. Magnitude, orientation, direction, and vorticity are shown.

5.1 Relationship to Other Vector Field Visualization Techniques

It is also possible to set the reaction rate to zero and use just the diffusion tensor to create other types of visualizations. To produce a LIC-like image, we change the diffusion tensor to be 1D, Figure (15a). To produce convective patterns, such as those proposed by Preußer [20], we use a highly anisotropic 2D diffusion tensor, Figure (15b). This shows that the diffusion model used is consistent with previously published results.

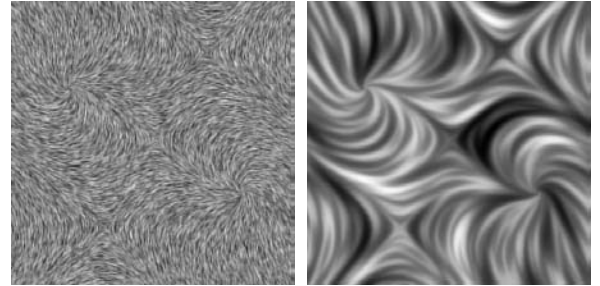


Figure 15: (a) Visualization with a LIC like appearance and (b) with a convective patterns using just the diffusion kinetics.

5.2 Comparison with Other Vector Field Visualization Techniques

We now compare the reaction-diffusion images with four different vector field visualization techniques. Figure (16) shows vector glyphs at regular intervals [25], Figure (17) shows oriented ellipses based on a Poisson distribution [12], Figure (18) shows a line integral convolution [3], and Figure (19) shows image guided streamlines [28]

Placing glyphs at regular intervals is much simpler and quicker than using a reaction-diffusion model, but, as previously discussed, occlusion can be a problem. Using a random Poisson distribution solves the occlusion problem but fails to provide any organization, which is often a key to producing an effective visualization. Using a reaction-diffusion model overcomes the occlusion problem because the spots have a density that is based upon the vector magnitude. Another problem with regular and random intervals is that they may mislead the eye by forming a pattern that may not be part of actual vector field. Conversely, the reaction-diffusion model forms spots in a pattern that follows the underlying structure of the field.

When comparing the reaction-diffusion method to LIC, we can see that both techniques visualize the vector field in a manner that is natural and easy to observe by producing a dense image representation of the vector field. With reaction-diffusion images, different reaction rates produce spots at different densities. The less dense the spots, the greater the chance that areas of interest may

be missed. However, images with a high density of spots may be difficult to view because of the Moray patterns that can form. As such, the density of the spots is a critical component for an effective reaction-diffusion vector field visualization. Currently, the only way to control the density is by using different reaction rates. However, we are also investigating different reaction-diffusion models, such as the Gray-Scott model.

Unlike traditional LIC images, which do not contain magnitude or direction information, the reaction-diffusion model is able to naturally incorporate this information into the visualization. Including the magnitude and direction greatly enhances the visualization. LIC images, along with other noise-based techniques, can be extended to show the magnitude, but these techniques do so at a loss of vector field detail because of blurring used to emphasize the magnitude [5, 13].

Next, we compare the reaction-diffusion image to a visualization using the image-guided streamline technique developed by Turk and Banks [28]. Both techniques are similar in that both are able to show magnitude, orientation, and direction. However, the reaction-diffusion technique represents magnitude more intuitively than the image guided technique. This is because instead of using length to represent magnitude the reaction-diffusion technique uses a width, which is more intuitive.

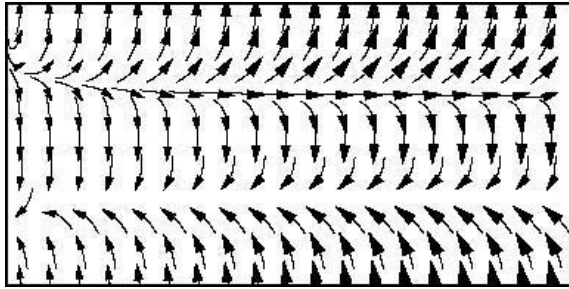


Figure 16: Uniform sampled vector glyph image of the vector field used in Figure (11).

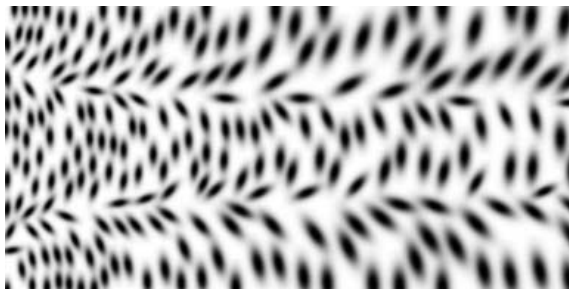


Figure 17: Randomly placed ellipse image of the vector field used in Figure (11).

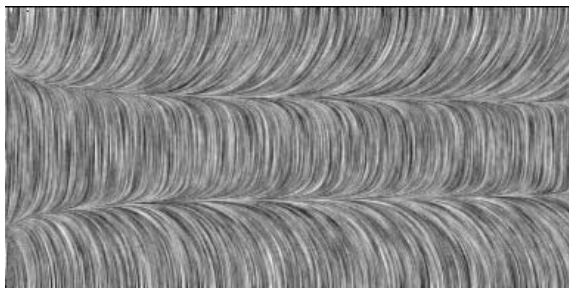


Figure 18: LIC image of the vector field used in Figure (11).

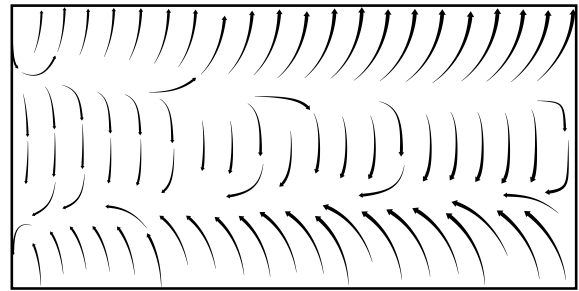


Figure 19: Image guided streamlines generated by Turk and Bank's algorithm of the vector field used in Figure (11).

One of the drawbacks of using a reaction-diffusion model compared to the other techniques is the computational expense. Using an explicit formulation, the patterns take 15-25k iterations to form and become stable. Using a GPU implementation based on Lefohn et al. [15] this takes less than a minute. Whereas it is possible to produce LIC images at interactive rates [2]. We are currently investigating GPU based multi-grid and implicit integration techniques, which should reduce the computational expense.

An additional problem that can occur during pattern formation is that spots can form and fail to separate, as shown in Figure (3a). Where this happens is random and appears to be dependent on the initial conditions. We have observed that it tends to happen more frequently with smaller spots.

One of the greatest benefits of using a reaction-diffusion model is the ability to seamlessly integrate uncertainty measurements in the model. None of the other techniques, with the exception of vector glyphs, are able to show uncertainty as part of their representation [17]. Although vector glyphs can show uncertainty, occlusion remains a problem in their use.

An additional benefit is that the reaction-diffusion technique can be used alone or to augment other techniques. Furthermore, it is possible to use the centroids of the spots to provide a set of seed points for placing streamlines and scaled glyphs.

6 CONCLUSIONS AND FUTURE WORK

We have introduced the use of a reaction-diffusion model that can produce patterns with different shapes, sizes, orientations, and directions for visualizing vector fields. We are able to control the pattern formation by mapping two of the vector field components, orientation and magnitude, to the diffusion and reaction kinetics, respectively. In addition, we also can map an orientation uncertainty to the diffusion kinetics. This mapping produces a spot pattern that is highly representative of the underlying vector field. To show direction we have applied a light to dark fading texture to each spot.

The principle advantage of the reaction-diffusion model over existing vector field visualization techniques is that the pattern size and density that naturally arises from the reaction-diffusion model accurately represents the underlying vector field. Further, the shape of the pattern (e.g. the spots) not only contains information concerning magnitude, orientation, and direction but also may contain other information, such as uncertainty or vorticity.

We have also demonstrated the use of the reaction-diffusion model for the automatic placement of streamlines or glyphs and shown how it can be augmented other techniques. Although we have not used color to highlight certain features, one could easily incorporate color to further enhance the visual attributes.

Future work includes extension of the reaction-diffusion algorithm to three dimensions. In such an extension, the reaction kinetics remain the same; only the diffusion kinetics must be extended. The output is a three-dimensional texture that can be volume ren-

dered using various techniques or may be applied to two dimensional surfaces. The image generated would have similar characteristics to those generated by Kindlmann and Weinstein [11] and Chambers and Rockwood [4] and unfortunately suffer from the same visualization problems.

Finally, there are a number of perceptual issues that require further investigation, including a formal user study such as the one performed by Laidlaw et. al. [14] to determine the effectiveness of the reaction-diffusion visualization technique in comparison to other vector field visualization techniques. One area of particular interest is quantifying the effectiveness of the natural patterns that form from using a reaction-diffusion model.

7 ACKNOWLEDGMENTS

This work was supported, in part, by the DOE SciDAC National Fusion Collaboratory and by grants from NSF, NIH, and DOE. The authors are grateful to Miriah Meyer for generating the images using a GPU implementation, Ravi Samtaney and Scott Kruger for providing data and to Greg Turk for providing the code for the image guided streamline.

8 APPENDIX

One of the difficulties in using a reaction-diffusion model is the inherent instability of the system. Below are the parameters used to obtain the stable pattern shown in Figure (1a), which are applied to Eqs. (1-4) using a discrete central difference Laplacian on a uniform grid.

$$a = 4.0$$

$$b = 4.0$$

$$D_a = 1.0 / 4.0$$

$$D_b = 1.0 / 16.0$$

$$a = 16.0 \pm 1\%$$

$$b = 12.0 \pm 1\%$$

$$s = 1.0 / 64.0$$

REFERENCES

- [1] D.C. Banks and B.A. Singer. A predictor-corrector technique for visualizing steady flow. *IEEE Transactions on Visualization and Computer Graphics*, 2:151–163, 1995.
- [2] U. Bordoloi and H.W. Shen. Hardware accelerated interactive vector field visualization: A level of detail approach. *Computer Graphics Forum*, 21(3), 2002.
- [3] B. Cabral and C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of SIGGRAPH 93*, pages 263–270. ACM SIGGRAPH, 1993.
- [4] P. Chambers and A. Rockwood. Visualization of solid reaction-diffusion systems. *IEEE Computer Graphics and Applications*, pages 7–11, 1995.
- [5] W.C. de Leeuw and J.J. van Wijk. Enhanced spot noise for vector field visualization. In *IEEE Visualization 95 Proceedings*, pages 233–239, 1995.
- [6] H. Garcke, T. Preußer, M. Rumpf, A. Telea, U. Weikard, and J. van Wijk. A phase field model for continuous clustering on vector fields. *IEEE Transaction on Visualization and Computer Graphics*, 7(3):230–241, 2001.
- [7] J. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, May 1991.
- [8] L. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proceedings of IEEE Visualization '92*, pages 171–177. IEEE Computer Society Press, 1992.
- [9] G.E. Karniadakis and R. M. Kirby II. *Parallel Scientific Computing in C++ and MPI*. Cambridge, New York, 2003.
- [10] D.N. Kenwright and G.D. Mallinson. A 3-D streamline tracking algorithm using dual stream functions. In *IEEE Visualization 92*, pages 62–68, 1992.
- [11] G.L. Kindlmann, D.M. Weinstein, and D. Hart. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Trans. Visualization and Computer Graphics*, 6(2):124–138, 2000.
- [12] R.M. Kirby, H. Marmanis, and D.H. Laidlaw. Visualizing multivalued data from 2d incompressible flows using concepts from painting. In *Proceedings of the IEEE Visualization 99*, 1999.
- [13] Ming-Hoe Kiu and David C. Banks. Multi-frequency noise for LIC. In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 121–126, 1996.
- [14] David H. Laidlaw, Robert M. Kirby, Cullen D. Jackson, J. Scott Davidson, Timothy S. Miller, Marco da Silva, William H. Warren, and Michael Tarr. Comparing 2d vector field visualization methods: A user study. *IEEE Transactions on Visualization and Computer Graphics, Accepted for Publication*, 2004.
- [15] A.E. Lefohn, J.M. Kniss, C.D. Hansen, and R. Whitaker. Interactive deformation and visualization of level set surfaces using graphics hardware. In *IEEE Visualization 2003*, pages 75–82, October 2003.
- [16] J.D. Murray. *Mathematical Biology*. Springer-Verlag, New York, 1989.
- [17] A.T. Pang, C.M. Wittenbrink, and S.K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, November 1997.
- [18] F. Post, B. Vrolijk, H. Hauser, R.S. Laramée, and H. Doleisch. Feature extraction and visualization of flow fields. *Eurographics 2002*, pages 69–100, 2002.
- [19] F.J. Post, T. van Walsum, Post F.H., and D. Silver. Iconic techniques for feature visualization. In *IEEE Visualization 95*, pages 288–295, 1995.
- [20] T. Preußer and M. Rumpf. Anisotropic nonlinear diffusion in flow visualization. In *Proceedings of Visualization '99*, pages 325–332. IEEE Computer Society Press, Los Alamitos, CA, October 1999.
- [21] A. Rosenfeld and A.Kak. *Digital Picture Processing*. Academic Press, 1982.
- [22] C. Schroeder, W. Volpe and W. Lorensen. The stream polygon: A technique for 3D vector field visualization. In *IEEE Visualization '91*, pages 126–132. IEEE Press, 1991.
- [23] H.W. Shen, C.R. Johnson, and K.L. Ma. Global and local vector field visualization using enhanced line integral convolution. In *Symposium on Volume Visualization*, pages 63–70. IEEE Press, 1996.
- [24] A. Telea and J. J. van Wijk. Simplified representation of vector fields. In *Proceedings of the IEEE Visualization 99*, pages 35–42, 1999.
- [25] E.R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Conn., 2nd edition, 2001.
- [26] A.M. Turing. The chemical basis of morphogenesis. *Phil. Trans. Roy. Soc. Lond.*, B237:37–72, 1952.
- [27] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics (SIGGRAPH)*, 25(4):289–298, 1991.
- [28] G. Turk and D. Banks. Image-guided streamline placement. *Computer Graphics (SIGGRAPH)*, pages 453–460, 1996.
- [29] J.J. van Wijk. Spot noise: Texture synthesis for data visualization. *Computer Graphics*, 25(4):309–318, 1991.
- [30] R. Wegenkittl and Eduard Groller. Fast oriented line convolution for vector field visualization via the internet. In *Visualization '97*, pages 309–316, Phoenix, AZ., 1997. IEEE Press.
- [31] R. Westermann, C. Johnson, and T. Ertl. A level-set method for flow visualization. In *IEEE Visualization 2000*, pages 147–154, Salt Lake City, 2000. IEEE Computer Society.
- [32] A. Witkin and M. Kass. Reaction-diffusion textures. *Computer Graphics (SIGGRAPH)*, 25(4):299–308, 1991.
- [33] M. Zockler, D. Stalling, and H-C. Hege. Interactive visualization of 3D-vector fields using illuminated stream lines. In *Visualization '96*, San Francisco, CA., 1996. IEEE Press.