Image Processing for Volume Graphics

(A Course for SIGGRAPH 2001)

Terry S. Yoo

National Library of Medicine
National Institutes of Health
Building 38A, Room B1N30
8600 Rockville Pike
Bethesda, MD 20894

yoo@nlm.nih.gov

Phone: yoo - (301) 435-3268
Fax: (301) 402-4080

Raghu Machiraju

Computer and Information Science
The Ohio State University
395 Dreese Laboratories
2014 Neil Avenue
Columbus, OH 43210-1277, USA

raghu@cis.ohio-state.edu

Phone: raghu - (614) 292-6730
Fax: (614) 292-2911

Summary:

This course presents the essential tools and techniques for processing volume data as part of rendering and visualization. We will examine the processes of 1) linear and nonlinear filtering, 2) interpolation, 3) reconstruction, 4) feature extraction, 5) segmentation, and 6) model fitting. Both basic fundamentals of linear image processing as well as more advanced nonlinear methods in level set and energy-minimization-based techniques will be covered, illustrated by case studies and applications.

Image Processing for Volume Graphics

Organizers:

Terry S. Yoo
National Library of Medicine
National Institutes of Health

Raghu Machiraju
Department of Computer and Information Science
the Ohio State University

Coursel level: Intermediate

Proposed length: Full day

Summary Statement:

This course presents the essential tools and techniques for processing volume data as part of rendering and visualization. We will examine the processes of 1) linear and nonlinear filtering, 2) interpolation, 3) reconstruction, 4) feature extraction, 5) segmentation, and 6) model fitting. Both basic fundamentals of linear image processing as well as more advanced nonlinear methods in level set and energy-minimization-based techniques will be covered, illustrated by case studies and applications.

Expanded Statement:

Conventional volume rendering has been used to visualize volume data as a sampled density map. However, more and more people are looking at high dimensional data that has noise, occluding surfaces, limited resolution, etc., requiring users to do more "processing". Advanced volume processing is what enables filtering, interpolation, reconstruction, feature extraction, and model fitting. This course covers the reconstruction of continuous models from sampled data, the application of transfer functions for shading and classification, and the segmentation and projection of the reconstructed values for rendering. The goal is to illuminate structures within the data without unwanted artifacts or masking detail.

Prerequisites

Basic knowledge of 3D computer graphics and an understanding of the basic principles of image-processing.

Topics:

Participants will gain insights into the mathematics of the elements of effective volume visualizations and the processes by which they are created. In addition, we will cover emerging topics in volume data processing including level sets, shape extraction using adaptive deformable systems and model-based segmentation.

Course Speakers:

Guido Gerig
University of North Carolina at Chapel Hill

Ioannis Kakadiaris
University of Houston

Raghu Machiraju
the Ohio State University

Torston Moeller
Simon Fraser University

Ross Whitaker
University of Utah

Terry S. Yoo
National Library of Medicine, NIH

Course Syllabus:

Our goal is to elevate the level of the debate regarding the mathematics of volume graphics. We will treat the issues of reconstruction, analysis, and resampling from the perspective of continuous mathematical and statistical models. In order to keep the material from being dry and theoretical, a great deal of emphasis is being placed on examples and case studies. Examples will be presented periodically throughout the course, illuminating the motivations, benefits, and potential pitfalls of the image processing techniques being discussed. The afternoon will be dedicated to emerging techniques relevant to both 3D image processing and volume graphics. The selected topics represent active areas of research that should interest those who wish to explore new research in volume graphics.

We have designed the course to flow from one topic to the next, beginning with the elementary mathematics of volume image processing, continuing to linear and then nonlinear methods. In the afternoon, emerging topics, too, flow from basic tools, through model-based methods, and end with new approaches in level sets.

Course Syllabus

1st morning session - (90 min. total)

INTRODUCTION TO VOLUME DATA FOR COMPUTER GRAPHICS [Yoo] (10 mins)

The connection between mathematics and visualization

Sources of volume data:

- Scanning of physical objects and medical subjects.
- Range data.
- Simulation and CFD.

Topics to be covered and not to be covered, & what we hope to accomplish.

FILTERING AND FREQUENCY FUNDAMENTALS [Yoo] (30 mins)

Filters

Frequency – a biological example

Fourier Transforms

The convolution theorem and the Fourier slice theorem

LINEAR FILTERING AND FILTER DESIGN [Machiraju] (50 mins)

Derivatives of volume data.

Sources of error and error propagation in filtering.

Approximation, interpolation, and anti-aliasing.

---BREAK (15 min.)---

2nd morning session - (105 min. total)

ASSESSMENT OF QUALITY IN VOLUME RENDERING [Moeller and Machiraju] (90 mins)

Function Reconstruction from Discrete Data.

Exploring transfer functions for volume rendering: Cause and effect examples.

Error analysis of reconstruction processing and derivative filters

INTRODUCTION TO NONLINEAR FILTERING [Yoo] (15 mins)

The diffusion equation.

Nonlinear diffusion.

Geometry driven diffusion.

---LUNCH---

Course Syllabus (cont.)

1st afternoon session - (100 Mins total)

ELASTICALLY ADAPTIVE DEFORMABLE MODELS [Kakadiaris] (50 min.)

Generating energy functionals for segmentation.
Deformable surfaces and contours.
Finite element and finite differencing techniques for deformable models.
Examples.

MODEL BASED SEGMENTATION [Gerig] (50 mins)

Shape representation: Alternatives to triangular meshes.
Shape variability.
Object evolution toward a target shape.
Examples.

---BREAK (15 min.)---

2nd afternoon session - (95 min. total)

LEVEL SETS [Whitaker] (90 min.)

Blobby models and beyond.
Level set basics.
Evolving isosurfaces.
Applications of level set methods to morphing and reconstruction of range data.

WRAPUP [Yoo or Machiraju] (5 min.)

Suggested reading.
Suggested venues, journals, and conferences.

Speaker Information

Guido Gerig
Taylor Grandy Professor of Computer Science and Psychiatry
Department of Computer Science
University of North Carolina at Chapel Hill
CB #3175, Sitterson Hall
Chapel Hill, NC 27599-3175

(919) 962-1919
fax: (919) 962-1799
gerig@cs.unc.edu

Ioannis A. Kakadiaris
Assistant Professor
Department of Computer Science
University of Houston
516 PGH, MS CSC 3475
4800 Calhoun
Houston, TX 77204-3475, USA

Tel: (713) 743-1255
fax: (713) 743 1250
ioannisk@uh.edu

Torsten Moeller
Assistant Professor
School of Computing Science
8888 University St.
Simon Fraser University
Burnaby, B.C. V5A 1S6
Canada

(604) 291-3774
torsten@cs.sfu.ca

Raghu Machiraju
Assistant Professor
Department of Computer and Information Science
The Ohio State University
395 Dreese Laboratories
2014 Neil Avenue
Columbus, OH 43210-1277, USA

(614) 292-6730
fax: (614) 292-2911
raghu@cis.ohio-state.edu

Ross T. Whitaker
Assistant Professor
School of Computing
4540 Merrill Engineering Building
The University of Utah
Salt Lake City, UT 84112-9205

(801) 581-8224
fax: (801) 581-5843
whitaker@cs.utah.edu

Terry S. Yoo
Computer Scientist
National Library of Medicine
National Institutes of Health
8600 Rockville Pike
Bethesda, MD 20894, USA

(301) 435-3268
fax: (301) 402-4080
yoo@nlm.nih.gov

Speaker Biosketches

Guido Gerig is a Taylor Grandy Professor of Computer Science and Psychiatry at the University of North Carolina at Chapel Hill, where he studies methods for segmentation and visualization of multidimensional medical data as well as the statistical analysis of 3D shape. He holds a MSc. in Natural Sciences, and both a Ph.D. and a PD in Electrical Engineering and Computer Science all from ETH Zurich. Guido serves on several program committees and editorial boards for medical image analysis journals and conferences, and he was awarded the UNC Computer Science Students Association Teaching Award in 1999.

Ioannis A. Kakadiaris is an Assistant Professor in the Computer Science Department of the University of Houston. He received the Ptychion (B.S.) (1989) in Physics from the University of Athens, an M.Sc. (1991) in Computer Science from the Northeastern University, and his PhD (1997) in Computer Science from the University of Pennsylvania. Dr. Kakadiaris is the Director of the Visual Computing Lab and his expertise includes physics-based modeling, computer vision, computer graphics and medical imaging. He received the NSF Faculty Early Career Award for his proposal, "An Integrated Framework for Data-Driven Representations and Algorithms in Visual Computing" in 2000.

Raghu Machiraju is an Assistant Professor of Computer and Information Science at the Ohio State University. His research interests include visualization, graphics, image analysis and high performance computing. He obtained his Doctorate from the Ohio State University in 1996. Previously he served as an Assistant Professor of Computer Science at Mississippi State and as a research scientist at the NSF ERC for Computational Field Simulation. He received the NSF Faculty Early Career Award for his proposal, "On the Assessment of Volume Rendering Algorithms in Visual Computing" in 1998. Raghu has participated in related tutorials at IEEE Vis 2000.

Torsten Moeller received his PhD in Computer and Information Science from Ohio State University in June 1999. He received a Vordiplom (BSc) in mathematical computer science from Humboldt University of Berlin, Germany. He is currently an assistant professor at the School Of Computing Science at Simon Fraser University, where he is co-director of the Graphics, Usability and Visualization Lab. His research interests include the fields of Scientific Visualization and Computer Graphics. He is especially interested in interactive and accurate volume rendering methods for regular and irregular data.

Ross Whitaker is currently an assistant professor at the University of Utah, Department of Computer Science. His research interests include: computer vision, image processing, medical imaging, and computer graphics/visualization. He received his B.S. degree in Electrical Engineering and Engineering Physics from Princeton University in 1986 and his Ph.D. in Computer Science from the University of North Carolina at Chapel Hill in 1993. Previously he has been a research scientist in the User Interaction and Visualization Group at the European Computer-Industry Research Centre in Munich, Germany and an assistant professor of Electrical and Computer Engineering at the University of Tennessee.

Terry S. Yoo is a Computer Scientist in the Office of High Performance Computing and Communications, National Library of Medicine, NIH, where he explores the processing and visualizing of 3D medical data, interactive 3D graphics, and computational geometry. Previously as a professor of Radiology, he managed a research program in Interventional MRI with the University of Mississippi. Terry holds an A.B. in Biology from Harvard, and a M.S. and Ph.D. in Computer Science from UNC Chapel Hill. He has been a speaker and organizer of successful courses on medical visualization and 3D image processing (SIGGRAPH 1993, 1994, 1998, Vis 2000).

Contents

Summary ..... **i**

Course Description ……...**ii**

Speaker Information ..... **vi**

Speaker Biographies ……...**vii**

Contents.....**viii**

Chapter 1: Digital Filtering and Sampling **1**

Chapter 2: Evaluation and Design of Filters..... **39**

Paper: "Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering," T.Moeller, R. Machiraju, K. Mueller, R. Yagel **80**

Paper: "A Comparison of Normal Estimation Schemes," T.Moeller, R. Machiraju, K. Mueller, R. Yagel...……..**89**

Paper: "Design of Accurate and Smooth Filters for Function and Derivative Reconstruction," T.Moeller, K. Mueller, Y. Kurzion, R. Machiraju, R. Yagel **97**

Chapter 4: Feature Detection for Volume Rendering..... **107**

Paper: "The Transfer Function Bake-Off," H. Pfister, B. Lorensen, C. Bajaj, G.Kindlmann, W. Schroeder, L. Sobierajski-Avilla, K. Martin, R. Machiraju, J. Lee …….…..**132**

Paper: "Semi-Automatic Generation of Transfer Functions," G. Kindlmann, J. W. Durkin……..**138**

Paper: "Design Galleries: A general Approach to Setting Parameters for Computer Graphics and Animation," J. Marks et al. ……**146**

Chapter 5 Volume Segmentation.....176

Chapter 6 3D Model Based Segmentation 192

Paper: "Three-Dimensional Model-based Segmentation, " A. Kelemen, G. Szekely, G.Gerig……...**219**

Paper: "Three-Dimensional Segmentation using Elastic Deformation, " G.Szekely, A. Kelemen, C. Brechbuhler,G.Gerig …….....**232**

Paper: "Parametrization of closed surfaces for 3-D shape description," Brechbühler, G. Gerig and O. Kübler **……...…..248**

Chapter 7 Isosurfaces and Level-Set Surface Models 268

Digital Filtering and Sampling Fundamentals

Terry S. Yoo
Office of High Performance Computing and Communications
The National Library of Medicine, National Institutes of Health

1 Introduction

Digital sampling theory is intrinsic to computer graphics, image processing, and visualization. Representing complex shapes and flows using pixels, voxels, and other discrete atomic data types creates repercussions that pervade all of volume graphics. Familiar artifacts such as aliasing can only be understood through digital sampling and filtering theory. Likewise, effective reconstruction, the inverse of sampling, requires a clear understanding of digital sampling theory.

This chapter is a review or introduction (depending on your background) of the digital theories that underlie sampling and filtering. This material will be covered from a practical viewpoint: what should we know about the basic principles of digital sampling theory? What are the repercussions of which we should be aware? How do these theoretical results affect our output images? What can you do about it, and where are the pitfalls that should be avoided? This presentation is intended as a cursory introduction or review only. We will not explore proofs or rigorous mathematics associated with the presented theorems or tools.

For example, the presentation of the Fourier transform will be far from exhaustive, not even listing all of the important mathematical properties of the transform, but rather just those properties that are relevant to volume graphics. We also will not be presenting any implementations. However, we will provide pointers to references for more rigorous treatments of this material, as well as forums, journals, and venues where current research may be found.

2 Images

In this course, we are principally concerned with volume images. A volume image, ϕ, is a function or mapping from $\Re^3$ to $\Re^n$ where n = 1 for the typical, scalar–valued volume. More precisely:

$$\phi : U \mapsto \Re \text{ and } U \subset \Re^3, \tag{1}$$

where U is the domain of the volume. The image ϕ will also frequently be written as a function $\phi(x, y, z)$.

Throughout this introductory chapter, however, many example images will be presented either as projections from a 1-D or a 2-D domain. Most exercises in sampling theory are more easily examined in 1D (at least in print), and generalized to higher dimensions. Wherever we explore images of fewer than three dimensions, we will use the function notation (*i.e.*, $\phi(x)$ or $\phi(x, y)$).

We distinguish discrete, sampled digital images from the description of images as contiuous mappings of functions in real space. If F is a discrete sampling of ϕ then we can say that

$$F_{i,j,k} = \phi(x_i, y_i, z_i). \tag{2}$$

We also use notation to distinguish between discrete and continous differential operators. To take the approximate partial derivative in the x direction of a discretely sampled image we can say that

$$\phi_x(x_i, y_i, z_i) = \left.\frac{\partial \phi}{\partial x}\right|_{x_i, y_i, z_i} \approx \delta_x f_{i,j,k}, \tag{3}$$

where

$$\delta_x f_{i,j,k} \equiv \frac{f_{i+1,j,k} - f_{i-1,j,k}}{x_{i+1} - x_{i-1}} = \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2h}, \tag{4}$$

where h is the grid spacing, which we normally assume to be 1–pixel width. The value $\delta_x f_{i,j,k}$ is an approximation of the instantaneous partial derivative of $\phi(x_i, y_i, z_i)$. taken at (x_i, y_i, z_i) in the x direction. Equation 4 is the method of *central differences* commonly used to take derivatives of uniformly, discretely sampled datasets.

3 Linear Filtering

When was the last time you used the convolution integral? If you've run a blurring function, taken a derivative, done some edge enhancement, or other simple operation, it is likely that you've used convolution (either the continuous integral form or the discrete iterative summation). Filters are used to make measurements, detect features, smooth noisy signals, and deconvolve operations from previous operations (e.g., deblurring the known optical artifacts from a telescope). The effect of the filter depends not on the operation of convolution, but rather on the nature of input filter kernel.

3.1 Convolution

Convolution, denoted with the operator $\otimes$ can be described in 1-D as a continuous operation applying a filter kernel $h(x)$ (itself an image) to an image $\phi(x)$ using the following integral form:

$$\phi(x) \otimes h(x) = \int_{-\infty}^{\infty} \phi(x - \tau) h(\tau)\, d\tau \tag{5}$$

Note that the expression $\phi(x) \otimes h(x)$ itself describes an image mapping. Thus the following equality holds:

$$\phi(x) \otimes h(x) = (\phi \otimes h)(x) \tag{6}$$

The expressions $\phi(x) \otimes h(x)$ and $(\phi \otimes h)(x)$ are used interchangably, with the position index x often omitted to help streamline and clarify the notation especially when higher dimensional (*e.g.*, 3-D) images are involved. In the practice of digital image processing, $h(x)$ typically does not have infinite extent, nor is it infinitely dense; however, when working in the continuous domain of digital filtering theory, functions or signals that can be considered infinitely wide (thus avoiding the complications of truncation) are often more convenient. We will elaborate later on discrete, bandlimited functions in the discussion on sampling.

We can generalize the 1-D convolution operation to 2-D and 3-D images. Thus, in 2-D, convolution becomes:

$$\phi(x, y) \otimes h(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(x - \tau, y - \nu) h(\tau, \nu)\, d\nu\, d\tau \tag{7}$$

Similarly, convolution in 3-D is expressed as:

$$\phi(x, y, z) \otimes h(x, y, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(x - \tau, y - \nu, z - \omega) h(\tau, \nu, \omega)\, d\omega\, d\nu\, d\tau \tag{8}$$

3.1.1 Example - Convolution as Noise Reduction (1-D)

One of the most common uses of convolution is as a filter for the suppression of noise in an image. Consider the Gaussian function as a smoothing filter kernel. If convolved with a relatively noisy signal, the result is an output image that locally averages the image intensities. The output image typically has reduced noise at the cost of some "sharpness" of the original image. That is, the resulting image has less perceived noise, but also has blurred features.

Figure 3.1.1 shows a 1-D step discontinuity, or edge. The figure ranges continuously from $x = 0$ to $x = 255$ with a step function at $x = 128$. The signal to noise ratio is approximately 4:1.

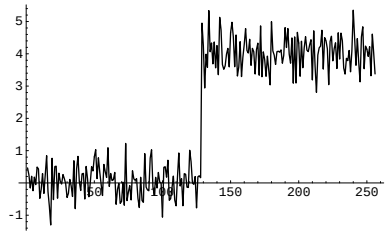
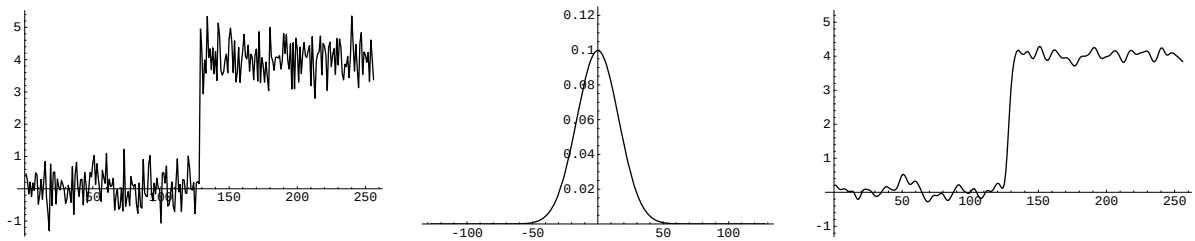


Figure 1: A 1-D Noisy Step Edge: An example input signal $\phi(x)$.



(a) Input $\phi(x)$

(b) a 1-D Gausssian kernel, $g(x, \sigma)$ where $\sigma = 16$

(c) output of $\phi(x) \otimes g(x, 16)$

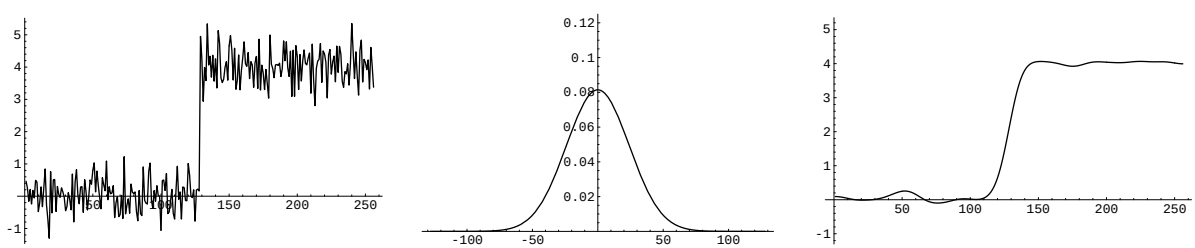
Figure 2: Gausssian filtering of input $\phi(x)$ where $\sigma = 16$

Figures 2, 3, and 4, show a progression of filtered versions of the input from figure 3.1.1 with Gaussian filter kernels of increasing aperture or width. Notice how the perceived noise in the signal decreases with aperture and the consequent softening of the gradient information at the discontinuity. This trade of detail and resolution for noise reduction is one of the many considerations in the design of linear filter systems.

3.2 Properties of the Convolution Operation

The properties of a linear filtering operation often depend on the nature of the filter kernel. For instance, the "shape" of the kernel will determine whether the operation remains invariant with respect to rotation. However, there are many inherent properties of the convolution operation. The convolution operation has many useful properties. It is linear, commutative, associative, and distributive over addition.

In mathematical terms, convolution is commutative and reflexive:

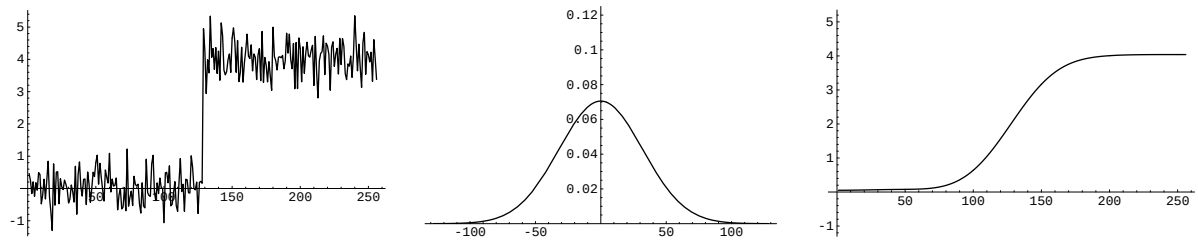


(a) Input $\phi(x)$

(b) a 1-D Gausssian kernel, $g(x, \sigma)$ where $\sigma = 24$

(c) output of $\phi(x) \otimes g(x, 24)$

Figure 3: Gausssian filtering of input $\phi(x)$ where $\sigma = 24$



(a) Input $\phi(x)$

(b) a 1-D Gausssian kernel, $g(x, \sigma)$ where $\sigma = 32$

(c) output of $\phi(x) \otimes g(x, 32)$

Figure 4: Gausssian filtering of input $\phi(x)$ where $\sigma = 32$

$$p \otimes q = q \otimes p \tag{9}$$

Convolution is also associative:

$$(p \otimes q) \otimes r = p \otimes (q \otimes r) \tag{10}$$

Convolution is distributive over addition:

$$p \otimes (q + r) = p \otimes q + p \otimes r \tag{11}$$

These combined properties create the justification for using convolution as the principal operation in linear filtering.

3.2.1 2-D Example - Convolution as Feature Detection and Enhancement

Linear filter kernels can be selected to detect and enhance features. For example, if a difference of two Gaussian functions is used as a filter kernel, edge information can be both detected and enhanced. The difference of an image filtered with two Gaussians of different apertures is a process called "unsharp masking." The result is an *edge-enhanced* image.

Specifically, given a 2-D input image generating function $\phi(x, y)$ and two Gaussian filter kernels of differing aperture, $g(x, y, \sigma_1)$ and $g(x, y, \sigma_2)$ where $\sigma_1 < \sigma_2$, an edge enhanced image $\phi\prime(x, y)$ can be formed by a linear combination of the two filters. That is:

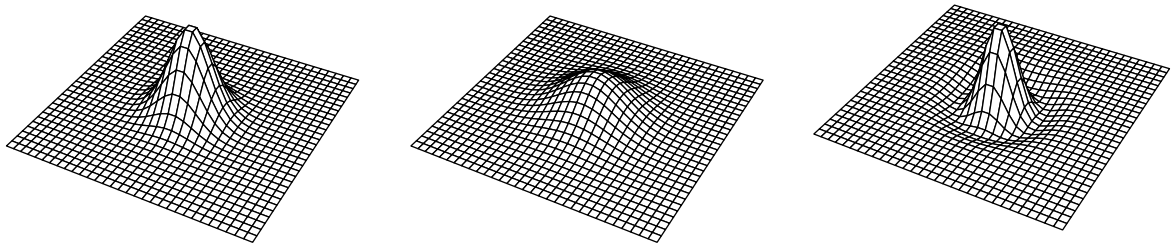
$$\phi'(x, y) = \phi(x, y) \otimes (g(x, y, \sigma_1) - g(x, y, \sigma_2)) \tag{12}$$

From equation 11 it follows that $\phi\prime(x, y)$ is also the subtraction of two filtered images.

$$\phi'(x, y) = (\phi(x, y) \otimes (g(x, y, \sigma_1)) - (\phi(x, y) \otimes g(x, y, \sigma_2))) \tag{13}$$

In other words, the difference of two "blurred" versions of the same image yields a version of the same image where edge information is accentuated.

A specific use of this technique is the detection of edges. If one applies two filter kernels that have unit gain (they do not amplify the signal) of differing widths and subtracts the results, the result is an image where regions of continuous intensity have a zero value, areas near boundaries are strongly negative or

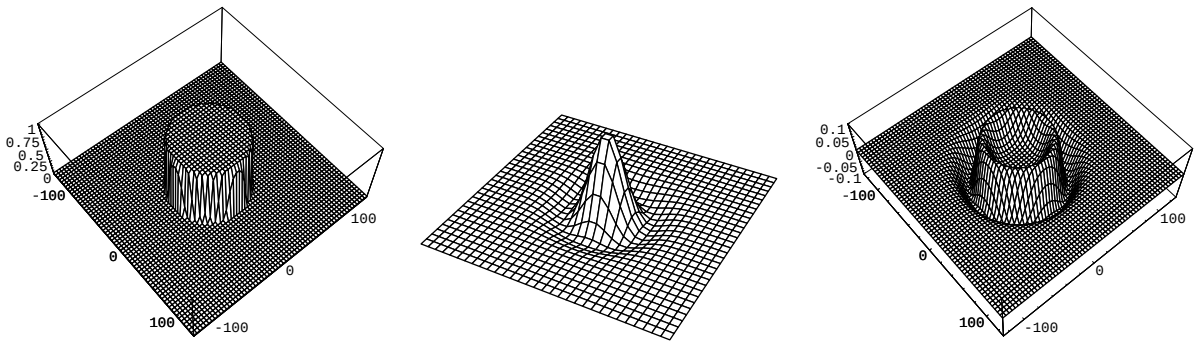


(a) 2-D Gaussian kernel $g(x, y, \sigma_1)$ with $\sigma_1 = 8$

(b) 2-D Gaussian kernel $g(x, y, \sigma_2)$ with $\sigma_2 = 12$

(c) $g(x, y, \sigma_1) - g(x, y, \sigma_2)$ where $\sigma_1 = 8$ and $\sigma_2 = 12$

Figure 5: a Difference of Gausssians filter kernel (depicted in 3-D as a height field).

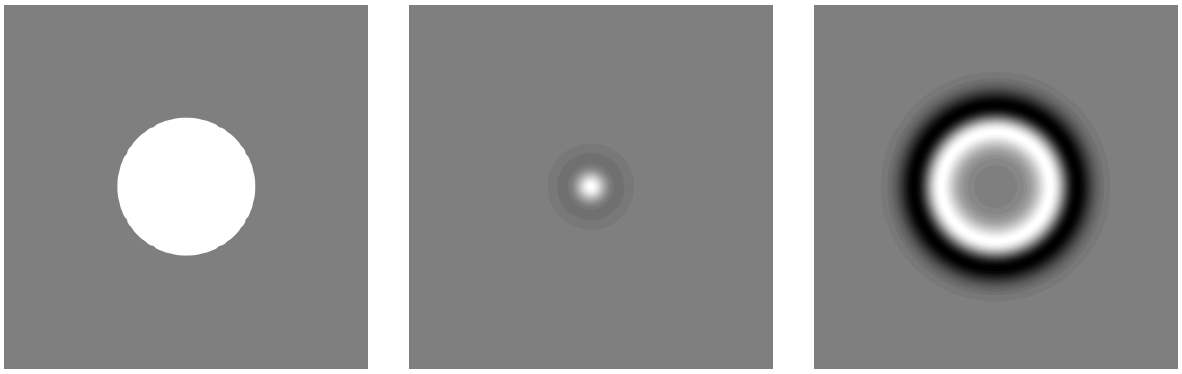


(a) 2-D Input $\phi(x, y)$ with $\sigma_1 = 16$

(b) $g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32)$

(c) $\phi(x, y) \otimes (g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32))$

Figure 6: a 2-D image filtered by a Difference of Gausssians (depicted in 3-D as a height field).



(a) 2-D Input $\phi(x, y)$ with $\sigma_1 = 16$

(b) $g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32)$

(c) $\phi(x, y) \otimes (g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32))$

Figure 7: a 2-D image filtered by a Difference of Gausssians (depicted as a 2-D greyscale density field).

(a) 2-D Input $\phi(x, y)$

(b) $2g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32)$

(c) $\phi(x, y) \otimes (g(x, y, \sigma_1 = 16) - g(x, y, \sigma_2 = 32))$

Figure 8: a Difference of Gausssians filter used for contrast enhancement (depicted as a 2-D greyscale density field).

strongly positive (depending on which side of a boundary you are on) and the boundaries itself denoted by the closed curve of *zero-crossings*, those points where the function crosses from positive to negative values. Figure 5 shows such a differnce of Gaussian kernel as a 3-D projection where intensity is plotted as a height field above an x-y plane. Figure 6 is the same kernel show in figure 5 applied to a circular pulse function. These views are also represented as a 3-D projection of intensity as a height field. Figure 7 is the same functions as in figure 6 depicted as a 2-D intensity field. The areas of contiguous intensity result in a flat signal value of approximately zero. At the boundaries, there is a rise in the output intensity followed by a negative dip in intensity values. The zero crossing represents the boundary between the foreground intensity and the background.

With a slight adjustment of the filter kernel, applying coefficients to each of the Gaussian kernels, we can modify the edge detection kernel into a contrast enhancement kernel. Figure **??** shows an adjusted filter kernel with a combined gain of 1; that is, the difference of Gaussians will return the same value in areas of even intensity, but activates near boundaries to create a band of intensities that exaggerate the discontinuity.

3.3 Differentiation - Measurement Through Linear Filtering

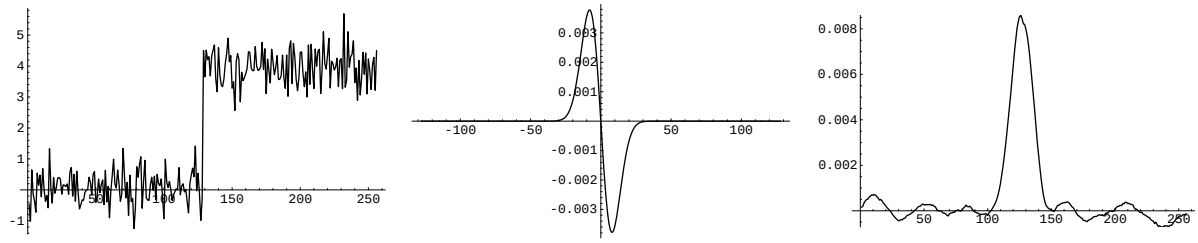
Differentiation is seldom portrayed as a convolution; however, the measurement of the instantaneous rate of change of intensities is in fact subject to all of the attributes of the convolution operation. The derivative operator is a convolution kernel that is infinitesimal in width, but is applied everywhere. Put another way, we can explicitly denote differentiation as convolution using the $\otimes$ operator as seen in the following example using the partial derivative in the x direction:

$$\frac{\partial}{\partial x}\phi = \frac{\partial}{\partial x} \otimes \phi \tag{14}$$

Differentiation is susceptible to noise, so it is often necessary to regularize a noisy dataset before computing its derivative. Therefore, derivatives are often taken of smoothed functions. It follows from the associative and commutative properties of convolution that:

$$h(x, y, z) \otimes \frac{\partial}{\partial x}\phi(x, y, z) = \frac{\partial}{\partial x}h(x, y, z) \otimes \phi(x, y, z) \tag{15}$$

In other words, the derivative of a function ϕ convolved by a filter kernel h is equivalent to convolving the



(a) 1-D Input $\phi(x)$ (b) $\frac{\partial}{\partial x}g(x,\sigma)$ where $\sigma = 3$ (c) $\phi(x) \otimes \frac{\partial}{\partial x}g(x,\sigma)$

Figure 9: Taking derivatives of a noisy input by convolution with the derivative of a filter kernel.

derivative of the h with ϕ. This suggests that one of the easiest ways to compute the derivative of a volume dataset is through convolution with the derivative of some smoothing kernel.

3.3.1 Example - Convolution as Measurement (1-D)

To show how derivatives can be approximated through convolution, we suggest the use of the Gaussian $g(x)$ as a regularizing (smoothing) filter kernel. Its infinitely differentiable properties make it attractive as a measurement and scaling function.

Figure 9 depicts a noisy 1-D input signal for which derivative information is desired. When convolved with the derivative of a Gaussian, the resulting output reports the derivative of a smoothed function. By the commutative and associative properties of convolution, it is also a smoothed version of the derivative of the input. This techniques for differentiating difficult functions can be extended to higher orders and is quite commonplace in the practice of volume image processing.

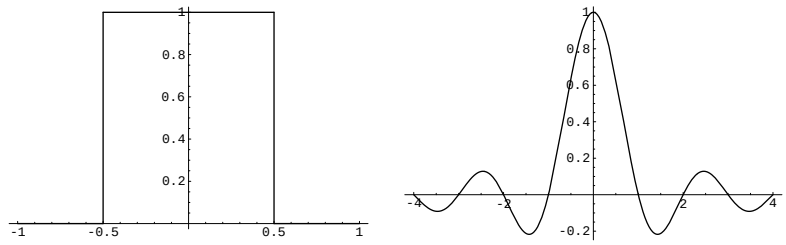
3.4 Convolution of Discretely Sampled Data

Convolution of discretely sampled data is similar to the continuous form, except that discrete summation is substituted for integration. Also, since discrete filter kernels seldom have infinite extent, the limits of the summations do not range from $-\infty$ to ∞ but are rather constrained to the size of the filter kernel. In 2-D, a discrete convolution looks like

$$P_{x,y} \otimes Q_{x,y} = \sum_{i}^{domain_x[Q]} \sum_{j}^{domain_y[Q]} P_{x-i,y-j} Q_{i,j} \tag{16}$$

The discrete version of convolution shares all of the attributes of its continous cousin. Convolution of discretely sampled data is commutative, associative, and distributive over addtion. Among its many uses, convolution can be used to smooth noisy data, detect or enhance edgeness, or make measurements of the image. Derivatives of sampled digital images can be approximated through discrete convolution using a well-chosen kernel. An example of discrete convolution for differentiation is the *central differences operation* shown in equation 4. From the perspective of linear filtering of discrete data, all such operations can now be seen to be either convolutions or adaptations of a convolution operation. For more information on differentiation of discrete image data through convolution, see the attached paper.

The primary caveat (as in *caveat emptor*) regarding discrete convolution is that all such discrete operations are subject to error induced by the effects of sampling, a class of artifacts known as *aliasing*. How much aliasing? We'll explore that later. Before we can appreciate the problems associated with sampling, we must first understand the nature of the sampling operation and its effects on the information embedded within the image. For that, we use an image analysis and manipulation tool called the Fourier Transform.



(a) The nearest neighbor interpolant (Box)

(b) Frequency Space nearest neighbor interpolant (Sinc)

Figure 10: The Box Filter and its Fourier Transform, the Sinc Function.

4 The Fourier Transform

One of the most important tools for understanding images (1-D, 2-D, 3-D or even higher) is the Fourier Transform. The Fourier transform of an image decomposes the input signal into a series sum of sinusoidal functions. The representation of an image as a series of frequencies is a computationally useful means of viewing and manipulating image data. The relationship between an input function and its Fourier transform is governed by the following equation:

$$\mathcal{F}(\phi(x)) = \int_{-\infty}^{\infty} \phi(x) e^{-i2\pi x\nu} dx = \Phi(\nu) \tag{17}$$

This relationship maps the domain and range of space, x and intensity, $\phi(x)$ to a new space where the domain and range are frequency, ν, and magnitude + phase, $\Phi(\nu)$, respectively. The transform space is often called *frequency space*, and the transformed input is described as the frequency space representation. As a transform, the process is also reversible. The inverse Fourier transform is:

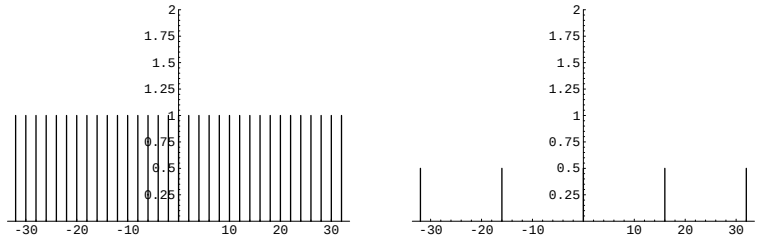
$$\mathcal{F}^{-1}(\Phi(\nu)) = \int_{-\infty}^{\infty} \Phi(\nu) e^{i2\pi x\nu} d\nu = \phi(x) \tag{18}$$

There are some simple generalizations of these formulas to 2-D, 3-D, and higher dimensions. Among its most important attributes, the Fourier transform is separable in each of the orthonnormal basis dimensions. One simply recasts the transform equation with additional independent variables of integration and within the integrand to extend it to higher dimensions. A complete description and exhaustive derivation of the Fourier transform and its properties is beyond the scope of this tutorial. However, there are some key properties of the frequency space and the transform that creates it that we will discuss to illuminate the problems of sampling in computer graphics.

4.1 Some Properties of the Fourier Transforms

Lets begin by looking at some functions commonly used in manipulating images. The four functions of interest as part of this discussion are the box filter (a square pulse, used as a nearest neighbor interpolant in reconstruction), the comb function (used to discretely sample data in the image domain), the pyramid function (a pyramid shaped filter used for linear interpolation in reconstruction), and the Gaussian (a good all purpose filter, except for its infinite extent).

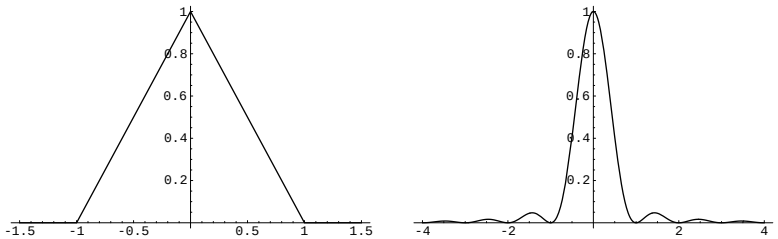
Figures 10, 11, 12, and 13 show the box, comb, pyramid, and Gaussian functions represented in image space along with their Fourier transforms, represented in frequency space. By studying this array of functions, one pair of important characteristics of the relationships between image space and freqency space become clear.



(a) Sampling Comb

(b) Frequency Space Sampling Function

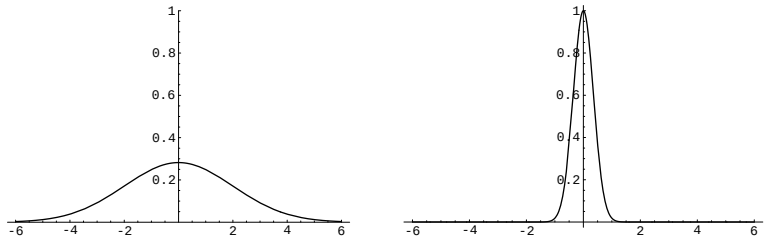
Figure 11: The Sampling Function and its Fourier transform.



(a) Linear Interpolating Pyramid Function

(b) The Fourier Transform of the Pyramid

Figure 12: The Linear Interpolation Function and its Fourier transform.



(a) Image Space Gaussian

(b) Frequency Space Gaussian

Figure 13: The Gaussian Function and its Fourier transform.

Spatial Scaling $\mapsto$ Inverse Frequency Scale Change An increase in the magnitude (height) of the function in the spatial domain leads to an inversely proportional broadening of the function in the frequency domain as well as an inversely proportional decrease of the height of the function in the frequency domain.

Frequency Scaling $\mapsto$ Inverse Spatial Scale Change The converse of the consequences of spatial scaling are true for frequency scaling. An increase in the magnitude (height) of the function in the frequency domain leads to an inversely proportional broadening of the function in the spatial domain as well as an inversely proportional decrease of the height of the function in the spatial domain.

Mathematically speaking:

$$\mathcal{F}(\phi(ax)) = \frac{1}{|a|}\Phi\left(\frac{\nu}{a}\right) \tag{19}$$

and

$$\mathcal{F}^{-1}(\Phi(a\nu)) = \frac{1}{|a|}\phi\left(\frac{\nu}{a}\right) \tag{20}$$

Conversely:

$$\mathcal{F}\frac{1}{|a|}\phi\left(\frac{\nu}{a}\right)(\phi(ax)) = \Phi(a\nu) \tag{21}$$

and

$$\mathcal{F}^{-1}\frac{1}{|a|}\Phi\left(\frac{\nu}{a}\right) = \phi(ax) \tag{22}$$

There are many other properties of the Fourier transform associated with its symmetry and its use of complex numbers to describe complicated behaviors of signals. We have neither the space or time to explore them all (For a more complete treatment, see the excellent text by E. O. Brigham.). However, the two related properties of spatial/frequency scaling mapping onto inverse scale change have profound impact on image analysis and scene generation. The emphasize them and address their importance in the following sections.

4.2 The Convolution Theorem

The convolution theorem relates frequency space to linear filtering. This is perhaps the most important theorem for linear system analysis. Simply, convolution in image space between a filter kernel and an input translates to multiplication of their Fourier transforms in frequency space. That is, given two functions $\phi(x)$ and $h(x)$ (*e.g.*, an input ϕ and a filter kernel h), and their Fourier transforms, $\Phi(\nu)$ and $H(\nu)$ the following relation holds:

$$\mathcal{F}(\phi(x) \otimes h(x)) = \Phi(\nu)H(\nu) \tag{23}$$

As with most relationships across the transform, the converse can also be easily generated.

$$\mathcal{F}^{-1}(\Phi(\nu) \otimes H(\nu)) = \phi(x)h(x) \tag{24}$$

Succintly, practitioners often say, *convolution in one domain is multiplication in the other domain.* Given this theorem, it is easy to conceive of why convolution is distributive over addition, symmetric, refexive, commutative, etc. It shares most of the properties of multiplication.

Linear filtering under these circumstances can now be considered in a different light. It can be changed from an integral with infinite extent in image space (a difficult continuous operation to implement with a digital computer) to a multiplication in frequency space, with only two Fourier transforms and one inverse Fourier transform to be calculated.

$$\phi(x) * h(x) = \mathcal{F}^{-1}(\mathcal{F}(\phi(x))\mathcal{F}(h(x))) \tag{25}$$

Depending on the size of both the filter kernel and in the input signal, casting the convolution problem in the frequency domain often makes difficult iterative problems more tractable. The Fourier transform itself is an integral with infinite extent. A discrete implementation that can be implemented on a digital computer is required to translate the Fourier transform from a useful mathematical abstraction to a practical tool.

4.3 The Discrete Fourier transform

How do we parley all of this frequency space theory into practical use? A typical digital image does not have infinite extent! Nor can it be sampled at infinitesimal rates, since it has already been sampled at a particular discrete sampling rate. The application of the Fourier transform to discretely sampled data results in the *Discrete Fourier Transform.*

Sinusoids are periodic, and for it to be computable, a decomposition of an input signal or image must also be considered periodic. To create a periodic signal, a discrete image with n uniformly spaced samples is *wrapped* so that samples 0 through n-1 index the data array exactly, but sample n "wraps" to 0. In fact, if n is the number of uniform samples, the effective index x_{eff} of a location x is, $x_{eff} = x \bmod n$. This can be applied to any dimension (*i.e.*, x, y, or z). This concept of a discrete image as a periodic signal can also be conceived as an infinite concatenation of the image to itself.

This repetition allows us to treat digital images as periodic signals. Moreover, it allows us to represent them as the sum of a finite (countable and finite) sinusoidal functions. The maximum representable wavelength is governed by dimensions and sampling rate of the input image. Consider a discrete input image F with n samples in each dimension. If the input image is considered to be a single period, we can map the domain of the input image to the interval from 0 to 2π (or from $-\pi$ to π). The dimension of each voxel then becomes $\frac{2\pi}{n}$.

The result is the discretization of the Fourier transform, with the sampled input function F transformed to a sampled frequency space representation representing the sum of n separate frequencies. The discrete inverse Fourier transform of the frequency space representation reproduces the original sampled data without loss of fidelity to the sampled signal. There may be a loss in the process of sampling. The implications to sampling become clear on an examination of the frequency space representation.

What remains is all the rest of sampling error, aliasing artifact, and digital fitler design. We cover some of these issues in the accompanying papers and provide references for extended reading on the subject.

5 Summary and Discussion

Filtering and frequency space are essential to the understanding of digital sampling theory. As we have seen in the convolution theorem, linear filtering and Fourier analysis are tightly linked, with profound implications in the understanding and manipulation of digital data. Aliasing and other sampling errors are directly related to the properties of the Fourier transform and the behavior of filters and input signals when discretized.

Frequency space arises everywhere, so a command of the basics is very important. The source of many volume images is in fact a manipulation of frequency space. The principals of image reconstruction in X-ray CT scanners and Magentic Resonance Imaging (MRI) are all based on uses of frequency space. Iterative methods, filtered back projection, etc. all have important roots in Fourier analysis. The resulting algorithms, even if they do not use a Fourier slice projection approach are often best understood in frequency space.

Basic control of these tools will help you get a handle on your data. Sampling, aliasing, level of detail, perspective texture correction (*e.g.*, "mip mapping"), and other common graphics techniques are only to be fully understood with through the window of frequency spaces.

Digital Filtering References

Terry S. Yoo
The National Library of Medicine
National Institutes of Health

1. Brigham, E.O. 1974. The fast fourier transform. Prentice Hall.

2. Cabral, B., N. Cam, and J. Foran. 1994. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. Proc. 1994 ACM Symp. On Vol. Vis. ACM. 91-98.

3. Castleman, K.R. 1979. Digital image processing. Prentice-Hall Inc. USA

4. Chellappa, R. and A. Jain. 1993. Markov random fields: Theory and application. Academic Press.

5. Duda, R.O. and P.E. Hart. 1973. Pattern classification and scene analysis. John Wiley and Sons.

6. Drebin, R.A., L.C. Carpenter, and P. Hanrahan. 1988. Volume rendering. Computer Graphics. 22(4). August 1998

7. Drebin, R.A., L.C. Carpenter, and P. Hanrahan. 1989. Methods and apparatus for imaging volume data with shading. U.S. Patent number 4835712, May 1989.

8. Geman, S. and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Analysis Machine Intell., 6, 721-741.

9. Gibson, S. 1998. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In Proc. MICCAI ’98, Lecture Notes in Computer Science. 1496. Springer-Verlag. 888-898.

10. Gibson, S. 1998. Using distance maps for accurate surface representation in sampled volumes. Proc. 1998 Symp. On Vol. Vis. ACM. 23-30.

11. Glassner, A. 1995. Principles of digital image synthesis. Morgan Kaufman.

12. Gomes, J., and L. Vehlo. 1997. Image processing for computer graphics. Springer Verlag.

13. Hanrahan, P. 1990. Three pass affine transforms for volume rendering. Computer Graphics. 24(5). (November 1990). 71-78.

14. Jain, A.K. 1989. Fundamentals of digital image processing. Prentice Hall.

15. Kajiya, J. T. and B. P. Von Herzen. 1984. Ray tracing volume densities. Computer Graphics. 18(3) 165-174. July 1984.

16. Krueger, W. 1990. Volume rendering and data feature enhancement. Computer Graphics. 24(5). 21-26.

17. Levoy, M.S. 1988. Volume rendering: display of surfaces from volume data. IEEE Computer Graphics and Applications. 8(3). May 1988.

18. Lichtenbelt, B. R. Crane, and S. Naqvi. 1998. Introduction to volume rendering. Prentice Hall.

19. Malzbender, T. 1993. Fourier volume rendering. ACM Trans. On Graphics. 12(3). 233-250.

20. Shannon, C.E. 1949. Communication in the presence of noise. Proceedings IRE, 37.

21. Smith, A. R. 1983. Digital filtering tutorial for computer graphics. Technical memo number 27. Lucasfilm, Ltd. (revised)

22. Totsuka, T. and M.S. Levoy. 1993. Frequency domain volume rendering. Proc. SIGGRAPH 93. ACM SIGGRAPH. 271-278.

23. Westover, L.A. 1989. Interactive volume rendering. Proceedings of the Chapel hill workshop on volume visualization. May 1989. 9-16.

24. Westover, L.A. 1990. Footprint evaluation for volume rendering. Computer Graphics. 24(4). August 1990. 367-376.

25. Westover, L.A. 1991. Splatting: a parallel, feed-forward volume rendering algorithm. Ph.D. dissertation, the University of North Carolina at Chapel Hill. (UNC Technical report TR91-029). August 1991.

Image Processing Fundamentals

•***Terry S. Yoo***

the National Library of Medicine, NIH

Do What? How?

- Find the surface normal to the embedded surface?
- Locate an object within the volume?
- Take the derivative of a kidney?

- How do I evaluate the error introduced by sampling?
- What is aliasing, really?
- How does this all work together in volume graphics?

Image Basics

- Given a discrete volume dataset, *vol*[x][y][z]
- Imagine a volume generating function, f, such that for any particular point $p_i = (x_i,y_i,z_i)$:

$$f(x_i,y_i,z_i) = vol[x_i][y_i][z_i]$$

- Must reconstruct f(x,y,z) from *vol*[x][y][z].
- Can make measurements of f(x,y,z), (e.g.,derivatives)
- HOW?

Convolution

- Sampling.
- Interpolation.
- Reconstruction.
- Low pass filtering.
- Noise suppression.
- Edge enhancement.
- Derivative measurement.
- Linear scale space analysis.

Convolution

Convolution integral

$$h(x) \otimes I(x) = \int_{-\infty}^{\infty} h(\tau) I(x - \tau)\, d\tau$$

Convolution & the Gaussian

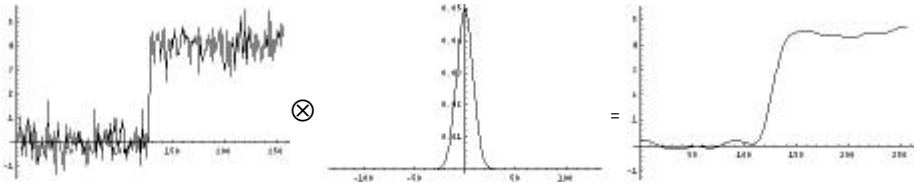
Convolution integral

$$h(x) \otimes I(x) = \int_{-\infty}^{\infty} h(\tau) I(x - \tau)\, d\tau$$

Gaussian as a convolution kernel
(with spatial scale parameter, σ)

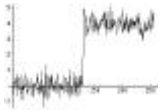
$$h(x) = G(\sigma, x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{x^2}{2\sigma^2}}$$

Smoothing - noise filtering

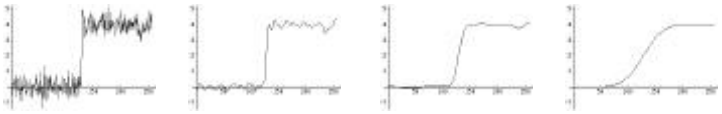


Input, $\phi(x)$ $\otimes$ Kernel, $G(x,\sigma)$ $= \phi(x) \otimes G(x,\sigma)$

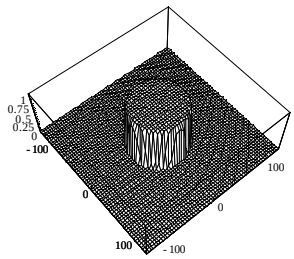
Linear Scale Space



$$\mu_I(x|\sigma) = G(\sigma,x) \otimes I(x) = \int_{-\infty}^{\infty} G(\sigma,\tau) I(x-\tau)\, d\tau$$

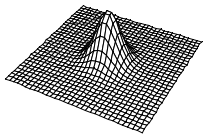


Convolution in 2D



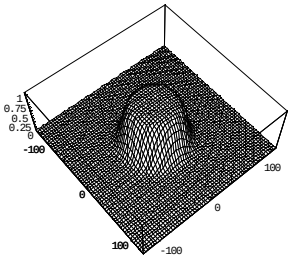
Input, $\phi(p)$

$\otimes$



Kernel, $G(p,\sigma)$

$= \phi(p) \otimes G(p,\sigma)$



Properties of the Convolution Operator

<u>Property:</u>

- Commutative.
- Associative
- Distributive over addition.

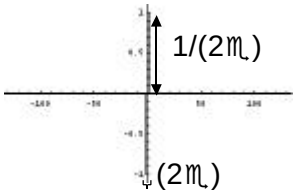
<u>Mathematically:</u>

- $f \otimes h = h \otimes f$
- $(f \otimes h) \otimes g = f \otimes (h \otimes g)$
- $f \otimes (h + g) = (f \otimes h) + (f \otimes g)$

Convolution & differentiation

- Problem: Real-world data are seldom accompanied with continuous functions, suitable for differentiation.
- Observation: Differentiation is just a convolution!

$$\frac{\partial}{\partial x}\phi = \frac{\partial}{\partial x} \otimes \phi$$

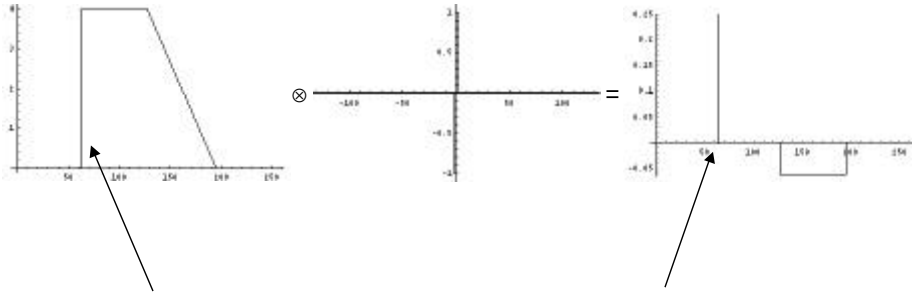


- Problem: Derivatives of real-world data are not always defined.
- Observation: Differentiation is associative and commutative.

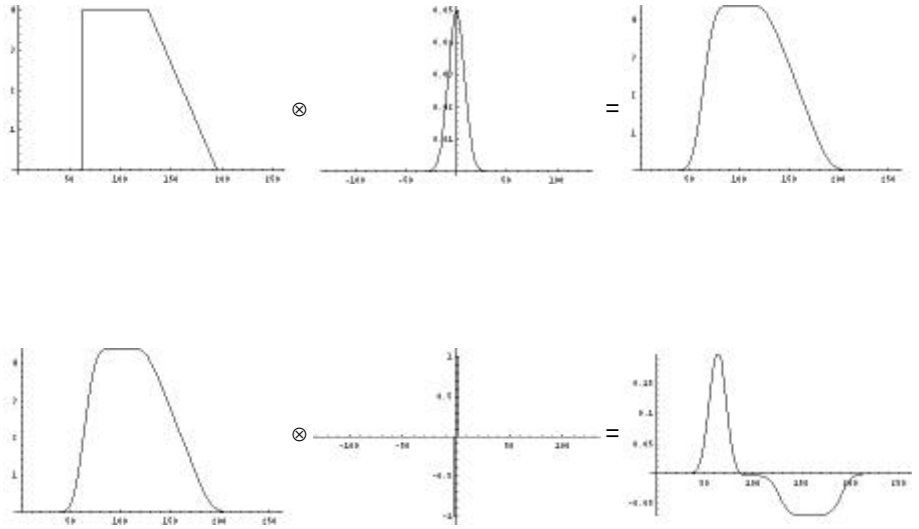
$$h(x) \otimes \partial/\partial x \otimes f(x) = \partial/\partial x \otimes h(x) \otimes f(x)$$

- Solution: Combine differentiation with a regularizing kernel.

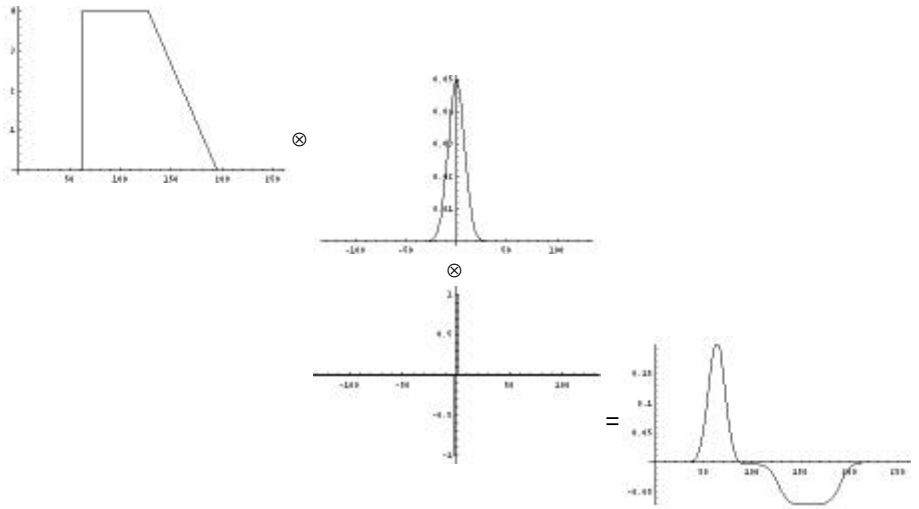
Differentiation



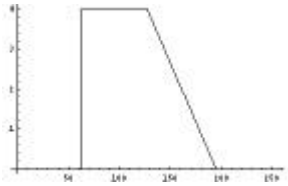
Differentiation



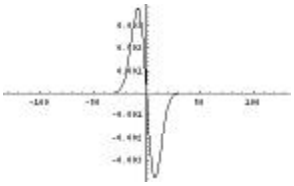
Differentiation



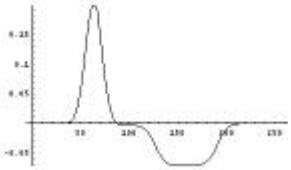
Differentiation



⊗



=



Now for something different…

- ***But not completely different...***

$$e^{i\pi} = -1$$

$$e^{\alpha + i\beta} = \cos \alpha + i \sin \beta$$

The Fourier Transform

- A different mirror with which to view images.
- The Fourier transform

$$\mathcal{F}(\phi(x)) = \int_{-\infty}^{\infty} \phi(x) e^{-i2\pi x\nu} dx = \Phi(\nu)$$

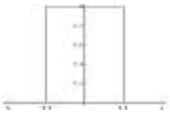
- The result transforms a function of x in image space to a function of n in frequency space.
- There is an Inverse Fourier transform for undoing this process.

$$\mathcal{F}^{-1}(\Phi(\nu)) = \int_{-\infty}^{\infty} \Phi(\nu) e^{i2\pi x\nu} d\nu = \phi(x)$$

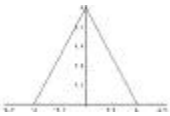
Fourier Transforms of common functions

<u>Image Space:</u>

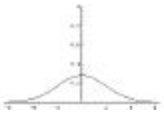
Box
(nearest neighbor)



Pyramid
(linear interpolant)



Gaussian
(std. distribution)

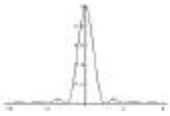


Comb / Shah
(sampling function)

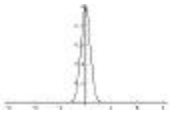
<u>Frequency Space:</u>

F(Box)
sinc

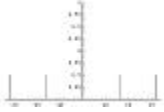
F(Pyramid)
(linear interpolant)



Gaussian
(non-normalized)



Comb



Properties of the Fourier Transform

Property:

- Frequency Scaling ⊙
  Inverse Spatial Scale Change

$$\mathcal{F}\frac{1}{|a|}\phi\left(\frac{\nu}{a}\right)(\phi(ax)) = \Phi(a\nu)$$

and

$$\mathcal{F}^{-1}\frac{1}{|a|}\Phi\left(\frac{\nu}{a}\right) = \phi(ax)$$

Property:

- Spatial Scaling ⊙
  Inverse Frequency Scale Change

$$\mathcal{F}(\phi(ax)) = \frac{1}{|a|}\Phi\left(\frac{\nu}{a}\right)$$

and

$$\mathcal{F}^{-1}(\Phi(a\nu)) = \frac{1}{|a|}\phi\left(\frac{\nu}{a}\right)$$

The Convolution Theorem

- Convolution in space = multiplication in frequency
- Convolution in frequency = multiplication in space

$$\mathcal{F}(\phi(x) \otimes h(x)) = \Phi(\nu)H(\nu)$$

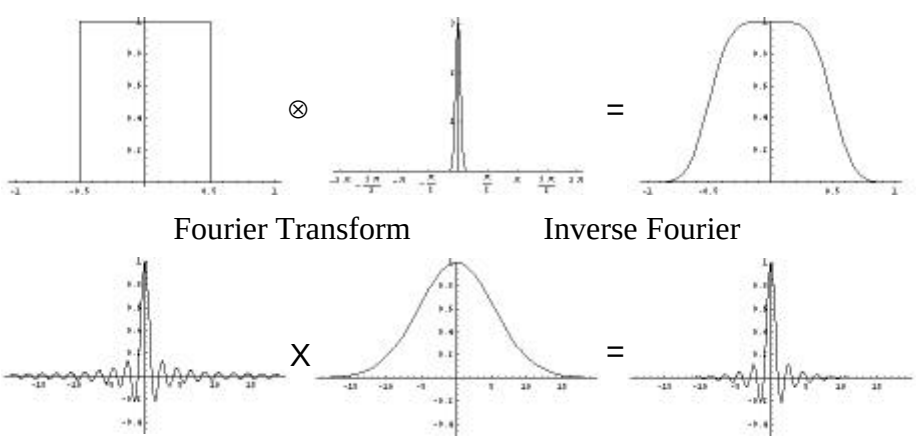
$$\mathcal{F}^{-1}(\Phi(\nu) \otimes H(\nu)) = \phi(x)h(x)$$

Revisiting Convolution

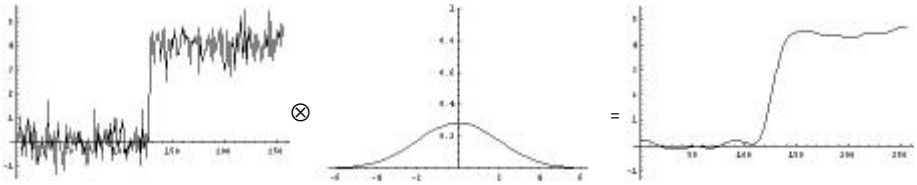
•now see convolution as a transform, a multiplication, and an inverse transform.

$$\phi(x) \circledast h(x) = \mathcal{F}^{-1}(\mathcal{F}(\phi(x))\mathcal{F}(h(x)))$$

Revisiting Convolution (2)

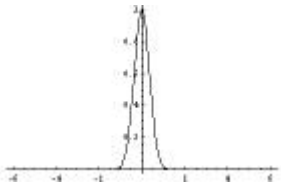


Revisiting Convolution (3)



Input, $\phi(x)$ $\otimes$ Kernel, $g(x,\sigma)$ $= \phi(x) \otimes g(x,\sigma)$

$\downarrow$ Fourier Transform $\downarrow$ Inverse Fourier $\uparrow$

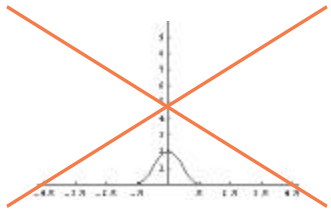


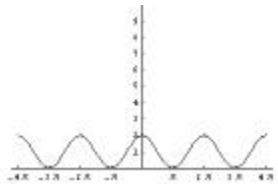
F(Input), $\Phi(x)$ $\times$ Kernel, $G(x,\sigma)$ $= \Phi(x) \diamond G(\nu,\sigma)$

Discrete Fourier Transforms

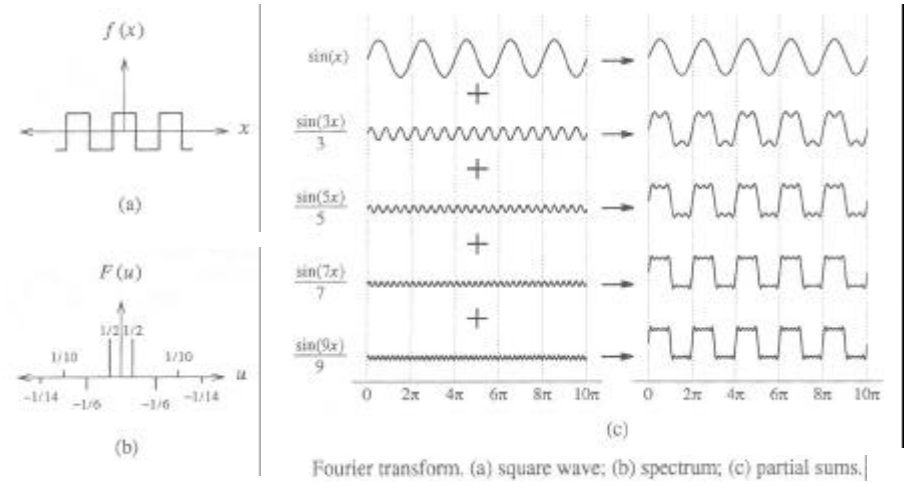
- Periodic.
  - *Assume that the function is a single period of a infinitely repeating function.*
  - *Or: Think of it as an image that wraps onto itself like a doughnut (torus).*
- Discrete.
  - *If there are n samples in the spatial domain, there will be n samples in frequency domain, too.*

Discrete Fourier Transforms (1)





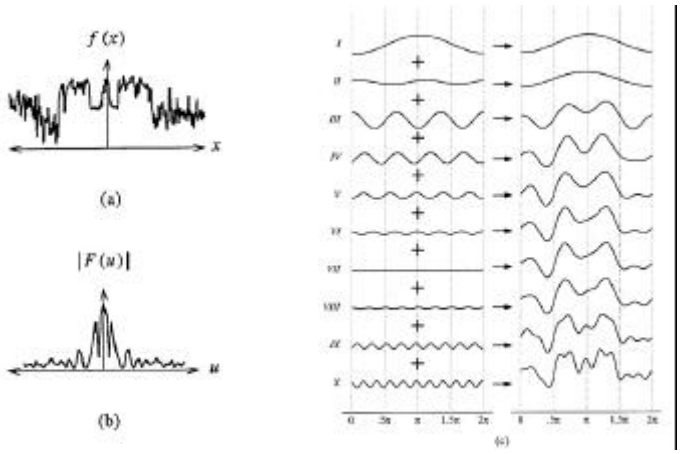
Discrete Fourier Transforms (2)



Fourier transform. (a) square wave; (b) spectrum; (c) partial sums.

(from G. Wolberg, 1990, Digital Image Warping, IEEE Press)

Discrete Fourier Transforms (3)

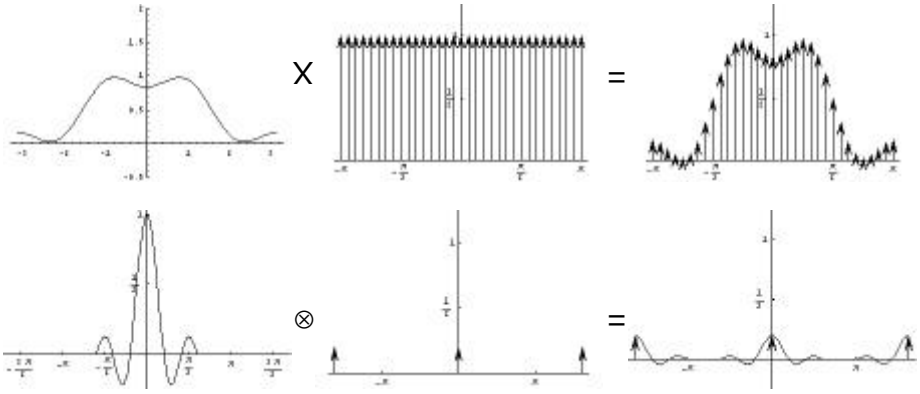


Fourier transform. (a) aperiodic signal; (b) spectrum; (c) partial sums.

(from G. Wolberg, 1990, Digital Image Warping, IEEE Press)

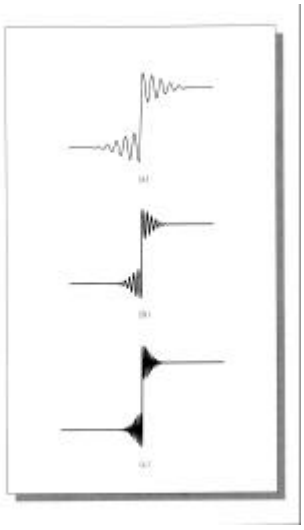
Sampling

- ***now see sampling as a multiplication with frequency issues.***



Sampling (2)

Note: adding more (higher) frequencies alters the period of the ringing, but does not reduce the amplitude.



(from A. Glassner, 1995, Principles of Digital Image Synthesis, Morgan Kaufman)

Sampling (3)

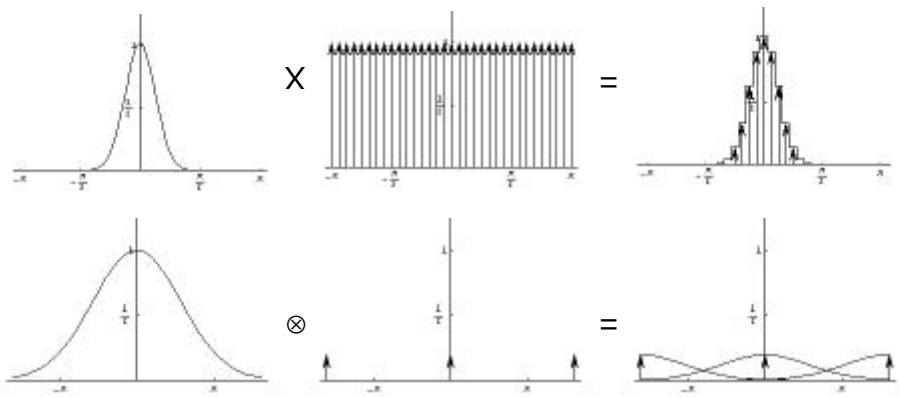
- Bandlimited.
  - A signal ϕ is considered *bandlimited* if its Fourier transform, $F(\phi(x)) = \Phi(\omega)$ satisfies the following condition

$$\Phi(\omega) = 0 \quad \text{and} \quad \Phi(-\omega) = 0 \quad \text{for all } \omega > \omega_{limit}$$

- Satisfies the Nyquist criterion.
  - The discrete signal does not contain frequencies higher than 1/2 the sampling frequency

Aliasing

- Signal that isn't Bandlimited.
- Sampling effects.



Sources of Aliasing

Non-bandlimited signal

Low sampling rate (below Nyquist)

Non perfect reconstruction

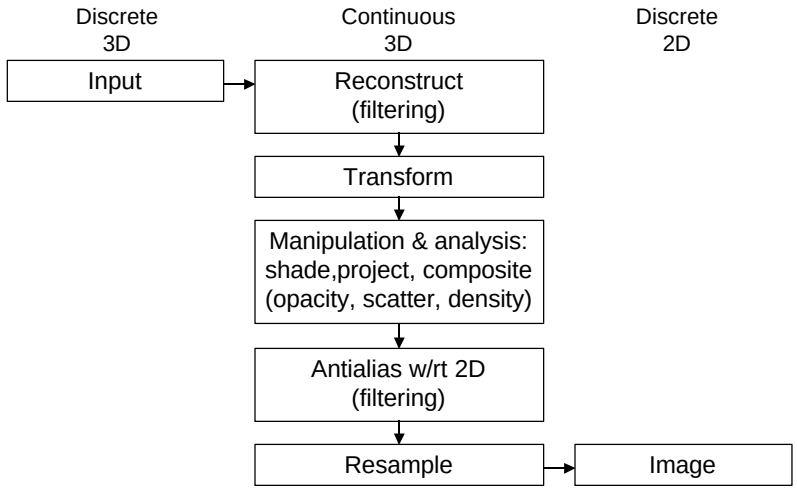
Possible Errors

- Post-aliasing
  - –Reconstruction filter passes frequencies beyond the Nyquist frequency (of duplicated frequency spectrum) => frequency components of the original signal appear in the reconstructed signal at different frequencies.
- Smoothing
  - –Frequencies below the Nyquist frequency are attenuated.

Possible Errors(2)

- Ringing (overshoot)
  - –Occurs when trying to sample/reconstruct discontinuity.
- Anisotropy
  - –Caused by non-spherically symmetric filters.
  - –Requires filters that are invariant with respect to rotation.

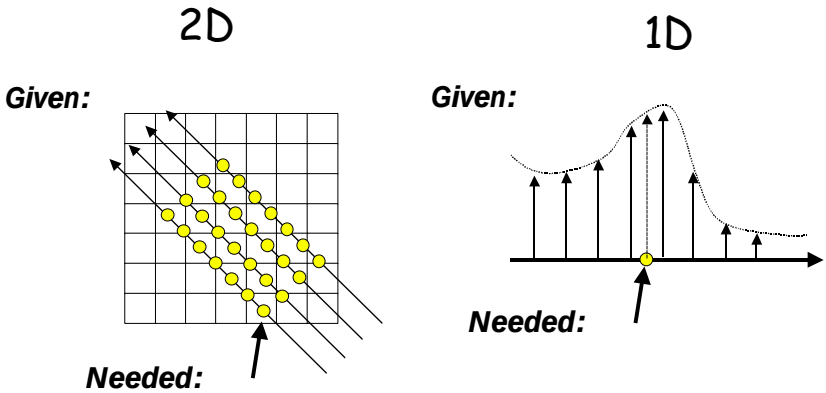
the pipeline (westover ‘91)



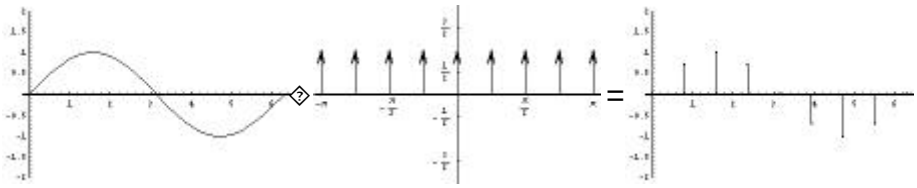
input

- Bandlimited.
- Appropriately sampled: above the Nyquist frequency.

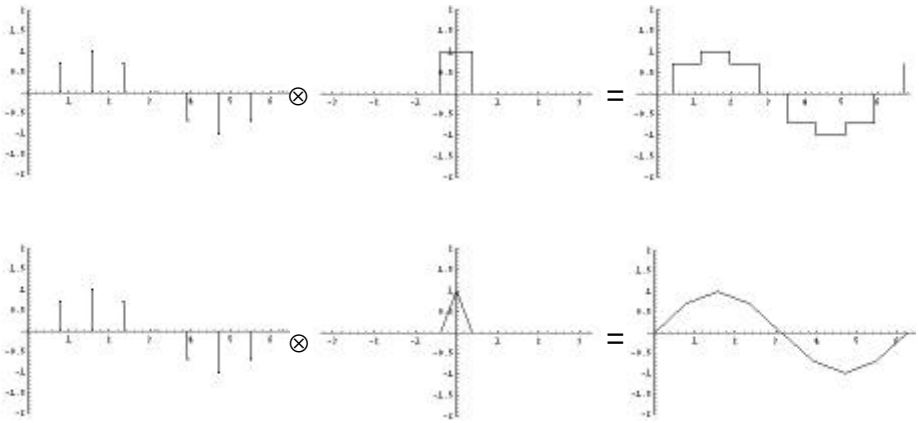
Reconstruction - Interpolation



Example in 1D



How? Convolution



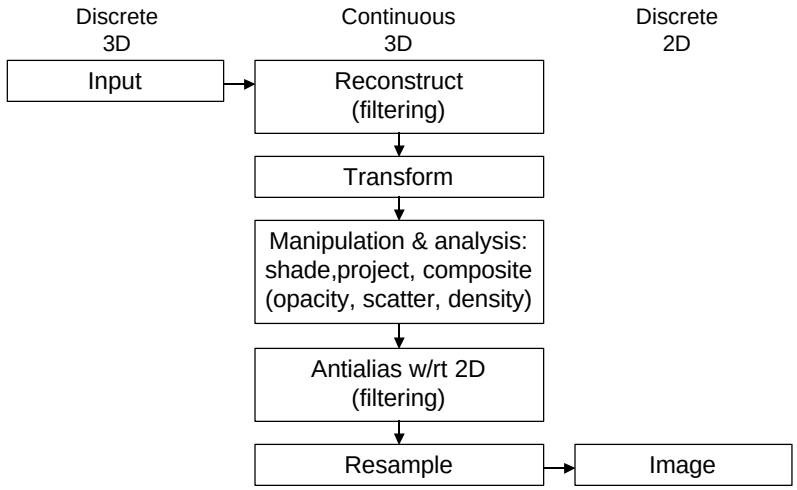
Interpolation (summary)

- Very important; regardless of algorithm
  expensive => done very often for one image
- Requirements for good reconstruction
  - *performance*
  - *stability of the numerical algorithm*
  - *accuracy*

Nearest neighbor

Linear

the pipeline (westover '91)



transformation - magnify / minify

- Remember, spatial scaling = inverse frequency scale.
- Magnification / scaling of the reconstructed input ⇒ transformation / minimization of the sampling function.
- Minification / scaling of the reconstructed input ⇒ transformation / magnification of the sampling function.

Resample

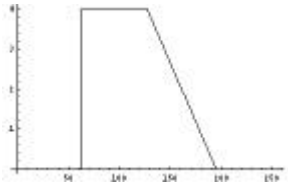
- with antialiasing
- without antialiasing

Summary

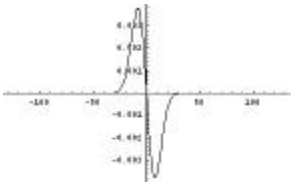
- For (n = 1; n < image_size; n++)

```
gradx[n] =  image[n-1] * -0.5 + image[n+1] * 0.5;
```

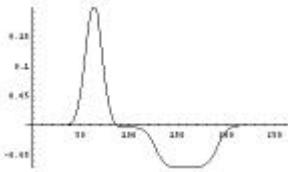
You're really approximating...



⊗



=



With all the consequences in the frequency domain!

Summary

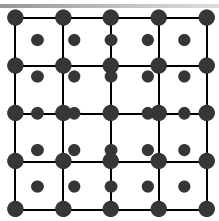
- Convolution is a basic operation, used in:
  - interpolation, reconstruction.
  - Noise filtering.
  - Differentiation, measurement.
  - Statistics.
- The frequency domain.
  - It exists.
  - Operations in discrete images have frequency based consequences.
  - It's happening whether you're watching for it or not.

Evaluation & Design of Filters

Torsten Moeller, Simon Fraser Univ.
Raghu Machiraju, The Ohio State University

Other Significant Contributors:
Klaus Mueller, SUNY-Stony Brook
Roni Yagel, Insight Theraputics

Image Processing

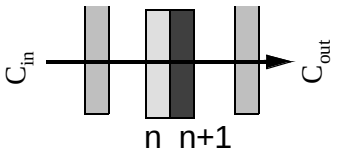


Nearest neighbor

Linear Interpolation

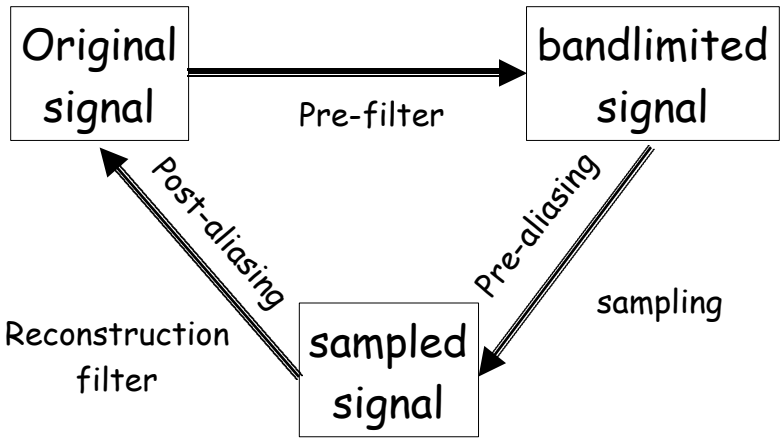
Practical Volume Rendering

- Numerical evaluation of Rendering Integral
- discretization of rays necessary
- Interpolate - *f(t)*
- Determine opacity - a(*t*)
- Determine shade - c(*t*)



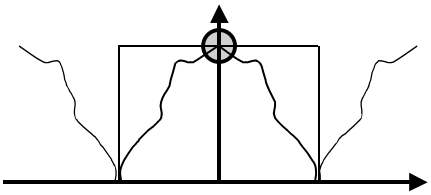
SIGGRAPH 2001

Signal Sources & Processing



SIGGRAPH 2001

Functional Assumptions

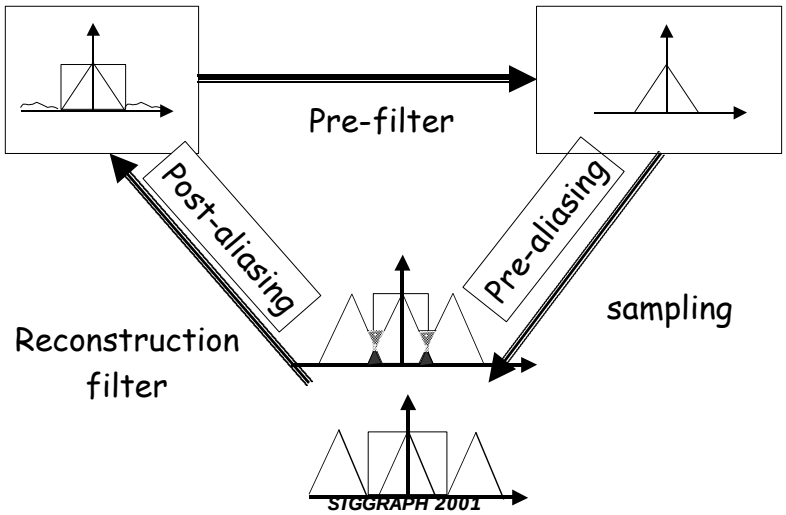


BandLimited Function: $F(\omega) = 0, \omega > \omega_{Nyqyuist}$

Sampled signal originated from a smooth polynomial signal

SIGGRAPH 2001

Signal Processing



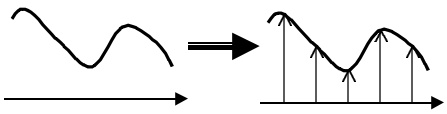
SIGGRAPH 2001

Framework

Numerical Analysis approach:

- discretization:

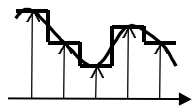
$$f[k] = f(kT)$$



- reconstruction: Interpolation

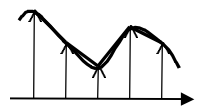
Nearest neighbor

ppi with zero-degree polynomial



Linear interpolation

ppi with first degree polynomial



Problem Definition

- Given a set of *n+1* points $\{(x_i, f_i)\}$.
- Find a function $f(x)$ for an arbitrary x.

Approximation:

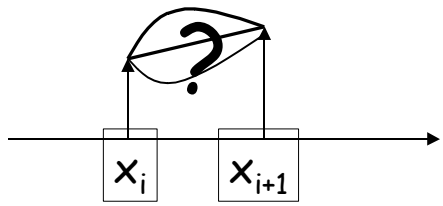
$$f(x_i) - f_i = e$$

Interpolation:

$$f(x_i) - f_i = 0$$

Problem Definition

- Generally there is **no way** to predict the values in-between the given (measured) points (x_i, f_i).
- Need knowledge of the space of functions, that our measurements were made in.



Mathematical Approach

- Usual assumptions are:
  - space of smooth functions C^n
  - space of bandlimited functions
- Function spaces characterized by specific basis functions $\phi_i(x)$,
- Each function expressed as a linear combination of these bases:

$$f(x) = a_0 f_0(x) + a_1 f_1(x) + \ldots$$

Mathematical Approach (2)

- different possible bases:
  - represent different function spaces
  - vary in quality
  - vary in computational efficiency
- Resort to:
  - Polynomial interpolation (Lagrange basis, Newton form)
  - Piecewise polynomial interpolation (Hermite, cubic splines, Bezier, B-splines)
  - Orthogonal polynomials (Legendre, Chebychev, others)
  - Trigonometric functions
  - Wavelet basis

Polynomial Interpolation

- Given: Set of n+1 points $S = \{(x_i, f_i)\}$
- Sought: Representation of f(x) in the polynomial basis $\phi_i(x) = x^i$
- there is a unique polynomial p_n of degree n, that interpolates S

$$f(x) \approx p_n(x) = a_0 x^0 + a_1 x^1 + \ldots + a_n x^n$$

Lagrange Polynomials

- Defined as:

$$l_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

- We conclude:

$$p_n(x) = \sum_{k=0}^{n} f_k l_k(x)$$

- Problems: very costly to evaluate.
- Solution: Power form or Horner s Method
  Better - Newtons form

Accuracy

- If $f \in C^{n+1}(I)$, then $\forall z \in I\ \exists x \in (\min(z, x_{\min}), \max(z, x_{\max}))$:

$$f(z) - p_n(z) = \frac{r_n(z) f^{(n+1)}(x)}{(n+1)!}$$

- where

$$r_n(z) = \prod_{k=0}^{n} (x - x_k)$$

- *Warning 1:* For z outside $[x_{\min}, x_{\max}]$ $r_n(z)$ grows way too fast.
- *Warning 2:* There are functions for which $p_n(x)$ doesn t approach f(x).

Piecewise Polynomial Interp.

- Many possible schemes for PPI, which differ in their accuracy, and features
  - Hermite,
  - p^{th} degree Splines
  - Bezier
  - B-splines
  - NURBS.

Orthogonal polynomials

- Computation of coefficients of $p_n(x)$ is unstable.
- Basis x^i is not orthogonal.
- Rewrite polynomial $p_n(x)$ in a different, *orthogonal*, basis P_i:

$$p_n(x) = b_0 P_0(x) + b_1 P_1(x) + \ldots + b_n P_n(x)$$

Orthogonal polynomials (2)

- with condition, that inner product is zero:

$$\left\langle P_i, P_j \right\rangle_w = 0$$

- where *w* is a weighting function. For a continuous domain, we have:

$$\left\langle P_i, P_j \right\rangle_w = \int_a^b P_i(x) P_j(x) w(x) dx$$

Orthogonal polynomials (3)

- Common orthogonal polynomials:
  - Legendre,
  - Chebychev,
  - Trigonometric functions (not polynomials)
  - Bessel functions (not polynomials).

Trigonometric functions

- More than just a tool for interpolation
- Widely used for analysis of signals
- Basis functions:

$$f_w(x) = e^{iwx}$$

- Transform

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{iux} du$$

$$F(u) = \frac{1}{2p} \int_{-\infty}^{\infty} f(x) e^{-iux} dx$$

Trigonometric functions (2)

- Discrete transform - is an approximation of the continuous transform, but constructed, such that the inverse transform works:

$$f_j = \sum_{k=0}^{n-1} F_k e^{i2pjk/n}$$

$$F_k = \frac{1}{n} \sum_{j=0}^{n-1} f_j e^{-i2pjk/n}$$

- With basis functions:

$$f_j^n(k) = e^{i2pjk/n}$$

Wavelet basis

- Major idea - decompose the function space into smaller spaces:

$$V_{j+1} = V_j \oplus W_j$$

- Typically this multi-scale representation can be exploited:

$$V_{j+1} = V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_j$$

Wavelet basis (2)

- Basis $\varphi_{j,l}(x)$ for smooth spaces V_j fulfills dilation equation:

$$j\,(x) = 2\sum_k h_k j\,(2x-k)$$

$$j_{\,j,l}(x) = \sqrt{2^j}\, j\,(2^j x - l)$$

- Basis $\psi_{j,l}(x)$ for detail spaces W_j fulfills wavelet equation:

$$y\,(x) = 2\sum_k g_k j\,(2x-k)$$

Wavelet basis (3)

- Synthesis of a function (assuming $\varphi_{j,l}(x)$, $\psi_{j,l}(x)$ are orthogonal):

$$f(x) = \sum_{j,k} b_{jk} y_{j,k}(x)$$

- Analysis of a function (assuming $\varphi_{j,l}(x)$, $\psi_{j,l}(x)$ are orthogonal):

$$b_{JK} = \int_{-\infty}^{\infty} f(x) y_{J,K}(x) dx$$

Requirements

- Performance
- Stability of the numerical algorithm
- Accuracy (numerical + perceptual)
- Smoothness of reconstructed function

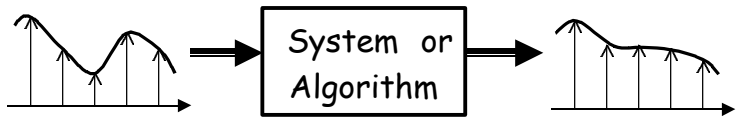
Requirements (2)

- Accuracy considerations depend on underlying function space.
- For smooth function spaces, we consider asymptotic error behaviors.
- For bandlimited spaces, we consider blurring, aliasing and overshoot (Frequency domain).
- Not considered Perceptual metrics

Reconstruction in Practice

Engineering approach:

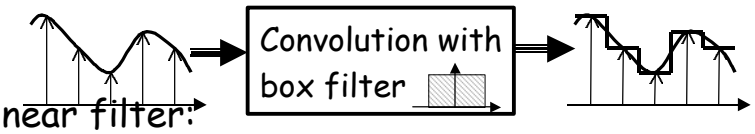
black-box with reconstruction filter



Reconstruction in Practice (2)

Engineering approach (cont.):

- nearest neighbor

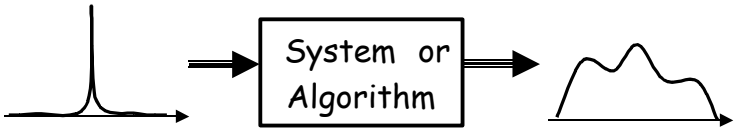


- linear filter:

SIGGRAPH 2001

Reconstruction - Convolution

- How can we characterize our black box ?
- Black-box/algorithm:
  - linear
  - time-invariant
- then it can be characterized by the response to an impulse :



SIGGRAPH 2001

Convolution (2)

- Finite Impulse Response (FIR) vs.
- Infinite Impulse Response (IIR)
- Impulse:

$$d(x) = 0, \text{if } x \neq 0$$

$$\int_{-\infty}^{\infty} d(x)dx = 1$$

- discrete impulse:

$$d[k] = 0, \text{if } k \neq 0$$

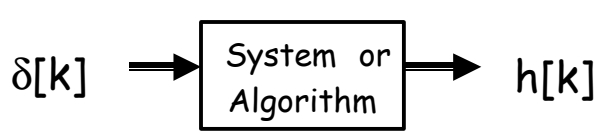
$$d[0] = 1$$

Convolution (3)

- An arbitrary signal x[k] can be written as:

$$x[k] = \cdots + x[-1]d[k+1] + x[0]d[k] + x[1]d[k-1] + \cdots$$

- Let impulse response be h[k]:

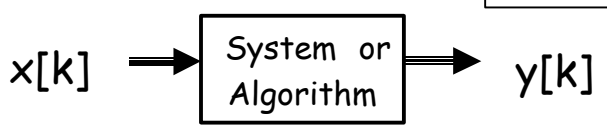


Convolution (4)

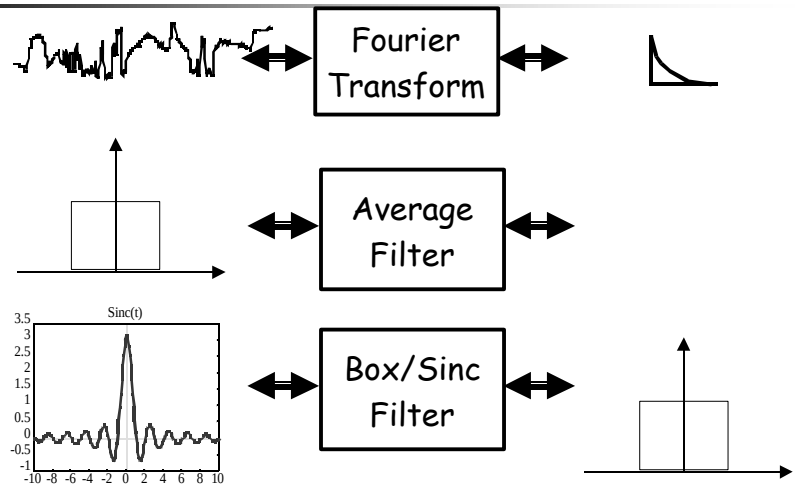
- for a time-invariant system h[k-n] would be the impulse response to a delayed impulse δ[k-n]
- hence, if y[k] is the response of our system to the input x[k] (and we assume a linear system):

$$y[k] = \sum_{n=-N}^{N} x[n]h[k-n]$$

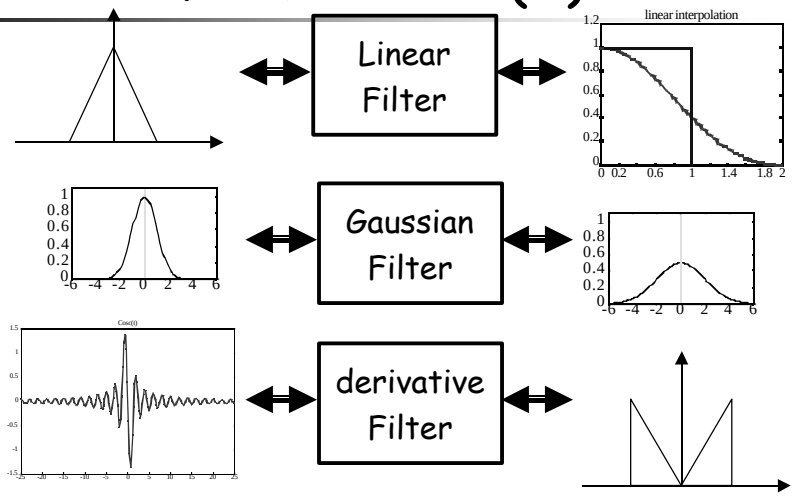
IIR - N=inf.
FIR - N<inf.



Transforms Pairs



Transforms Pairs (2)



SIGGRAPH 2001

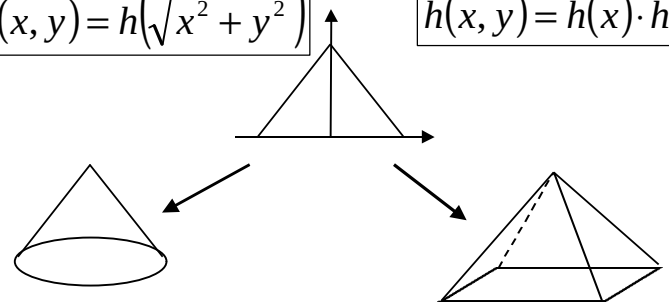
Higher Dimensions

- An-isotropic Filters
- (radially symmetric)

$$h(x,y) = h\left(\sqrt{x^2 + y^2}\right)$$

- separable filters

$$h(x,y) = h(x) \cdot h(y)$$



SIGGRAPH 2001

Filter Evaluation

Frequency Domain

- global features
- robust noise handling
- no blurring
- no aliasing

⇩ Signal Processing

Spatial Domain

- local features
- minimal error
- smoothness

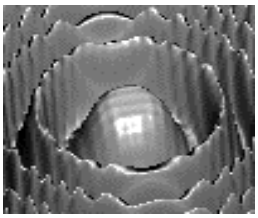
⇩ Approximation Theory/Analysis

Frequency Domain Issues

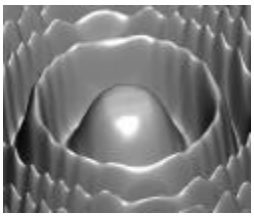
- Postaliasing
  - reconstruction filter passes frequencies beyond the Nyquist frequency (of duplicated frequency spectrum)
  - Frequency components of the original signal appear in the reconstructed signal at different frequencies
- Smoothing
  - frequencies below the Nyquist frequency are attenuated

Frequency Domain Issues (2)

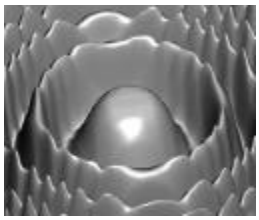
- Ringing (overshoot)
  - occurs when trying to sample/reconstruct discontinuity
- Anisotropy
  - caused by not spherically symmetric filters



Aliasing

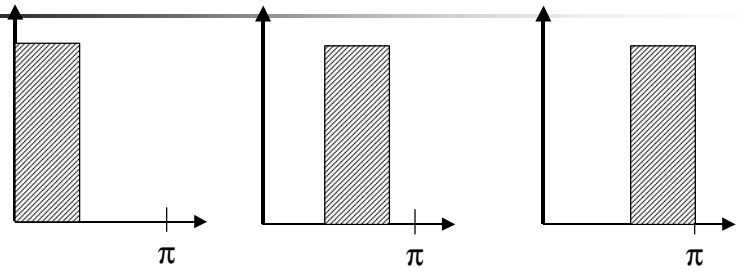


OK



Blurring

Ideal Reconstruction



Low-pass filter | band-pass filter | high-pass filter

Realizable filters do not have sharp transitions; also have ringing in pass/stop bands

Evaluation of Filters

- Frequency methods
  - Discrete Fourier Transform
  - Divergence from Ideal
  - Laplace Transform
- Spatial Methods
  - Discrete Moments
  - Asymptotic Error
  - Smoothness

Fourier Transforms

- Let s look at a special input sequence:

$$x[k] = e^{iwk}$$

- then:

$$y[k] = \sum_{n=-N}^{N} e^{iw(k-n)} h[n]$$

$$= e^{iwk} \sum_{n=-N}^{N} e^{-iwn} h[n]$$

$$= H(w) e^{iwk}$$

Fourier Transforms (2)

- Hence e^{iwk} is an eigen-function and H(ω) its eigenvalue
- H(ω) is the Fourier-Transform of the h[n] and hence characterizes the underlying system in terms of frequencies
- H(ω) is periodic with period 2π
- H(ω) is decomposed into
  - phase (angle) response $\sphericalangle H(w)$
  - magnitude response $|H(w)|$

Example

$$y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$$

$$x(n) = e^{in\omega}$$

$$y(n) = \left(\frac{1}{2} + \frac{1}{2}e^{-n\omega}\right)e^{in\omega}$$

$$C(\omega) = \left(\frac{1}{2} + \frac{1}{2}e^{-n\omega}\right)$$

$$C(\omega) = \left(\cos\frac{\omega}{2}\right)e^{-i\frac{\omega}{2}}$$

$$C(0) = 1$$

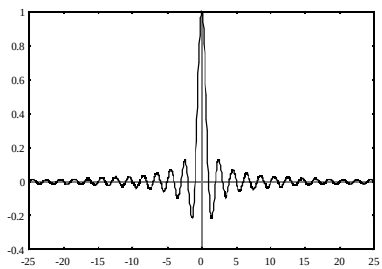
$$C(\pi) = 0$$

Ideal Interpolation

Spatial Domain:

- convolution is exact

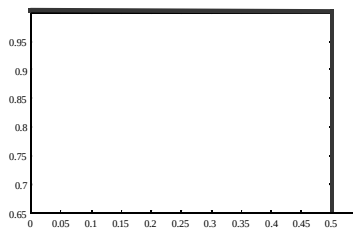
$$f_r(x) - f(x) = 0$$



Frequency Domain:

- cut off freq. replica

$$\mathrm{Sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

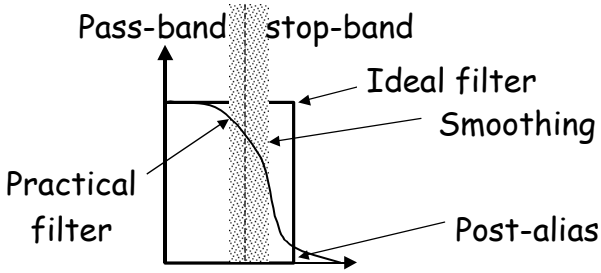


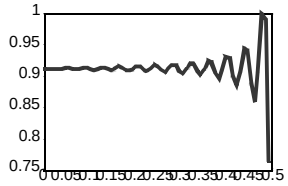
SIGGRAPH 2001

Or Ideal Reconstruction

- Box filter in frequency domain =
- Sinc Filter in spatial domain

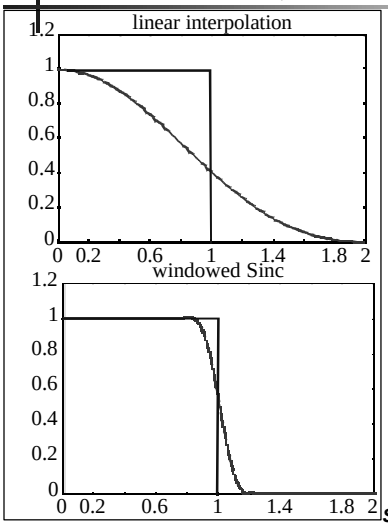
impossible to realize (really?)

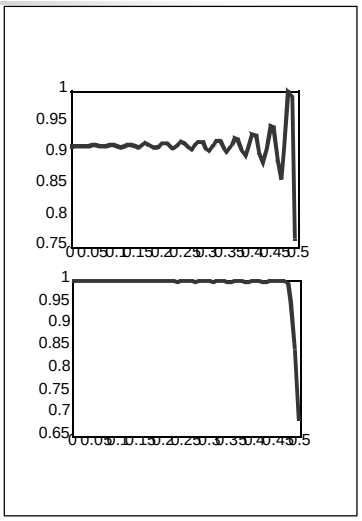




SIGGRAPH 2001

An Easy Fix Windowing





SIGGRAPH 2001

Windows ?

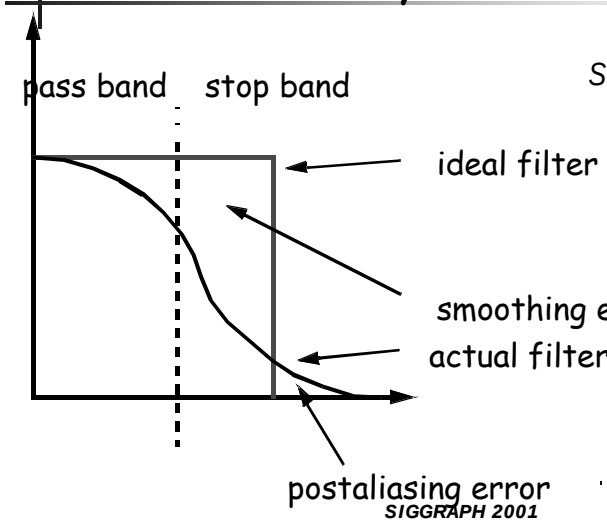
$$\mathrm{wind}(t, M) = \begin{cases} 1 & -M \le t \le M \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{wind}(t, M) = \begin{cases} 1 + t/M & -M \le t \le 0 \\ 1 - t/M & 0 \le t \le M \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{wind}(t, M) = \begin{cases} \alpha + \beta \cos\left(\pi \frac{t}{M}\right) & -M \le t \le M \\ 0 & \text{otherwise} \end{cases}$$

SIGGRAPH 2001

Far From Ideal - Marschner, Lobb



$$S(h) = 1 - \frac{1}{|R|}\int_R |H|^2 dw$$

$$P(h) = \frac{1}{|R|}\int_R |H|^2 dw$$

Carlbom

Approximates ideal frequency response with error

Uses Remez algorithm to minimize error:

With ideal interpolation filter

Where τ is the offset

Ideal Derivatives

Spatial Domain:

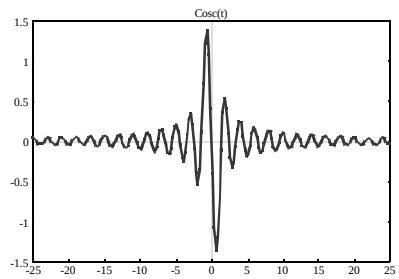
- convolution is exact

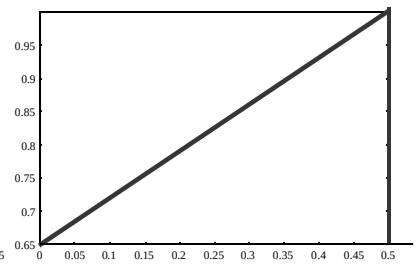
$$f_r^d(x) - f'(x) = 0$$

Frequency Domain:

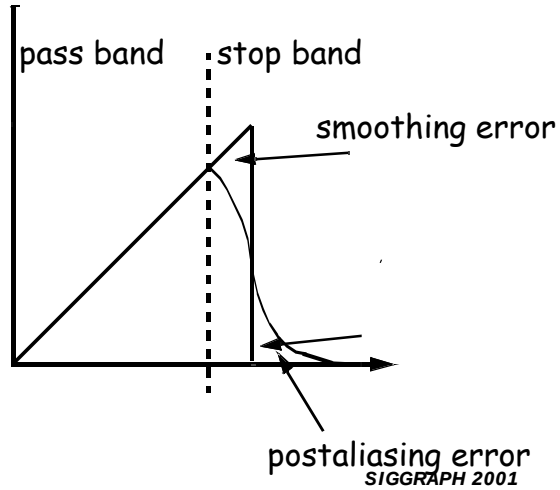
- cut off freq. replica

$$\mathrm{Cosc}(x) = \frac{\cos(\pi x)}{x} - \frac{\sin(\pi x)}{\pi x^2}$$

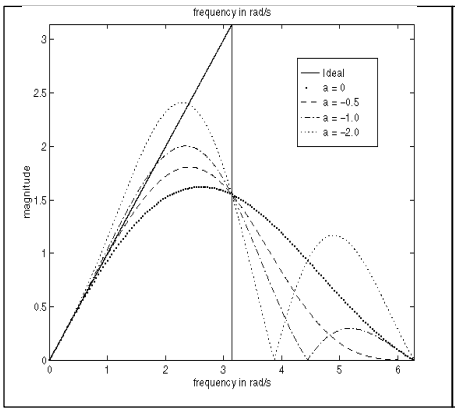


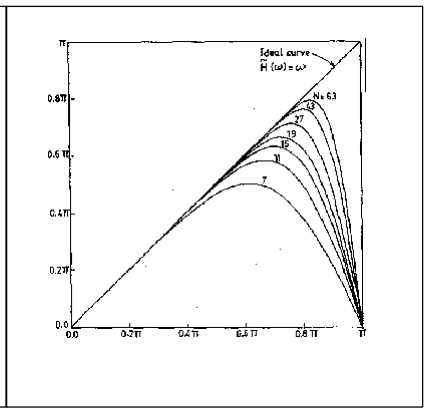


Far From Ideal



Practical Derivative Filters

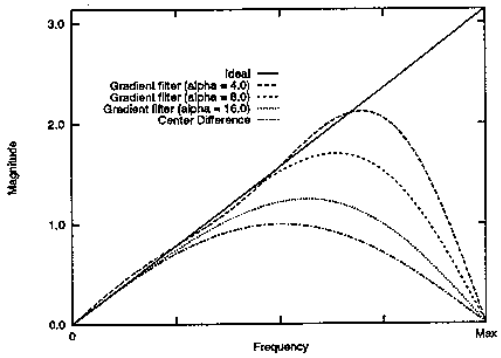




Lichtenbelt et al

Dutta Roy & Kumar

QuickFix Windowing (Goss)



- Window of ideal derivative filter (Cosc).
- Kaiser window (based on Bessel Functions)

Example Fourier Trans.

$$y(n) = \frac{1}{2}x(n) - \frac{1}{2}x(n-1)$$

$$x(n) = e^{inw}$$

$$D(w) = \left(\frac{1}{2} - \frac{1}{2}e^{-nw}\right)$$

$$C(w) = \left(sin\frac{w}{2}\right)e^{-i\frac{w}{2}}$$

$$C(0) = 0$$

$$C(p) = 1$$

Accuracy

- Taylor Series:
  - approximate the error of the numerical algorithm
  - evaluate its asymptotic behavior.
- Assumption:

  some derivatives of the underlying function *f* exist.

$$f(x) = f(t) + f'(t)\frac{(x-t)}{1!} + f''(t)\frac{(x-t)^2}{2!} + \ldots$$

Keys

Constructs cubic interpolation kernel, such that it is symmetric, interpolating, C^2

Leads to cardinal splines

Uses Taylor series expansion - optimal $\alpha = -0.5$

Compares these methods in spatial as well as in frequency domain

$$h(t) = \begin{cases} (\alpha+2)|t|^3 - (\alpha+3)|t|^2 + 1 & 0 < |t| < 1 \\ \alpha|t|^3 - 5\alpha|t|^2 + 8\alpha|t| - 4\alpha & 1 < |t| < 2 \\ 0 & 2 < |t| \end{cases}$$

Mitchell, Netravali

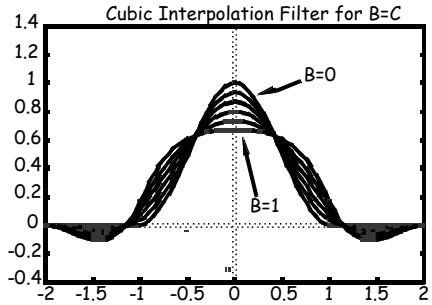
Construct cubic filters (piecewise polynomial), that are C^1 normalized

$$h(t) = \begin{cases} \left(2-\frac{3}{2}B-C\right)|t|^3 + (-3+2B+C)|t|^2 + \left(1-\frac{B}{3}\right) & 0 \le |t| < 1 \\ \left(-\frac{1}{6}B-C\right)|t|^3 + (B+5C)|t|^2 + (-2B-8C)|t| + \left(\frac{4}{3}B+4C\right) & 1 \le |t| < 2 \\ 0 & 2 \le |t| \end{cases}$$

Numerical study:

$$f(t) - f_r^h(t) = (2C + B - 1)Tf'(t)r(t) + O(T^2)$$

Mitchell, Netravali (contd.)



Spatial Domain Evaluation

- Reconstruction of the Nth derivative:

$$f_r(t) = a_0 f + \ldots + a_{N-1} f^{(N-1)} + a_N f^{(N)} + a_{N+1} f^{(N+1)} + \ldots$$

- *Analyze* Are all $a_l(\tau) = 0$ for $l<N$?
- *Normalize* Is $a_N(\tau) = 1$?
- *Classify* How many $a_l(\tau)=0$ for $l>N$?
- *Error* How large is the error term?

Apply Taylor Series

- Numerical Reconstruction - Convolution

$$f_r(t) = \ldots + f(k-1)w(t-(k-1)) + f(k)w(t-k) + \ldots$$

- Apply Taylor Series to Convolution Sum

$$f_r(t) = a_0(\tau) f(t) + a_1(\tau) f'(t) + a_2(\tau) f''(t) + \ldots$$

SIGGRAPH 2001

What are all these a s ?

$$f_r^w(t) = \sum_{l=0}^{L} a_l^w(\tau) f^{(l)}(t) + r_{L,i}^w(\tau)$$

$$a_l^w(\tau) = \frac{T^l}{l!} \sum_{k=-\infty}^{\infty} (k-\tau)^l w(\tau - k)$$

- They are nothing but discrete moments !
- And one can also prescribe the accuracy of reconstructed function

SIGGRAPH 2001

Example

- Good interpolation filters

  $\boxed{a_0(\tau)=1}\quad a_1(\tau)\approx 0\quad a_2(\tau)\approx 0$

- Good derivative filters

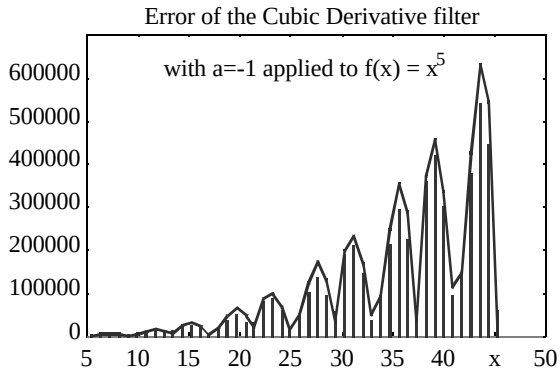
  $a_0(\tau)=0\quad \boxed{a_1(\tau)=1}\quad a_2(\tau)\approx 0$

- Good 2nd derivative filters

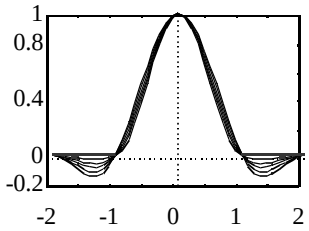
  $a_0(\tau)=0\quad a_1(\tau)=0\quad \boxed{a_2(\tau)=1}$

Error Estimation

$$r^{w}_{L,i}(\tau) \le \left(\max_{\xi \in [(i-M)T,\,(i+M)T]} \left(f^{(L+1)}(\xi)\right)\right)\left|a^{w}_{L+1}(\tau)\right|$$



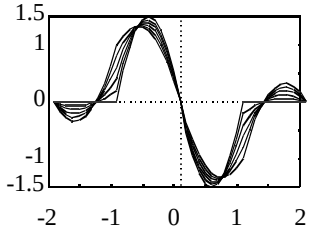
Catmull-Rom Interpolation



$a_0=1$, $a_1=T(2\alpha+1)(1-2t)(t-1)$

$a_2 = 2T^2(2\alpha+1)t^2(t-1)^2$, $a_3=T^3t(1-2t)(1-t)$

Catmull-Rom Derivative



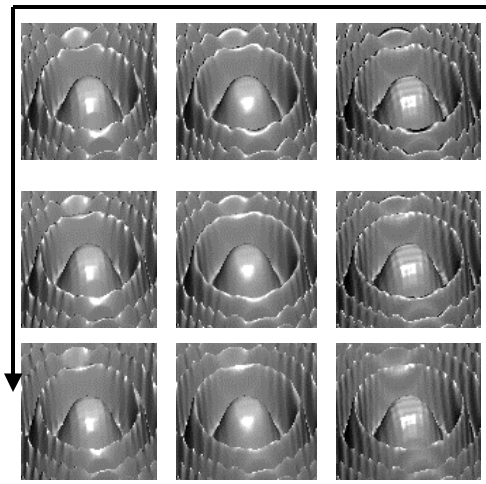
$a_0=1$, $a_1=T(2\alpha+2)(1-6t)(t-1)$

$a_2 = 3T^2(2\alpha+1)\ T(1-\tau)(1-2\tau)$, $a_3=T^3(6\tau^2-6\tau+1)$

Proof is in the Pictures

Derivative
α = -0.2, -.5, -1.0

Interpolation
α = -0.2, -.5, -1.0



SIGGRAPH 2001

Normal Estimation Schemes

Method (FD)H - derivative first
- Compute normal at grid points
- Interpolate new normals

Method (FH)D - interpolation first
- Reconstruct continuous function *f(t)*
- Apply discrete derivative filter

Method F(DH) - continuous derivative (new)
- Convolve derivative filter with interpolation filter
- Apply this filter to the data samples

Method FH - analytic derivative
- Analytical derivative of the interpolation filter

SIGGRAPH 2001

Normal Estimation Schemes

Central Difference + Linear Interpolation:

$a_0=1,\ a_1=T\ a_2 = 0,\ a_3=T^3(-3\tau^2+3\tau+1)$

Cubic Derivative (α=-0.5):

$a_0=1,\ a_1=0,\ a_2 =0,\ a_3=T^3(6\tau^2-6\tau+1)$

Turns out Simple Estimation is not that bad after all !

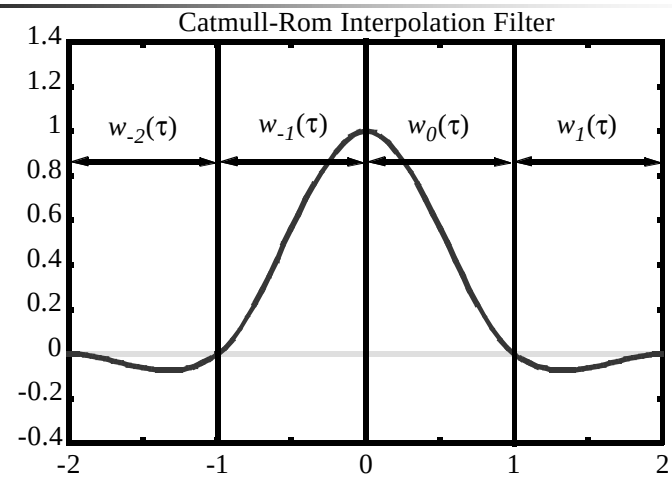
Spatial Filter Design

Which **DERIVATIVE** of the original function do we want to reconstruct?

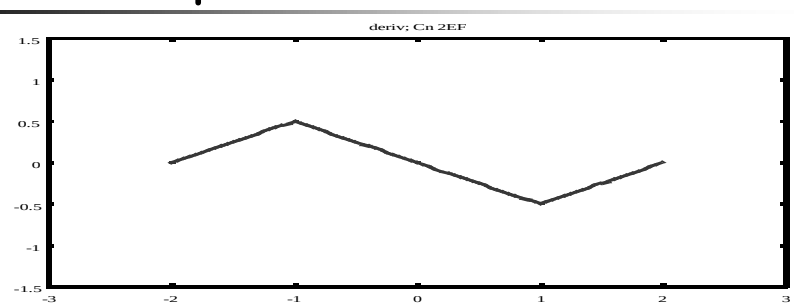
What **ACCURACY** do we require from the reconstruction process?

How **SMOOTH** (space C^n) should the reconstructed function be?

Piecewise Catmull-Rom

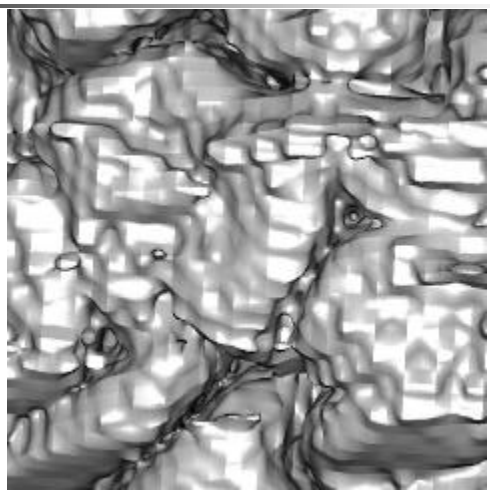


Example



E.g. $w_{-2} = \frac{1}{2}\tau^2 - \frac{1}{6} + \frac{1}{6}a$

Accuracy Not Enough



SIGGRAPH 2001

Smoothness

- Smoothness = Visual Artifacts
- Reconstructed Function:

$$f_r(t) = \ldots + f(k-1)w(t-(k-1)) + f(k)w(t-k) + \ldots$$

- For f to be smooth (in C^M) the filter weights must be smooth:

$$w_k \in C^M \qquad\qquad w_k^{(m)}(1) = w_{k+1}^{(m)}(0)$$

SIGGRAPH 2001

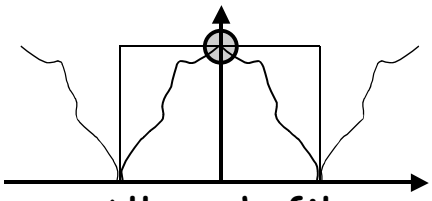
Frequency Considerations

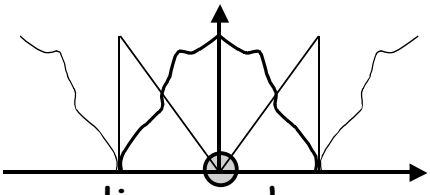
Accuracy

$$H^{(n)}(0) = j^n a_n(\tau)$$

Smoothness

$$\left|H(\omega)\right| \le \frac{C}{\left|\omega\right|^M}$$

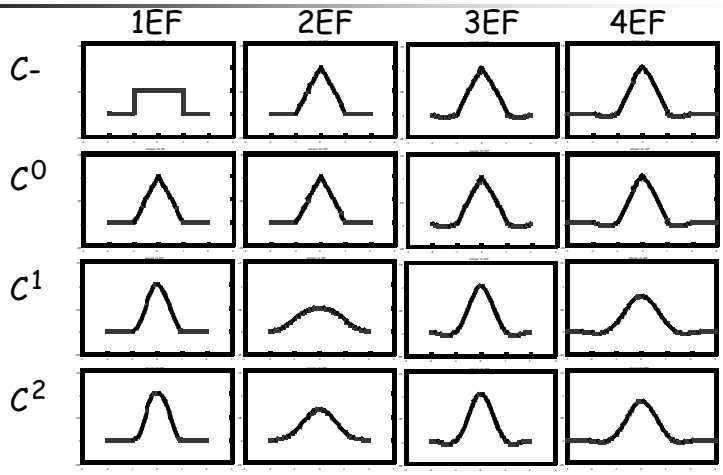




All such filters have linear phase.

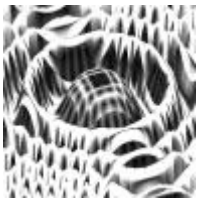
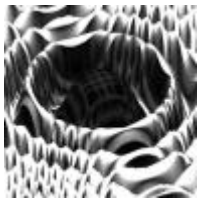
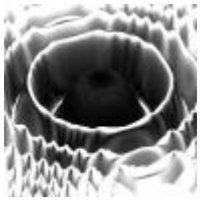
SIGGRAPH 2001

Interpolation filters

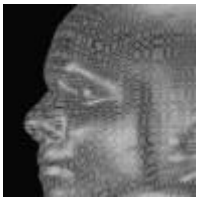
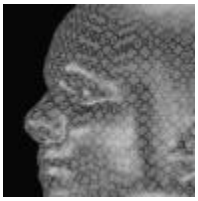


SIGGRAPH 2001

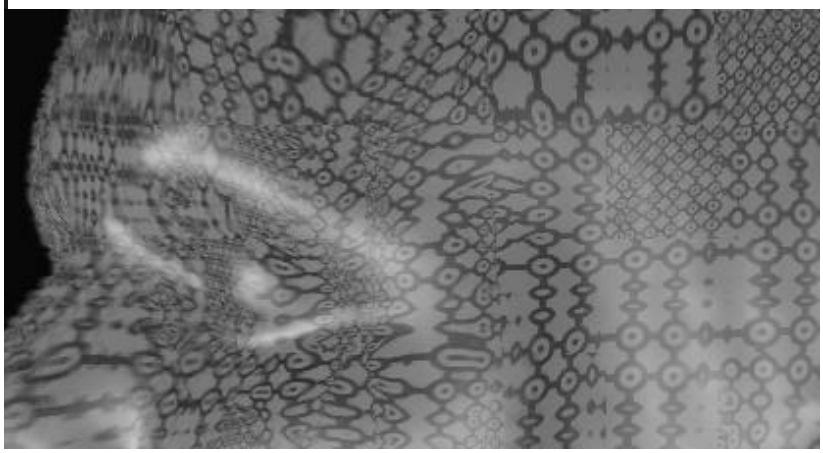
Results - Marschner/Lobb

	linear error derivative filter	cubic error derivative filter
discontinuous derivative filter		
C^0 continuous derivative filter		

Results - Pattern Mapping

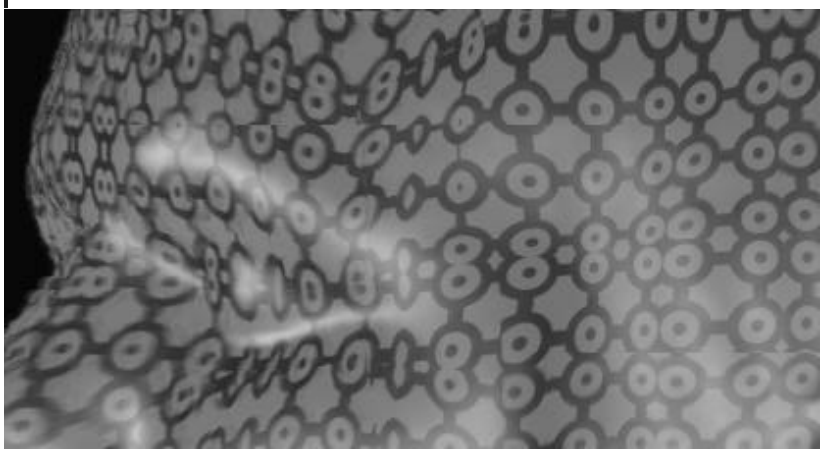
discontinuous linear error derivative filter	C^0 continuous quadratic error derivative filter	C^1 continuous quadratic error derivative filter
		

Results - Pattern Mapping



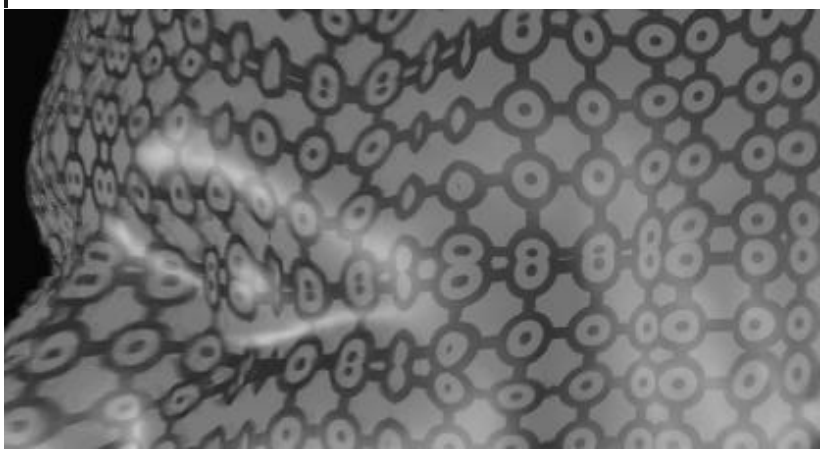
SIGGRAPH 2001

Results - Pattern Mapping



SIGGRAPH 2001

Results - Pattern Mapping



Derivative Example

2EF first derivative filter:

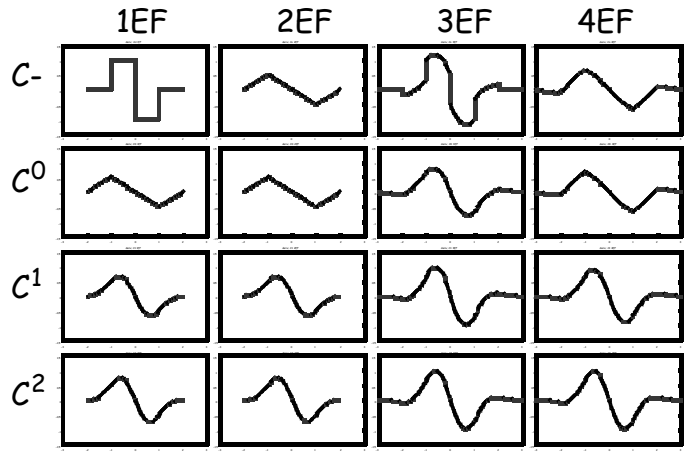
$$f_r(t) = a_0 f + a_1 f' + a_2 f'' + \ldots$$

$$a_0 = 0 = w_{-2} + w_{-1} + w_0 + w_1$$

$$a_1 = 1 = w_{-2}(2-\tau) + w_{-1}(1-\tau) + w_0(-\tau) + w_1(-1-\tau)$$

$$a_2 = 0 = w_{-2}(2-\tau)^2 + w_{-1}(1-\tau)^2 + w_0(-\tau)^2 + w_1(-1-\tau)^2$$

Derivative Filters



Summary

- To evaluate filters spatial domain approach yields constraints on filters
- Frequency domain approach allows
  - interpretation of designs
  - wise choice of parameters

Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering

Torsten Möller[1,3], Raghu Machiraju[1,3], Klaus Mueller[2], Roni Yagel[1,2,3]
[1]Department of Computer and Information Science
[2]Biomedical Engineering Center
[3]The Advanced Computing Center for the Arts and Design
The Ohio State University, Columbus, Ohio

ABSTRACT

We describe a new method for analyzing, classifying, and evaluating filters, which can be applied to interpolation filters, and derivative filters. Our analysis is based on the Taylor series expansion of a convolution sum and some assumptions on the behavior of the data function. As a result of our analysis, we derive the need and the method for normalization of derivative filter coefficients. As an example, we demonstrate the utilization of our methods to the analysis of the class of cardinal cubic filters. Since our technique is not restricted to interpolation filters, we can show that the Catmull-Rom spline filter and its derivative are the most accurate reconstruction and derivative filter among this class of filters. We show that the derivative filter has a much higher impact on the rendered volume than the interpolation filter. We demonstrate the use of these optimal filters for accurate interpolation and gradient estimation in volume rendering.

1 INTRODUCTION

Reconstruction of a continuous function and possibly its derivatives from a set of samples is one of the fundamental operations in visualization algorithms. In volume rendering, for instance, we must be able to interpolate the data set at arbitrary locations to evaluate the rendering integral. The gradient (or first derivative of the function) is important in classifying the volume and applying a proper illumination model [5][12].

Assumptions - We denote by $f(t)$ a continuous function (the *signal*) which is sampled into the discrete function $f_k = f(kT)$, where T is the sampling distance and k is an integer. In computer imaging $f(t)$ is not available; we have only f_k, which is the discrete image we need to manipulate. An important assumption we make is that the continuous signal f is sampled at or above the *Nyquist frequency* [17][21]. Inherent to this assumption is that the underlying function is bandlimited and hence analytic, i.e., all derivatives exist at all points. In fact, signals commonly found in volume visualization are bandlimited because, during the process of acquiring digital images, acquisition devices (e.g., cameras, scanners) perform a filtering operation and bandlimit the function. Images generated by numerical simulations of physical phenomena (common in disciplines such as computational fluid dynamics) are also bandlimited because, typically, robust numerical solutions can be obtained only if the algorithm incorporates a smoothing step. Finally, all rendering and scan-conversion algorithms, in order to provide antialiased images, typically employ a filtering step that bandlimits the image. Malzbender presents similar observations for volumes obtained through medical acquisition devices (e.g CT, MRI) [14]. Much has been written about the reconstruction of sampled data sets in the fields of signal processing [17] (1D data) and image processing [3][6] (2D data), applied numerical mathematics [1][10][21].

Motivation - Before resampling, we must reconstruct from f_k the continuous function, which we denote by $f_r^h(t)$. Here, h denotes the low-pass interpolation filter. Before we can classify or apply any shading method to our data set, we need to reconstruct the derivative of $f(t)$ from the known samples f_k. We denote the derivative of the continuous function $f(t)$ by $f'(t)$ and the reconstructed derivative by $f_r^d(t)$. Here, d denotes the high-pass derivative filter. The filters h and d are usually chosen without much thought towards the adverse effects of the filter performance. The trilinear and central difference filters are often used for the reconstruction of the underlying function and its derivative, because they are inexpensive. However, the use of the trilinear filter results in blurring and aliasing in the final image, while the application of the central difference filter results in the loss of fine details.

With recent advances in hardware it is now possible to consider the use of better, albeit computational expensive filters. As a result, there exists a need for quantitative and qualitative methods to evaluate the goodness of the interpolation and derivative filters. Quantitative methods are useful since they provide an error metric to compare and contrast filters. Also, they can lead to the selection of optimal filters. On the other hand, qualitative methods allow the classification of the filters into categories and may lead to the application of further metrics e.g perceptual.

For our evaluation methods, we also require that the function is included in the evaluation. Also, the evaluation should be conducted in the spatial domain instead of the more cumbersome frequency domain.

With this background and assumptions in mind we summarize, in Section 2, what previous research has been done in this field. In Section 3 we introduce our concept of Taylor series expansion of the convolution sum. Because of their importance, we single out the case of interpolation and derivative filters. In Section 4, we illustrate an application of our general methods to the group of cardinal cubic interpolation and derivative filters. In Section 5 we show some experimental results, and, in Section 6 we suggest steps for furthering this research. Finally, in Section 7, we summarize our findings.

[1] 395 Dreese Lab, 2015 Neil Ave, Columbus, OH 43210
{moeller | raghu | yagel@cis.ohio-state.edu}

[2] 260 Bevis Hall, 1080 Carmack Rd., Columbus, OH 43210
{klaus@chaos.bme.ohio-state.edu}

0-7803-3708-5/96 ..$4.00 © 1996 IEEE

2 RELATED WORK

Researchers have generally studied and evaluated filters in frequency domain. One of the earliest comparative studies of interpolation filters for image resampling was done by Parker et al [19]. They compared nearest neighbor, linear, cubic B-Spline, and two members of the class of cardinal cubic splines, through a discussion of their respective frequency spectra. They found the Catmull-Rom spline to be superior.

A thorough study of cardinal cubic splines in frequency domain was performed by Park and Schowengerdt [18]. They found that the optimal interpolation filter of this class highly depends on the signal to which it will be applied. For most applications, the parameter α will be around -0.6 or close to the Catmull-Rom spline, for which the parameter is -0.5. Keys [11] showed that the latter filter is optimal, within the class of cardinal splines, in the sense that it interpolates the original function with the smallest spatial error. By using a Taylor series expansion of the convolution sum, he found that the Catmull-Rom spline interpolation filter has an error proportional to the cube of the sampling distance.

Mitchell and Netravali [16] introduce a more general class of cubic splines, which we refer to as BC-splines. Cardinal cubics are a subclass of these cubics. Mitchell and Netravali conducted a study of more than 500 sample images, classifying the parameter space into different regions of dominating image artifacts such as blurring, ringing, and anisotropy. They also found, by using a Taylor series expansion, that filters in which $B + 2C = 1$ are most accurate numerically and have an error proportional to the square of the sampling distance. Neither Keys nor Mitchell and Netravali approximate the actual error of their filters.

A recent comparative study by Marschner and Lobb [15] proposes the use of different metrics for different image artifacts. Specifically, they introduce metrics in the frequency domain to measure the smoothing, postaliasing, and overshoot of an interpolation filter and found the windowed *Sinc* filter to behave the best. Unfortunately, their metrics do not depend on the actual function to be reconstructed, an issue that Park and Schowengerdt found to be crucial for frequency analysis.

All the aforementioned approaches neglect to take derivative filters into account in their studies, a precondition to compare rendered and shaded images. A good survey of existing derivative filters can be found in the paper by Dutta Roy and Kumar [7] in which they describe the design of maximal linear filters in frequency domain. Their filters can be adapted to various frequency ranges, an important consideration for practical applications.

Goss [9] extends the idea of windowed filters to derivative filters. He uses a Kaiser window for the ideal derivative filter, which is shown, e.g. in [2], to be the derivative of the *Sinc* filter and which we denote as the *Cosc* filter, but does not explain why this windowing is necessary and why a Kaiser window will work reasonably well. The work by Bentum [2] uses the cardinal cubics as a basis to develop derivative filters. Although he shows different plots of these filters for different parameters, he does not analytically compare these different filters.

While most of the existing research concentrates on frequency analysis, we believe that the spatial analysis is just as important. If the local error can be kept small, then the effect of image artifacts also diminishes. In fact, we find that the results of Keys' spatial analysis are nearly identical to the results of the frequency analysis done by Park and Schowengerdt.

In this work we develop tools for the spatial analysis of interpolation and derivative filters. We show the importance of the normalization of the filter coefficients and how this step is performed. Specifically for the class of cardinal splines, we derive known and new results in Section 4 using our new spatial method.

3 TAYLOR SERIES EXPANSION OF THE CONVOLUTION SUM

To reconstruct a continuous function $f(t)$ or its derivative $f'(t)$ from a set of sampled points f_k, we compute a weighted average of these samples. This process is also known as convolution of the sampled signal with a filter which determines the weights. If we convolve the samples with a continuous interpolation filter h, we reconstruct the original function $f(t)$. Similarly, if we convolve the samples with a continuous derivative filter d, we can reconstruct the nth derivative of the original function. We denote the result by $f_r^w(t)$, where w denotes the kind of filter used. This can be written as:

$$f_r^w(t) = \sum_{k=-\infty}^{\infty} f_k \cdot w(\frac{t}{T} - k) \tag{1}$$

Now f_k represents the samples of the original function $f(t)$ at the values kT, where T is the sampling distance. Since we assume that all the derivatives of $f(t)$ exist, we can expand $f_k = f(kT)$ in a Taylor series in f around t. Therefore, we write:

$$f_k = f(kT) = \sum_{n=0}^{N} \frac{f^{(n)}(t)}{n!}(kT-t)^n + \frac{f^{(N+1)}(\xi_k)}{(N+1)!}(kT-t)^{(N+1)}$$

where $\xi_k \in [t, kT]$. Substituting this Taylor series expansion into Equation (1) and reordering the terms according to the derivatives of $f(t)$, we can rewrite Equation (1) as:

$$f_r^w(t) = \sum_{n=0}^{N} a_n^w(t) f^{(n)}(t) + r_N^w(t) \tag{2}$$

where the coefficients $a_n^w(t)$ and the remainder term $r_N^w(t)$ are:

$$a_n^w(t) = \frac{1}{n!}\sum_{k=-\infty}^{\infty}(kT-t)^n w(\frac{t}{T}-k)$$

$$r_N^w(t) = \frac{1}{(N+1)!}\sum_{k=-\infty}^{\infty} f^{(N+1)}(\xi_k)(kT-t)^{(N+1)} w(\frac{t}{T}-k)$$

This gives us the impression that the values of the coefficients a_n^w and the remainder term r_N^w depend on t. This is somewhat misleading. Let us first have a look at the coefficients a_n^w. For practical reconstructions of signals, we do not use an infinitely long filter. Therefore, a filter w has a finite filter length that we call M. The filter w is defined to be zero outside the interval $[-M, M]$. Next, we observe that we are evaluating the filter w at points exactly one unit length apart. These weights are applied to an n-th degree polynomial $(kT-t)^n$ sampled also at points exactly T apart, centered at t. Therefore, it makes sense to rewrite t in terms of an offset τ from T, written as:

$$t = (i+\tau)T, \text{ where } 0 \le \tau < 1, \text{ and } i \in Z$$

Therefore, the coefficients a_n^w can be expressed in terms of this offset:

$$a_{n,i,T}^w(\tau) = a_n^w((i+\tau)T) = \frac{1}{n!}\sum_{k=i-M}^{i+M}((k-i)T-\tau T)^n w(\tau-(k-i))$$

which can be simplified to:

$$a^{w}_{n,i,T}(\tau) = \frac{T^n}{n!} \sum_{k=-M}^{M} (k-\tau)^n w(\tau - k) \quad (3)$$

Here, we clearly see that a^{w}_{n} depends not on t itself, but rather on the distance to the next sampling point (expressed by τ). This is because τ tells us how 'far away' a reconstructed value is from an already known sampled value. Therefore, we can quantify how 'hard' the reconstruction process really is. Since $a^{w}_{n,i,T}$ does not depend on i and, T is set by the data acquisition step (and therefore cannot be changed during the reconstruction process), we will drop the appropriate subscripts in the expression for a^{w}_{n}.

If we do the same analysis for the remainder term r^{w}_{N}, we find:

$$r^{w}_{N,i}(\tau) = \frac{T^{(N+1)}}{(N+1)!} \sum_{k=-M}^{M} f^{(N+1)}(\xi_{k,i})(k-\tau)^{(N+1)} w(\tau - k) \quad (4)$$

where $\xi_{k,i} \in [t, (k+i)T]$. Finally, the convolution sum in Equation (1) can be expressed as:

$$f^{w}_{r}(t) = \sum_{n=0}^{N} a^{w}_{n}(\tau) f^{(n)}(t) + r^{w}_{N,i}(\tau)$$

Our objective is to quantify and classify the error occurring during the reconstruction process. We can do this by comparing the a^{w}_{n} and r^{w}_{N} of various filters w. The principal idea is to choose the largest N such that all the coefficients a^{w}_{n} evaluate to zero, with the exception of a^{w}_{0} for interpolation filters and a^{w}_{m} for m-th order derivative filters. This coefficient should evaluate to one, since we want to reconstruct the continuous function or its m-th order derivative respectively. Choosing the value of N in this way, the reconstruction error is simply the remainder term r^{w}_{N}. We observe that the coefficients depend solely on the underlying filter w. This leads us to a conceptually ideal way to compare and classify different filters. We can put all the filters characterized by the same N in one class. The reconstruction error r^{w}_{N} is of the order $O(T^{N+1})$. That means that for typical applications, $|r^{w}_{N_1}|$ will be smaller than $|r^{w}_{N_2}|$, iff $N_1 > N_2$. Therefore, in general, we prefer filters in a class with largest N. The filters in class $(N-1)$ we simply call N-th degree error filters (N-EF) to comply with standard nomenclature in numerical mathematics. We can further distinguish among filters in the same class using their absolute errors $|r^{w}_{N}|$.

This classification is very important and should strongly influence the choice of a filter for a given application. A N-EF will reconstruct a polynomial of $(N-1)$th or lower degree (the original function as well as its derivative) without any errors. In many applications, the underlying data can be sufficiently modeled with lower degree polynomials. Therefore, a 3EF or 4EF may be sufficient.

This classification scheme is important in determining the sufficient and necessary resolution of voxelization (discretization) algorithms. One restriction might occur, when the researcher cannot change the resolution, e.g. the resolution of an MRI scanner. In such cases, one needs to consider the error terms quantitatively only, because the placement of a filter in a N-EF group depends on the *asymptotic* behavior of the filter error. Therefore, one can find examples where, for some T, a specific filter in the 2EF group will perform better than a specific filter of the 3EF group.

3.1 Continuous Interpolation Filters

In the case of an interpolation filter h, we need to require that a^{h}_{0} be exactly one. This requirement is also recommended by Mitchell and Netravali [16]. To achieve this, we simply normalize the filter coefficients by dividing them by a^{h}_{0}. Without this normalization step, we cannot rely on the accuracy of the interpolated function.

3.2 Continuous Derivative Filters

As with interpolation filters, we also need to normalize a derivative filter d. For simplicity, we will concentrate on the derivative filter for the reconstruction of the first derivative $f'(t)$ of the original function. Higher order derivative filters can be treated similarly.

We are reconstructing the derivative of the function instead of the function itself and therefore need to set the coefficient in front of $f'(t)$ to one. That means we must normalize it by dividing the filter coefficients by a^{d}_{1}. This is less known, yet very important for a reliable normal estimation. The reconstruction of derivatives is still more complicated than this; we also need to require that a^{d}_{0} is zero! In fact, if this condition does not hold, the result is useless. As an example, let us examine the *Cosc* filter, the ideal derivative filter. It has been shown that the ideal derivative filter is simply the derivative of the ideal interpolation filter *Sinc* [2]. Therefore, we have

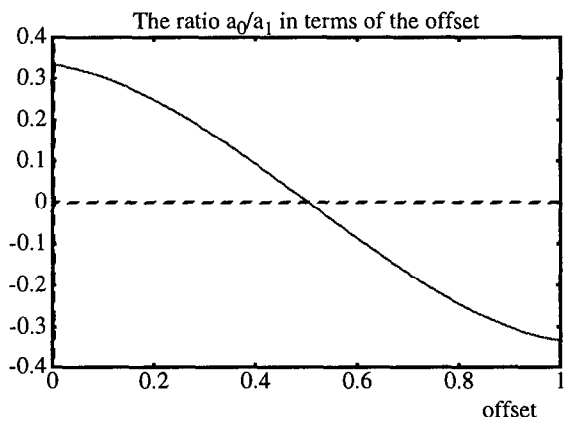


FIGURE 1. We computed the normalized coefficient a^{d}_{0} in terms of the offset to the next sampled value. We find it only to be zero for $\tau = 0.5$.

$$Cosc(t) = Sinc'(t) = \begin{cases} 0 & t = 0 \\ \frac{1}{t}\left(\cos(\pi t) - \frac{\sin(\pi t)}{\pi t}\right) & t \neq 0 \end{cases}$$

The *Cosc* filter (like the *Sinc* filter) is an infinite filter and therefore not applicable. In the case of the *Sinc* filter, we could use a truncated *Sinc* filter, which is equal to the *Sinc* filter for all $|t| \leq M$ and zero outside of this interval. In the case of the *Cosc* filter, we cannot simply use a truncated *Cosc* filter, since a^{w}_{0} will not be zero. To demonstrate this, let us set M to three, which results in six filter weights for the reconstruction. This is a rather expensive filter. In Figure 1 we plotted the normalized coefficient $a^{d}_{0}(\tau)/a^{d}_{1}(\tau)$. We found this function to be varying between -0.4 and 0.4. Notice the behavior of this specific truncated *Cosc* filter on a linear function as shown in Figure 2. We would expect a function close to constant one (the derivative of the linear function $f(x) = x$), but instead we see a linearly increasing error on $f(t)$. In order to get a correct result, we need to subtract $\left(a^{d}_{0}(\tau)/a^{d}_{1}(\tau)\right) f(t)$ from the reconstructed value. That means we actually need to reconstruct the original function $f(t)$ in order to compute its derivative. But this would require another convolution with an interpola-

tion filter, which would be inefficient. A proposed way around this problem is to window the truncated *Cosc* filter [9].

We conclude that a careful analysis of a filter is necessary before its use.

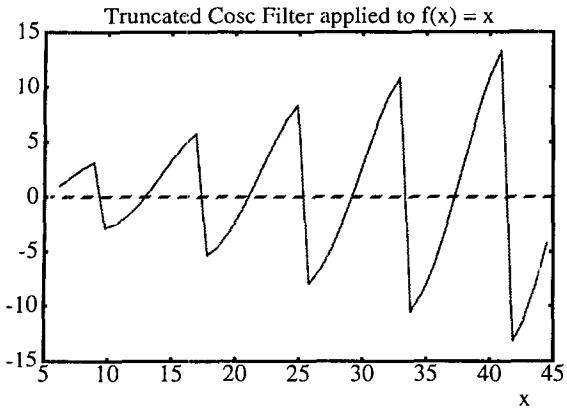


FIGURE 2. We truncated the Cosc filter at $M=3$. Since the coefficient a_0^d is not zero, we end up with artifacts that make the filter useless. (The expected result is the constant function one.)

3.3 An Approximation of the Error r_N^w

The error as computed in Equation 4 is more a theoretical result than a practical one, because we do not know the $\xi_{k,i}$. To get a fair idea about the behavior of the actual error, we suggest an approximation of r_N^w. In practical applications the length M of the used filter w is usually small, since a larger filter width M results in reduced efficiency. Therefore, we conclude that the interval $[(i-M)T, (i+M)T]$, in which all the $\xi_{k,i}$ are to be found, is relatively small. In addition, for MRI and CT data sets we found a behavior of the data sets in the frequency spectrum corresponding to the function $1/\omega^r$ for small r. Therefore, we do not expect a lot of high frequencies, which would result in a fast changing function within a short interval. It is reasonable to assume that especially $f^{(N+1)}(t)$ will not change much on a small interval of length $2M$. We conclude, that:

$$r_{N,i}^w(\tau) \le \left(\max_{\xi \in [(i-M)T,\ (i+M)T]} (f^{(N+1)}(\xi)) \right) \left| \frac{T^{(N+1)}}{(N+1)!} \sum_{k=-M}^{M} (k-\tau)^{(N+1)} w(\tau-k) \right|$$

or,

$$r_{N,i}^w(\tau) \le \left(\max_{\xi \in [(i-M)T,\ (i+M)T]} (f^{(N+1)}(\xi)) \right) \left| a_{N+1}^w(\tau) \right| \quad (5)$$

If we can approximate the $(N+1)$st derivative of the underlying function, then we can approximate the actual error. Even if this is not possible, we can at least compute a_{N+1}^w to get an idea about the scale of the error. How well this error-bound approximates the actual error can be seen in Figure 3, where we used the derivative of the cubic interpolation filter to compute the derivative of a quintic polynomial. People usually model practical data sets with cubic polynomials locally. We have chosen a quintic polynomial since it is a supergroup of cubic polynomials and has higher variations and faster changing derivatives.

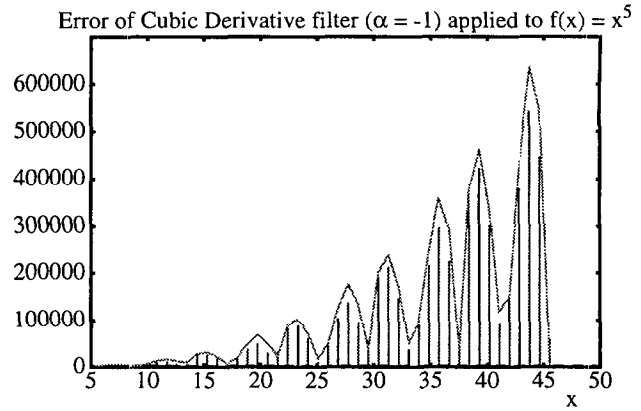


FIGURE 3. The absolute error of the cubic derivative filter with the parameter α set to -1 applied to the polynomial of 5th degree. Also we have *computed* the error bound according to Equation 5 using the result from Table 1 in Section 4.2. We can see that our error bound is very tight and almost exact. The actual error is shown as error bars and the computed error-bound is drawn as a connected curve.

4 OPTIMAL CUBIC FILTER

Now, let us apply the methods we developed in Section 3 to the group of cubic filters. The group of cubic filters are described as:

$$h(t) = \begin{cases} (\alpha+2)|t|^3 - (\alpha+3)|t|^2 + 1 & 0 < |t| < 1 \\ \alpha|t|^3 - 5\alpha|t|^2 + 8\alpha|t| - 4\alpha & 1 < |t| < 2 \\ 0 & 2 < |t| \end{cases}$$

This class of filters is derived by Keys [11]. He shows that for $\alpha = -0.5$ this filter is a 3EF. He also uses a Taylor series expansion to derive this result. In Section 4.1 we will derive this result again, to demonstrate the power of our method. Bentum [2] developed a continuous derivative filter originating from the cubic interpolation filter. This filter is simply the derivative of the cubic interpolation filter, and can be written as:

$$d(t) = \begin{cases} -3\alpha t^2 - 10\alpha t - 8\alpha & -2 < t < -1 \\ -3(\alpha+2)t^2 - 2(\alpha+3)t & -1 < t < 0 \\ 3(\alpha+2)t^2 - 2(\alpha+3)t & 0 < t < 1 \\ 3\alpha t^2 - 10\alpha t + 8\alpha & 1 < t < 2 \\ 0 & 2 \le |t| \end{cases}$$

In section Section 4.2 we show that for this derivative filter as well, $\alpha = -0.5$ is optimal and produces a 2EF.

4.1 The cubic interpolation filter

The cubic filter has a window size of two, i.e. an overall extent of four. Therefore, we have four weights to consider. These are:

$$
\begin{aligned}
h(\tau) &= (\alpha+2)\tau^3 - (\alpha+3)\tau^2 + 1 \\
h(\tau-1) &= -(\alpha+2)\tau^3 + (2\alpha+3)\tau^2 - \alpha\tau \\
h(\tau+1) &= \alpha\tau^3 - 2\alpha\tau^2 + \alpha\tau \\
h(\tau-2) &= -\alpha\tau^3 + \alpha\tau^2
\end{aligned}
\tag{6}
$$

We are using these filter weights to compute the coefficients a_n^h from Equation 3. The results are summarized in Table 1.

TABLE 1. Coefficients for the cubic interpolation filter

a_0^h	1
a_1^h	$T(2\alpha+1)\tau(1-2\tau)(\tau-1)$
a_2^h	$2T^2(2\alpha+1)\tau^2(\tau-1)^2$
a_3^h (evaluated only for $\alpha = -0.5$)	$\frac{T^3}{6}\tau(1-2\tau)(1-\tau)$

For a_0^h we compute their sum and find that this coefficient is exactly one. This is an important result, for it tells us that cubic interpolation filters do not need to be normalized. This saves many computations and makes this class of filters more efficient and more attractive for practical applications.

Considering a_1^h we find, as Keys [11] did, that the cubic filter with $\alpha = -0.5$, also known as the Catmull-Rom-Spline, has the best behavior in terms of numerical accuracy and makes this cubic interpolation filter at least a 2EF. Additionally, we find, that in the special case of $\tau = 0.5$ the cubic interpolation filter is, for any choice of α, at least a 2EF too. This means we interpolate exactly in the middle of two sampling points. If we neither choose α to be -0.5 nor interpolate in the middle of two sampling points, the cubic interpolation filter represents a linear order filter (1EF) and a_1^h substituted in Equation 5 represents an error bound for this class of cubic filters.

For further analysis in the case of $\alpha = -0.5$ or $\tau = 0.5$, we evaluate a_2^h. For $\tau = 0.5$ it will not be zero, and therefore substituted in Equation 5 presents an error approximation. In the case of $\alpha = -0.5$, a_2^h will evaluate to zero and in order to approximate the error, we need to analyze a_3^h.

Computing a_3^h for $\alpha = -0.5$, we find that this coefficient is not zero and therefore, substituted in Equation 5, represents an error-bound for the Catmull-Rom cubic filter. Thus this filter is a 3EF filter. We conclude that the choice $\alpha = -0.5$ results in an optimal cubic interpolation filter. This filter is optimal in terms of numerical accuracy. Keys presented this same result, except, that he did not provide an error-bound and did not mention the special case in which the offset τ is set to -0.5.

With τ set to -0.5, we find that also a_3^h evaluates to zero and we get an even better filter. This special discrete interpolation filter has the following filter coefficients:

$$
\begin{aligned}
h(\pm 0.5) &= 9/16 \\
h(\pm 1.5) &= -1/16
\end{aligned}
$$

For this filter, we find similarly $a_4^h(0.5) = -3T^4/128$, which, again, we plug into our error approximation of Equation 5. Therefore, when both conditions hold ($\tau = 0.5$ and $\alpha = -0.5$), we have a 4EF filter.

4.2 The cubic derivative filter

Bentum [2] introduced the cubic derivative filter, but did not provide an analysis nor an analytical comparison of its performance for different values of the parameter α. Although this filter is really just a quadratic filter, we prefer to call it 'cubic derivative' filter, since its parent is the cubic interpolation filter, which is a very well known and commonly used class of filters. The four relevant weights in this case are:

$$
\begin{aligned}
d(\tau) &= 3(\alpha+2)\tau^2 - 2(\alpha+3)\tau \\
d(\tau-1) &= -3(\alpha+2)\tau^2 + 2(2\alpha+3)\tau - \alpha \\
d(\tau+1) &= 3\alpha\tau^2 - 4\alpha\tau + \alpha \\
d(\tau-2) &= -3\alpha\tau^2 + 2\alpha\tau
\end{aligned}
$$

These are the derivatives of the weights for the interpolation filter in Equation 6. As in the previous section, we compute the coefficients in Table 2 using Equation 3.ex

To compute a_0^d we compute the sum of these weights and find that a_0^d is indeed zero. In Section 3.2 we found that this is a requirement for good derivative filters, and we also saw that it is not obvious this condition holds. Here, we note once again that cubic filters are a well-defined class of filters. As shown in Section 3.2, we need to normalize the other coefficients by a_1^d. Note that for $\alpha = -0.5$, the normalization step is a simple division by the sample distance T. This again saves time and the convolution becomes more efficient, making this class of filters useful for fast, accurate volume rendering. Especially in the case of $T = 1$ this division does not need to be performed. In the given literature we could not find a mention of this necessary normalization step. We also want to point out that in the case of $a_1^d = 0$, we really have a filter which computes a higher order derivative. This is the case, e.g. if $\alpha = -1.5$ and $\tau = 0.5$. We not only find a_1^d to be zero but also a_2^d. That means that we actually recover at least the third derivative of the underlying function. A more thorough analysis, where one would compute a_3^d or higher, is beyond the scope of this paper.

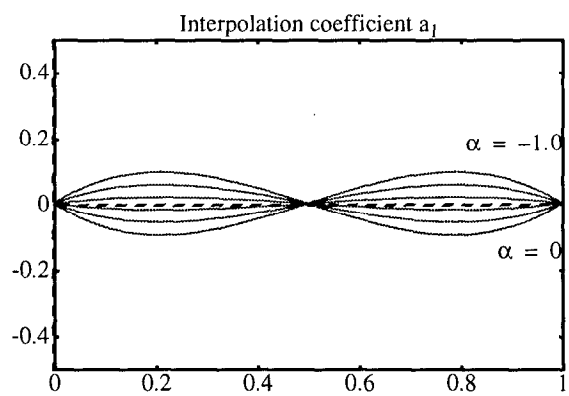


FIGURE 4. Here we plot the coefficient a_1^h of the cubic interpolation filter for varying α. α is 0, -0.2, -0.4, -0.6, -0.8 and -1. We set T to one.

As is the case for the cubic interpolation filter, we discover that for $\alpha = -0.5$ or $\tau = 0.5$ the coefficient a_2^d is zero, leading us to a derivative filter of higher order error. If none of these two conditions hold, we substitute the normalized a_2^d into Equation 5

and find an error bound. In the case that $\alpha = -0.5$, we substitute a_3^d/T into Equation 5 for an error bound. Since we normalize by T, this filter is a 2NF.

Again, we conclude that for the cubic derivative filter $\alpha = -0.5$ is the optimal filter in terms of numerical accuracy. Note that we lose one degree of accuracy for the first derivative filter. The cubic interpolation filter is a 3EF. However, the cubic derivative filter is only a 2EF.

In the case that $\alpha \neq -0.5$, we find that both filters are linear error filter (1EF). For a better perspective we plotted the error coefficients of the error term in Equation (5). For the interpolation filter we plotted a_1^h for different α in Figure 4 and for the derivative filter we plotted a_2^d/a_1^d in Figure 5. We find that the actual error term is larger in magnitude for the derivative filter.

TABLE 2. Coefficients for the cubic derivative filter.

a_0^d	0
a_1^d	$T\left(1 + (2\alpha+1)\left(-6\tau^2 + 6\tau - 1\right)\right)$
a_2^d	$3T^2(2\alpha+1)\tau(1-\tau)(1-2\tau)$
a_3^d (evaluated only for $\alpha = -0.5$)	$\frac{T^3}{6}\left(6\tau^2 - 6\tau + 1\right)$

These are important observations, for in order to obtain as reliable data for derivatives as for the interpolated signal, we need to apply more sophisticated filters. Furthermore, if we use the same α for interpolation and derivative filter as Marschner and Lobb [15], the error introduced by the derivative filter is larger.

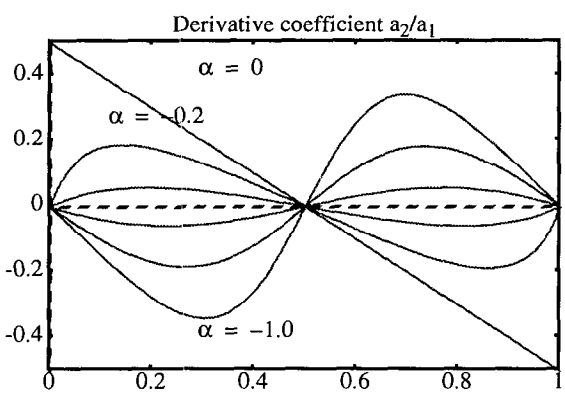


FIGURE 5. We plot the normalized coefficient a_2^d/a_1^d of the cubic derivative filter for varying α. α is 0, -0.2, -0.4, -0.6, -0.8 and -1. We set T to one.

5 EXPERIMENTAL RESULTS

For our experiments we used an analytic data set and a MRI data set. The synthetic data set is the same function as the one used by Marschner and Lobb [15], sampled into a 40 x 40 x 40 volume lattice. A high definition ray caster, sampling the volume at a step size of 0.05 voxel lengths, was employed to display the function's opaque isosurface at 0.5 of the maximum function value. In contrast to the images given in [15], Figure 6 focuses on the center section of the function, where the distinct effects induced by the various interpolation-derivative filter pairings are most apparent.

In Figure 6.1 through Figure 6.9 we rendered this data set with varying α for interpolation and derivative filters. Along the rows we decrease α for the interpolation filter from top ($\alpha = -0.2$) to bottom ($\alpha = -1$). Along the columns we decrease α for the derivative filter from left ($\alpha = -0.2$) to right ($\alpha = -1$). We find that the differences between rows are not as striking as the differences between columns. That demonstrates our findings in Section 4.2, where we concluded that the derivative filter has more influence on image quality than the interpolation filter. In order to be able to compare interpolation filters, we suggest to fix the derivative filter, preferably one with error that is negligeable compared to the errors in the interpolation filter.

The second data set is an MRI of a human head, also used by Bentum [2]. Here, we fixed the interpolation filter to the optimal cubic filter and varied α for the derivative filter. In Figure FIGURE 7..1 through Figure FIGURE 7..3, we find that the best image is achieved when α is set to -0.5. That is exactly what we expected from our analysis in Section 4.2. Figure FIGURE 8..1 through Figure FIGURE 8..3 show the same set of filters applied to a small section of the brain and rendered from a close-up view.

6 FUTURE GOALS

In many applications, especially volume rendering, we want to both reconstruct the underlying function and/or its derivative and also resample it on a new grid. Therefore, it is necessary to study the overall error expressed in the L_2 error norm. We are working on developing better tools to study this error. Eventually, we want to come up with techniques similar to the ones presented in this paper that will allow us to classify different filters and also to quantify them, efficiently computing their L_2 error.

In terms of filter design, we can use our tools to design filters of arbitrary order. Setting the coefficients in Equation 2 to zero for a given N, we end up with N linear equations for $2M+1$ coefficients. We can solve this linear system, matching N and M appropriately, and we find a NEF. It would be interesting to study how this filter behaves in terms of the offset τ, for it would enable us to construct filters of arbitrary accuracy. Similar to the adaptive filter design of Machiraju et al [13], we can use these different filters adaptively in different areas of the function. Knowing the error, caused by convolving a particular filter with a particular application, we want to find ways to adapt the type of filter we use, according to a given error tolerance.

We plan to extend the techniques describes in Section 3.2 for higher order derivatives and to the evaluation of filters others than the cardinal cubic splines, such BC splines [16] and other optimal filters [4].

It is also important to study the behavior of different filters, when applied to rapidly changing functions. (Such behavior could be caused by noise.) Our analysis remains valid, except for the error estimation in Equation 5, where we assumed slowly changing functions.

7 SUMMARY

In this paper we applied a Taylor series expansion to the convolution sum. This resulted in an alternative representation of the convolution sum which lead to a qualitative and quantitative comparison of both reconstruction and derivative filters. We found that

the normalization of the filter coefficients is important for accurate reconstruction.

We then applied these techniques to the analysis of the class of cardinal cubic splines. We derived several special filters, which are numerically optimal within this class. Especially, we concentrated our efforts on interpolation and derivative filters and found that, when both are applied to a function, the error introduced by derivative filters are more significant than those caused by interpolation filters.

We expect the techniques developed here to be applicable to the design and evaluation of other reconstruction and high order derivative filters.

ACKNOWLEDGMENTS

We thank Ed Overman of the Department of Mathematics for providing useful background information and important references. We thank Wayne Carlson and the Advanced Center for Arts for the use of their computing facilities and Mark Fontana for reviewing the manuscript. This project was partially supported Department of Defense USAMRDC 94228001, and by the Advanced Research Projects Agency Contract DABT63-C-0056.

8 REFERENCESS

[1] Baker G., Overman E., "The Art of Scientific Computing", Draft Edition, The Ohio State University Bookstore. 1995

[2] Bentum M.J., "Interactive Visualization of Volume Data", Ph.D. Thesis, University of Twente, Enschwede, The Netherlands, 1995.

[3] Bracewell, R.N., *Two Dimensional Imaging*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1995.

[4] Deslauriers G., Dubuc S., "Symmetric Iterative Interpolation Processes, *CONAP*, 5(1):49-68,1989.

[5] Drebin R.A., Carpenter L., Hanrahan P., "Volume Rendering", *Computer Graphics*, 22(4):51-58, August 1988.

[6] Dudgeon D.E., Russell M.M., *Multidimensional Digital Signal Processing*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1984.

[7] Dutta Roy S. C. and B. Kumar, "Digital Differentiators", in *Handbook of Statistics*, N. K. Bise and C. R. Rao eds., Vol. 10:159-205, 1993.

[8] Glassner A., *Principles of Digital Image Synthesis*, Morgan Kaufmann, 1995.

[9] Goss M.E., "An Adjustable Gradient Filter for Volume Visualization Image Enhancement", *Proceedings of Graphics Interface'94*:67-74, Toronto, Ontario, 1994.

[10] Hamming R.W., *Numerical Methods for Scientists and Engineers*, Dover Publications Inc., Mineola, N.Y., 1986.

[11] Keys R.G., "Cubic Convolution Interpolation for Digital Image Processing", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29(6):1153-1160, December 1981.

[12] Levoy M., "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, 8(5):29-37, May 1988.

[13] Machiraju R., Yagel R., "Reconstruction Error Characterization and Control: A Sampling Theory Approach", submitted for publication *IEEE Transactions on Computer Graphics and Visualization*.

[14] Malzbender T., "Fourier Volume Rendering", *ACM Transactions on Computer Graphics*, 12(3):233-250, July 1993.

[15] Marschner S.R. and Lobb R.J., "An Evaluation of Reconstruction Filters for Volume Rendering", *Proceedings of Visualization '94*, IEEE CS Press:100-107, October 1994.

[16] Mitchell D.P. and Netravali A.N., "Reconstruction Filters in Computer Graphics", *Computer Graphics*, 22(4):221-228, August 1988.

[17] Oppenheim A.V., Schafer R.W., *Discrete-Time Signal Processing*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1989.

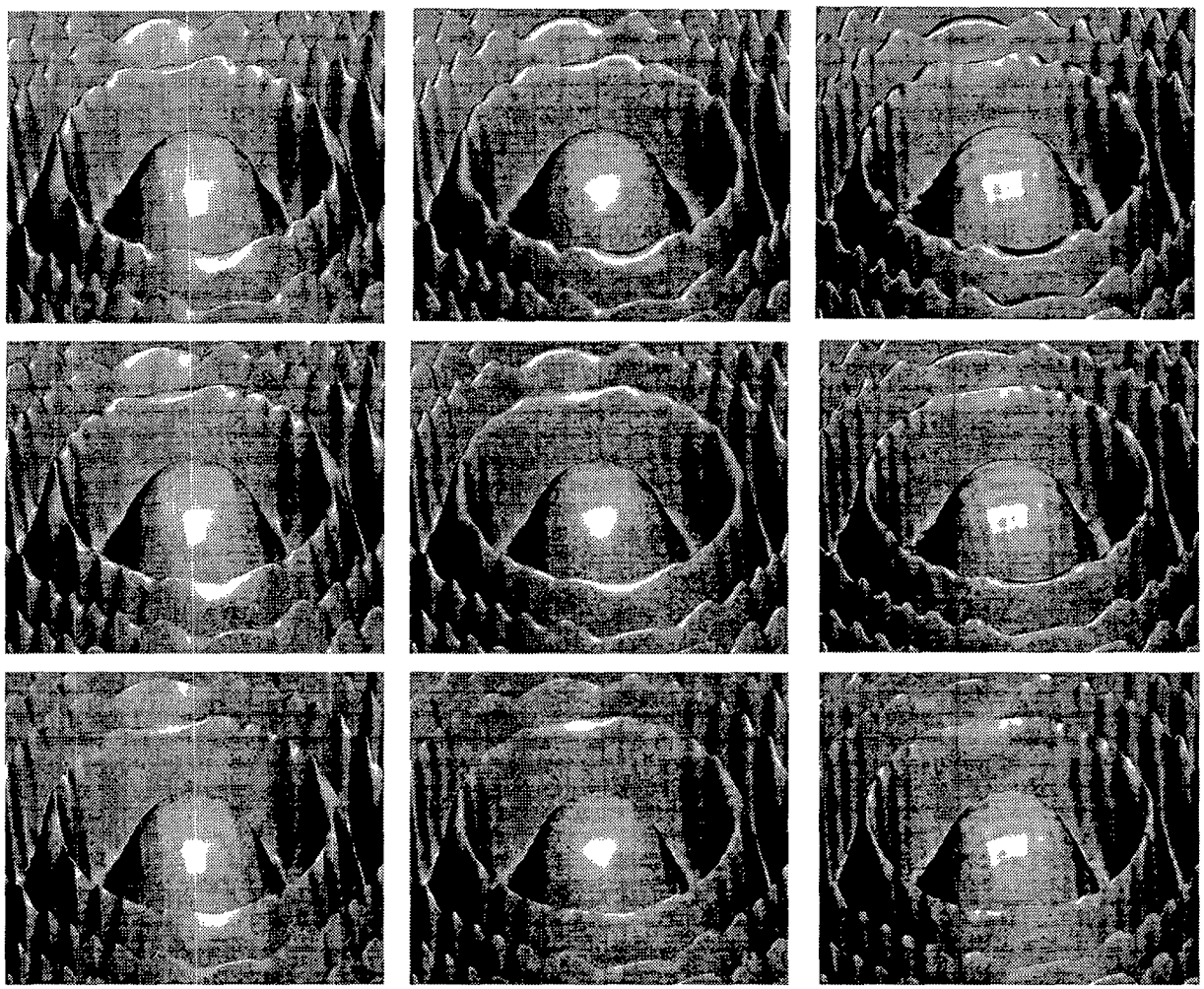
[18] Park S.K., Schowengerdt R.A., "Image Reconstruction by Parametric Cubic Convolution", *Computer Vision, Graphics, and Image Processing*, 23:258-272, 1983.

[19] Parker, J.A., Kenyon, R.V., Troxel, D.E., "Comparison of Interpolating Methods for Image Resampling", *IEEE Transactions on Medical Imaging*, MI-2(1):31-39, March 1983.

[20] Simoncelli E.P., "Design of Multi-Dimensional Derivative Filters", presented at the *IEEE International Conference on Image Processing*, Austin TX, November 1994. Also found at http://www.cis.upenn.edu/~eero/publications.html.

[21] Stenger F., *Numerical Methods Based on Sinc and Analytic Functions*, Springer Verlag, 1993.

Derivative: $\alpha = -0.2$ | Derivative: $\alpha = -0.5$ | Derivative: $\alpha = -1.0$



Interpolation: $\alpha = -0.2$

Interpolation: $\alpha = -0.5$

Interpolation: $\alpha = -1.0$

FIGURE 6. Marschner Image (3x3)

FIGURE 7. Head 1,2,3

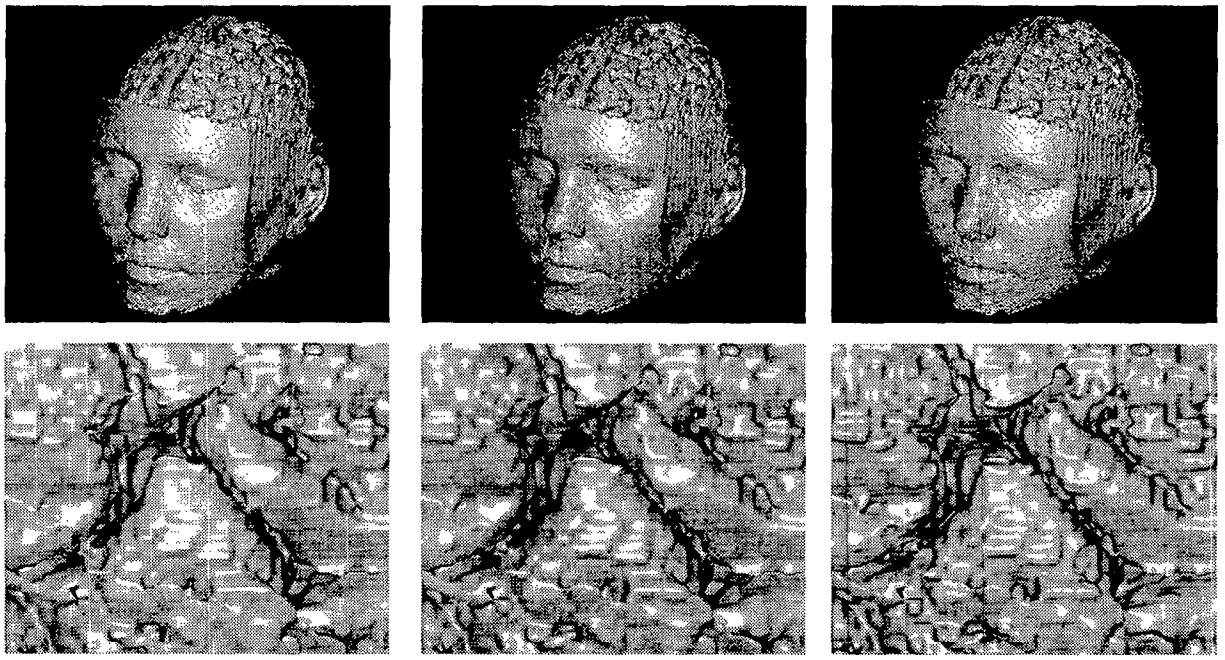


FIGURE 8. Brain 1, 2, 3

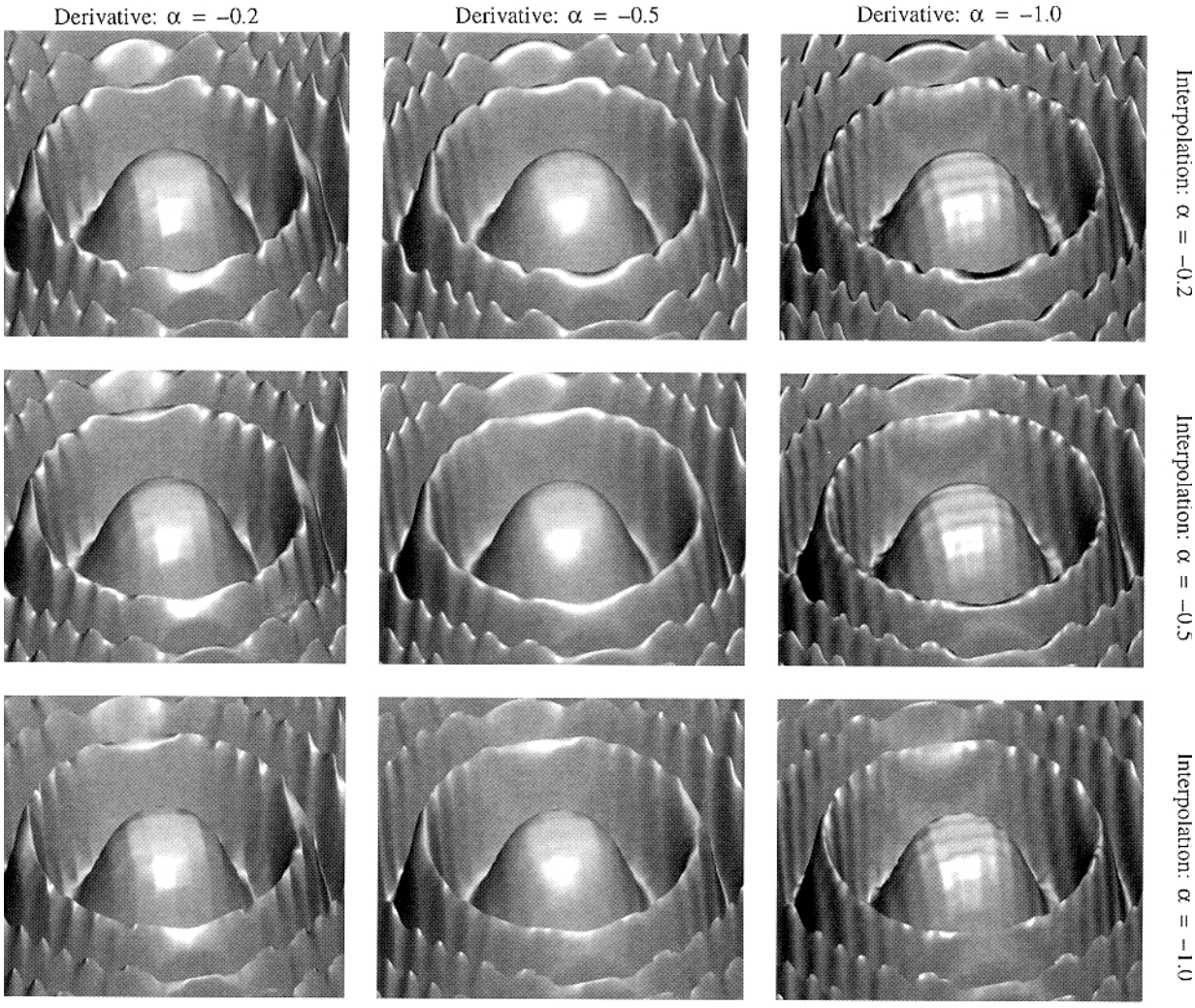


FIGURE 6. Marschner Image (3x3)

FIGURE 7. Head 1,2,3

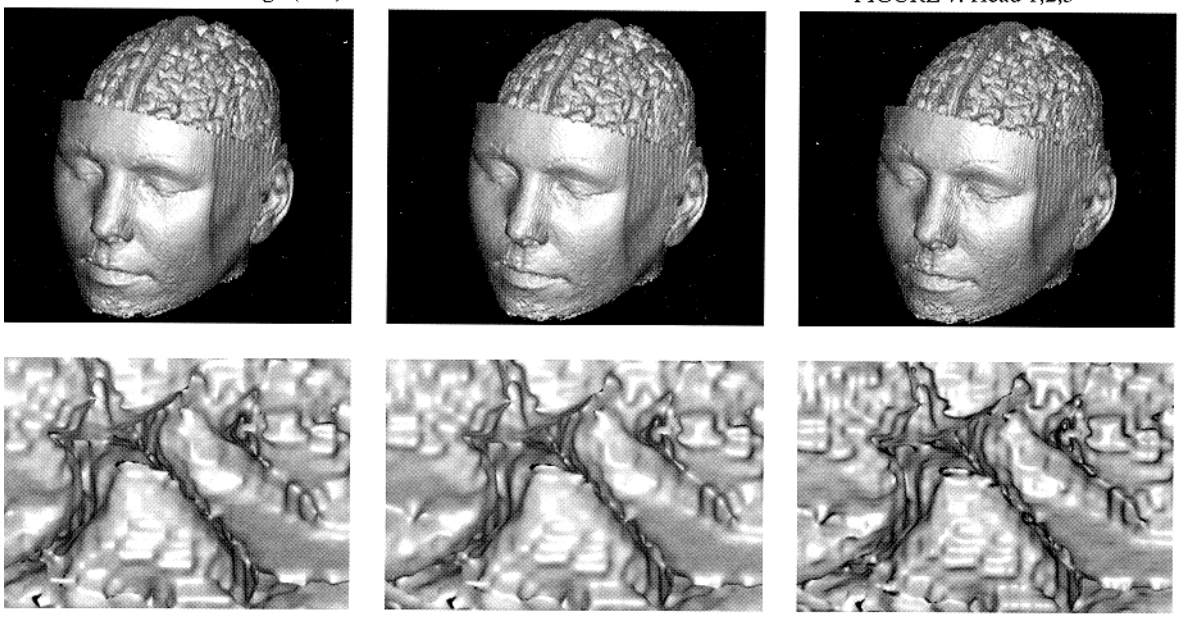


FIGURE 8. Brain 1, 2, 3

Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering
Torsten Möller, Raghu Machiraju, Klaus Mueller, Roni Yagel

A Comparison Of Normal Estimation Schemes

Torsten Möller[1,2], Raghu Machiraju[3,4], Klaus Mueller[1], Roni Yagel[1,2]

[1]Department of Computer and Information Science
[2]The Advanced Computing Center for the Arts and Design
The Ohio State University, Columbus, Ohio

[3]NSF Engineering Research Center for Computational Field Simulation
[4]Department of Computer Science
Mississippi State University, Mississippi

Abstract

The task of reconstructing the derivative of a discrete function is essential for its shading and rendering as well as being widely used in image processing and analysis. We survey the possible methods for normal estimation in volume rendering and divide them into two classes based on the delivered numerical accuracy. The three members of the first class determine the normal in two steps by employing both interpolation and derivative filters. Among these is a new method which has never been realized. The members of the first class are all equally accurate. The second class has only one member and employs a continuous derivative filter obtained through the analytic derivation of an interpolation filter. We use the new method to analytically compare the accuracy of the first class with that of the second. As a result of our analysis we show that even inexpensive schemes can in fact be more accurate than high order methods. We describe the theoretical computational cost of applying the schemes in a volume rendering application and provide guidelines for helping one choose a scheme for estimating derivatives. In particular we find that the new method can be very inexpensive and can compete with the normal estimations which pre-shade and pre-classify the volume [8].

Keywords: interpolation filters, derivative filters, filter design, normal estimation, Taylor series expansion, efficient volume rendering

1 INTRODUCTION

Reconstruction of a continuous function and its derivatives from a set of samples is one of the fundamental operations in visualization algorithms. In volume rendering, for instance, we must be able to interpolate the function at arbitrary locations to obtain the volume densities. The gradient (the first derivative of the function) is employed in both volume classification and shading [3][8]. If the gradient estimation is done carelessly, shading and classification will yield wrong colors and opacities. Since the derivative of a function indicates the velocity of change of the function values, the presence of noise especially will lead to incorrect images [4].

There have been various studies and comparisons of accurate interpolation filters, a summary of which is given in [10][12]. However, as is also shown in [12], the derivative approximation has a larger impact on the quality of the image and therefore deserves a thorough analysis, which is the goal of this paper.

The ideal derivative filter is the *Cosc* filter, which is the derivative of the ideal interpolation filter (*Sinc*) [1][4]. For a practical use of the *Sinc* filter, windowing is suggested [7]. Goss [6] extends the idea of windowing from interpolation filters to derivative filters. He uses a Kaiser window to mitigate the adverse effects of the truncated ideal derivative filter. Bentum et al. [1] use the Cardinal cubic splines to develop derivative filters. A good survey of existing digital derivative filters can be found in the paper by Dutta Roy and Kumar [4].

While all of the previous work focuses on the design of derivative filters, no work is known to us, that tries to conduct a comparative study of gradient filters. Especially, in the case of volume rendering, most algorithms are driven by efficiency considerations and may decompose the gradient estimation in one or two steps. One step is typically the interpolation of the normals or of the data values with a continuous interpolation filter. The other step is the application of a digital derivative filter (e.g. central differences) in order to compute the normal at the sampling location. However, there have been schemes proposed, that estimate the normal at an arbitrary point in the volume in one step [1]. The goal of this paper is to enumerate and classify the different schemes of gradient estimation and to analyze them in terms of accuracy and efficiency.

In this paper, we denote by $f(t)$ a continuous function (the *signal*) which is sampled into the discrete function $f[k] = f(kT)$, where T is the sampling distance and k is an integer. In computer imaging, $f(t)$ is not available; we only have $f[k]$. We denote by $h(t)$ the continuous function kernel used for interpolation and by $d[k]$ the digital (i.e. only defined for integer arguments) derivative filter.

We employ a Taylor series expansion of the convolution sum for our numerical analysis, as introduced in [12]. Our Taylor series expansion provides both qualitative and quantitative means of analyzing filters. In Section 3, this analysis is expanded to the convolution of two filters. The methods of [12] are briefly summarized.

1.1 Taylor Expansion of the Convolution Sum

To reconstruct a continuous function $f(t)$ or its derivative $f'(t)$ from a set of sample points $f[k]$, we convolve $f[k]$ with a filter kernel, i.e. we compute a weighted average of these samples. By convolving the sampled signal $f[k]$ with a continuous interpolation filter h, we reconstruct an approximation of the original function $f(t)$. Similarly, if we convolve the samples with a continuous derivative filter d, we reconstruct an approximation of the derivative of the original function. We denote the result of this operation by $f_r^w(t)$, where w is the filter used. Formally, this can be written as:

$$f_r^w(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot w(\frac{t}{T} - k) \qquad (1)$$

Now we can expand $f[k] = f(kT)$ into a Taylor series about t. The Taylor series expansion at that point would be:

$$f[k] = \sum_{n=0}^{N} \frac{f^{(n)}(t)}{n!}(kT-t)^n + \frac{f^{(N+1)}(\xi_k)}{(N+1)!}(kT-t)^{(N+1)}$$

where $f^{(n)}(t)$ is the n-th derivative of f and $\xi_k \in [t, kT]$

Substituting the Taylor series expansion into the convolution sum of Equation 1, leads to an alternative representation for the reconstructed value at a point t:

[1,2] 395 Dreese Lab, Columbus, OH 43210,
{moeller, mueller, yagel}@cis.ohio-state.edu

[3,4] NSF Engineering Research Center for Computational Field Simulation, P.O. Box 9627, Mississippi State, MS 39762,
raghu@erc.msstate.edu

0-8186-8262-0/97 $10.00 Copyright 1997 IEEE.

$$f_r^w(t) = \sum_{n=0}^{N} a_n^w(\tau) f^{(n)}(t) + r_{N,i}^w(\tau)$$

$$a_n^w(\tau) = \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k-\tau)^n w(\tau - k) \qquad (2)$$

$$r_{N,i}^w(\tau) \le \left(\max_{\xi \in [(i-M)T,\,(i+M)T]} (f^{(N+1)}(\xi)) \right) \left| a_{N+1}^w(\tau) \right|$$

or

$$r_N^w(\tau) \approx a_{N+1}^w(\tau) f^{(N+1)}(t)$$

where τ is chosen such that $t = (i+\tau)T$, with $0 \le \tau < 1$, and i is an integer. It is noteworthy that the derived Taylor coefficients a and the remainder term r only depend on the offset to the nearest sampling point, i.e., they are periodic in the sampling distance T. For further details, please refer to [12].

The characterization of the filtering process in Equation 2 imposes 4 different criteria for a good normal estimation scheme. First of all, we require a_0^w to be zero. Secondly we have to normalize by a_1^w in order to reconstruct the actual derivative as opposed to a multiple of it. Further by determining the largest N, such that a_N^w is zero, we can determine the asymptotic error behavior of a filter for a decreasing sampling distance T. Finally, the remainder term r gives us an indication of the absolute error of that filter.

This expansion of the convolution sum assumes that at least the first N derivatives of the function f exist, where N depends on our error analysis. This condition is generally met in practice since image and volume acquisition devices such as scanners and cameras inherently perform a low-pass filtering operation that bandlimits the functions [2]. Numerical simulations of physical phenomena, as performed in the field of computational fluid dynamics, usually generate bandlimited images as well since typically robust numerical solutions can be obtained only if the algorithm incorporates a smoothing step [15]. Finally, all rendering and scan-conversion algorithms, in order to provide antialiased images, usually also employ a filtering step that bandlimits the image. Bandlimited functions do not contain frequencies higher then a certain limiting frequency in their spectra. One can conclude, that bandlimited functions are analytic functions and all N derivatives exist.

The remainder of the paper is organized as follows. In Section 2, we summarize the different schemes for normal estimation. In Section 3, we modify the Taylor series expansion of the convolution operation for the specific use of cascading two filters, and compare the schemes of Section 2 numerically. In Section 4, we examine possible implementations of the normal estimation schemes and compare their efficiency. Experimental results are also presented in Section 5. Finally, in Section 6, we summarize the results of this paper and discuss some open questions.

2 GRADIENT RECONSTRUCTION FROM THE VOLUME SAMPLES

We will use the symbol F to represent the discrete function $f[k]$. Further, we let D and H denote the derivative and interpolation operators, respectively. In the process of volume rendering there are two additional operators applied to the data. The first is the *transfer function*, which maps the raw data values into material attributes such as color, opacity, and reflectivity. We denote this operator, also called *classification function*, by C. The second operator applied to the data is *shading*, which illuminates the data. The shading operator, which we denote by S, takes as input material attributes, light attributes, and the surface normal, and produces a displayable value (e.g., $RGB\alpha$).

Since S needs the output of C, shading will always be performed after classification. Since S needs the function's derivative, it will always be after D. We now present four different ways of computing the function derivatives. Except for the first approach, $(FD)H$, in all others the operators CS will be performed *after* the interpolated derivative has been computed.

2.1 Method $(FD)H$ - Derivative First

One way of computing the derivative at a point t of a discrete function $f[k]$ is to first compute the normal at the grid points kT and then interpolate these normals, producing the derivative at the desired location t. This is the method most commonly used in volume graphics [6][9]. The first step, the computation of the derivative at the grid points, can be expressed in the following convolution:

$$f^d[k] = \sum_{l=-\infty}^{\infty} f[l] \cdot d[k-l]$$

Now the derivative at an arbitrary point can be interpolated as:

$$f_r^{dh}(t) = \sum_{k=-\infty}^{\infty} f^d[k] \cdot h(\frac{t}{T} - k)$$

$$= \sum_{k=-\infty}^{\infty} \left(\sum_{l=-\infty}^{\infty} f[l] \cdot d[k-l] \right) \cdot h(\frac{t}{T} - k)$$

Square brackets are used to emphasize the discrete nature of the operator. Since a convolution in spatial domain is the same as a multiplication in frequency domain, we conclude the following frequency characterization of the above operation:

$$F_r^{dh}(\frac{\omega}{T}) = (F_D(\omega) D_D(\omega)) H(\omega) \qquad (3)$$

Here $D_D(\omega)$ denotes the Fourier transform of the discrete derivative filter and $F_D(\omega)$ denotes the Fourier transform of the sampled function $f[k]$. The Fourier transform of a discrete function contains replicated frequency spectra at $k2\pi$ (where k is an arbitrary integer). Therefore $D_D(\omega)$ and $F_D(\omega)$ are periodic functions with period 2π. Following the Fourier transform in Equation 3, we will refer to this method as $(FD)H$.

Unlike all other methods described in this paper, some algorithms ([8][3][16]) perform interpolation after classification and shading. Normal values are computed at the grid points and classification is also applied to the original data values. Then, these data points are shaded. The final $RGB\alpha$ volume is then interpolated at the appropriate sampling points. Using our notation, this method can be summarized by $(((FD)C)S)H$. This is indeed an efficient method, since CS does not have to be computed for every sample point (which is the case for all other methods described in this paper where interpolation is done *before CS*) but rather it is computed only for the original data points. However, this method will produce correct results only if both C and S are linear operators. The result of employing a non-linear transfer function or illumination model may, for example, cause the appearance of errors or pseudo-features that are non-existent in the original data. In the case of S, one must therefore allow illumination models consisting of only ambient lighting. In the case of C, the linearity restriction may not be acceptable for many applications. For example, if we want to find the opacity in-between two data values a and b (using linear interpolation), we would find $(C(a)+C(b))/2$ by performing classification first. However we would find $C((a+b)/2)$ by performing interpolation first. Obviously, if C is a non-linear operator, the two results will be different. We therefore concentrate our analysis and discussion in the more general and accurate methods that per-

form *CS* only *after* gradient estimation and interpolation.

2.2 Method *(FH)D* - Interpolation First

In this approach, we first reconstruct the continuous function $f(t)$ from the sampled values f_k and then apply the discrete derivative filter d [10][14]. Since the derivative filter is discrete, we only need to evaluate the convolution sum of the interpolated function at discrete points. The interpolated function $f_r^h(t)$ can be expressed as a convolution of the samples f_k using the interpolation filter h:

$$f_r^h(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot h(\frac{t}{T} - k)$$

The reconstructed derivative can be computed by:

$$f_r^{dh}(t) = \sum_{k=-\infty}^{\infty} d[k] \cdot f_r^h(t - kT)$$

Using similar arguments as above, we find the Fourier Transform to be:

$$F_r(\frac{\omega}{T}) = (F_D(\omega)H(\omega))D_D(\omega) \tag{4}$$

Using our previous notation scheme, we refer to this method as method *(FH)D*.

2.3 Method *F(DH)* - Continuous Derivative

Looking at all possible combinations of applying the interpolation filter and the derivative filter to the discrete signal, we are led to a theoretical result. Namely, that we can first convolve the digital derivative filter with the continuous interpolation filter. The result will be a new continuous derivative filter which we can apply to the data samples, enabling us to reconstruct the derivative at any arbitrary point t directly. This can be written as:

$$f_r^{dh}(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot dh(\frac{t}{T} - k)$$

where the continuous derivative filter $dh(t)$ is obtained as the convolution of the digital filter $d[k]$ with the interpolation filter h:

$$dh(t) = \sum_{k=-\infty}^{\infty} d[k] \cdot h(\frac{t}{T} - k)$$

We can show that the frequency representation of this process is:

$$F_r^{dh}(\frac{\omega}{T}) = F_D(\omega)(D_D(\omega)H(\omega)) \tag{5}$$

therefore referring to this method as *F(DH)*. The benefit of this scheme is more conceptual at this moment. In Section 3 we show how it can be used for a convenient analysis of the normal estimation process. Further we will show in Section 4, that this method can also be the most efficient to use for volume rendering algorithms.

2.4 Method FH' - Analytic Derivative

A fourth method to compute the gradient of a discrete function is to convolve the samples $f[k]$ with the analytical derivative of the interpolation filter h:

$$f_r^{h'}(t) = \sum_{k=-\infty}^{\infty} \frac{f_k}{T} \cdot h'(\frac{t}{T} - k)$$

In this case, h' represents a continuous derivative filter, allowing us to reconstruct the continuous derivative $f'(t)$ directly from the samples $f[k]$ in just one convolution. This is very similar to the previous method *F(DH)*. It differs only in the way we construct the derivative filter: In method *F(DH)* we compute a convolution sum for the continuous derivative filter, while in this method we compute the continuous derivative filter analytically. Bentum et al. [1] apply this idea to cardinal splines, and Marschner and Lobb [11] use this for the BC-splines. The Fourier transform of the derivative of a function is simply the scaled Fourier transform of that function multiplied by $i\omega$ ($i^2 = -1$)[2]. Therefore, we find that the Fourier transform of $f_r^{h'}(t)$ is:

$$F_r^{h'}(\frac{\omega}{T}) = F_D(\omega)\left(\frac{i\omega}{T}H(\omega)\right)$$

and we refer to this method as FH'.

3 NUMERICAL ACCURACY

Comparing Equations 3, 4, and 5 we easily find that these three methods are numerically equivalent and thus produce the exact same reconstructed derivative of f. Therefore, we will concentrate on comparing the methods *(FD)H*, *(FH)D*, *F(DH)* with the method FH'. In order to compare the numerical accuracy of the methods, we use the tools developed in [12] and summarized in Section 1.1.

For method FH', w in Equation 1 is simply the derivative of the interpolation filter h. For other methods, we choose the derivative filter described in Section 2.3. To clarify the notation, we will replace w by dh. To better compute the coefficients $a_n^{dh}(\tau)$ of Equation 2 for the derivative filter dh, we will substitute the convolution sum of the derivative and interpolation filters into the expression for $a_n^{dh}(\tau)$ in Equation 2:

$$a_n^{dh}(\tau) = \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k-\tau)^n \left(\sum_{l=-\infty}^{\infty} d[l] \cdot h(\tau - k - l) \right)$$

which simplifies to:

$$a_n^{dh}(\tau) = \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} d[l] \cdot \left(\sum_{k=-\infty}^{\infty} (k-\tau)^n h(\tau - k - l) \right)$$

Substituting m for $k+l$ in the inner sum, we get:

$$a_n^{dh}(\tau) = \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} d[l] \cdot \left(\sum_{m=-\infty}^{\infty} (m-\tau-l)^n h(\tau - m) \right)$$

$$= \frac{T^n}{n!} \sum_{l=-\infty}^{\infty} d[l] \left(\sum_{m=-\infty}^{\infty} \left(\sum_{i=0}^{n} \binom{n}{i} (m-\tau)^i (-l)^{n-i} \right) h(\tau - m) \right)$$

which resolves to

$$a_n^{dh}(\tau) = \frac{T^n}{n!} \sum_{i=0}^{n} \binom{n}{i} \left(\sum_{l=-\infty}^{\infty} (-l)^{n-i} d[l] \right) \left(\sum_{m=-\infty}^{\infty} (m-\tau)^i h(\tau - m) \right)$$

$$= \sum_{i=0}^{n} a_{n-i}^{d}(0) a_i^{h}(\tau)$$

This means that the error coefficient of a convolution filter is simply the convolution of the error coefficients of both filters. In Table 1, we have computed the coefficients for some commonly used filter combinations. The first column shows the error coefficients for the probably most common used filter combination of linear interpolation and central differences, abbreviated by DL. Another common choice is the combination of a cubic interpolation filter (we have chosen the class of cubic cardinal splines) with central differences. We let DC denote this filter class. For the class of analytic derivative filters we have chosen the derivative of the cubic interpolation filter, as introduced in [1]. We use C to represent this filter class.

In the case that α=-0.5, a_2^d is zero for all three methods and

	central difference + linear interpolation (DL)	central difference + cubic interpolation (DC)	cubic derivative (C)
a_0^d	0	0	0
a_1^d	T	T	$T(1+(2\alpha+1)(-6\tau^2+6\tau-1))$
a_2^d	0	$-T^2(2\alpha+1)\tau(1-\tau)(1-2\tau)$	$3T^2(2\alpha+1)\tau(1-\tau)(1-2\tau)$
a_3^d (normalized)	$\frac{T^2}{6}(1+3\tau-3\tau^2)$	$\alpha = -0.5: \frac{T^2}{6}$ $\tau = 0: \frac{T^2}{6}$ $\tau = 0.5: \frac{T^2}{6}\left(\frac{6}{4}\alpha+\frac{14}{8}\right)$	$\alpha = -0.5: \frac{T^2}{6}(6\tau^2-6\tau+1)$ $\tau = 0: \frac{T^2}{6}$ $\tau = 0.5: \frac{T^2}{6}\left(\frac{14\alpha+3}{8\alpha+12}\right)$

Table 1.Coefficients for some commonly used filter combinations

we must compare a_3^d. One can easily prove that this coefficient for DL is always greater than $T^2/6$ (the coefficient for DC), which in turn is greater than the coefficient for C. This implies that, the worst behavior is observed for DL, and C is more accurate than DC.

Therefore we conclude, that the optimal filter to use is C for $\alpha = -0.5$. However, one might be interested to use different α in different situations. For instance Park and Schowengerdt [13] conclude from a frequency study of the cardinal cubic splines, that some α (different than -0.5) might yield better images. They find that α depends strongly on the underlying function to be reconstructed. Therefore it is of interest to analyze the spatial error for different α as well.

In the case that $\alpha \neq -0.5$, the coefficient for a_2^d is zero only for the method *DL*. In order to compare the error coefficient among the methods *DC* and *C*, we compare a_2^d for both filters. As we have pointed out in [12], these coefficients need to be normalized. Fig. 1 shows a plot of a_2^d after its normalization. Note that T simply scales both plots equally. Therefore, it can be set to one. In Fig. 1, one can clearly see that the error coefficient for *DC* is smaller than the error coefficient for C. Therefore, we conclude that *DC* is superior to *C* when $\alpha \neq -0.5$. This is a rather unexpected result, since one would naturally expect the analytic derivative of a filter to be more exact and therefore to perform better. As we have just seen, this is not necessarily the case.

For the special cases that $\tau = 0$ and $\tau = 0.5$ (where $a_3^d = 0$) we found by comparing a_3^d, that C is more accurate than DC force $\alpha \in [-3,-0.6]$. Another value to consider is the second derivative of the underlying function. When it goes to zero, we also have to use the error coefficient a_3^d for an error comparison.

We are left to compare the error behavior of the most common method DL with the other two methods. Again, for the special cases, where the second coefficient or the second derivative of the function go to zero, we must compare a_3^d in order to find the most accurate filter. For the other cases however, we can follow the following analysis. If we have influence on the original sampling distance T for our applications, we can always find a T, such that the combination of central difference and linear interpolation is superior to the other two methods. In other words, DL is asymptotically better than DC and C. However in most practical applications we are given a data set with fixed sampling distance T. In these cases we need to weight the actual error of the filters and conclude from this comparison which filter is more accurate. If we are comparing DL and DC, we want to find out for which α DC performs better than DL. Mathematically:

$$\|\varepsilon_{DL}\| \geq \|\varepsilon_{DC}\|$$

Using the second error approximation of Equation 2, we find the following criteria:

$$\left\|a_3^{DL}(\tau)f^{(3)}(t)\right\| \geq \left\|a_2^{DC}(\tau)f^{(2)}(t)\right\|$$

We can conclude that the choice of α very much depends on the resampling offset τ and the actual data. After some algebraic manipulations, we can conclude:

$$T\frac{5}{4}\left\|\frac{f^{(3)}(t)}{f^{(2)}(t)}\right\| \geq \alpha + 0.5 \geq -T\frac{5}{4}\left\|\frac{f^{(3)}(t)}{f^{(2)}(t)}\right\| \qquad (6)$$

For α in this range the method DC is more accurate than DL. As expected, the choice of the most accurate filter strongly depends on the underlying data.

For a similar comparison of the methods *DL* and *C* we find:

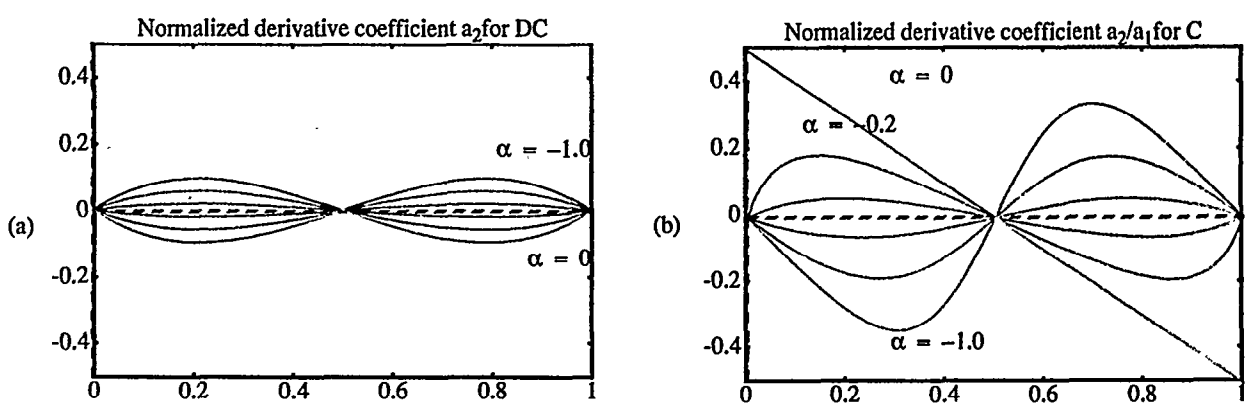


FIGURE 1. In both plots we set T to 1.0. α takes the values 0, -0.2, -0.4, -0.6, -0.8, -1. (a) The coefficient a_2^h of the central difference and cubic interpolation filter for varying α. (b) The normalized coefficient a_2^d/a_1^d of the cubic derivative filter for varying α.

$$\frac{1}{\frac{14}{75}\left\|\frac{f^{(2)}(t)}{f^{(3)}(t)}\right\|+2} \geq \alpha + 0.5 \geq -\frac{1}{\frac{14}{75}\left\|\frac{f^{(2)}(t)}{f^{(3)}(t)}\right\|+2} \quad (7)$$

An important observation that we draw from Equation 6 and Equation 7 is the dependency of the comparative accuracy on the sampling distance. The higher the sampling rate the smaller the range in which C or DC performs better than DL. This means that for densely sampled data sets a combination of linear interpolation and central difference is not only efficient, but also recovers the derivative accurately. That can also be explained in the frequency domain. The higher the sampling rate, the further apart the frequency spectra are placed. In other words, the signal's aliases are more separated. Thus, the deficiencies of the central difference operator at higher frequencies do not impose a problem since no signal aliases exist in this frequency range. This is an important and new result, since it tells us, that for some data sets DL is just as accurate, as the other two (more expensive) methods DC and C.

4 EFFICIENCY CONSIDERATIONS

In this section, we compare the four methods *(FH)D*, *(FD)H*, *F(HD)*, and *FH'* from an efficiency perspective. While the first three methods are equivalent from an accuracy standpoint, they are not so from an efficiency point of view. This section also contrasts the overall computational effort of these four shading-deferring methods with the demands of the popular, but less accurate, pre-shading scheme [8]. We denote this approach as *((FD)CS)H*, where *C* and *S* stand for classification and shading that occur after gradient computation but before color interpolation. Our comparisons will be valid for the 3D case only (a typical application will be volume rendering algorithms). If we compare normal estimation schemes in other dimensions, our analysis will be similar.

In the following discussion, we distinguish between implementations that compute all results on-the-fly, and implementations that utilize some form of caching to reuse previously computed results. The latter approaches obviously requires an extra amount of memory and cache management overhead. We now introduce some terminology:

E_λ: The computational effort to apply the operator λ where λ can be H, D, DH, H', or CS.

$\|\lambda\|$: Number of filter weights used for applying the operator λ.

n: Number of data elements (voxels).

m: Number of samples.

For digital filters $\|\lambda\|$ is obviously the length of the filter, but for continuous filters (e.g. cardinal splines) $\|\lambda\|$ is usually the filter support, i.e. the number of sample values, that are influenced by the filter. Since the filter operation is the weighted sum of $\|\lambda\|$ elements, we usually have $E_\lambda = 2\|\lambda\| - 1$ for a straight forward implementation of $\|\lambda\|$ multiplications and $\|\lambda\| - 1$ additions. However, for some special filters, there will be a more efficient implementation. For instance, the central difference filter (in one dimension), can be implemented in 2 operations (one subtraction and one division by 2) as opposed to 3 operations (two multiplications by 0.5 and one subtraction). Therefore, we find it important to separate between E_λ and $\|\lambda\|$.

In the following discussion we will discuss the cost of reconstructing the function and its derivative at all the sample points. We will also comment on the cost of applying the classification and shading operators.

4.1 *(FD)H* - Derivative First

In this method, we first compute the gradient at all grid voxels within the extent of the interpolation filter h, and then interpolate these gradients using H. An on-the-fly approach would have to compute $\|H\|$ gradients for a total cost of $\|H\|E_D$, followed by three interpolations to compute the three gradient components and one interpolation to compute the data value itself. The total cost is thus:

$$m(\|H\| \cdot E_D + 3E_H + E_H)$$

By storing computed gradients in a gradient volume, one could reduce the cost to:

$$n \cdot E_D + m(3E_H + E_H)$$

The process of classification and shading will require additional $m \cdot E_{CS}$ cost and the total cost will then be:

$$n \cdot E_D + m(4E_H + E_{CS})$$

However, in the *((FD)CS)H* method, classification and shading are applied to the data values, and the interpolation filter is applied to the resulting $RGB\alpha$ values. Therefore, the total cost for this method, assuming caching, is:

$$n \cdot (E_D + E_{CS}) + m(4E_H)$$

Since in most cases, to assure proper sampling, $n \leq m$, the computational advantage of this method is clear. Moreover, when classification and illumination does not change for multiple rendering, the cost of the first component in the last two equations is amortized and can therefore be ignored. If we ignore the shading component then the cost of reconstructing the function and its derivative assuming caching is given by:

$$n \cdot E_D + m(4E_H)$$

4.2 *(FH)D* - Interpolation First

The *(FH)D* method computes the derivative at a ray sample location from a set of additionally interpolated samples in the neighborhood of the sample location. In parallel (orthographic) rendering of volumes the data is resampled into a new grid. If this grid is cached somewhere, one can perform the derivative calculations using the data values at that grid.

Without caching, in order to compute the derivative at a sample location, *(FH)D* interpolates $\|D\|$ additional samples, each at a cost of E_H, and uses them to obtain three axis derivatives at the cost E_D. Another interpolation at the sample location, each at a cost of E_H, yields the function value. The total cost for reconstructing the function and its derivative is:

$$m(\|D\| \cdot E_H + E_D + E_H)$$

Later, these samples are classified and shaded, with an additional cost (for the whole volume) of $m \cdot E_{CS}$. However, if caching is employed, only one interpolation is needed per sample, and the D operator uses only existing samples. Therefore the total cost for reconstructing the function and its derivative:

$$m(E_D + E_H)$$

4.3 *F(DH)* - Continuous Derivative

Here the derivative filter is pre-convolved with the interpolation filter which increases its size. The gradients are then computed by convolving the volume by this combined *DH*-filter. The total cost for computing the function and its derivative is then given by:

$$m(E_{DH} + E_H)$$

This is the most direct method of the three methods presented so far and there is no caching mechanism available to gain some speedup.

4.4 Method *FH'* - Analytic Derivative

This method is not equivalent to the previous three in terms of

accuracy, as other sections of this paper demonstrate. FH' uses a special gradient filter derived from the interpolation filter to estimate the gradients. Since this derivative filter has the same size as H the corresponding cost for computing the function and its derivative is:

$$m(E_H + E_H).$$

4.5 Summary and Numerical Examples

We are now ready to compare the theoretical cost functions presented in the previous subsections and provide some numerical examples to highlight the differences. Table 2 lists all costs derived above and gives two numerical examples: In case 1, H and H' are cubic filters ($\|H\| = 4^3$), D is a central difference filter ($\|D\| = 6$), and in case 2, H is a trilinear filter ($\|H\| = 2^3$) and D is again the central difference filter. For the following discussion we count the number of floating point operations associated with each operator, but we do not distinguish between additions, multiplications or divisions. In this case, the cost of E_H for H being cubic is $2 \cdot 4^3 - 1 = 127$, of E_H for H being trilinear is $2 \cdot 2^3 - 1 = 15$, of E_D for D being central difference the cost is 6. Since the derivative filters are directional filters, and E_{DH} denotes the cost of computing all three derivative components, we find that E_{DH} is three times the cost of one derivative component operation. In order to find the directional derivative, we convolve the interpolation filter of size $s \cdot s \cdot s = s^3$ with a 1D derivative filter of length k (in our case - central differences - $k = 2$). That results in a filter of size $(s + k - 1) \cdot s \cdot s$. Therefore we find the cost of E_{DH} for H being cubic is 477 and for H being trilinear is 69.

As expected the analytical derivative method (FH') is the most efficient one. However, as we showed in Section 3, it is not necessarily the most accurate. Among the other three schemes (which are numerically identical), we find our new method ($F(DH)$) most efficient if there is no caching. However, if caching is available, $(FH)D$ is certainly the most efficient way to compute the normal and the data value at this point. Therefore, we conclude that in terms of efficiency and in terms of accuracy, there is no need for the most commonly used method $(FD)H$ (in the case of deferred shading). As was pointed out already in Section 4.1, if we do shading at the grid locations, we might find a more efficient algorithm, yet trading speed for accuracy.

5 EXPERIMENTS

The images were rendered employing a simple raycaster to find isosurfaces.The volumes were resampled at an interval of 0.05 voxel lengths. At each sampling point, the raycaster first applied the interpolation kernel to reconstruct the function at that point. If the reconstructed value was above a pre-set isovalue, the derivative filter was used to compute the 3D gradient. Shading was then performed using the traditional Phong lighting model [5] with diffuse and specular reflections. The obtained color and opacity were composited with the previous ray values, and the ray was terminated after the opacity reached a value close to one. Since for all our filters both the interpolation and the derivative kernel were separable, the filter operations could be efficiently performed using a scheme similar to the one given by Bentum et al [1].

For our experiments we used an analytic data set, derived from the same function as the one used by Marschner and Lobb [11]. Specifically, we used:

$$f(x, y, z) = \frac{1}{2} - \frac{2}{5}\sin\left(\frac{\pi}{2}z\right) + \frac{1}{10}\cos\left(12\pi\cos\left(\frac{\pi}{2}\sqrt{x^2 + y^2}\right)\right)$$

Since we study different derivative filters, we have fixed the interpolation filter to be the Catmull-Rom interpolation filter - a cubic filter with small error as was also shown in [12]. From Equations 6 and 7 we learn that the range of α where C and/or DC performs better than DL is dependent on the data set. To address this issue, we have computed the ratio $\|f^{(3)}(t)/f^{(2)}(t)\|$ analytically for the data points for the three axis directions x, y, and z, where we reconstruct and collected them in a histogram, plotted in Fig. 2. In order to guarantee that all data points are reconstructed more accurately using DC (or C) than DL, we would have to choose the minimal ratio. This ratio is zero and therefore we can conclude that only for $\alpha = -0.5$ we can guarantee, that the derivative reconstruction at any single point will be better for the methods DC and C as opposed to DL. In order to get practical results, we could choose a higher ratio of $\|f^{(3)}(t)/f^{(2)}(t)\|$, giving up on the accuracy assurance for some reconstructed values. If we for instance choose the ratio 7, we still guarantee all z directional derivatives to be estimated more accurately. Approximatly 8% of the directional derivatives in y will be more accurate by DL, and only 3.8% of the directional derivatives in x will be better by DL.

When we plug in the ratio of 7 into Equations 6 and 7, we find the theoretical result that for $\alpha \in [-0.78, -0.22]$, DC performs better than DL and for $\alpha \in [-0.65, -0.34]$, C performs better than DL. These theoretical ranges have steered our experiments and in Fig. 3 (see color plates) we have rendered the Marschner-Lobb data set for several different α. For a better (analytical) understanding of these rendered images, we have also drawn the angular error images in Fig. 4. For each reconstructed normal we computed the actual normal and recorded their angular difference. The grey value of 255 was displayed for an angular error of 5 degrees.

For the first row of images we have used $\alpha = -0.5$. Following our analysis in Section 3, we expect that $\varepsilon(C) < \varepsilon(DC) < \varepsilon(DL)$, where $\varepsilon(A)$ denotes the error measure of image A. The first row of Fig. 3 shows the different images for $\alpha = -0.5$. Although the differences are small, one can find DC to be better, than DL. Although the image for DC is overall smoother, it's error image in Fig. 4 reveals a much higher error than for C.

The images for $\alpha = -1.0$ show the opposite behavior. From our analysis we deduce the following error behavior: $\varepsilon(DL) < \varepsilon(DC) < \varepsilon(C)$. From Fig. 3 we conclude, that C clearly is the worst image. Also a visual comparison of DC and DL leads to the conclusion, that DL is better than DC. The error images in Fig. 4 support this analysis.

The rows for $\alpha = -0.6$ and $\alpha = -0.7$ show rather a transitional phase. Since the change of the filter weights happens continuously, we cannot necessarily expect a sudden sharp change in the image

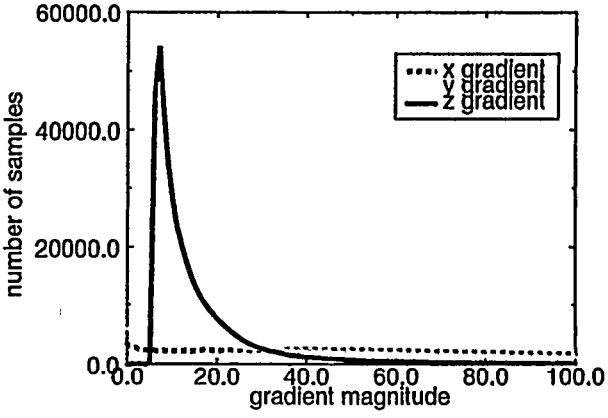


FIGURE 2. The ratio of $\|f^{(3)}(t)/f^{(2)}(t)\|$ for the directional derivatives in x, y, and z direction respectively for the Marschner Lobb data set.

	Theoretical Cost		Cubic Interp. and Central Diff.		Trilinear Interp. and Central Diff.	
	No Caching	Caching	No Cache	Cache	No Cache	Cache
(FD)H	$m(\|H\| \cdot E_D + 3E_H + E_H)$	$n \cdot E_D + m(4E_H)$	$1084m$	$9n+508m$	$132m$	$9n+60m$
((FD)CS)H	---	$n \cdot E_D + m(4E_H)$	---	$9n+508m$	---	$9n+60m$
(FH)D	$m(\|D\| \cdot E_H + E_D + E_H)$	$m(E_D + E_H)$	$898m$	$136m$	$114m$	$24m$
F(DH)	$m(E_{HD} + E_H)$	---	$604m$	---	$84m$	---
FH'	$m(2E_H)$	---	$254m$	---	$30m$	---

Table 2. Comparison of efficiency of the normal estimation schemes

quality. The differences in the image quality can be better studied using the error images in Fig. 4. We can conclude, that for $\alpha = -0.6$ our results follow our theoretical analysis: $\varepsilon(DC) < \varepsilon(C) < \varepsilon(DL)$. However, for $\alpha = -0.7$ it is debatable, which method is preferable in terms of image quality. Analytically we show $\varepsilon(DC) < \varepsilon(DL) < \varepsilon(C)$. It is clear, that the image for C is the least appealing to the viewer.

6 CONCLUSIONS AND FUTURE GOALS

We have classified the different techniques of normal estimation into four groups, and we have developed a new scheme *F(DH)*. We showed that the schemes *(FD)H*, *(FH)D* and *F(DH)* are numerically equivalent, and then extended the idea of classifying filters using Taylor series expansion to the convolution of two filters. We found that computing the analytic derivative of a filter kernel (method *FH'*) is not always more accurate than using a combination of that filter with the central difference kernel (any of the methods FDH). Therefore, a careful analysis of existing filters and filter combinations is suggested.

The new scheme *F(DH)* opens up new ways to design continuous derivative filters. Furthermore, this method of normal estimation is also the second most cost-efficient one, if no caching is performed (with *FH'* being the most cost effective one). However, if caching is enabled, then the method *(FH)D* is clearly preferable over any other method in terms of efficiency. In fact, what is believed as one of the most commonly used methods, *(FD)H*, is one of the slowest normal estimation method. The only advantage one could gain is the pre-calculation of the shading operation at the grid voxels, as Levoy [8] has proposed it. However, as was pointed out in Section 2, this method is certainly not preferable if accurately rendered images are required.

One of our immediate goals is to compare various combinations of known derivative and interpolation filters in order to find new derivative filters. We also would like to extend the error analysis to frequency space so that we can examine any aliasing and smoothing errors. Finally, it would contribute to the accuracy of our analysis to include a noise model. We also believe that it is very important to further investigate the shading and classification steps in terms of numerical accuracy.

Acknowledgments

We thank Prof. Wayne Carlson and the Advanced Computing Center for the Arts and Design for the use of their computing facilities and Prof. Robert Moorhead of the NSF Engineering Research Center, Mississippi State University as well as Tom Malzbender of Hewlett-Packard Labs for providing encouragement and support. This project was partially supported by the Department of Defense USAMRDC 94228001, by the Advanced Research Projects Agency Contract DABT63-C-0056 and the NSF Research Center for Computational Field Simulations.

References

[1] Bentum M.J., Lichtenbelt B.B.A., Malzbender T., "Frequency Analysis of Gradient Estimators in Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242-254, September 1996.

[2] Bracewell R.N., *Two Dimensional Imaging*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1995.

[3] Drebin R.A., L. Carpenter L., Hanrahan P., "Volume Rendering", *Computer Graphics*, 22(4):51-58, August 1988.

[4] Dutta Roy S.C., Kumar B., "Digital Differentiators", in *Handbook of Statistics*, N. K. Bise and C. R. Rao eds., vol. 10: 159-205, 1993.

[5] Foley J.D., van Dam A., Feiner S.K., Hughes J.F., *Computer Graphics, Principles and Practice*, Addison-Wesley, Reading, Massachusetts, 1990.

[6] Goss M.E., "An Adjustable Gradient Filter for Volume Visualization Image Enhancement", *Proceedings of Graphics Interface'94*, pp. 67-74, Toronto, Ontario, 1994.

[7] Hamming R.W., *Digital Filters*, Prentice Hall Inc., Second Edition, Englewoods Cliffs, NJ, 1983.

[8] Levoy M., "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, 8(5):29-37, May 1988.

[9] Lorensen W.E., Cline H., "Marching Cubes: a High Resolution 3D Surface Reconstruction Algorithm", *Computer Graphics*, 21(4):163-169, July 1987.

[10] Machiraju R.K., Yagel R., "Reconstruction Error Characterization and Control: A Sampling Theory Approach", *IEEE Transactions on Visualization and Computer Graphics*, ITVCG 2(4):364-376, December 1996.

[11] Marschner S.R. and Lobb R.J., "An Evaluation of Reconstruction Filters for Volume Rendering", *Proceedings of Visualization '94*, IEEE CS Press, pp. 100-107, October 1994.

[12] Möller T., Machiraju R.K., Mueller K., Yagel R., "Evaluation and Design of Filters Using a Taylor Series Expansion", *IEEE Transactions on Visualization and Computer Graphics*, ITVCG 3(2): 184-199, June 1997.

[13] Park S.K., Schowengerdt R.A., "Image Reconstruction by Parametric Cubic Convolution", *Computer Vision, Graphics, and Image Processing*, vol. 23:258-272, 1983

[14] Pfister H., Kaufman A.E., "Cube-4 - A Scalable Architecture for Real-Time Volume Rendering", *Proceedings of the 1996 Symposium on Volume Visualization*, pp. 47-54, October 1996.

[15] Warsi Z.U.A, *Fluid Dynamics: Theoretical and Computational Approaches*, CRC Press, Boca Raton, FL, 1992.

[16] Westover L., "Footprint Evaluation for Volume Rendering", *Computer Graphics (SIGGRAPH '90 Proceedings)*, vol. 24:367-376, August 1990.

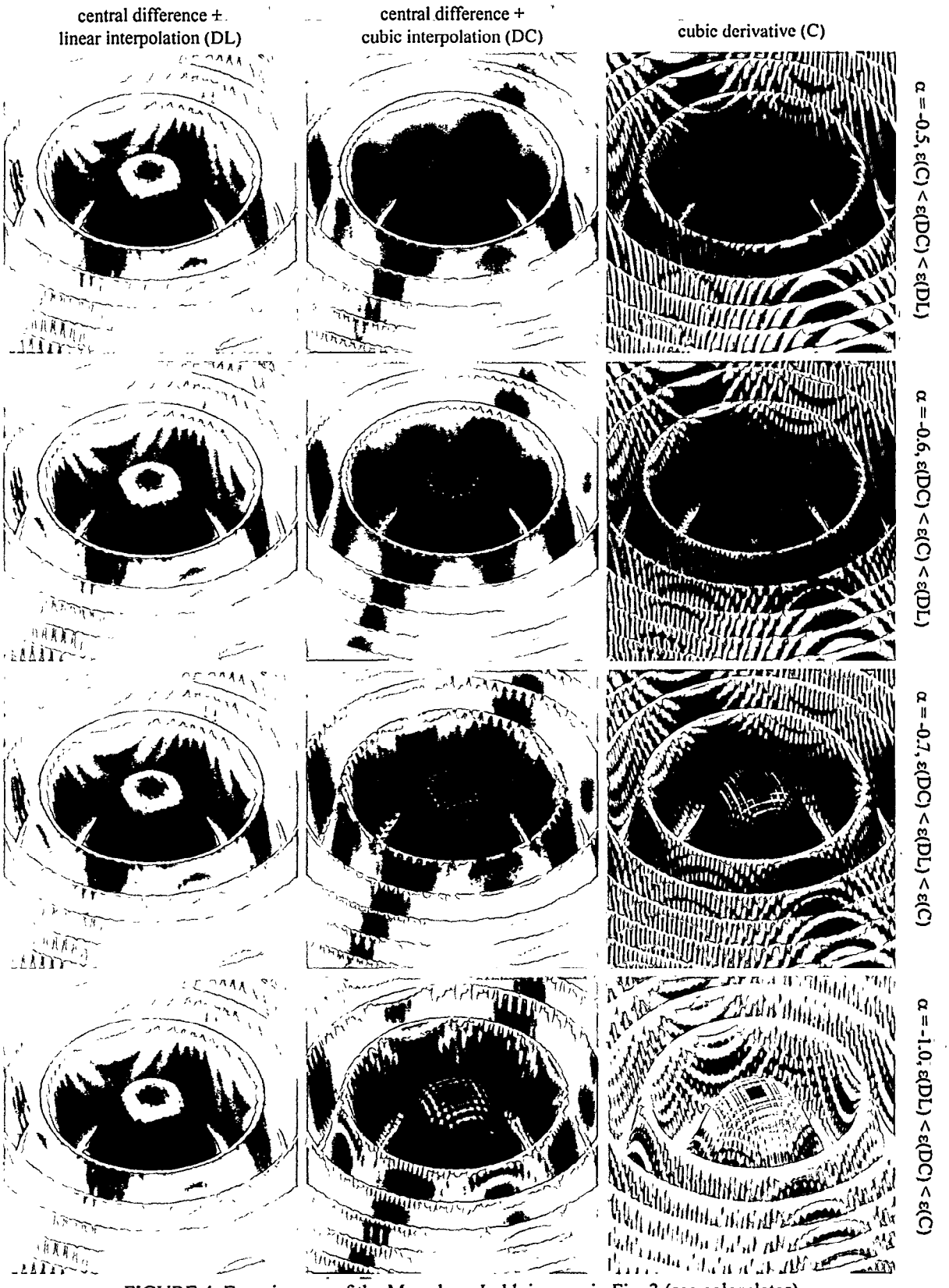


FIGURE 4. Error images of the Marschner Lobb images in Fig. 3 (see colorplates)

Design Of Accurate And Smooth Filters For Function And Derivative Reconstruction

Torsten Möller[1,2], Klaus Mueller[1], Yair Kurzion[1], Raghu Machiraju[3,4], Roni Yagel[1,2]

[1]Department Of Computer And Information Science
[2]The Advanced Computing Center For The Arts And Design
The Ohio State University, Columbus, Ohio

[3]NSF Engineering Research Center For Computational Field Simulation
[4]Department of Computer Science
Mississippi State University, Mississippi

ABSTRACT

The correct choice of function and derivative reconstruction filters is paramount to obtaining highly accurate renderings. Most filter choices are limited to a set of commonly used functions, and the visualization practitioner has so far no way to state his preferences in a convenient fashion. Much work has been done towards the design and specification of filters using frequency based methods. However, for visualization algorithms it is more natural to specify a filter in terms of the smoothness of the resulting reconstructed function and the spatial reconstruction error. Hence, in this paper, we present a methodology for designing filters based on spatial smoothness and accuracy criteria. We first state our design criteria and then provide an example of a filter design exercise. We also use the filters so designed for volume rendering of sampled data sets and a synthetic test function. We demonstrate that our results compare favorably with existing methods.

Keywords: *Interpolation* (G.1.1) *Approximation* (G.1.2) *Quadrature and Numerical Differentiation* (G.1.4) *Picture/Image Generation* (I.3.3) *Reconstruction* (I.4.5)

Other Keywords: *Volume Rendering, Filter Design, interpolation, derivatives*

1. INTRODUCTION

The reconstruction of a function and its derivatives from a set of given samples of that function is a fundamental operation in many areas. Computer graphics, scientific visualization, and image processing are just a few examples. In all these areas, a set of samples of an unknown function is usually all we know of that function. Hence, the reconstruction of the function between sample points is rather arbitrary, and one cannot talk about an *ideal* reconstruction function. The notion of *ideal* reconstruction is based on the assumption, that a given function is a member of a certain functional space, e.g. the L^2 space or the bandlimited function space. This functional space (and therefore the *ideal* reconstruction method) is usually determined by the particular application. In visualization, and in other fields, we assure that the given function belongs to the space of smooth functions C^n, where n is an integer.

A very important and often studied space in the class of all smooth functional spaces, is the space of bandlimited functions (a subclass of C^∞). They are often studied in the frequency domain using a signal processing approach. Although these methods are capable of controlling *global* errors such as blurring and aliasing, no *local* spatial assessment of their accuracy can be conducted directly. It turns out that the ideal reconstruction filters for the space of bandlimited functions are impractical to use. Hence research in this area has focused on finding efficient filters that approximate the ideal filter [1][4][7][9][10][14][15].

Another body of work has concentrated on minimizing the local spatial error for design and evaluation of filters [11][16][17][20]. The local error was measured and minimized using a Taylor series expansion. Since visual perception, judged by ringing, aliasing and blurring, was of concern, the frequency behavior of the resulting filter was discussed. In addition, spatial design gives an easy control over the size of the filter, and hence on the efficiency of the resulting filter, a property frequency-based methods do not have. However, it was found that the sole concern for numerical accuracy can lead to discontinuous filters, which can produce visual artifacts that are easily detected [17]. The goal of this paper is to overcome this problem by introducing a smoothness requirement into the filter design process.

All filter designs in the spatial domain have built filters according to an accuracy criteria. In this paper we introduce, for the first time, filter design criteria for interpolation and derivative filters which yield functions with a minimal numerical error and still maintain good spectral properties. The only assumption that we require of the original function (represented by the given samples) is that it is smooth and a member of the functional space C^n. We also show how our design criteria relate to criteria in frequency domain.

Our filter design is not restricted to cubic polynomial basis functions, but can generate filters of arbitrary smoothness and accu-

{moeller, mueller, kurzion, yagel}@cis.ohio-state.edu, raghu@erc.msstate.edu

racy. In this paper, we design optimized piecewise polynomial interpolation filters according to a set of smoothness and accuracy requirements. These filters are drawn from the set of all piecewise polynomial filters, a more general class of filters than the popular BC-splines. Since our methods also apply easily to any derivative filter design, we find optimal piecewise polynomial gradient filters as well. Our results, which go beyond the de-facto standard of the popular cubic BC-splines introduced by Mitchell and Netravali [16], are summarized in Table 1 and Table 2. These tables provide a guide to which filters should be used in most applications. Furthermore, we provide the practitioner with an easy and fast way to design filters that are specific to their applications by determining an application oriented set of smoothness and accuracy criteria.

The outline of this paper is as follows; Section 2 summarizes previous research in this field. In Section 3, we introduce the design criteria that we use in Section 4 to design new filters. In Section 5 we present some experimental results and in Section 6 we suggest steps for furthering this research. Finally, in Section 7, we summarize our findings.

2. Previous Research

Two of the more important and well studied reconstruction algorithms are interpolation and gradient estimation. In volume rendering, we must be able to interpolate the function at arbitrary locations to obtain the volume densities needed for arbitrary viewing. The gradient (the first derivative of the function) is employed in both volume classification and shading [6][13]. If the gradient estimation is done incorrectly, shading and classification will yield misleading colors and opacities.

Many researchers have shown that the *Sinc* filter is an ideal interpolation filter for the space of bandlimited functions (a subclass of C^{∞}). In this space the *Cosc* filter, which is the analytic derivative of the *Sinc* filter, is an ideal derivative filter [1][7][19]. These filters completely cut off the frequencies above a certain Nyquist frequency. Because of this discontinuity in the frequency domain, those filters have infinite support in the spatial domain and therefore are impractical to use for digital signals. Windowing the *Sinc* filter was introduced in order to smoothly limit this filter spatially [10][19]. Carlbom [4] computed an approximation to a modified *Sinc* filter with a minimized Chebychev error. Goss [9] extended the idea of windowing from interpolation filters to derivative filters. He used a Kaiser window to mitigate the adverse effects of the truncated ideal derivative filter. Instead of trying to find a good approximation to the ideal filter for all frequency ranges, Dutta Roy and Kumar's filter design [7] can be easily adapted to find good approximations for select frequency ranges.

A comparative study by Marschner and Lobb [15] proposed the use of different error metrics for various reconstruction artifacts of interpolation filters. These error metrics operate in the frequency domain and measure the smoothing, post-aliasing, and overshoot attributes of an interpolation filter. This study showed that the windowed *Sinc* filter has the best behavior.

In the spatial domain, Keys [11] analyzed a certain class of cubic splines, also called *cardinal splines*, using a Taylor series expansion. He showed that, within this class, the Catmull-Rom spline is optimal in the sense that it interpolates the original function with the smallest asymptotic spatial error. He also graphically compared the Catmull-Rom spline with the ideal interpolation filter, noticing that it is suitable for practical applications in computer graphics. Mitchell and Netravali [16] introduced a more general class of cubic splines which we refer to as *BC cubic splines* or in short, *BC-splines*. Cardinal cubic splines are a subclass of the BC-splines. Mitchell and Netravali conducted a study involving more than 500 sample images, classifying the parameter space into different regions of dominating reconstruction artifacts such as blurring, ringing, and anisotropy. They found, by using a Taylor series expansion, that filters for which $B + 2C = 1$ are the most numerically accurate within the class of BC-splines and have an error proportional to the square of the sampling distance. They also found, through their empirical studies, that these filters, although numerically superior, are not always visually superior.

Recently, we have shown [17] that the derivative approximation has a larger impact on the quality of the volume rendered image than the interpolation operation and therefore deserves a thorough analysis. Unfortunately, not much work has been done in the spatial design of derivative filters. Bentum et al. [1] use the Cardinal cubic splines as a basis to constructing the derivative filter through an analytic derivation of the interpolation filter. Although the authors illustrate the effect of various parameters on these filters via a number of frequency plots, they do not analytically compare the different filters. We (in [17]) have developed tools for the spatial analysis of both interpolation and derivative filters of arbitrary order. We used a Taylor series of the convolution sum in order to come up with four evaluation criteria. These criteria include asymptotic, as well as absolute, local error effects of the filter on the reconstructed function. We use these criteria in our current paper as a way to control the numerical error of the filters that we design. Using the methods developed in [17] we conducted a comparison of various derivative (normal vector) reconstruction methods and classified them into four reconstruction schemes [18].

Since we will employ the results of [17] throughout our paper, we include a summery here:

2.1 Taylor Expansion of the Convolution Sum

To reconstruct a continuous function $f(t)$ or its derivative $f'(t)$ from a set of sample points $f[k]$, we convolve $f[k]$ with a continuous filter kernel w. The filter w can be either an interpolation or a derivative filter. We denote the result of this operation by $f_r^w(t)$. Formally, this can be written as:

$$f_r^w(t) = \sum_{k=-\infty}^{\infty} f[k] \cdot w(\frac{t}{T} - k), \tag{1}$$

where T is the sampling distance. Now we can expand $f[k] = f(kT)$ into a Taylor series of $N+1$ terms about t. The Taylor series expansion at that point would be:

$$f[k] = \sum_{n=0}^{N} \frac{f^{(n)}(t)}{n!}(kT-t)^n + \frac{f^{(N+1)}(\xi_k)}{(N+1)!}(kT-t)^{(N+1)}$$

where $f^{(n)}(t)$ is the n-th derivative of f and $\xi_k \in [t, kT]$.

Substituting the Taylor series expansion into the convolution sum of Equation 1, leads to an alternative representation for the reconstructed value at a point t:

$$
\begin{aligned}
f_r^w(t) &= \sum_{n=0}^{N} a_n^w(\tau) f^{(n)}(t) + r_{N,i}^w(\tau) \\
a_n^w(\tau) &= \frac{T^n}{n!} \sum_{k=-\infty}^{\infty} (k-\tau)^n w(\tau - k) \\
r_{N,i}^w(\tau) &\le \left(\max_{\xi \in [(i-M)T,\ (i+M)T]} (f^{(N+1)}(\xi)) \right) \left| a_{N+1}^w(\tau) \right| \\
&\text{or} \\
r_N^w(\tau) &\approx a_{N+1}^w(\tau) f^{(N+1)}(t)
\end{aligned}
\tag{2}
$$

where τ is chosen such that $t = (i+\tau)T$, with $0 \le \tau < 1$, and i is an integer. It is noteworthy that the derived error coefficients a only depend on the offset τ to the nearest sampling point, i.e., they are periodic in the sampling distance T. For further details, please refer to [17].

The characterization of the filtering process in Equation 2 imposes four different criteria for a good reconstruction scheme of the k-th derivative. First of all, we require a_n^w to be zero for all n smaller than k. Secondly we have to normalize by a_k^w in order to reconstruct the actual derivative as opposed to some multiple of it. Further by determining the largest N, such that a_N^w is zero, we can determine the asymptotic error behavior of a filter for a decreasing sampling distance T. Finally, the remainder term r gives us an indication of the absolute error of that filter.

This expansion of the convolution sum assumes that at least the first N derivatives of the function f exist, where N depends on our error analysis. Hence, we assume that the underlying function is a member of the class of smooth functions C^N. This condition is generally met in practice [2][17][21].

3. Design Criteria

Whenever we are trying to reconstruct a function from sample points we are hoping that the reconstruction process performs *well* and we don't get many *artifacts*. However our understanding of such terms like *good reconstruction* as well as *artifacts* during this process is usually highly dependent on the specific application. Most applications share an attempt to recover the original sampled function as accurately as possible. In order to measure the accuracy of the process, one must have an idea about the type of original function from which the samples were recovered. As we have pointed out in Section 2.1, it is not restrictive to most applications to assume that the original function is continuous to some degree n and therefore belongs to the class of functions C^n. This is the only assumption, that we require for our filter design.

In addition of hoping for an accurate function reconstruction, almost all applications will require the reconstruction of a smooth function. Since we assume a smooth original function, it is natural to expect a smooth function as the result of the reconstruction process. A smooth reconstruction will also guarantee the disappearances of image artifacts in visualization and imaging applications. Another application is CAD in which designers reconstruct surfaces from a set of sample points (knots) using basis functions that are developed to yield surfaces of C^1, C^2 or higher continuity. The reason for smooth function reconstruction is that our visual system is capable of detecting and enhancing even small discontinuities in images. For example, in Fig. 2b and Fig. 4b (see color plates) we reconstructed a test function (introduced in [15]) and an MRI data set with a filter that was designed solely by requiring high accuracy [17] and therefore yields very little absolute error. It, however, suffers from discontinuities, leading to a discontinuous reconstructed function. Consequently, we would like to reconstruct a function, that is a member of the continuous function space C^n.

For practical applications, the efficiency of the reconstruction process is of great importance as well. In volume rendering, the efficiency of a reconstruction filter, which is employed routinely many times [1], is a source of great concern. It is desirable to use as few samples as possible in order to reconstruct the function at a new location.

We conclude that for general filter design we have to answer three questions:

- What derivative of the original function do we want to reconstruct?
- What accuracy do we require from the reconstruction process?
- What space C^n should the reconstructed function belong to?

Commonly there is also the question of how many filter weights should the filter have. We have elegantly answered this question by minimizing the number of weights and by designing the most efficient filter fulfilling the constraints of the design.

The first two questions can easily be expressed using the framework developed in [17]. Assuming that we want to reconstruct the k-th derivative of the given digital signal f, we simply require that all error coefficients a_n^w in Equation 2 be zero, where $n < k$. Further, we require that the coefficient of the kth derivative be one. Formally, this can be expressed as:

<u>*Condition 1*</u>: $a_n^w = 0$ for all $n < k$ and $a_k^w = 1$.

The major goal of the design in spatial domain is numerical accuracy. We gain numerical accuracy by requiring the error coefficients a_n^w beyond k to be zero. This leads to what we call N-EF filters (that is, Error Function of the Nth order) ([17]):

<u>*Condition 2:*</u> $a_n^w = 0$ for all $k < n < N + k - 1$.

It is no restriction to consider the filter w to be composed of elements w_k, which are defined by

$$w_k = w_k(\tau) = w(\tau + k).$$

An example of this piecewise decomposition of the filter w is illustrated in Fig. 1. Now it is easy to see that the first two conditions yield an equation system in the unknowns w_k. The solution of that equation system will define a filter w, that fulfills *Condition 1* and *2*. Since this is a linear equation system, we can easily solve it symbolically by Gaussian elimination, which yields a solution for the w_k. This concludes the first step of our function design, defining a class of filters, that guarantee an N-EF accurate reconstruction of the kth derivative of the original function.

As we have pointed out earlier, not every filter of this class yields a continuous reconstructed function and therefore might lead to undesirable artifacts. Hence, we desire the reconstructed function to be part of a smooth function space C^M. From Equation 1 it is clear that we need to require our filter w to be in this class. In order to be a member of C^M, a function w and its M derivatives must all

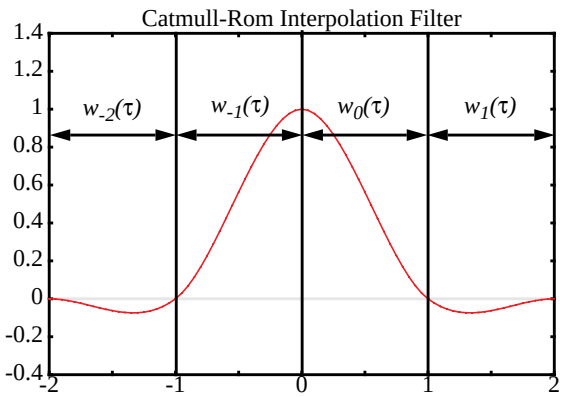


FIGURE 1. A piecewise polynomial interpolation filter using four filter weights $w_k = w_k(\tau) = w(\tau + k)$. Essentially the filter has p parts. For both symmetric and anti-symmetric filters p is even.

be continuous everywhere, including every open interval $(k, k+1)$ for every integer k, and also at all the integer points k themselves. Since the equation system of *Conditions 1* and *2* yields a piecewise filter kernel, we can mathematically express the smoothness criteria as:

<u>*Condition 3:*</u> $w_k(\tau) \in C^M$ and $w_k^{(m)}(1) = w_{k+1}^{(m)}(0)$ for all k and all $m < M$, where $w_k^{(m)}$ denotes the m-th derivative of w_k.

After solving the equation system of *Conditions 1* and *2* and determining the smoothness of the desired filter, we have a new set of criteria for our filter that needs to be met. In order to design an actual filter, we have to find a solution that fulfills all these conditions. While it is not necessary to restrict oneself to piecewise polynomial filters, we have done so here. The reason for this is that they are easy to use and implement, and are therefore very popular. Now *Conditions 1, 2* and *3* translate to a linear equation system in the coefficients of piecewise polynomials. The solution of this equation system yields a class of polynomials. These can be further restricted by choosing efficient filters, i.e. with the least number of filter weights and small degrees of polynomials. This concludes the filter design. Summarizing the filter design includes the following steps:

<u>Step 1:</u> Solve a linear equation system created by Conditions 1 and 2 in the pieces w_k of the filter w.

<u>Step 2:</u> Choose a set of basis functions for the representation of w_k.

<u>Step 3:</u> Solve for the coefficients of the basis functions, considering Condition 3 as well as the solution of Step 1.

Our design criteria also have validity in the frequency domain. It can be shown that our accuracy criteria, defined by specifying the error coefficients a_k^w, translate to conditions on the frequency representation of the filter w at the DC value. a_0^w represents the DC value itself and a_k^w the k-th derivative of the frequency spectra at that point. This is a very desirable condition and was suggested as a filter design criteria by Dutta Roy et al. [7], for designing maximal linear filters. Since it is impractical to use an ideal reconstruction filter (in the C^∞ sense), their idea was to design filters that come very close to the ideal filter in parts of the frequency spectra and includes some important frequencies. For general applications, we would expect to have the most important frequencies around the DC value.

Since the accuracy criteria only fixes the frequency domain at a single point, it is not enough to guarantee well behaved filters. Our smoothness criterion in Condition 3 constructs filters w of the class C^M. That means they can be decomposed (using a Taylor series) into a polynomial of Mth degree and a remainder term. Now, the polynomial of Mth degree translates into a function defined as $\omega^{-(M-1)}$ in frequency space [3]. This guarantees a quick decay of our reconstruction filter. The higher the smoothness condition, the quicker the decay. This ensures that aliasing effects of our designed filter diminish with increasing M.

Having explained the general design process, we turn to demonstrate it by ways of an example.

4. EXAMPLE

Let us assume we want to construct a derivative filter. We expect this derivative filter to be somewhat reliable in terms of accuracy, so we choose a 2EF filter. Further, we aim for a C^1 continuous filter. That leads to three conditions to fulfill:

1. derivative filter:

$$\begin{aligned} a_0(\tau) &= 0 \\ a_1(\tau) &= 1 \end{aligned} \tag{3}$$

2. numerical accuracy 2EF:

$$a_2(\tau) = 0 \tag{4}$$

3. smoothness C^1:

$$w \in C^1 \tag{5}$$

Here $a_k(\tau)$ are the error coefficients defined in Equation 2 with a positive offset τ, $0 \le \tau < 1$. The filter to be constructed is w. Decomposing our filter w in pieces w_k as mentioned in Section 3, we can write the three conditions above in terms of the filter weights $w_k = w_k(\tau) = w(\tau + k)$. We will use the notation w_k and $w_k(\tau)$ interchangeably. Using the definition of the error coefficients in Equation 2 will simplify the conditions of Equation 3 through Equation 5 to:

1. derivative filter:

$$\begin{aligned} w_{-2} + w_{-1} + w_0 + w_1 &= 0 \\ T\left((2-\tau)\, w_{-2} + (1-\tau)\, w_{-1} + (-\tau)\, w_0 + (-1-\tau)\, w_1 \right) &= 1 \end{aligned} \tag{6}$$

2. numerical accuracy 2EF:

$$(2-\tau)^2 w_{-2} + (1-\tau)^2 w_{-1} + (-\tau)^2 w_0 + (-1-\tau)^2 w_1 = 0 \tag{7}$$

3. smoothness: (here w'_n denotes the derivative of w_n)

$$w_k \in C^1 \qquad \text{and}$$

$$\begin{aligned} w_{-2}(0) &= 0 & & & w'_{-2}(0) &= 0 \\ w_{-2}(1) &= w_{-1}(0) & & & w'_{-2}(1) &= w'_{-1}(0) \\ w_{-1}(1) &= w_0(0) & & \text{and} & w'_{-1}(1) &= w'_0(0) \\ w_0(1) &= w_1(0) & & & w'_0(1) &= w'_1(0) \\ w_1(1) &= 0 & & & w'_1(1) &= 0 \end{aligned} \tag{8}$$

The choice of the number of filter weights w_k (which is the same as the number of piecewise, non-zero parts of the function w) is rather arbitrary. If we choose too many, the resulting filter becomes inefficient. If we choose too few, the equation system might not lead to a solution at all. Since we are trying to design cost-efficient filters, we'd like to have as few as possible filter weights. Since, in computer graphics, we are interested in anti-symmetric derivative filters (symmetric interpolation filters) and the weights $w_k(\tau)$ are defined over integer intervals, we always need an even number of weights (Fig. 1). Conditions 1 and 2 already impose three equations on the filter weights, thus we expect at least four weights to be necessary for our resulting filter.

The equation system in Equation 6 and Equation 7 has three equations in the four unknowns w_k. Therefore it is under-determined and leads to the following set of solutions (setting T to 1):

$$\begin{aligned} w_1 &= w_1 \\ w_0 &= -3w_1 + \tau - \frac{3}{2} \\ w_{-1} &= 3w_1 - 2\tau + 2 \\ w_{-2} &= -w_1 + \tau - \frac{1}{2} \end{aligned} \tag{9}$$

Any filter w whose filter weights fulfill Equation 9 is guaranteed to be a 2EF first derivative filter. The actual filter can be constructed using specific basis functions for the w_k and insuring that our smoothness condition (Equation 8) is fulfilled. An obvious choice for the C^1 continuous basis function would be polynomials, since polynomials are a member of C^∞. Using our notation $w_k = w_k(\tau)$, we require:

$$w_k(\tau) = C_k\tau^2 + B_k\tau + A_k ,$$

where the coefficients C_k, B_k, A_k are unknown and remain to be determined. Here again the choice of a second order polynomial is rather arbitrary. If we choose too high of a degree, we get an inefficient, parameter depending solution. For too low of a polynomial degree, we might not get a solution, since not all constraints on the w_k from Equation 9 and *Condition 3* can be fulfilled. Substituting this definition of the filter weights w_k into Equation 8 yields the following condition on their coefficients:

$$\begin{aligned} A_{-2} &= 0 & & & B_{-2} &= 0 \\ C_{-2} + B_{-2} + A_{-2} &= A_{-1} & & & 2C_{-2} + B_{-2} &= B_{-1} \\ C_{-1} + B_{-1} + A_{-1} &= A_0 & & \text{and} & 2C_{-1} + B_{-1} &= B_0 \\ C_0 + B_0 + A_0 &= A_1 & & & 2C_0 + B_0 &= B_1 \\ C_1 + B_1 + A_1 &= 0 & & & 2C_1 + B_1 &= 0 \end{aligned}$$

Substituting the polynomial definition of the filter weights w_k into Equation 9 yields these conditions on their coefficients:

$$\begin{aligned} C_0 &= -3C_1 & B_0 &= -3B_1 + 1 & A_0 &= -3A_1 - \frac{3}{2} \\ C_{-1} &= 3C_1 & B_{-1} &= 3B_1 - 2 & A_{-1} &= 3A_1 + 2 \\ C_{-2} &= -C_1 & B_{-2} &= -B_1 + 1 & A_{-2} &= -A_1 - \frac{1}{2} \end{aligned}$$

Further, requiring anti-symmetric filters, yields the following conditions:

$$w_{k-1}(\tau) = -w_{-k}(1-\tau) ,$$

which translates to

$$\begin{aligned} C_{k-1} &= -C_{-k} \\ B_{k-1} &= 2C_{-k} + B_{-k} \\ A_{k-1} &= C_{-k} + B_{-k} + A_{-k} \end{aligned}$$

in all positive k. This leads to an equation system in the coefficients, which solved and substituted into Equation 9, leads to the following filter weights:

$$\begin{aligned} w_1 &= -\frac{1}{2}\tau^2 + \tau - \frac{1}{2} \\ w_0 &= \frac{3}{2}\tau^2 - 2\tau \\ w_{-1} &= -\frac{3}{2}\tau^2 + \tau + \frac{1}{2} \\ w_{-2} &= \frac{1}{2}\tau^2 \end{aligned} .$$

This concludes our filter design of a C^1 2EF first derivative filter.

In the Appendix we list all the interpolation and first derivative filters that we constructed using different accuracy and smoothness criteria. Because of space constraint we have only given the polynomial coefficients in a matrix M. The filter weights are computed by:

$$\begin{bmatrix} w_{-3}(\tau) \\ w_{-2}(\tau) \\ w_{-1}(\tau) \\ w_0(\tau) \\ w_1(\tau) \\ w_2(\tau) \end{bmatrix} = M \begin{bmatrix} \tau^3 \\ \tau^2 \\ \tau \\ 1 \end{bmatrix}$$

for a cubic filter with 6 weights. The size of M is adapted by the size of the filter and the degree of the polynomial. All filters are laid out in a table where the rows represent the smoothness criteria and the columns represent the accuracy criteria. We have looked at 1EF through 4EF filters and C^0 through C^3 smoothness criteria. We also included filters that were constructed without Condition 3 - smoothness. That simply leads to discontinuous filters. Those filters might be of interest for applications that care about accuracy only, for example in cases where the resulting function is used for measurement, rather than visual inspection. The advantage of these filters is that they are sometimes faster while having only lit-

tle (numerical) error. We have argued in [17] that these filters might even lead to reasonable images under certain conditions and therefore represent an efficient alternative that should not be discarded.

The most general usable filters are probably C^0, C^1 and 2EF, 3EF filters. For the interpolation filters we find that the most efficient C^1-3EF filter is the well known Catmull-Rom spline, also found by others to be the most accurate BC-spline. It is also noteworthy that this filter is not the best filter in the class of cubic BC-splines in terms of smoothness. We also found that the BC-filter for which B=1 and C=0 is a C^2-2EF filter. Therefore this filter might be preferable over the Catmull-Rom spline for some applications. In order to improve on the Catmull-Rom filter in terms of accuracy one requires 6 filter weights. In order to improve smoothness of the reconstructed function while maintaining the same accuracy, one has to choose at least a fourth degree polynomial. The best filter with just 2 filter weights would be either a 2EF or a C^3 continuous filter.

For derivative filters, the filter C^1-2EF is probably a good first derivative filter. It is one of the best possible that only requires 4 filter weights and is still only a quadratic filter. In order to improve on it, we would either have to go to 6 filter weights or to a fourth degree polynomial. It is also interesting to note, that this filter is the analytic derivative of the C^2-2EF interpolation filter, which was a BC-spline with B=1 and C=0.

5. EXPERIMENTS

The images were rendered employing a simple ray-caster to find the iso-surfaces. The volumes were sampled at an interval of 0.05 voxel lengths. At each sampling point, the ray-caster first applied an interpolation kernel (we used the Catmull-Rom cubic spline) to reconstruct the function at that point. If the reconstructed value was above a pre-set iso-value, the derivative filter was used to compute the 3D gradient. Shading was then performed using the traditional Phong lighting model [8] with diffuse and specular reflections. The obtained color and opacity were composited with the previous ray values, and the ray was terminated after the opacity reached a value close to 1.0. Since both the interpolation and the derivative kernel were separable, for all our filters, the filtering operations could be efficiently performed using a scheme similar to the one given in [1] and [18].

For our experiments we used an analytic data set and an MRI data set. The analytic data set is derived from the same function as the one used by Marschner and Lobb [15]. Since, due to spatial constraints, it is not possible to include the entire set of images that can be obtained using all given filters, summarized in Table 1 and Table 2, we have chosen the discontinuous and C^0 1EF filters as well as the discontinuous and C^0 3EF filters. Fig. 2 (Fig. 2,4,5 in color plates) shows the synthetic data set. In order to better visualize the influence of the filters we also computed the angular error images. For each reconstructed normal we computed the actual normal and recorded their angular difference. The grey value of 255 was displayed for an angular error of 15 degrees. The discontinuous 1EF filter is simply the well known central difference filter, and the discontinuous 3EF filter is the filter that we have found to be a filter yielding better accuracy in our previous work [17]. Here it is clearly visible, that filter design solely based on accuracy criteria will not lead to acceptable images. Adding a simple smoothness constraint, which is reflected in the C^0 filters, results in very accurate images that are free of visible artifacts. It also becomes very clear, especially in the error images of Fig. 3, that the 3EF filter will lead superior images in terms of numerical accuracy.

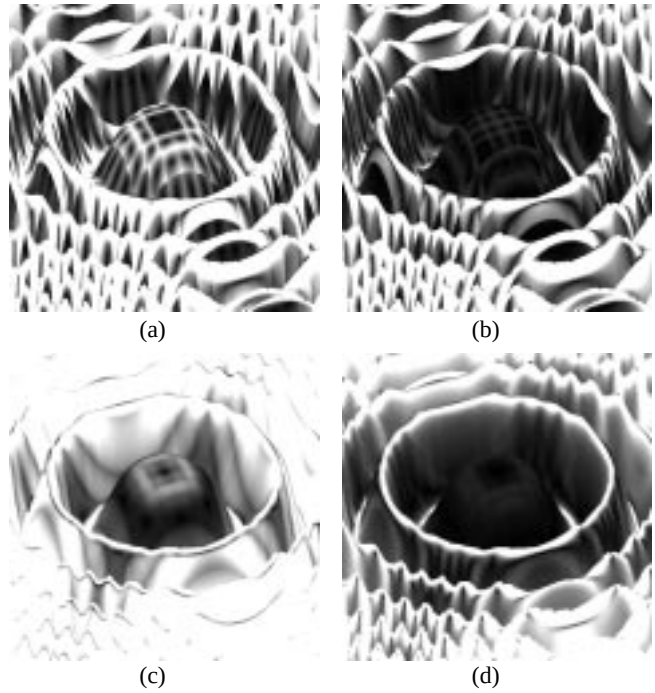


FIGURE 3. Error images of the Marschner Lobb data set rendered using the following derivative filter (a) discontinuous 1EF (b) discontinuous 3EF (c) C^0-1EF (d) C^0-3EF; darker colors mean lesser error

The same behavior as for the analytic data set can also be observed for the MRI data set in Fig. 4. This data set is a close up view of an MRI of a human brain. Here, we also fixed the interpolation filter to the Catmull-Rom filter and varied the derivative filter in the same way as we did for the Marschner Lobb images.

Another application requiring smooth reconstruction filters is the size preserving pattern mapping of Kurzion et al. [12]. Here, the problem is to continuously map a texture to a parametric surface or implicit surface, including volumetric iso-surfaces, at a constant density. In the past, only manual mappings were able to perform this task, while this paper introduces an automatic method. The authors use the curvature of a surface at a point in order to continuously vary the scale of the mapped image. This curvature is approximated using the derivative of the underlying function. A C^1 continuous filter is essential for the success of this method as it ensures continuous mapping of texture on the surface. We used a 16^3 grid (a shrunken down version of the original 128^3 head) for calculating the curvature. This means that the head is composed of rectangular patches on which the normal derivative is calculated by the same 4^3 grid samples. Fig. 5a uses the central difference filter, which gives a very poor estimation of the curvature, hence the mapping of the density varies sharply between patches. Fig. 5b uses a C^0-2EF filter that generates a very constant density across the head, but shows discontinuities along the patch lines. The filter we designed for this application is the C^1 continuous 2EF derivative filter of Table 2. Fig. 5c shows an application of this filter and we observe that all previous problems no longer exist.

6. FURTHER RESEARCH

Since, in this paper, we restricted ourselves to piecewise polynomial filters, we would like to explore different basis functions. Especially using trigonometric basis functions like *sine* and *cosine*, we would like to explore possible similarities to windowed *Sinc* and *Cosc* filters. We hope that this might improve the efficiency of the filters, maintaining the same smoothness and accuracy while using fewer filter weights.

Our accuracy criteria, as outlined here, have also been used to specify and design filters in other domains, especially in wavelet-based multiresolution analyses, by Daubechies and others [5]. We hope to explore the effect of our smoothness criteria also for defining new basis functions in the wavelet domain and to study its effects on the wavelet transform.

In applications of computer graphics, especially scientific visualization, the final image is not just influenced by the interpolation and derivative method, but sometimes also by a shading equation and compositing operations. Therefore, it is necessary to analyze the overall error expressed in the L^2 error norm. We are working on developing better tools to study this error. Once we are able to characterize the complete rendering pipeline, we might be able to adapt the reconstruction filters in a way to compensate for the overall error and to produce better images.

7. CONCLUSIONS

In this work, we have introduced design criteria for the design of optimal and smooth reconstruction filters. It was demonstrated, contrary to previous beliefs, that it is not sufficient to base filter designs just on accuracy criteria. Rather, our design criteria are not only the accuracy of the reconstructed function, but also its characterization as a function in the space C^M of all M-times continuous derivable functions. We further demonstrated, by ways of an example, a C^1 2EF filter, how such a filter can be constructed. We then provided a table, listing all optimal interpolation and derivative filters that match smoothness and accuracy criteria of up to C^3 continuous and 4EF, respectively. These tables may serve as a guide on what filters should be used for a certain application. The filters listed in these tables go much beyond the very popular class of piecewise polynomial filters, described in the past by Mitchell and Netravali's BC-splines [16]. Furthermore, this paper gives the practitioner an easy way to design specific filters to match their specific application, by determining a very application oriented set of criteria - smoothness and accuracy.

8. ACKNOWLEDGMENTS

We thank Prof. Ed Overman of the Department of Mathematics at Ohio State University for providing useful background information and important references. We thank Prof. Wayne Carlson and the Advanced Computing Center for the Arts and Design for the use of their computing facilities and Prof. Robert Moorehead of the NSF Engineering Research Center, Mississippi State University for providing encouragement and support.

9. REFERENCES

[1] Bentum M.J., Lichtenbelt B.B.A., Malzbender T., "Frequency Analysis of Gradient Estimators in Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, 2(3):242-254, September 1996.

[2] Bracewell R.N., *Two Dimensional Imaging*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1995.

[3] Bronstein E.N., Semendjajew K.A., "Taschenbuch der Mathematik," 22nd ed., G. Grosche, V. Ziegler, B. Ziegler: editors, BSB B.G. Teubner Verlagsgesellschaft, Leipzig, 1985.

[4] Carlbom I., "Optimal Filter Design for Volume Reconstruction and Visualization", *Proceedings of IEEE Conference on Visualization '93*, pp. 54-61, October 1993.

[5] Daubechies I., *Ten Lectures on Wavelets*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA, 1992.

[6] Drebin R.A., L. Carpenter L., Hanrahan P., "Volume Rendering", *Computer Graphics*, 22(4):51-58, August 1988.

[7] Dutta Roy S.C., Kumar B., "Digital Differentiators", in *Handbook of Statistics*, N. K. Bise and C. R. Rao eds., vol. 10: 159-205, 1993.

[8] Foley J.D., van Dam A., Feiner S.K., Hughes J.F., *Computer Graphics, Principles and Practice*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1990.

[9] Goss M.E., "An Adjustable Gradient Filter for Volume Visualization Image Enhancement", *Proceedings of Graphics Interface'94*, pp. 67-74, Toronto, Ontario, 1994.

[10] Hamming R.W., *Digital Filters*, Prentice Hall Inc., Second Edition, Englewoods Cliffs, NJ, 1983.

[11] Keys R.G., "Cubic Convolution Interpolation for Digital Image Processing", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29(6):1153-1160, December 1981.

[12] Kurzion Y., Möller T., Yagel R., "Size Preserving Pattern Mapping", in *IEEE Conference on Visualization '98*.

[13] Levoy M., "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, 8(5):29-37, May 1988.

[14] Machiraju R.K., Yagel R., "Reconstruction Error Characterization and Control: A Sampling Theory Approach", *IEEE Transactions on Visualization and Computer Graphics*, TVCG 2(4):364-376, December 1996.

[15] Marschner S.R. and Lobb R.J., "An Evaluation of Reconstruction Filters for Volume Rendering", *Proceedings of Visualization '94*, IEEE CS Press, pp. 100-107, October 1994.

[16] Mitchell D.P., Netravali A.N., "Reconstruction Filters in Computer Graphics", *Computer Graphics*, 22(4):221-228, August 1988.

[17] Möller T., Machiraju R.K., Mueller K., Yagel R., "Evaluation and Design of Filters Using a Taylor Series Expansion", *IEEE Transactions on Visualization and Computer Graphics*, ITVCG 3(2): 184-199, June 1997.

[18] Möller T., Machiraju R.K., Mueller K., Yagel R., "A Comparison Of Normal Estimation Schemes", *Proceedings of IEEE Conference on Visualization '97*, pp. 19-26, October 1997.

[19] Oppenheim A.V., Schafer R.W., *Discrete-Time Signal Processing*, Prentice Hall Inc., Englewoods Cliffs, NJ, 1989.

[20] Park S.K., Schowengerdt R.A., "Image Reconstruction by Parametric Cubic Convolution", *Computer Vision, Graphics, and Image Processing*, vol. 23:258-272, 1983

[21] Warsi Z.U.A, *Fluid Dynamics: Theoretical and Computational Approaches*, CRC Press, Boca Raton, FL, 1992.

10. APPENDIX

10.1 Interpolation filters

TABLE 1. the new actual filters for interpolation filters

	1EF	2EF	3EF	4EF
C-	$\begin{bmatrix} w_{-1} \\ w_0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{4} & 2a-\frac{1}{4} & -a \\ -\frac{1}{4} & -\left(6a-\frac{5}{4}\right) & 3a \\ -\frac{1}{4} & 6a-\frac{3}{4} & -(3a-1) \\ \frac{1}{4} & -\left(2a+\frac{1}{4}\right) & a \end{bmatrix}$	$\begin{bmatrix} \frac{1}{6} & 0 & -\frac{1}{6} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & -1 & -\frac{1}{2} & 1 \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{3} & 0 \end{bmatrix}$
C0	$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{4} & \frac{5}{4} & 0 \\ -\frac{1}{4} & -\frac{3}{4} & 1 \\ \frac{1}{4} & -\frac{1}{4} & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{6} & 0 & -\frac{1}{6} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & -1 & -\frac{1}{2} & 1 \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{3} & 0 \end{bmatrix}$
C1	$\begin{bmatrix} -2 & 3 & 0 & 0 \\ 2 & -3 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{4} & 0 & 0 \\ -\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ -\frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ -\frac{3}{2} & 2 & \frac{1}{2} & 0 \\ \frac{3}{2} & -\frac{5}{2} & 0 & 1 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \end{bmatrix}$	$\begin{bmatrix} -a & \left(\frac{3}{2}a-\frac{1}{24}\right) & 0 & 0 \\ \left(5a+\frac{1}{6}\right) & -\left(\frac{15}{2}a-\frac{1}{8}\right) & -\frac{1}{12} & \frac{1}{2}a-\frac{1}{24} \\ -\left(10a+\frac{1}{2}\right) & 15a+\frac{5}{12} & \frac{2}{3} & -\left(2a-\frac{1}{6}\right) \\ 10a+\frac{1}{2} & -\left(15a+\frac{13}{12}\right) & 0 & 3a+\frac{3}{4} \\ -\left(5a+\frac{1}{6}\right) & \frac{15}{2}a+\frac{5}{8} & -\frac{2}{3} & -\left(2a-\frac{1}{6}\right) \\ a & -\left(\frac{3}{2}a+\frac{1}{24}\right) & \frac{1}{12} & \frac{1}{2}a-\frac{1}{24} \end{bmatrix}$
C2	$\begin{bmatrix} 6 & -15 & 10 & 0 & 0 & 0 \\ -6 & 15 & -10 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & -1 & 0 & \frac{2}{3} \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \end{bmatrix}$	$\begin{bmatrix} -1 & \frac{5}{2} & -\frac{3}{2} & 0 & 0 & 0 \\ 3 & -\frac{15}{2} & \frac{9}{2} & \frac{1}{2} & \frac{1}{2} & 0 \\ -3 & \frac{15}{2} & -\frac{9}{2} & -1 & 0 & 1 \\ 1 & -\frac{5}{2} & \frac{3}{2} & \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix}$	$\begin{bmatrix} -\frac{1}{36} & 0 & 0 & 0 \\ \frac{11}{36} & -\frac{1}{12} & -\frac{1}{12} & -\frac{1}{36} \\ -\frac{7}{9} & \frac{5}{6} & \frac{2}{3} & \frac{1}{9} \\ \frac{7}{9} & -\frac{3}{2} & 0 & \frac{5}{6} \\ -\frac{11}{36} & \frac{5}{6} & -\frac{2}{3} & \frac{1}{9} \\ \frac{1}{36} & -\frac{1}{12} & \frac{1}{12} & -\frac{1}{36} \end{bmatrix}$
C3	$\begin{bmatrix} -20 & 70 & -84 & 35 & 0 & 0 & 0 & 0 \\ 20 & -70 & 84 & -35 & 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -\frac{1}{10} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{3}{10} & -\frac{3}{4} & 0 & \frac{1}{2} & \frac{1}{2} & \frac{3}{20} \\ -\frac{3}{10} & \frac{3}{4} & 0 & -1 & 0 & \frac{7}{10} \\ \frac{1}{10} & -\frac{1}{4} & 0 & \frac{1}{2} & -\frac{1}{2} & \frac{3}{20} \end{bmatrix}$	$\begin{bmatrix} 3 & -\frac{21}{2} & \frac{25}{2} & -5 & 0 & 0 & 0 & 0 \\ -9 & \frac{63}{2} & -\frac{75}{2} & 15 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 9 & -\frac{63}{2} & \frac{75}{2} & -15 & 0 & -1 & 0 & 1 \\ -3 & \frac{21}{2} & -\frac{25}{2} & 5 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{30} & -\frac{1}{16} & 0 & 0 & 0 & 0 \\ -\frac{1}{6} & \frac{17}{48} & \frac{1}{12} & -\frac{1}{24} & -\frac{1}{12} & -\frac{7}{240} \\ \frac{1}{3} & -\frac{19}{24} & -\frac{1}{6} & \frac{2}{3} & \frac{2}{3} & \frac{7}{60} \\ -\frac{1}{3} & \frac{7}{8} & 0 & \frac{5}{4} & 0 & \frac{33}{40} \\ \frac{1}{6} & -\frac{23}{48} & \frac{1}{6} & \frac{2}{3} & -\frac{2}{3} & \frac{7}{60} \\ -\frac{1}{30} & \frac{5}{48} & -\frac{1}{12} & -\frac{1}{24} & \frac{1}{12} & -\frac{7}{240} \end{bmatrix}$

10.2 First Derivative filters

TABLE 2. the new actual derivative filters

	1EF	2EF	3EF	4EF
C-	$\begin{bmatrix} w_{-2} \\ w_{-1} \\ w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{2} & a \\ -\frac{1}{2} & \frac{1}{2}-3a \\ -\frac{1}{2} & 3a \\ \frac{1}{2} & -\frac{1}{2}-a \end{bmatrix}$	$\begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{6} \\ -\frac{3}{2} & 1 & 1 \\ \frac{3}{2} & -2 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{3} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{12} & a & -\left(a+\frac{1}{6}\right) & b \\ -\frac{1}{4} & -\left(5a-\frac{1}{4}\right) & 5a+\frac{3}{4} & -\left(5b+\frac{1}{12}\right) \\ \frac{1}{6} & 10a-\frac{1}{2} & -\left(10a+\frac{1}{3}\right) & 10b+\frac{2}{3} \\ \frac{1}{6} & -10a & 10a-\frac{5}{6} & -10b \\ -\frac{1}{4} & 5a+\frac{1}{2} & -\left(5a-\frac{1}{2}\right) & 5b-\frac{2}{3} \\ \frac{1}{12} & -\left(a+\frac{1}{4}\right) & a+\frac{1}{12} & -\left(b-\frac{1}{12}\right) \end{bmatrix}$
C0	$\begin{bmatrix} \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & 0 \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & 0 \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} a & -\left(a+\frac{1}{12}\right) & 0 \\ -\left(5a-\frac{1}{2}\right) & 5a+\frac{1}{4} & -\frac{1}{12} \\ 10a-\frac{3}{2} & -\left(10a-\frac{5}{6}\right) & \frac{2}{3} \\ -\left(10a-\frac{3}{2}\right) & 10a-\frac{13}{6} & 0 \\ 5a-\frac{1}{2} & -\left(5a-\frac{5}{4}\right) & -\frac{2}{3} \\ -a & a-\frac{1}{12} & \frac{1}{12} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{12} & a & -\left(a+\frac{1}{6}\right) & 0 \\ -\frac{1}{4} & -\left(5a-\frac{1}{4}\right) & 5a+\frac{3}{4} & -\frac{1}{12} \\ \frac{1}{6} & 10a-\frac{1}{2} & -\left(10a+\frac{1}{3}\right) & \frac{2}{3} \\ \frac{1}{6} & -10a & 10a-\frac{5}{6} & 0 \\ -\frac{1}{4} & 5a+\frac{1}{2} & -\left(5a-\frac{1}{2}\right) & -\frac{2}{3} \\ \frac{1}{12} & -\left(a+\frac{1}{4}\right) & a+\frac{1}{12} & \frac{1}{12} \end{bmatrix}$
C1	$\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -\frac{3}{2} & 1 & \frac{1}{2} \\ \frac{3}{2} & -2 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{2} & 0 & 0 \\ -\frac{3}{2} & 1 & \frac{1}{2} \\ \frac{3}{2} & -2 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} -\frac{1}{12} & 0 & 0 \\ \frac{11}{12} & -\frac{1}{6} & -\frac{1}{12} \\ -\frac{7}{3} & \frac{5}{3} & \frac{2}{3} \\ \frac{7}{3} & -3 & 0 \\ -\frac{11}{12} & \frac{5}{3} & -\frac{2}{3} \\ \frac{1}{12} & -\frac{1}{6} & \frac{1}{12} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{12} & -\frac{1}{6} & 0 & 0 \\ -\frac{1}{4} & \frac{13}{12} & -\frac{1}{12} & -\frac{1}{12} \\ \frac{1}{6} & -\frac{13}{6} & \frac{4}{3} & \frac{2}{3} \\ \frac{1}{6} & \frac{5}{3} & -\frac{5}{2} & 0 \\ -\frac{1}{4} & -\frac{1}{3} & \frac{4}{3} & -\frac{2}{3} \\ \frac{1}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{1}{12} \end{bmatrix}$
C2	$\begin{bmatrix} -\frac{1}{2} & 1 & 0 & 0 & 0 \\ \frac{3}{2} & -3 & 0 & 1 & \frac{1}{2} \\ -\frac{3}{2} & 3 & 0 & -2 & 0 \\ \frac{1}{2} & -1 & 0 & 1 & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} -\frac{1}{2} & 1 & 0 & 0 & 0 \\ \frac{3}{2} & -3 & 0 & 1 & \frac{1}{2} \\ -\frac{3}{2} & 3 & 0 & -2 & 0 \\ \frac{1}{2} & -1 & 0 & 1 & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{6} & -\frac{1}{4} & 0 & 0 & 0 \\ -\frac{5}{6} & \frac{17}{12} & \frac{1}{4} & -\frac{1}{12} & -\frac{1}{12} \\ \frac{5}{3} & -\frac{19}{6} & -\frac{1}{2} & \frac{4}{3} & \frac{2}{3} \\ -\frac{5}{3} & \frac{7}{2} & 0 & -\frac{5}{2} & 0 \\ \frac{5}{6} & -\frac{23}{12} & \frac{1}{2} & \frac{4}{3} & -\frac{2}{3} \\ -\frac{1}{6} & \frac{5}{12} & -\frac{1}{4} & -\frac{1}{12} & \frac{1}{12} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{6} & -\frac{1}{4} & 0 & 0 & 0 \\ -\frac{5}{6} & \frac{17}{12} & \frac{1}{4} & -\frac{1}{12} & -\frac{1}{12} \\ \frac{5}{3} & -\frac{19}{6} & -\frac{1}{2} & \frac{4}{3} & \frac{2}{3} \\ -\frac{5}{3} & \frac{7}{2} & 0 & -\frac{5}{2} & 0 \\ \frac{5}{6} & -\frac{23}{12} & \frac{1}{2} & \frac{4}{3} & -\frac{2}{3} \\ -\frac{1}{6} & \frac{5}{12} & -\frac{1}{4} & -\frac{1}{12} & \frac{1}{12} \end{bmatrix}$
C3	$\begin{bmatrix} -\frac{3}{4} & \frac{5}{4} & 0 & 0 & 0 & 0 \\ \frac{3}{4} & 0 & -\frac{5}{2} & 0 & \frac{5}{4} & \frac{1}{2} \\ \frac{3}{4} & -\frac{15}{4} & 5 & 0 & -\frac{5}{2} & 0 \\ -\frac{3}{4} & \frac{5}{2} & -\frac{5}{2} & 0 & \frac{5}{4} & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} 1 & -3 & \frac{5}{2} & 0 & 0 & 0 & 0 \\ -3 & 9 & -\frac{15}{2} & 0 & 0 & 1 & \frac{1}{2} \\ 3 & -9 & \frac{15}{2} & 0 & 0 & -2 & 0 \\ -1 & 3 & -\frac{5}{2} & 0 & 0 & 1 & -\frac{1}{2} \end{bmatrix}$	$\begin{bmatrix} \frac{3}{16} & -\frac{13}{48} & 0 & 0 & 0 & 0 \\ -\frac{9}{16} & \frac{5}{12} & \frac{19}{24} & \frac{1}{4} & -\frac{7}{48} & -\frac{1}{12} \\ \frac{3}{8} & \frac{25}{24} & -\frac{19}{6} & -\frac{1}{2} & \frac{19}{12} & \frac{2}{3} \\ \frac{3}{8} & -\frac{35}{12} & \frac{19}{4} & 0 & -\frac{23}{8} & 0 \\ -\frac{9}{16} & \frac{115}{48} & -\frac{19}{6} & \frac{1}{2} & \frac{19}{12} & -\frac{2}{3} \\ \frac{3}{16} & -\frac{2}{3} & \frac{19}{24} & -\frac{1}{4} & -\frac{7}{48} & \frac{1}{12} \end{bmatrix}$	$\begin{bmatrix} -\frac{1}{4} & \frac{3}{4} & -\frac{7}{12} & 0 & 0 & 0 & 0 \\ \frac{5}{4} & -\frac{15}{4} & \frac{35}{12} & \frac{1}{6} & \frac{1}{4} & -\frac{1}{12} & -\frac{1}{12} \\ -\frac{5}{2} & \frac{15}{2} & -\frac{35}{6} & -\frac{2}{3} & -\frac{1}{2} & \frac{4}{3} & \frac{2}{3} \\ \frac{5}{2} & -\frac{15}{2} & \frac{35}{6} & 1 & 0 & -\frac{5}{2} & 0 \\ -\frac{5}{4} & \frac{15}{4} & -\frac{35}{12} & -\frac{2}{3} & \frac{1}{2} & \frac{4}{3} & -\frac{2}{3} \\ \frac{1}{4} & -\frac{3}{4} & \frac{7}{12} & \frac{1}{6} & -\frac{1}{4} & -\frac{1}{12} & \frac{1}{12} \end{bmatrix}$

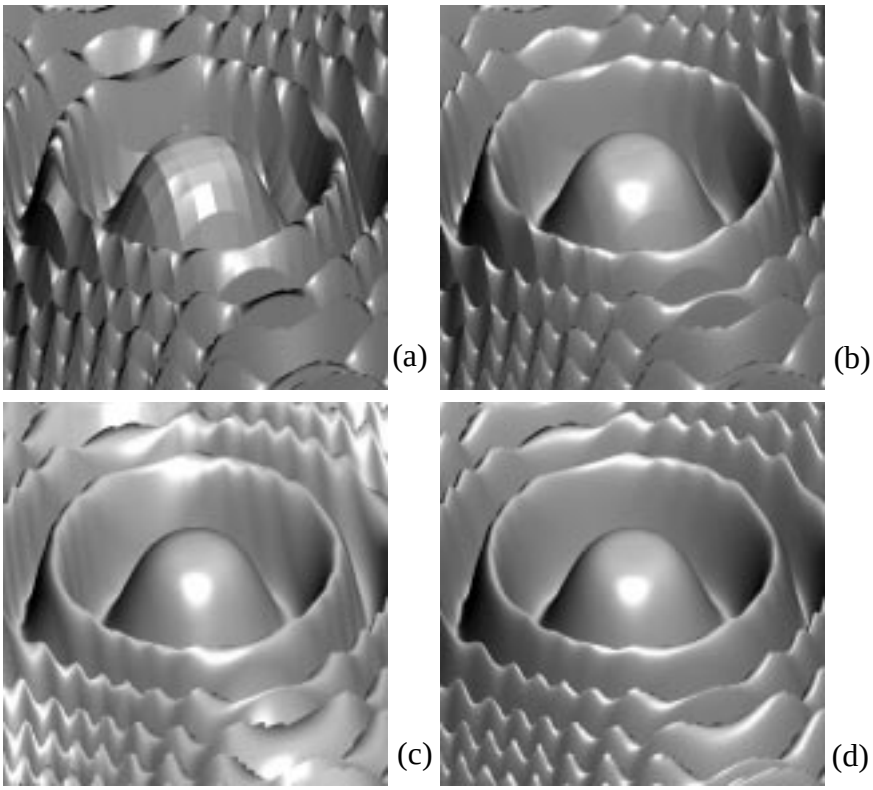


FIGURE 2. Marschner Lobb data set rendered using the following derivative filter (a) discontinuous 1EF (b) discontinuous 3EF (c) C^0-1EF (d) C^0-3EF (e) C^3-4EF

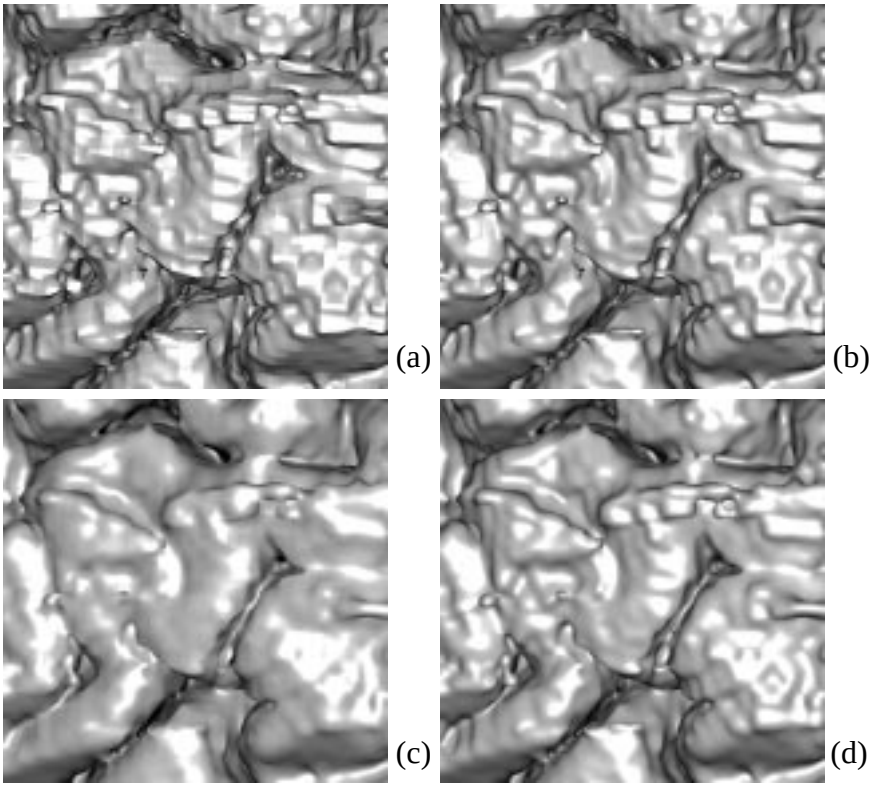


FIGURE 4. MRI data set rendered using the following derivative filter (a) discontinuous 1EF (b) discontinuous 3EF (c) C^0-1EF (d) C^0-3EF

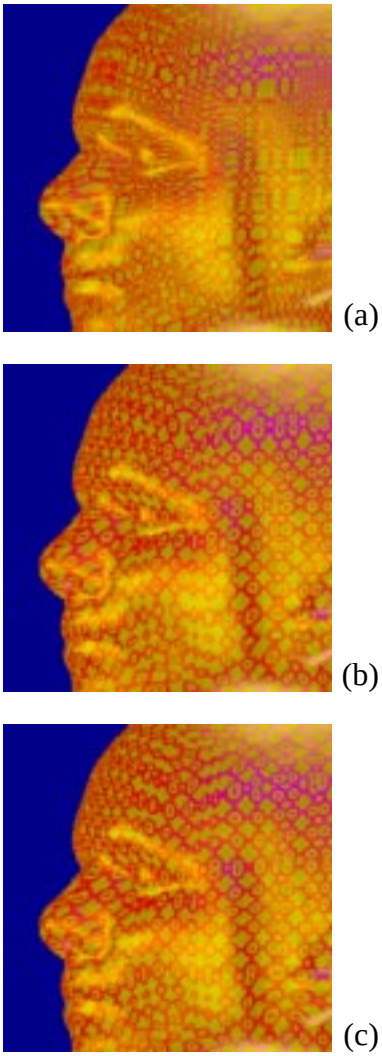


FIGURE 5. Size preserving pattern mapping of a texture on an MRI scan of a human head using (a) a discontinuous 1EF derivative filter (central differences) (b) a C^0-2EF derivative filter (c) a C^1-2EF derivative filter in order to determine the pattern density.

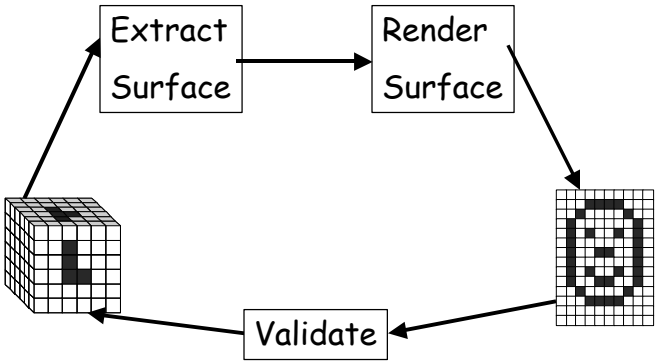
Feature Detection for Volume Rendering

Raghu Machiraju
The Ohio State University

Contributions:
Jinho Lee,Shiva Tenginaki, Ohio State
Joe Marks, Hanspeter Pfister, MERL

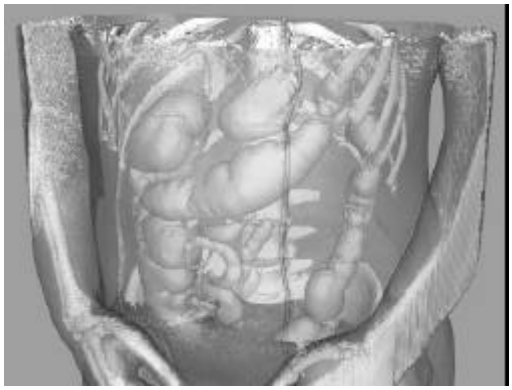
Siggraph 2001

Iso-Surface Detection



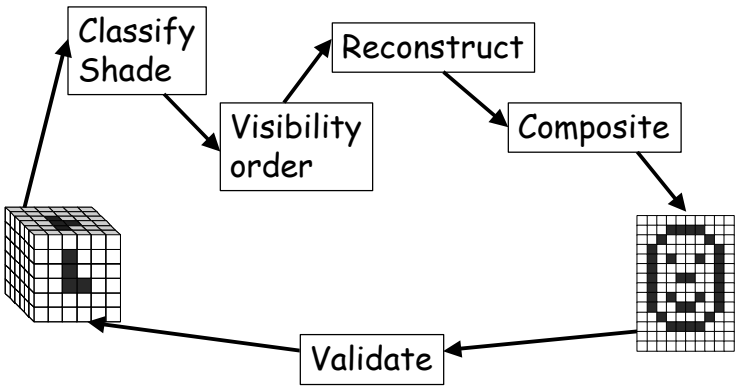
Siggraph 2001

Which Iso-value ?



Siggraph 2001

Direct Rendering Pipeline



Siggraph 2001

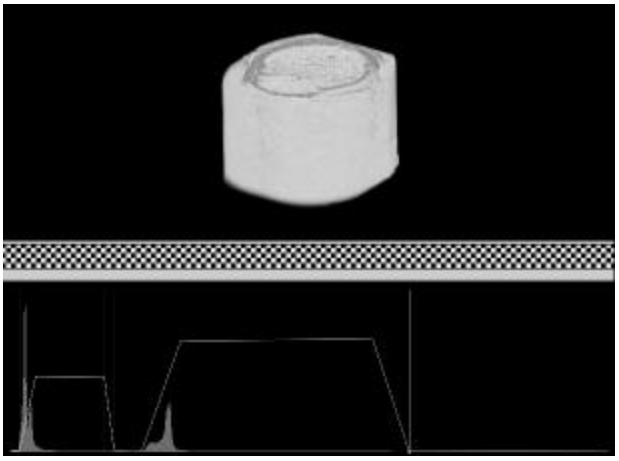
Transfer Functions

- Determine classification/shading step on the rendering pipeline
- Assign visual optical properties to numerical values of dataset
- Typically, color, **opacity**, and emittance
- Opacity transfer function makes features of interest opaque

TFs and Salient Iso-Values

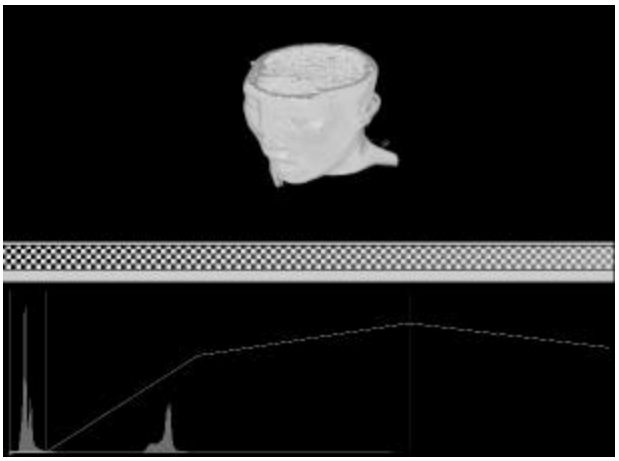
- Key - Determine material interfaces, e.g., air-bone, bone-tissue
- Salient iso-values exist on either side of a boundary
- TF generation and salient iso-value determination are related

Transfer Function-1



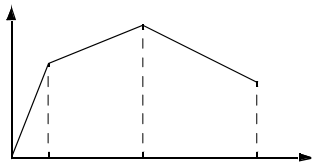
Siggraph 2001

Transfer Function-2



Siggraph 2001

Common Transfer Functions



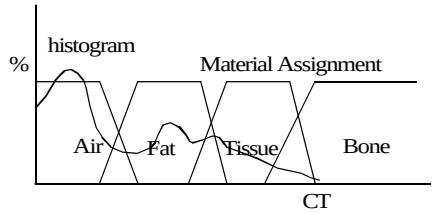
$$\alpha(\mathbf{x_i}) = \left|\nabla f(\mathbf{x_i})\right| \cdot \alpha_{v_{n+1}} \left(\frac{f(\mathbf{x_i}) - f_{v_n}}{f_{v_{n+1}} - f_{v_n}} \right) + \alpha_{v_n} \cdot \left(\frac{f_{v_{n+1}} - f(\mathbf{x_i})}{f_{v_{n+1}} - f_{v_n}} \right)$$

Challenge: Find placement of knots with

- Little aliasing in classified dataset
- Detect structures

Siggraph 2001

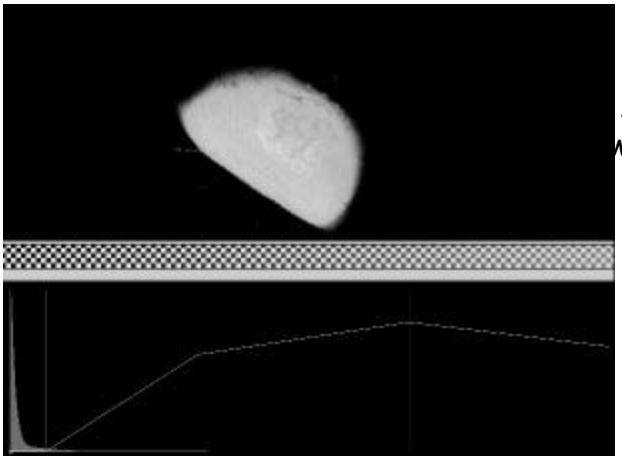
Basic Task



- Segmentation of voxel intensity histogram usually non-linear division and based on presence of features.
- Functional variation within each segment - Usually linear, although non-linear ones can be selected
- CT datasets - OK

Siggraph 2001

Error & Trial - MRI



Cannot always segment histogram with error and trial !

Obstacles

- Various combinations of homogenous materials, although finite, can be large.
- Segmentation of MRI histogram is very difficult given noise and low discrimination.
- Blending effects along the depth direction complicates matters.
- Lack of spatial component makes finding transfer functions frustrating.
- Viewing and shading parameters defy complete control and comprehension of the process.

Desirable Tools

- Should aid user in discovery process
- Assumptions about data minimal
- Flexible and tangible interface

Taxonomy of Methods

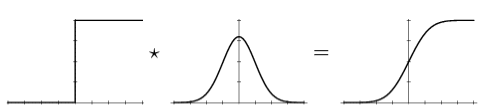
- Trial and error, with minimum computer aid
- Data-centric, using an underlying data model
  - Histogram Volume (Kindlmann and Durkin)
- Data driven, with no underlying assumed model
  - Contour Spectrum (Bajaj et al)
  - Higher Order Moments
- Image-centric, using organized sampling
  - Design Gallery (Marks et al)

Semi-Automatic Method

- Goal: boundaries between homogeneous regions
- Compute Histogram Volume
  - 3D histogram of data value and 1st and 2nd derivatives
  - Project out spatial component of the boundary
- Create Distance Function
  - Signed distance to middle of nearest boundary
  - Bridge unintuitive space of data values to an intuitive, spatial domain

Boundary Model

- Assume to be blurred by Gaussian during measurement process

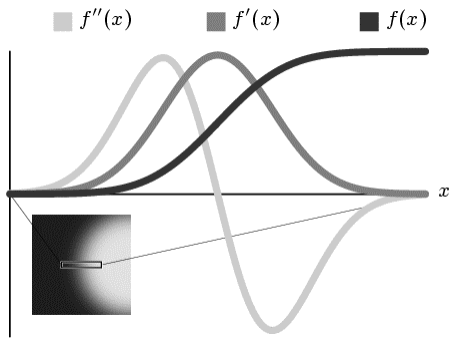


(a) Step function (b) Gaussian (c) Measured boundary prior to sampling

$$v = f(x) = v_{min} + (v_{max} - v_{min})\frac{1+\mathrm{erf}(\frac{x}{\sigma\sqrt{2}})}{2}$$

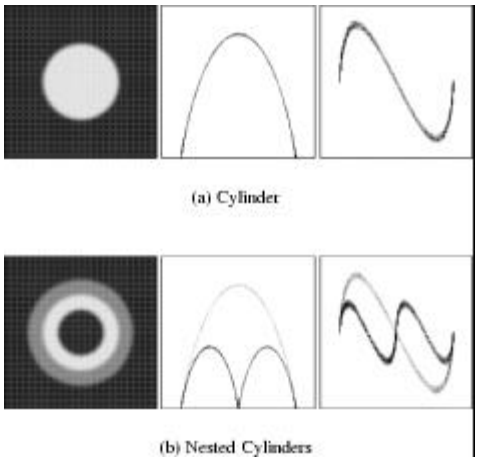
Relationship between f, f , f

Measuring f, f , f across boundary



Siggraph 2001

Plotting Histogram Volume



Number of zero crossings determines boundaries !

Siggraph 2001

TF Generation

- g(v) : average f over all the positions x where f(x) = v
- h(v) : average f over all the positions x where f(x) = v
- g(v) and h(v) obtained from histogram volume
- Recover model specific σ using max(g(v)) and max(h(v))

TF Generation

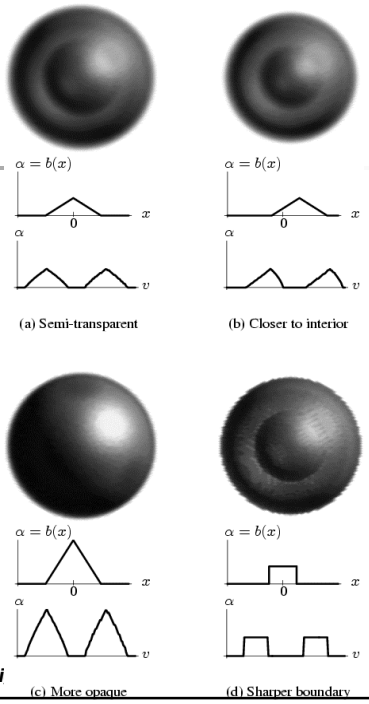
- Define mapping p(v) from value v to a boundary position

$$p(v) = \frac{-\sigma^2 h(v)}{g(v)} \approx \frac{-\sigma^2 f''(f^{-1}(v))}{f'(f^{-1}(v))} = x$$

- Define *boundary emphasis function, b(x)*, to map a boundary position to opacity
- The *final opacity function a(v)* can be defined as

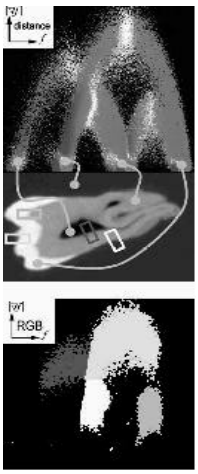
$$\alpha(v) = b(p(v))$$

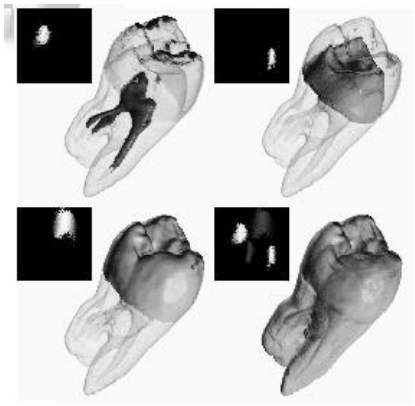
Examples



Si

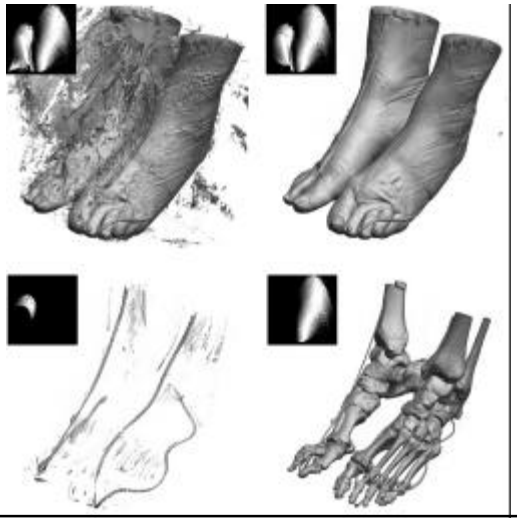
Result - CT Teeth !



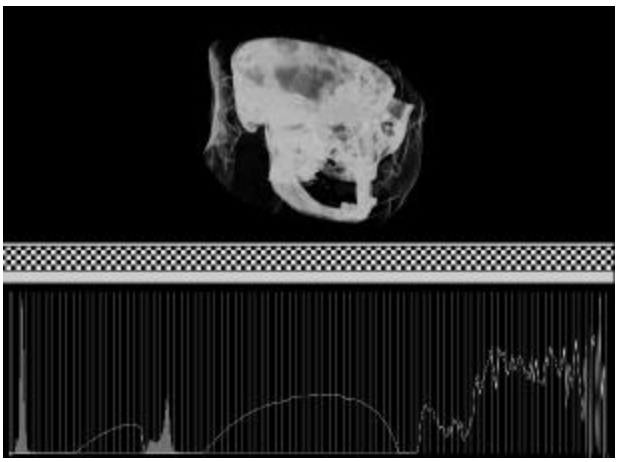


Siggraph 2001

Result - CT Foot !

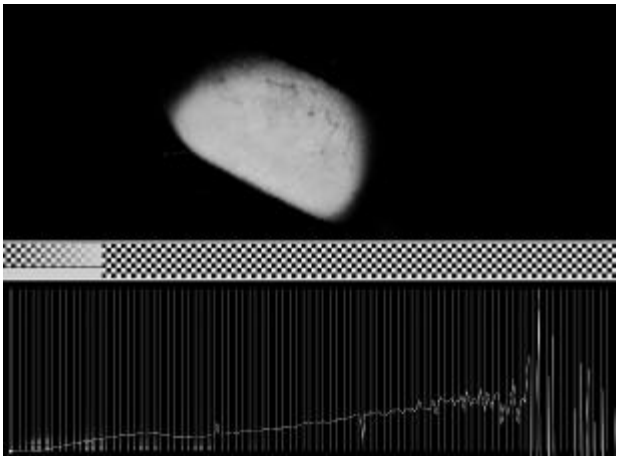


Result MRI Skull !



Siggraph 2001

Result MRI Temple Bone ?



Siggraph 2001

Summary

- Very impressive results
- Intuitive interface
- Strong assumptions about boundary characteristics
- Cannot detect boundaries for some MR datasets

Siggraph 2001

Data-Centric No Model

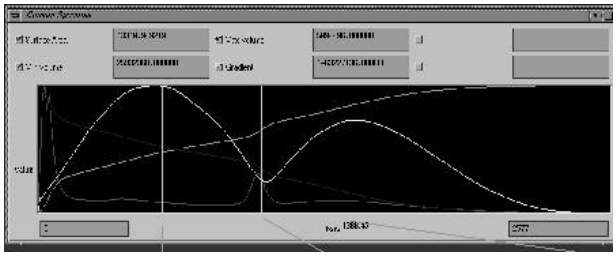
- Determine signatures
- Histogram based
  - Using Factor Analysis on histograms
  - Use localized higher order moments of histogram features
- Using characteristic curves of contours
- Determine salient iso-values !

Contour Spectrum

- 2D Interface: X-axis = scalar value; y-value = normalized signature
- Select ranges for queries
- Signatures:
  - Length
  - Area
  - Gradient integral (length of integral * constant)
- Compute signatures from contours

Contour Signatures

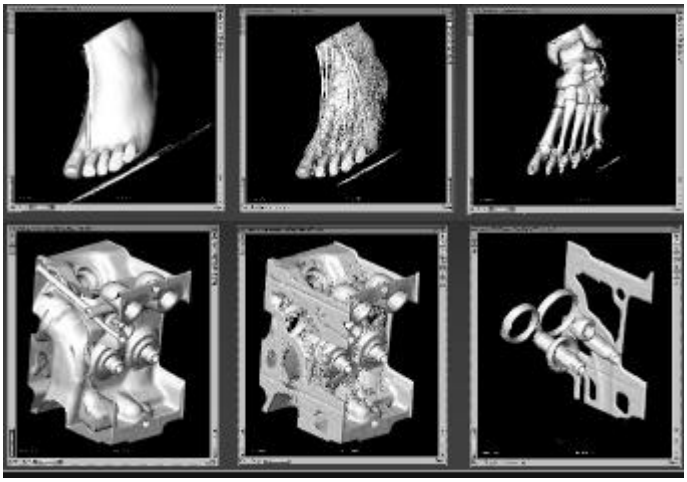
- Length of contour is C^0-spline function
- Area within each contour is a C^1-spline function



Siggraph 2001

Results - CT

Gradient integral needed for boundary detection



Using Statistical Moments

- Histogram equalization uses first moment

$$p_u(x) = \frac{\text{number of voxels with data value } x}{\text{total number of voxels in the dataset}}$$

- Higher Order Central Moments:

$$m_i = \sum_{x=0}^{L-1} (x-m)^i \, p_u(x)$$

- Variance = μ_2

Siggraph 2001

Higher Order Moments

- Skewness = m_3 - Symmetric distributions, $m_3=0$
- Kurtosis = m_4-3 Normal distribution, $m_4=0$
- Infinite number of central moments can completely describe distribution $p_u(x)$
- Using 3D windowing, calculate and assign m_2, m_3, m_4 for each voxel

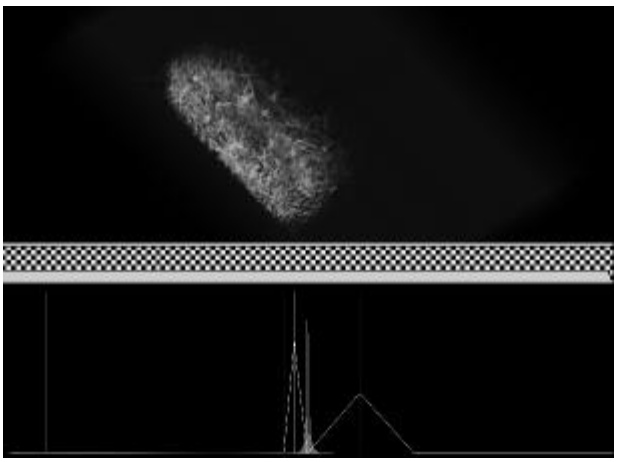
Siggraph 2001

HOMs

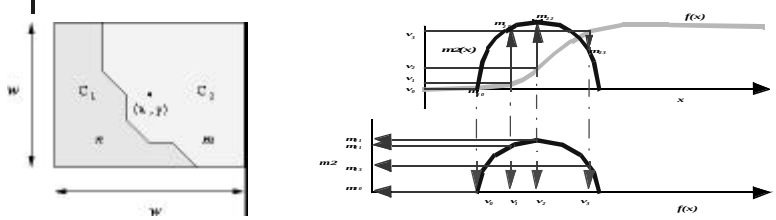
Second Order

Fourth Order

Example Third Order



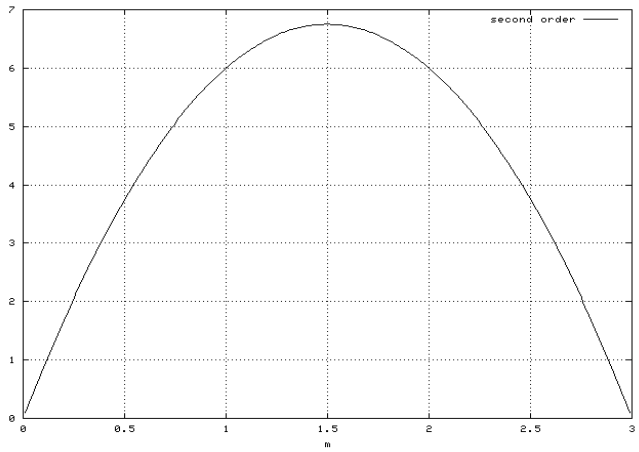
Voxels and HOMs



- 2-mixture material model for small cells
- Boundary exists only if $m, n > 0$
- Determine cummulants and localized HOMs
- Maximas and minimas correspond to material interfaces !

Siggraph 2001

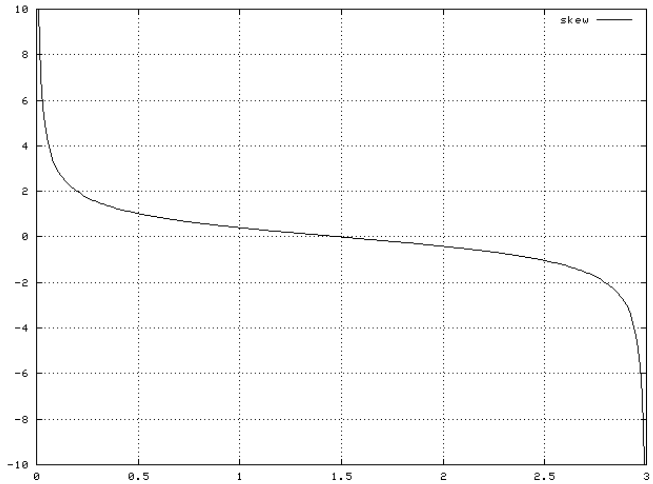
Second Order !



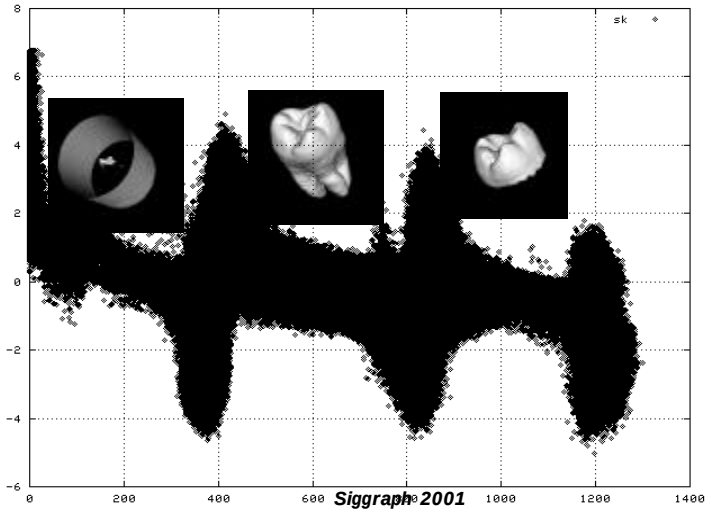
Siggraph 2001

Skewness !

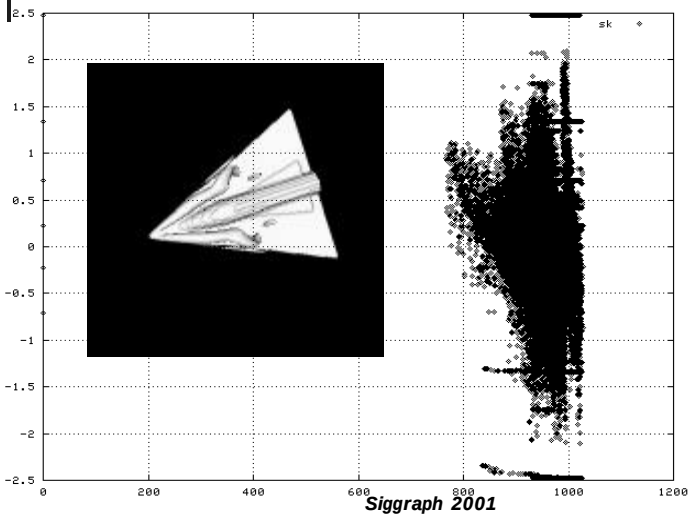
Zero
Crossings !



The Teeth Again !



Another Example !



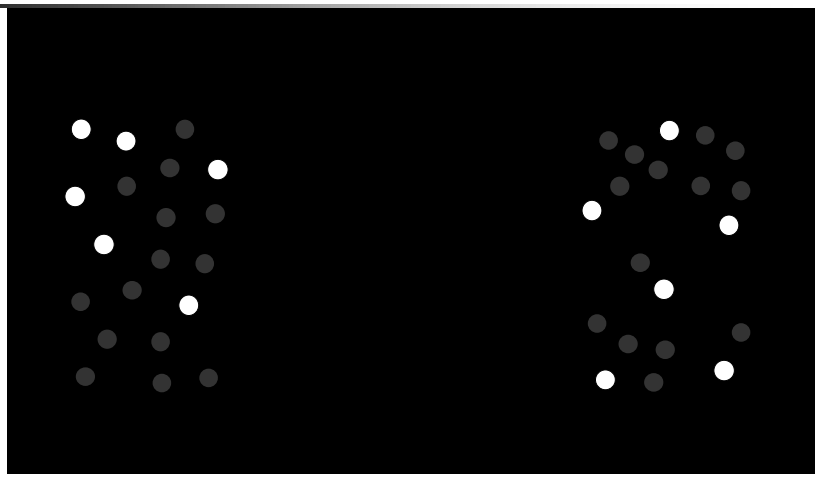
Summary

- Signature-based methods are new
- Have promise
- Much needs to be done to develop statistical and geometrical basis
- User interface needs more thought !

Image Centric Design Gallery

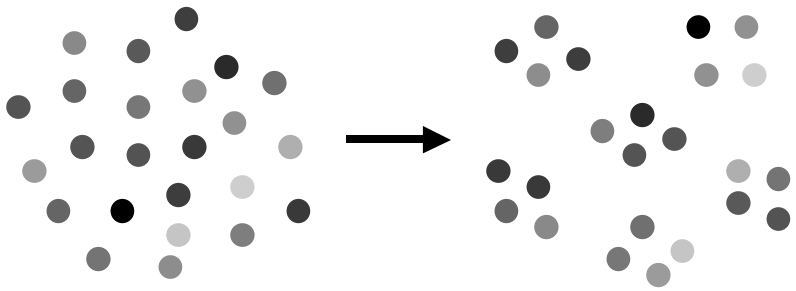
- Sometimes finding an optimal parameters for graphic or animation is problematic
- A different idea:
  - Compute a covering of possible graphics
  - Display them in a gallery using evolutionary computation

Dispersion



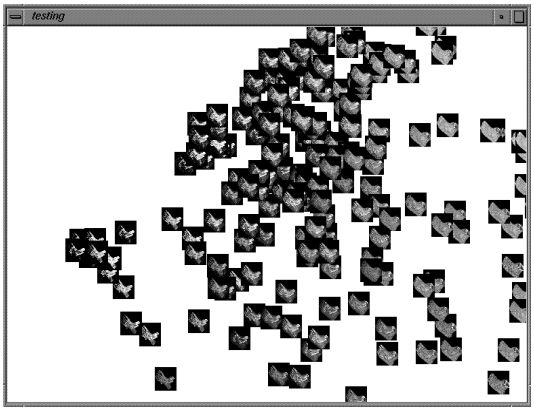
Arrangement

Organize the outputs for easy browsing



Siggraph 2001

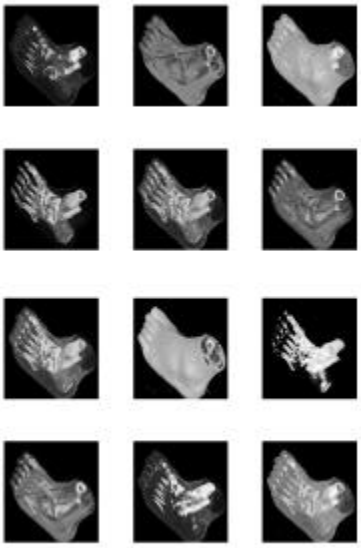
Browsing with volDG



Use MDS !

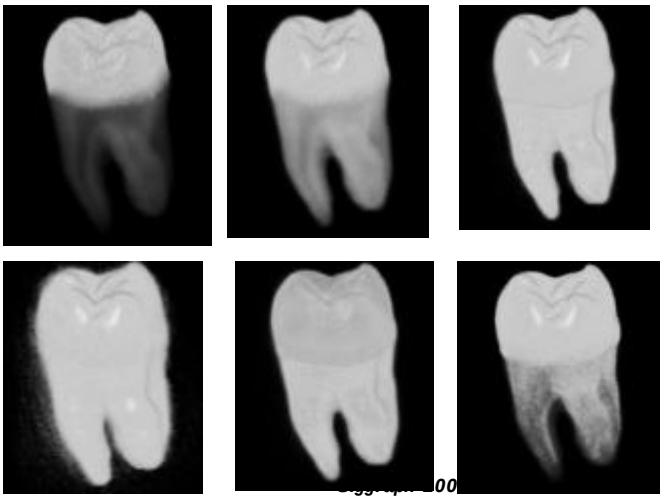
Siggraph 2001

Example



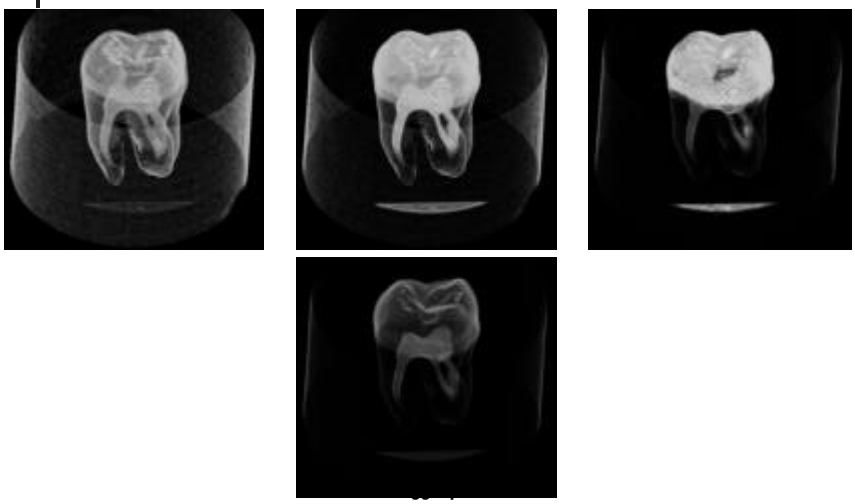
Siggraph 2001

Example



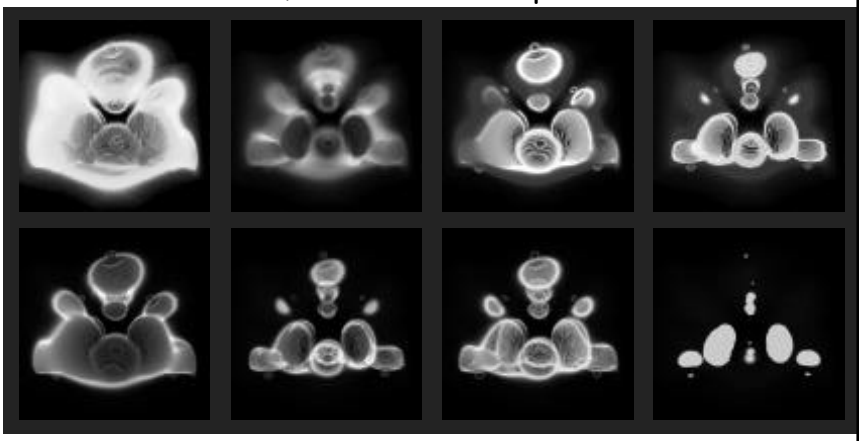
Siggraph 2001

Example HOM volume



Example

Same data, different transparencies



Siggraph 2001

Summary

- Very flexible change volumes, metrics
- Does not include user in generation !
- Not goal-oriented no guided search !
- Lack of customization

Which Approach is Better ?

Too soon to tell !

Top ten problems in volume visualization (VolViz '92).

No general method still exists

Data-centric methods with and without data-models show most promise !

Visualization Viewpoints

Editors: Theresa-Marie Rhyne and Lloyd Treinish

The Transfer Function Bake-Off

Hanspeter Pfister, Bill Lorensen, Chandrajit Bajaj, Gordon Kindlmann, Will Schroeder, Lisa Sobierajski Avila, Ken Martin, Raghu Machiraju, and Jinho Lee

Direct volume rendering is a key technology for visualizing large 3D data sets from scientific or medical applications. Transfer functions are particularly important to the quality of direct volume-rendered images. A transfer function assigns optical properties, such as color and opacity, to original values of the data set being visualized. Unfortunately, finding good transfer functions proves difficult. Pat Hanrahan called it one of the top 10 problems in volume visualization in his inspiring keynote address at the 1992 Symposium on Volume Visualization. And it seems that today, almost a decade later, there are still no good solutions at hand. Or are there?

In a panel discussion at the Visualization 2000 conference, we pitched four of the currently most promising approaches to transfer function design against each other. The four approaches and their advocates are

- trial and error, with minimum computer aid (Will Schroeder);
- data-centric, with no underlying assumed model (Chandrajit Bajaj);
- data-centric, using an underlying data model (Gordon Kindlmann); and
- image-centric, using organized sampling (Hanspeter Pfister).

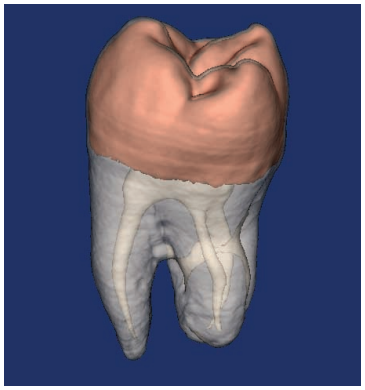
Ahead of time, each of the four panelists received three volume data sets from Bill Lorensen. The data are static 3D scalar volumes sampled on rectilinear grids. The panelists' task was to create meaningful volume renderings using their respective approaches to transfer function design. During the panel session, each panelist presented a summary of the method and results of the visualization, including visual results (images and animations), performance (timings and memory use), and observations (how easy or hard it was, what the findings were, and so on). At the end of the panel session, Bill Lorensen discussed the content of the volume data, what an experienced visualization practitioner would have hoped to find, and how well the panelists' methods achieved this goal. Bill also announced a winner.

This was a unique event: alternative approaches to a pressing research problem went head-to-head, on multiple real-world data sets, and with an objective quality metric (Bill Lorensen). The panel took place in an atmosphere of lighthearted fun, but with a serious goal, namely to emphasize the importance of further research in transfer function design. This article presents the four methods in more detail and answers such questions as: How well did they do? Which method works best? And who won the bake-off?

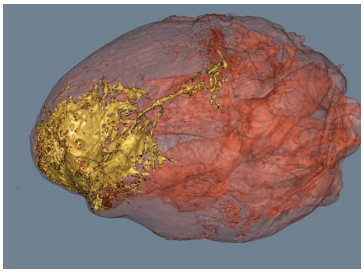
Data sets

Bill Lorensen chose three volume data sets to represent a variety of challenges for the bake-off. Figures 1 through 3 show marching-cube isosurface renderings of the data for comparison to the direct volume rendering images presented by the panelists. The data is available for noncommercial use at http://visual.nlm.nih.gov/.

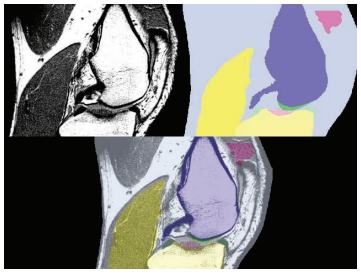
The first data set was generated at GE Aircraft Engines in Evendale, Ohio. The data is industrial x-ray computed tomography (CT) data of a human tooth. The axial



1 The tooth data set modeled with isosurfaces.



2 The sheep heart modeled with two isosurfaces.



3 The segmented MRI knee.

0272-1716/01/$10.00 © 2001 IEEE

slices are ordered from bottom to top, one slice per file. The pixel samples are spaced 1 mm within each slice, and the slices are 1 mm apart. This data set was the easiest of the three to work with. Figure 1 shows the two materials in the tooth extracted as isosurfaces.

The second data set is magnetic resonance image (MRI) data of a sheep heart generated at the Center for In-Vivo Microscopy, Duke University, North Carolina (http://wwwcivm.mc.duke.edu/). These axial slices are ordered from top to bottom, one slice per file. The pixel samples are spaced 1 mm within each slice, and the slices are 1 mm apart. The heart data is a bit more challenging to visualize than the tooth, because the heart has a variety of tissues including some damaged tissue that was caused by blocking circulation to part of the heart. Figure 2 shows the normal and damaged (yellow) tissue in the sheep heart.

The final data set was generated at Brigham and Women's Hospital Surgical Planning Laboratory (http://splweb.bwh.harvard.edu:8000/). The data is clinical MRI data of the knee. These sagittal slices are ordered from left to right. The pixel samples are spaced .25 mm within each slice, and the slices are 1.5 mm apart. This was the most challenging of the three data sets to visualize. Meaningful visualizations of this knee data set are only possible using sophisticated segmentation techniques. Figure 3 shows segmentation performed at Brigham and Women's Hospital, Surgical Planning Lab.

Trial and error

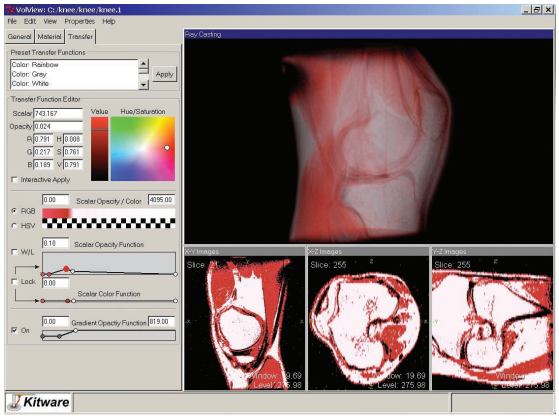
William Schroeder, Lisa Sobierajski Avila, and Ken Martin
Kitware

The widespread use of volume rendering has been hampered by the difficulty of creating effective transfer functions. The complexity of the transfer function is further exacerbated by the blending effects along the depth direction. As a result, recent research has focused on automatic and semiautomatic techniques for creating transfer functions.

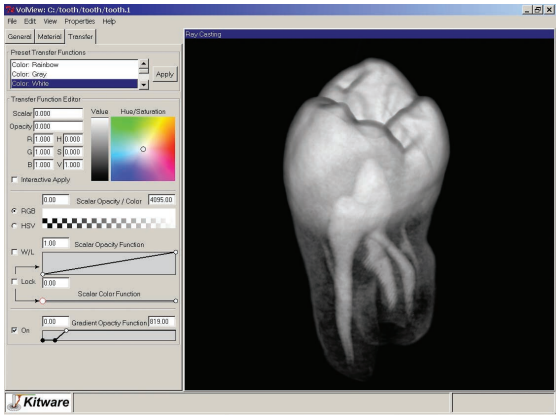
Such methods are potentially dangerous because the techniques remove the human from the visualization process. Visualization isn't just about generating pretty pictures. It's also a vehicle of exploration by which the observer comes to understand the data. You can easily imagine semiautomatic and automatic techniques that generate images that fulfill the observer's expectations, but aren't necessarily true to the nature of the data. Thus, we believe that creating a transfer function is a necessary part of the visualization (that is, data exploration) process. Methods that assist the user in creating transfer functions—and thus improve the efficiency of data exploration—are beneficial. Methods that eliminate the human from the exploration process are dangerous and should be avoided.

Figures 4 through 6 demonstrate these ideas. We used Kitware's VolView volume rendering system and the RTViz VolumePro volume-rendering hardware to generate the images quickly. VolView allows interactive, intuitive creation of transfer functions, while the VolumePro board enables maximum interactive response (up to 30 frames per second). For example, we created both the knee (see Figure 4) and tooth images (see Figure 5) in less than five minutes from start-up to image capture.

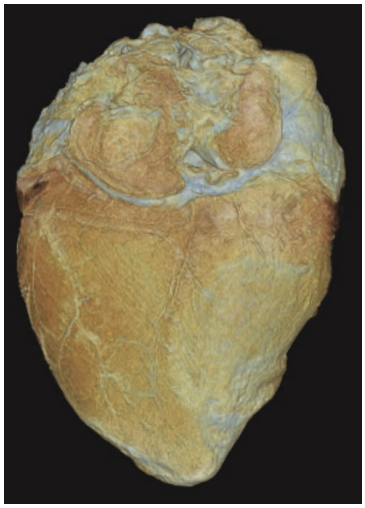
The sheep heart (see Figure 6) was much more challenging to render, requiring approximately 20 minutes to create the final image. However, the time to generate the image was essential: the exploratory process of adjusting the transfer functions taught us much about the data set. For example, after exploring the tooth for five minutes, we felt that we fully understood the important structures within the data. In contrast, the five min-



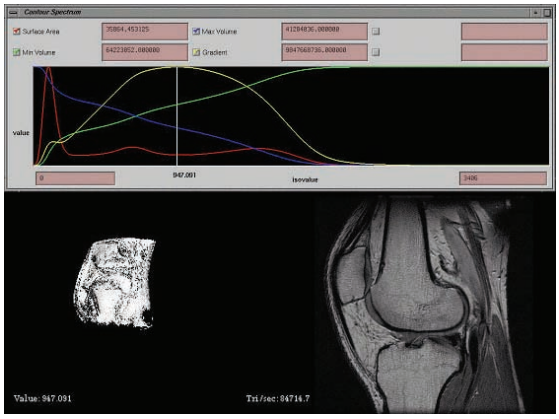
4 Rendering of the knee data set through trial and error.



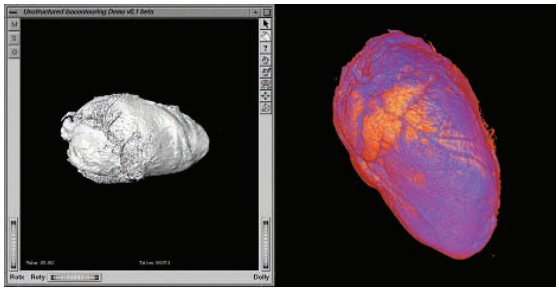
5 Rendering of the tooth data set through trial and error.



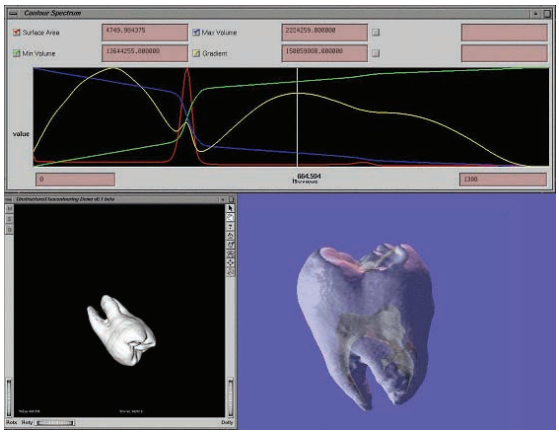
6 Rendering of the sheep data set through trial and error.



7 While the maximum of the gradient integral function (yellow signature curve) determined the average scalar intensity for the knee bone, the multiple peaks in the surface area corresponded to the multiple muscular tissues captured by the imaging data set.



8 The isocontour surface automatically selected by the maximum of the gradient integral signature function (left). A volume rendering of the sheep data set. The data set's primary isocontour value indexes the color map range as it centers on purple (right).



9 The gradient integral function again shows four distinctive peaks for the four material types present in the the tooth data set. The left and right renderings were obtained by determining the intensity values for the peaks from the contour spectrum and using that to assign a white color with no transparency (for the left rendering) and purple, pink, and white colors with varied transparency assignments (for the right rendering).

utes spent visualizing the knee taught us that the data was fairly complex, requiring additional segmentation processing for more effective visualization.

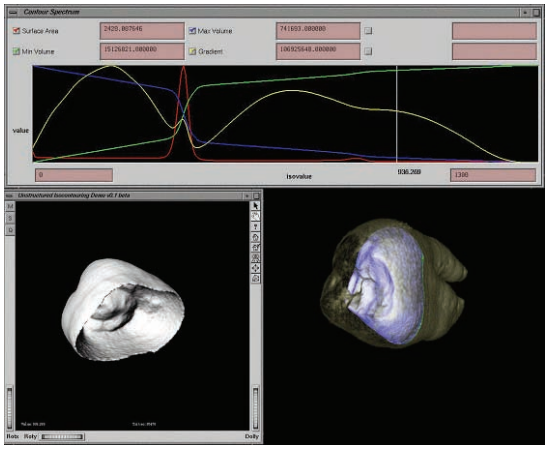
Data-centric, without data model

Chandrajit Bajaj
University of Texas at Austin

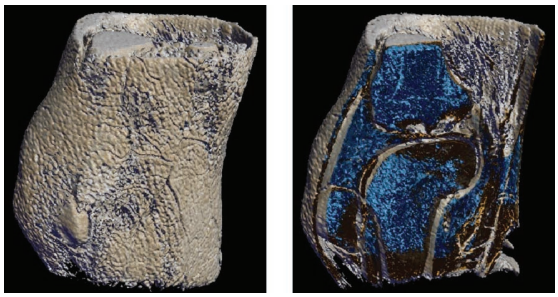
In addition to computational and space complexity issues, user interfaces have a tremendous impact on a visualization environment's level of interactivity.

A contour spectrum consists of computed metrics over a scalar field. On the basis of such metrics you can define a set of functions that provide a useful tool to enhance a data set's interactive query. One primary advantage of the contour spectrum interface is that it lets you display—in a 2D image—a global view of the examined scalar field, independent of its dimension. For example, in a 3D isosurface display, one contour component may be hidden inside another. If you associate the isocontour display with the contour tree, it becomes immediately clear that the current isosurface has two components. Hence, you might need a clipping plane to look inside the current isosurface. For time-varying data, we can compute functional properties over time and display it with a 2D interface. This gives users a global overview of the time-varying function and lets them interact with both the isovalue and time step.

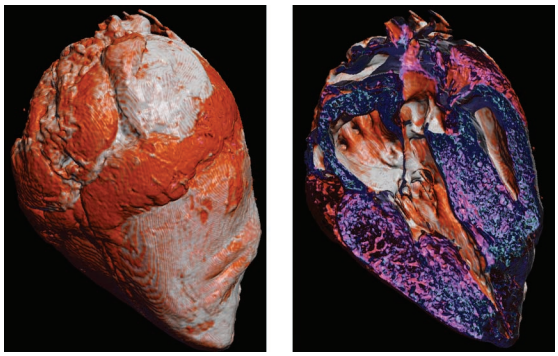
All three of the challenge data sets were imaging data sets (static scalar fields over structured rectilinear meshes). The primary characteristic function I used was the gradient integral function curve (shown in yellow in Figures 7 through 10), which automatically separated the various materials in each of the imaging data and generated the appropriate color and opacity map for the final volume rendering. For details of the signature function computations and the contour spectrum, please see the IEEE Visualization Conference 1997 paper[1] or the Web pages where these tools have been applied to various domains http://www.ticam.utexas.edu/CCV/projects/VisualEyes.



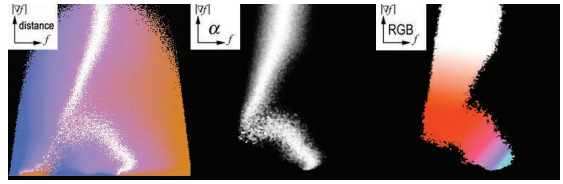
10 Alternate transfer function selection for the tooth data set highlights the inner tooth surface cap of different material types (and higher density) than the outer surface shown in Figure 9.



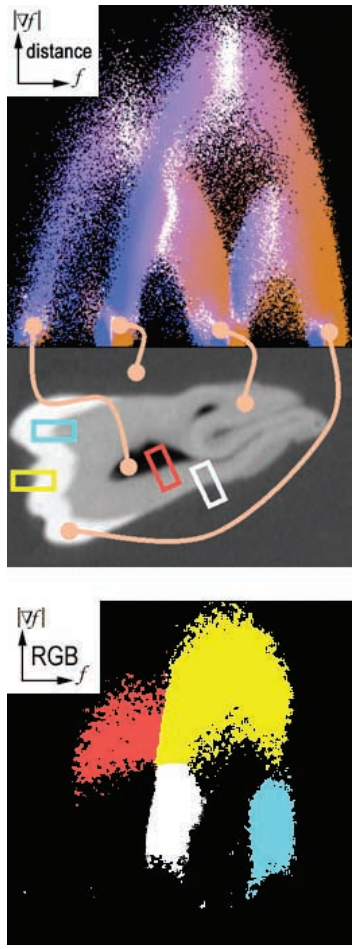
11 Renderings of the knee data set using a semiautomatically generated 1D opacity function.



13 Renderings of the heart data set using the transfer functions in Figure 12. The two segments in the opacity function correspond with two boundaries: one between the heart and the background (left) and one for fine structures within the heart tissue (right).



12 Making a 2D transfer function (from left to right): an automatically generated distance map (white indicates the boundary center), a semiautomatic opacity function, and a manually created color map.



14 Automatically created 2D distance map for the tooth data set. In the cross-section, the colors of the small rectangles (marking the different boundaries) indicate the corresponding boundary colors in the transfer function.

Data-centric, with data model

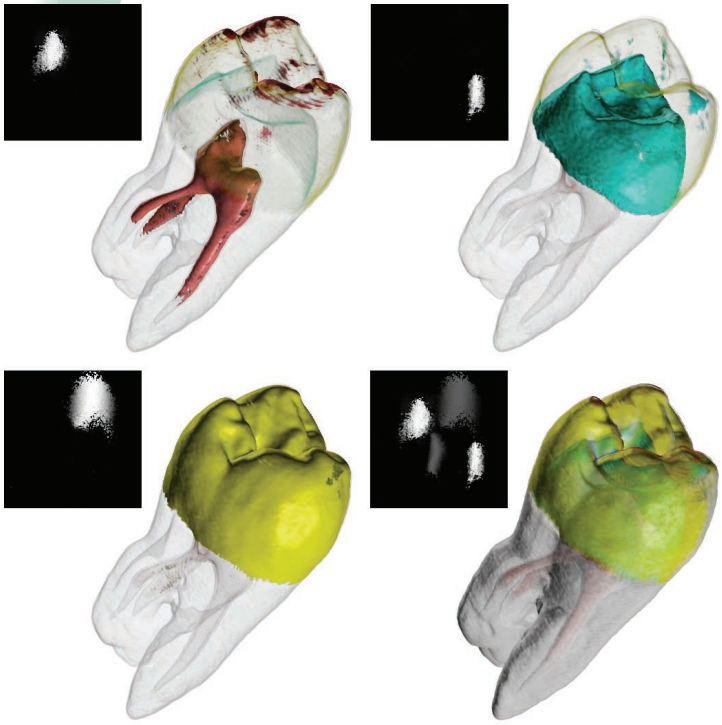
Gordon Kindlmann
University of Utah

For many medical volume data sets, a good transfer function makes opaque only those values consistently associated with a boundary between materials. The semiautomatic method borrows edge detection concepts from computer vision in order to "locate" boundaries in the 1D space of data value (voxel intensity), since that's the domain in which transfer functions are specified. The method starts with a preprocess that takes a few minutes and requires minimal user input: creating a 3D histogram of data value versus first and second derivatives and then distilling this into a *distance map* that records the relationship between data value and boundary proximity. Using the distance map, users can interactively experiment with different settings, but the transfer functions are usefully constrained by the boundary information measured in the given data set. (For more details, see the 1998 IEEE Symposium on Volume Visualization paper[2] or visit http://www.cs.utah.edu/~gk/MS.)

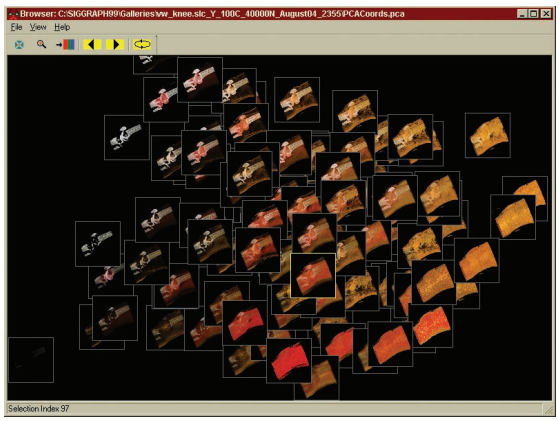
Of course, this method has trouble on data sets in which there are noise and coarse boundary sampling, such as the knee MRI scan. As Figure 11 shows, the method detected the boundary between air and skin and rendered it clearly, but the boundaries among the various internal tissues are less clear.

One benefit of the semiautomatic method is the ability to create transfer functions that map not just data value, but also a 2D space of data value and gradient magnitude. Although they can often disambiguate complex material boundaries, 2D transfer functions are even more difficult to specify by hand than traditional 1D functions. The sheep heart MRI data set is a good example. Figure 12 shows the 2D distance map and transfer function used to make the renderings in Figure 13.

You might think that for a clean data set such as the tooth CT scan, transfer functions based on data value alone could accurately convey all the boundaries. However, the tooth cross-section in Figure 14 shows the data set to have four boundaries between four materials (from dark to bright: pulp, background, dentine, and enamel). The presence of four arcs in the distance map



15 **Renderings of the tooth data set. Opacity functions are inset in each image; surface colors are determined by the color map in Figure 14.**



16 **VolDG user interface.**

(Figure 15) show that the semiautomatic method successfully discerned all the boundaries and thereby facilitated the renderings in Figure 15, which emphasize and color the boundaries in sequence. In particular, with an isosurface or 1D transfer function, it would have been impossible to isolate the dentine-enamel boundary shown in cyan.

The current version of the semiautomatic method assumes a specific mathematical boundary model; this may reduce its effectiveness on some data sets. However, the method ultimately derives its utility from combining a common volume-rendering task ("show me the boundaries") with a characterization of boundaries in terms of easily measured derivatives. A tool such as Design Galleries can answer What's possible? in the space of all transfer functions, while this approach intends to answer What's probable?—that is, What's most likely to be a good transfer function, assuming the goal of visualizing boundaries? Unconstrained exploration of transfer functions is sometimes needed, but interactivity in a visualization tool proves more valuable when the interface itself embodies information and constraints derived from the data in question.

Image-centric, using organized sampling

Hanspeter Pfister
Mitsubishi Electric Research Laboratories
Raghu Machiraju and Jinho Lee
The Ohio State University

Along the lines of the Design Gallery approach,[3] my colleagues and I developed VolDG—Design Galleries for Volume Graphics as a viable alternative to facilitate transfer function selection. The image-centric transfer function design of VolDG focuses on what matters most to the user: the image. VolDG evaluates transfer functions on the basis of the images they produce, not in terms of data set properties. Instead of asking the computer What's best? we ask the computer What's possible? The computer picks a set of input-parameter vectors that span the space of output values as much as possible; the user simply selects from among the presented possibilities.

As Figure 16 shows, VolDG interfaces present the user with the broadest selection—automatically generated and organized—of perceptually different images that can be produced by varying transfer functions.

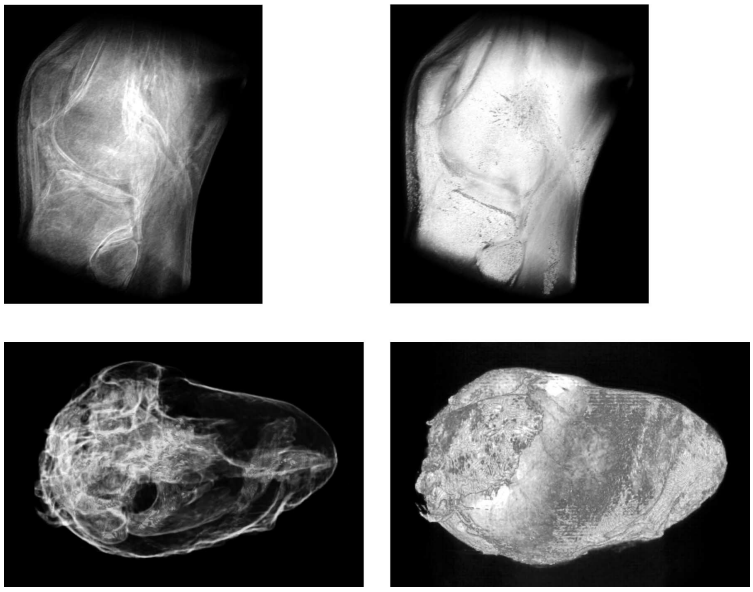
The VolDG approach's principal technical challenges are dispersion (finding a set of input-parameter vectors that optimally generates dissimilar output values) and arrangement (arranging the resulting designs for easy browsing). For dispersion, we use a form of evolutionary computation. For arrangement, we use multidimensional scaling. The dispersion process can require rendering hundreds or thousands of candidate images and therefore benefits greatly from hardware acceleration by Mitsubishi's VolumePro board. In addition, expedient rendering aids the interrogative process between user and computer.

We built our current system on top of the popular Visualization Toolkit (vtk) to using the VolumePro board. The real-time volume-rendering speed of the VolumePro board lets large galleries be generated in minutes. VolDG is freely available at http://www.merl.com/projects/dg/.

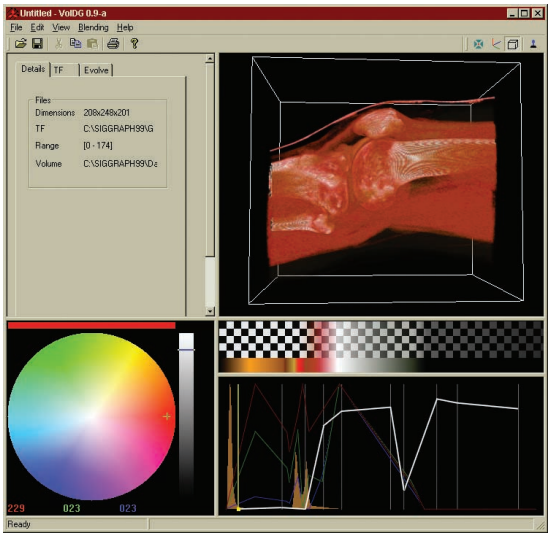
Even though VolDG can manipulate both color and opacity transfer functions, we believe that generating gray-scale images leads to the most insight into unknown data sets. Figure 17 shows two representative images of the knee and sheep heart data set. Note that VolDG automatically detected the interior structure in both images. This is remarkable because MRI data is notoriously difficult to deal with.

Figure 18 shows the transfer function for one MRI knee image. Note that the function is piecewise linear. More sophisticated curve parameterizations, such as B-splines or wavelets, could improve the results of VolDG.

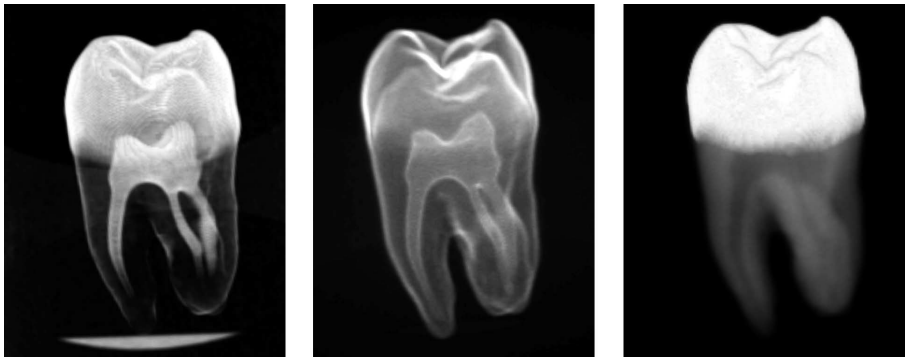
Not surprisingly, the tooth CT scan shows better interior structure and detail (see Figure 19). Each gallery



17 VolDG-generated images of the knee (top) and sheep heart (bottom).



18 Transfer function generated by VolDG.



19 VolDG-generated images of the tooth.

with 200 images and 20,000 iterations took approximately 7 hours to generate. Absolutely no user intervention was necessary. A smaller gallery with 50 images and 1,000 iterations only takes about 3 to 4 minutes. Plus, users can interactively browse the gallery.

Discussion

Bill Lorensen
GE Corporate Research and Development

Since its introduction in the late 1980s, volume visualization has had limited success in cost-effective applications. Advances in image quality and feature sets continue to outpace the technology's acceptance in commercial products.

Several factors contribute to the slow adoption of volume visualization:

- A lack of proven application areas. Routine use of 3D in medicine is still, for the most part, limited to research and teaching hospitals.
- There's no agreement on software application programming interfaces (APIs) for volume visualization. This limitation translates to risk for commercial products that adopt one vendor's API over another's.
- Volume visualization is slow, requiring expensive workstations with large amounts of memory and special graphics hardware extensions.
- The volume techniques are difficult to use by all but an experienced engineer or scientist.

Fortunately, various universities and companies are addressing these limitations:

- New scanners are presenting much more information than a radiologist can possibly review on a slice-by-slice basis. Three-dimensional visualization could be the key to increasing productivity.
- APIs are emerging that fit within current graphics and visualization systems.
- Low-cost, special-purpose hardware is now available for personal computers. And, the general-purpose processor speeds continue to improve. Texture-mapping hardware is available on cheap graphics cards. In addition, 3D texture mapping could also get cheap if the gamers find a use for it.

However, ease of use is still an issue. Volume visualization has the potential to significantly reduce the amount of time to segment medical data. We need fast, robust techniques to create color and opacity transfer

functions before volume rendering can move from the lab to the hospital. It doesn't matter whether the techniques are automatic, semiautomatic, or manual. They just need to be fast and simple.

Summary and conclusions

Bill Lorensen
GE Corporate Research and Development

The four groups on the panel weren't given a specific task to perform. Their sole goal was to produce images that would impress the panel judge (me). All the groups gave a short presentation of their results. Each team performed admirably, presenting a variety of renderings for all the data sets.

The Kitware panelist produced "artistic" renderings of each data set by manually choosing transfer functions. The MERL approach presented dozens of alternatives for each data set, requiring the user to choose an appropriate rendering. The Texas algorithm automatically created transfer functions based on metrics derived from the data. The Utah panelist also presented an automatic technique that followed a more traditional feature-extraction approach, designed to find boundaries in the data.

As the judge, I was biased against techniques that required too much or too little human interaction. This bias eliminated the manual Kitware approach and the automatic MERL technique. I had a difficult time deciding the winner between the two remaining datacentric approaches. The Texas reliance on observable metrics in the data seems to be more intuitive than the Utah approach. However, in my opinion, the Utah algorithm shows the most promise and is most likely to stimulate future research in the area of automatic transfer function synthesis.

Rectilinear, static grids are the simplest volumetric data, yet it's obvious that many problems exist in developing effective renderings. Other types of volumetric data are even more challenging, such as time-varying data, time-varying grids, irregular grids, scattered data, or nonscalar fields. We hope that this panel encourages further research in transfer function design, particularly for more complex, difficult-to-visualize volumetric data sets. ■

Acknowledgments

We'd like to thank the National Library of Medicine, National Institutes of Health, and Visible Human Project for hosting the data distribution Web site (http://visual.nlm.nih.gov/).

Also, we'd like to acknowledge that Vamsidhar Juvvigunta, Joe Marks, and Kathy Ryall were the main collaborators on VolDG.

References

1. C. Bajaj, V. Pascucci, and D. Schikore, "The Contour Spectrum," *Proc. 1997 IEEE Visualization Conf.*, IEEE CS Press, Los Alamitos, Calif., Oct. 1997, pp. 167-173.
2. G. Kindlman and J. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering," *Proc. 1998 IEEE Symp. Volume Visualization*, IEEE CS Press, Los Alamitos, Calif., Oct. 1998, pp. 79-86.
3. J. Marks et al., "Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation," *Proc. Siggraph 97,* ACM Press, New York, Aug. 1997, pp. 389-400.

Readers may contact Pfister at MERL, 201 Broadway, Cambridge, MA 02139, email pfister@merl.com.

Contact department editors Rhyne and Treinish by email at rhyne@siggraph.org and lloydt@us.ibm.com.

Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering

Gordon Kindlmann * and James W. Durkin †

Program of Computer Graphics
Cornell University

Abstract

Although direct volume rendering is a powerful tool for visualizing complex structures within volume data, the size and complexity of the parameter space controlling the rendering process makes generating an informative rendering challenging. In particular, the specification of the transfer function — the mapping from data values to renderable optical properties — is frequently a time-consuming and unintuitive task. Ideally, the data being visualized should itself suggest an appropriate transfer function that brings out the features of interest without obscuring them with elements of little importance. We demonstrate that this is possible for a large class of scalar volume data, namely that where the regions of interest are the boundaries between different materials. A transfer function which makes boundaries readily visible can be generated from the relationship between three quantities: the data value and its first and second directional derivatives along the gradient direction. A data structure we term the *histogram volume* captures the relationship between these quantities throughout the volume in a position independent, computationally efficient fashion. We describe the theoretical importance of the quantities measured by the histogram volume, the implementation issues in its calculation, and a method for semi-automatic transfer function generation through its analysis. We conclude with results of the method on both idealized synthetic data as well as real world datasets.

1 Introduction

1.1 The Task of Finding Transfer Functions

Transfer functions make a volume dataset visible by assigning renderable optical properties to the numerical values which comprise the dataset. The most general transfer functions are those that assign opacity, color, and emittance [12]. Useful renderings can often be obtained, however, from transfer functions which assign just opacity, with the color and brightness derived from simulated lights which illuminate the volume according to some shading model. We use the term *opacity functions* to refer to this limited subset of transfer functions. During the rendering process, the sampled and interpolated data values are passed through the opacity function to determine their contribution to the final image. Since the opacity function does not normally take into account the position of the region being rendered, the role of the opacity function is to make opaque those data values which consistently correspond, across the whole volume, to features of interest. This paper addresses only the problem of setting opacity functions, as this is a non-trivial yet manageable problem whose solution is pertinent to more general transfer function specification issues.

Finding a good transfer function is critical to producing an informative rendering, but even if the only variable which needs to be set is opacity, it is a difficult task. Looking through slices of the volume dataset allows one to spatially locate features of interest, and a means of reading off data values from a user-specified point on the slice can help in setting an opacity function to highlight those features, but there is no way to know how representative of the whole feature, in three dimensions, these individually sampled values are. User interfaces for opacity function specification typically allow the user to alter the opacity function by directly editing its graph, usually as a series of linear ramps joining adjustable control points. This interface does not itself guide the user towards a useful setting, as the movement of the control points is unconstrained and unrelated to the underlying data. Thus finding a good opacity function tends to be a slow and frustrating trial and error process, with seemingly minor changes in an opacity function leading to drastic changes in the rendered image. This is made more confusing by the interaction of other rendering parameters such as shading, lighting, and viewing angle.

1.2 Direct Volume Rendering of Boundaries

A significant assumption made in this paper is that the features of interest in the scalar volume are the boundary regions between areas of relatively homogeneous material[1]. For instance, this is often true of datasets from medical imaging. But if the goal is to render the boundaries of objects, why use direct volume rendering, and not isosurface rendering? Although this question itself deserves investigation, it is widely accepted that direct volume rendering avoids the binary classification inherent in isosurface rendering — either the isosurface passes through a voxel or not [11]. To the extent that an object's surface is associated with a range of values, an opacity function can make a range of values opaque or translucent. This becomes especially useful when noise or measurement artifacts upset the correlation between data value and material type.

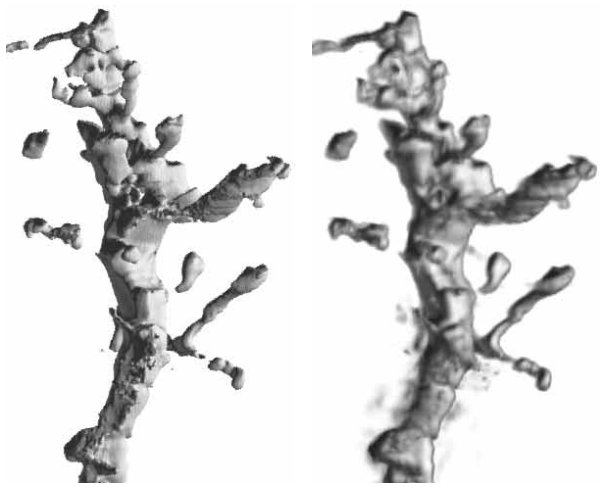
As a quick illustration of this, consider a dataset generated from limited angle tomography [6], where there are often streaks and blurriness in the data caused by the unavailability of projections at some range of angles. This type of data is studied in the Collaboratory for Microscopic Digital Anatomy[19], an ongoing project aimed at providing remote, networked access to sophisticated microscopy resources. Fig. 1 shows two renderings of a mammalian neuron dataset, using the same viewing angle, shading, and lighting parameters, but rendered with different algorithms: a non-polygonal ray-cast isosurface rendering and a shear-warp direct volume rendering produced with the Stanford VolPack rendering library [10]. Towards the bottom of the direct volume rendered image, there is some fogginess surrounding the surface, and the surface itself is not very clear. As can be confirmed by looking directly at slices of the data itself, this corresponds exactly to a region of the

* Address: Department of Computer Science, University of Utah, Salt Lake City, UT 84102. Email: gk@cs.utah.edu.

† Address: Program of Computer Graphics, Cornell University, Ithaca, NY 14853. Email: jwd@graphics.cornell.edu.

[1] We use *boundary* to refer, not to an infinitesimally thin seperating surface between two areas of disparate data value, but to the thin region wherein the data value transitions from one material value to the other.

dataset where the material boundary is in fact poorly defined. The surface rendering, however, shows as distinct a surface here as everywhere else, and in this case the poor surface definition in the data is manifested as a region of rough texture. This can be misleading, as there is no way to know from this rendering alone that the rough texture is due to measurement artifacts, and not a feature on the dendrite itself.



(a) Isosurface Rendering (b) Direct Volume Rendering

Figure 1: Two renderings of a spiny dendrite from a cortical pyramidal neuron. The volume dataset was reconstructed from images of a 2 micron thick section acquired with an intermediate high voltage electron microscope at the National Center for Microscopy and Imaging Research, San Diego, California, using single-tilt axis tomography. Specimen kindly provided by Prof. K. Hama of the National Institute for Physiological Sciences, Okazaki, Japan.

2 Related Work

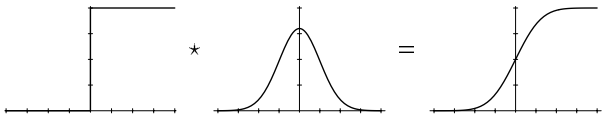
Two methods have been proposed for assisting the user in the exploration of possible transfer functions. He et al. [7] use genetic algorithms to breed a good transfer function for a given dataset. Judging from small thumbnail renderings, the user picks desirable transfer functions from an automatically generated population, until the iterative process of image selection and transfer function intercombination converges. Alternatively, the system can run automatically by using some user-specified objective function (entropy, energy, or variance) to evaluate rendered images. Marks et al. [13] address the problem of "parameter tweaking" in general, with applications including light placement for rendering, motion control for articulated figure animation, as well as transfer functions in direct volume rendering. The goal is to create a visual interface to the complex parameter space by using an image difference metric to arrange renderings from a wide variety of transfer functions into a "design gallery", from which the user selects the most appealing rendering. While both of these methods reportedly succeed in finding useful transfer functions, and while they both allow the user to inspect the transfer function behind a rendering, the systems are fundamentally designed for finding good renderings, not for finding good transfer functions. Both processes are entirely driven by analysis of rendered images, and not of the dataset itself. Rather than having an high-level interface to *control* the transfer function, the user has to *choose* a transfer function from among those randomly generated, making it hard to gain insight into what makes a transfer function appropriate for a given dataset.

Other visualization tools have been described which are more driven by the data itself. Bergman et al. [3] describe a perceptually informed rule-based method for colormap selection which takes into account the data's spatial frequency characteristics and the purpose of the visualization. Closer to the goal of the current paper is the contour spectrum, described by Bajaj et al. [1], which helps the user find isovalues for effective isosurface volume visualizations of unstructured triangular meshes. By exploiting the mathematical properties of the mesh, important measures of an isosurface such as surface area and mean gradient magnitude can be computed with great efficiency, and the results of these measurements are integrated into the same interface which is used to set the isovalue. By providing a compact visual representation of the metrics evaluated over the range of possible isovalues, the user can readily decide, based on their rendering goals, which isolevel to use. The importance to the current paper is that the contour spectrum is a good example of how an interface can use measured properties of the data to guide the user through the parameter space controlling the rendering.

3 Ideal Boundary Characterization

3.1 Boundary Model

Since our particular goal is the visualization of material boundaries, we have chosen a model for what constitutes an ideal boundary and developed methods around that. We assume that at their boundary, objects have a sharp, discontinuous change in the physical property measured by the values in the dataset, but that the measurement process is band-limited with a Gaussian frequency response, causing measured boundaries to be blurred by a Gaussian. Fig. 2 shows a step function representing an ideal boundary prior to measurement, the Gaussian which performs the band-limiting by blurring, and the resulting measured boundary (prior to sampling). The resulting curve happens to be the integral of a Gaussian, which is called the *error function* [9]. Actual measurement devices band-limit, so they always blur boundaries somewhat, though their frequency response is never exactly a Gaussian, since this has infinite support. Although certain mathematical properties of the Gaussian are exploited later, we have not found the inexact match of real-world sampling to the Gaussian ideal to limit application of our technique. A final assumption made for the purposes of this analysis is that the blurring is isotropic, that is, uniform in all directions. Again, our methods will often work even if a given dataset does not have this characteristic, but results may be improved if it is preprocessed to approximate isotropic blurring.



(a) Step function (b) Gaussian (c) Measured boundary prior to sampling

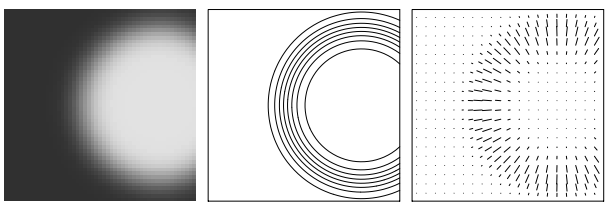
Figure 2: Boundaries are step functions blurred by a Gaussian.

3.2 Directional Derivatives along the Gradient

Although it was suggested in Section 1.2 that isosurfaces are not always sufficient for visualizing objects in real world volume data, the method presented in this paper still indirectly employs them as an indicator of object shape. That is, based on the mathematical property that the gradient vector at some position always points

perpendicular to an isosurface through that position, we use the gradient vector as a way of finding the direction which passes perpendicularly through the object boundary. Even though isosurfaces do not always conform to the local shape of the underlying object, if we average over the whole volume, the gradient vector does tend to point perpendicular to the object boundary. We rely on the statistical properties of the histogram to provide the overall picture of the boundary characteristics.

The directional derivative of a scalar field f along a vector $\mathbf{v}$, denoted $\mathbf{D}_{\mathbf{v}} f$, is the derivative of f as one moves along a straight path in the $\mathbf{v}$ direction. This paper studies f and its derivatives as one cuts directly through the object boundary — moving along the gradient direction — in order to create an opacity function. Because the direction along which we are computing the directional derivative is always that of the gradient, we employ a mild abuse of notation, using f' and f'' to signify the first and second directional derivative along the gradient direction, even though these would be more properly denoted by $\mathbf{D}_{\widehat{\nabla f}} f$ and $\mathbf{D}^2_{\widehat{\nabla f}} f$, where $\widehat{\nabla f}$ is the gradient direction. We treat f as if it were a function of just one variable, keeping in mind that the axis along which we analyze f always follows $\widehat{\nabla f}$, which constantly changes orientation depending on position. Fig. 3 shows how the gradient direction changes with position to stay normal to the isosurfaces of a simple object.



(a) $f(\mathbf{x})$ (b) Isosurfaces of f (c) ∇f

Figure 3: ∇f is always normal to f's isosurfaces.

Fig. 4 analyzes one segment of the cross-section of this same object. Shown are plots of the data value and the first and second derivatives as one moves across the boundary. Because of band-limiting, the measured boundary is spread over a range of positions, but an exact location for the boundary can be defined with either the maximum in f', or the zero-crossing in f''. Indeed, two edge detectors common in computer vision, Canny [4] and Marr-Hildreth [14], use the f' and f'' criteria, respectively, to find edges.

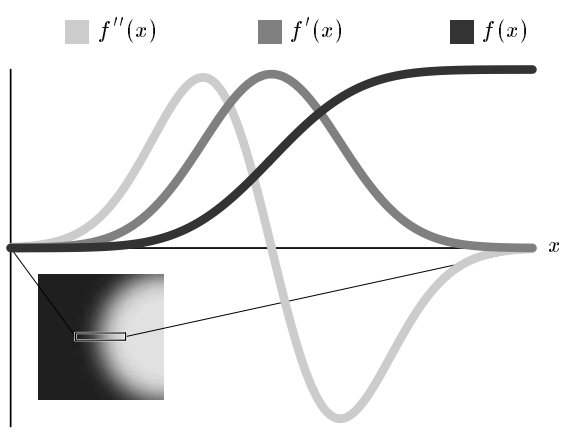


Figure 4: Measuring f, f', and f'' across boundary.

3.3 Relationship Between f, f', and f''

As our goal is to find functions of data value which highlight boundary regions, our problem is rather different than that addressed by edge detectors. Because the opacity function will be applied throughout the volume irrespective of position, we must locate the boundary not in the spatial domain, but in the range of data values. In contrast, edge detectors locate boundaries in the spatial domain. Yet, we still want to borrow from computer vision the notion that boundaries are somehow associated with a maximum in f' and/or a zero-crossing in f''. To see how this is possible, consider just the relationship between f and f'. As both of these are functions of position, they can be plotted with a three-dimensional graph, as in Fig. 5. The three-dimensional curve can be projected downward to form the plot of data value versus position, and projected to the right to show first derivative versus position. Projecting the curve along the position axis, however, eliminates the position information, and reveals the relationship between data value and first derivative. Because the data value increases monotonically, there is a (non-linear) one-to-one relationship between position and data value, so the first derivative f', which had been a function of *position* x, can also be expressed as a function of *data value* f. This is what the third projection in Fig. 5 depicts.

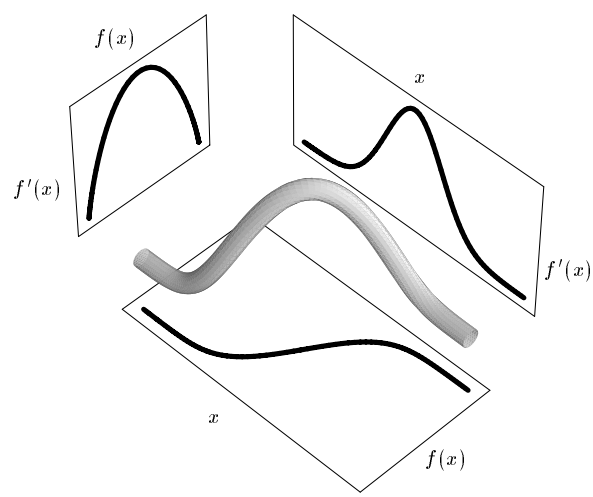


Figure 5: f, f' and position x.

The same projections can be done for data value and its second derivative, as seen in Fig. 6. Projecting the curve downward or to the right produces the graphs of data value or second derivative versus position (first seen in Fig. 4), while projecting along the position axis reveals the relationship between data value and its second derivative.

Finally, having "projected out" position information, one can make a three-dimensional graph of the first and second derivatives as functions of *data value*, as seen in Fig. 7. The significance of this curve is that it provides a basis for automatically generating opacity functions. If a three dimensional record of the relationship between f, f' and f'' for a given dataset contains curves of the type shown in Fig. 7, we can assume that they are manifestations of boundaries in the volume. With a tool to detect those curves and their position, one could generate an opacity function which makes the data values corresponding to the middle of the boundary (indicated with cross-hairs in Fig. 7) the most opaque, and the resulting rendering should show the detected boundaries. Short of that, one could use a measure which responds to some specific feature of the curve (say, the zero crossing in f'') and base an opacity function on that. This is what the current paper seeks to do.

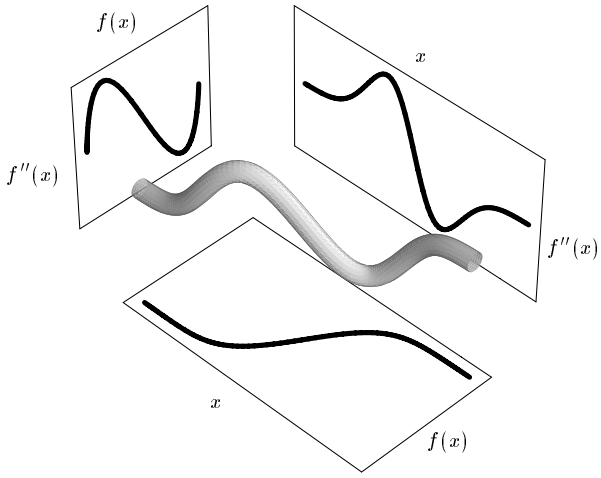


Figure 6: f, f'' and position x.

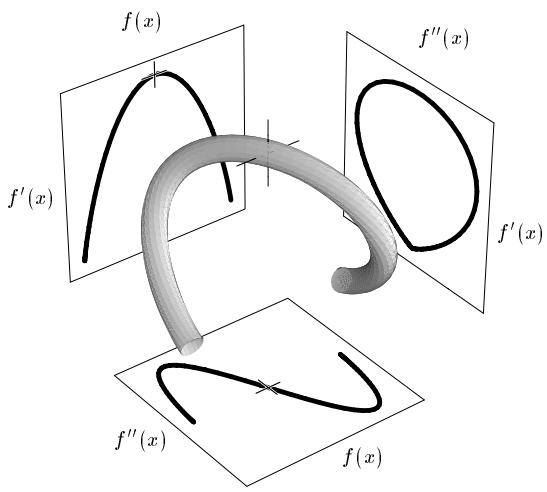


Figure 7: The underlying relationship of f, f', and f''.

4 The Histogram Volume

4.1 Histogram Volume Structure

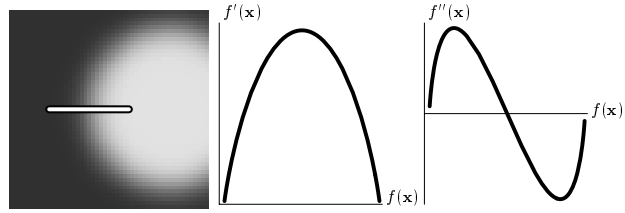
To measure the relationship between the data value and its derivatives described in the last section, we use a three-dimensional histogram we term a *histogram volume*. There is one axis for each of the three quantities f, f', and f'', and each axis is divided into some number of (one-dimensional) bins, causing the interior volume to be divided into a three-dimensional array of bins. The histogram volume has two defining characteristics:

1. Each bin in the histogram volume represents the combination of a small range of values in each of the three variables f, f', and f''.

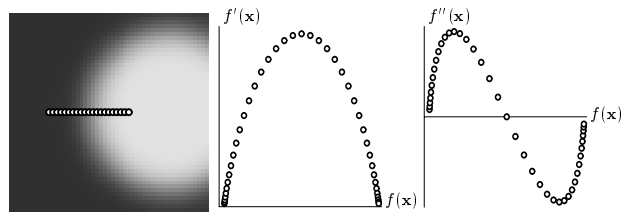
2. The value stored in each bin signifies the number of voxels in the original volume within that same combination of ranges of these three variables.

4.2 Histogram Volume Creation

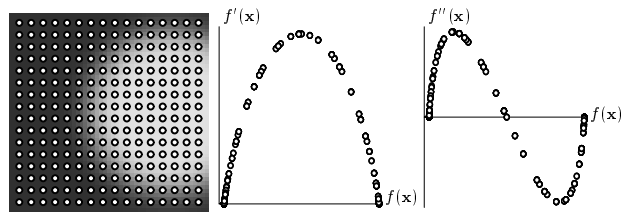
Fig. 7 illustrated the position-independent relationship between f, f', and f'' that characterized an ideal boundary. To find that relationship, however, we afforded ourselves the luxury of first knowing *where* the boundary was in the idealized dataset. For instance, Fig. 4 was produced with knowledge of where to place a path so as to cross through the boundary. In the case of real volume data, however, the positions of the boundaries are not known, but the same relationship between f, f' and f'' needs to be revealed by some measurement technique.



(a) Continuous linear sampling



(b) Dense linear sampling



(c) Sampling at gridpoints

Figure 8: Sampling the boundary: from continuous to discrete.

It is sufficient to measure f, f', and f'' at each point of a uniform lattice. Fig. 8(a) shows a boundary being sampled continuously to produce smooth graphs of f' and f'' versus f. In Fig. 8(b), the sampling is along the same path, but is now discrete. The smooth graphs have been replaced by scatterplots, but the sequence of measurements traces out the same curves as before, indicating that discrete sampling and the resulting scatterplots are sufficient to illuminate the important relationships between f and its derivatives. Finally, in Fig. 8(c), the boundary is sampled everywhere on a uniform grid. Though now the points are distributed differently — many more hits have accumulated along where f' and f'' are near zero — the scatterplots trace out the save curves as before. By sampling everywhere, we no longer require knowledge of boundary location, and the "global" derivative characteristics of the boundary have been measured. This is precisely the sort of information relevant to opacity function generation.

The approach taken in this paper is to measure f and its directional derivatives exactly once per voxel, at the original sample points of the dataset. One might be concerned that sampling merely at the original data points is not a sufficient sampling density to produce the curves seen in Figs. 7 and 8. However, with real volume data this will not be a problem, since the band-limiting in data acquisition assures there will always be some blurring, and since the boundaries of real objects tend to assume a variety of positions and orientations relative to the sampling grid.

4.3 Implementation

One implementation issue in the histogram volume creation is how many bins to use. There is a trade-off between storage and processing requirements versus having sufficient resolution in the histogram volume to discern the patterns from which we generate the opacity functions. In our experiments, good results were obtained with histogram volumes of sizes between 80^3 and 256^3 bins, though there is no reason that the histogram volumes need to have equal resolution on each axis. Also, we have found it sufficient to use only 8 bits to represent the values in the histogram volume, scaling and/or clipping the number of hits to the range 0–255 if necessary.

A more subtle issue is what range of values to include along each axis. Obviously, for the data value axis, the full range should be included, since we intend to capture all the values at which boundaries might occur. But along the axes for first and second derivative, it makes sense to include something less than the full range. Since derivative measures are by nature sensitive to noise, including the full range of derivative values in the histogram volume may cause the important and meaningful sub-range of values to be compressed to a smaller number of bins, thereby hampering the later step of detecting patterns in the histogram volume. We do not have an *a priori* knowledge of the meaningful ranges of derivatives values, so currently the derivative value ranges are set with an educated guess. This is a matter in need of further research and automation.

The most significant implementation issue is the method of measuring the first and second directional derivatives. The first derivative is actually just the gradient magnitude. From vector calculus [15] we have:

$$\mathbf{D}_{\mathbf{v}} f = \nabla f \cdot \mathbf{v}, \tag{1}$$

thus

$$\mathbf{D}_{\widehat{\nabla f}} f = \nabla f \cdot \widehat{\nabla f} = \nabla f \cdot \frac{\nabla f}{\|\nabla f\|} = \|\nabla f\|. \tag{2}$$

Unfortunately there is no similarly compact formula for $\mathbf{D}^2_{\widehat{\nabla f}} f$, the second directional derivative along the gradient direction. Twice applying Eqn. 1 gives:

$$\begin{aligned} \mathbf{D}^2_{\widehat{\nabla f}} f &= \mathbf{D}_{\widehat{\nabla f}}(\|\nabla f\|) = \nabla(\|\nabla f\|) \cdot \widehat{\nabla f} \\ &= \frac{1}{\|\nabla f\|} \nabla(\|\nabla f\|) \cdot \nabla f \end{aligned} \tag{3}$$

Or, using the Taylor expansion of f along $\widehat{\nabla f}$ [5] gives:

$$\mathbf{D}^2_{\widehat{\nabla f}} f = \frac{1}{\|\nabla f\|^2} (\nabla f)^{\mathrm{T}} \, \mathbf{H} f \, \nabla f \tag{4}$$

where $\mathbf{H} f$ is the Hessian of f, a 3×3 matrix of second partial derivatives of f [15]. Alternatively, we can use the Laplacian $\nabla^2 f$ to approximate $\mathbf{D}^2_{\widehat{\nabla f}} f$:

$$\mathbf{D}^2_{\widehat{\nabla f}} f \approx \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{5}$$

The approximation is exact only where the isosurfaces have zero mean surface curvature [5].

These three expressions for $\mathbf{D}^2_{\widehat{\nabla f}} f$ each suggest different implementations for the second derivative measure. Although this would benefit from more detailed study, we can still make useful observations regarding the comparative merits of each. While we have found the Hessian method to be the most numerically accurate, the others have proven sufficiently accurate in practice to make them appealing by virtue of their computation efficiency. The Laplacian computation is direct and inexpensive, but the most sensitive to quantization noise. The gradient of the gradient magnitude (Eqn. 3) is better, and its computational expense is lessened if the gradient magnitude has already been computed everywhere for the sake of volume rendering (e.g., as part of shading calculations).

By measuring the derivatives only at the original data points, the calculation of the first and second partial derivatives required in the above expressions is greatly facilitated by the use of discrete convolution masks applied at the data points; we have used standard central differences. Thus, our task is somewhat distinct from the usual problem of derivative measurement in volume rendering, where a primary concern is continuity of the derivative between sample points to allow for correct shading of interpolated data values [2, 16].

The general algorithm for creating the histogram is straightforward:

1. Initialize the histogram volume to all zeroes.

2. Make one pass through the volume looking for the highest values of f' and f'', and the lowest value of f''; assume zero for the lowest value of f'. Set ranges on the histogram volume axes accordingly.

3. On a second pass through the volume,

   3a. Measure f, f', and f'' at each voxel,

   3b. Determine which bin in the histogram volume corresponds to the measured combination of f, f', and f'', and

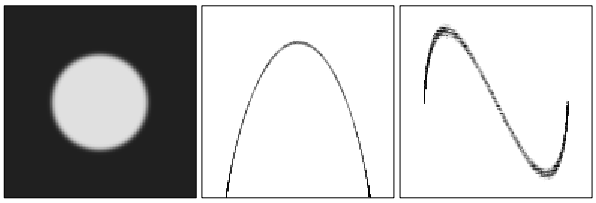
   3c. Increment the bin's value.

4.4 Histogram Volume Inspection

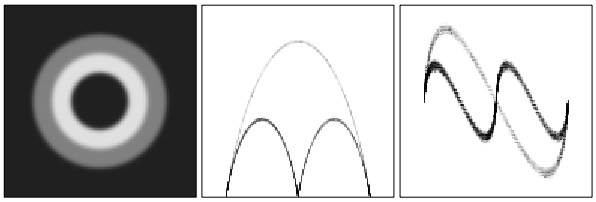
It is possible to gain some insight into the object boundaries of the original dataset by simple visualization of the histogram volume after it has been calculated. One may be tempted to simply volume render the histogram volume from arbitrary views, but this usually turns out to be unrevealing due to the speckled and noisy nature of the histogram volume. A better way is to use summed-voxel of projections the histogram volume, projecting along either the f' or the f'' axis, to produce scatterplots of f'' versus f or f' versus f, respectively. This allows testing of the premise in Section 3.3 — if there are boundaries in the original dataset that conform to the boundary model, there should be curves like that of Fig. 7 in the histogram volume.

Fig. 9 shows cross-sections and scatterplots for two synthetic datasets. The curves in the scatterplots are exactly the form seen in Fig. 8, and one can see an important property of the histogram volume — for each pair of materials that share a boundary, there is a curve in the histogram volume. This property is again visible in Fig. 10, wherein various computed tomography datasets are being analyzed. Though the scatterplots are noisier, it clear that the histogram volume is successfully capturing information about the materials and their boundaries.

It should be noted that a related technique has been used in computer vision for feature identification. Panda and Rosenfeld [18] use two-dimensional scatterplots of data value and gradient magnitude to perform image thresholding for night vision applications. They, however, do not assume a boundary model, instead limiting their analysis of the scatterplot to identifying particular distributions within regions of low and/or high gradient magnitude.



(a) Cylinder



(b) Nested Cylinders

Figure 9: Dataset Slice, f' versus f, and f'' versus f.

5 Opacity Function Generation

5.1 Mathematical Boundary Analysis

In order to develop a method for opacity function generation that uses our boundary model and the information stored in the histogram volume, it is helpful to look at the equation we have used to describe the ideal boundary data value as a function of position:

$$v = f(x) = v_{min} + (v_{max} - v_{min})\frac{1 + \mathrm{erf}(\frac{x}{\sigma\sqrt{2}})}{2} \tag{6}$$

Just as we have taken "position" to always be along an axis pointing in the gradient direction, we define zero to always be the position of the inflection point in the boundary. v_{min} and v_{max} are the data values of the materials on either side of the boundary. As $\mathrm{erf}()$ ranges from -1 to 1, v ranges from v_{min} to v_{max}. The parameter controlling the amount of boundary blurring is σ. The first and second derivatives of f are as follows:

$$f'(x) = \frac{v_{max} - v_{min}}{\sigma\sqrt{2\pi}} \exp(-\frac{x^2}{2\sigma^2}) \tag{7}$$

$$f''(x) = -\frac{x(v_{max} - v_{min})}{\sigma^3\sqrt{2\pi}} \exp(-\frac{x^2}{2\sigma^2}) \tag{8}$$

Our choice of boundary parameterization means that $f'(x)$ is a Gaussian, with σ being the usual standard deviation. Since the Gaussian has inflection points at $\pm\sigma$, this is where $f''(x)$ attains its extrema. The same positions can serve as artificial delimiters for the extent of the boundary — we *define* the "thickness" of the boundary to be 2σ. Note that the thickness of a boundary can be recovered if the maximum values of f' and f'' are known:

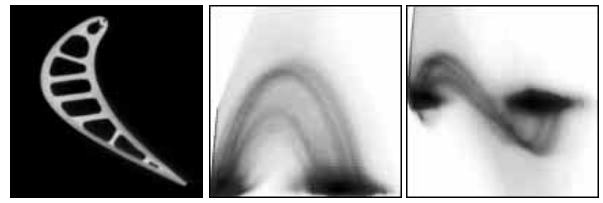
$$\frac{f'(0)}{f''(-\sigma)} = \sigma\sqrt{e} \tag{9}$$

More importantly, once σ is known, we can recover the position x knowing only the values of f' and f'':

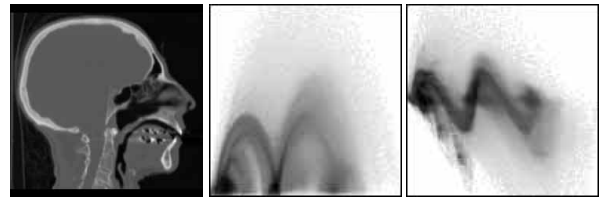
$$\frac{f''(x)}{f'(x)} = -\frac{x}{\sigma^2} \tag{10}$$

5.2 Opacity functions of data value

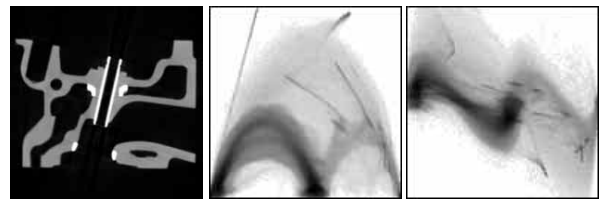
Before using Eqn. 10 as the basis for opacity function generation, we define some important functions of data value: $g(v)$ is the average first directional derivative of f over all the positions $\mathbf{x}$ at



(a) Turbine Blade



(b) Head



(c) Engine Block

Figure 10: Dataset Slice, f' versus f, and f'' versus f.

which $f(\mathbf{x}) = v$, and $h(v)$ is likewise the average second directional derivative at value v. These two functions can be obtained from the histogram volume by slicing it at data value v, and finding the centroid of the scatterplot of f' and f'' at that value. The f' axis coordinate of the centroid is $g(v)$, and the f'' axis coordinate is $h(v)$.

Knowing $g(v)$ and $h(v)$ for all v, one can find the ratio of their maxima to recover σ with Eqn. 9, assuming that g attains its maxima at $f(0)$, and that h attains its maxima at $f(-\sigma)$. With this information, we define a mapping $p(v)$ from data value to an approximate position along a boundary:

$$\begin{aligned} p(v) &= \frac{-\sigma^2 h(v)}{g(v)} \\ &\approx \frac{-\sigma^2 f''(f^{-1}(v))}{f'(f^{-1}(v))} = x \end{aligned} \tag{11}$$

Roughly speaking, $p(v)$ tells us on which side of the nearest boundary a data value v tends to fall. For values closer to v_{min}, $p(v)$ will be negative; for values closer to v_{max}, $p(v)$ will be positive. In practice, we have found it useful to modify Eqn. 11 to account for the fact that the gradient magnitude at the interior of materials is rarely exactly zero. Knowing how it differs from zero is again a matter of experience, but assuming one can find a g_{thresh} which approximates the ambient gradient magnitude, Eqn. 11 is reformulated, with a slight loss of mathematical accuracy, as

$$p(v) = \frac{-\sigma^2 h(v)}{\max(g(v) - g_{thresh}, 0)} \tag{12}$$

The user supplies the last piece of information needed: a function $b(x)$ we term the *boundary emphasis function*, which maps from position along a boundary to opacity. As the intent is to make only boundaries visible in the rendering, $b(x)$ should be non-zero only near zero. For this reason, we have not been especially careful to prevent $p(v)$ from attaining infinite values due to a low $g(v)$;

such a data value v should not contribute to the final image. With $b(x)$, the user can directly control whether rendered boundaries will appear thick or thin, sharp or fuzzy, and the proximity of the rendered boundary to the object interior. The final opacity function $\alpha(v)$ is then defined as

$$\alpha(v) = b(p(v)) \tag{13}$$

Instead of exploring the parameter space of all possible opacity functions, the user explores the parameter space of $b(x)$ and lets the information from the histogram volume, embodied in $p(v)$, constrain the search to those opacity functions which display object boundaries. Defining opacity as a function of position within a boundary then becomes a more intuitive task than defining opacity as a function of data value, as there is a more predictable relationship between changes made to the boundary emphasis function and the corresponding change in rendered results. As there is a potential for inaccuracy in the calculation of σ from Eqn. 9, the user may need to experiment with different scalings in the domain of $b(x)$. Fig. 11 shows how the choice of boundary emphasis function affects the opacity function and the rendered image, for a synthetically created dataset containing two concentric spheres at distinct data values. It should be stressed that the user does not set the location of the peaks in $\alpha(v)$, since this is determined by the information in $p(v)$, but the user can influence the location, as well as the width, height, and shape of the peaks. This is the main benefit of this method: if the histogram volume has successfully captured information about the boundaries in the dataset, the user enjoys high-level control over the character of the rendered boundaries without having to worry about the exact specification of $\alpha(v)$. Yet, the $\alpha(v)$ so generated is sensible enough that it could be edited by hand if desired. For example, since this technique will attempt to make *all* boundaries opaque, a useful supplement to the interface would be a feature which allows supression of the peaks in $\alpha(v)$ for one or more boundaries.

Even though we have made some strong assumptions about the boundary characteristics in the volume dataset, the technique described here typically works well even if the material boundaries are not "ideal". Essentially, by taking the ratio of the second and first derivatives, and by having $b(x)$ assign opacity to positions around zero, we are more apt to make opaque those data values associated with both low second derivatives and high first derivatives. Or, even if $p(v)$ is not a perfect indicator of "position relative to boundary", the sign change in f'' around its zero-crossing affords us some control over whether we want to emphasize regions closer to or further from the object's interior. Fig. 12 on the accompanying colorplate shows a rendering of an MRI dataset which does not have ideal boundaries but for which this technique still works.

5.3 Opacity functions of data value and gradient magnitude

So far the opacity functions under consideration have assigned opacity based on data value alone. Higher quality renderings can sometimes be obtained, however, if the opacity is assigned as a function of both data value and gradient magnitude. Defining these two-dimensional opacity functions by hand is especially challenging because there are even more degrees of freedom than in one-dimensional, value-based opacity functions. Fortunately, the ideas presented so far easily generalize to allow semi-automatic generation of two-dimensional opacity functions.

Analogous to the definition of $h(v)$, we define $h(v, g)$ to be average second derivative over all locations where the data value is v and the gradient magnitude is g; this is also easily calculated from the histogram volume. We similarly define a new position function,

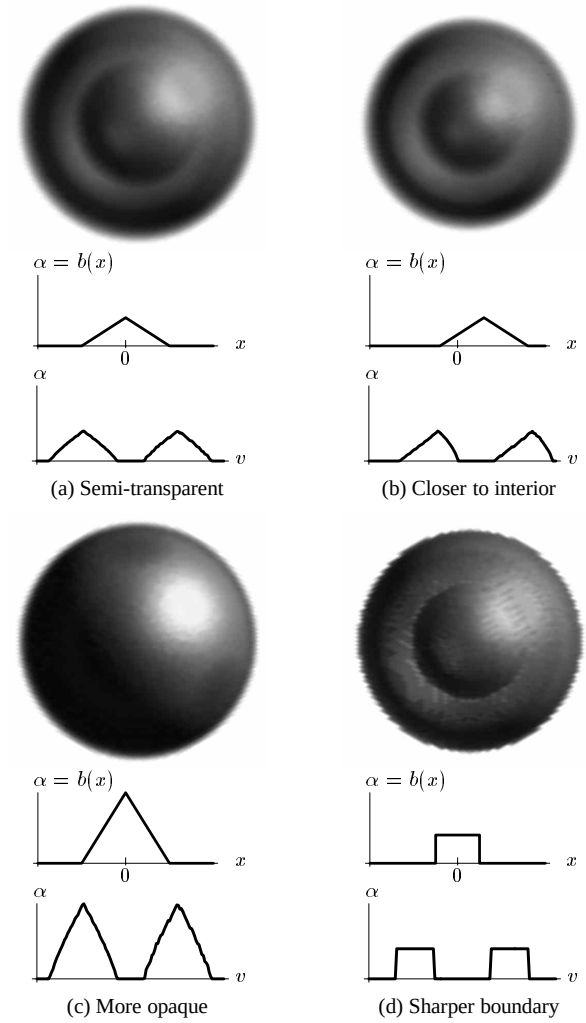


Figure 11: Relationship between $b(x)$, $\alpha(v)$, and the rendered result.

and from that an opacity function:

$$p(v, g) = \frac{-\sigma^2 h(v, g)}{\max(g - g_{thresh}, 0)} \tag{14}$$

$$\alpha(v, g) = b(p(v, g)) \tag{15}$$

σ is calculated as before; the ratio of the extremum of the average first and second derivatives. The benefit of this kind of opacity function is that it can distinguish between boundaries that have overlapping ranges of values. For instance, the nested cylinders volume in Fig. 9 and the engine block volume in Fig. 10 each have one boundary which overlaps the two other boundaries in data value, spanning from the higher of the two material values to the background value. Selectively rendering this single boundary is impossible with a value-based opacity function, but because the boundary has a distinct curve in the plot of data value versus first derivative, it is possible to create an opacity function which selects only the voxels comprising this boundary. As it did in the case of value-based opacity functions, the technique presented here will generate two-dimensional opacity functions which make all detected boundaries opaque; a simple "lasso" tool could then be used to select different regions in the two-dimensional opacity function to render one boundary at a time. In Fig. 13 on the colorplate, the feet of the

female Visible Human CT dataset [17] are rendered with four different two-dimensional opacity functions. Using a modification of an automatically generated opacity function, one rendering shows almost exclusively the registration cord laced around the body prior to scanning.

The lower right rendering in Fig. 13 demonstrates another advantage of two-dimensional opacity functions — the ability to accurately render the surface of a material which attains a wide range of data values, as is the case for the bone tissue in this same CT scan. Different parts of the bone surface are more radio-opaque than others, leading to a wide range of data values associated with bone, which in turn causes a wide range of gradient magnitudes within the boundary region between bone and soft tissue. Knowing the average second derivative for each location in (v, g) space, we can make opaque only those voxels near the middle of the boundary (near the zero-crossing in f''), regardless of the bone data value. As is visible in the opacity function generated with Eqn. 14, this implies that as gradient magnitude increases, there is an upward shift in the data values which should be made most opaque. This kind of careful opacity assignment is not possible with a simple value-based opacity function, though it is reminiscent of the two-dimensional opacity functions described by Levoy [11]. Although space does not permit a detailed comparison between our approach and Levoy's, the main difference is that (ideally) the measured first and second derivative information serves to constrain the opacity function generation so as to only show boundaries, while in Levoy's method the user still has to experiment to find the right parameter settings.

6 Conclusions and Future Work

We have shown that semi-automatic generation of opacity functions is possible for datasets where the regions of interest are boundaries between materials of relatively constant data value. The histogram volume structure presented here captures information about the boundaries present in the volume and facilitates a high-level interface to opacity function creation. The user controls which portions of the boundary are to be made opaque, without having to know the data values that occur in the boundary.

Given that boundaries in the volume are always manifested by a curve of a particular shape in the histogram volume, it makes sense to apply computer vision object recognition techniques to the histogram volume. We are investigating the feasibility of using the Hough transform to detect the curves in the histogram volume and measure their intensity [8]. Also, it may be possible to adapt the methods to non-scalar data, such as comes from multi-echo MRI. Finally, as mentioned before, we are interested in performing perceptual studies to validate the claim that direct volume rendering can, unlike isosurface rendering, accurately convey surface quality or measurement uncertainty to the viewer.

7 Acknowledgements

This work was supported in part by grants from the National Science Foundation (ASC 93-18180) to Mark Ellisman, Donald Greenberg, and Sid Karin, from the National Institutes of Health National Center for Research Resources (RR 04050) to Mark Ellisman, and by the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC 89-20219). We gratefully acknowledge equipment grants from Hewlett-Packard, on whose workstations this research was conducted. Turbine blade dataset courtesy of GE Corporate Research and Development.

References

[1] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The Contour Spectrum. In *Proceedings Visualization '97*, pages 167–173, 1997.

[2] Mark J. Bentum, Barthold B.A. Lichtenbelt, and Tom Malzbender. Frequency Analysis of Gradient Estimators in Volume Rendering. *IEEE Trans. Visualization and Computer Graphics*, 2(3):242–254, 1996.

[3] Lawrence D. Bergman, Bernice E. Rogowitz, and Lloyd A. Treinish. A Rule-based Tool for Assisting Colormap Selection. In *Proceedings Visualization '95*, pages 118–125, 1995.

[4] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[5] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*, chapter 4: Edge Detection. MIT Press, Cambridge, MA, 1993.

[6] Joachim Frank. *Electron Tomography*, pages 205–213. Plenum, New York, 1992.

[7] Taosong He, Lichan Hong, Arie Kaufman, and Hanspeter Pfister. Generation of Transfer Functions with Stochastic Search Techniques. In *Proceedings Visualization '96*, pages 227–234, 1996.

[8] J. Illingworts and J. Kittler. A Survey of the Hough transform. *IEEE Computer Vision, Graphics, and Image Processing*, 44:87–116, 1988.

[9] Granino A. Korn and Theresa M. Korn. *Mathematical Handbook for Scientists and Engineers*, page 820. McGraw-Hill, New York, 1968.

[10] Philip Lacroute and Marc Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform. In *ACM Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 451–458, July 1994.

[11] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics & Applications*, 8(5):29–37, 1988.

[12] Barthold Lichtenbelt, Randy Crane, and Shaz Naqvi. *Introduction to Volume Rendering*, chapter 4. Prentice-Hall, New Jersey, 1998.

[13] J. Marks, B. Andalman, P.A. Beardsley, and H. Pfister et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *ACM Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 389–400, August 1997.

[14] D. Marr and E. C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B 207:187–217, 1980.

[15] Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus*, chapter 2.6, 4.2. W.H. Freeman and Company, New York, 1996.

[16] Torsten Möller, Raghu Machiraju, Klaus Mueller, and Roni Yagel. A Comparison of Normal Estimation Schemes. In *Proceedings Visualization '97*, pages 19–25, 1997.

[17] National Library of Medicine (U.S.) Board of Regents. Electronic imaging: Report of the Board of Regents. U.S. Department of Health and Human Services, Public Health Service, National Institutes of Health. NIH Publication 90-2197, 1990.

[18] Durga P. Panda and Azriel Rosenfeld. Image Segmentation by Pixel Classification in (Grey Level, Edge Value) Space. *IEEE Trans. on Computers*, 27(9):875–879, September 1978.

[19] S.J. Young, G.Y. Fan, D. Hessler, S. Lamont, T.T. Elvins, M. Hadida, G. Hanyzewski, J.W. Durkin, P. Hubbard, G. Kindlmann, E. Wong, D. Greenberg, S. Karin, and M.H. Ellisman. Implementing a Collaboratory for Microscopic Digital Anatomy. *Supercomputer Applications and High Performance Computing*, 10(2/3):170–181, 1996.

Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation

J. Marks* MERL | B. Andalman Harvard Univ. | P.A. Beardsley MERL | W. Freeman MERL | S. Gibson MERL | J. Hodgins Georgia Tech. | T. Kang CMU

B. Mirtich MERL | H. Pfister MERL | W. Ruml Harvard Univ. | K. Ryall Harvard Univ. | J. Seims Univ. of Washington | S. Shieber Harvard Univ.

Abstract

Image rendering maps scene parameters to output pixel values; animation maps motion-control parameters to trajectory values. Because these mapping functions are usually multidimensional, nonlinear, and discontinuous, finding input parameters that yield desirable output values is often a painful process of manual tweaking. Interactive evolution and inverse design are two general methodologies for computer-assisted parameter setting in which the computer plays a prominent role. In this paper we present another such methodology. *Design Gallery*™ (DG) interfaces present the user with the broadest selection, automatically generated and organized, of perceptually different graphics or animations that can be produced by varying a given input-parameter vector. The principal technical challenges posed by the DG approach are *dispersion*, finding a set of input-parameter vectors that optimally disperses the resulting output-value vectors, and *arrangement*, organizing the resulting graphics for easy and intuitive browsing by the user. We describe the use of DG interfaces for several parameter-setting problems: light selection and placement for image rendering, both standard and image-based; opacity and color transfer-function specification for volume rendering; and motion control for particle-system and articulated-figure animation.

CR Categories: I.2.6 [Artificial Intelligence]: Problem Solving, Control Methods and Search—heuristic methods; I.3.6 [Computer Graphics]: Methodology and Techniques—interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Keywords: Animation, computer-aided design, image rendering, lighting, motion synthesis, particle systems, physical modeling, visualization, volume rendering.

1 Introduction

Parameter tweaking is one of the vexations of computer graphics. Finding input parameters that yield a desirable output is difficult and tedious for many rendering, modeling, and motion-control processes. The notion of having the computer assist actively in setting parameters is therefore appealing. One such computer-assisted methodology is interactive evolution [11, 21, 23]: the computer explores the space of possible parameter settings, and the user acts as an objective-function oracle, interactively selecting computer-suggested alternatives for further exploration. A more automatic methodology is inverse design, e.g., [10, 12, 14, 19, 22, 25, 27]: the computer searches for parameter settings that optimize a user-supplied, mathematically stated objective function.

Unfortunately, there are many interesting and important graphics processes for which interactive evolution and inverse design are not very useful. These processes share two common characteristics:

- High computational cost: if the process cannot be computed in near real time, interactive evolution becomes unusable.
- Unquantifiable output qualities: even though desirable graphics may be readily identified by inspection, it may not be possible to quantify a priori the qualities that make them desirable. This lack of a suitable objective function rules out the use of inverse design.

In this paper we present a third methodology for computer-assisted parameter setting that is especially applicable to graphics processes that exhibit one or both of these characteristics. *Design Gallery* (DG) interfaces present the user with the broadest selection, automatically generated and organized, of perceptually different graphics or animations that can be produced by varying a given input-parameter vector. Because the selection is generated automatically, it can be done as a preprocess so that any high computational costs are hidden from the user. Furthermore, the DG approach requires only a measure of similarity between graphics, which can often be quantified even when optimality cannot.

A DG system includes several key elements. The *input vector* is a list of parameters that control the generation of the output graphic via a *mapping* process. The *output vector* is a list of values that summarizes the subjectively relevant qualities of the output graphic. The *distance metric* on the space of output vectors approximates the perceptual similarity of the corresponding output graphics. The *dispersion* method is used to find a set of input vectors that map to a well-distributed set of output vectors, and hence output graphics. The dispersed graphics are presented to the user through a perceptually reasonable *arrangement* method that makes use of the distance metric. These six elements — input vector, mapping, output vector, distance metric, dispersion, and arrangement — characterize a DG system. The creator of a DG system chooses the input vector, output vector, and the distance metric for a specific mapping process. For particular instances of the process, the computer performs the dispersion, the mapping of input vectors to output vectors, and the arrangement of final graphics in a gallery. The end user need only recognize and select appealing graphics from the gallery.

*Address: MERL – A Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, MA 02139, U.S.A. E-mail: marks@merl.com.

We explain and illustrate the use of DGs for several common parameter-setting problems: light selection and placement for image rendering, both standard and image-based; opacity and color transfer-function specification for volume rendering; and motion control for particle-system and articulated-figure animation. During the discussion, we describe the input and output vectors for each mapping process, and present various methods for dispersion and arrangement that we have used in building DG systems.

2 Light Selection and Placement

Setting lighting parameters is an essential precursor to image rendering. Previous attempts at computer-assisted lighting specification have used inverse design. For example, the user can specify the location of highlights and shadows in the image [15], pixel intensities [19], or subjective impressions of illumination [10]; the computer then attempts to determine lighting parameters that best meet the given objectives, using geometric [15] or optimization [10, 19] techniques. Unfortunately, the formulation of lighting specification as an inverse problem has some significant drawbacks. High-quality image rendering (e.g., raytracing or radiosity) is costly; to make the computer's search task tractable, the user may have to fix the light positions [10, 19], thereby grossly limiting the illuminations that can be considered. A more intrinsic difficulty is that of requiring the user to quantify a priori the desired illuminative characteristics of the resulting image. This requirement may be satisfiable in an architectural context [10], but seems very challenging in a more general cinematographic context [8]. The most difficult lighting parameters to set are those relating to light type and placement, so they have been the focus of our efforts.

2.1 Input and Output Vectors

For the light selection and placement problem, we begin with a scene model comprising surfaces and viewing parameters. The goal is to explore different ways of lighting the scene, so the input vector includes a light position, a light type, and a light direction if needed. The light position is located somewhere on one of the surfaces distinguished as a *light hook* surface by the user. The light type comes from a user-defined group, and describes attributes of the light: its basic class (e.g., point, area, or spotlight); whether or not it casts shadows; its falloff behavior (e.g., none, linear, or quadratic); and class-specific parameters (e.g., the beam angle of a spotlight). Directional lights are aimed at randomly chosen points on designated *light target* surfaces.

The output vector should be a concise, efficiently computed set of values that summarizes the perceptual qualities of the final image. Thus, output vectors are based on pixel luminances from several low-resolution thumbnail images (32×25 pixels and smaller). The luminances at resolution ρ are weighted by a factor $f(\rho)$. The distance metric on the output vector is the standard L^1 (Manhattan) distance. As a result, the distance between output vectors corresponding to images q and r is

$$\sum_{\rho \in \{1, \frac{1}{16}, \frac{1}{16^2}, \ldots\}} \sum_{x,y} f(\rho) |Y_q^\rho(x,y) - Y_r^\rho(x,y)| \qquad (1)$$

where $Y_q^\rho(x,y)$ is the luminance of the pixel at location (x,y) in image q at resolution ρ.[1]

[1] Since we start with a low-resolution thumbnail, the filtered images of even lower resolution called for in the expression will be truly tiny. Nevertheless, they do contain useful information: two barely nonoverlapping narrow-beam spotlights will generate a high (and somewhat misleading) difference score at the highest resolution, but smaller, more appropriate difference scores at lower resolutions because the beams will overlap in the lower-resolution images. The effect of the weighting function $f(\rho)$ is subtle, but we have found it preferable to weight higher-resolution images slightly more than lower-resolution ones.

2.2 Dispersion

The dispersion phase selects an appropriate subset of input vectors from a random sample over the input space. Specifically, T lights are generated at each of H positions distributed uniformly over the light hook surfaces. This procedure yields a set L of $H \times T$ input vectors. Typical values are $H = 500$ and $T = 8$, in which case $|L| = 4000$.[2] For each input vector in L, thumbnail images are generated, and the corresponding output vector is determined as described above. The dispersion algorithm outlined in Figure 1 then finds a set $I \subset L$ with good spread among output vectors. The first step is the elimination of lights that dimly illuminate the visible part of the scene, because they are obscured or point away from the scene geometry; these lights are unlikely to be of interest to the user and can confound the rest of the dispersion process. Thumbnail images whose average luminance is less than a cutoff factor c are eliminated from the set L. (Typical useful values of c are in the range 1%–5% of the maximum luminance value.) The subset I is assembled by repeatedly adding to I the light in L whose output vector is most different from its closest match in the nascent I. The size of I is

Input:
L, a set of lights and corresponding thumbnail images.
$n < |L|$, the size of the selected subset.
c, an average-luminance cutoff factor.

Output:
$I \subset L$, a set of n dispersed lights and their images.

Procedure:

```
SELECTION_DISPERSE(L, n, c) {
    L ← L \ find_dims(c, L);
    I ← ∅;
    for i ← 1 to n do {
        p_score ← −∞;
        foreach q ∈ L do {
            q_score ← ∞;
            foreach r ∈ I do
                if image_diff(q, r) < q_score then
                    q_score ← image_diff(q, r);
            if q_score > p_score then {
                p_score ← q_score;
                p ← q;
            }
        }
        I ← I ∪ {p};
        L ← L \ {p};
    }
}
```

Notes:
\ denotes set difference.
find_dims(c, L) returns those lights in L with average luminance less than c.
image_diff(q, r) returns the value computed by Equation 1.

Figure 1: A selection-based dispersion heuristic.

[2] We picked these numbers to allow overnight batch processing of the entire DG process for one scene on a single MIPS R10000 processor.

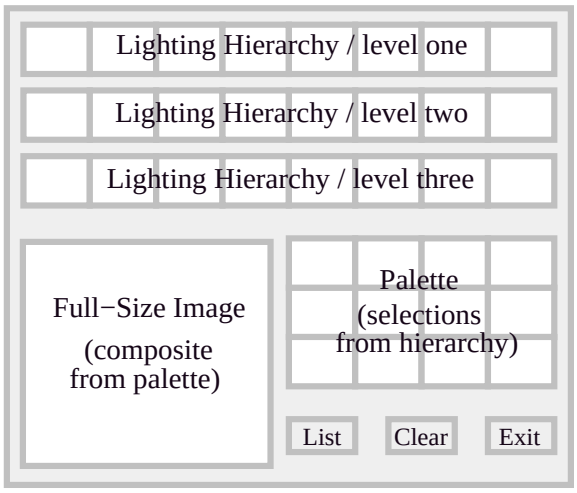


Figure 2: User-interface map.

determined by the interface, as described below; $|I| = 584$ for the examples we discuss in the paper.

2.3 Arrangement

We would like the set of lights I to be large, so that the user will have many complementary lights from which to choose. However, the greater the size of I, the more difficult it will be for the user to browse the lights effectively. We accommodate these contradictory requirements by arranging the set I in a fully balanced hierarchy in which lights that produce similar illumination effects are grouped together. We accomplish this goal of the arrangement phase by graph partitioning. A complete graph is formed in which the vertices correspond to the lights in I, and edge costs are given by the inverse of the distance metric used in the dispersion phase. An optimal w-way partition of this graph would comprise w disjoint vertex subsets of equal cardinality such that the cost of the cut set, the total cost of all edges that connect vertices in different subsets, is minimized. Optimal graph partitioning is NP-hard [4], but many good heuristics have been developed for this problem [1]. Our partitioning code is based on an algorithm and software developed by Karypis and Kumar [9]. Once the initial w-way partition is formed, representative lights for each partition are selected, and installed in the hierarchy. The partitioned subsets, minus their representative vertices, are then processed recursively until a hierarchy with branching factor w and height h is completed.

The values for w and h are dictated by the user interface, whose structure is depicted in Figure 2, and actual examples of which are shown in Figures 9–11. For each light in the final set I, medium-size (128×100 pixels) and full-size (512×400 pixels) images are generated for use in the interface. The user is presented with a row of eight images that serve as the first level of the light hierarchy. Clicking on one of these images causes its eight children in the hierarchy to be presented in the next row of images. The third and final level in the hierarchy is accessed by clicking on an image from the second row. Thus $w = 8$ and $h = 3$. In turn, these parameters determine the cardinality of I: $|I| = \sum_{j=1}^{h} w^j = 584$. This particular interface provides additional application-specific functionality that exploits the additive nature of light [6]. Images can be dragged to the palette, where light intensity and temperature can be varied interactively. Multiple images are composited to form a full-size image in the lower left.

2.4 Results

The DG in Figure 9 contains a scene inspired by an example from [8]. The floor, ceiling, and all four walls (only the rear one is visible) were designated light-hook surfaces. The surfaces comprising the figures were designated light-target surfaces, as was the back wall. The 584 lights in the gallery were selected from 5,000 randomly generated lights in the dispersion phase. The cost of computing this and the other light-selection-and-placement DGs shown here was dominated by the cost of raytracing the 584 full-size images used in the display, which took approximately five hours on a MIPS R10000 processor.

Figure 10 contains a scene with richer geometry. The ceiling, and the area around the base of the statue were designated light-hook surfaces. The surfaces of the two heads, the doors, the tree, and the statue were designated light-target surfaces. The gallery lights were selected from 3,000 randomly generated lights in the dispersion phase.

Finally, Figure 11 shows a DG for synthetic lighting of a photograph (inset at lower right). A point- and line-based 3D model is extracted from a triplet of scene images, each taken from a different viewpoint. This reconstruction process is completely automatic, as described in [2]. Points and lines are then aggregated semi-automatically into planes. An illumination of the final recovered model is used to modulate intensity in one of the original photographs.

3 Opacity and Color Transfer Functions for Volume Rendering

Choosing the opacity and color transfer functions for volume rendering is another tedious and difficult manual task amenable to a DG approach.[3] We developed DG interfaces for two data sets: the simulated electron density of a protein, and a CT scan of a human pelvis.

3.1 Input and Output Vectors

The protein data set contains values in the interval $[0, 255]$. The opacity transfer function over this domain is parameterized by a polyline with eight control points, for a total of 16 values. The polyline is low-pass filtered before it is used. The color transfer function is parameterized by five values that segment the data into six subranges, which are arbitrarily assigned the colors red, yellow, green, cyan, blue, and magenta. Thus color is being used only to identify subranges of the data, and not to convey any quantitative relations among the data. Figure 3 illustrates a sample opacity and color transfer function. The complete input vector comprises 23 parameters.

For the scene-lighting DG, the output vector contains approximately 850 weighted pixel luminances. This kind of resolution is necessary because lights can cause completely local illumination effects in a synthetically rendered image, effects that should be representable in the output vector. In comparison, changes to transfer functions will generally affect many pixels throughout a volume-rendered image. We can take advantage of this homogeneity by including only a handful of pixels in the output vector. Currently we use eight pixels, selected manually for each data set. Representing all of their YUV values requires 24 values in the output vector, and standard Euclidean distance is used as the output-space metric. Dispersion on the basis of eight pixels from different parts of the image produces excellent dispersion of complete images at a much reduced computational cost.

[3] The application of both interactive evolution and inverse design to this problem is the subject of [7].

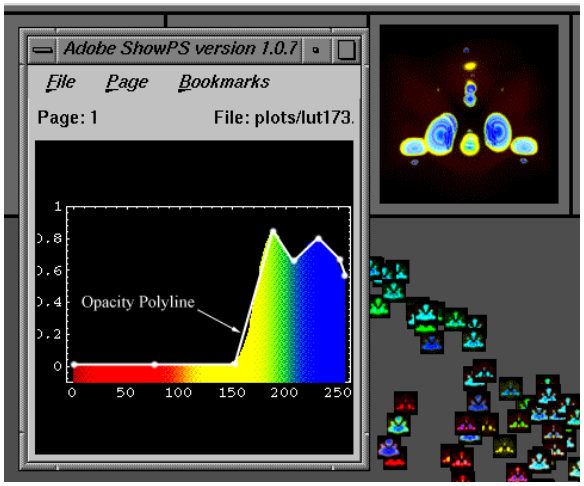


Figure 3: Pop-up display depicting transfer functions.

3.2 Dispersion

The dispersion heuristic in Figure 1 works by distilling a set of randomly generated input vectors down to a well-dispersed subset. Although simple, this method has the drawback of not utilizing what is learned via random sampling about the mapping from input to output vectors. In contrast, the dispersion heuristic in Figure 4 uses an evolutionary strategy that adapts its sampling over time in response to what it implicitly learns, and consequently performs much better. It starts with an initial set of random input vectors. These vectors are then perturbed randomly. Perturbed vectors are substituted for existing vectors in the set if the substitution improves dispersion. The key notion of dispersion used is nearest-neighbor distance in the space of output vectors.

3.3 Arrangement

The arrangement method based on graph partitioning that is presented in §2.3 results in a simple and easy-to-use interface. Unfortunately, sometimes the partition contains anomalies, e.g., dissimilar lights placed in the same subset of the partition. This problem is due to limitations of the partitioning method (no heuristic partitioning strategy guarantees an optimal partition), and to the structure of the set of output vectors, which may not map well to any regular hierarchical partition.

For the volume-rendering application, we used an alternative arrangement method that eschews a partition-based or hierarchical framework and instead illustrates the structure of the set of output vectors graphically in a 2D layout. An interface for this arrangement method is shown in Figure 5. A thumbnail, which in this case is a small, low-resolution volume-rendered image, is generated for each final output vector. The thumbnails are arranged in the center display panel, in a manner that correlates the distance between thumbnails with the distance between the associated output vectors. The thumbnail display panel can be panned and zoomed. Selecting a thumbnail brings up a full-size image, which can then be moved to the surrounding image gallery. Mousing on an image in the gallery highlights its associated thumbnail, and vice versa.

Thumbnail layout is accomplished using a multidimensional scaling (MDS) [3] method due to Torgerson [24].[4] Given a matrix of distances between points, MDS procedures compute an embedding of the points in a low-dimensional Euclidean space (2D in our case) such that the interpoint distances in the embedding closely match those in the given matrix. Torgerson's "classical scaling" method, although simpler and less general than iterative methods, is fast and robust. When the interpoint distances come from an embedding of the points in a high-dimensional Euclidean space (which is true for the applications we discuss here, although it need not be true in general), classical scaling is equivalent to an efficient technique for computing a principal-component analysis of the points [5, 13].

The layouts computed by classical scaling are not without anomalies — as we are using it, this MDS method is a projection from a high-dimensional space onto a 2D space, which cannot be done without loss of information — but they do reflect the underlying structure of the output vectors well enough to allow effective browsing. One important practical detail: since full-size versions of all the images returned by the dispersion procedure must be rendered anyway, it is convenient and better to compute distances from these full-size images in the arrangement phase, instead of from the eight pixels used in the dispersion phase.

[4] The use of more sophisticated MDS techniques for arranging a database of images is being investigated by Rubner et al. [18].

Input:
A random set of input vectors, I, and their corresponding output vectors, O. $|I| = |O| = n$.

A trial count, t.

Output:
Modified sets of input and output vectors, I and O.

Procedure:

```
EVOLUTION_DISPERSE(I, O, t) {
    for i ← 1 to t do {
        j ← rand_int(1, n);
        u ← perturb(I[j], i);
        map(u, v);
        k ← worst_index(O);
        if is_better(v, O[k], O) then {
            I[k] ← u;
            O[k] ← v;
        }
        else if is_better(v, O[j], O) then {
            I[j] ← u;
            O[j] ← v;
        }
    }
}
```

Notes:

rand_int$(1, n)$ returns a random integer in the range $[1, n]$.

perturb$(I[j], i)$ returns a copy of $I[j]$ in which all the elements have been perturbed. The magnitude of the perturbations is inversely proportional to i.

map(u, v) maps input vector u to output vector v using an application-specific mapping process.

worst_index(O) returns the index of the output vector in O with minimum nearest-neighbor distance. Ties are broken using the average distance to all other vectors in O.

is_better$(v, O[k], O)$ returns true if the nearest neighbor to v in $O \setminus \{O[k]\}$ is further away than the nearest neighbor to $O[k]$ in O. Ties are broken using average distance to all other vectors in the relevant set.

Figure 4: An evolutionary dispersion heuristic.

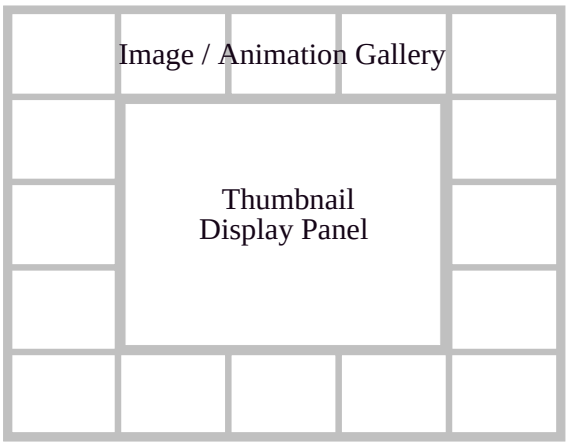


Figure 5: A more flexible user interface.

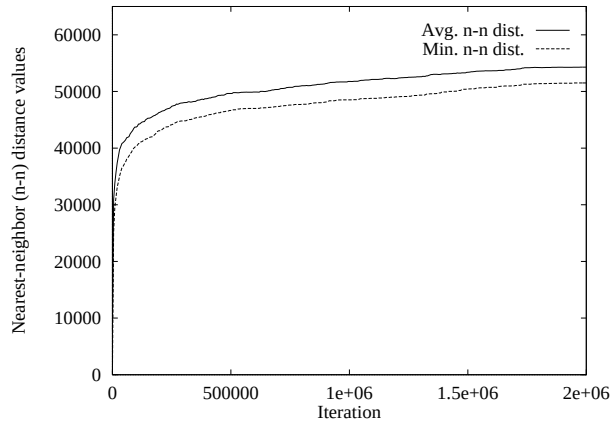


Figure 6: Nearest-neighbor distances over time.

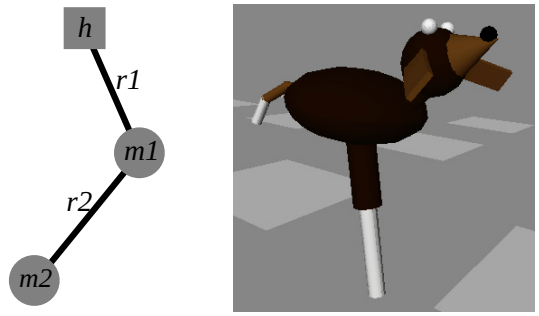


Figure 7: Articulated linkages.

3.4 Results

Figure 12 illustrates the DG for the volume rendering of the protein data set. The dispersion procedure returned 256 dispersed input and output vectors. A selection of images is shown in the surrounding image galleries. The lines that connect images with their thumbnails give some indication of how images congregate in the thumbnail display. (During interactive use the association between thumbnails and images is done preferably by dynamic highlighting, as described above.) Figure 3 shows the result of clicking on one of the images in the image gallery: the corresponding opacity and color transfer functions are depicted in a pop-up window, allowing the user to see how image and data relate.

The performance of the dispersion heuristic from this experiment is documented in Figure 6; this data is representative of all the DG experiments that use the evolutionary dispersion heuristic. The curves show how two values, the minimum and average nearest-neighbor distances in the set of output vectors, increase over time. Improvement is rapid at first: the minimum and average nearest-neighborhood distances in the initial random set are 184 and 7,789, respectively. However, the rate of improvement drops quickly. Although we used a trial count of $t = 2,000,000$ (see Figure 4), it is clear that relatively little improvement occurred after $t = 500,000$. To reach this point requires $8 \times 500,000 = 4,000,000$ raycast operations and takes less than 40 minutes on a single MIPS R10000 processor. This duration is roughly one-sixth of that needed to render the 256 full-size images (300×300 pixels) for the DG.

A second volume-rendering experiment was performed using a computed tomography (CT) data set for a human pelvis. These data values are presegmented into four disjoint subranges, one each for air, fat, muscle, and bone. The input vector specifies the y-coordinates of 12 opacity control points; the x-coordinates are held fixed. The input vector does not specify a color transfer function, since standard colors are used for the different tissue types. The output vector, distance metric, dispersion, and arrangement were identical to the protein-rendering experiment. Figure 13 illustrates the DG for the volume rendering of the pelvis data set.

4 Animation Applications

Motion control in animation involves extensive parameter tuning because the mapping from input parameters to graphical output is nonintuitive, unpredictable, and costly to compute.[5] For these reasons, motion control is very amenable to a DG approach. Building a DG interface for animation is similar to building one for still images (we reuse the dispersion and arrangement code from §3 virtually without change); the major differences are in computing the output-vector components. We now discuss three DG systems for animation tasks, focusing on this latter issue.

4.1 2D Double Pendulum

The 2D double pendulum is a simple dynamic system with rich behavior that makes it an ideal test case for parameter-setting methodologies.[6] A double pendulum consists of an attachment point h, two bobs of masses m_1 and m_2, and two massless rods of lengths r_1 and r_2, connected as shown in Figure 7. Our pendulum also includes motors at the joints at h and m_1 that can apply sinusoidal time-varying torques. The input vector comprises the rod lengths, the bob masses, the initial angular positions and velocities of the rods, and the amplitude, frequency, and phase of both sinusoidal torques, for a total of 14 parameters.

Choosing a suitable output vector proved to be the most difficult part of the DG process for the double pendulum, as well as for the other motion-control applications; several rounds of experimentation were needed (see §5 for more details). The output vector must

[5] Both interactive evolution [26] and inverse design [12, 14, 22, 25, 27] have been applied previously to motion control.

[6] Even without the application of external torques at its joints, the 2D double pendulum exhibits chaotic behavior [20].

capture the behavior of the system over time. For the double pendulum, the output vector has 12 parameters: the differences in rod lengths and bob masses, the average Cartesian coordinates of each bob, and logarithms of the average angular velocity, the number of velocity reversals, and the number of revolutions for each rod. Euclidean distance is used as the distance metric on this output space.

The mapping from input vector to output vector is accomplished by dynamically simulating 20 seconds of the pendulum's motion, and using the algorithm in Figure 4 for dispersion. Arrangement is accomplished using the MDS layout method of §3.3. The displayed thumbnails are static images of the final state of the pendulum, along with a trail of the lower bob over the final few seconds. We found that these images give enough clues about the full animation to enable effective browsing. Thumbnails can be dragged into gallery slots, all of which can be animated simultaneously by clicking on any occupied slot.

Figure 14 shows the DG for the double pendulum. As before, the overlaid lines show where animations in the gallery are located in the thumbnail display. The plateau in nearest-neighbor distance is reached after $170,000$ dispersion iterations, which take 6.5 hours on a single MIPS R10000 processor.

4.2 3D Hopper Dog

The previous DG is useful in finding and understanding the full range of motions possible for the pendulum under a given control regime. However, complete generality is not always a useful goal: the animator may have some preconceived idea of a motion that needs subtle refinement to add nuance and detail. The 3D hopper dog, shown in Figure 7, is an articulated linkage with rigid links connected by rotary joints. It has a head, ears, and tail, and moves by hopping on its single leg. It has 24 degrees of freedom (DOF). The hopper dog is actuated by a control system that tries to maintain a desired forward velocity and hopping height, as well as desired positions for joints in some of the appendages. The equations of motion for the system are generated using a commercially available package[17]; dynamic simulation is used to produce the animations.

We started with a basic hopping motion, and then used a DG approach to explore seven input quantities in order to achieve stylistic, physically attainable gaits. The seven quantities are: the forward velocity, the hopping height, and the positions of 2-DOF ear joints, a 2-DOF tail joint, and a 1-DOF neck joint. For each of these seven, a time-varying sinusoid specifies the desired trajectory, with the minimum value, maximum value, and frequency specified in the input vector, which therefore contains 21 values.

In this particular case, the elements of the output vector correspond closely to those of the input vector. The 14-element output vector contains the averages and variances of the same seven quantities, and is obtained by dynamically simulating 30 seconds of the hopper dog's motion. (Output vectors from simulations in which the hopper dog falls are discarded automatically.) As for the previous two applications, the output-space distance metric is Euclidean, and the arrangement method and interface from §3.3 are used. The hopper-dog DG is illustrated in the video proceedings.

4.3 Particle Systems

Particle systems are useful for modeling a variety of phenomena such as fire, clouds, water, and explosions [16]. A useful particle-system editor might have 40 or more parameters that the animator can set, so achieving desired effects can be tedious. As in the previous subsection, we use a DG interface to refine an animator's rough approximation to a desired animation.

The subject for our experiment is a hypothetical beam weapon for NASA space shuttles. A first draft was produced by hand using a regular particle-system editor; a still from midway through the animation is shown in Figure 8. The input vector contains the subset of particle-system controls that the animator wishes to have tweaked. In this example the controls govern: the mean and variance of particle velocities, particle acceleration, rate of particle production, particle lifetime, resilience and friction coefficient of collision surfaces, and perturbation vectors for surface normals. Among the parameters that are held fixed are the origin, average direction, and color of the beam.

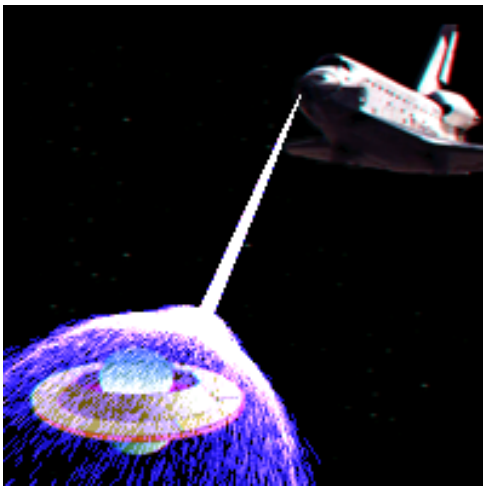


Figure 8: A still from a particle-system animation.

For efficiency reasons, DG output vectors are based on subsampled versions of the final graphic where possible, thereby reducing computational costs and allowing more of the space to be explored. For example, static images can be rendered at low resolution (§2 and §3). The subsampling strategy for the particle animation is to simulate only every 500th particle generated during the dispersion phase, and to examine the state of the particle system at just two distinct points in time: once midway through the simulation, and once at the end. The output vector comprises measures of the number of particles, their average distance from the origin and the individual variation in this distance, their spread from the average beam, the average velocity of the entire system, and the individual variation from this average (we take logs of all of these quantities except for the beam spread). These six measures are included for each of the two distinguished times, resulting in 12 output parameters. Euclidean distance is the metric on the output space.

Figure 15 shows the DG of variations on the animator's original sketch from Figure 8. The dispersion and arrangement methods from §3 are used to generate the DG. Each thumbnail is the midway still from the corresponding animation. (The user can optionally select thumbnails from different stages in the animation.) As with the double-pendulum DG, thumbnails can be dragged to gallery slots and animated therein. Also as before, lines connect animation stills with their associated thumbnails. The dispersion heuristic ran for $t = 100,000$ iterations, at which point it appeared to reach a plateau. This number of trials took approximately six hours on a MIPS R10000 processor. Generating the 256 animations in the DG with their full complement of particles took a little under five hours on the same processor.

5 Discussion

Table 1 summarizes the DGs described in this paper, in terms of

Application	Light selection & placement	Volume rendering	Double pendulum	Particle system	Hopper dog
Input Vector	Light type, location, and direction	Control points for opacity/color transfer functions	Pendulum dimensions, initial conditions, motor torques	Animator-specified subset of particle control parameters	Desired trajectory sinusoids
Output Vector	Luminances of thumbnail pixels	YUV values for eight pixels	Trajectory statistics (mainly logs of time averages and variances)		
Distance Metric	Manhattan	Euclidean			
Mapping	Raytracing	Volume rendering	2D dynamic simulation	3D particle simulation	3D dynamic simulation
Dispersion	Selection from random sample over neighborhood	Evolution from full random sample		Evolution from random sample over neighborhood	
Arrangement	Graph partitioning	Multidimensional scaling			

Table 1: Summary of Design Gallery experiments.

the six basic elements of a DG system. Some of the variation in this table is application specific, while the remainder stems from our investigation of alternative dispersion and arrangement methods. All of the galleries described in the paper produce a useful variety of output graphics.

Using a DG for a particular instance of a design problem is fairly straightforward for the end user. Aside from browsing the final DG, the user's only other task may be to loosely focus the dispersion process by, for example, selecting suitable light-hook and light-target surfaces (§2), or by specifying a relevant subset of particle-control parameters (§4.3). However, creating a DG system for an entire class of design problems is more difficult. The DG-system creator is responsible for choosing the structure of the input and output vectors, and the distance metric on the output space. Thus, the creator needs a better understanding of the design problem than the end user. Of the creator's tasks, the simplest is choosing the distance metric: very standard metrics sufficed for all applications we tried. Choosing the input vector is also straightforward. Even when there are many possible ways to parameterize the input, our experience is that choosing an acceptable parameterization is not hard.

The most difficult task of the DG-system creator is devising an output vector. The first two DGs in Table 1 work on static images. In these examples, the perceptual similarity between images correlates well with subsampled image or pixel differences, hence the output vectors comprise subsampled image and pixel values. An added advantage is that the ranges of all components of the output vector are bounded and known. Finding measures that capture the perceptual qualities of a complete animation is harder. The DG systems for animation tasks required several experiments to get a suitable output vector, although the process became easier for each successive system. Among the lessons learned in developing output vectors for motion-control problems, the two most important precepts are, with hindsight, fairly obvious:

- Take the log of quantities that have a large dynamic range. For many such quantities, e.g., velocity, human ability to resolve changes in magnitude diminishes as the magnitude increases. To uniformly sample the perceptual space, one must therefore sample the lower end of the dynamic range more thoroughly.

- The relative weights of the output-vector parameters matter. In general, the output-vector parameters should be scaled so that they each have approximately the same dynamic range, otherwise only the parameters with the largest ranges will be dispersed effectively.

What inevitably happened with a poorly chosen output vector was that the dispersion algorithm found a malicious way to get unfortunate and unexpected spread in one of the vector coordinates, usually through a degenerate set of input parameters, e.g., pendulums with extremely short links and very high rpm's, and particle systems with only a few particles, but very high variance in velocity.

In our experiments, we investigated two dispersion methods and two arrangement methods. The dispersion method of Figure 4 is more complex, but performs better. However, an advantage of the simpler method in Figure 1 is that it may be easier to parallelize. Two arrangement methods were also tried, one based on graph partitioning and the other on MDS. Both allowed the user to navigate through the output graphics effectively, and both had their fans among our group of informal testers. Layout and organizational anomalies were occasionally evident in both interfaces, but they did not hinder the user's ability to peruse the output graphics.

6 Conclusion

Design Gallery interfaces are a useful tool for many applications in computer graphics that require tuning parameters to achieve desired effects. The basic DG strategy is to extract from the set of all possible graphics a subset with optimal coverage. A variety of dispersion and arrangement methods can be used to construct galleries. The construction phase is typically computationally intensive and occurs off-line, for example, during an overnight run. After the gallery is built, the user is able to quickly and easily browse through the space of output graphics.

Inverse design is one technique for setting parameters, but it is only feasible when the user can articulate or quantify what is desired. DGs replace this requirement with the much weaker one of quantifying similarity between graphics. Unlike interactive evolution, DGs are feasible even when the graphics-generating process has high computational cost. Finally, DGs are useful even when the user has absolutely no idea what is desired, but wants to know what the possibilities are. This is often the first step in the creative design process.

7 Acknowledgments

The protein data set used in §3 was made available by the Scripps Clinic, La Jolla, California. Volume rendering was performed using the VolVis system from SUNY Stony Brook, as modified by T. He. D. Allison, R. Kaplan, a. shelat, Y. Siegal, and R. Surdulescu provided helpful comments and code. The university participants were supported in part by grants from MERL and from the NSF (Grants No. IRI-9350192 and No. IRI-9457621), and by a Packard Fellowship.

References

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: a survey. *Integration: The VLSI Journal*, 19:1–81, 1995.

[2] P. A. Beardsley, A. P. Zisserman, and D. W. Murray. Sequential updating of projective and affine structure from motion. *International Journal of Computer Vision*, 1997. In press.

[3] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 1997.

[4] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[5] J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–338, 1966.

[6] P. Haeberli. Synthetic lighting for photography. URL http://-www.sgi.com/grafica/synth/index.html, Jan. 1992.

[7] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *Proc. of Visualization 96*, pages 227–234, San Francisco, California, Oct. 1996.

[8] J. Kahrs, S. Calahan, D. Carson, and S. Poster. Pixel cinematography: a lighting approach for computer graphics. Notes for Course #30, SIGGRAPH 96, New Orleans, Louisiana, Aug. 1996.

[9] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Technical report, Dept. of Computer Science, Univ. of Minnesota, 1995. See also URL http://-www.cs.umn.edu/˜karypis/metis/metis.html.

[10] J. K. Kawai, J. S. Painter, and M. F. Cohen. Radioptimization – goal-based rendering. In *SIGGRAPH 93 Conf. Proc.*, pages 147–154, Anaheim, California, Aug. 1993.

[11] S. Kochhar. A prototype system for design automation via the browsing paradigm. In *Proc. of Graphics Interface 90*, pages 156–166, Halifax, Nova Scotia, May 1990.

[12] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. In *SIGGRAPH 94 Conf. Proc.*, pages 35–42, Orlando, Florida, July 1994.

[13] H. Murakami and B. V. K. V. Kumar. Efficient calculation of primary images from a set of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-4(5):511–515, Sept. 1982.

[14] J. T. Ngo and J. Marks. Spacetime constraints revisited. In *SIGGRAPH 93 Conf. Proc.*, pages 343–350, Anaheim, California, Aug. 1993.

[15] P. Poulin and A. Fournier. Lights from highlights and shadows. In *Proc. of the 1992 Symposium on Interactive Graphics*, pages 31–38, Boston, Massachusetts, Mar. 1992. In *Computer Graphics* 25(2), 1992.

[16] W. T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Trans. on Graphics*, 2:91–108, Apr. 1983.

[17] D. E. Rosenthal and M. A. Sherman. High performance multibody simulations via symbolic equation manipulation and Kane's method. *Journal of Astronautical Sciences*, 34(3):223–239, 1986.

[18] Y. Rubner, L. J. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proc. of the DARPA Image Understanding Workshop*, New Orleans, May 1997.

[19] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. Painting with light. In *SIGGRAPH 93 Conf. Proc.*, pages 143–146, Anaheim, California, Aug. 1993.

[20] T. Shinbrot, C. Grebogi, J. Wisdom, and J. A. Yorke. Chaos in a double pendulum. *American Journal of Physics*, 60(6):491–499, 1992.

[21] K. Sims. Artificial evolution for computer graphics. In *Computer Graphics (SIGGRAPH 91 Conf. Proc.)*, volume 25, pages 319–328, Las Vegas, Nevada, July 1991.

[22] K. Sims. Evolving virtual creatures. In *SIGGRAPH 94 Conf. Proc.*, pages 15–22, Orlando, Florida, July 1994.

[23] S. Todd and W. Latham. *Evolutionary Art and Computers*. Academic Press, London, 1992.

[24] W. S. Torgerson. *Theory and Methods of Scaling*. Wiley, New York, 1958. See especially pages 254–259.

[25] M. van de Panne and E. Fiume. Sensor-actuator networks. In *SIGGRAPH 93 Conf. Proc.*, pages 335–342, Anaheim, California, Aug. 1993.

[26] J. Ventrella. Disney meets Darwin – the evolution of funny animated figures. In *Proc. of Computer Animation 95*, pages 35–43, Apr. 1995.

[27] A. Witkin and M. Kass. Spacetime constraints. In *Computer Graphics (SIGGRAPH 88 Conf. Proc.)*, volume 22, pages 159–168, Atlanta, Georgia, Aug. 1988.

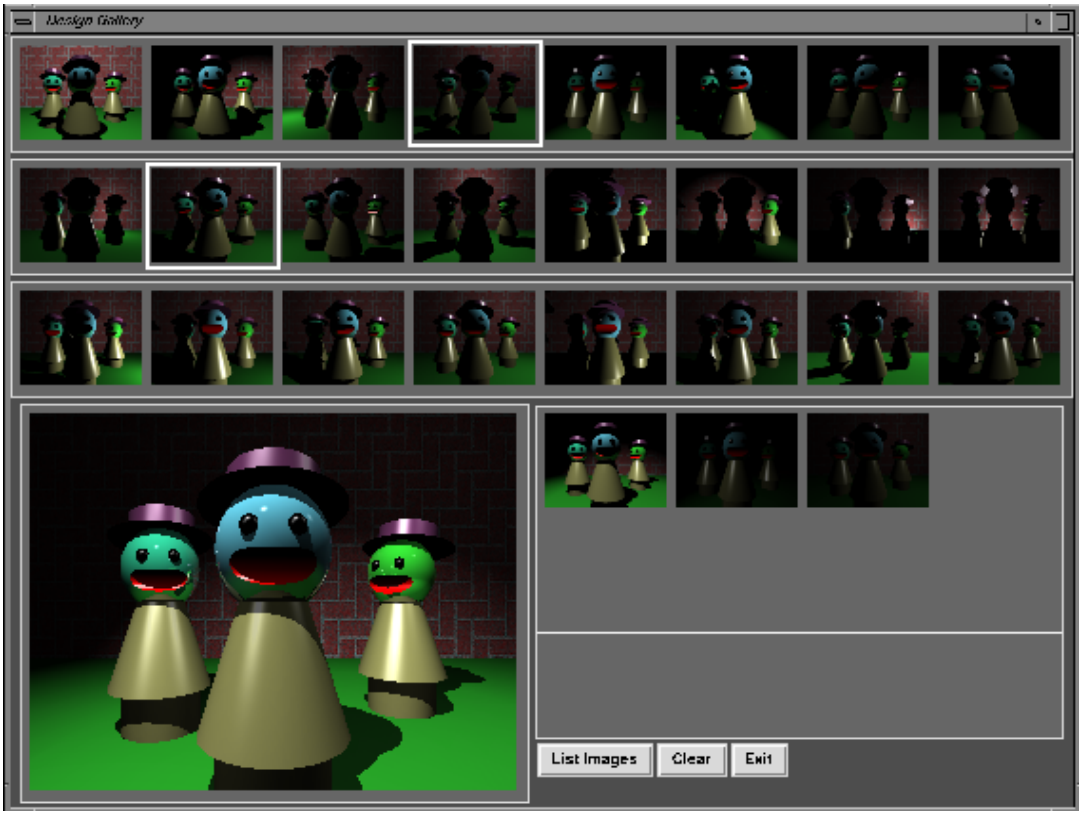


Figure 9: A DG for light selection and placement.

Figure 10: Another DG for light selection and placement.

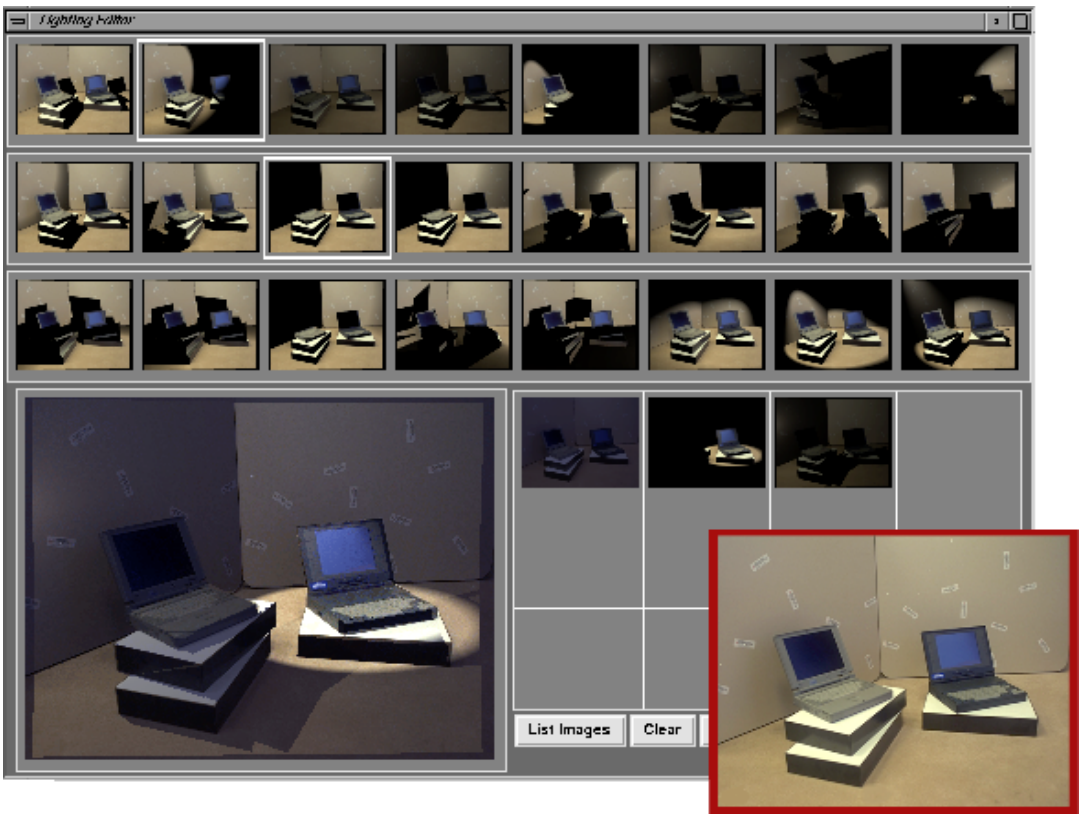


Figure 11: Light selection and placement for synthetic lighting of a photograph.

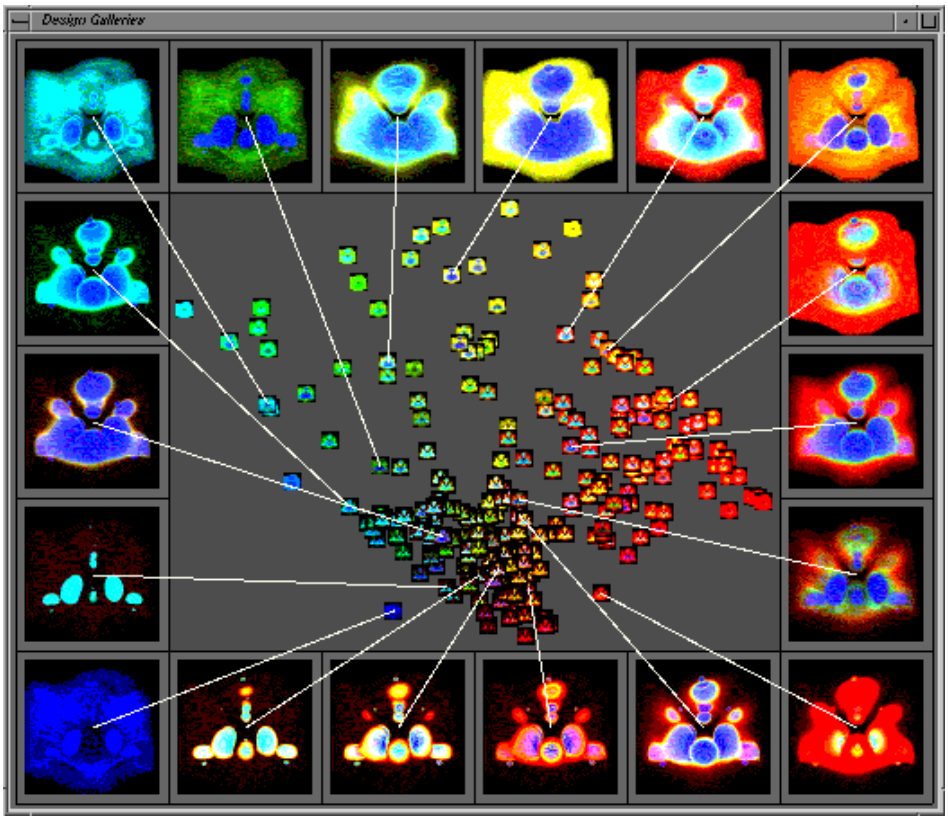


Figure 12: A DG with different opacity and color transfer functions.

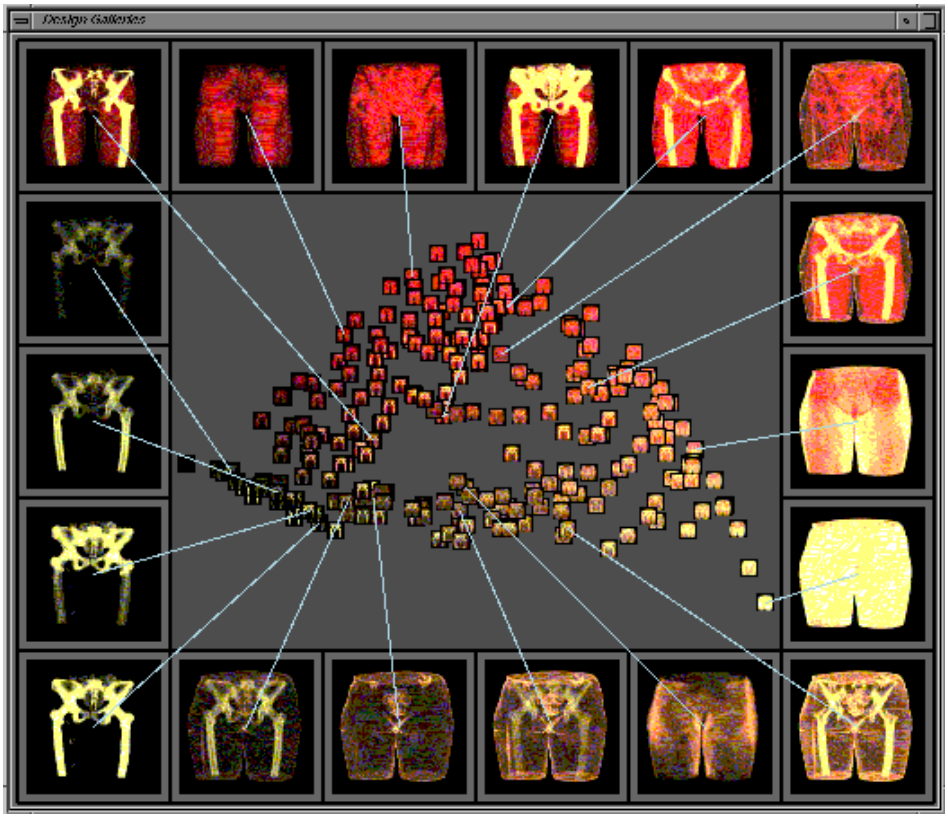


Figure 13: A DG with different opacity transfer functions.

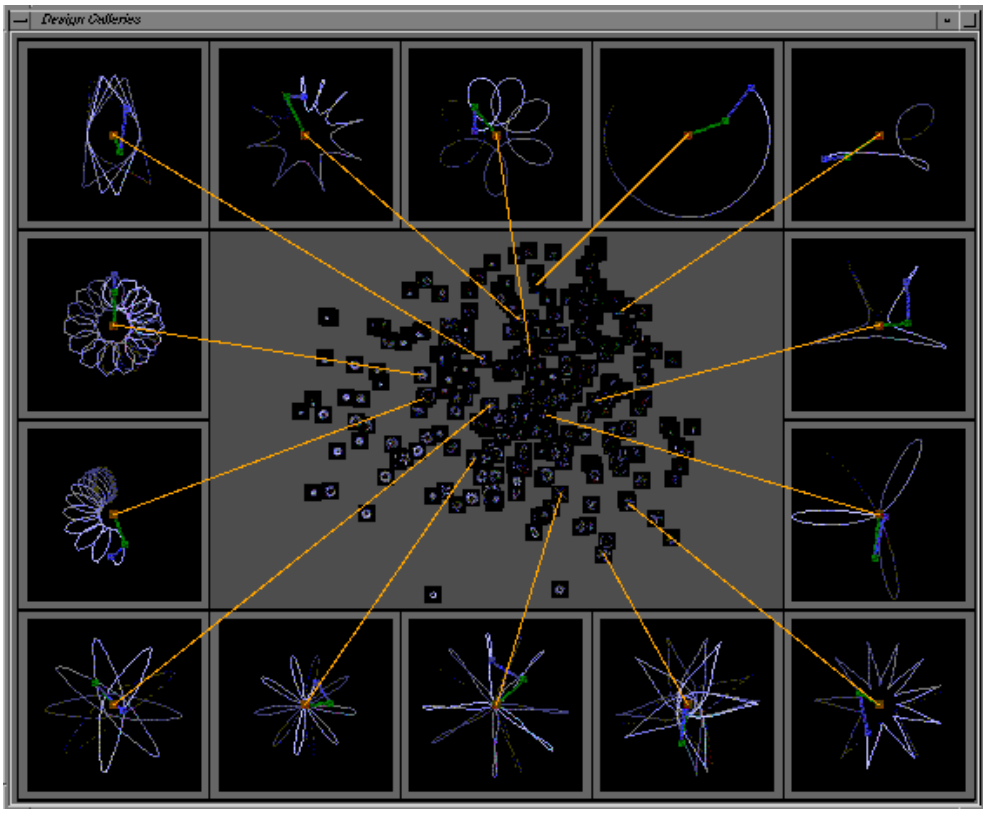


Figure 14: A DG for an actuated 2D double pendulum.

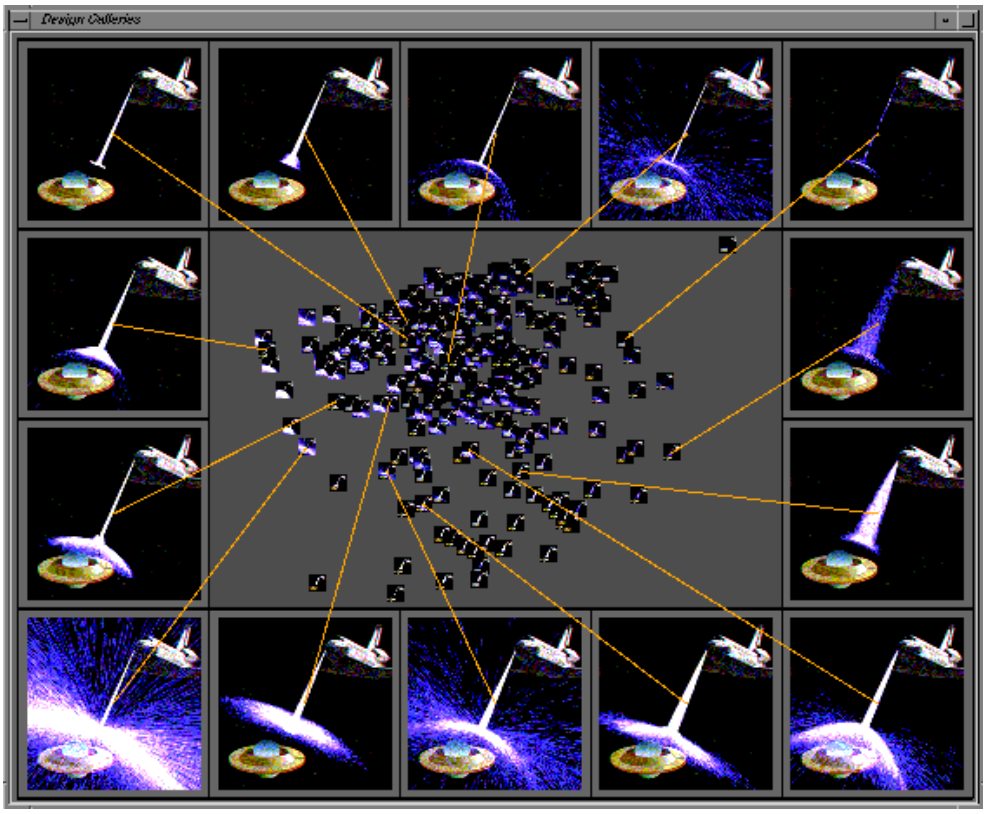


Figure 15: A DG for a particle system.

Salient Iso-Surface DetectionThrough Model-Independent Statistical Signatures

Shivaraj Tenginaki, Jinho Lee, Raghu Machiraju
Computer and Information Science, The Ohio State University

Volume graphics has not been accepted for widespread use. One of the inhibiting reasons is the lack of general methods for data-analysis and simple interfaces for data exploration. An error-and-trial iterative procedure is often used to select a desirable transfer function or mine the dataset for salient iso-values. Thus, simple, robust and general procedures are needed to detect salient iso-values and guide transfer-function choice. New semi-automatic methods that are also data-centric have shown much promise [1][8]. However, general and robust methods are still needed for data-exploration and analysis. In this paper, we propose general model-independent statistical methods based on central moments of data. Using these techniques we show how salient iso-surfaces at material boundaries can be determined. We provide examples from the medical and computational domain to demonstrate the effectiveness of our methods.

1. INTRODUCTION

Direct volume rendering is a key technology for the visualization of large 3D datasets from scientific or medical applications. Consider the example of visualizing a 3D Computed Tomography (CT) dataset of the Visible Man from the National Library of Medicine. It is known for instance that the bowels can be extracted when the intensity value is set at 700. The muscle can be displayed when intensity range includes the value 1010. For many datasets, one is often ignorant of the salient iso-values; rather they have to be determined. In other words one needs to divide the voxel intensity space into segments that delineate homogenous materials, if at all possible. This is akin to feature mining in sample space. Similarly, one can employ transfer functions for opacity and color that best produce material interfaces and boundaries. Salient iso-value detection and transfer function are related [8] and in this paper we will limit our discussion to the former.

Pat Hanrahan called data exploration (including transfer function design) one of the top ten problems in volume visualization in his inspiring keynote address at the Symposium on Volume Visualization '92. Recent research has focused on automatic and semi-automatic techniques for creating transfer functions and data exploration [1][2][4][5][8][10]. In the panel session on transfer function design at the Visualization'00 conference [13]three classes of techniques were identified. One of the classes included techniques that required an error-and-trial approach. The other classes contained techniques that are either image- or data-centric. Of all the techniques, it was felt that data-centric techniques held most promise. These techniques required assumptions to be made about the data [8] or that computable signature functions be obtained [1]. Image-centric methods [10] on the other hand are based on searching a large space and offer little user control. The effectiveness of error-and-trial methods rests very heavily on the expertise and intuition of the user. In any case, the need for new work that offered general solutions was felt. These methods should be at least semi-automatic giving assistance to the practitioner rather than completely automatic. Methods that eliminate the human from the exploration process are dangerous and should be avoided since they can generate images that may fulfill the expectations of the observer, but are not necessarily true to the nature of the data.

In this paper, we propose to employ data-signatures to explore data. These signatures are obtained from localized p-order central moments. The histogram is an example of the first-order central moment. There exist strong relationships between the various central moments and can be used to locate salient iso-values. The relationships between these signatures can be exploited in a manner similar to what was done between first and second derivatives in [8]. An attractive trait is that these methods are general and robust to noise. No assumption about the boundary's thickness and even it's suitability be made for a given dataset. Also, the signatures based on moments are all related in a more comprehensible manner than those used in [1] and are relatively much easier to compute. It is also our belief that such a method can lead to discrimination in noisy datasets. Finally, our proposed method is not completely automatic; it provides cues to the presence of material interfaces.

Section 2 describes previous work in salient contour extraction and transfer function design. In Section 3 we introduce various mathematical concepts and, derive and analyze our fundamental equation, the general moment equation. Section 4

describes the use of local higher order moments to detect boundaries in spatial domain and ample space. In Section 5 we present results that validate our analysis, while Section 6 offers a summary and describes future work.

2. PREVIOUS WORK

In general, the visualization process should be guided by information about the goal of the visualization, and specific information about the particular dataset in question. There exist techniques that employ a model of the desired feature. e.g, the boundary between homogeneous regions. This approach was taken by Kindlmann and Durkin [8]. An initial step in this process is the definition of a boundary. The boundary is essentially a Gaussian smoothed step function. The spatial component of the boundary is then removed by creating a 3D histogram of the data value and its first and second derivative. This histogram is very informative and the presence of sharp boundaries can be easily discerned from the projected plots of f vs. f' and f vs. f''. The number of zero-crossings in the second plot (or maximas in the first plot) essentially determines the number of interfaces. Based on analysis of this histogram, a distance function is created which tries to bridge the unintuitive space of data values to a synthetic, but intuitive, spatial domain: a signed distance to the middle of the nearest boundary. The calculation of the distance function is largely automated. Defining opacity as a function of position within a boundary becomes a more intuitive task than defining opacity as a function of data value. The success of this method is limited when the boundary model is not Gaussian and far from being ideal. Other notable work has been conducted by Sato et al [14]. They define a gradient based measure for certain shapes.

What information about the dataset can one exploit? Bajaj and his associates devised the contour spectrum which can guide the selection of isovalues for contouring [1]. A 1D plot of geometric and topological characteristics or signatures (volume and gradient integral) are plotted against the function value. A viable selection of salient iso-values is thus facilitated. The contour spectrum technique does not employ any particular model for the boundary interface. Rather, they exploit geometrical and/or topological properties of the volume.

Statistical techniques are gaining popularity in data-mining applications. Only a few reported statistics-based methods exist for volume data-analysis and exploration. In [3][9] statistical inference methods (e.g. Bayesian) are employed to determine the material density of each voxel. Multiscale statistical techniques have also been proposed by Yoo [16]. Histograms and local higher order moments have been used in computer vision and image analysis research to describe shapes of objects. The histogram certainly has some potential as described in [8]. However, not much has been explored with histograms as a statistical entity to facilitate volumetric data-analysis.

Instead of tweaking input parameters another approach such as inverse design can be used for data-analysis. Such methods have been again reported in data-mining literature. Inverse design is a general paradigm for computer-aided design of graphics, where the user supplies an objective function over the output values. This function will generate a high score (say) for desirable output values, and a low score for other output values. The computer then searches for a set of input parameters that will maximize the objective-function score. He et al. implemented a genetic algorithm to seek transfer functions with limited success [5]. A primary reason for failure is the difficulty of specifying a suitable objective function. A more viable approach entails a search that creates an ensemble of unique images which are then displayed in such a fashion that the differences are easily discerned. Embodying this approach is the Design Galleries approach that was used to explore parameter spaces for a host of graphics and animation applications [10]. Generated images are used in an evolutionary search algorithm in the parameter space. On termination, the chosen images are generated by an unique set of transfer functions. The problem with inverse design methods are that they are automatic in nature and do not allow for much control to be exercised on the process.

3. STATISTICAL SIGNATURES OF DATA

In this section we first define the concepts related to higher order moments. We then introduce our definition of a boundary, and proceed to define local higher order moments. This is followed by a derivation of the general moment equation, and its analysis. We use results of this section to present techniques for detecting boundaries in Section 4.

3.1 Higher Order Moments (HOMs)

Higher order moments (HOMs) are model-independent statistical estimators of central tendency of a data distribution, i.e. they measure the tendency of a distribution to cluster around some particular value [15]. Usually, the value around which clustering of a distribution, $x_1, x_2, x_3, \ldots, x_N$, is measured is the *mean (M)* of the distribution, given by:

$$M = \frac{1}{N}\sum_{j=1}^{N} x_j \tag{1}$$

The *higher order moment* of the same distribution is defined as:

$$m_k = \frac{1}{N}\sum_{j=1}^{N} (x_j - M)^k \tag{2}$$

where, m_k is the *higher order moment* of order *k*. We can see that are infinitely many HOMs. The second moment evaluates local variance, while the third and fourth moments evaluate the local skewness and kurtosis. For a symmetric set of samples, the skewness, m_3, evaluates close to zero, while, for positively skewed samples, the histogram of the sample will have a heavy tail on the right. A sample set obtained from the *Normal* distribution will have kurtosis, $m_4 - 3$, close to zero. When a sample is more dense than that derived from a normal distribution, then the sample's likely distribution is ascribed a negative kurtosis value. Otherwise it is positive. Thus, one can completely characterize the shape of any function if infinite number of central moments, are known. HOMs are used in nonlinear signal processing for signal and image estimation [11].

3.2 Boundary Model and Local Higher Order Moments

The methods presented in this paper for following model of boundary (though, all the discussion below is for R^2, it generalizes easily to higher dimensions). Consider a region R^2 of size *w x w* in a dataset, centered at a point *P(x,y)*, see Figure 1. At boundary, the sample value of the points in the dataset changes from C_1 to C_2. Here we assume that R^2 is small enough to contain just one boundary. As a consequence of this assumption, in the presence of a boundary, there exist exactly two homogenous sub-regions of C_1 and C_2 in R^2.It follows that absence of a boundary implies single homogenous region in R^2. Similar mixture model was employed in [9]. However, we employ much simpler analysis.

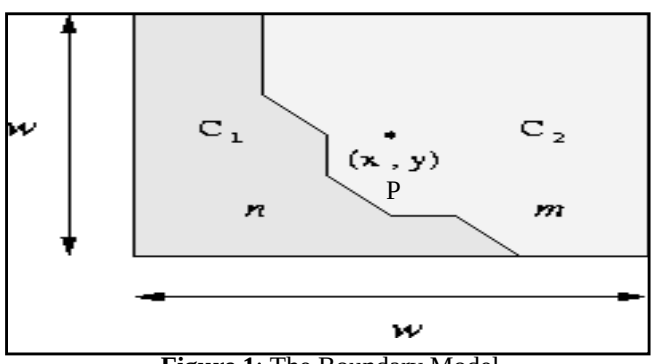


Figure 1: The Boundary Model

Since, the size of R^2 is *w x* w, it follows that there are w^2 "particles" or sample points in R^2. We call the particles with sample value C_2 as *boundary practices*, since they are introduced by the presence of the boundary. Further, let *m* be the number of such particles. Also, let *n* be the number of "particles" or sample points with sample value C_1 in R^2, it follows that $n + m = w^2$. We make following observations: if $m = 0$ or $n = 0$, which means that there is no boundary in R^2, and we classify such regions as a *non-boundary region*, else if $m > 0$ and $n>0$ then there exists a boundary in R^2, and we classify such regions as a *boundary region.*

Further, if *n* is less than *m*, then the point *P* lies to the "left" of the boundary in R^2, and we classify the region as a *left-boundary region*. If *n* is greater than *m*, then point *P* lies to the "right" of the boundary, and we classify the region as a *right-boundary region*. If *n* equal to *m*, then the point *P* lies "on" the boundary in R^2, and we classify the region as an *on-boundary region*.

For purpose of further analysis, we introduce the concept of *local higher order moments (LHOMs)*. LHOMs are HOMs calculated over the distribution of the sample values in a region R^2 of a given dataset (compared to HOMs which are defined over the distribution of the sample values in a dataset). Associated is also the concept of *local mean* which is the *mean* of sample values in a region R^2 of a given dataset. The equation of *local mean (LM)* could be derived from Equation 1 as:

$$LM = \frac{1}{w^2}\sum_{R^2} x \qquad , \forall x \quad \varepsilon \quad R^2 \tag{3}$$

The LHOM of a region R^2 is thus defined as:

$$m_k = \frac{1}{w^2}\sum_{R^2} (x - LM)^k \qquad , \forall x \quad \varepsilon \quad R^2 \tag{4}$$

where, m_k is the LHOM of order k. It should be noted any region in a distribution has infinitely many LHOMs. We now show the derivation of the general moment equation based on the concepts introduced above.

3.3 Derivation of the General Moment Equation

In this section we proceed to simplify Equation 4, based on our model of boundary. Using the model of boundary defined previously, the expression for local mean, *LM*, reduces to:

$$LM = \frac{(nC_1 + mC_2)}{(n+m)} \tag{5}$$

Further, the expression LHOM of *k-th* order reduces to:

$$\begin{aligned} m_k &= \frac{1}{(n+m)}\left\{ n\left[C_1 - \frac{(nC_1 + mC_2)}{(n+m)}\right]^k + m\left[C_2 - \frac{(nC_1 + mC_2)}{(n+m)}\right]^k \right\} \\ &= \frac{1}{(n+m)}\left\{ n\left[\frac{m(C_1 - C_2)}{(n+m)}\right]^k + m\left[\frac{n(C_2 - C_1)}{(n+m)}\right]^k \right\} \\ &= \frac{(C_2 - C_1)^k}{(n+m)^{k+1}}(mn^k + (-1)^k nm^k) \end{aligned} \tag{6}$$

Let $\Delta = C_2 - C_1$, and substitute $n + m = w^2$, in the above equation. This yields the general moment equation:

$$m_k = \frac{\Delta^k}{w^{k+1}}(mn^k + (-1)^k nm^k) \tag{7}$$

Equation 7 can be rewritten as:

$$m_k = \frac{\Delta^k}{w^{k+1}} f(m, n) \tag{8}$$

where, *f(m,n)* is defined as the *distribution function*. It determines behavior of the *k-th* order LHOM in boundary regions. We now analyze Equation 7 for moments of various order and examine various LHOMs for boundary presence.

3.4 Analysis of the General Moment Equation

If $\Delta \neq 0$ then R^2 is a *boundary* region (else C_1 would be same as C_2, and Δ would be zero). From Equation 7, it is clear that for any non-zero Δ all even LHOMs cannot be non-zero, i.e. a *boundary* region implies all non-zero even LHOMs. Conversely, if any even LHOMs is nonzero in R^2, then Δ cannot be zero in R^2. That is all non-zero even LHOMs imply that the associated region is a *boundary* region. The same argument holds for non-even LHOMs, except that a non-even LHOM will be zero if the associated region is an *on-boundary* region. Since *non-boundary* regions have zero LHOMs, we confine our interest to *boundary* regions.

Second Order LHOM: The general moment equation for second order LHOM, reduces to:

$$m_2 = \frac{\Delta^2}{(w^2)^2}(mn) \tag{9}$$

It can be shown that this equation has one maxima at $m=n=w^2/2$, i.e when the associated region is an *on-boundary* region. The value of m_2 at the location of this maxima is, $\frac{\Delta^2}{4}$, which is independent of the size of the associated region and the actual distribution in the region.

Third Order LHOM: The general moment equation for the third order LHOM reduces to:

$$m_3 = \frac{\Delta^3}{(w^2)^3}mn(n-m) \tag{10}$$

It can be shown that this equation has one maxima at, $m = \frac{w^2}{2}\left(1-\frac{1}{\sqrt{3}}\right)$, and one minima at, $m = \frac{w^2}{2}\left(1+\frac{1}{\sqrt{3}}\right)$. The value of m_3 at the location of the minima is, $\frac{\Delta^3}{6\sqrt{3}}$, and its value at the location of the minima is, $-\frac{\Delta^3}{6\sqrt{3}}$. Also, this equation is zero at $m=n=w^2/2$, i.e when the associated region is an *on-boundary* region. Here again, it is to be noted that the value of m_3 at the location of maxima and minima is independent of the size of the associated region and the actual distribution in the region.

Fourth Order LHOM: The general moment equation for the fourth order moment reduces to:

$$m_4 = \frac{\Delta^4}{(w^2)^4}mn(w^4-3mn) \tag{11}$$

It can be shown that this equation has two maximas. One maxima is at, $m = \frac{w^2}{2}\left(1-\frac{1}{\sqrt{3}}\right)$, and the second maxima is at, $m = \frac{w^2}{2}\left(1+\frac{1}{\sqrt{3}}\right)$. The value of m_4 at the location of these maximas is, $\frac{\Delta^4}{12}$. Also, this equation has one local minima at $m=n=w^2/2$, i.e when the associated region is an *on-boundary* region. The value of m_4 at the location of this minima is, $\frac{\Delta^4}{16}$. Here again, it is to be noted that the value of m_4 at the location of maximas, and at the location of minima is independent of the size of the associated region and the actual distribution in the region.

Skew: is a non dimensional quantify, it characterizes the degree of asymmetry of a distribution around its mean. It is defined as:

$$S = \frac{m_3}{(m_2)^{3/2}} \tag{12}$$

It can be further shown that the general moment equation for *skew* in a region R^2 reduces to:

$$S = \frac{(n-m)}{\sqrt{nm}} \tag{13}$$

Also, it can be shown that this equation is monotonically decreasing, and *skew* is zero at $m=n=w^2/2$, i.e when the associated region is an *on-boundary* region. It follows that *left-boundary* regions have positive *skew*, and *right-boundary* regions have negative *skew.*

Kurtosis: is a non-dimensional quantity which measures the relative peakness or flatness of a distribution. It is defined as:

$$K = \frac{m_4 - 3m_2^2}{m_2^2} \tag{14}$$

It can be shown that the general moment equation for *kurtosis* in a region R^2 reduces to:

$$K = \frac{w^4}{mn} - 6 \tag{15}$$

It can shown that this equation has a minima at $m=n=w^2/2$, i.e when the associated region is an *on-boundary* region. The value of *kurtosis* at the location of this minima is -2. It is important that for the *on-boundary* regions the value of *kurtosis* is a constant.

Figure 2, shows the plots of the *distribution function* corresponding to the various LHOMs. The plots confirm the analysis presented above for the second, third, and fourth order LHOMs, and also for *skew* and *kurtosis*.

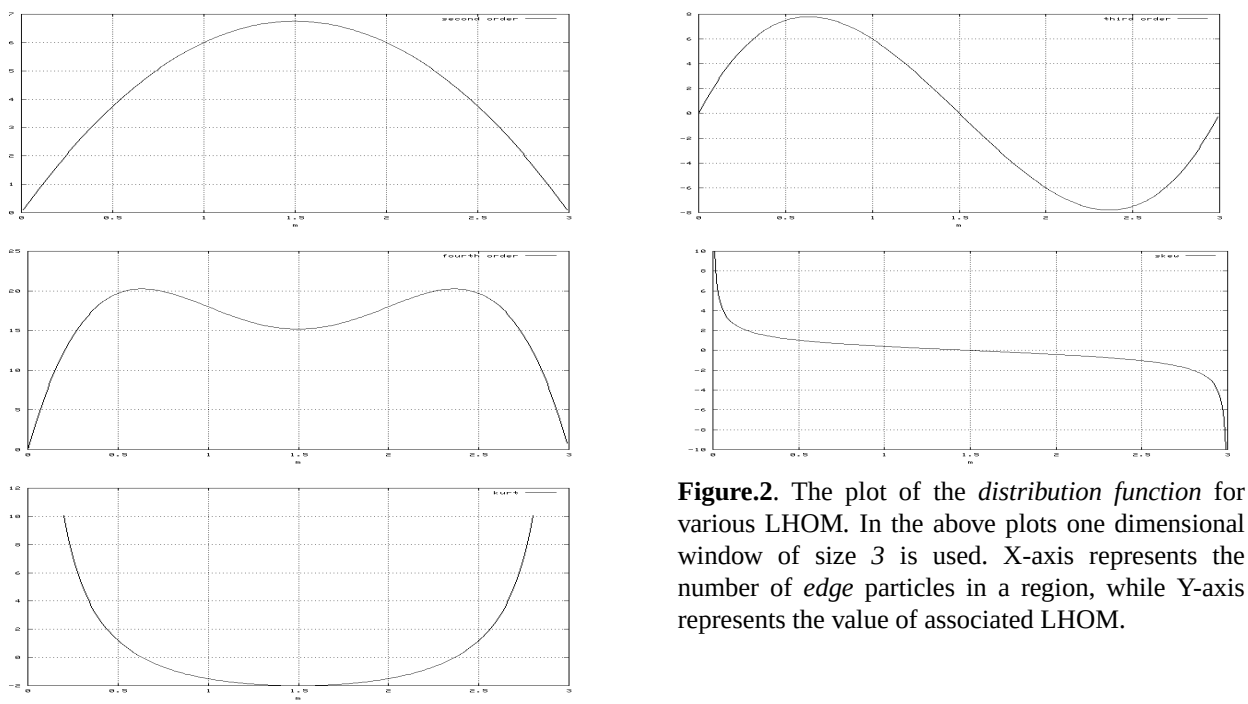


Figure.2. The plot of the *distribution function* for various LHOM. In the above plots one dimensional window of size *3* is used. X-axis represents the number of *edge* particles in a region, while Y-axis represents the value of associated LHOM.

4. BOUNDARY DETECTION USING HOMS

From the above analysis and plots in Figure 2, we know that if a region for which we are calculating LHOM is an *on-boundary region*, then all of the following hold: second order LHOM is locally maximum, third order LHOM is locally zero, fourth order LHOM is locally minimum, skew has a zero crossing, and *Kurtosis* has a constant minima of -2. Using any of these criteria it is easy to detect *on-boundary* regions in a given dataset.

Figure 3, shows the plots for the LHOMs for a step function, from these plots it is clear thar the *on-boundary regions*, occur at the location of boundaries in the original signal. Since, we can use LHOMs to detect *on-boundary* regions, we can indirectly obtain the spatial location of boundaries in a dataset. In the graph of fourth order LHOM we see just

a single maxima, instead of a minima located between the two maximas. This is probably because due to the closeness of the maximas and the minima they appear merged when plotted.

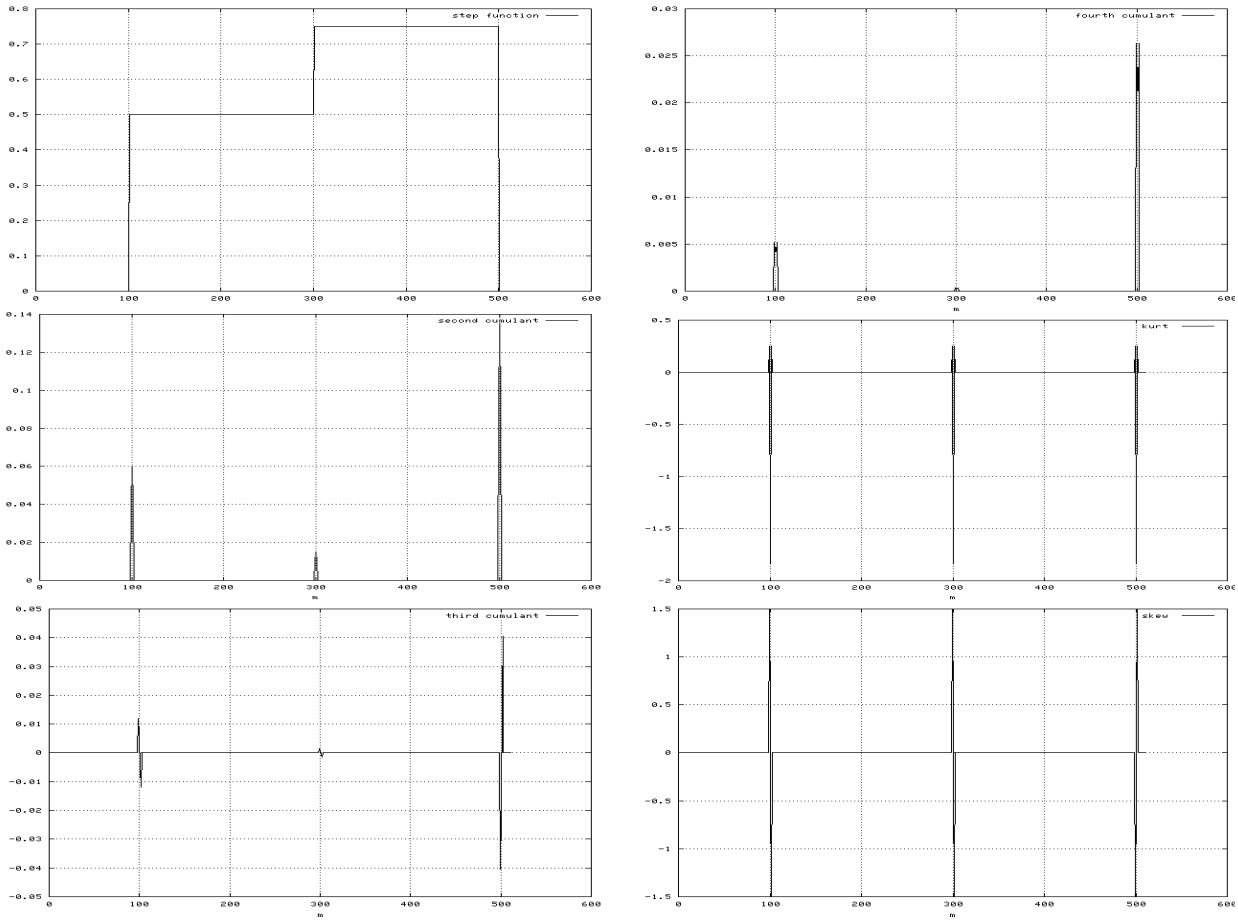


Figure 3: LHOM plots for a step function.

For detection of salient iso-values in a dataset we need a method for obtaining boundaries in sample space rather than in spatial space. From the previous analysis we have observed the behavior of LHOMs in the spatial domain in the neighborhood of boundary. If we obtain a scatter plot of a LHOM *vs.* the sample values in for all possible regions in a dataset, we expect to see similar behavior as in the graph of the LHOM in the spatial domain. This follows from exactly same arguments offered by Kindlmann and Durkin in[8].Figure 4 illustrates this point.

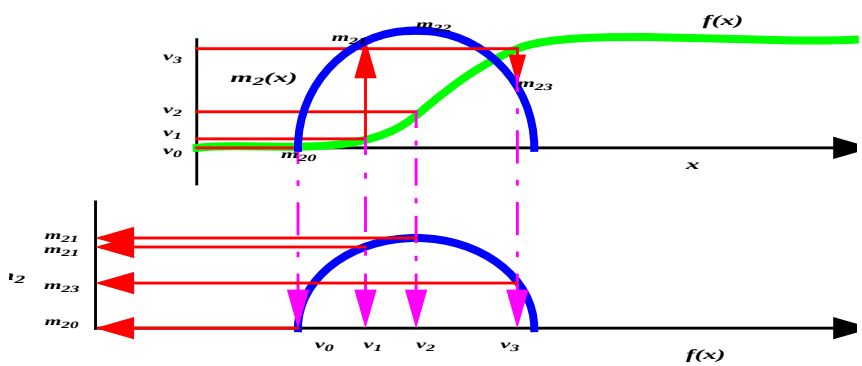


Figure 4: The relation between m_2 vs. x, and m_2 vs. $f(x)$.

5. RESULTS

Figure 5, shows the scatter plots for LHOMs *vs.* sample values for CT Tooth dataset. The width w of each region R^3 is *5.* Values corresponding to all the possible regions, i.e. regions associated with all the sample values in the dataset are considered. The shape of the scatter plots justifies the reasoning given in previous section for salient iso-surface detection using local central moments. We see from the plots that there are three distinct boundaries. These are in neighborhood of 200, 600, and 1100. The iso-surfaces corresponding to these boundaries are shown in Figure 6. Figure 7, shows *skew* scatter plot for the CT Head dataset, here again our methodology predicts three distinct boundaries in the neighborhood of 600, 900, and 3500. The iso-surfaces corresponding to these boundaries are shown in the same figure. It is clear that the first iso-value represents the skin, the second skull, and third teeth. Figure 8, shows the result of our analysis for computational dataset of flow over a delta wing. Here again we see that boundaries are in the range of 800, and 1000. Iso-surface associated with an iso-value in this range is also shown in the same figure. The shape of the object, and the shape of the vortex and the shock in the flow are clearly visible.

6. SUMMARY AND FUTURE WORK

We presented a method to explore volume data for salient iso-values. We employ statistical signatures based on localized central moments. We relate these localized moments to presence of boundaries. Later we show the effectiveness of boundary and salient iso-value detection through examples. Future work includes MRI datasets which yield noisy LHOM plots. Denoising techniques are needed for further use. Also we wish to exploit the relationship of moments to boundaries to create simple yet tangible interfaces for volume exploration.

7. ACKNOWLEDGMENTS

The authors wish to thank Joe Marks, Hanspeter Pfister of Mitsubishi Electric Research Laboratories (MERL) for support. The work has been supported by grants from a NSF CAREER award, MERL and DoD.

8. REFERENCES

[1] C. L. Bajaj, V. Pascucci, and D. R. Schikore, “The Contour Spectrum,” Proceedings of Visualization '97, pp. 167-175.

[2] Y.-K. Chang, A. P. Rockwood, and Q. He, “Direct rendering of freeform volumes,” Computer-aided Design, 27(7), 1995, pp. 553-558.

[3] R. Drebin, L. Carpenter, and P. Hanrahan, “Volume Rendering,” Computer Graphics, Vol. 22 No. 4, August 1988, pp. 65 - 74.

[4] S. Fang, T. Biddlecome, and M. Tuceryan, “Image-based transfer function design for data exploration in volume visualization,” Proceedings of Visualization’98, pp. 319-326.

[5] T. He, L. Hong, A. Kaufman, and H. Pfister, “Generation of transfer functions with stochastic search techniques,” Proceedings of Visualization '96, 1996, pp. 227-234.

[6] J. Hladùvka, A. König, and E. Gröller, “Curvature-Based Transfer Functions for Direct Volume Rendering,” Proceedings of 16th Spring Conference on Computer Graphics, May 2000, Comenius University, Brataslava.

[7] A. K. Jain, *Image Processing*, Prentice-Hall, 1986.

[8] G. Kindlmann and J. W. Durkin, “Semi-automatic generation of transfer functions for direct volume rendering,” Proceedings of the 1998 IEEE Symposium on Volume visualization, 1998, pp. 79-86.

[9] D. H. Laidlaw, K. W. Fleischer, and A. H. Barr, “Partial-volume Bayesian classification of material mixtures in MR volume data using voxel histograms, “IEEE Transactions on Medical Imaging, Vol. 17, No. 1, Feb. 1998, pp.

74-86.

[10] J. Marks, B. Andalman, P.A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, "Design galleries: a general approach to setting parameters for computer graphics and animation," Proceedings of Siggraph'97, Pages 389 - 400.

[11] C. L. Nikias, and A. P. Petropulu, *Higher-Order Spectra Analysis:A Nonlinear Signal Processing Framework*, PRT Prentice Hall, New Jersey.

[12] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The VolumePro Real-Time Ray-Casting System," Proceedings of SIGGRAPH 99, p. 251-260, Los Angeles, CA, August 1999.

[13] H. Pfister, "The Transfer Function Bake-Off," Panel, Proceedings of Visualization'00, October 2000.

[14] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis, "Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering," IEEE Transactions on Visualization and Computer Graphics, Vol. 6, No. 2, April/June 2000, pp. 160-180.

[15] A. Stuart, and J.K. Ord, *Kendall's Advanced Theory of Statistics*, 5th ed. Griffin and Co.,London.

[16] T. S. Yoo, Image Geometry Through Multiscale Statistics. Ph.D. Dissertation, University of North Carolina at Chapel Hill, Department of Computer Science, 1996.

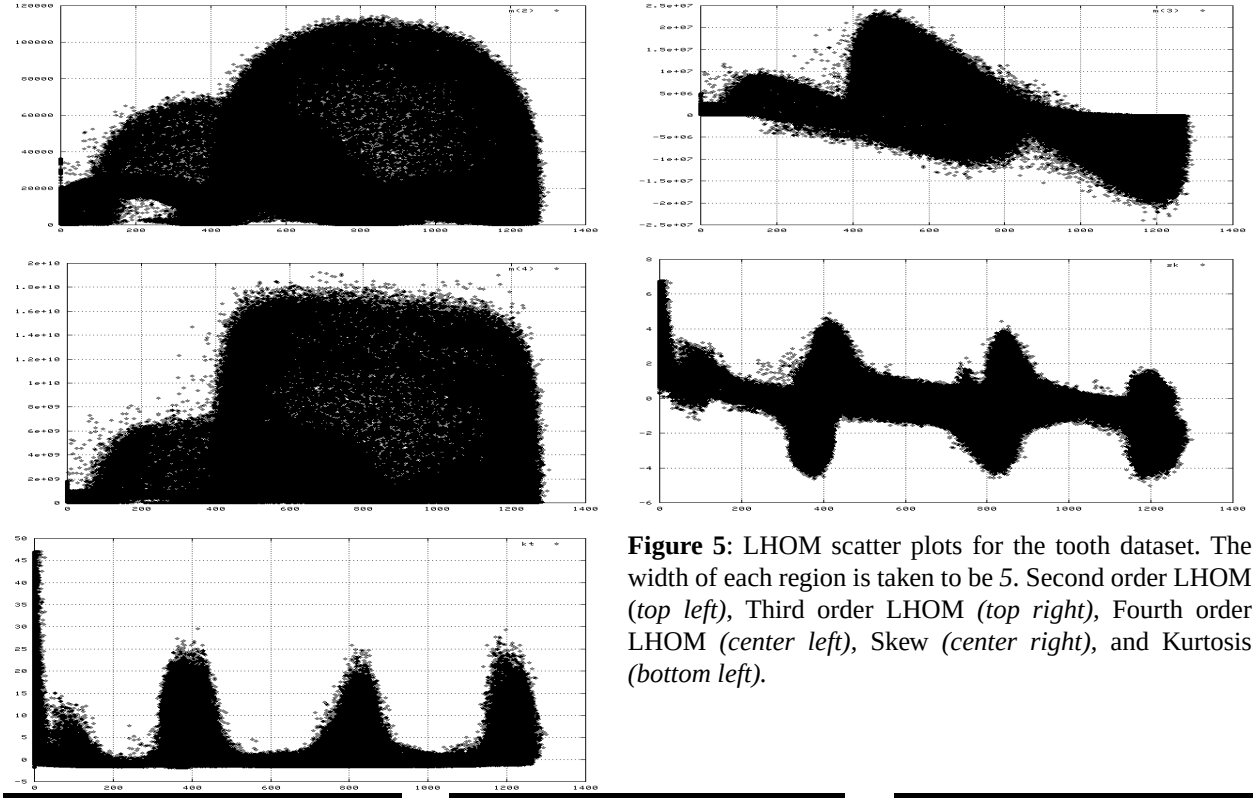


Figure 5: LHOM scatter plots for the tooth dataset. The width of each region is taken to be *5*. Second order LHOM (*top left)*, Third order LHOM *(top right)*, Fourth order LHOM *(center left)*, Skew *(center right)*, and Kurtosis *(bottom left).*

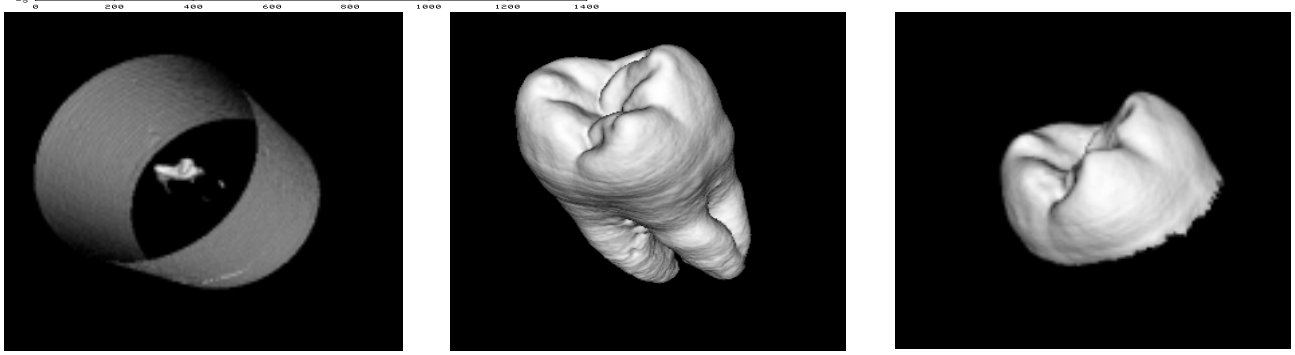


Figure 6: Iso-surfaces associated with the three boundaries predicted above in Figure 5. *(left)* 200, *(center)* 600, and *(right)* 1100

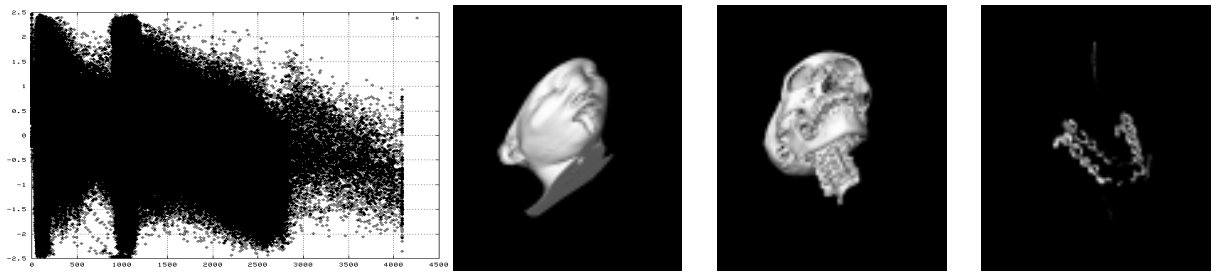


Figure 7: *(far left) skew* plot for the head dataset and iso-surfaces predicted by the plot: *(left)* 600, *(center)* 900, and *(right)* 3500. The width of each region is *3.*

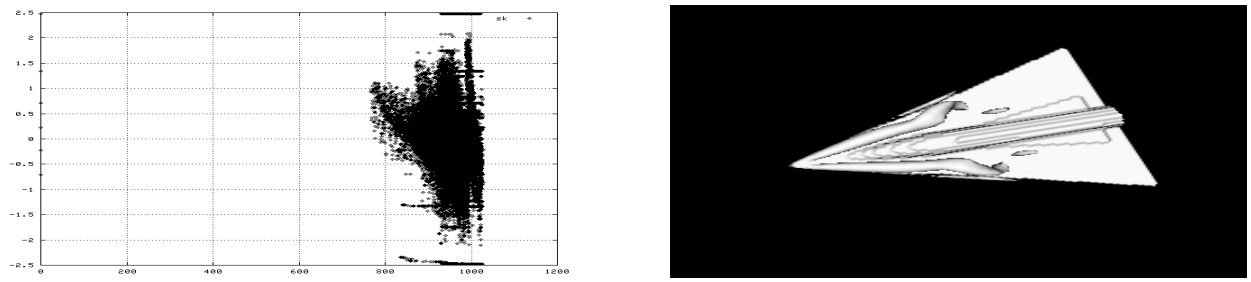


Figure 8: *(left) skew* plot for the delta wing dataset, and *(right)* iso-surface for isovalue of 900. Region width is *3*.

References - Image Processing for Graphics and Visualization

Raghu Machiraju, The Ohio State University

1. Optical Models and Rendering Integral

[1] J. Kajiya and B. Von Herzen, "Ray Tracing Volume Densities," Computer Graphics, Vol. 18 No. 3 (July 1984) pp. 165 - 174.

[2] N. Max, "Efficient Light Propagation for Multiple Anisotropic Volume Scattering," in "Photorealistic Rendering Techniques" (G. Sakas, P. Shirley, and S. Mu"ller, eds.) Springer Verlag, Heidelberg (1995) pp. 87 - 104.

[3] N. Max,"Optical Models for Direct Volume Rendering," IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 2 (1995) pp. 99 - 108.

[4] K. Novins and J. Arvo, "Controlled Precision Volume Rendering,"Proceedings of 1992 Workshop on Volume Visualization, Boston, October 1992, ACM Order No. 429922, pp. 83 - 89.

[5] P. Williams and N. Max, "A volume density optical model," Proceedings of 1992 Workshop on Volume Visualization, Boston, October 1992, ACM Order No. 429922, pp. 61-68.

[6] P. Sabella, "A rendering algorithm for visualizing 3D scalar fields," ComputerGraphics, Vol. 22 No. 4 (August 1988) pp. 51 - 58.

[7] P. Shirley and A. Tuchman "A Polygonal Approximation to Direct Scalar Volume Rendering," Computer Graphics, Vol. 24 No. 5 (November 1990) pp. 63 - 70.

2. Volume Graphics

[8] B. Cabral, N. Cam, and J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," Proceedings of the 1994 Symposium on Volume Rendering, ACM Press, (1995) pp. 91 - 98.

[9] E. Catmull and A .R. Smith, "3-D transformations of images in scanline order,"Computer Graphics Vol. 14, No. 3 (1980, Siggraph '80 Proceedings) pp. 279 - 285.

[10] R. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," Computer Graphics, Vol. 22 No. 4 (August 1988) pp. 65 - 74.

[11] M. Levoy, "Display of surfaces from volume data," IEEE CG&A Vol. 8, No. 3 (May 1988) pp. 29 - 37.

[12] P. Lacroute and M. Levoy, "Fast volume rendering using shear-warp factorization of the viewing transformation," ACM Computer Graphics Proceedings, Annual Conference Series (1994) pp. 451 - 458.

[13] N. Max, P. Hanrahan, and R. Crawfis "Area and volume coherence for efficient visualization of 3D scalar functions," Computer Graphics, Vol. 24 No. 5.

[14] M. Meissner, J. Huang, D. Bartz, K. Mueller, R. Crawfis, "A practical comparison of popular volume rendering algorithms," Proceedings of 2000 Symposium on Volume Rendering, October, 2000.

[15] T. Totsuka and M. Levoy, "Frequency Domain Volume Rendering," ACM Computer Graphics Proceedings, Annual Conference Series (1993) pp. 271-278.

[16] C. Upson and M. Keeler, "V-BUFFER: Visible Volume Rendering,"Computer Graphics, Vol. 22, No. 4 (August 1988) pp. 59 - 64.

[17] L. Westover, "Footprint evaluation for volume rendering," Computer Graphics,Vol. 24 No. 4 (August 1990) pp. 367 - 376.

[18] R. Yagel, D. Reed, A. Law, P-W. Shih, and N. Shareef, "Hardware assisted volume rendering of unstructured grids by incremental slicing," Proceedings of the 1996 Symposium on Volume Rendering, IEEE Computer Society Press, (1996)pp. 55 - 62.

3. Function and Derivative Reconstruction

[19] M. J. Bentum, B. Lichtenbelt, and T. Malzbender, "Frequency Analysis of Gradient Estimators in Volume Rendering", IEEE Transactions on Visualization and Computer Graphics, 2(3):242-254, September 1996.

[20] I. Carlbom, "Optimal filter design for volume reconstruction and visualization",Proceedings of IEEE Visualization93, October 1993.

[21] S. C. Dutta Roy and B. Kumar, "Digital Differentiators", Handbook of Statistics, N. K.Bose and C. R. Rao eds., vol. 10: 159-205, 1993.

[22] M. E. Goss, "An Adjustable Gradient Filter for Volume Visualization Image Enhancement", Proceedings of Graphics Interface'94, pp. 67-74, Toronto, Ontario, 1994.

[23] R.G. Keys, "Cubic Convolution Interpolation for Digital ImageProcessing", IEEE Transactions on Acoustics, Speech, and SignalProcessing, ASSP-29(6), pp. 1153-1160, December 1981.

[24] R. K. Machiraju and R. Yagel, "Reconstruction Error Characterization and Control: ASampling Theory Approach", IEEE Transactions on Visualization and Computer Graphics, 2(4):364-376, December 1996.

[25] S.R. Marschner and R.J. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering", Proceedings of Visualization `94, pp. 100-107, October 1994.

[26] D.P. Mitchell and A.N. Netravali, "Reconstruction Filters in Computer Graphics",Computer Graphics, 22(4), pp. 221-228,August 1988.

[27] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering", Proceedings of the 1996 Symposium on Volume Visualization, pp. 71-78, October 1996.

[28] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "A Comparison Of Normal Estimation Schemes", Proceedings of IEEE Conference on Visualization 1997, pp. 19-26, October 1997.

[29] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "Evaluation and Design of Filters Using a Taylor Series Expansion", IEEE Transactions on Visualization and Computer Graphics, ITVCG 3(2): 184-199, June 1997.

[30] T. Möller, K. Mueller, Y. Kurzion, R. Machiraju, and R.Yagel, "Design of Accurate and Smooth Filters for Function and Derivative Reconstruction", Proceedings of the 1998 Symposium on Volume Visualization, pp. 143-151, October 1998.

[31] K. Mueller, T. Möller, J. E. Swan II, R. Crawfis, N. Shareef, and R. Yagel, "Splatting Errors and Antialiasing", IEEE Transactions on Visualization and Computer Graphics, ITVCG 4(2): 178-191, June 1998.

4. Transfer Function Design

[32] C. L. Bajaj, V. Pascucci, and D. R. Schikore, "The Contour Spectrum," Proceedings of Visualization '97, pp. 167-175.

[33] Y.-K. Chang, A. P. Rockwood, and Q. He. "Direct rendering of freeform volumes, " Computer-aided Design, 27(7), 1995, pp. 553-558.

[34] R. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," ComputerGraphics, Vol. 22 No. 4, August 1988, pp. 65 - 74.

[35] S. Fang, T. Biddlecome, and M. Tuceryan, "Image-based transfer function design for data exploration in volume visualization," Proceedings of Visualization’98, pp. 319-326.

[36] T. He, L. Hong, A. Kaufman, and H. Pfister, "Generation of transfer functions with stochastic search techniques," Proceedings of Visualization '96, 1996, pp. 227-234.

[37] J. Hladùvka, A. König, and E. Gröller, "Curvature-Based Transfer Functions for Direct Volume Rendering," {roceedings of 16th Spring Conference on Computer Graphics, May 2000, Comenius University, Brataslava.

[38] G. Kindlmann and J. W. Durkin, "Semi-automatic generation of transfer functions for direct volume rendering," Proceedings of the 1998 IEEE Symposium on Volume visualization, 1998, pp. 79-86.

[39] D. H. Laidlaw, K. W. Fleischer, and A. H. Barr, "Partial-volume Bayesian classification of material mixtures in MR volume data using voxel histograms, " IEEE Transactions on Medical Imaging, Vol. 17, No. 1, Feb. 1998, pp. 74-86.

[40] J. Marks, B. Andalman, P.A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber, "Design galleries: a general approach to setting parameters for computer graphics and animation," Proceedings of Siggraph’97, Pages 389 - 400.

[41] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis, "Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering," IEEE Transactions on Visualuzation and Computer Graphics, Vol. 6, No. 2, April/June 2000, pp. 160-180.

5. Wavelets and Multiscale Analysis

[42] G. Belkyin, "On the Representation of Operators in Bases of Compactly Supported Wavelets", SIAM J. on Numerical Analysis, 6(6): 1716-1740.

[43] G-P. Bonneau, S. Hahmann, and G. M. Nielson, "BLaC-Wavelets: A Multiresolution Analysis with Non-Nested Spaces, " IEEE Visualization '96, October 1996, pp. 43-48.

[44] R. R. Coifman and M. V. Wickerhauser, "Wavelets and adapted waveform analysis",Wavelets: Mathematics and Applications, Edited by J. J. Benedetto and M. W. Frazier, CRC Press, Boca Raton, FL, 1994, pp. 399-423.

[45] I. Daubechies, Ten Lectures on Wavelets, CBMS-NSF Regional Conference Series inApplied Mathematics, SIAM, Philadelphia, PA, 1992.

[46] G. Fernández, S. Periaswamy, and W. Sweldens, "LIFTPACK: A Software Package for Wavelet Transforms using Lifting," In M. Unser, A. Aldroubi, and A. F. Laine, editors, Wavelet Applications in Signal and Image Processing IV, pp. 396-408, Proc. SPIE 2825, 1996.

[47] A. Fournier, Wavelets and Their Applications in Computer Graphics, Course Notes, Siggraph '95.

[48] I. Guskov, W. Sweldens and P. Schröder, "Multiresolution Signal Processing for Meshes, " Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series, pp. 325-334 (August 1999, Los Angeles, California).

[49] M. Holschneider, Wavelets: An Analysis Tool, Oxford University Press, 1995.

[50] B. Jawreth and W. Sweldens, "An Overview of Wavelet Based MultiresolutionAnalyses," SIAM Review, September 1994, 30(3):377-410.

[51] L. Kobbelt and P. Schröder, "A multiresolution framework for variational subdivision, " ACM Transactions on Graphics, 17(4), October 1997, pp. 209-237.

[52] S. Mallat, "A Theory for Multiresoultion Signal Decomposition: The WaveletRepresentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 11, No. 7, July 1989, pp. 674-693.

[53] G. Strang and T. Nguyen, Wavelets and Filter Banks, Wellesley-Cambridge Press, 1996.

[54] E. J. Stollnitz, T. D. DeRose and D. Salesin, Wavelets for Computer Graphics:Theory and Applications, Morgan Kaufmann, San Francisco, 1996.

[55] W. Sweldems,"Wavelets: What Next ?," Proceedings of IEEE, Vol. 84, No. 4,pp.680-685, 1996.

[56] W. Sweldens, "The Lifting Scheme: A custom design construction of bi-orthogonal wavelets," Appl. Comput. Harmon. Anal., Vol. 2, No.3, pp. 186-200, 1996.

[57] W. Sweldens, "Wavelets and the lifting scheme: {A} 5 minute tour," Z. Angew. Math. Mech. Vol. 76 (Suppl. 2), 1996, pp. 41-44.

[58] W. Sweldens, The lifting scheme: A construction of second generation wavelets,SIAM J. Math. Anal., Vol. 29, No.2, 1997, pp. 511-546.

[59] M. Udea and S. Lodha, "Wavelets: An Elementary Intoduction and Examples," Technical Report, University of California-Santa Cruz, UCSC-CRL-94-47, January 1995.

[60] G. G. Walter, Wavelets and Other Orthogonal Systems with Applications, CRCPress, Boca Raton, FL, 1994.

[61] Y. Meyer, "Wavelets and Operators", Cambridge University Press, 1992.

[62] Y. Meyer, Wavelets: Algorithms and Applications, SIAM 1993.

6. Wavelets in Image Analysis

[63] J-P. Antoine , P. Carrette , R. Murenzi and B. Piette, "Image Analysis with Two Dimensional Continuous Wavelet Transform," Signal Processing 31(1993):241-272.

[64] M. Antonini, M. Barlaud, P. Mathieu, and Ingrid Daubechies", "Image Coding Using Wavelet Transform," IEEE Transactions on Image Processing, Vol. 1, No. 2, April 1992, pp. 205-220.

[65] C. H. Chen and G. G. Lee, "On Digital Mammogram Segmentation and Microcalcification Detection Using Multiresolution Wavelet Analysis," Graphical Models and Image Processing, 59 (5), September 1997, pp. 349-364.

[66] P. De Neve, K. Denecker, W. Philips, and I. Lemahieu, "An Advanced Color Representation for Lossy Compression of CMYK Prepress Images," Computer Graphics Forum, 19(1), March 1998, pp. 3-12.

[67] D. L. Donoho and I. M. Johnstone, Ideal Spatial Adaptation via Wavelet Shrinkage," Biometrika 81, pp. 422-455.

[68] T. Frajka, P. G. Sherwood, and K. Zeger, "Progressive Image Coding with Spatially Variable Resolution," Proceedings of International Conference on Image Processing, October 1997, pp. 53-56.

[69] J. Froment, S. Mallat, "Second Generation Compact Image Coding with Wavelets," Wavelets - A Tutorial in Theory and Applications, Chui C. K. (editor), pp. 655-678, Academic Press, 1992.

[70] A. Gaddipati, R. Machiraju and R. Yagel, "Steering Image generation using Wavelet-Based Perceptual Metric," Computer Graphics Forum (Proceedings of Eurographics `97), 16(3), August 1997, pp. 241-252.

[71] M. Gross and R. Koch, "Visualization of Multidimensional Shape and TextureFeatures in Laser Range Data Using Complex-Valued Gabor Wavelets", IEEE Transactions on Visualization and Computer Graphics, 1(1):44-59.

[72] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast Multiresolution Image Querying", Computer Graphics (Proceedings of Siggraph'95), pp. 277-286.

[73] A. Laine, S. Schuler, J. Fan, and W. Huda, "Mammographic Feature Enhancement by Multiscale Analysis", IEEE Transactions on Medical Imaging, December 1994, 13(40):725-740.

[74] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales", J. Applied Statistics, 21(2): 224--270.

[75] J. Lu and D. M. Healy, "Contrast enhancement via multiscale gradient transformations," Proceedings of the First IEEE Intl. Conf. on Image Processing, Austin TX, Nov. 1994, II:482-486.

[76] R. Machiraju, A. Gaddipati, and R. Yagel, "Detection and Enhancement of Scale Coherent Structures Using Wavelet Transform Products," Proceedings of the Technical Conference on Wavelets in Signal and Image Processing, SPIE Annual Meeting, Vol. 3169, San Diego, CA, July 27 - August 2, 1997, pp. 458-469.

[77] S. Mallat and L. Hwang, "Singularity Detection with Wavelets", IEEE Transactions on Information Theory, March 1992,38(2):617-643.

[78] S. Mallat and S. Zhong, "Characterization of Signals from Multiscale Edges," IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1992, 14(7):710-732.

[79] F. Ma, W. Wang, W. W. Tsang, Z. Tang, and S. Xia, "Probabilistic Segmentation of Volume Data for Visualization Using SOM-PNN Classifier," Proceedings of 1998 Volume Visualization Symposium, October 1998, pp. 71-78.

[80] A. Said and W. A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Transactions on Circuits and Systems for Video TechnologyVol. 6, No. 3, June 1996, pp. 243-250.

[81] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Transactions on Signal Processing, December 1993, Vol. 41, No. 12, pp. 3445-3462.

[82] J. M. Shapiro, "Smart Compression Using the Embedded Zerotree Wavelet (EZW) Algorithms," Proceedings of the Asilomar Conference of Signals, Systems, Computers, Nov. 1993, pp. 486-490.

[83] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable Multi-Scale-Transforms [or "What's Wrong with Orthonormal Wavelets]", IEEE Trans. Information Theory, Special Issue on Wavelets, March 1992, 38(2):587-607.

[84] P. Schroeder and W. Sweldens,"Spherical Textures: Texture Processing", Proceedings of Sixth Eurographics Workshop on Rendering, Dublin, Ireland, June 1995, pp. 252-263.

[85] J.-L. Starck, F. Murtagh, and A. Bijaoui, "Multiresolution support applied to image filtering and restoration", Graphical Models and Image Processing, 1995, 57:420-431.

[86] L. Wang and M. Goldberg, "Reduced-difference pyramid: a data structure for progressive image transmission," Opt. Eng., 28(7), 708-716, 1989.

[87] Y. Xu, J. B. Weaver, D. M. Healy, and J. Lu, "Wavelet Transform Domain Filters: A Spatially Selective Noise Filtration Technique," IEEE Transactions On Image Processing, Vol. 3, No. 6, November 1994, pp. 747-758.

[88] Witkin A., "Scale Space Filtering", Proceedings of International Joint Conferenceon Artificial Intelligence, Karlsruhe, 1983.

7. Transform Methods (including Wavelets) in Volume Graphics and Representation

[89] X. S. Du and R. J. Moorhead, "Multiresolutional Visualization of Evolving Distributed Simulations Using Wavelets and MPI," Proceedings of SPIE, Vol. 3017, SPIE/IS&T Electronic Imaging ‘97, San Jose, CA, February 1997, pp. 298-309.

[90] W. O. Cochran, J. C. Hart, and J. P. Flynn, "Fractal Volume Compression," IEEE Transactions on Visualization and Computer Graphics, Vol. 2, No. 3, December 1996, pp. 313-322.

[91] J. Fowler and R. Yagel, "Loseless Compression of Volume Data," Proceedings of 1994 Symposium on Volume Visualization, Washington DC, October 1994, pp. 43-50.

[92] M. H. Ghavamnia and X. D. Yang, "Direct Rendering of Laplacian Pyramid Compressed Volume Data", Proceedings of Visualization'95, Atlanta, GA, October 1995, pp. 192-199.

[93] M. H. Gross and L. Lippert, A. Dreger and R. Koch, "A new method to approximate the volume-rendering equation using wavelet bases and piecewise polynomials," Computers & Graphics, 19 (1), January 1995, pp. 47-62.

[94] M. H. Gross, L. Lippert, R. Dittrich and S. Häring, "Two methods for wavelet-based volume rendering," Computers & Graphics, 21(2), March 1997, pp. 237-252.

[95] B. Guo,"A multiscale model for structure-based volume rendering," IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 4, December 1995, pp. 291-301.

[96] L. Lippert, M. H. Gross, and C. Kurmann, "Compression Domain Volume Rendering for Distributed Environments, "Computer Graphics Forum, 16(3), August 1997, pp. 95-108.

[97] L. Lippert and M. H. Gross, "Fast Wavelet Based Volume Rendering by Accumulation of Transparent Texture Maps, "Computer Graphics Forum, 14(3), August 1995, pp. 431-444.

[98] I. Ihm and S. Park, "Wavelet-Based 3D Compression Scheme for Interactive Visualization of Very Large Volume Data, "Computer Graphics Forum, 18(1), March 1999, pp. 3-15.

[99] R. Machiraju, Z. Zhu, B. Fry, and R. Moorhead, "Structure Significant Representation of Computational Field Simulation Datasets," IEEE Transactions of Visualization and Graphics, Vol. 4, No 2, pp. 117-132.

[100] N. E. Miller, P. C. Wong, M. Brewster, and H. Foote, "TOPIC ISLANDS - A Wavelet-Based Text Visualization System," Proceedings of IEEE Visualization '98, October 1998, pp. 189-196.

[101] S. Muraki, "Volume data and wavelet transform," IEEE Computer Graphics andApplications, July 1993, 13(4):50-56.

[102] S. Muraki, "Multiscale {3D} Edge Representation of Volume Data by a {DOG}Wavelet," Proceedings of 1994 Symposium on Volume Visualization, October 1994, Washington D.C, pp. 35-42.

[103] G. M. Nielson, I-H. Jung, and J. Sung, "Haar Wavelets over Triangular Domains with Applications to Multiresolution Models for Flow over a Sphere," Proceedings of IEEE Visualization '97, November 1997, pp. 143-150.

[104] G. M. Nielson, I-H. Jung, and J. Sung, "Wavelets Over Curvilinear Grids," Proceedings of IEEE Visualization '98, October 1998, pp. 313-318.

[105] P. Ning and L. Hesselink, "Vector Quantization for Volume Rendering," Proceedings of 1992 Workshop on Volume Visualization, October 1992, pp. 69-74.

[106] R. Sánchez and M. Carvajal, "Wavelet Based Adaptive Interpolation for Volume Rendering," Proceedings of 1998 Volume Visualization Symposium, October 1998, pp. 127-134.

[107] P. C. Wong and R. D. Bergeron, "Multiresolution Multidimensional Wavelet Brushing," Proceedings of IEEE Visualization '96, October 1996, pp. 141-148.

[108] H. Tao and R. Moorhead, "Progressive Transmission of Scientific Data Using Biorthogonal Wavelet Transform," Proceedings of Visualization’94, October 1994, pp. 93-99.

[109] R. Westermann, "A Multiresolution Framework for Volume Rendering," Proceedings of 1994 Symposium on Volume Visualization, October 1994, pp. 51-58.

[110] R. Westermann, "Compression Domain Rendering of Time-Resolved Volume Data," Proceedings of Visualization’95, October 1995, pp.168-175.

[111] R. Westerman and T. Ertl, "A Multiscale Approach to Integrated Volume Segmentation and Rendering," Proceedings of Eurographics’97, Budapest, Hungary, September 1997,pp. 117-128.

[112] A. Trott, R. Moorhead, and J. McGinley, "Wavelets Applied to Lossless Compression and Progressive Transmission of Floating Point Data in 3-D Curvilinear Grids," Proceedings of IEEE Visualization '96, October 1996, pp. 385-388.

[113] B-L. Yeo and L. Bede,"Volume Rendering of DCT-based Compressed 3D Scalar Data",IEEE Trans. on Visualization and Computer Graphics, March 1995, 1(1):29-43.

Volume Segmentation

Terry S. Yoo
Office of High Performance Computing and Communications
The National Library of Medicine
National Institutes of Health

Ioannis Kakadiaris
Department of Computer Science
University of Houston

1 Overview

This chapter is a review of mechanisms and methods for partitioning a volume dataset into coherent regions or sub-volumes that share similar characteristics. A full description of these techniques is beyond the scope of this chapter and the reader is referred to the references cited for additional information. In addition a nuber of excellnet surveys insegmentation exist in the literature [1, 2, 3, 4, 5, 6, 7].

Challenges: Segmentation is a difficult task due to: 1) data quality (noise and sampling artifacts), and 2) variability in the shape of target structures. There exist general methods that can be applied to a variety of data, however methods specialized to particular applications often achieve better perfoemance bytaking into consideration domain-specific knowledge. Thus, selecting the most appropriate method for segmentation is a formitable task.

Discrete vs Continous segmentations: Segmenation techniques partition discrere volume data in disjoint regions. Subvoxel accuracy is particuarly important in the applications for which the structrute of interest is on the same order of magnitute as the voxel resoltion. Continusous segmenation methods (e.g., deformable models) can provide subvoxel accuarcy, however its dififuclt to validate this level of accuarcy in real data.

User Interaction: The level of user interaction required by different segmentation methods varies widely ranging from manual delineation of the interesting struucture in various slices to selection of a seed point for region growing or initial position for a deformabler model. Methods that depend on high levels of user interaction are both consuming and valnurable to inter-operator variabilities.

Post-Processing: Due to the difficulty of the task, it is often the case that a post-processing is applied to remove invalid object boundaries.

Validation: The performance of the segmentation methods is quantified by validation experiments [8]. Validation can be done by: 1) comparing the results of the method with manually outlined segmentations [9, 10], 2) physcial phantoms [11], or 3) computational phantoms [12]. Each of these validation methods presents its challenges. For example, manual segmentations could be flawed, or physical phantoms that accurately depict the properties of the target structure are difficult to be build, while computational phantoms may employ oversimplyfying assumptions in simulating the data acuisistion process. As it was pointed out in a recent article by Duncan and Ayache [13], we are beg ining to see the organization of databses with ground trouth in addition to the efforts to form standartised datasets [12]. For example, the Internet Brain Segmentation Repository at harvard provides manual segmentations by experts along with the magnetic resonance brain image data [14]

Applications: There are a number applications in which segmented volume data are currently being used. Among them study of anatomical structures [15, 16], diagnosis [17], localization of pathology [4], quantification of tissue volumes [18], treatment planning [19] , and computer-integrated surgery [20, 21].

The distinctions in terminology that we are using follow some historical divisions of computer vision research: we divide the mechanisms presented here into structural and statistical segmentation approaches.

2 Structural Methods

By and large, the structural segmentation methods presented in this chapter will be modeling methods driven by local volume image geometry. These techniques borrow heavily from calculus and the field of differential geometry. Structural segmentation algorithms operate in volume image space, attempting to place delimiting boundaries around contiguous volume sections.

2.1 Deformable Models

In this section, we briefly summarize the research on the area of deformable models being focused on revealing the fundamental principles and refer the reader to the cited literature for a full treatment of the topic. Other reviews on deformable models can be found in [22, 7, 23]. Deformable models are curves, surfaces or solids defined within an image or volume domain and they move under the influence of external and internal forces. In the physics-based modeling paradigm, the data apply forces (external forces) to the deformable model and as a result the model moves towards the data, while internal forces keep the model smooth during deformation. Deformable models have the following advantages: 1) they offer a coherent and consistent mathematical description, 2) they are robust to noise and boundary gaps, and 3) they can offer a subvoxel accuracy for the boundary representation that may be important at a number of applications, especially the biomedical ones. Figure **??** depicts examples of using deformable models to obtain a parametric curve and a parametric surface to describe the data depicted. The term *deformable models* first appeared in the work by Terzopoulos and his collaborators [24, 25, 26]. Since the seminal publication on Snakes [25] a number of names such as snakes, active contours or surfaces, balloons, deformable contours or surfaces, have been used to denote a deformable model. The deformable models can be classified in two broad categories: *physically-inspired parametric deformable models* [27, 28, 29] and geometric deformable models [30, 31, 32, 33]. On one hand, parametric deformable models use parametric forms, allow for direct interaction and they have a compact representation. However, changes in the topology of the models requires special procedures, since a new parameterization has to be constructed whenever the topology change occurs [34, 35]. On the other hand, geometric deformable models represent curves and surfaces implicitly as a level set of a higher-dimensional scalar function (more information can be found in Section **??**). These models, which are based on theory of curve evolution [36, 37, 38] and the level set method [39, 40] allow changes in the topology of the model easily. In this section, we limit our discussion in the applications of deformable models in segmentation. Other applications of the deformable models can be found at [41, 42, 43, 44].

Physically-based deformable models can be divided in: 1) energy-minimizing, 2) dynamic and 3) probabilistic deformable models. Using the energy minimization formulation, the goal is to find a parametric model that minimizes the weighted sum of internal energy and potential energy. The internal energy specifies the tension or the smoothness of the surface of the model. The potential energy is defined over the volume domain and typically possesses local minima at the edges occurring at object boundaries. Minimizing the total energy yields internal forces and potential forces. Internal forces maintain the smoothness of the model while external forces attract the model towards the desired object interfaces. Using the generalized force formulation allows using general types of forces. These forces do not have to be potential forces, that is forces that have to be written as the negative gradient of potential energy functions. A number of types of external forces have been proposed in the literature. For example, multiscale Gaussian potential forces [25, 45], pressure forces [28, 46, 47], distance potential forces [48], gradient vector flow [48], distance forces [49, 50], and interactive forces [25, 29]. An alternate view of deformable models casts the model fitting process in a probabilistic framework to permit the incorporation of prior model and sensor information as probability distributions [51].

Various numerical implementations of deformable models have been reported in the literature. For examples, the finite difference method [25], dynamic programming [27], and greedy algorithm [52] have been used to implement deformable contours, while finite difference methods [45] and finite element methods [29, 53, 54] have been used to implement deformable surfaces. So far, we have presented the formulation of a deformable model as a continuous curve or surface. In practice, however, it is sometimes more straightforward to design the deformable models from a discrete point of view. Example of work in this area includes [55, 56, 57, 58, 59].

A number of extensions have been proposed to the parametric deformable models. The first category of extensions refers to modeling global shape properties such as orientation and size. Modeling of global properties can provide greater robustness to initialization. Reducing the dependence on initialization has been the topic of active research [28, 30, 31, 60]. Furthermore, global properties are important in object recognition and image interpretation applications because they can be characterized using only a few parameters [61]. The second category of extensions is the incorporation of additional prior knowledge into the models by training the model to constrain the deformation to the deformations encountered in a specific class of structures. This extension is very beneficial for the applications where the shape of the target structure varies across the acquired data. An example of methods in which prior knowledge is employed to guide the segmentation are the atlas-guided approaches [62, 63, 64] or methods that build and employ shape priors [65, 66]. These methods have some of their roots in the visionary efforts of Bajcsy and her collaborators in 1983 [67]. Atlas-guided approaches have been applied main in MR brain imaging and the atlas has been generated by compiling information from manually segmented data. The problem of segmentation is then transformed to one of

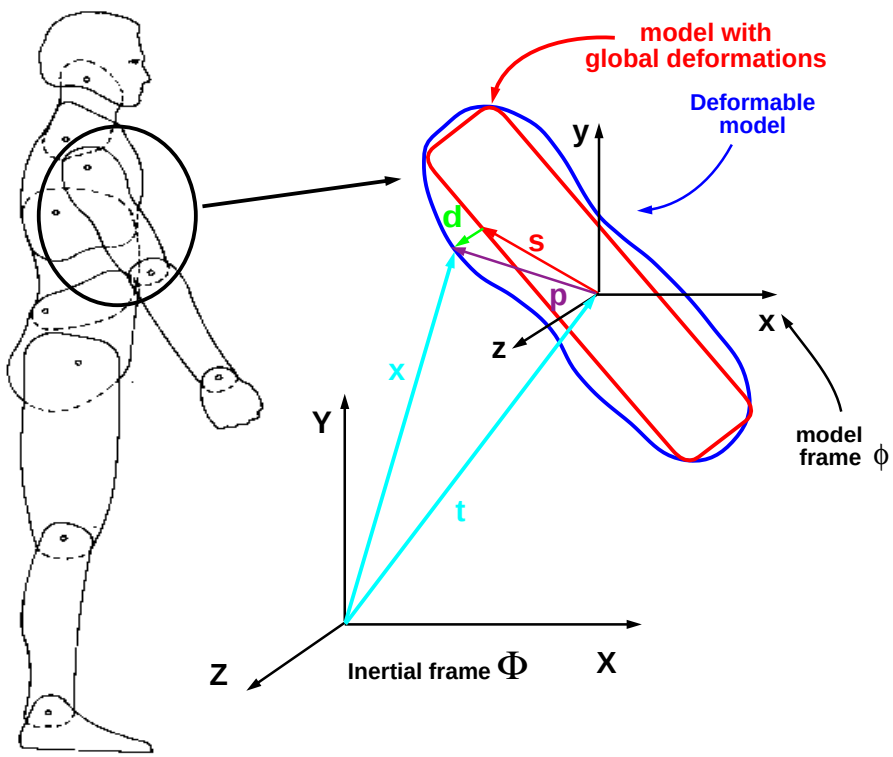


Figure 1: Coordinate systems for deformable models.

finding a one-to-one transformation that maps the atlas to the volume that needs to segmented. This atlas warping is accomplished by using both linear [62, 68, 69] and non-linear transformations [70, 63, 71]. To capture the anatomic variability, probabilistic atlases have been build.

2.2 Deformable Model Geometry

We briefly review the notation used for deformable models in [54, 61, 72]. The models used are two- and three-dimensional. The material coordinates $\mathbf{u}$ ($\mathbf{u} = (v)$ and $\mathbf{u} = (u, v)$ for the two- and three-dimensional case, respectively) of a point on these models are specified over a domain Ω. The position of a point on the model relative to an inertial frame of reference Φ in space is given by a vector-valued, time varying function. In particular, the three-dimensional position of a point w.r.t. a world coordinate system is the result of the translation and rotation of its position with respect to a non-inertial, model-centered coordinate frame ϕ (Fig. 1). Therefore, the position of a point (with material coordinates $\mathbf{u}$) on a deformable model at time t with respect to an inertial frame of reference Φ is given by the formula:

$$ {}^{\Phi}\mathbf{x}(\mathbf{u}, t) = {}^{\Phi}\mathbf{t}(t) + {}^{\Phi}_{\phi}\mathbf{R}(t)\, {}^{\phi}\mathbf{p}(\mathbf{u}, t) \ , \tag{1} $$

where ${}^{\Phi}\mathbf{t}$ is the position of the origin O of the model frame ϕ with respect to the frame Φ (the model's translation), and ${}^{\Phi}_{\phi}\mathbf{R}$ is the matrix that encapsulates the orientation of ϕ with respect to Φ. To introduce global and local deformations the position of a model point with material coordinate $\mathbf{u}$ w.r.t. the model frame ${}^{\phi}\mathbf{p}(\mathbf{u}, t)$ is expressed as the sum of a reference shape ${}^{\phi}\mathbf{s}(\mathbf{u}, t)$ and a local displacement ${}^{\phi}\mathbf{d}(\mathbf{u}, t)$ as given by the formula:

$$ {}^{\phi}\mathbf{p}(\mathbf{u}, t) = {}^{\phi}\mathbf{s}(\mathbf{u}, t) + {}^{\phi_i}\mathbf{d}(\mathbf{u}, t). \tag{2} $$

The reference shape captures the salient shape features of the model and it is the result of applying global deformation function $\mathbf{T}$ (such as tapering and bending) to a geometric primitive $\mathbf{e} = (e_x, e_y, e_z)^{\top}$. In particular,

$$ {}^{\phi_i}\mathbf{s}(v, t) = (s_x, s_y, s_z)^{\top} = \mathbf{T}(\mathbf{e}; \mathbf{q}_T), \tag{3} $$

where the global deformations defined by $\mathbf{T}$ depend on the parameters $\mathbf{q}_T$. A superellipsoid $\mathbf{e}(v)$:$[-\pi, \pi) \rightarrow \mathbb{R}^2$ with global shape parameters $\mathbf{q_e} = (a_1, a_2, \epsilon_1)^\top$ is employed as a two-dimensional shape primitive, while a superquadric $\mathbf{e}(u, v)$:$[-\pi/2, \pi/2)$x$[-\pi, \pi) \rightarrow \mathbb{R}^3$ with global shape parameters $\mathbf{q_e} = (a_1, a_2, a_3, \epsilon_1, \epsilon_2)^\top$ is employed as a three-dimensional shape primitive. The finite element method is employed to represent the continuous surface of the deformable model in the form of weighted sums of local polynomial basis functions. The finite element method provides an analytic, piecewise polynomial surface representation. Local displacements $\mathbf{d}$ are computed based on the use of triangular finite elements. Associated with every finite element node i is a nodal vector variable $\mathbf{q}_{d,i}$. We collect all the nodal variables into a vector of local degrees of freedom $\mathbf{q}_d = (\ldots, \mathbf{q}_{d,i}^\top, \ldots)^\top$, and we compute the local displacement $\mathbf{d}$ based on the finite element theory as $\mathbf{d} = \mathbf{S}\mathbf{q}_d$. $\mathbf{S}$ is the shape matrix whose entries are the finite element shape functions. Finally, we incorporate into the vector $\mathbf{q} = (\mathbf{q}_c^\top, \mathbf{q}_\theta^\top, \mathbf{q}_s^\top, \mathbf{q}_d^\top)^\top$ the degrees of freedom of our model which consist of the parameters necessary to define the translation $\mathbf{q}_c$, rotation $\mathbf{q}_{theta}$, global $\mathbf{q}_s$ and local deformations $\mathbf{q}_d$ of the model [61]. When fitting the model to data the goal is to recover the vector of degrees of freedom $\mathbf{q}$. This is achieved in a physics-based way [42]. In particular, the simplified equations of motion [61] take the general form

$$\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_q, \tag{4}$$

where $\mathbf{f}_q$ are the generalized external forces computed from the 2D or 3D forces applied from the data to the model (more details on the computation of the generalized forces are provided in [61]) and $\mathbf{K}$ is the stiffness matrix.

2.3 Elastically Adaptive Deformable Models

Deformable model formulations provide a powerful mechanism for quantitatively modeling and analyzing an object's shape, structure and motion [73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 61, 83, 84, 85, 54]. Deformable models offer a data-driven recovery process, in which forces derived from the image deform the model until it fits the data. However, most such formulations assume that the user correctly initializes the model's elastic parameters which significantly affect the goodness of fit of the model to the given data. For example, in [61] it was assumed that the elastic parameters of the finite elements used for shape estimation remain constant in space and on time and that the user chooses the initial values. However, the speed and accuracy of fitting the models to the data depends on the values selected for the elastic parameters of the model. This is a significant limitation in model fitting applications where the user assumes no a priori knowledge of the complexity of the given data.

Recently, Kakadiaris and Metaxas [86] developed a formal methodology to automatically determine a deformable model's elastic parameters which generalizes the previous formulation [61]. According to the deformable model framework the surface of the model is tessellated into a grid of finite elements. Each finite element has its own elastic parameters and these parameters may vary in time during the fitting. Their technique is based on the use of a model for the adaptation of the model's elastic parameters. The characteristic of this method is that each elastic parameter is modified based on the local error of fit and the local rate of change of the error of fit. If the model's initial elastic parameters are sufficient to fit the given data within a user specified tolerance, then their change during the fitting process will be minimal. Otherwise, they will gradually change based on the above criteria. In particular, the elastic parameters decrease when the model has not fit the data to make the model more elastic and allow for fitting. On the contrary, when the model is close to the data the elastic parameters increase to make the model more stiff. This increase of the elastic parameters has the effect of anchoring the model to the portion of the data it has fit and also improves the continuity of the solution.

3 Adaptive Elastic Parameters

A dynamic deformable model has kinetic energy and deformation strain energy $\mathcal{E}$. In particular, the deformation energy that is imposed upon the model depends on the desired continuity for the deformable model. According to the theory of elasticity, the relationship between the stresses (σ) and strains (ϵ) of an elastic material is expressed as

$$\sigma = \frac{d\mathcal{E}}{d\epsilon} = \mathbf{C}\epsilon \tag{5}$$

for a linear material, and as

$$d\sigma = \mathbf{C}(\epsilon)d\epsilon \tag{6}$$

for a non-linear material. Furthermore, by assuming a small stress-strain displacement[1] and the use of finite elements, we can take it one step further to:

$$\epsilon = \mathcal{P}\mathbf{d} = \mathcal{P}\mathbf{S}\mathbf{q}_d, \tag{7}$$

where $\mathbf{d}$ is the material displacement, $\mathbf{S}$ is the finite element shape matrix, $\mathbf{q}_d$ are the FEM nodal displacements, and the symmetric matrix $\mathcal{P}$ is derived from the local deformation strain energy. Based on the above definitions and the theory of elasticity, we can express the (linear or nonlinear) elastic deformation energy $\mathcal{E}$ w.r.t. to the FEM coordinates as

$$\mathcal{E} = \int \epsilon^\top \sigma \, d\mathrm{u} = \mathbf{q}_d^\top \, [\int (\mathcal{P}\mathbf{S})^\top \mathbf{C}(\epsilon)(\mathcal{P}\mathbf{S}) \, d\mathrm{u} \,] \, \mathbf{q}_d, \tag{8}$$

where the stiffness matrix $\mathbf{K}$ is defined as

$$\mathbf{K} = \int (\mathcal{P}\mathbf{S})^\top \mathbf{C}(\epsilon)(\mathcal{P}\mathbf{S}) \, d\mathrm{u}. \tag{9}$$

In the past, a combined membrane and a thin-plate energy deformation energy has been used which can be written in the general form:

$$\mathcal{E} = \frac{1}{2} \int (w_1 e_{11}^2 + w_2 e_{22}^2 + w_3 e_{33}^2 + w_4 e_{12}^2 + w_5 e_{23}^2 + w_6 e_{13}^2) \, d\mathrm{u}, \tag{10}$$

where e_{ij} are the components of the strain vector ϵ. The nonnegative weighting functions w_{ij} control the elasticity of the material.

Most finite element implementations for segmentation applications, assume that the elastic parameters w_i are constant across the deformable model and during the model fitting process. They are also initialized manually in the beginning of the shape estimation process. This may result in lengthy manual experimentations to identify the correct initial elastic parameter values. Second, since these parameters are assumed constant across the model, accurate shape estimation may never be achieved in case of complex data. Clearly, a technique for automatically adjusting a deformable model's elastic parameters in a local fashion is necessary.

The contribution of Kakadiaris and Metaxas [86] is the development of a new method for automatically modifying the elastic parameters w_i for each of the model's finite elements. The model for the modification of each of the model's elastic parameters is based on ideas from the theory of PD (Proportional-Derivative) control. In particular, the model's elastic parameters are modified during the fitting process based on the local error of fit and the rate of change of the error of fit. In all of the experiments, the fitting process is started with the same initial value $w_0 = 0.005$ for all the model's elastic parameters. In addition, each finite element j, ($j = 1, \ldots, k$, where k is the number of finite elements of model i) has its own elastic parameters w_i^j. The fitting of the deformable model to the given data is based on Equation (4). For each finite element j, the error of fit $\boldsymbol{\psi}^j$ is defined to be the average of the error of fit of its nodes $k(k = 1, \ldots, m$, where m is the number of nodes of finite element j). In particular,

$$\boldsymbol{\psi}^j = \frac{\sum_{k=1}^{m} (\boldsymbol{\rho}_k^j - \mathbf{x}_k^j)}{m}, \tag{11}$$

where $\mathbf{x}_k^j$ is the position of the kth node of element j and $\boldsymbol{\rho}_k^j$ is the average position of the data points assigned to this node, based on algorithms defined in [61].

During the fitting process, the values of each of the elastic parameters w_i^j for each finite element j are modified based on

$$w_i^j(t) = (w_0 - w_{min}) \;\; e^{sgn(\boldsymbol{\psi}^j . \dot{\boldsymbol{\psi}}^j)(||\boldsymbol{\psi}^j|| + ||\dot{\boldsymbol{\psi}}^j||)} + w_{min}, \tag{12}$$

where w_{min} is the minimum value for all the elastic parameters which for a membrane and/or thin-plate deformable model is 10^{-5}, and sgn is the $sign$ function. Notice that $sgn(\boldsymbol{\psi}^j . \dot{\boldsymbol{\psi}}^j)$ is negative or zero when the model is converging towards the data and positive otherwise. The whole process of fitting and elastic parameter adaptation terminates when the error of fit for each finite element is below a tolerance ψ_{tol} specified by the user.

The adaptation of the elastic parameters of a deformable model using the method described above has the following desired properties. As seen in Equation (12), the change in each of the w_i^j's is always w.r.t. their initial value w_0.

[1] The theory easily generalizes to large stress-strains.

Initially, since the error of fit is large, while the rate of change of the error of fit is small or zero, the values of the w_i^j's decrease exponentially to quickly improve the fitting. In the intermediate steps of the fitting, the values of the w_i^j's stabilize and are not modified significantly, since the sum of the error of fit and the rate of change of the error of fit does not change substantially. When the model is very close to the desired data, the sum of the error of fit and the rate of change of the error of fit decrease (the forces assigned to each node are now small) and the result is an increase of the w_i^j's towards w_0. This results in a model that achieves a smooth solution to the data where necessary and also better "holds" the model to the desired data.

When the model has almost fit the data, the values of the w_i^j's start to exponentially increase again towards their initial value. Therefore, the elastic parameters oscillate mostly between w_0 and w_{min}. Due to the introduction of $sgn(\boldsymbol{\psi}^j.\dot{\boldsymbol{\psi}}^j)$ in Equation (12), the model's elastic parameters are automatically increased beyond w_0 if the model has fit the data and tries to deviate from them. Therefore, the model resists deviation from the data once it has fit them. This is an additional desired property in cases where the model has partially fit the data. It will allow the portion of the model that has not fit the data to become more elastic and fit the data, while the portion that has fit them will not be modified or become more stiff in case there is any deviation from the data.

3.0.1 Dynamic Shape Estimation

The above model for the modification of the model's elastic parameters does not take into account the noise in the data. In [87] it was shown how the dynamics of a deformable model can be incorporated into an extended Kalman filtering framework to formally account for noise in the data. However, an extension to this formulation is necessary in order to incorporate the model for the modification of the model's elastic parameters into a Kalman filler. This is necessary because the model's elastic parameters are not degrees of freedom, since they do not appear in $\mathbf{q}$, but are the unknown parameters that determine the value of the stiffness matrix $\mathbf{K}$. Therefore, this problem is that of parameter identification in a dynamic system. In [88, 89], the theory of parameter identification in dynamic systems is presented. Based on this approach, with the addition that of a model for the modification of the deformable model's elastic parameters[2] the state vector of the system is augmented to include the model's elastic parameters. Therefore, the new state vector is of the form

$$\mathbf{u} = \begin{bmatrix} \mathbf{w} \\ \mathbf{q} \end{bmatrix}, \tag{13}$$

where $\mathbf{w}$ is the vector of the model's elasticity parameters with components w_i^j.

Based on the above modification, we introduce the following definitions necessary to define the equations of a modified extended Kalman filter. Let the observation vector $\mathbf{z}(t)$ denote time-varying input data. We can relate $\mathbf{z}(t)$ to the model's state vector $\mathbf{u}(t)$ through the nonlinear observation equation

$$\mathbf{z} = \mathbf{h}(\mathbf{u}) + \mathbf{v}, \tag{14}$$

where $\mathbf{v}(t)$ represents uncorrelated measurement errors as a zero mean white noise process with known covariance $\mathbf{V}(t)$, i.e., $\mathbf{v}(t) \sim \mathbf{N}(\mathbf{0}, \mathbf{V}(t))$. If $\mathbf{z}$ consists of observations of time varying positions of model points at material coordinates u_k on the model's surface, the components of $\mathbf{h}$ are computed using (1) evaluated at u_k[3] (see also [42]). In case of computing an image potential, then what is being measured at every node of the model is the difference $\mathbf{z} - \mathbf{h}(\mathbf{u})$, which is what is needed for an extended Kalman filter formulation. In addition, let us also assume that $\mathbf{a}(t)$ represents uncorrelated modeling errors as a zero mean white noise process with known covariance i.e., $\mathbf{a}(t) \sim \mathbf{N}(\mathbf{0}, \mathbf{Q}(t))$.

Based on the above definitions and Equation (4), the *modified* extended Kalman filter equations for our dynamic system take the following form:

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}) + \mathbf{a}, \tag{15}$$

$$\mathbf{z} = \mathbf{h}(\mathbf{u}) + \mathbf{v}, \tag{16}$$

$$\tag{17}$$

[2] According to that theory the time derivative of the elastic parameters should have been zero as opposed to the one used herein.

[3] Note that the definition of function ξ in Equation (1) does not depend on $\mathbf{w}$.

where

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} \dot{\mathbf{w}} \\ -\mathbf{K}\mathbf{q} \end{bmatrix}, \dot{\mathbf{w}} = (\dot{w}_1^1, ..., \dot{w}_i^j, ..., \dot{w}_6^k)^\top, \tag{18}$$

$$\dot{w}_{i_j} = (w_0 - w_{min}) \; e^{sgn(\boldsymbol{\psi}_j.\dot{\boldsymbol{\psi}}_j)(||\boldsymbol{\psi}_j||+||\dot{\boldsymbol{\psi}}_j||)} \; sgn(\boldsymbol{\psi}_j.\dot{\boldsymbol{\psi}}_j) \; (||\dot{\boldsymbol{\psi}}_j|| + ||\ddot{\boldsymbol{\psi}}_j||). \tag{19}$$

Note that due to the modification in the state vector, we now have a fully nonlinear extended Kalman filter as opposed to our previous formulations [61]. However, the filter converges to the right solution since we impose a correct behavior to the model for the adaptation of the model's elastic parameters and the model dynamics are appropriate for our applications.

The state estimation equation for uncorrelated system and measurement noises (i.e., $E[\mathbf{a}(t)\mathbf{v}^\top(t)] = 0$) is

$$\dot{\hat{\mathbf{u}}} = \mathbf{f}(\hat{\mathbf{u}}) + \mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}\left(\mathbf{z} - \mathbf{h}(\hat{\mathbf{u}})\right), \tag{20}$$

where $\mathbf{H}$ is computed from

$$\mathbf{H} = \left.\frac{\partial \mathbf{h}(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}=\hat{\mathbf{u}}}. \tag{21}$$

The expression $\mathbf{G}(t) = \mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}$ is known as the Kalman gain matrix. The symmetric error covariance matrix $\mathbf{P}(t)$ is the solution of the matrix Riccati equation

$$\dot{\mathbf{P}} = \mathbf{F}\mathbf{P} + \mathbf{P}\mathbf{F}^\top + \mathbf{Q} - \mathbf{P}\mathbf{H}^\top\mathbf{V}^{-1}\mathbf{H}\mathbf{P}, \tag{22}$$

where

$$\mathbf{F}(\mathbf{u}) = \left.\frac{\partial \mathbf{f}(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}=\hat{\mathbf{u}}}. \tag{23}$$

The improvement offered from the Kalman filter formulation is that one can formally introduce data noise statistics into the model.

3.0.2 Implementation

Since the model's equations of motion are numerically well-conditioned, the full Kalman filter formulated above can be partitioned into two separate filters for computational efficiency. The first filter includes the translation, rotation and global deformations, and the second one includes only the local deformations. While the computation to the solution of the Riccati equation for the first filter is fast since the associated degrees of freedom are few, this is not the case for the second filter whose state vector includes the model's elastic parameters and the local degrees of freedom.

To solve the matrix Riccati equations at interactive rates in the latter case, we take advantage of the decomposition of the model's surface into finite elements. A similar approach was used by Metaxas [42] for the computation of the stiffness matrix $\mathbf{K}$. Based on the covariance of each component of $\mathbf{u}$ that corresponds to the variable of the second Kalman filter at a given step, the contribution of each element to (22) is computed using the right hand side of this equation for each element. This per-element computation of (22) results in matrices of very small dimensions compared to the size of the respective matrices for the whole system. This computation can be done in parallel and once all the elements are looped through, the contribution from each element is placed at the appropriate location in $\mathbf{P}$ (in an identical way to the computation of $\mathbf{K}$). Then we solve (22). This significantly improves the speed of the calculations and is justified since the model's surface is partitioned into finite elements.

3.0.3 Examples

Based on the above implementation, all our experiments run at interactive rates on a Silicon Graphics R10000 Indigo^2 workstation. Furthermore, we always started from a unit covariance matrix $\mathbf{P}$. However, the subsequent structure of $\mathbf{P}$ is not diagonal and has a form similar to $\mathbf{K}$. Notice that any other reasonable initial condition will work if our dynamic model is appropriate for our applications. For the global deformations we used a superellipsoid or a superquadric, the elastic parameters were always initialized to $w_0 = 0.005$, and we used an adaptive Euler integration method for increased stability. In addition, the noise in the data is small and we defined $\mathbf{V}$ as $\mathbf{V} = 0.1\mathbf{I}$.

As a first example, we present the results of applying the elastically adaptive deformable models technique to the semi-automated identification of the myocardial borders from breath-hold MRI. The data were obtained from the Department of Radiology at the University of Pennsylvania. The dataset included sixteen slice locations, from the Left Ventricle (LV) apex to the level of the aortic valve. In order to determine the location of the borders with higher accuracy we magnify each image four times, and then we convolve it with an 8x8 Gaussian mask. An initial superellipsoid model of the model was placed manually at the vicinity of the border of the first slice. In this study, we concentrated on the identification of the LV endocardial contour for locations 4 to 11 in which the contour is visible (Figures 2 and 3). During the fitting process, the results from fitting one slice, were used as the initial model for the next slice, as if we had an evolving curve over time. Therefore the user only initializes the model in the first slice.

The second example presents the fitting of an elastically adaptive deformable model to the 1269 three-dimensional data points of a Viewpoint® model of a human head. Each of the figures 4(a-h) and 5(a-h) is composed of four parts. The two sub-images at the upper row of each sub-figure depict the model as seen from the front and back, respectively. The lower left sub-image of each figure depicts a side view of the model while the lower right sub-image depicts the side view of the model along with the three-dimensional data. Each finite element of the three-dimensional model is color coded to depict the value of the elastic parameters at that time instant.

References

[1] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277–1294, 1993.

[2] J. C. Bezdek, L. O. Hall, and L. P. Clarke, "Review of MR image segmentation techniques using pattern recognition," *Medical Physics*, vol. 20, pp. 1033–1048, 1993.

[3] P. Suetens, E. Bellon, D. Vandermeulen, M. Smet, G. Marchal, J. Nuyts, and L. Mortelmans, "Image segmentation: methods and applications in diagnostic radiology and nuclear medicine," *European Journal of Radiology*, vol. 17, pp. 14–21, 1993.

[4] A. P. Zijdenbos and B. M. Dawant, "Brain segmentation and white matter lesion detection in MR images," *Critical Reviews in Biomedical Engineering*, vol. 22, pp. 401–465, 1994.

[5] L. P. Clarke, R. P. Velthuizen, M. A. Camacho, J. J. Heine, and M. et al, "MRI segmentation: methods and applications," *Magnetic Resonce Imaging*, vol. 13, pp. 343–368, 1995.

[6] N. Ayache, P. Cinquin, I. Cohen, L. Cohen, F. Leitner, and O. Monga, "Segmentation of complex 3D medical objects: A challenge and a requirement for computer assisted surgery planning and performing," in *Computer Integrated Surgery (to appear)* (R. Taylor, S. Lavallee, G. Burdea, and R. Mosges, eds.), Cambridge, MA: MIT Press, 1994.

[7] D. Pham, C. Xu, and J. Prince, "A survey of current methods in medical image segmentation," Tech. Rep. JHU/ECE 99-01, Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD, 1999.

[8] Y. J. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognition Letters*, vol. 29, pp. 1335–1346, 1996.

[9] P. Wust, J. Gellermann, J. Beier, S. Wegner, and W. Tilly, "Evaluation of segmentation algorithms for generation of patient models in radiofrequency hyperthermia," *Physics in Medicine and Biology*, vol. 43, pp. 3295–3307, 1998.

[10] M. Vaidyanathan, L. P. Clarke, C. H. L. O. Hall, and R. Velthuizen, "Monitoring brain tumor response to therapy using MRI segmentation," *Magnetic Resonance Imaging*, vol. 15, pp. 323–334, 1997.

[11] T. Lei and W. Sewchand, "Statistical approach to X-ray CT imaging and its applications in image analysis part ii: A new stochastic model-based image segmentation technique for X-ray CT image," *IEEE Transactions on Medical Imaging*, vol. 11, no. 1, pp. 62–69, 1992.

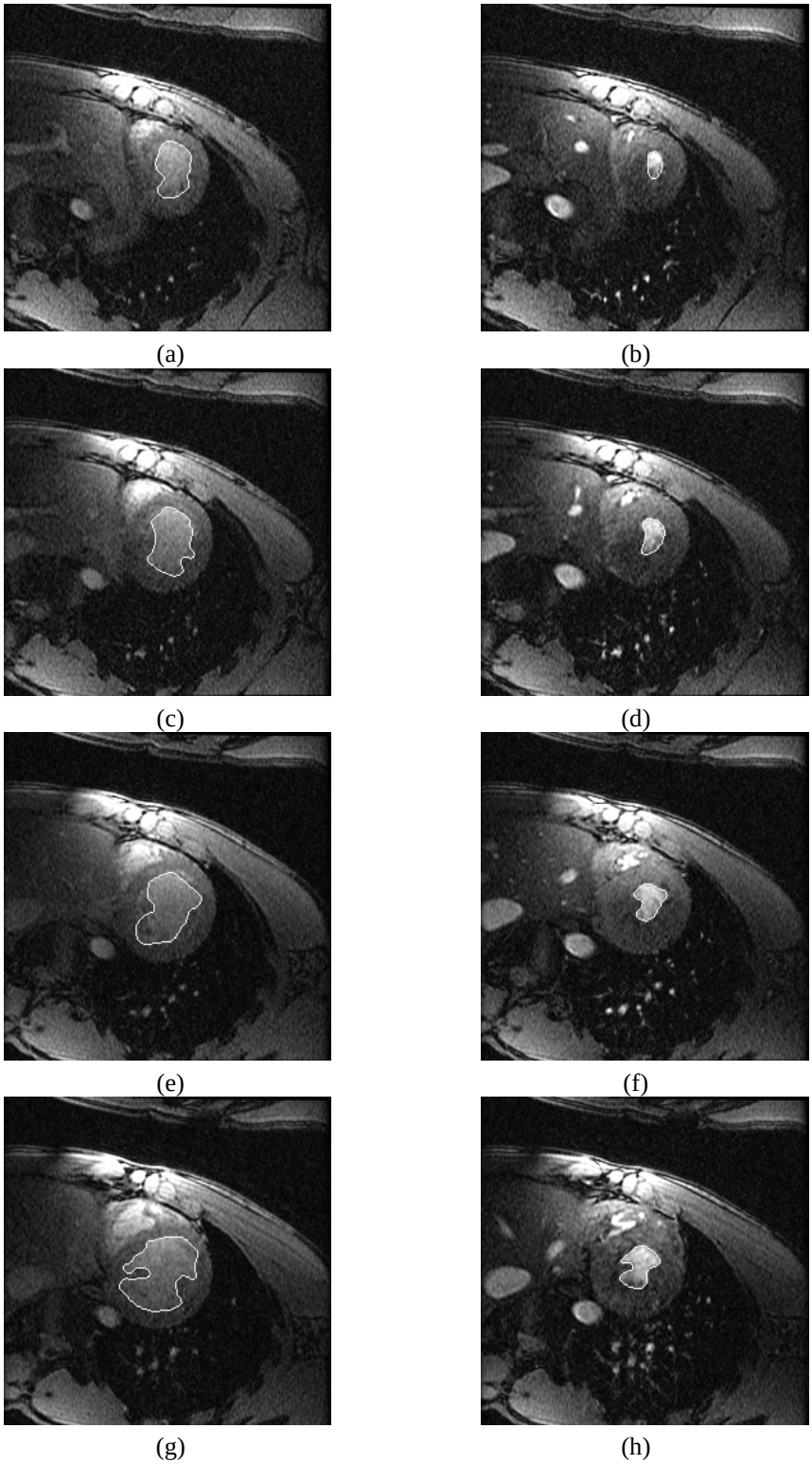


Figure 2: Semi-automated identification of the myocardial borders. (a, c, e, g) End diastole phase, and (b, d, f, h) end systole phase for locations 4, 5, 6 and 7, respectively.

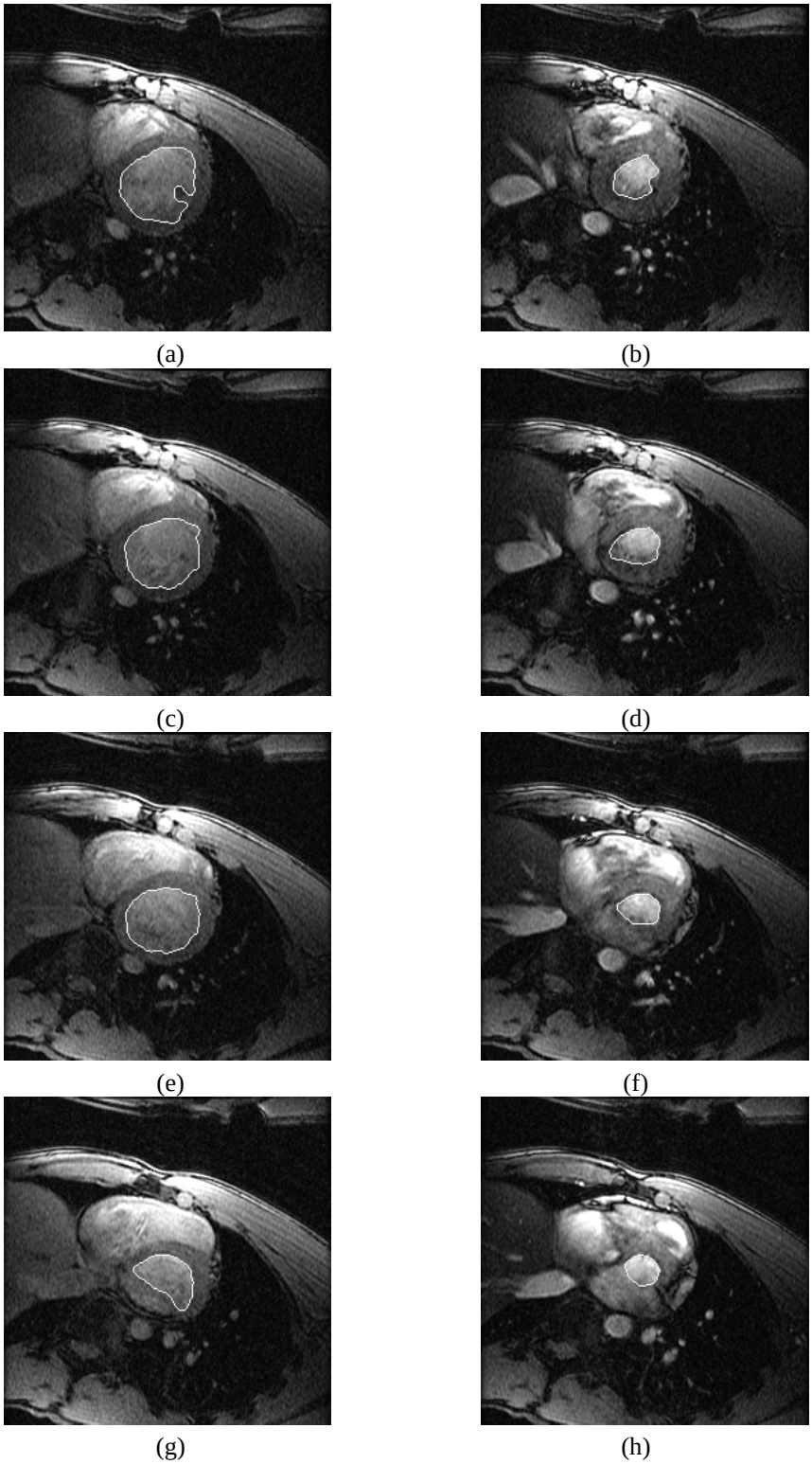


Figure 3: Semi-automated identification of the myocardial borders. (a, c, e, g) End diastole phase, and (b, d, f, h) end systole phase for locations 8, 9, 10 and 11, respectively.

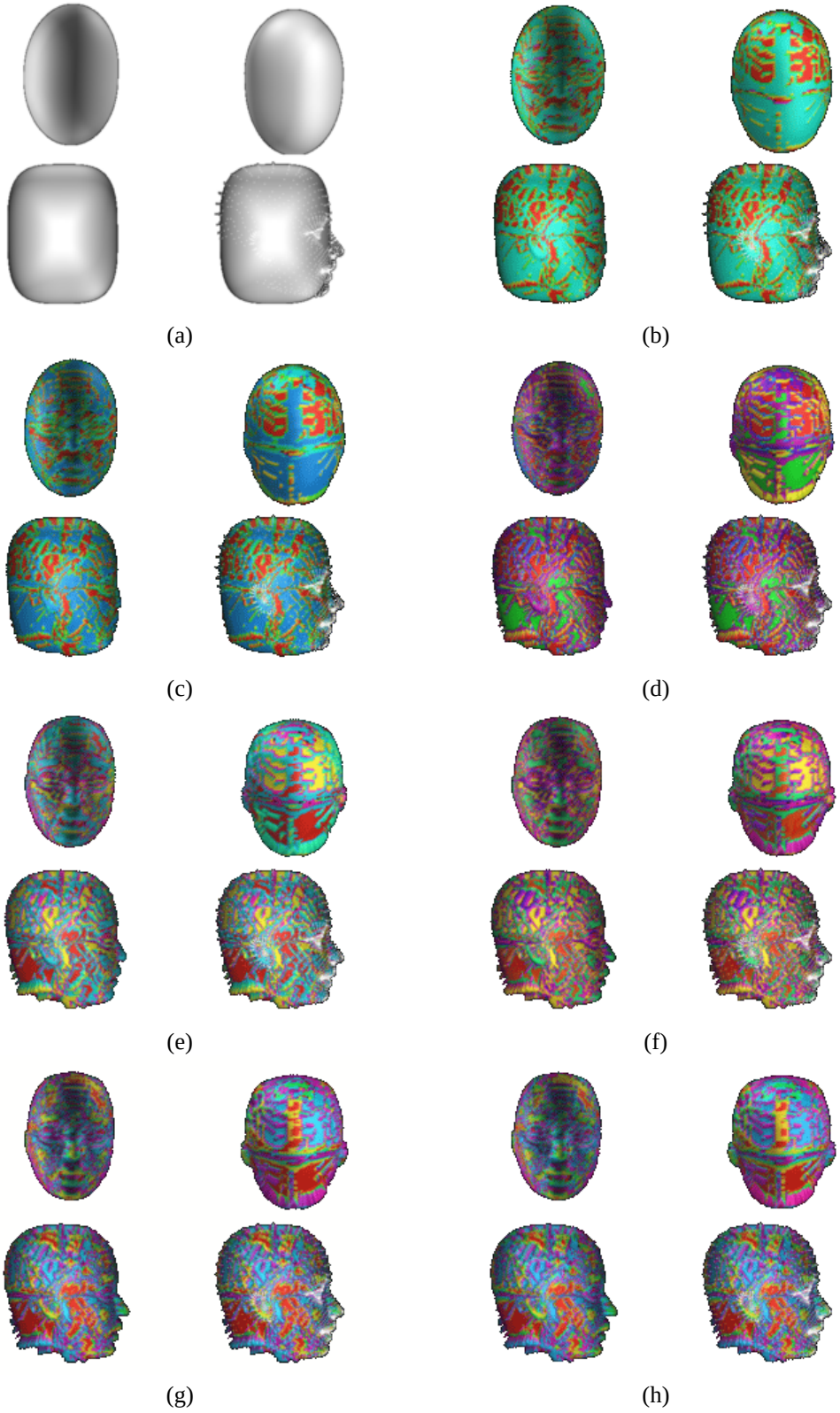


Figure 4: Fitting the three-dimensional data.

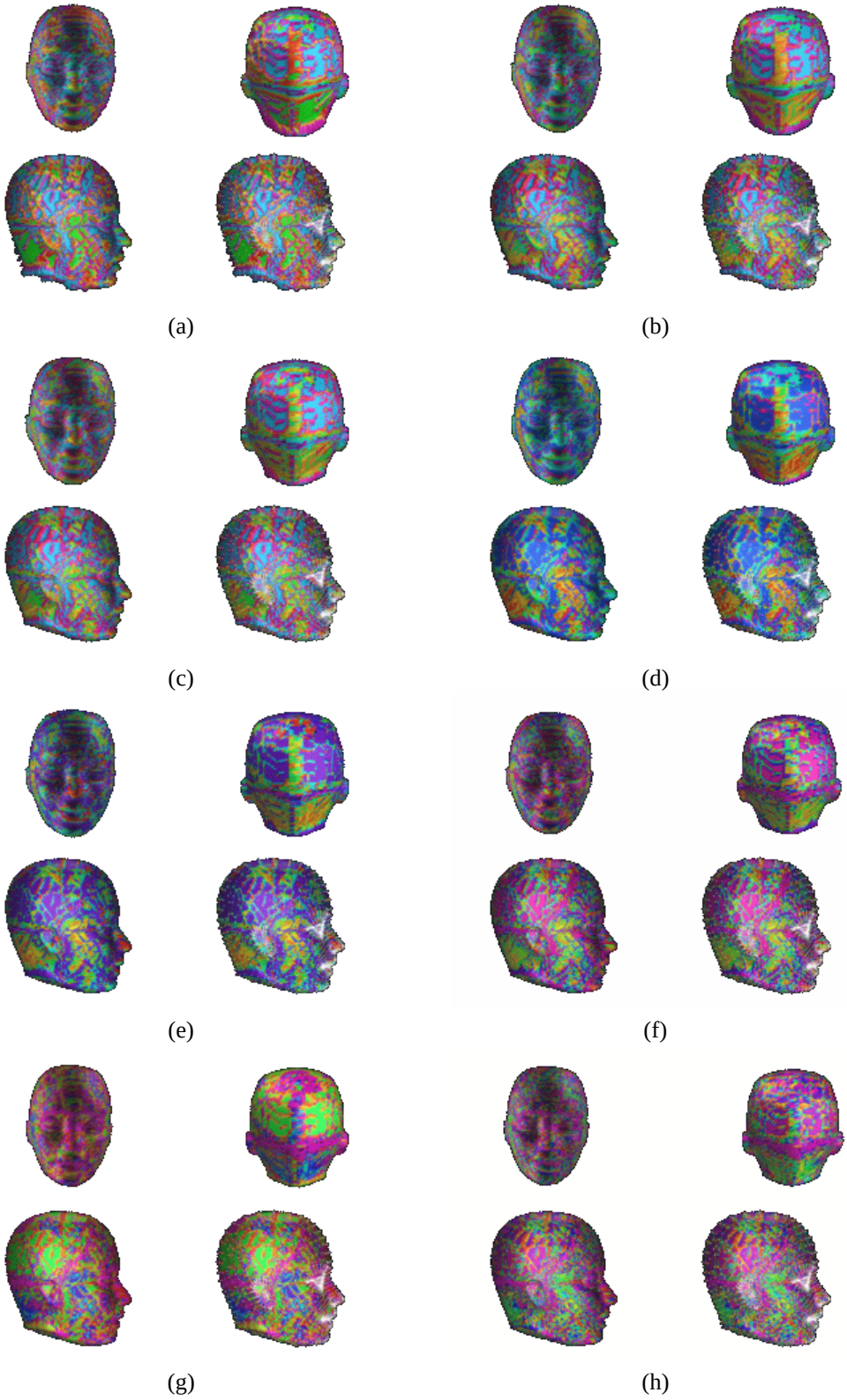


Figure 5: Fitting the three-dimensional data (Cont.).

[12] D. Collins, A. Zijdenbos, V. Kollokian, J. Sled, N. Kabani, C. Holmes, and A. Evans, "Design and construction of a realistic digital brain phantom," *IEEE Transactions on Medical Imaging*, vol. 17, pp. 463–468, June 1998.

[13] J. Duncan and N. Ayache, "Medical image analysis: Progress over two decades and the challenges ahead," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 85–106, January 2000.

[14] J. C. Rajapakse and F. Fruggel, "Segmentation of mr images with intensity inhomogeneities," *Image and Vision Computing*, vol. 16, pp. 165–180, March 1998. http://neuro-www.mgh.harvard.edu/cma/ibsr.

[15] A. J. Worth, N. Makris, V. S. Caviness, and D. N. Kennedy, "Neuroanatomical segmentation in MRI: technological objectives," *Int'l J. Pattern Recognition Artificial Intelligence*, vol. 11, pp. 1161–1187, 1997.

[16] C. A. Davatzikos and J. L. Prince, "An active contour model for mapping the cortex," *T-MI*, vol. 14, pp. 65–80, 1995.

[17] P. Taylor, "Invited review: computer aids for decision-making in diagnostic radiology a literature review," *Brit. J. Radiol.*, vol. 68, pp. 945–957, 1995.

[18] S. M. Lawrie and S. S. Abukmeil, "Brain abnormality in schizophrenia: a systematic and quantitative review of volumetric magnetic resonance imaging studies," *British Journal of Psychiatry*, vol. 172, pp. 110–120, 1998.

[19] V. S. Khoo, D. P. Dearnaley, D. J. Finnigan, A. Padhani, S. F. Tanner, and M. O. Leac, "Magnetic resonance imaging (MRI): considerations and applications in radiotheraphy treatment planning," *Radiother. Oncol.*, vol. 42, pp. 1–15, 1997.

[20] N. Ayache, P. Cinquin, I. Cohen, L. Cohen, F. Leitner, and O. Monga, "Segmentation of complex three-dimensional medical objects: a challenge and a requirement for computer-assisted surgery planning and performance," in *Computer-Integrated Surgery: Technology and Clinical Applications* (R. H. Taylor, S. Lavallee, G. C. Burdea, and R. Mosges, eds.), pp. 59–74, MIT Press, 1996.

[21] W. E. L. Grimson, G. J. Ettinger, T. Kapur, M. E. Leventon, and W. M. Wells, "Utilizing segmented MRI data in image-guided surgery," *Int'l J. Pattern Recognition Artificial Intelligence*, vol. 11, pp. 1367–1397, 1997.

[22] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey," *Medical Image Analysis*, vol. 1, no. 2, pp. 91–108, 1996.

[23] I. Bankman, ed., *Handbook of Medical Image Processing*, ch. Deformable Models in Medical Image Analysis: A survey, 1999 Update. 1999. In Press.

[24] D. Terzopoulos, "On matching deformable models to images.," *Reprinted in Topical Meeting on Machine Vision, Technical Digest Series*, vol. 12, pp. 160–167, 1987.

[25] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[26] D. Terzopoulos and K. Fleischer, "Deformable models," *The Visual Computer*, vol. 4, pp. 306–331, 1988.

[27] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855–867, 1990.

[28] L. Cohen, "On active contour models and balloons," *CVGIP: Image Understanding*, vol. 53, pp. 211–218, March 1991.

[29] T. McInerney and D. Terzopoulos, "A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis," *Comp. Med. Imag. Graph.*, vol. 19, no. 1, pp. 69–83, 1995.

[30] V. Caselles, F. Catte, and F. Dibos, "A geometric model for active contour," *Numerische Mathematik*, vol. 66, pp. 1–31, 1993.

[31] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 158–175, 1995.

[32] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," in *Proceedings of the Fifth International Conference on Computer Vision*, (Boston, MA), pp. 694–699, IEEE Computer Society Press, June 20-22 1995.

[33] R. T. Whitaker, "Volumetric deformable models: Active blobs," in *Visualization In Biomedical Computing 1994* (R. A. Robb, ed.), (Mayo Clinic, Rochester, Minnesota), pp. 122–134, SPIE, 1994.

[34] R. Durikovic, K. Kaneda, and H. Yamashita, "Dynamic contour: a texture approach and contour operations," *The Visual Computer*, vol. 11, pp. 277–289, 1995.

[35] T. McInerney and D. Terzopoulos, "Topologically adaptable snakes," in *Proceedings of the Fifth International Conference on Computer Vision*, (Boston, MA), pp. 840–845, IEEE Computer Society Press, June 20-22 1995.

[36] G. Sapiro and A. Tannenbaum, "Affine invariant scale-space," *Int'l J.Comp. Vis.*, vol. 11, no. 1, pp. 25–44, 1993.

[37] B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker, "Shapes, shocks, and deformations I: the components of two-dimensional shape and the reaction-diffusion space," *IJCV*, vol. 15, pp. 189–224, 1995.

[38] R. Kimmel, A. Amir, and A. M. Bruckstein, "Finding shortest paths on surfaces using level sets propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 635–640, 1995.

[39] S. Osher and J. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[40] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Material Science*. Cambridge, UK: Cambridge University Press, 2nd ed., 1999.

[41] A. Blake and A. Yuille, *Active Vision*. Cambridge, Massachusetts: The MIT Press, 1992.

[42] D. Metaxas, *Physics-Based Deformable Models: Applications to Computer Vision, Graphics and Medical Imaging*. Kluwer-Academic Publishers, November 1996.

[43] A. Blake and M. Isard, *Active Contours: The Application of Techniques from Graphics, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. New York: Springer-Verlag, 1998.

[44] A. Singh, D. Goldgof, and D. Terzopoulos, *Deformable models in medical image analysis*. Los Alamitos, CA: IEEE Computer Society, 1998.

[45] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering shape and non-rigid motion," *Artiffcial Intelligence*, vol. 36, pp. 91–123, 1988.

[46] R. Ronfard, "Region-based strategies for active contour models," *IJCV*, vol. 13, no. 2, pp. 229–251, 1994.

[47] C. S. Poon and M. Braun, "Image segmentation by a deformable contour model incorporating region analysis," *Phys. Med. Biol.*, vol. 42, pp. 1833–1841, 1997.

[48] G. Borgefors, "Distance transformations in arbitrary dimensions," *Comp. Vis. Graph. Imag. Proc.*, vol. 27, pp. 321–345, 1984.

[49] H. Delingette, "Simplex meshes: a general representation for 3d shapereconstruc-tion," tech. rep., INRIA, Sophia-Antipolis, France, 1994.

[50] D. MacDonald, D. Avis, and A. C. Evans, "Multiple surface identification and matching in magnetic resonance images," in *vbc94* (R. Robb, ed.), vol. 2359 of *SPIE Proc.*, (Bellingham, WA), pp. 160–169, SPIE, 1994.

[51] D. Terzopoulos and R. Szeliski, "Tracking with Kalman snakes," in *Active Vision* (A. Blake and A. Yuille, eds.), pp. 3–20, Cambridge, MA: MIT Press, 1992.

[52] D. J. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Imag. Under.*, vol. 55, no. 1, pp. 14–26, 1992.

[53] I. Cohen, L. D. Cohen, and N. Ayache, "Using deformable surfaces to segment 3-d images and infer differential structures," *CVGIP: Imag. Under.*, vol. 56, no. 2, pp. 242–263, 1992.

[54] D. Terzopoulos and D. Metaxas, "Dynamic 3D models with local and global deformations: Deformable superquadrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, pp. 703–714, 1991.

[55] A. Gupta, L. von Kurowski, A. Singh, D. Geiger, C. C. Liang, M. Y. Chiu, L. P. Adler, M. Haacke, and D. L. Wilson, "Cardiac MR image segmentation using deformable models," in *Proceedings of IEEE Conf. Computers in Cardiology*, pp. 747–750, 1993.

[56] H. Delingette, "Adaptive and deformable models based on simplex meshes," in *Proceedings of the 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, (Austin, TX), pp. 152–157, IEEE Computer Society Press, November 11-12 1994.

[57] S. Kumar and D. Goldgof, "Automatic tracking of spamm grid and the estimation of deformation parameters from cardiac MR images," *T-MI*, vol. 13, no. 122-132, 1994.

[58] S. Lobregt and M. A. Viergever, "A discrete dynamic contour model," *T-MI*, vol. 14, pp. 12–24, 1995.

[59] C. Nastar and N. Ayache, "Frequency-based nonrigid motion analysis: application to four dimensional medical images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1067–1079, 1996.

[60] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, pp. 359–369, 1998.

[61] D. Metaxas and D. Terzopoulos, "Shape and nonrigid motion estimation through physics based synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 580–591, 1993.

[62] J. Talairach and P. Tournoux, *Co-Planar Stereotaxic Atlas of the Human Brain. 3-Dimensional Pro-portional System: An Approach to Cerebral Imaging*. Stuttgart, NY: Thieme Medical Publisher, Inc., 1988.

[63] C. Davatzikos, "Spatial normalization of 3d images using deformable models," *Journal of Computer Assisted Tomography*, vol. 20, pp. 656–665, 1996.

[64] T. Cootes, C. Beeston, G. Edwards, and C. Taylor, "A unified framework for atlas matching using active appearance models," in *Proceedings on Information Processing in Medical Imaging*, (Visegrad, Hungary), pp. 322–333, June 1999.

[65] L. H. Staib and J. Duncan, "Model-based deformable surface finding for medical images," *IEEE Transactions on Medical Imaging*, vol. 78, pp. 720–731, October 1996.

[66] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape modelstheir training and application," *Computer Vision and Image Understanding*, vol. 61, 38-59 1995.

[67] R. Bajcsy, E. Lieberson, and M. Reivich, "A computerized system for elastic matching of deformed radiographic images to idealized atlas images," *Journal Computer Assisted Tomography*, vol. 7, pp. 618–625, 1983.

[68] J. L. Lancaster, L. H. Rainey, J. L. Summerlin, C. S. Freitas, P. T. Fox, A. C. Evans, A. W. Toga, and J. C. Mazziotta, "Automated labeling of the human brain: A preliminary report on the development and evaluation of a forward-transform method," *Human Brain Mapping*, vol. 5, pp. 238–242, 1997.

[69] N. C. A. R. Rajarethinam and e. a. T. Cizadlo, "Automatic atlas-based volume estimation of human brain regions from MR images," *Journal of Computer Assisted Tomography*, vol. 20, pp. 98–106, 1996.

[70] D. L. Collins, C. J. Holmes, T. M. Peters, and A. C. Evans, "Automatic 3-d model-based neuroanatomical segmentation," *Human Brain Mapping*, vol. 3, pp. 190–208, 1995.

[71] S. Sandor and R. Leahy, "Surface-based labeling of cortical anatomy using a deformable atlas.," *IEEE Transactions on Medical Imaging*, vol. 16, pp. 41– 54, 1997.

[72] I. A. Kakadiaris, *Motion-Based Part Segmentation, Shape and Motion Estimation of Multi-Part Objects: Application to Human Body Tracking*. PhD dissertation, Department of Computer Science, University of Pennsylvania, Philadelphia, PA, October 1996.

[73] A. Blake and M. Isard, "3D Position, attitude and shape input using video tracking of hands and lips," in *Proceedings of SIGGRAPH '94*, Annual Conference Series, (Orlando, FL), pp. 185–192, ACM SIGGRAPH, ACM Press, July 24–29 1994.

[74] L. Cohen, "On active contour models and balloons," *CVGIP: Image Understanding*, vol. 53, pp. 211–218, March 1991.

[75] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 223–240, 1988.

[76] W. C. Huang and D. B. Goldgof, "Adaptive-size meshes for rigid and nonrigid shape analysis and synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 611–616, June 1993.

[77] I. A. Kakadiaris and D. Metaxas, "3D Human body model acquisition from multiple views," in *Proceedings of the Fifth International Conference on Computer Vision*, (Boston, MA), pp. 618–623, IEEE Computer Society Press, June 20-22 1995.

[78] I. A. Kakadiaris and D. Metaxas, "Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection," in *Proceedings of the 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (San Francisco, CA), pp. 81–87, IEEE Computer Society Press, New York, NY, June 18-20 1996.

[79] I. Kakadiaris, D. Metaxas, and R. Bajcsy, "Inferring 2D object structure from the deformation of apparent contours," *Computer Vision and Image Understanding*, vol. 65, pp. 129–147, February 1997.

[80] I. A. Kakadiaris and D. Metaxas, "3D Human body model acquisition from multiple views," *International Journal on Computer Vision*, vol. 30, no. 3, pp. 191–218, 1998.

[81] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," in *Proceedings of the International Conference on Computer Vision*, (London, UK), pp. 259–268, IEEE Computer Society Press, June 8-11 1987.

[82] O. V. Larsen, P. Radeva, and E. Marti, "Bounds on the optimal elasticity parameters for a snake," in *8th International Conference on Image Analysis and Processing, ICIAP'95*, pp. 37–44, Springer Verlag, 1995.

[83] D. Metaxas, E. Koh, and N. Badler, "Multi-level shape representation using global deformations and locally adaptive finite elements," *International Journal on Computer Vision*, vol. 25, pp. 46–42, October 1997.

[84] A. Pentland and B. Horowitz, "Recovery of nonrigid motion and structure," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 730–742, July 1991.

[85] R. Samadani, "Adaptive snakes: Control of damping and material parameters," in *SPIE Geometric methods in Computer Vision*, no. 1570, IEEE Computer Society Press, 1991.

[86] D. Metaxas and I. A. Kakadiaris, "Elastically adaptive deformable models," in *Proceedings of the Fourth European Conference on Computer Vision* (B. Buxton and R. Cipolla, eds.), Lecture Notes in Computer Science, (Cambridge, UK), pp. II:550–559, Springer, April 14-18 1996.

[87] D. Metaxas, *Physics-Based Modeling of Nonrigid Objects for Vision and Graphics*. PhD dissertation, University of Toronto, Department of Computer Science, 1992.

[88] A. Gelb, *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1974.

[89] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Applications.* Prentice Hall, 1993.

Three-dimensional Model-based Segmentation

Guido Gerig
UNC Chapel Hill

Collaborators UNC	**Collaborators ETH**
Martin Styner	András Kelemen (Ph.D. thesis)
Sean Ho	Gábor Székely

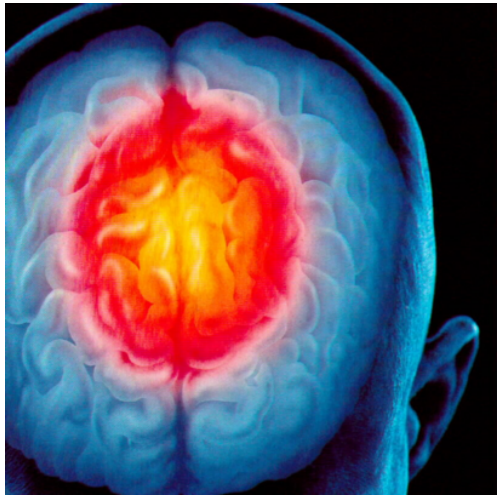
Contents

- Motivation, driving problems
- State-of-the-art
- Status of own research
  - ETH approach
  - UNC DSL
- Open issues, Plan

Motivation, Driving Problems

- large increase of amount of medical image data sets
- 3-D and 4-D images become information source in various research domains
- volume images carry detailed morphological and functional information
- need for computer-assisted tools for
  - extraction/representation of anatomical structures
  - morphometric measurements and shape analysis
- Problem: lack of appropriate analysis tools (efficiency, reliability)

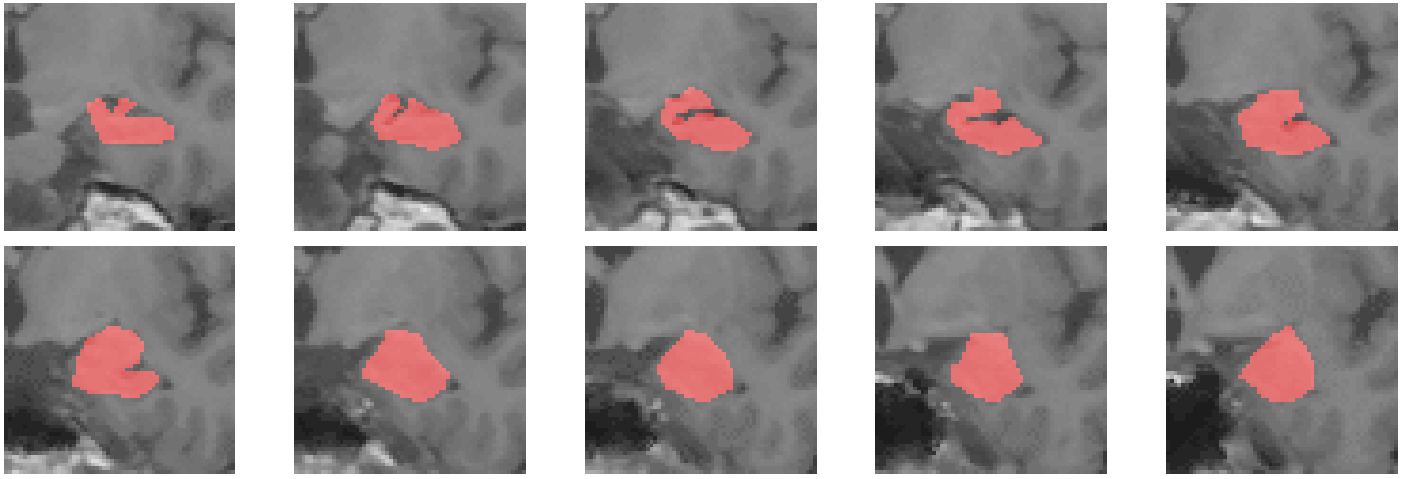
Non-invasive imaging: in-vivo studies, morphometry, function, temporal studies



Imaging technologies are changing the way science is done

(Science, Vol. 261, July 1993, R.P. Crease)

Manual 3-D Segmentation



Advantages: Interactive

Disadvantages: Inaccurate
Poor Reproducibility
3-D from 2-D Slices
Slow and Tedious

Methods for morphometric analysis of brain structures

- Atlas-based subdivision of brain images:
  Mazziota, 1995 ICBM / Warfield, 1996
  Christensen, Vannier, Miller 1997
  Ayache and Thirion 1996 / Evans and Collignon 1995

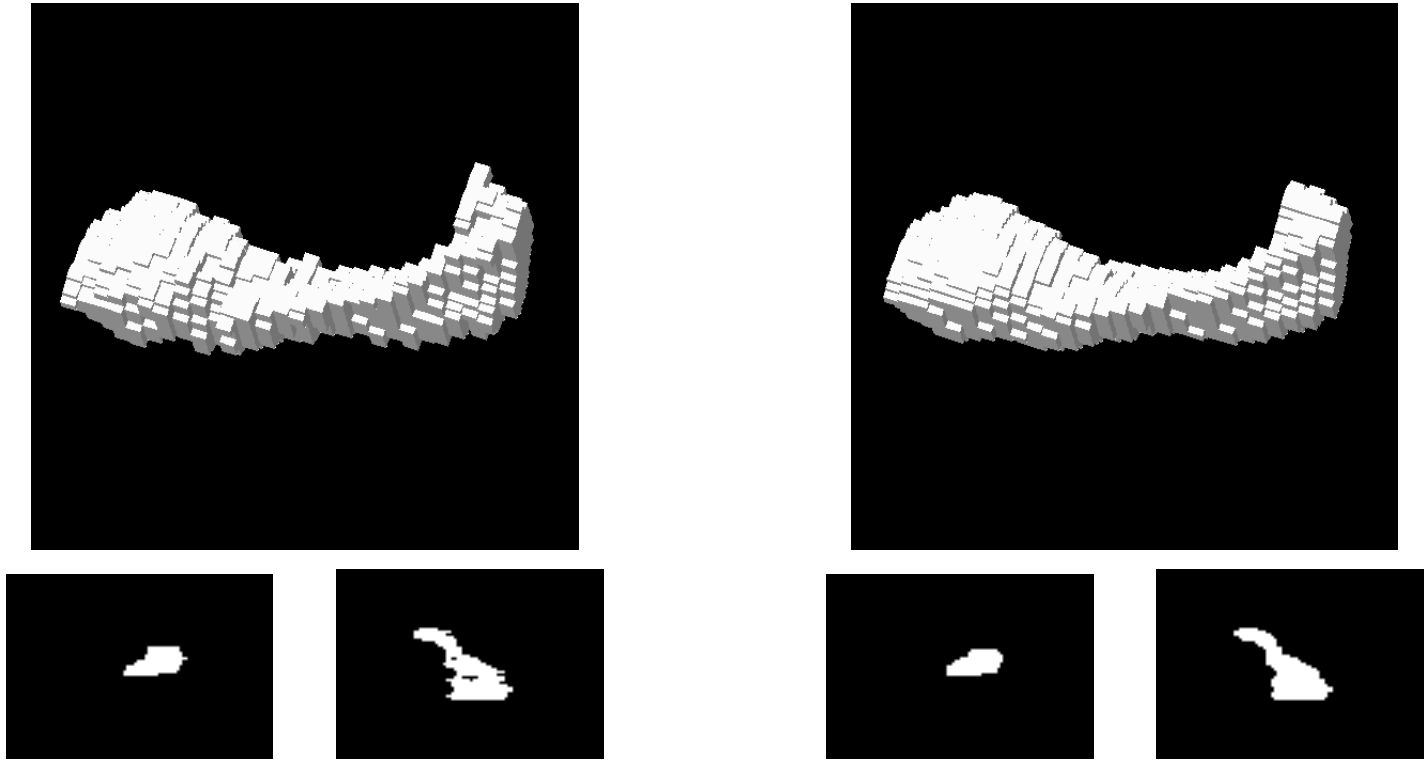
- Model-based segmentation of brain structures:
  Cootes and Taylor, 1995, 1998 / Chakraborty, 1996
  Delinguette, 1997 / Duncan and Staib, 1996
  Vemuri, 1996 / Kelemen, Szekely, Gerig, 1996

Automatic Segmentation: Related Work

- L.H. Staib & J.S. Duncan: 3D fourier organ models, region and gradient based segmentation.
- T.F. Cootes & C.J. Taylor: 2D statistical organ models, modelling gray level appearance, least-squares based segmentation.
- G. Szekely et al.: 2D statistical organ models, gradient based segmentation
- B. Vemuri et al.: Multiresolution stochastic hybrid shape models

Model Building

Segmented MRI data from R. Kikinis/M.Shenton, Harvard Medical School, BWH



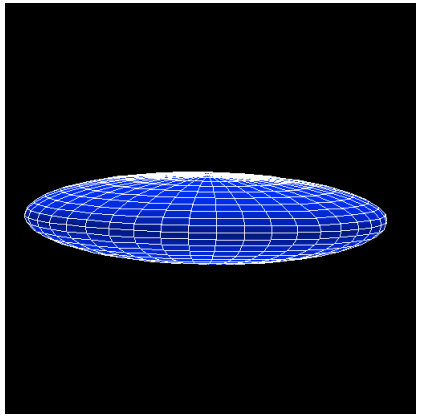
Parametrization 1

co-ordinate functions: $$\mathbf{r}(\theta,\phi) = \begin{pmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{pmatrix}$$

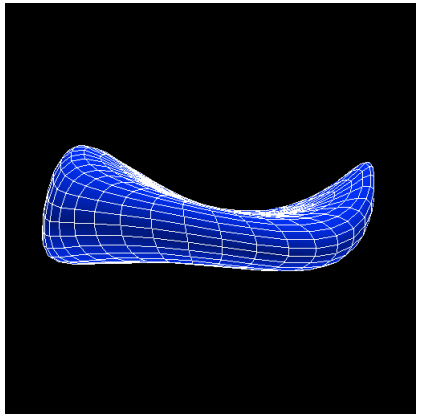
series of orthogonal basis functions: $$\mathbf{r}(\theta,\phi) = \sum_{k=0}^{K} \sum_{m=-k}^{k} \mathbf{c}_k^m Y_k^m(\theta,\phi)$$

descriptors: $$\mathbf{c}_k^m = \begin{pmatrix} c_{x_k}^m \\ c_{y_k}^m \\ c_{z_k}^m \end{pmatrix}$$

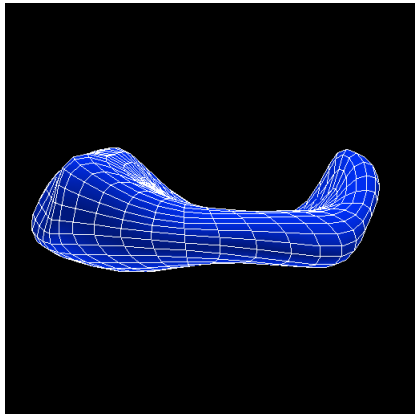
Parametrization 2



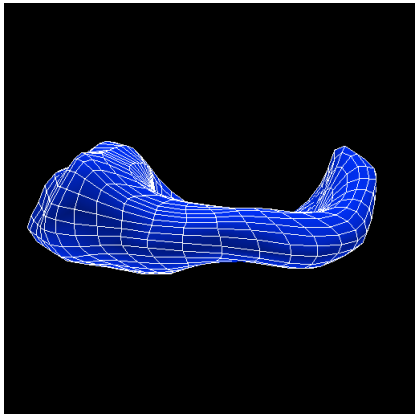
1 Harmonic



3 Harmonics



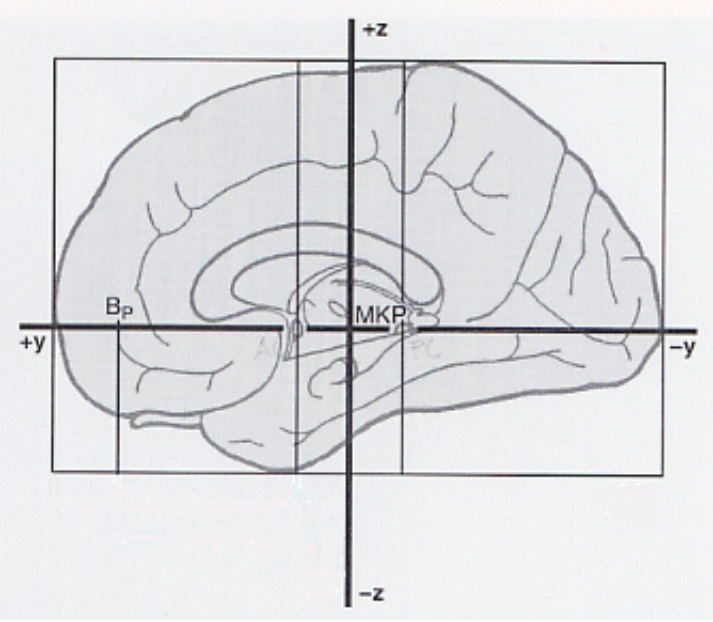
6 Harmonics

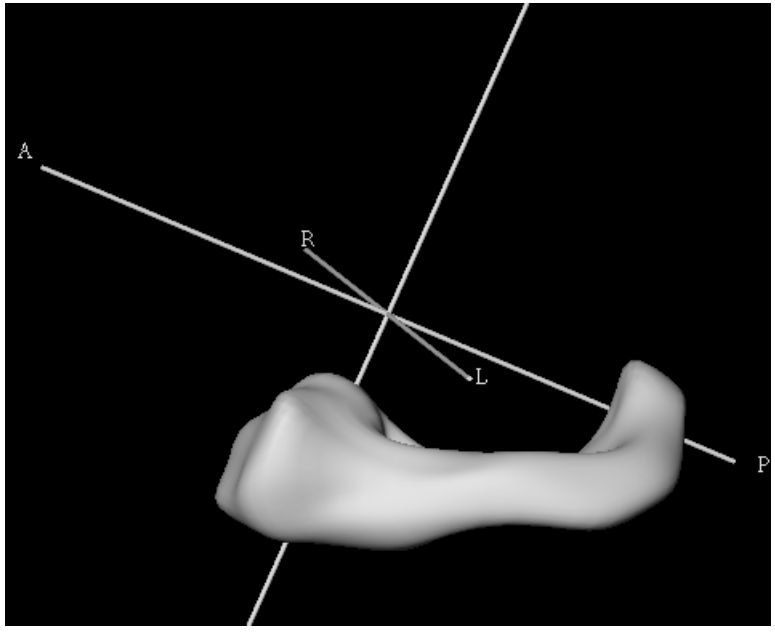


10 Harmonics

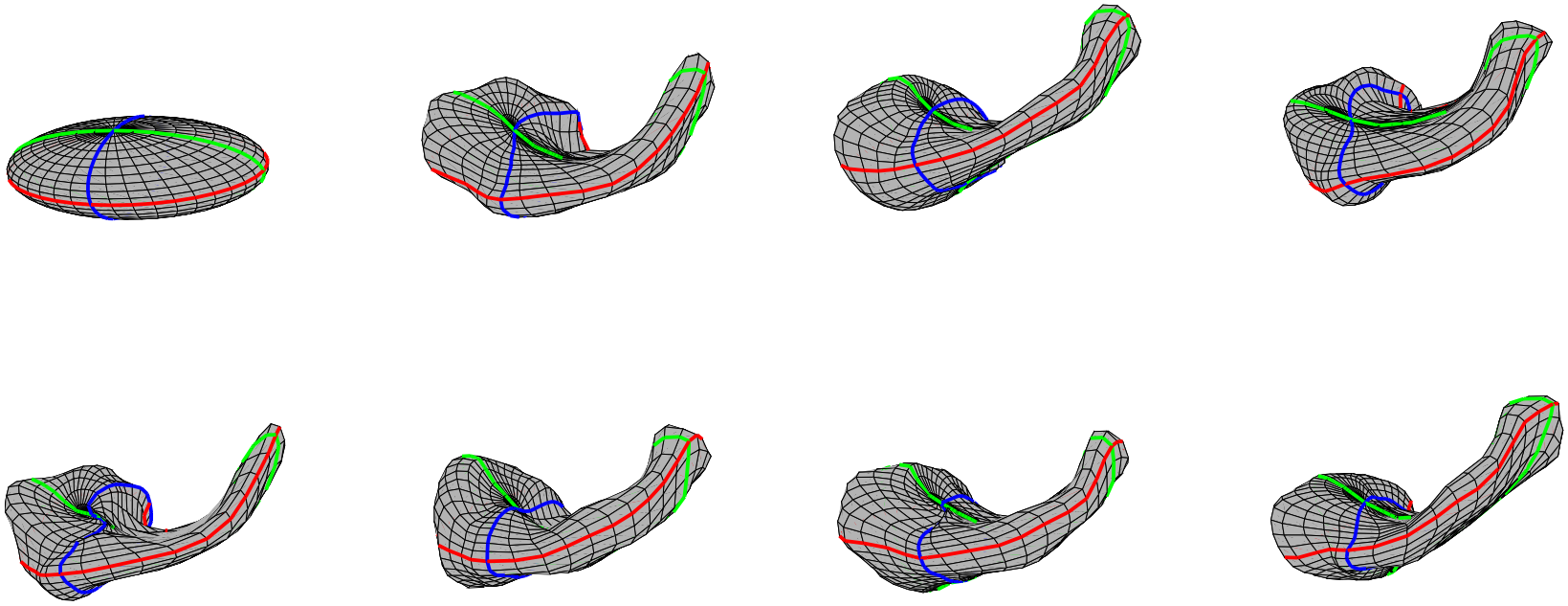
Normalization in Object Space

Normalization in Object Space: Midsagittal Plane and AC-PC Line





Normalization in Parameter Space



Computing the Statistical Model

mean model: $$\bar{\mathbf{c}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{c}_i$$

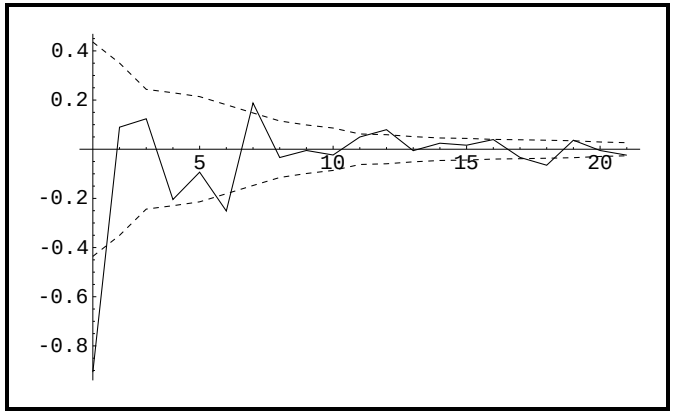
shape deviation: $$d\mathbf{c}_i = (\mathbf{c}_i - \bar{\mathbf{c}})$$

$$\Sigma = \frac{1}{N-1}\sum_i d\mathbf{c}_i \cdot d\mathbf{c}_i^T$$

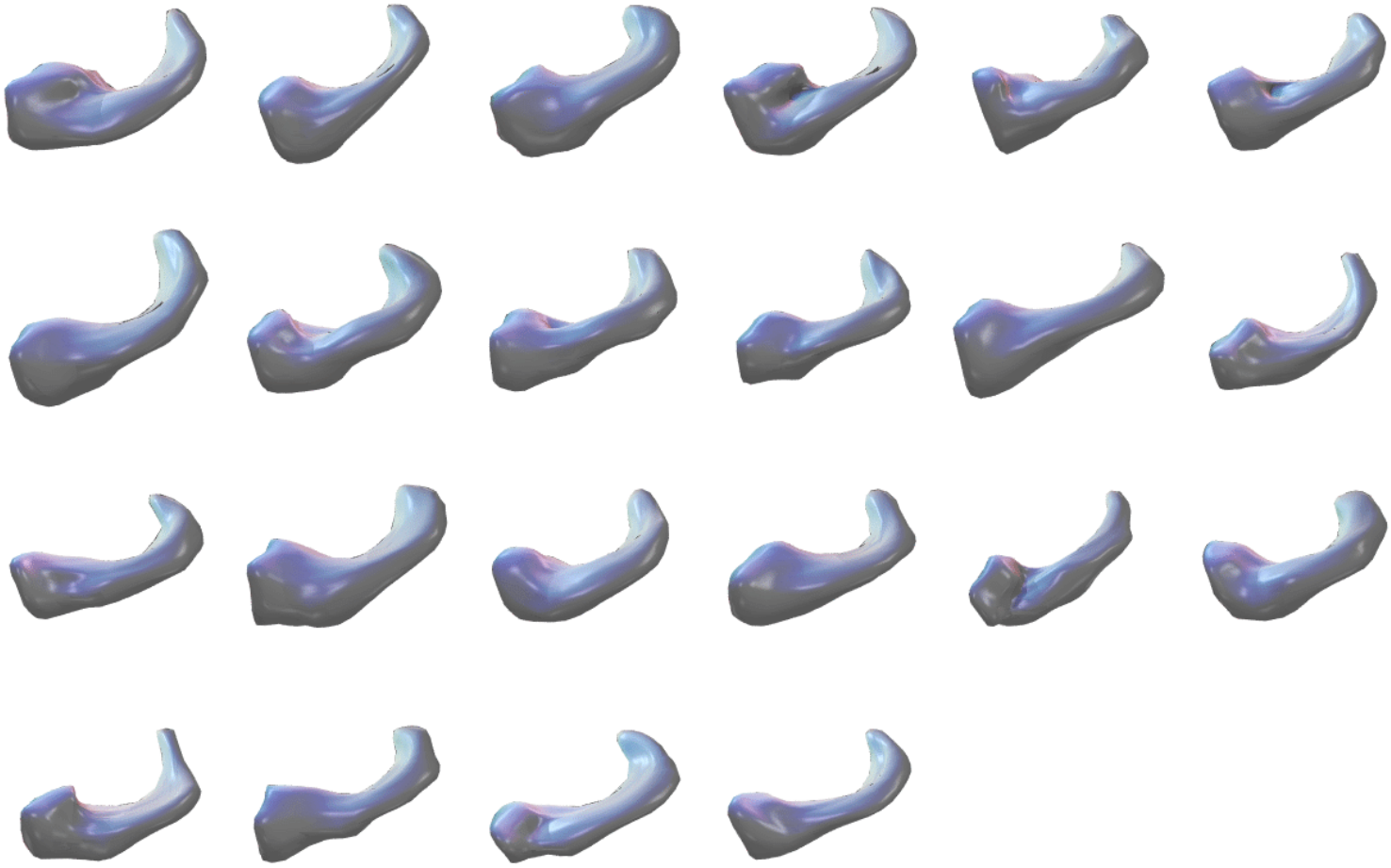
eigen analysis: $$\Sigma \cdot \mathbf{P} = \mathbf{P} \cdot \Lambda$$

individual shape: $$\mathbf{c}_i = \bar{\mathbf{c}} + \mathbf{P} \cdot \mathbf{b}_i$$

weights: $$\mathbf{b}_i = \mathbf{P}^T(\mathbf{c}_i - \bar{\mathbf{c}})$$



Individual Left-Hippocampi

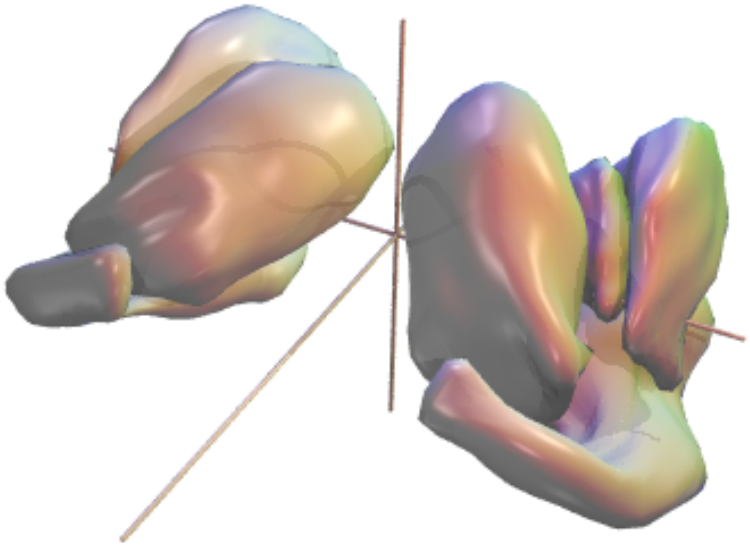


Eigendeformations of Left-Hippocampus

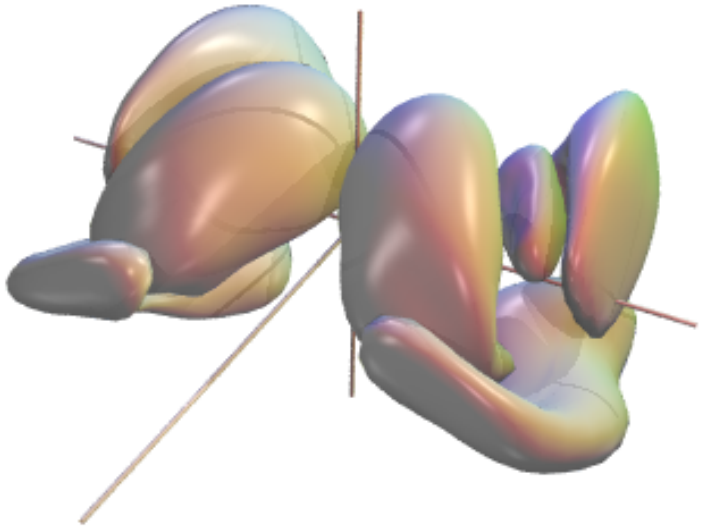
$b_1 = -2\sqrt{\lambda_1}$ $b_1 = -1\sqrt{\lambda_1}$ $b_1 = 0$ $b_1 = 1\sqrt{\lambda_1}$ $b_1 = 2\sqrt{\lambda_1}$

$b_2 = -2\sqrt{\lambda_2}$ $b_2 = -1\sqrt{\lambda_2}$ $b_2 = 0$ $b_2 = 1\sqrt{\lambda_2}$ $b_2 = 2\sqrt{\lambda_2}$

Individual and Average Brain Objects

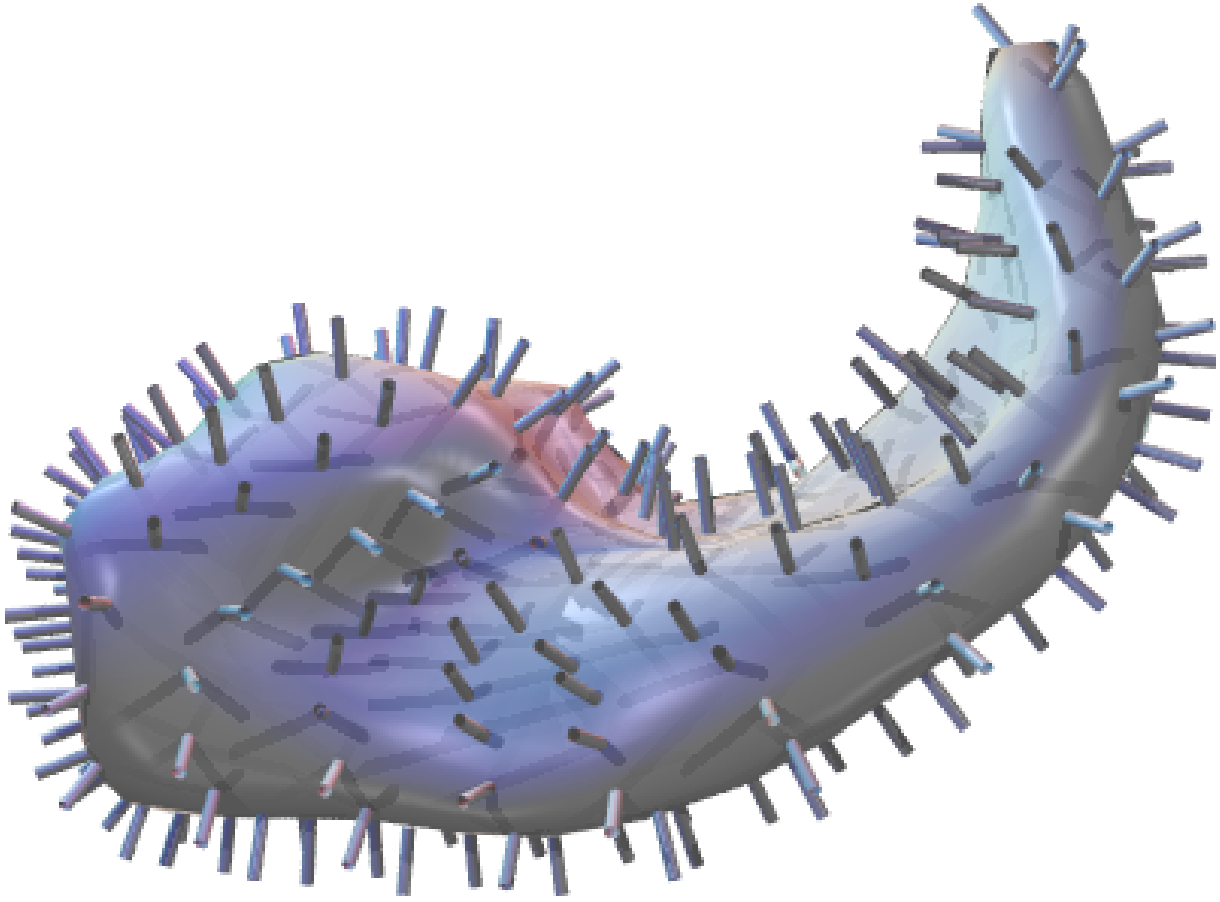


Individual Structures

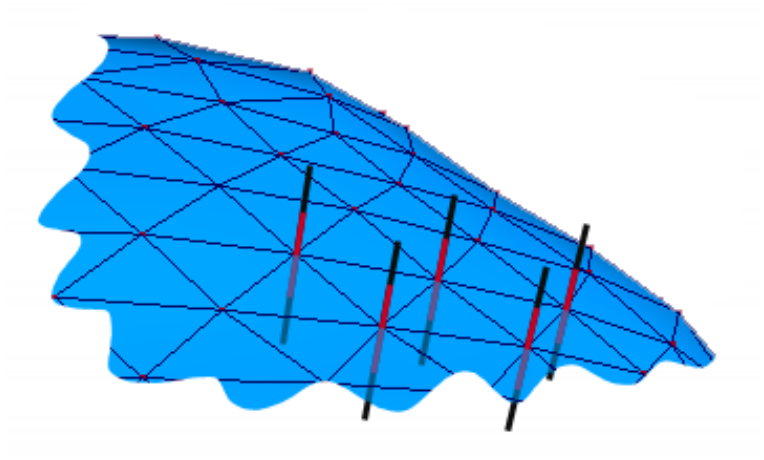


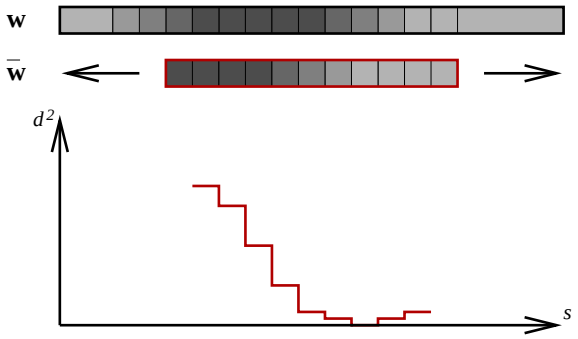
Average Models

Modelling Gray Level Appearance



Matching Profiles





$$d^2_{Maha}(s) = (\mathbf{w}(s) - \overline{\mathbf{w}})\Sigma_{\mathbf{w}}^{-1}(\mathbf{w}(s) - \overline{\mathbf{w}})$$

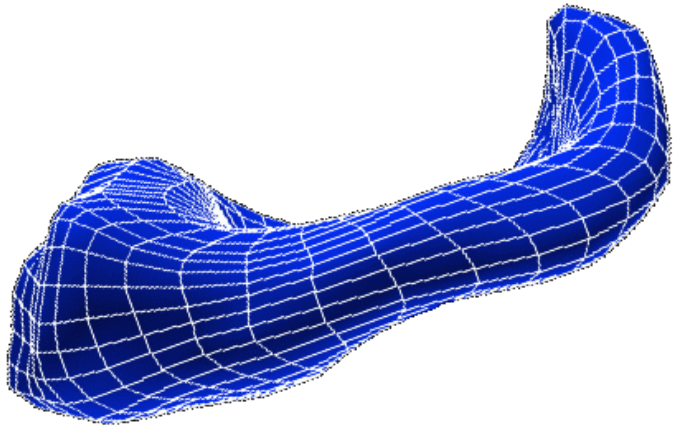
Object Representations

Surface Points:

Local Description
Regular Sampling
Position of Profiles

$$\mathbf{x} = \mathbf{A}\mathbf{c}$$

Spherical Harmonics:



Global Shape Description
Finding Correspondence
Computing Surface Normals

Computing the Fit

Parameter Space:

Statistics in spher. harm.:

$$\mathbf{c} = \bar{\mathbf{c}} + \mathbf{P_c b}$$

Multiplying by $\mathbf{A}$:

$$\mathbf{Ac} = \mathbf{A}\bar{\mathbf{c}} + \mathbf{AP_c b}$$

Object Space:

Statistics in coordinates:

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P_x b}$$

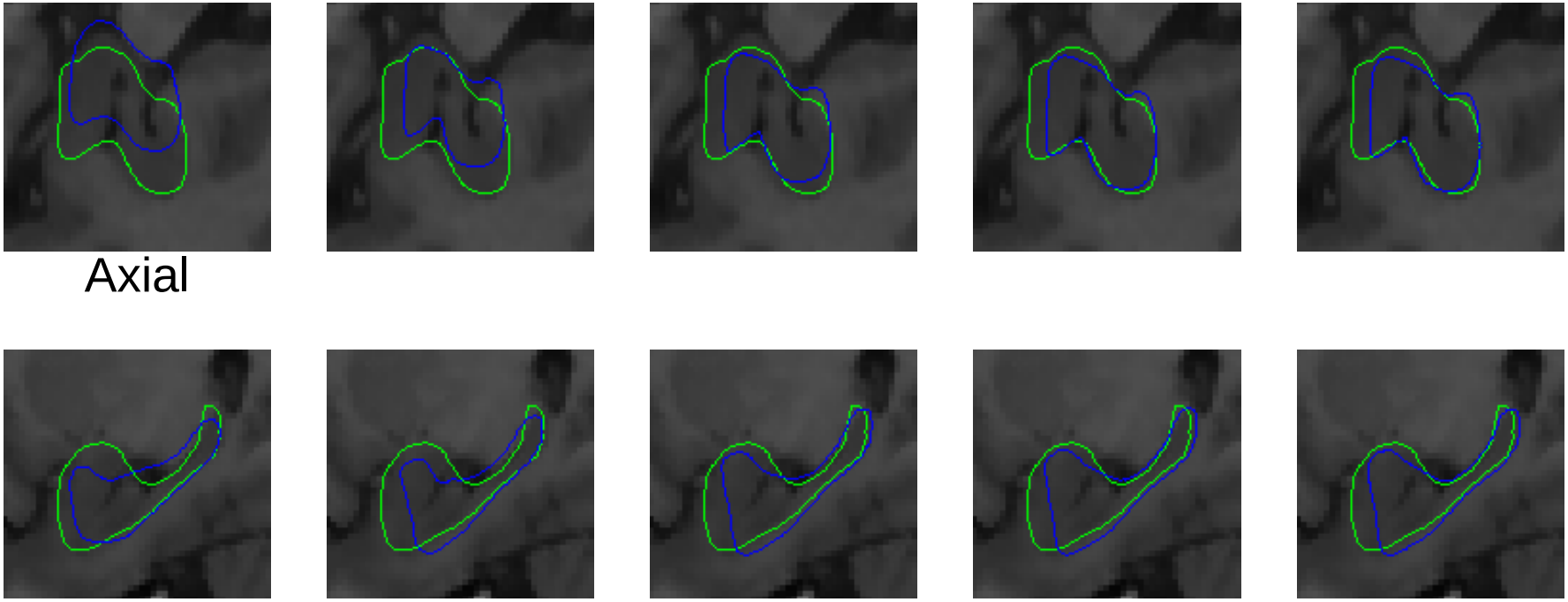
Altering coordinates with $d\mathbf{x}$:

$$(\mathbf{x} + d\mathbf{x}) = \bar{\mathbf{x}} + \mathbf{P_x}(\mathbf{b} + d\mathbf{b})$$

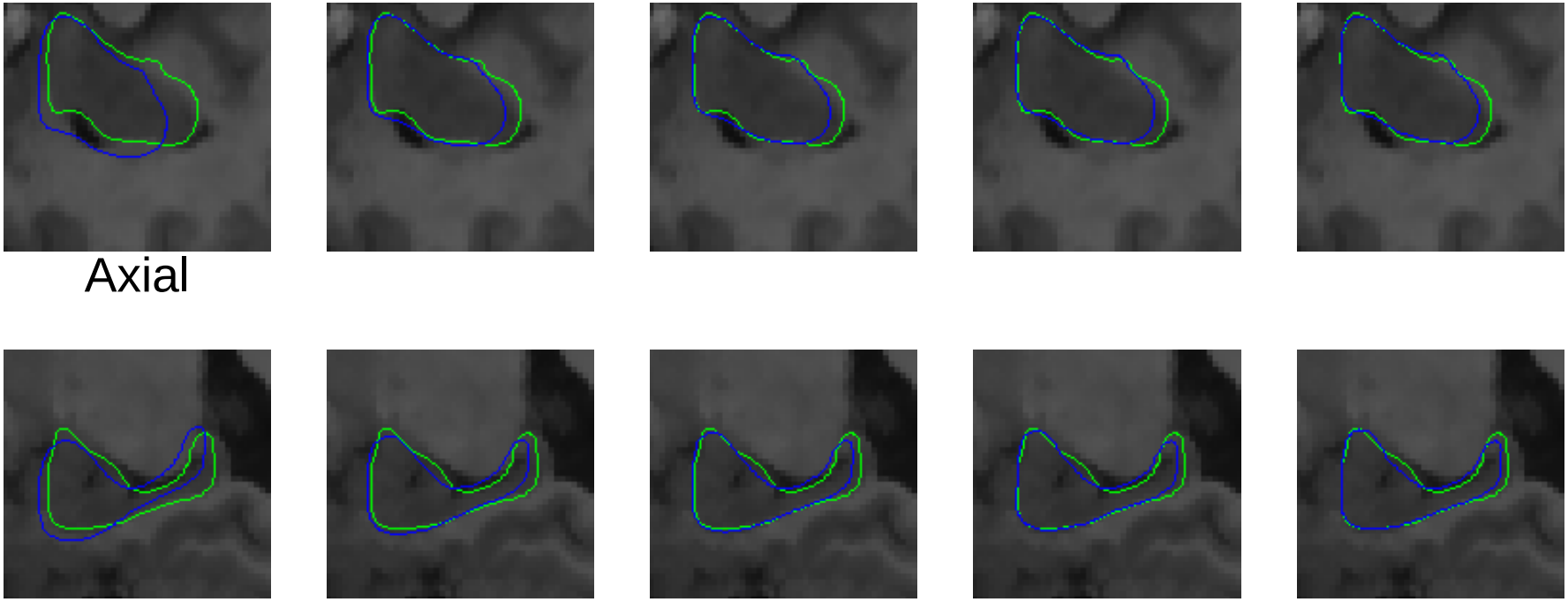
Set of eq. to solve:

$$d\mathbf{x} = \mathbf{P_x} d\mathbf{b}$$

Segmentation Results 1



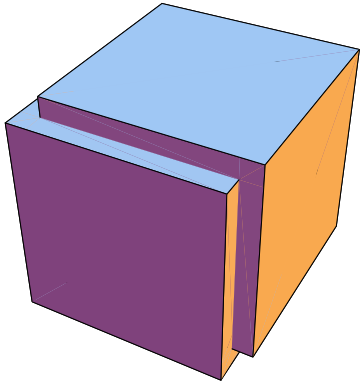
Segmentation Results 2



Validation: Volume Overlap

Test Object

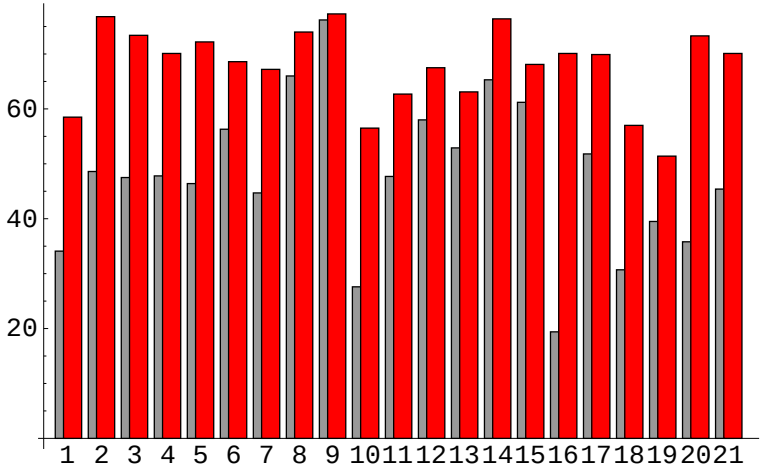
Two cubes of 10x10x10 voxels, one shifted along its space diagonal by 1 voxel:



Overlap measure = $\frac{A \cap B}{A \cup B} = 57\%$

Hippocampus

Overlap measure between manual and automatic segmentation before (gray) and after (red) deformation:



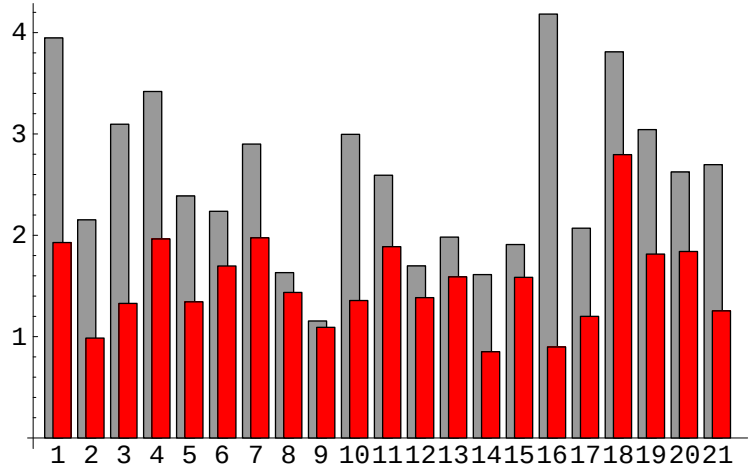
Validation: Surface Distance

Parseval's theorem:

$$\oint \| \mathbf{x}(\mathbf{u}) \|^2 \, d\mathbf{u} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} | \mathbf{c} |^2 = 4\pi \cdot \mathrm{MSD}$$

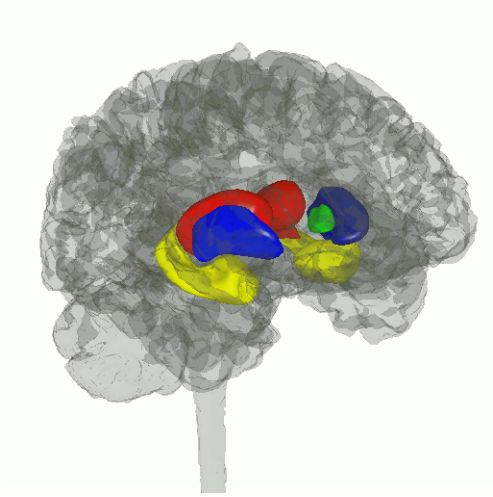
MSD stands for "mean squared distance" measured from the origin of the coordinate system.

Average distances in mm between manual and automatic segmentation before (gray) and after (red) deformation:



Conclusions

- Fully automatic 3-D segmentation
- Model includes geometry and gray-level profiles
- Statistical shape models for several brain structures
- Reproducible results
- Computation time: 2-5’ on SUN Ultra-10
- Good approximation to point-to-point correspondence
- Comprehensive tests and validation



Open Issues

Statistical Surface Model:

- Global shape model, no access to local shape properties
- No representation of figure vs. subfigures
- Simultaneous segmentation of set of anatomical shapes
- Robustness of statistics (PCA with small sample number and high dimensionality)
- Number of training shapes?
- Constrained segmentation: too restrictive?
- Point-to-point correspondence in 3-D to be improved
- Limited to simply-connected objects
- Segmentation of diseased organs?

Future Directions

- Combination of medial and surface object representation:
  3-D Voronoi skeleton and DSL model
- Generate simulated shape models for testing statistical surface shapes and for the DSL parametrization development
- Test ability of spherical harmonic model and its restricted deformation to cope with local warps as generated by DSL warps
- Apply shape segmentation and representation methods to serial 3-D image data to study progression of disease (make use of correlation)
- Study of shape parameter changes to explain shape changes over time

Elastic Model-Based Segmentation of 3-D Neuroradiological Data Sets

András Kelemen, Gábor Székely and Guido Gerig

Abstract— **This paper presents a new technique for the automatic model-based segmentation of 3-D objects from volumetric image data. The development closely follows the seminal work of Taylor and Cootes on active shape models but is based on a hierarchical parametric object description rather than a point distribution model. The segmentation system includes both the building of statistical models and the automatic segmentation of new image data sets via a restricted elastic deformation of shape models. Geometric models are derived from a sample set of image data which have been segmented by experts. The surfaces of these binary objects are converted into parametric surface representations which are normalized to get an invariant object-centered coordinate system. Surface representations are expanded into series of spherical harmonics which provide parametric descriptions of object shapes. It is shown that invariant object surface parametrization provides a good approximation to automatically determine object homology in terms of sets of corresponding sets of surface points. Gray-level information near object boundaries is represented by 1-D intensity profiles normal to the surface. Considering automatic segmentation of brain structures as our driving application, our choice of coordinates for object alignment was the well-accepted stereotactic coordinate system. Major variation of object shapes around the mean shape, also referred to as shape Eigenmodes, are calculated in shape parameter space rather than the feature space of point coordinates. Segmentation makes use of the object shape statistics by restricting possible elastic deformations into the range of the training shapes. The mean shapes are initialized in a new data set by specifying the landmarks of the stereotactic coordinate system. The model elastically deforms driven by the displacement forces across the object's surface which are generated by matching local intensity profiles. Elastical deformations are limited by setting bounds for the maximum variations in Eigenmode space. The technique has been applied to automatically segment left and right hippocampus, thalamus, putamen and globus pallidus from volumetric magnetic resonance scans taken from schizophrenia studies. The results have been validated by comparison of automatic segmentation with the results obtained by interactive expert segmentation.**

I. Introduction

SEGMENTATION of anatomical objects from large 3D medical data sets, obtained from routine Magnetic Resonance Imaging (MRI) examinations, for example, represents a necessary yet difficult issue in medical image analysis. With the steady increase of routine use of 3D imaging methods like magnetic resonance imaging (MRI), computer tomography (CT) and 3D ultrasound in radiological diagnosis, monitoring, radiotherapy and surgical planning, for example, there is a clear need for improved and efficient methods for the extraction of anatomical structures and for a description by morphometric analysis. In some limited applications, segmentation can be achieved with minimal user interaction by applying simple and efficient image processing methods, which can be applied routinely [8].

In many clinical applications such as computer assisted neurosurgery or radiotherapy planning, a large number of organs have to be identified in the radiological data sets. While a careful and time-consuming analysis may be acceptable for outlining complex pathological objects, no real justification for such a procedure can be found for the delineation of normal, healthy organs at risk.

Delineation of organ boundaries is also necessary in various types of clinical studies, where the correlation between morphological changes and therapeutical actions or clinical diagnosis has to be analyzed. In order to get statistically significant results, a large number of data sets has to be segmented. For such applications manual segmentation becomes questionable not only because of the amount of work, but also with regard to the poor reproducibility of the results. The necessity of obtaining high reproducibility and the need to increase efficiency motivates the development of computer-assisted, automated procedures.

II. Model-based segmentation of 3D radiological data

Elastically deformable contour and surface models, so-called snakes [9], have been proposed as tools for supporting manual object delineation. While such procedures can be extended to 3-D [29], [4], their initialization is a critical issue. Most often, the initial guess must be very close to the sought contour to guarantee a satisfying result [16]. An excellent overview of elastically deformable models can be found in [13]. The primary reason for the need of a precise snake initialization is the presence of disturbing attractors in the image. These attractors do not belong to the object contour but force the snake into undesired local energy minima. The procedure could become more robust if the deformation of a snake would be limited to shapes within the normal variation of a class of object boundaries.

Elastically deformable parametric models offer a straightforward way for the inclusion of prior knowledge in the image segmentation process. This is done by incorporating prior statistics to constrain the variation of the parameters of the elastic model. Such procedures have been developed by Vemuri and Radisavljevic [31] using a hybrid primitive called deformable superquadrics constructed in a multi-resolution wavelet base, or by Staib and Duncan [23] for deformable Fourier models.

For complex shapes described by a large number of of-

A. Kelemen and G. Székely are with the Image Science Laboratory, Swiss Federal Institute of Technology, ETH-Zentrum, CH-8092 Zurich, Switzerland, szekely@vision.ee.ethz.ch

G. Gerig is with the Departments of Computer Science and Psychiatry at University of North Carolina at Chapel Hill, USA

ten highly correlated parameters, the use of such priors may become tedious. The modal analysis as proposed by Pentland and Sclaroff [17] offers a promising alternative by changing the basis from the original modeling functions to the eigenmodes of the deformation matrix. The dominant part of the deformations can thus be characterized by only a few eigenmodes, substantially reducing the dimensionality of the object descriptor space. Methods using modal analysis have been successfully applied to medical image analysis by Sclaroff and Pentland [20] and Nastar and Ayache [15], for example.

Cootes *et al.* [5] combined the power of parametric deformable shape descriptors with statistical modal analysis. They use active shape models, which restrict the possible deformations using the statistics of training samples. Object shapes are described by the point distribution model (PDM) [7], [6], which represents the object outline by a subset of boundary points. There must be a one-to-one correspondence between these points in the different outlines of the training set. After normalization to size, orientation and position, they provide the basis for the statistical analysis of the object shape deformations. The mean point positions and their modes of variation (i.e. the eigenvectors corresponding to the largest eigenvalues of their covariance matrix) are used for limiting the object deformations to a reasonable linear subspace of the complete parameter space. Principal component analysis has also been used for the characterization of anatomical shape variability using other shape parametrization schemes like invariant moments [18], [19], e.g.

For a large training set containing several anatomical structures, the generation of the PDM parametrization becomes very tedious and, because of the lack of a reasonable automatization, can be a source of errors, suggesting alternative approaches for form parametrization. Staib and Duncan have already demonstrated segmentation by parametrically deformable elastic models for 2-D outlines [23] and 3-D object surfaces [22], [24] using Fourier-descriptors. In our previous work [26] we combined the statistical modal analysis with parametrization based on 2-D Fourier-descriptors. Using spatial normalization based on the generally accepted Talairach coordinate system [28] we demonstrated that fully automatic segmentation of organ contours on 2-D image slices can be achieved. In this previous paper the feasibility of a 3-D extension of this method has already been investigated. We have demonstrated that based on a general surface parametrization scheme [3] the concept can be generalized for 3-D organ surfaces with spherical topology using spherical harmonics as shape descriptors. This paper summarizes the basic concepts of the newly developed 3-D segmentation system and also presents evaluation results using a collection of 22 volumetric MR brain data sets.

The 3-D segmentation discussed here is based on statistical shape models generated from a collection of manually segmented MR image data sets of different subjects. The process can be divided into two major phases; a model-building stage, and the automatic segmentation of large series of data sets.

- In the training phase, the results of interactive segmentation of sample data sets are used to create a statistical shape model which describes the average as well as the major linear variation modes.
- The model is placed into new, unknown data sets and is elastically deformed to optimally fit the measured data.

The generation of the statistical model will be discussed in detail in the following sections. The purely geometrical statistical model proposed in our earlier paper [26] has been extended by incorporating gray-valued profiles across the organ surface, implementing the concept proposed by Cootes and Taylor [5], [6] for 3-D models.

The matching process is initialized using the average geometrical model resulting from this training phase. A two-stage algorithm, described in section VII, is used to deform this model to optimally fit the features of a new data set, while still restricting the deformations to the variability allowed by the statistical model. This algorithm makes full use of the gray-value profiles normal to the surface, which is efficiently calculated by using a dual representation of the object both as a collection of sample points and as a parametrized surface.

III. 3D object model

A. Training set

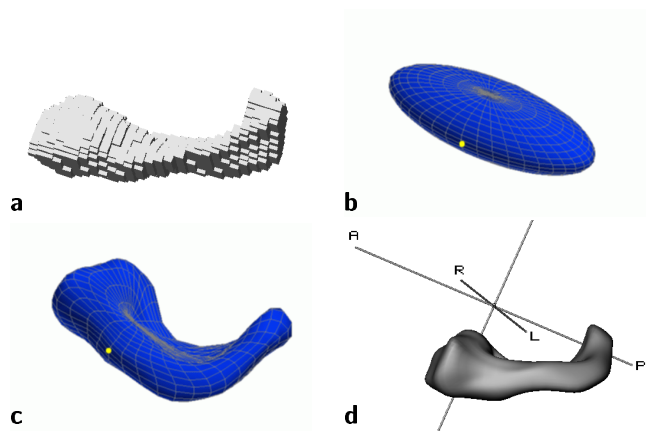


Fig. 1. Model building: interactive segmentation of a left hippocampus (**a**), reconstruction from surface shape descriptors up to degree one (**b**), reconstruction up to degree ten (**c**) and normalization of the shape pose in object space (**d**).

Today's routine practice for 3-D segmentation involves slice-by-slice manual processing of high-resolution volume data. Working on large series of similar scans, human observers knowlegeable in anatomy become experts and produce reliable segmentation results, although at the cost of a considerable amount of time per data set. Realistic figures are several hours to one day per volume data set for only a small set of structures. Regions in 2-D image slices corresponding to cross-sections of 3-D objects are outlined and painted by different interactive tools performing delineation purely manually [25] or partially supported by image data using energy minimization schemes discussed

above. The series of binary regions segmented from consecutive slices form volumetric voxel objects. Figure 1.a illustrates the result of an expert segmentation of the left hippocampus from a magnetic resonance head dataset.

Our initial training set consisting of 30 male brain MR volumes, a courtesy of the Harvard Medical School in Boston, has been processed this way. These datasets have been acquired and deep gray matter structures have been processed in the frame of a comparative psychiatric study [21]. In each volume six brain regions have been manually labeled in both hemispheres, these regions include the *amygdala/hippocampus complex*, the *parahippocampal gyrus*, *thalamus*, *caudate nucleus*, *putamen*, and *globus pallidus*. Figure 6 compares the segmented four different brain objects for an individual case with the average models.

In order to demonstrate some model building aspects which require a larger training set, we will also refer to a set of 71 *corpus callosum* outlines, a courtesy of the European BIOMORPH project.

B. Parametric shape representation

In the following the term "*parametrization*" will be used in two different ways. On the one hand, the process which maps two parameter values (s,t) to each point on a surface is called *surface parametrization*. These surface mappings parametrize an object shape with respect to surface coordinates $\boldsymbol{v}(s,t)$:

$$\boldsymbol{v}(s,t) = \begin{pmatrix} v_1(s,t) \\ v_2(s,t) \\ v_3(s,t) \end{pmatrix} = \begin{pmatrix} x(s,t) \\ y(s,t) \\ z(s,t) \end{pmatrix} \tag{1}$$

To make a clear distinction, *surface parametrization* will refer to this mapping procedure. In the following we will consider only surfaces with spherical topology, which is true for a broad class of anatomically interesting organ boundaries, in particular for all studied structures of the basal ganglia. Such surfaces can be parametrized by two polar variables (θ and ϕ) and therefore defined by three explicit functions over those:

$$\boldsymbol{v}(\theta,\phi) = \begin{pmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{pmatrix} \quad . \tag{2}$$

Secondly, *shape parametrization* denotes the computation of object shape descriptors parametrizing these coordinate functions. One possibility is an expansion over a complete set of basis functions. With respect to computing elastic shape deformations, the choice of basis functions is not critical. B-splines or wavelets could be used as well as other local representations. As discussed later, shape correspondence among multiple individuals is obtained by rotating a (θ,ϕ) parameter net over the object surface to a canonical position based on global surface parametrization. We therefore make use of the hierarchical shape representation offered by spherical harmonics, resulting in the following (truncated) series expansion

$$\boldsymbol{v}(\theta,\phi,\boldsymbol{p}) = \sum_{k=0}^{K} \sum_{m=-k}^{k} \boldsymbol{c}_k^m \; Y_k^m(\theta,\phi) \quad , \tag{3}$$

where

$$\boldsymbol{c}_k^m = \begin{pmatrix} {c_x}_k^m \\ {c_y}_k^m \\ {c_z}_k^m \end{pmatrix} \quad . \tag{4}$$

The coefficients $\boldsymbol{c}_l^m$ are three-dimensional vectors with components ${c_x}_l^m$, ${c_y}_l^m$ and ${c_z}_l^m$ with degree l and order m. A detailed description can be found in Brechbühler *et al.* [3]. All the $\boldsymbol{c}_l^m$ with components (x,y,z) define the shape description vector

$$\begin{aligned} \boldsymbol{p} \;=\; & (c_{x_0}^0, c_{y_0}^0, c_{z_0}^0, c_{x_1}^{-1}, c_{x_1}^0, c_{x_1}^1, c_{y_1}^{-1}, c_{y_1}^0, c_{y_1}^1, \\ & c_{z_1}^{-1}, c_{z_1}^0, c_{z_1}^1, \ldots c_{x_K}^{-K} \ldots c_{z_K}^K)^\top \end{aligned}$$

Figures 1.b and 1.c illustrate the hierarchical property of spherical harmonics: reconstructing the shape from coefficients up to degree 1 results in an ellipsoid. Incorporating more descriptors (up to degree 10 in Fig. 1.c) increases the level of details and more closely approximates the original shape.

C. Anatomic reference coordinate system

Our driving application is the automatic segmentation of deep gray matter structures of the human brain. We begin by choosing the standard stereotactic coordinate system proposed by Talairach for global alignment of the head image data sets. Basic features used for alignment are the approximation of the inter-hemispheric fissure by a mid-sagittal plane and the definition of the anterior and posterior commissure (AC-PC) (see Fig. 2). The identification of the symmetry plane of the brain and the position of the AC-PC line is performed manually by selecting reference points on 2-D slices of the volumetric images. Each data set is transformed into canonical coordinates by 3-D rotation and scaling as illustrated in Fig. 1.d.

In comparison to a fully object-centered spatial normalization the segmentation method described in this paper can incorporate small deviations of translation and orientation into the shape statistics. This allows us to reproducibly define a global coordinate system based on a small set of significant external landmarks for initializing shape models at their most likely positions.

In our earlier work [26] we found that in medical images there is no real justification for separating similarity transformation from shape deformation because of the strong correlation between the position and shape of organs. Accordingly, for images representing anatomy the *relative position, rotation, and size* of healthy organs is restricted in a similar and correlated way to *their elastic deformation*. We therefore introduced models incorporating this full biological variability with respect to a natural reference sytstem, which resulted in a much more robust segmentation process

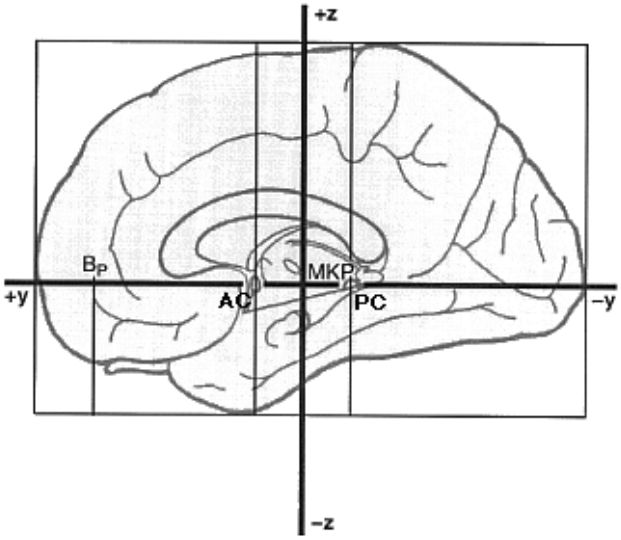


Fig. 2. Stereotactic coordinate system used for object space normalization

in 2D. The Talairach reference used here is a straightforward extension to our aerlier AC-PC based 2-D coordinate system.

IV. Correspondence by parametrization

During the present study we established surface correspondence between the items of the training set by an area-preserving parametrization followed by the object-oriented normalization of its starting point. This fully automatic procedure, which will be described in more detail in this section, has certainly strong limitations and provides only a first step in establishing correspondence. However, preliminary 2D studies have been revealed, that corrections resulting from feature-based correspondence search based on curvature [27], e.g., are minor, and arc-length based parametrization provides a good first approximation.

The shape representation proposed in this paper results in an continuous mapping function between similar objects. This is done using a uniform parametrization of closed surfaces and by calculating an invariant object centered description (Brechbühler *et al.* [2], [3]). By sampling of the spherical reference surface this method can also be used to generate corresponding pairs of surface points.

A key step in the shape description of a surface is its mapping to the parameter space, the sphere. Any point on the surface must map to exactly one point on the sphere, and vice versa. The location on the sphere corresponding to a surface point defines the *surface parameters* of the point. It can be represented as two polar or three Cartesian coordinates, related through the bijection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sin\theta\,\cos\phi \\ \sin\theta\,\sin\phi \\ \cos\theta \end{pmatrix} .$$

Mapping a surface to the sphere assigns parameters to every surface point. The mapping must be continuous, i.e. neighboring points in one space must map to neighbors in the other space. Our approach is to achieve a correspondence between different objects by constructing a mapping that preserves areas. Based on the voxel representation, such a mapping assigns the square facets on the object surface to a portion of the surface of the unit sphere. It is not possible in general to map every surface facet to a spherical square: distortions cannot be avoided, but they should be minimal.

The surface parametrization, i.e., the embedding of the object surface graph into the surface of the unit sphere, is solved as a constrained optimization problem, looking for the optimal coordinates of all vertices [3]. However, the resulting representation of the surface by a parameter net with homogeneous cells is so far only determined up to a 3-D rotation in parameter space. Point to point correspondence of surfaces of different objects would require parameters which do not depend on the relative position of the parameter net. The object can be rotated to a canonical position in parameter space by making use of the hierarchical shape description provided by spherical harmonic descriptors. The coefficients of the spherical harmonic function of different degrees provide a measure of the spatial frequency constituents that compose the structure. As higher frequency components are included, more detailed features of the object appear. To define a standard position we only consider the contribution of the spherical harmonics of degree one, which is an ellipsoid representing the coarse elongation of the object in 3-D space. We rotate the parameter space so that the north pole ($\theta = 0$) will be at one end of the shortest main axis, and the point where the zero meridian ($\phi = 0$) crosses the equator ($\theta = \pi/2$) is at one end of the longest main axis. Figures 1.b and 1.c illustrate the location of the middle main axis on the reconstruction up to degree one and ten respectively.

Objects of similar shape will get a standard parametrization which becomes comparable, i.e., parameter coordinates (θ, ϕ) are located in similar regions of the object shape across the set of objects (see Figure 3). Corresponding points on different object surfaces are therefore found by calculating a canonical parametrization rather than by interactive selection of labeled sets of 3-D points.

The normalization techniques described here require the precondition that coefficients of degree 1 represent a real ellipsoid. If, however, the ellipsoid degenerates to an ellipsoid of revolution or a sphere, the technique will fail to derive stabile main axes. Objects of higher symmetries, such as regular polygons and polyhedra are a good example for the limitations of the normalization technique.

V. Capturing statistical information of shape

After transformation to canonical coordinates, the object descriptors are related to the same reference system and can be directly compared. An existing procedure for describing a class of objects follows our 2-D method as described in Székely *et al.* [26], where the calculations are carried out in the domain of shape descriptors rather than the Cartesian coordinates of points in object space.

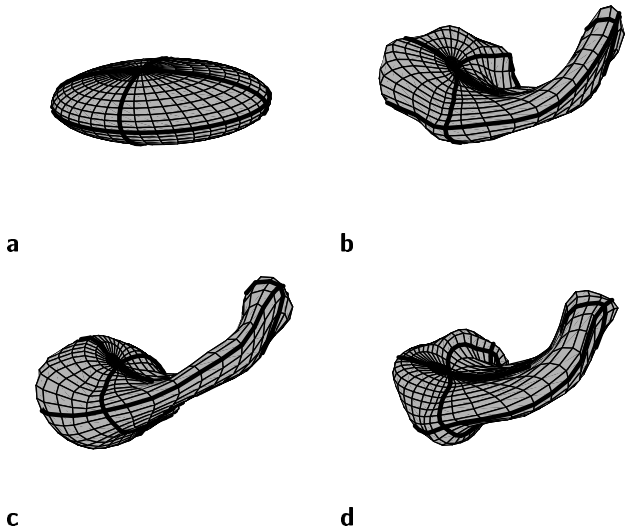


Fig. 3. Corresponding parameter values for $\theta = \pi/2$, $\phi = 0, \pi$, and $\phi = \pi/2, 3\pi/2$ (thick lines) illustrated on an ellipsoid (a) and on three individual left hippocampal structures.

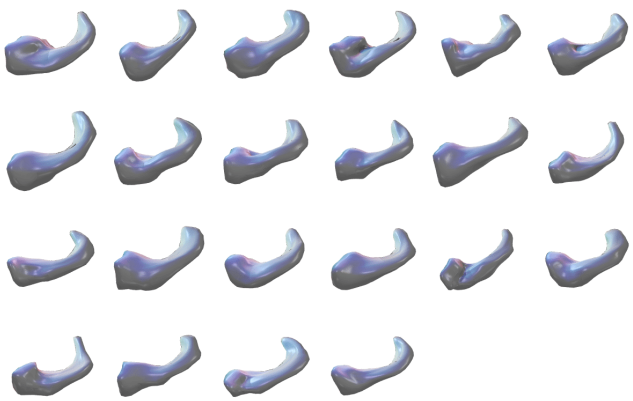


Fig. 4. Illustration of all 22 left hippocampal structures of the training sets, normalized and reconstructed from their descriptors.

A. Principal component analysis of a set of shapes

The mean model is determined by averaging the descriptors $\boldsymbol{c}_j$ of the N individual shapes (see Fig. 4).

$$\bar{\boldsymbol{c}} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{c}_j \tag{5}$$

Eigenanalysis of the covariance matrix $\boldsymbol{\Sigma}$ results in eigenvalues and eigenvectors representing the significant modes of shape variation.

$$\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{j}(\boldsymbol{c}_j - \bar{\boldsymbol{c}}) \cdot (\boldsymbol{c}_j - \bar{\boldsymbol{c}})^T \tag{6}$$

$$\boldsymbol{\Sigma}\boldsymbol{P}_c = \boldsymbol{\Lambda}\boldsymbol{P}_c \ , \tag{7}$$

where the columns of $\boldsymbol{P_c}$ hold the eigenvectors and the diagonal matrix $\boldsymbol{\Lambda}$ the eigenvalues λ_j of Σ. Vectors $\mathbf{b}_j$ describe the deviation of individual shapes $\boldsymbol{c}_j$ from the mean shape using weights in eigenvector space, and are given below

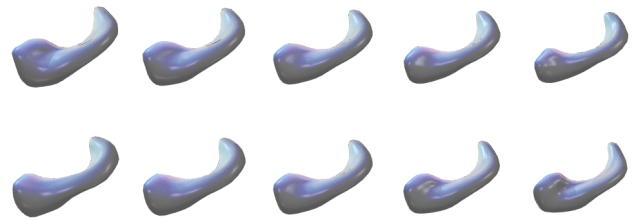


Fig. 5. Largest two modes of variation for $b_j = -2\sqrt{\lambda_j} \ldots 2\sqrt{\lambda_j}$. In the middle column, $b_j = 0$ represents the mean model.

$$\boldsymbol{c}_j = \bar{\boldsymbol{c}} + \boldsymbol{P}_c \boldsymbol{b}_j \tag{8}$$

Figure 5 illustrates the largest two eigenmodes of the hippocampus training set, while Figure 7 depicts the square root of eigenvalues sorted by size (dotted line) together with components of one individual vector $\boldsymbol{b}_j$. As after the first few eigenvectors the variance becomes very small, the first t largest eigenmodes have been taken for building a flexible model that explains the biological variability of the hippocampal shape. Any shape in this linear subspace is approximated by combining the mean shape and a weighted sum of the deviations obtained from the first few modes

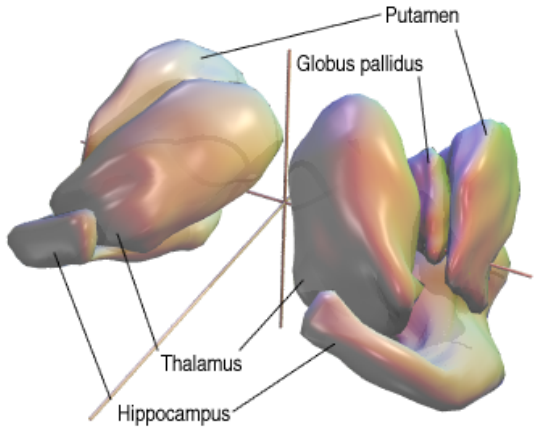
$$\boldsymbol{c} = \bar{\boldsymbol{c}} + \boldsymbol{P}_c \boldsymbol{b} \ , \tag{9}$$

where $\boldsymbol{b}$ is a vector of weights, one for each eigenvector, and since eigenvectors are orthogonal, $\boldsymbol{P}_c^T \boldsymbol{P}_c = \boldsymbol{I}$, $\boldsymbol{b}$ of a given shape $\boldsymbol{c}$ can be computed using

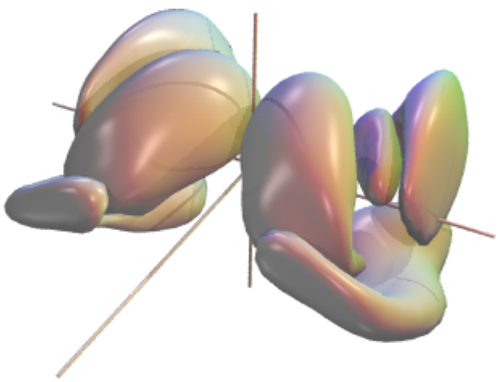
$$\boldsymbol{b} = \boldsymbol{P}_c^T (\boldsymbol{c} - \bar{\boldsymbol{c}}) \ . \tag{10}$$

The vector $\boldsymbol{b}$ can also be thought of as a new and more compact representation of the shape in the new basis of the deformation modes instead of the spherical harmonics. Eq. 10 describes how to generate new examples of the shapes by varying the parameters $\boldsymbol{b}$ within suitable limits, so that the new shapes remain similar to those in the training set. The limits for each element of $\boldsymbol{b}$ are derived by examining the distributions of the parameter values required to generate the training set. If Gaussian distributions are assumed the variances of the elements of $\boldsymbol{b}$ are given by the corresponding eigenvalues.

To choose the appropriate number of eigenmodes for the shape representation the followings has to be taken into consideration. Supposing, Fourier harmonics up to degree n_l has been used, there will be $3n_l^2$ free parameters describing the shape, this results in a covariance matrix of the size $(3n_l^2) \times (3n_l^2)$ and theoretically in $3n_l^2$ different eigenmodes. However, if the training set only consists of $N < (3n_l^2)$ samples there will be only $N-1$ linearly independent columns or rows in Σ and also that many eigenmodes in $\boldsymbol{P_c}$. It fol-



a



b

Fig. 6. Left and right thalamus, globus pallidus, putamen and hippocampus in one individual case (**a**) and their average models computed from all cases (**b**).

lows that the number of modes, t, should be smaller than both N and $3n_l^2$:

$$t < \min(N, 3n_l^2) \ . \tag{11}$$

One method for calculating t is to choose the smallest number of modes such that the sum of their variances explain a sufficiently large proportion of λ_T, the total variance of all the independent variables, where

$$\lambda_T = \sum_{k=1}^{\min(N,3n_l^2)} \lambda_k \tag{12}$$

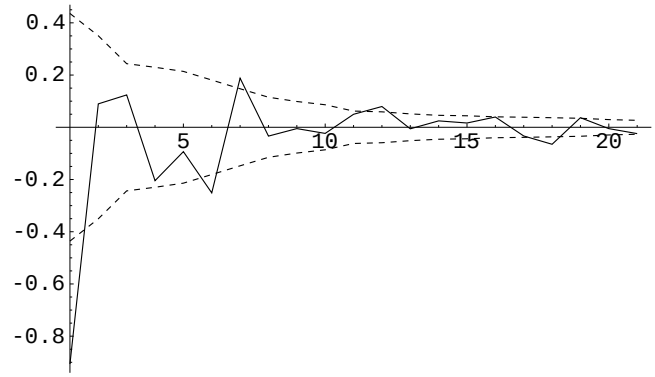


Fig. 7. Statistics of shape deformation. The dotted line represents the square root of eigenvalues $\sqrt{\lambda_j}$ sorted by decreasing size. The continuous line illustrates the components of an individual vector b_j, which describes the deviation of the shape $\boldsymbol{c}_j$ from the mean shape $\bar{\boldsymbol{c}}$.

and λ_k is the kth diagonal element of $\boldsymbol{\Lambda}$.

Neglecting eigenmodes corresponding to small eigenvalues is only reasonable if the shape variation is globally distributed along the whole object surface. Some diseases are thought to be correlated with very small localized anatomical differences. Eigenmodes describing highly localized (but still significant) variations should not therefore be discarded, even if the corresponding eigenvalue is small. Thus, choosing eigenmodes related to maximal local deformations results in a better representation of the training set. To investigate the influence of local deformations on the selected eigenmodes we have resampled the surface of our models, and characterized each deformation mode by the surface point with the largest deviation d_k. For our models both sorting criteria, λ_k and d_k, produced identical ordering of deformation modes.

B. Validation of statistical models

In the example above 22 samples of *hippocampi* have been used to derive a statistical model. Applying equation 12 we find that the 10 largest eigenvalues express 99% of the variation represented in the training set. It is important to note, that this does not correspond to the true anatomical variability of the organ shape, just that of the limited representation provided by the selected parametric shape descriptors. Accordingly, the *parametrized 22 shapes* of the training set can be described with minor error using the model, but no information is provided for shapes not included in the initial population.

The description error of a shape not included in the training set can be computed by first projecting its descriptors $\boldsymbol{c}$ into the subspace of the major eigenvectors (eq. 10), then approximating coefficients $\tilde{\boldsymbol{c}}$ from the projection (eq. 9), and finally comparing $\boldsymbol{c}$ and $\tilde{\boldsymbol{c}}$. The difference between $\boldsymbol{c}$ and $\tilde{\boldsymbol{c}}$ is given by the Euclidean distance of the two vectors:

$$\epsilon = D_{\mathrm{Eucl}}(\boldsymbol{c}, \tilde{\boldsymbol{c}}) \, . \tag{13}$$

It has to be mentioned that the value of ϵ is an absolute measure of shape similarity in an *abstract, object-dependent*

parameter space and consequently allows comparisons only between representations of the same, but not of different organs.

To demonstrate the predictive ability of the statistical model, we investigated, how the quality of the model increases while incorporating more individuals in the training set. We first have built a statistical model using 11 arbitrarily chosen shapes out of the entire set, as needed to be able to compute 10 deformation modes, and determine the above measure of segmentation error for the remaining shapes not included in the training set. Repeating the computations for a statistically significant subset of the [22 choose 11] combinations of the entire set, and finally averaging the errors, we obtain a measure for the 11-shape model as shown in Figures 8.a and 8.b by the first data points. The size of the training set is then increased one-by-one and the average error is computed based on segmentation results using the largest 10 eigenmodes to obtain the rest of the data points in the same figure. One can observe that the average error decreases as the model grows.

While this decrease of the error is significant in practically all cases, for our 3D training sets even the insertion of the last item leads to a significant improvement of the model. As a comparison, a similar experiment for the statistical model generated from 71 2-D outlines of the *corpus callosum* on midsaggital brain slices has been performed. The result is shown on Figure 8.c. One can clearly see, that in this case the model becomes saturated and the addition of a certain number new shapes to the training set does not increase the models's information content. On the other hand, adding new individuals to our 3-D models could further increase their ability to describe unseen shapes of the same kind. This apparent deficiency of our 3D models also explains the difference in segmentation quality we only experience in 3-D between shapes included and not included in the training set, as discussed later on in section VIII.

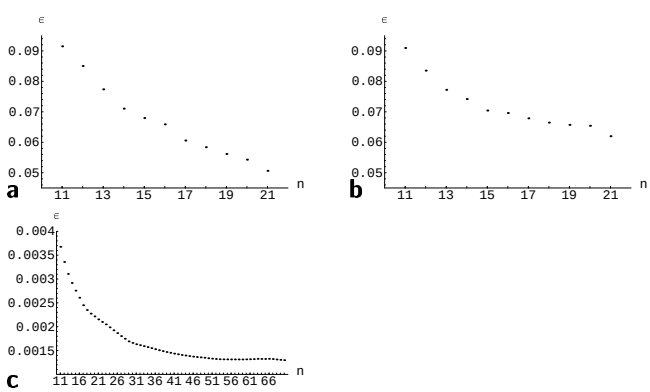


Fig. 8. Predictive ability of the models derived from the training sets of 22 *hippocampi* in the left (**a**) and right (**b**) hemisphere, and the same performance curve computed for the training set containing 71 *corpora callosa* (**c**). It displays the average representation error ϵ for shapes not included in the training set when using the 10 largest eigenmodes while increasing the number of shapes used for the generation of the statistical model from $n = 11$ to $n = N - 1$, where N is the number of available shapes for the organ in question.

VI. Modeling grey-level environment

Organ geometry represents only a part of the full information provided by the original volumetric image data sets. Besides organ shape, radiological interpretation heavily relies on local brightness and contrast information. Previous work clearly demonstrated that augmenting geometric models with information about the gray level environment of the model surface significantly improves the robustness of the segmentation ([6]). Therefore, we examine the statistics of the image intensity along 1-D profiles orthogonal to the object surface at a discrete set of sampling points.

A. Sampling of model surface

Equal processing of each part of the model surface is ensured by choosing a homogeneous distribution of sampling points and profiles over the surface parameter space. A perfectly regular sampling of a spherical surface does not exist, but we can find a good approximation by icosahedron subdivision, a technique often used in computer graphics to triangulate and display spheres at different scales. The algorithm takes an icosahedron inscribed in a sphere, and subdivides its faces as shown in Figure 9. The newly introduced vertices lie slightly inside the sphere, so we push them to the surface by properly normalizing their distance to the center to unity.

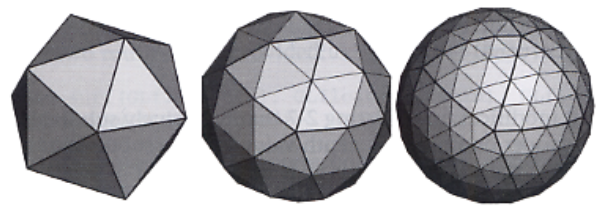


Fig. 9. Nearly regular sampling of spherical surfaces by icosahedron subdivision.

The necessary level of subdivision depends of course on the size of the object. For the hippocampus structures analyzed in this paper a subdivision of $k = 10$ resulted in a sampling distance of about 1 voxel on the surface. This subdivision results in $n = 12 + 30(k-1) + 20\frac{(k-1)(k-2)}{2} = 1002$ vertices. Computing the θ_i and ϕ_i values at each vertex coordinate i of the subdivided icosahedron and substituting them into

$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \sum_{l=0}^{K} \sum_{m=-l}^{l} \boldsymbol{c}_l^m Y_l^m(\theta_i, \phi_i), \tag{14}$$

$$i = 1 \ldots 1002 \tag{15}$$

we obtain a dual description of the object surface by the coordinates of a set of surface points $\boldsymbol{x}_i$. The equation above can be written in a more compact matrix form as

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{c}, \tag{16}$$

where $\boldsymbol{x}$ represents the coordinates in object space and $\boldsymbol{c}$ the spherical harmonics descriptors. $\boldsymbol{A}$ consists of the function values of $Y_l^m(\theta_i, \phi_i)$, one for each dimension, and describes the mapping between shape description space and object space coordinates.

For every surface point i in each data set j we can extract a profile $\boldsymbol{w}_{ij}$ of n_p sample points. The distance between sample points is the length of one voxel. The profiles are oriented normal to the object surface and centered at the surface points $\boldsymbol{x}_{ij}$, as illustrated in Figure 10. For each sample point i we can obtain a mean profile by averaging over the sample objects N:

$$\bar{\boldsymbol{w}}_i = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{w}_{ij}. \tag{17}$$

We calculate a $n_p \times n_p$ covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}_i}$ which gives us a statistical description of the expected profiles at each sample point:

$$\boldsymbol{\Sigma}_{\boldsymbol{w}_i} = \frac{1}{N-1}\sum_{j=1}^{N}(\boldsymbol{w}_{ij} - \bar{\boldsymbol{w}}_i)(\boldsymbol{w}_{ij} - \bar{\boldsymbol{w}}_i)^T \ . \tag{18}$$

Cootes et al. in [6] propose normalized derivative profiles giving invariance to uniform scaling of gray levels and constant shift. For our applications, however, we achieved best results using unnormalized original gray level profiles, as all our data sets have been acquired under the same imaging conditions. This allows us to avoid the information loss caused by any normalization procedure.

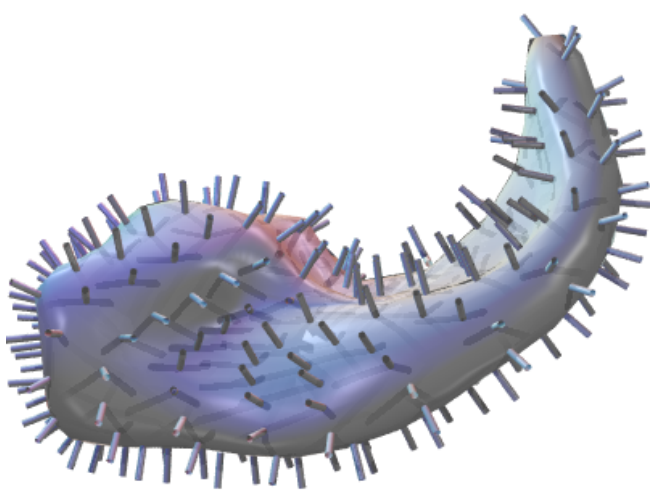


Fig. 10. Illustration of an individual left hippocampal shape with its profile vectors shown from the left side of the brain.

B. Dual surface representation

The points of the sampled surface can be considered as a new representation of the same object which can be obtained from the *spherical harmonic* descriptors by the linear transformation described by Equation (16). The deformation modes have been previously derived based on the *spherical harmonic* coefficients (referred as parameter space). To examine how the modes can be converted to the description of the surface based on sample points (referred as object space), we investigate the covariance matrix. In parameter space it is defined by

$$\boldsymbol{\Sigma}_c = \mathrm{Cov}[\mathrm{d}\boldsymbol{c}] = \mathrm{E}[\mathrm{d}\boldsymbol{c}\,\mathrm{d}\boldsymbol{c}^{\mathrm{T}}] \ . \tag{19}$$

Writing the same equation in object space and substituting the transformation matrix $\boldsymbol{A}$ we obtain

$$\boldsymbol{\Sigma}_x = \mathrm{Cov}[\mathrm{d}\boldsymbol{x}] = \mathrm{E}[\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{x}^{\mathrm{T}}] = \mathrm{E}[\boldsymbol{A}\mathrm{d}\boldsymbol{c}\,\mathrm{d}\boldsymbol{c}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}] = \boldsymbol{A}\boldsymbol{\Sigma}_{\mathrm{c}}\boldsymbol{A}^{\mathrm{T}} \ , \tag{20}$$

where $d\boldsymbol{x}$ denotes the deviation of an individual $\boldsymbol{x}$ from the average $\overline{\boldsymbol{x}}$ over the whole population. Performing principal component analysis on $\boldsymbol{\Sigma}_x$

$$\boldsymbol{\Sigma}_x \boldsymbol{P}_x = \boldsymbol{\Lambda}_x \boldsymbol{P}_x \ , \tag{21}$$

and substituting Equation (16) and Equation (20) into Equation (21) and multiplying both sides by $\boldsymbol{A}^T$, we obtain

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{\Sigma}_c\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{P}_c = \boldsymbol{\Lambda}_x\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{P}_c \ . \tag{22}$$

Comparing Equation (22) with Equation (7), it can be seen that $\boldsymbol{P}_c$ and $\boldsymbol{P}_x$ describe the same deformation modes if, and only if $\boldsymbol{A}^T\boldsymbol{A} = \alpha\boldsymbol{I}$, where α is a scalar and $\boldsymbol{I}$ the identity matrix. This requirement is fulfilled if $\boldsymbol{A}$ is an orthogonal matrix, furthermore if $\boldsymbol{A}$ is orthonormal then $\alpha = 1$.

In the 2-D case, columns of $\boldsymbol{A}$ are regularly sampled versions of $\cos(kt)$ and $\sin(kt)$ functions (with $k = 0 \ldots K$, and $0 \leq t < 2\pi$) which are known to be orthogonal, since they also build the orthogonal basis vectors of the discrete Fourier-transformation. It follows that eigenvectors $\boldsymbol{P}_x$ and eigenvalues $\boldsymbol{\Lambda}_x$ in object space can be easily computed from those defined in parameter space using the following equations:

$$\boldsymbol{P}_x = \boldsymbol{A}\boldsymbol{P}_c \tag{23}$$

$$\boldsymbol{\Lambda}_x = \alpha\boldsymbol{\Lambda}_c \ . \tag{24}$$

$\boldsymbol{P}_x$ in this case describes real eigenmodes. Thus, the obtained statistical shape representation is identical to *Point Distribution Models* introduced by Cootes *et al.* [6] with the exception that in our case point to point correspondence has been automatically approximated rather than manually determined.

Contrary to the special 2-D case, in 3-D matrix $\boldsymbol{A}$ is non-orthogonal because of the not perfectly regular sampling of the spherical surface. In our application it is more beneficial that corresponding deformation modes describe the same alteration in object and parameter space, than their exact orthogonality, hence we introduce quasi-eigenvectors $\boldsymbol{P_x}'$.

The shape statistics, as described in section V, can be expressed by equation (9). Multiplying both sides of this

equation by $\boldsymbol{A}$ we get the dual surface description by a set of surface points:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{A}\bar{\boldsymbol{c}} + \boldsymbol{A}\boldsymbol{P_c}\boldsymbol{b} \tag{25}$$

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{P_x}'\boldsymbol{b}\,, \tag{26}$$

where $\boldsymbol{P_x}'$ denotes the product $\boldsymbol{A}\boldsymbol{P_c}$ which represents the matrix of modes of shape variation expressed in object coordinate space. Recall that $\boldsymbol{P_c}$ is the matrix of eigenvectors in the shape descriptor space defined by the components of the elliptic harmonic descriptors $\boldsymbol{c}$. Thus, $\boldsymbol{P_x}'$ is not a real matrix of eigenvectors since its column vectors are also non-orthogonal, although they still represent the same shape deformations as eigenvectors in $\boldsymbol{P_c}$. This deviation from orthogonality can be characterized by the ratio between the average of the non-diagonal and diagonal elements of $\boldsymbol{P_x}'\boldsymbol{P_x}'^T$, which was 15.2626 in our case. Therefore, weight vectors $\boldsymbol{b}_i$ of individual shapes, which express the deviation from the mean model, remain the same in both shape representation schemes.

VII. Iterative segmentation scheme

Until now we have only described the creation of a flexible 3D model including geometric shape, gray level environment and statistics about normal shape variability. We now perform the segmentation step by elastically fitting this model to new 3D data sets. This is achieved with the following two steps:

- Initialization is done by transforming the model's coordinate system into that of the new data set.
- The elastic deformation of the surface until it best matches the new gray value environment.

A. Initialization of segmentation

Since the model has been built based on a normalization to the Talairach coordinate system, the determination of the symmetry plane of the brain and the position of the AC/PC line becomes an integral part of the initialization. Currently this is done manually but the determination of the symmetry plane and the AC/PC line by can be replaced in future by an automatic method [14], [12], [30], [11]. To derive the position of the midsagittal plane the user specifies the position of 3 or more points lying on the inter-hemispheric plane. The program computes parameters of a plane which have the best least squares fit to the given points. The more points are specified, the more robust is the fit. Having determined the plane the user finally marks the location of the AC and PC. A translation vector and a rotation matrix are computed to transform the model's coordinate system into the image space of the new data set.

B. Elastic deformation of model shape

We introduced two different representations of a surface, one based on the spherical harmonic descriptors and a second one based on the subdivided icosahedron. We attempt to use the advantages of both representations in our procedure. Spherical harmonic descriptors were necessary to find a correspondence between similar surfaces and they also allow the exact analytical computation of surface normals by

$$\boldsymbol{n}_i = \sum_{l=0}^{K}\sum_{m=-l}^{l} c_l^m \frac{\partial Y_l^m}{\partial\theta} \times \sum_{l=0}^{K}\sum_{m=-l}^{l} c_l^m \frac{\partial Y_l^m}{\partial\phi}. \tag{27}$$

However, they only represent a global description of an object shape. The surface points, on the other hand, give a local representation, which is essential to carry out an iterative refinement of the model, as will be described in the next section. Thus, we decided to keep both representations during the matching process, the relation between the two being tractable via the matrix $\boldsymbol{A}$.

C. Calculating displacements for surface points

After initialization of the surface model we calculate the displacement vector at each surface sample point which would move that point to a new position closer to the sought object. Since there is a model of a gray level profile for each point, the search tries to find an adjacent region which better matches this profile. A profile $\boldsymbol{w}$ of length $l(> n_p)$ normal to the surface is extracted at each model point. This new profile is shifted along the model profile in discrete steps s to find the position of the best match. This is given as the square of the Mahalanobis distance:

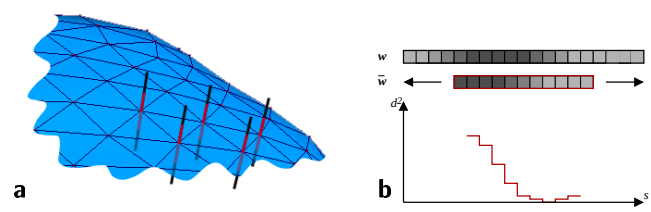


Fig. 11. Illustration of the surface matching process. Image **a** shows part of the model's triangulated surface with longer extracted (in black) and shorter model profiles (in grey). Image **b** visualizes the computation of a suggested movement for a single surface point.

$$d^2_{Maha}(s) = (\boldsymbol{w}(s) - \bar{\boldsymbol{w}})\boldsymbol{\Sigma_w}^{-1}(\boldsymbol{w}(s) - \bar{\boldsymbol{w}}) \tag{28}$$

where $\boldsymbol{w}(s)$ represents the sub-interval of the extracted profile at step s having a length of n_p. The location of the best fit is thus the one with minimal $d^2_{Maha}(s)$. Suppose s_{best} is the shift between the two profiles providing the best fit. We choose a displacement vector $\boldsymbol{dx}$ for each model point which is parallel to the profile, in the direction of the best fit and has magnitude s_{best}. Figure 11 illustrates this process.

D. Adjusting shape parameters

Having generated 3-D displacement vectors for each of the n model points

$$\boldsymbol{dx} = (dx_1, dy_1, dz_1, \ldots, dz_n), \tag{29}$$

we then adjust the shape parameters to move the model surface towards a new position. Since rotation, translation and scale are already incorporated in the model statistics, we do not have to deal with them separately. Of more concern are the calculated displacements $d\boldsymbol{x}$, as these could freely deform the shape of the object. In order to keep their resulting shape consistent with the statistical model, we restrict possible deformations by considering only the first few modes of variation. This will be solved by minimizing a sum of squares of differences between actual model point locations and the suggested new positions.

The shape statistics, as described by equation (26) represents the matrix of modes of shape variation expressed in object coordinate space. We seek an adjustment $d\boldsymbol{b}$ to $\boldsymbol{b}$ which causes a deformation in eigenspace which matches the optimal deformation $\boldsymbol{x}$ as closely as possible.

$$(\boldsymbol{x} + d\boldsymbol{x}) = \bar{\boldsymbol{x}} + \boldsymbol{P_x}'(\mathbf{b} + d\boldsymbol{b}) \,. \tag{30}$$

Subtracting equation (26) from equation (30) we get

$$d\boldsymbol{x} = \boldsymbol{P_x}' d\boldsymbol{b} \,. \tag{31}$$

This is an over-determined set of linear equations where the number of equations (3n) is much larger than the number of variables (the number of modes is usually restricted from around 5 to 10). Therefore a least squares approximation to the solution can be obtained using standard methods of linear algebra. Because of the orthogonality of $\boldsymbol{P_x}$ in 2-D, the least squares solution could be obtained by $d\boldsymbol{b} = \boldsymbol{P_x}^T d\boldsymbol{x}$. In 3-D, the non-orthogonality of $\boldsymbol{P_x}'$ does not allow of solving equation (31) this simple way. With this object, the general purpose least squares routine F04JAF from the NAG^{TM} Fortran library has been applied to obtain $d\boldsymbol{b}$.

The entire procedure is repeated iteratively and starts with the average model such that $\boldsymbol{b}_{t=0} = \mathbf{0}$. At each iteration step, we compute a new set of displacements from the match of profiles and update the shape deviation vector $\boldsymbol{b}$ until the variation of the shape remains below a threshold value for a certain number of iterations.

E. Shape constraints

There are two different kind of constraints we apply to keep the resulting shape consistent with the shape model. On the one hand, there is a limited number of eigenmodes due to the small number of individuals and the restriction of the number of modes. And on the other hand, after the weights have been updated by

$$\boldsymbol{b}_{t+1} = \boldsymbol{b}_t + d\boldsymbol{b}_t, \tag{32}$$

we constrain the components b_i of $\boldsymbol{b}$ using the standard deviation defined by the statistical model, which is given by the eigenvalues $\sqrt{\lambda_i}$ (see Fig. 7). Thus, each component of $\mathbf{b}_{i,t+1}$ lying outside of the interval $\pm a\sqrt{\lambda_i}$ will be truncated, where the constant a is set to 2.

VIII. Results and Validation of segmentation

Figure 12.a shows the initial placement of the left hippocampus model (white line) together with the hand-segmented contour (gray line) on a sagittal 2D slice and as a 3D scene viewed from the right side of the head. After about 100 iterations the model gives a sufficiently close fit to the data. The model used in this example had 5 degrees of freedom, and model profiles had a total length of 11 sample points while the extracted profiles a length of 19 sample points. The whole segmentation process takes about 2 minutes on a Sun Ultra 1 workstation and runs fully automatically after initializing the model with a new data set.

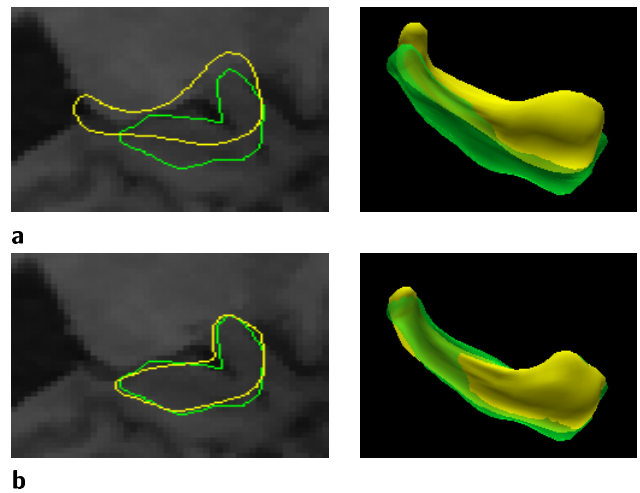


Fig. 12. Segmentation result of a left hippocampus on sagittal 2D slices and 3D views from the left hand side. Image **a** has been taken after initialization, while image **b** illustrates the final result after about 100 iterations.

The above procedure has been applied to all 21 data sets where the hippocampus had been manually segmented. To make optimal use of the relatively small data set, 21 models have been built leaving out one shape each time and applying the technique to this specific shape. The performance of the automatic segmentation has been tested by comparisons with manually segmented object shapes, which were used as a widely accepted standard given the lack of ground truth. $\boldsymbol{A}$ represents the model shape obtained by human experts, $\boldsymbol{B}$ the result of the new model-based segmentation.

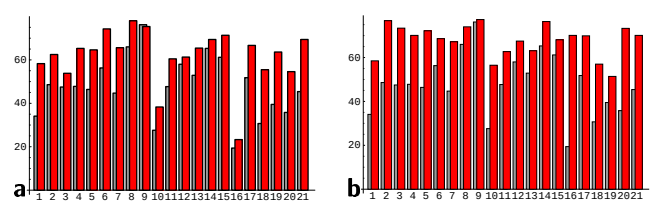


Fig. 13. Overlap measure $(\boldsymbol{A} \cap \boldsymbol{B})/(\boldsymbol{A} \cup \boldsymbol{B})$ in percentage calculated between manually (A) and automatic (B) segmented left hippocampi of 21 individuals. Bars in light grey illustrate the measure at initialization and in dark grey after deformation. In image **a** segmentations have been carried out with the leave-one-out method, while in **b** all shapes have been included to build the model.

The overlap measure $(\boldsymbol{A} \cap \boldsymbol{B})/(\boldsymbol{A} \cup \boldsymbol{B})$ shown in Figure 13.a is calculated on binary voxel maps, created by intersection of the object surfaces with the voxel grid. We therefore avoid the discretization errors by projecting the surfaces back to a voxel grid. The resulting measure is very sensitive to even small differences in overlap, both inside and outside of the object model, and therefore a strong test for segmentation accuracy. For example, two voxel cubes of a volume of $10 \times 10 \times 10$ shifted by one voxel along the space diagonal direction results in only a 57% overlap (729/1271), although the mean distance of surfaces is roughly one voxel.

The calculation of the mean distance of surfaces can be determined in an elegant way directly from the coefficients of the spherical harmonic expansion using Parseval's theorem relating the energy of the one-dimensional continuous signal $f(t)$ to its Fourier coefficients (a_n, b_n):

$$\frac{1}{T}\int_{-T}^{T}[f(t)]^2 dt = \frac{a_0^2}{2} + \sum_{1}^{\infty}(a_n^2 + b_n^2) \qquad (33)$$

The equation also applies to other orthogonal basis functions, such as spherical harmonics, and to higher dimensions as well. This way the average distance of a closed surface $\boldsymbol{x}(\boldsymbol{u})$ from the coordinate origin can be described as

$$\oint \parallel \boldsymbol{x}(\boldsymbol{u}) \parallel^2 d\boldsymbol{u} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \mid \boldsymbol{c}_l^m \mid^2 = 4\pi \cdot \mathrm{MSD} \ , \qquad (34)$$

where MSD stands for *mean squared distance* measured from the origin of the coordinate system. Similarly, the average distance between two surfaces given by $\boldsymbol{x}_1(\boldsymbol{u})$ and $\boldsymbol{x}_2(\boldsymbol{u})$ or $\boldsymbol{c}_{l1}^m$ and $\boldsymbol{c}_{l2}^m$ can be written as

$$\oint \parallel \boldsymbol{x}_1(\boldsymbol{u}) - \boldsymbol{x}_2(\boldsymbol{u}) \parallel^2 d\boldsymbol{u} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \mid \boldsymbol{c}_{l1}^m - \boldsymbol{c}_{l2}^m \mid^2 \ . \qquad (35)$$

providing an elegant way to calculate an error measure based on average surface distance from the spherical harmonic coefficients of the model and the segmentation result.

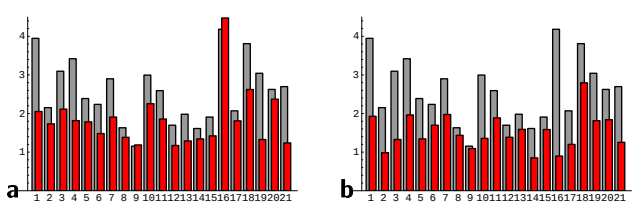


Fig. 14. Average distances in mm calculated between manually and automatic segmented left hippocampi of 21 individuals. The bars in light grey illustrate the mean distance of the initialization of the model in a new data set and the dark bars the final mean distance of surfaces to the model surface. The length of the hippocampus is about 40 mm. In image **a** segmentations have been carried out with the leave-one-out method, while in **b** all shapes have been included to build the model.

Figure 14.a nicely illustrates how the mean distance of surfaces is reduced by the iterative elastic deformation of the model. Again, we take the human expert's segmentation as ground truth and compare its surface with the result of the automatic segmentation. The bars in light grey illustrate the mean distance of the initialization of the model in a new data set, and the dark bars the final mean distance of surfaces to the model surface. The horizontal axis lists the series of 21 normal controls and schizophrenics that were used in this study.

To illustrate the performance drop caused by using statistically suboptimal models, obtained from a training set being too small represent the entire class of shapes, we also show segmentation results in Figure 13.b and Figure 14.b which has been computed whit a model including all 21 shapes. Experiments with the statistically optimal 2-D model of the *corpus callosum* showed, that including a shape into a saturated model does not significantly influence the quality of the segmentation. However, the restricted size of our training set did not allow to generate such saturated shape models even after including all available samples, leading to slight degradation of the segmentation results. In other worlds, our statistical 3-D models does not contain enough information to represent all possible shapes of a certain organ calling for the compilation of a larger training set.

IX. Conclusions

We present a new model-based 3-D segmentation technique that provides automatic segmentation of objects from volumetric image data. Tests with a large series of volumetric image data taken from different patient studies demonstrated that the method was robust and provides reproducible results.

The new technique uses elastic deformation of surface models, which carry statistical information of normal geometric shape variation and statistics about gray levels near the object surface. Our models has been derived from a series of interactively segmented training data sets. Thereby, the model represents a real anatomical shape rather than a simple geometric 3-D figure as obtained by CAD modeling, for example. Furthermore, information about the statistics of a normal shape deformation helps to constrain the elastic deformations. This is an important advantage since 3-D snake and balloon techniques are known to be prone to elastically deform to any smooth object shape and to be trapped by disturbing attractors not part of the sought shape.

Our approach has been significantly influenced by the research work of Cootes, Taylor et al. [5], [6]. However, the extension of their originally 2-D method to a true 3-D volumetric segmentation technique required various new solutions to single steps of the procedure.

Parametric shape representation: The most prominent distinction to [6] is that we use a parametric 3-D object shape representation rather than a point distribution model, and that shape statistics is calculated in the space of these shape parameters rather than point coordinates.

Statistical shape models: To overcome the problem of getting a reproducible interactive definition of a set of homol-

ogous points in 3-D space, the approach presented herein proposes an automatic definition of surface meshes with homogeneous distribution of nodes defined in a standard, canonical position.

Object alignment: We define the position and orientation of objects in a global coordinate system which is is defined by the type of application. Small translations and rotations of objects with respect to this coordinate system are *part of the statistical model.* Therefore, we do not separate a similarity transform for alignment and an elastic transform for remaining shape deformations as in [5].

Dual shape representations: Our approach makes use of two shape representations which are used in a vice-versa fashion, taking advantages of shape descriptors holding a compact global object characterization and of a set of surface points giving access to local shape properties.

Similarly to the experience of Cootes et al. [5] we too found that the modeling of gray level information near the object boundaries provides valuable additional information for a model placement and improves the robustness and stability of the iterative optimization scheme. An early version of our segmentation [26] used an energy minimization concept similar to standard snake techniques. This method was very sensitive to the quality of the initialization, and prone to be trapped by local energy minima. The additional use of gray-level profile information represents a strong restriction to the number of possible solutions and was demonstrated to be robust even in the presence of considerable mismatch between initialization and a new object.

Validation has been done by defining shape distance metrics and comparing the results of interactive outlining by experts, which is a common "gold standard" for comparisons, with the shapes obtained by model-based segmentation (see Fig. 14).

We noticed that the convergence is faster if only a small number of modes (usually 5) are involved, while a larger number of modes (usually 10) is required to find the exact contour. Thus, we plan to apply a relaxation method which gradually increases the number of modes. The convergence criteria is set by the size of the deformation of a surface.

The fundamental difficulty of the application of parametric statistical models for 3D organ segmentation remains the efficient establishment of correspondence between the single objects of the training set. This is a major research area at the moment and different approaches are under investigation [27], [10], [1]. The method proposed by Kotcheff [10] is of particular interest, as it addresses not only the problem of correspondence, but at the same time the question of the underlying distribution model for the shape parameters under investigation. This is another basic matter of concern if Principle Component Analysis applied. Using PCA implies a Gaussian noise model on the parametrization, which cannot be expected in general. The idea to use reparametrization for correspondence establishment in order to change the underlying parameter error distribution would offer an fundamentally new way for estimating correspondence in a whole shape population and is currently investigated. As an alternative, Independent Component Analysis looking for higher order correlations in the data can also be applied to cope with the problem of non-Gaussian distribution of the analyzed shape parameters. The application of more sophisticated methods will, however, be limited by the relatively small number of individual samples as compared to the degrees of freedom of the model.

The set of statistical models and the automatic and efficient segmentation technique (only a few minutes per data set) open new possibilities for the processing a large number of data sets as collected in clinical studies, for example in schizophrenia research. This will provide new statistical models with increased number of samples for normal controls and for different patient categories. These statistical models represent the first step in building an anatomical atlas based on a set of surfaces of anatomical shapes. Whereas the current segmentation technique would segment a series of objects independently, a future development could provide a combined modeling of several anatomical structures. The representation of anatomical objects by normalized shape descriptors further exploits its access to morphometric parameters. After segmenting a new set of image data, morphological properties of objects are available for comparative studies.

Acknowledgments

Ron Kikinis and Martha Shenton, Brigham and Women's Hospital, Harvard Medical School, Boston kindly provided the original MR and segmented data sets. We are thankful to Christian Brechbühler from our lab for providing the software for surface parametrization and shape description. We further acknowledge MR datasets provided by the European BIOMORPH consortium (BIOMED 2 95-0845). Finally, the authors would like to thank the anonymous reviewers for their comments and suggestions.

References

[1] F.L. Bookstein. Shape and the Information in Medical Images: A Decade of the Morphometric Synthesis. *Computer Vision and Image Understanding*, 66(2):97–118, May 1997.

[2] C. Brechbühler, G. Gerig, and O. Kübler. Surface parametrization and shape description. In *Visualization in Biomedical Computing*, pages 80–89, 1992.

[3] C. Brechbühler, G. Gerig, and O. Kübler. Parametrization of closed surfaces for 3-D shape description. *CVGIP: Image Understanding*, 61:154–170, 1995.

[4] I. Cohen, L.D. Cohen, and N. Ayache. Using Deformable Surfaces to Segment 3D Images and Infer Differential Structures. *CVGIP: Image Understanding*, 56(2):242–263, 1992.

[5] T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. The Use of Active Shape Models for Locating Structures in Medical Images. *Image and Vision Computing*, 12(6):355–366, 1994. Electronic version: http://s10d.smb.man.ac.uk/publications/index.htm.

[6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active Shape Models - Their Training and Application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[7] T.F. Cootes and C.J. Taylor. Active Shape Models – 'Smart Snakes'. In *British Mach. Vision Conf.*, pages 266–275. Springer-Verlag, 1992.

[8] G. Gerig, J. Martin, R. Kikinis, O. Kübler, M. Shenton, and F. Jolesz. Automatic Segmentation of Dual-Echo MR Head Data. In *IPMI'91*, pages 175–187. Wye, GB, 1991.

[9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. Comp. Vision*, 1(4):321–331, 1988.

[10] A.C.W Kotcheff and C.J Taylor. Automatic Construction of Eigenshape Models by Genetic Algorithm. In *Information Processing in Medical Imaging*, pages 1–14. Springer, 1997.

[11] F. Kruggel and G. Lohmann. Automatical Adaption of the Steretactical Coordinate System in Brain MRI Datasets. In *Information Processing in Medical Imaging*, pages 471–476. Sringer, June 1997.

[12] P. Marais, R. Guillemaud, M. Sakuma, A. Zisserman, and M. Brady. Visual Cerebral Asymmetry. In *Visualization in Biomedical Computing*, pages 411–416, 1996.

[13] T. McInerney and D. Terzopoulos. Deformable Models in Medical Image Analysis: A Survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[14] S. Minoshima, R. A. Koeppe, M. A. Mintun, K. L. Berger, S. F. Taylor, K. A. Frey, and D. E. Kuhl. Automated Detection of the Intercommissural Line for Stereotactic Localization of Functional Brain Images. *The Journal of Nuclear Medicine*, 34(2):322–329, 1993.

[15] Ch. Nastar and N. Ayache. Frequency-based Nonrigid Motion Analysis: Application to Four Dimensional Medical Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1067–1079, 1996.

[16] W. Neuenschwander, P. Fua, G. Székely, and O. Kübler. Initializing Snakes. In *CVPR'94*, pages 658–663, 1994.

[17] A. Pentland and A.P. Sclaroff. Closed-Form Solutions for Physically Based Shape Modelling and Recognition. *IEEE PAMI*, 13(7):715–729, 1991.

[18] F. Poupon, J.-F. Mangin, V. Frouin, and I. Mangin. 3D Multiobject Deformable Templates based on moment invariants. In *Xth Scandinavian Conference on Image Analysis*, pages 149–156, Lappeenranta, Finland, June 1997.

[19] F. Poupon, J.-F. Mangin, D. Hasboun, C. Poupon, V. Frouin, and I. Mangin. Multi-object deformable templates dedicated to the segmentation of brain deep structures. In *1st International Conference on Medical Image Computing and Computer Assisted Intervention*.

[20] S. Sclaroff and A.P. Pentland. On Modal Modelling for Medical Images: Underconstrained Shape Description and Data Compression. In *IEEE Workshop on Biomed. Image Anal.*, pages 70–79, Seattle, Washington, USA, 1994.

[21] M. Shenton. Abnormalities of the temporal lobe and thought disorders in schizofrenia. A quantitative magnetic resonance imaging study. *New England Journal of Medicine*, 327:604–612, 1992.

[22] L.H. Staib and J.S. Duncan. Boundary Finding with Parametrically Deformable Models. *IEEE Transactions on Pattern Analysis and Machine Intelligece*, 14(11):1061–1075, November 1992.

[23] L.H. Staib and J.S. Duncan. Deformable Fourier models for surface finding in 3D images. In *VBC'92*, pages 90–194, 1992.

[24] L.H. Staib and J.S. Duncan. Model-based Deformable Surface Finding for Medical Images. *IEEE Trans. Med. Imaging*, 15(5):1–12, 1996.

[25] G. Székely, M. Bajka, J. Hug, M. Manestar, P. Groscurth, and U. Haller. Anatomical model generation for laparoscopic surgery simulation. In *The Second Visible Human Project Conference Proceedings*, October 1998.

[26] G. Székely, A. Kelemen, Ch. Brechbühler, and G. Gerig. Segmentation of 2-D and 3-D objects from MRI volume data using constrained elastic deformations of flexible Fourier contour and surface models. *Medical Image Analysis*, 1(1):19 – 34, 1996.

[27] H.D. Tagare. Non-rigid Curve Correspondence for Estimating Heart Motion . In *Information Processing in Medical Imaging*, pages 489–494. Springer, 1997.

[28] J. Talairach and P. Tournoux. *Co-planar stereotaxic atlas of the human brain*. Thieme, Stuttgart, 1988.

[29] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models and 3D Object Reconstruction. *Int. J. Comp. Vision*, 1(3):211–221, 1988.

[30] J. Ph. Thirion, S. Prima, and G. Subsol. Statistical Analysis of Dissymmetry in Volumetric Medical Images. Technical Report 3178, INRIA, 1997. Electronic version: http://www.inria.fr/RRRT/RR-3178.html.

[31] B.C. Vemuri and A. Radisavljevic. Multiresolution Stochastic Hybrid Shape Models with Fractal Priors. *ACM Trans. Graphics*, 13(2):177–200, 1994.

Medical Image Analysis (1996) volume 1, number 1, pp 19–34
© Oxford University Press

Segmentation of 2-D and 3-D objects from MRI volume data using constrained elastic deformations of flexible Fourier contour and surface models

Gábor Székely*, András Kelemen, Christian Brechbühler and Guido Gerig

Communication Technology Laboratory, ETH-Zentrum, CH-8092 Zurich, Switzerland

Abstract
This paper describes a new model-based segmentation technique combining desirable properties of physical models (snakes), shape representation by Fourier parametrization, and modelling of natural shape variability. *Flexible parametric shape models* are represented by a parameter vector describing the mean contour and by a set of *eigenmodes of the parameters* characterizing the shape variation. Usually the segmentation process is divided into an initial placement of the mean model and an elastic deformation restricted to the model variability. This, however, leads to a separation of biological variation due to a global similarity transform from small-scale shape changes originating from elastic deformations of the *normalized model contours only*. The performance can be considerably improved by building shape models normalized with respect to a small set of stable landmarks (AC-PC in our application) and by explaining the remaining variability among a series of images with the model flexibility. This way the image interpretation is solved by a *new coarse-to-fine segmentation procedure* based on the set of deformation eigenmodes, making a separate initialization step unnecessary. Although straightforward, the extension to 3-D is severely impeded by difficulties arising during the generation of a proper surface parametrization for arbitrary objects with spherical topology. We apply a newly developed surface parametrization which achieves a uniform mapping between object surface and parameter space. The 3-D procedure is demonstrated by segmenting deep structures of the human brain from MR volume data.

Keywords: 3-D deformable models, 3-D shape analysis, segmentation of multidimensional images, statistical analysis of anatomical objects

Received October 3, 1995; revised January 12, 1996; accepted January 19, 1996

1. INTRODUCTION

Segmentation of anatomical objects from large 3-D medical data sets, which result for example from routine magnetic resonance imaging (MRI) examinations, represents one of the basic problems of medical image analysis. In some limited applications, segmentation could be achieved by minimal user interaction, providing procedures for the interpretation of medical scenes, which can be applied routinely (Gerig *et al.*, 1991). For general applications, however, adequate segmentation cannot be obtained without expert knowledge, requiring tedious manual interaction by a human specialist.

Elastically deformable models (snakes) (Kass *et al.*, 1988) have been proposed as tools for supporting manual object delineation. While such procedures can be extended to 3-D (Terzopoulos *et al.*, 1988; Cohen *et al.*, 1992), their initialization becomes difficult. Most often, the initial guess must be very close to the sought contour to guarantee a successful result (Neuenschwander *et al.*, 1994). While a careful and time-consuming analysis is acceptable for outlining complex pathological objects, no real justification for such a procedure can be found for the delineation of normal, healthy organs, as needed in radiation treatment planning, for example.

*Corresponding author
(e-mail: szekely@vision.ee.ethz.ch)

The primary reason for the need of a precise snake initialization is the presence of disturbing attractors in the image, which do not belong to the actual object contour but force the snake into local energy minima. If the deformation of a snake could be limited to shapes within the normal anatomic variation of organs, such local minima could be avoided.

Elastically deformable parametric models offer a straightforward way for the inclusion of prior knowledge in the image interpretation process by incorporating prior distributions on the elastic model parameters to be estimated. Such a procedure has been implemented, for example, by Vemuri and Radisavljevic (1994) using a hybrid primitive called deformable superquadrics constructed in a multi-resolution wavelet base or by Staib and Duncan (1992a) for deformable Fourier models.

For complex shapes described by a large number of possbly highly correlated parameters, the usage of such priors may become tedious. The modal analysis as proposed by Pentland and Sclaroff (1991) offers a promising alternative by changing the basis from the original modelling functions to the eigenmodes of the deformation matrix. This way the dominant part of the deformations can be characterized by the few largest eigenmodes, reducing the dimensionality of the object descriptor space substantially. Such modal analyses have been successfully applied to medical image analysis, for example, by Sclaroff and Pentland (1994) or Nastar and Ayache (1994).

Cootes *et al*. (1993) combined the power of parametric deformable shape descriptors with statistical modal analysis. They use active shape models, which strictly restrict their possible deformations according to the statistics of training samples. Object shapes are described by the point distribution model (PDM) (Cootes and Taylor, 1992), which represents the object outline by a subset of boundary points. There must be a one-to-one correspondence between these points in the different outlines of the training set. After normalization to size, orientation and position they provide the basis for the statistical analysis of the object shape deformations. The mean point positions and their modes of variation (i.e. the eigenvectors corresponding to the largest eigenvalues of their covariance matrix) are used for delimiting the object deformations to a reasonable linear subspace of the complete parameter space. They propose to solve the 3-D shape analysis problem by a slice-by-slice approach (Hill *et al.*, 1992) or by extending the PDM concept for the description of 3-D shapes (Hill *et al.*, 1993). Similar parametrization based on point-by-point correspondence was proposed for the 3-D shape analysis of brain structures by Martin *et al*. (1994). By using the free vibration modes of the nodal displacement matrix of an elastic body, they even separate physical deformation modes which can be modelled by elastic deformation models from experimental modes, evaluated from statistical analysis of a training sample.

While the idea of restricted elastic deformation of an average surface model is very promising, the parametrization of shapes by displacement of corresponding points on their surfaces is not a convenient technique. For a large training set containing many different anatomic structures, the generation of this parametrization seems to be very tedious and, because of the lack of a reasonable automatization, can be a source of errors. A similar modal analysis, however, can be performed for other contour parametrization techniques, as for example for the 2-D Fourier descriptors which were originally proposed by Persoon and Fu (1977) and Kuhl and Giardina (1982). Staib and Duncan have demonstrated segmentation by parametrically deformable elastic models for 2-D outlines (Staib and Duncan, 1992a) and 3-D object surfaces (Staib and Duncan, 1992b); their 3-D model was, however, limited to star-shaped surfaces. Here, we propose a novel technique based on modal analysis of the parameter vector of object contours, providing the desirable restriction of elastic deformations. The method uses automatic shape parametrization for any surfaces with spherical topology, thus avoiding the problem of finding corresponding points among different boundaries.

The paper is organized as follows. Section 2 discusses the new 2-D modal analysis procedure and the model-based segmentation by restricted deformation. The procedure is illustrated with the segmentation of the corpus callosum structure on mid-sagittal MR images of the human brain. Section 3 proposes several improvements of the procedure. Section 4 generalizes the method to 3-D, addressing the increased complexity requiring new mathematical and numerical solutions. Section 5 finally gives the conclusions and future directions for the development of automated, model-based segmentation techniques.

2. MODAL ANALYSIS OF 2-D FOURIER MODELS

In this section we first summarize the mathematics of Fourier descriptors for the parametrization of simple closed curves, and their use in snake-like elastic matching procedures. We summarize how to apply the modal analysis to 30 normalized outlines of the corpus callosum extracted from 2-D mid-sagittal MR images. The proposed procedure is a combination of an initialization using an average model based on template matching technique followed by an elastic deformation restricted to the major eigenmodes. We demonstrate that this separation of the similarity transform (translation, rotation and scaling) from the elastic deformation often leads to unsatisfactory segmentation results, calling for a modified segmentation process that provides a unified framework for initialization and elastic model matching.

2.1. Parametrization of 2-D contours with Fourier descriptors

The contour of a simply connected object (without holes) is represented by a closed curve with coordinates $(x(t), y(t))$, with t ranging from 0 to 2π. The coordinate functions can be expanded in a Fourier series. Restricting the series to degree K results in the parametric description

$$\begin{aligned} \boldsymbol{r}(t,\boldsymbol{p}) &= \begin{pmatrix} x(t,\boldsymbol{p}) \\ y(t,\boldsymbol{p}) \end{pmatrix} \\ &= \begin{pmatrix} a_0 \\ c_0 \end{pmatrix} + \sum_{k=1}^{K} \begin{pmatrix} a_k & b_k \\ c_k & d_k \end{pmatrix} \cdot \begin{pmatrix} \cos(kt) \\ \sin(kt) \end{pmatrix}. \end{aligned}$$

The outline is now completely described by the parameter-vector

$$\boldsymbol{p} = (a_0 \dots a_K, b_1 \dots b_K, c_0 \dots c_K, d_1 \dots d_K)^\top .$$

The parameters can be easily calculated from the sampling points of the outline $\boldsymbol{q}_0, \boldsymbol{q}_1, \dots, \boldsymbol{q}_P$ with $\boldsymbol{q}_0 = \boldsymbol{q}_P$ (we use maximally dense sampling of the boundary, as provided by the image raster). The resulting parametric shape description can be made invariant under similarity transformation by shifting, rotating and scaling according to the actual displacement, orientation and size of the ellipse determined by the first degree terms of the Fourier series. Similarly, the starting point is moved to a canonical position.

2.2. Fourier snakes

The snake technique as proposed by Kass *et al.* (1988) tries to find the position of a curve $\boldsymbol{r}(t,\boldsymbol{p})$, which minimizes the energy

$$E(\boldsymbol{p}) = E(\boldsymbol{r}(t,\boldsymbol{p})) = E_{\mathrm{I}}(\boldsymbol{r}(t,\boldsymbol{p})) + E_{\mathrm{D}}(\boldsymbol{r}(t,\boldsymbol{p})).$$

By varying $\boldsymbol{p}$, the curve deforms itself to minimize the image energy

$$E_{\mathrm{I}}(\boldsymbol{r}) = \int_0^{2\pi} P(\boldsymbol{r}(t,\boldsymbol{p}))\,\mathrm{d}t,$$

searching for an optimal position in the image, described by the potential $P(\boldsymbol{r}(t,\boldsymbol{p}))$. A typical choice takes $P(\boldsymbol{r}(t,\boldsymbol{p}))$ equal to the negative magnitude of the image gradient:

$$P(\boldsymbol{r}(t,\boldsymbol{p})) = -\left|\nabla I(\boldsymbol{r}(t,\boldsymbol{p}))\right|,$$

where I is the smoothed image. The deformation term $E_{\mathrm{D}}(\boldsymbol{r})$ is called the internal energy of the snake and serves as a regularization force. It restricts the elongation and bending of the snakes, and normally depends on the first and second derivatives of the curve $\boldsymbol{r}(t,\boldsymbol{p})$.

Staib and Duncan (1992a) propose a different energy model, which makes use of the normal direction of the parametrized curve and of the image gradient to achieve a higher selectivity. Normalizing the image potential by the contour length allows contraction and dilation of curves without affecting the energy function.

$$\begin{aligned} E_{\mathrm{I}}(\boldsymbol{r}(t,\boldsymbol{p})) = \; & \pm \left(\int_0^{2\pi} \nabla I(\boldsymbol{r}(t,\boldsymbol{p})) \cdot \dot{\boldsymbol{r}}_\perp(t,\boldsymbol{p})\,\mathrm{d}t \right) \\ & \cdot \left(\int_0^{2\pi} \|\dot{\boldsymbol{r}}(t,\boldsymbol{p})\|\,\mathrm{d}t \right)^{-1}. \end{aligned}$$

The sign of the energy $E_{\mathrm{I}}(\boldsymbol{r})$ will decide between segmenting bright objects on a dark background or vice versa. The polarity of the boundary can be neglected by using the absolute value of the dot product in the integration term.

One has to realize that cutting the Fourier expansion at a finite degree already serves as a regularization, leaving out high frequency variations of the coordinate functions. However, the internal energy cannot be neglected, basically for two reasons:

- even Fourier models of low degree can generate sharp cusps, and
- the parametrized contour can cross itself.

Whereas self-crossings of the outline are expensive to detect, discontinuities of the tangent can be evaluated from the curve parametrization. At such locations the curvature $\kappa(t_0,\boldsymbol{p})$ and its derivative $\dot{\kappa}(t_0,\boldsymbol{p})$ both become infinite. While high curvature of the boundary should not be excluded *a priori*, the curvature derivative is chosen to indicate discontinuities (see also Staib and Duncan, 1992), providing the following expression for the internal energy of the Fourier snake:

$$E_{\mathrm{D}}(\boldsymbol{r}(t,\boldsymbol{p})) = \int_0^{2\pi} \dot{\kappa}^2(t,\boldsymbol{p}) \cdot \|\dot{\boldsymbol{r}}(t,\boldsymbol{p})\|^2\,\mathrm{d}t\,.$$

For the minimization of the total energy function we used the E04JBF routine of the *NAG*™ library (1988), using a quasi-Newton algorithm for finding an unconstrained minimum of a function of several variables.

2.3. Model-based segmentation of the corpus callosum

We tested the proposed procedure on a collection of 30 mid-sagittal slices of MRI brain images. A few images from the training set are shown in the upper row of Figure 1. Our goal was the automatic segmentation of the outline of the corpus callosum from grey scale images.

2.3.1. Model building

The corpus callosum outline had to be segmented and parametrized for each image of the training set. We used

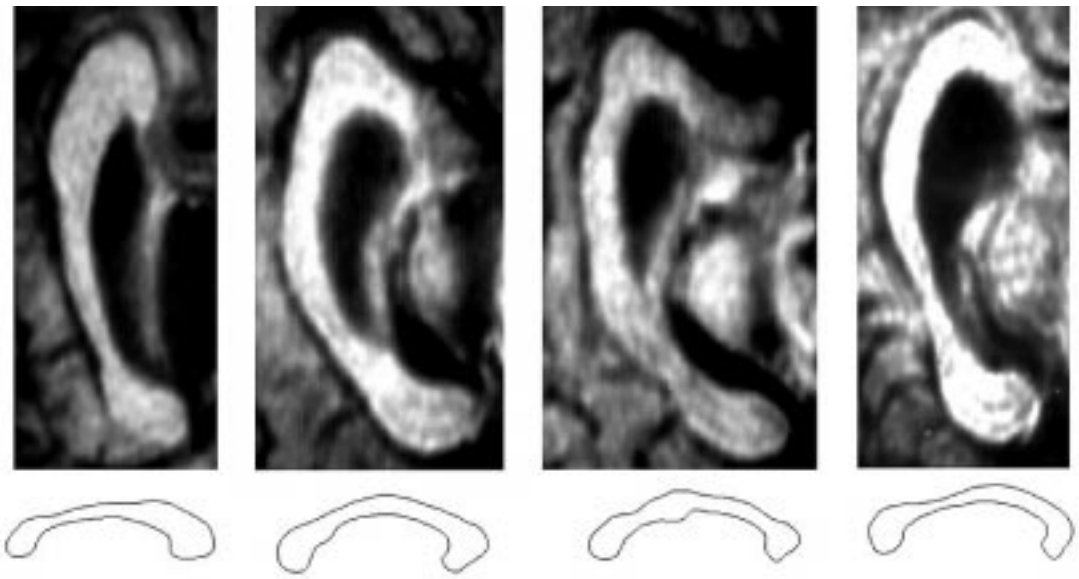


Figure 1. Training set for the analysis of the corpus callosum on mid-sagittal slices of MRI brain data sets. The upper row shows a region of interest of the original slice, the lower row contains the segmented outline in a standardized invariant configuration (Fourier coefficients up to degree 100).

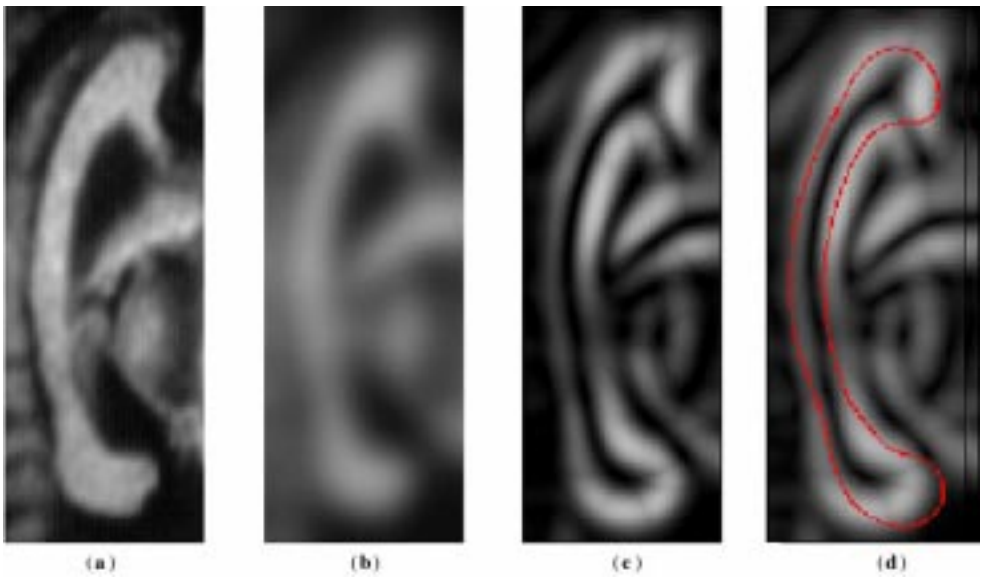


Figure 2. Template-matching example to find a similarity transform between the mean model curve and the gradient magnitude image. (**a**) Shows the original image, (**b**) its Gaussian smoothed version ($\sigma = 5$ pixels) and (**c**) the corresponding Canny gradient magnitude image, respectively. In (**d**) the optimal fit found by template matching is overlaid with the edge image.

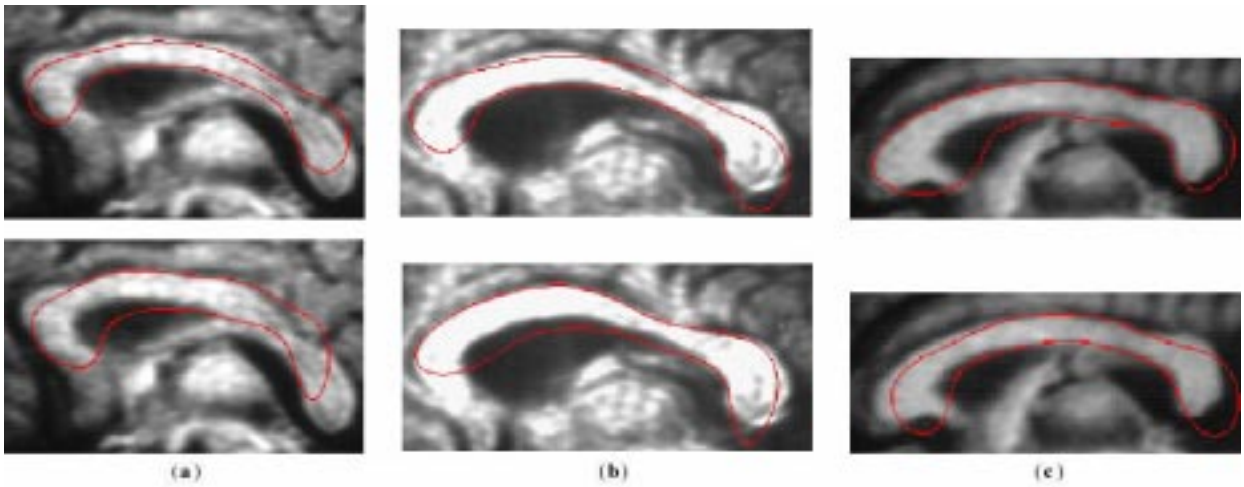


Figure 3. Segmentation examples (**a**–**c**) illustrating segmentation failures. Images in the top row illustrate the initial placement of the model curves (using template matching), the bottom row shows the segmentation result. The *same* initial model curve is used for the different images.

the Fourier snake program as described above to perform this task, with a manual initialization of the optimization. The segmentation results can be seen in the lower row of Figure 1. Fourier coefficients up to degree 100 have been used for the description. The resulting contours of the training set have been normalized to be invariant to translation, rotation, scaling and the starting point of the parametrization.

The mean model has been determined from the normalized outlines by simply averaging their parameters. In order to determine the major deformation modes defined by the above training set, we performed a principal component analysis of the covariance matrix of the normalized Fourier coefficients. As after the first few eigenvectors the variance becomes very small, the few largest eigenmodes have been taken to build a flexible model that explains the biological variability of the shape of the corpus callosum outline.

2.3.2. *Initial placement of model contour by template matching*

A segmentation of the corpus callosum from grey-valued images based on the deformation of the mean model requires a suitable initialization. Due to the normalization of the Fourier coefficients, the average model only expresses shape deformations up to a similarity transformation. Therefore, the initial placement has to provide a sufficiently good match between the model and the edges in the grey-valued image. A standard template-matching procedure was chosen to solve this first optimization problem by equidistantly sampling the possible parameters of the similarity transformation within a reasonable range. The goodness of fit was calculated for a Canny edge map on a relatively large scale, since the rigid transformation does not allow for elastic shape deformations. Figure 2 illustrates the result of the initialization by template matching. Figures 2a–c show the original image, the Gaussian smoothed image ($\sigma = 5$ pixels), and the Canny gradient magnitude, respectively. Figure 2d presents an overlay of the optimal fit between model curve and gradient image.

2.3.3. *Segmentation by restricted elastic deformation*

After initialization, we apply a modified version of the Fourier snakes program. Instead of optimizing in the complete space of the normalized Fourier coefficients, the optimization has been restricted in order to allow only deformations which are prominently represented in the training sample. In the first step the eigenvectors of the parameter covariance matrix are selected as a different set of basis functions in place of the harmonics. The restricted variation is achieved by choosing a subset of eigenmodes, usually the n largest ones, and calculating the optimization in this linear subspace. Starting with the result of the template matching, the model is elastically deformed until it fits (in a local optimum) the edges along the object contour.

The procedure described above provided satisfactory segmentation results in many cases. In some cases, however, we found it difficult to find the correct contour. Figure 3 shows a few examples of unsuccessful segmentations. Analysis of the results has shown that the separation of the similarity transform (in the initialization step) from the elastic deformation is mainly responsible for this failure. As the models have been normalized before the statistical analysis, translation,

rotation and scaling have actually been excluded from the elastic deformation step, forbidding minor corrections of the similarity transform determined by the template matching in the initialization step. Restricting the deformations to the subspace of the dominant eigenmodes made this situation even worse by not allowing corrections through free elastic deformations. This has led to the large segmentation errors demonstrated by the above examples.

The following section describes how these problems can be solved by incorporating the similarity transform into the analysis of biological variability. This way all covariation of the pose and elastic deformation parameters can be handled in a consistent way, too. The selected solution also avoids the somewhat artificial separation of the image analysis process into an initialization and elastic matching, leading to a two-step coarse-to-fine segmentation procedure.

3. IMPROVING FOURIER SNAKE SEGMENTATION

3.1. Alternative initialization techniques

One could try different ways to avoid the segmentation problems analyzed in the previous section.

- The desirable mixing of the effects of similarity transform and elastic deformation could be achieved by the incorporation of eigenvalues belonging to the the largest eigenvectors into the initialization (template matching) procedure. However, the dimensionality of the sampled search space would become far too high and would create, especially in view of a generalization to 3-D, an unacceptable computational burden.
- One could also try to extend the parameters of the elastic deformation with translation, rotation and scaling. This would lead to a higher dimensional local optimization problem without allowing the handling of the optimization of the parameters in one coherent framework.
- The selected optimization procedure finds the nearest local optimum on a high-dimensional strongly non-convex goal function. This makes the procedure highly sensitive to non-desired local optima. We are currently experimenting with alternative global optimization techniques as for example genetic algorithms as proposed by Hill and Taylor (1992) and the taboo search (Reeves, 1993) borrowed from discrete optimization, which may provide some improvement. However, one should realize that *the major problem does not lie in the initialization procedure*. The template matching usually provides a reasonable first approximation using the mean model, which still leads to unsatisfactory results due to the separation of the similarity transform and the elastic deformation described by the principal modes of the Fourier coefficients. This problem is addressed in the next paragraph.

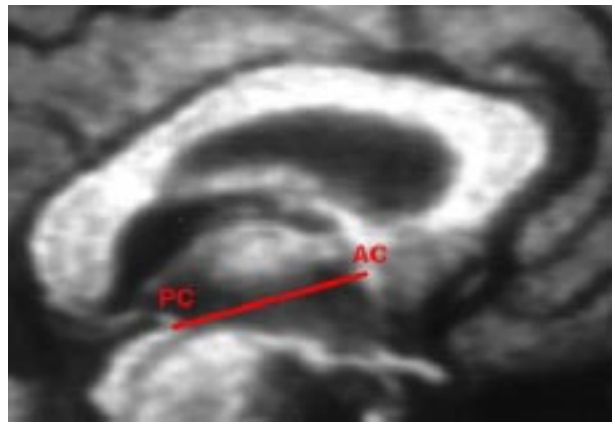


Figure 4. Position of the anterior commissure (AC) and the posterior commissure (PC) on a mid-sagittal MRI image slice. The connecting line represents the unit vector $\boldsymbol{x}$ of the external anatomical coordinate system.

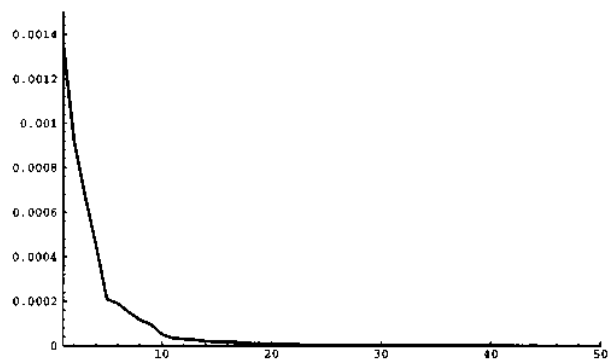


Figure 5. The 50 largest eigenvalues of the covariance matrix calculated from the Fourier parameters of the 30 curves in the training set.

3.2. Model incorporating full biological variability

We expect for images representing anatomy that the *relative position, rotation* and *size* of healthy organs is restricted in a similar way as their *elastic deformation*. If we could define a coordinate system fixed to the anatomy, there would be no reason for an unrestricted similarity transform which precedes the elastic matching. The Fourier descriptors of the organ outlines *originally contain this information*, but we suppress it by normalizing the coefficients. In the case of the corpus callosum, the AC/PC line is a generally accepted, well-detectable geometric feature of the mid-sagittal images, which represents such a standard coordinate system. The line from the anterior to the posterior commissure (AC/PC line) as illustrated in Figure 4 has been manually extracted for each image of the training set. After determination of the Fourier

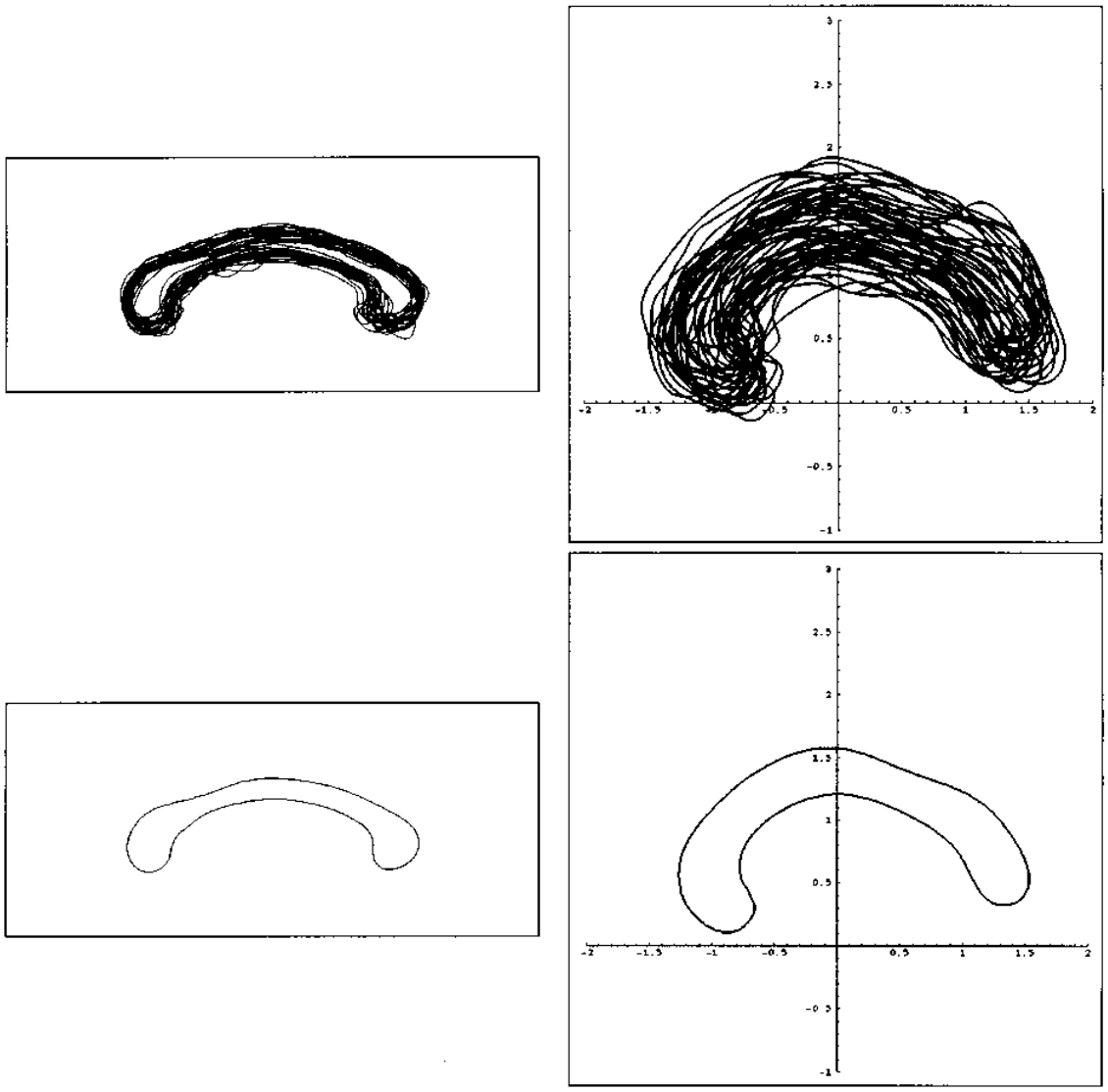


Figure 6. The upper row shows the segmented outlines of the training set with (left) and without (right) normalization of the pose parameters. The lower row shows the corresponding mean models.

coefficients, we apply normalization only to fix the starting point of the curve parametrization. The standardization of the images, necessary for the determination of the deformation modes, is based on a normalization of the AC/PC line to an $\boldsymbol{e}_x$ unit vector. After that, the same statistical analysis of the test set can be performed as previously explained, providing a mean model (now including its relative position and size to the AC/PC line), and the deformation modes which incorporate the parameters of the similarity transform, too. The resulting eigenvalues are plotted in decreasing order in Figure 5. One can see that the remaining variation after the 12th eigenvalue becomes negligible. Accordingly, all deformations have been restricted to the 12 largest eigenvectors.

Figure 6 illustrates the determination of the mean model by comparing the results using the normalized outlines as described in the previous section (left side) and the average model resulting without normalization presented in the anatomical reference frame (right side). One can see the variation of the

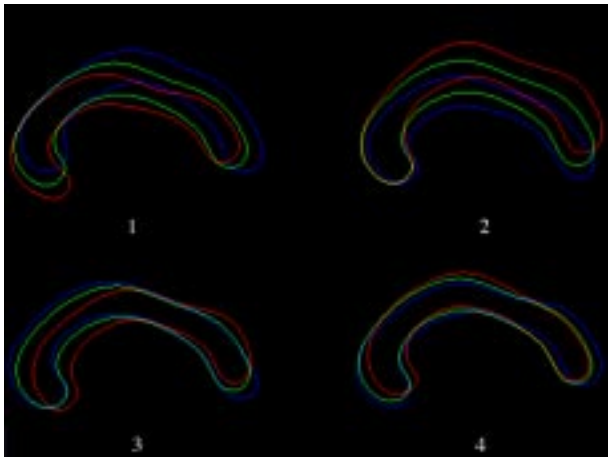


Figure 7. The first four eigenmodes of the deformations of the 30 objects in the training set. The calculations are based on contours represented by Fourier descriptors, which are normalized only with respect to the choice of the starting point. The deformation range amounts to $\pm\sqrt{2}$ eigenvalues.

pose of the training samples in the external coordinate system. At the same time, much more characteristic shape features are retained in the average outline if no normalization is performed before the training.

Figure 7 illustrates the deformations according to the first four eigenmodes, with deformations in the range of $\pm\sqrt{2}$ eigenvalues. It shows how the pose and elastic deformation parameters are mixed together in the dominant eigendeformations.

3.3. Segmentation

The determination of the AC/PC line now becomes part of the segmentation, since the model is built based on a normalization to these landmarks. The AC/PC line is determined manually for the image under analysis. The flexible model, characterized by the mean contour and the eigenmodes, now incorporates *changes of the position and local deformations* of the generic model, which makes the initialization step obsolete. In order to make the optimization robust against local extrema we applied a *two-step coarse to fine* strategy.

In the first step on the coarse level we use a procedure similar to template matching, as previously described. However, we now use a small set of a few dominant modes and calculate the best match in this linear subspace of major deformations by equidistantly sampling the eigenvalues of the selected deformation modes in a reasonable range. This way the complete parameter space is explored, and the result of this coarse segmentation can be used as a reliable initialization for the following local optimization step. One should realize that the larger the number of eigenmodes participating in this coarse step, the better the initial fit will be in the following step. On the other hand, the computational burden grows exponentially with the dimensionality of the sampled parameter space. The number of eigenmodes considered in this step can be selected by trading the fit quality and the computational requirements.

On the fine level the segmentation is performed in the homogeneous parameter space of the eigenvectors corresponding to the largest eigenvalues, now choosing a larger number of eigenvectors than on the coarse level. Due to the full incorporation of the parameters of the similarity transform into the eigendeformations, the fine tuning of the model pose can be performed within the variability determined by the training set. Figure 8 shows the results of the modified segmentation procedure, which achieved perfect results in almost all cases.

One should mention that in some applications, the identification of the small individual variations of the contour outline, which is not represented even by the whole set of the eigenmodes, is desirable. In this case our strategy can be extended with an additional finer level by applying an unrestricted deformation of the parametrized snake using the result of the previous two levels as initialization.

4. 3-D FOURIER MODELS OF HUMAN BRAIN STRUCTURES

Our goal is the generalization of the improved 2-D procedure to 3-D. The new tool should allow a *well reproducible* segmentation of objects in volume data with *minimal human interaction*. This section describes the first steps in this direction, the generation of 3-D Fourier models from manually segmented training data, and the use of unrestricted 3-D Fourier snakes for elastic matching in grey-valued volume images. As in the previous section, we first summarize the basic mathematics of the 3-D Fourier snakes, and then show first results on some MRI brain data. The major problem, finding a *homogeneous parametrization* of surfaces of *arbitrarily shaped objects*, is solved by applying a very recently developed new parametrization technique. It overcomes limitations given by other surface parametrization schemes, e.g. the torus topology presented in Staib and Duncan (1992b). The following description is guided by the example of segmenting deep grey matter structures of the human brain from MR volume data.

4.1. Description of surfaces by spherical harmonic functions

The description of the surfaces of simply connected 3-D objects in an arbitrary basis can be performed similarly to the

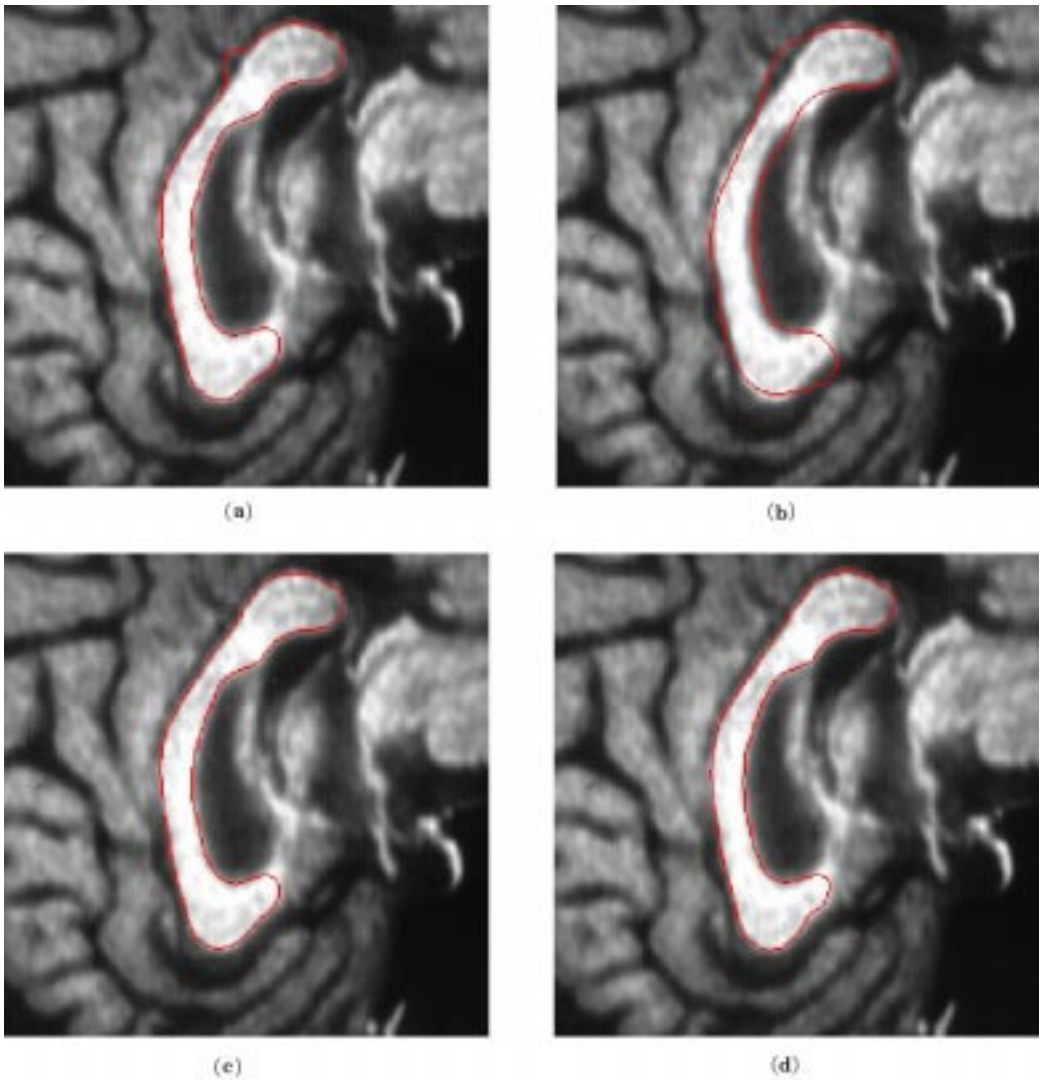


Figure 8. Segmentation of the corpus callosum using the manually determined AC/PC line as a reference coordinate system for model building and segmentation. (**a**) Shows the initialization, resulting from a coarse match in the subspace of the largest four deformation modes. (**b**–**d**) Illustrate single steps of the optimization in the subspace of the largest 12 deformation modes (allowing fine adjustments) and the final result.

2-D case. The surface will be parametrized by two variables, the θ and ϕ polar parameters, and will be defined by three explicit functions

$$\boldsymbol{r}(\theta, \phi) = \begin{pmatrix} x(\theta, \phi) \\ y(\theta, \phi) \\ z(\theta, \phi) \end{pmatrix}.$$

We emphasize that this is not a radial function. If we select the spherical harmonic functions (Y_l^m denotes the function of degree l and order m, see Greiner and Diehl (1964)) as a basis, the coordinate functions can be written as

$$\boldsymbol{r}(\theta, \phi, \boldsymbol{p}) = \sum_{k=0}^{K} \sum_{m=-k}^{k} \boldsymbol{c}_k^m Y_k^m(\theta, \phi),$$

where

$$\boldsymbol{c}_k^m = \begin{pmatrix} c_{x_k}^m \\ c_{y_k}^m \\ c_{z_k}^m \end{pmatrix}.$$

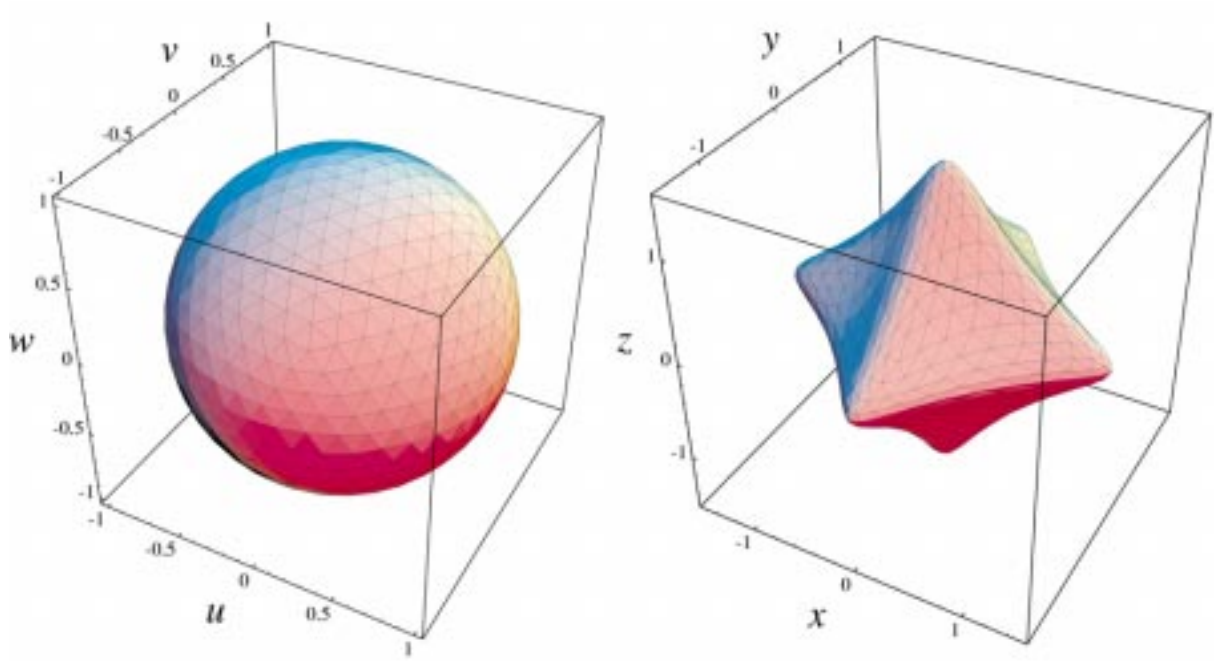


Figure 9. A simple function $U \hookrightarrow \mathbb{R}^3$, visualized by facetting the surfaces in triangles.

We again restricted the expansion to the first $K+1$ terms.

When the free variables θ and ϕ run over the whole sphere (e.g. $\theta = 0 \ldots \pi$, $\phi = 0 \ldots 2\pi$—see Figure 9, left), the point $\boldsymbol{r}(\theta, \phi)$ runs over the whole surface of our object (Figure 9, right). The sphere Ω_3 is considered a perfectly symmetric surface without any singular points or preferred directions.

The surface is then described by the parameters

$$\begin{aligned} \boldsymbol{p} \;=\; & (c_{x}{}^{0}_{0}, c_{y}{}^{0}_{0}, c_{z}{}^{0}_{0}, c_{x}{}^{-1}_{1}, c_{x}{}^{0}_{1}, c_{x}{}^{1}_{1}, c_{y}{}^{-1}_{1}, c_{y}{}^{0}_{1}, c_{y}{}^{1}_{1}, \\ & c_{z}{}^{-1}_{1}, c_{z}{}^{0}_{1}, c_{z}{}^{1}_{1}, \ldots c_{x}{}^{-K}_{K} \ldots c_{z}{}^{K}_{K})^{\top} . \end{aligned}$$

One has to realize that the Fourier parametrization is just one possibility for the parametric description of contours. Alternative methods, for example deformable superquadrics, have also been proposed in the literature (Terzopoulos and Metaxas, 1991). We would like to emphasize that any reasonable parametric shape model can be used within the presented formalism. We preferred to use Fourier parametrization as it, in contrast to superquadrics, imposes no predetermined symmetries upon the object surface and has no preferred directions in space. It is complete; no additional formalisms are needed to rotate, translate, taper, bend, twist or locally deform the model, as the surface of any simply connected object can be represented to any degree of detail in the simple yet comprehensive formalism.

In the following we summarize the surface parametrization procedure which is described in detail in Brechbühler *et al.* (1992, 1995).

4.2. The surface data structure

Medical CT or MRI images are examples of volumetric data. For each cuboidal cell (volume element or voxel) in a certain volume we have one or more measurements. When segmentation succeeds, one anatomical unit can be characterized by a binary data volume, in which every voxel contains either 1, which means it belongs to the unit, or 0, meaning it is in the background. The object is then the set of '1' voxels and can be pictured as a collection of small cubes, adopting the *cuberille* notion (Hermann *et al.*, 1979). The surface of a voxel object is a set of unit squares, all parallel to one of the three coordinate planes yz, xz or xy. The edges and vertices that bound the faces are also parts of the surface, which is represented as a data structure that reflects geometry as well as neighbourhood relations.

4.3. Parametrization of closed surfaces

A key step in the shape description of a surface is its mapping to the parameter space, the sphere Ω_3. Any point on the surface must map to exactly one point on the sphere, and vice versa. The location on the sphere corresponding to a surface point

defines the *parameters* of the point. It can be represented as two polar or three Cartesian coordinates, related through the bijection

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \sin\theta\ \cos\phi \\ \sin\theta\ \sin\phi \\ \cos\theta \end{pmatrix}.$$

Mapping a surface to the sphere assigns parameters to every surface point; therefore we call it *parametrization*. The mapping must be continuous, i.e. neighbouring points in one space must map to neighbours in the other space. It is possible and desirable to construct a mapping that preserves areas. Narrowing to the cuberille notion, Figure 10 symbolically illustrates this mapping of a selected facet from the object surface to a portion of Ω_3. It is not possible in general to map every surface facet to a spherical square: distortions cannot be avoided, but they should be minimal.

It emerges that the parametrization, i.e. the embedding of the object surface graph into the surface of the unit sphere, is a constrained optimization problem. The following paragraphs define the meaning of *variables*, *objective* (goal function), *constraints* and *starting values* in this context.

4.3.1. *Variables*

The coordinates of all vertices vary in the optimization. Using two (e.g. spherical) coordinates per vertex would be the most economic representation with respect to storage space, but it would make the equal treatment of all spatial directions difficult and pose the problem of discontinuity and singularities in the parameter space. We prefer Cartesian coordinates (u, v, w) for representing a location on the sphere, introducing one virtual degree of freedom per vertex. The number of variables is three times the number of vertices.

4.3.2. *Constraints*

Two kinds of equalities and one kind of inequality constrain the values that the variables can take.

(i) Every vertex must lie on the unit sphere in parameter space, i.e. $u^2+v^2+w^2 = 1$. This constraint compensates for the virtual degree of freedom and forces.

(ii) We ask for *area preservation*; any object surface region must map to a region of proportional area on the sphere. We include one constraint for each elementary facet: the area of the spherical quadrilateral must be 4π divided by the total number of faces.

(iii) All quadrilaterals on the sphere must remain convex; no angle α_k may become negative or exceed π.

4.3.3. *Objective function*

The *goal* is to minimize the distortion of the surface net in the mapping. It must tend to make the shape of all the mapped faces as similar to their original square form as possible. To fulfil this goal perfectly, a facet should map to a 'spherical square' (see Figure 10). This can in general not be reached for all faces, and we need to trade off between the distortions made at different vertices. We observe that the ideal shape of any face, a spherical square, minimizes the circumference $\sum_{i=0}^{3} s_i$ of any spherical quadrilateral with a given area. At the same time it maximizes $\sum_{i=0}^{3} \cos s_i$. These two measures are similar, but not equivalent if summed over the whole net, as they trade off distortions differently. The second measure punishes too-long sides more and honours too-short sides less than the first one, which is a desirable effect. It is also simpler to calculate; the cosine of a side is the dot product of its endpoints.

4.3.4. *Starting values*

The variables in our optimization are the positions on the unit sphere to which the vertices are mapped, and *initial values* mean a first rough mapping of the object's surface to the sphere. It is important for the optimization algorithm that the sphere be completely covered with faces and none of them overlap, even in the beginning.

Arbitrarily, two vertices are chosen as poles and a surface path connecting them as a date line. Discretizing and solving the Poisson equations $\Delta\theta = 0$ and $\Delta\phi = 0$—with appropriate boundary conditions for the poles and the date line—yields values (θ, ϕ) for each vertex. Figure 11 shows the initial parametrization thus defined for the small object from Figure 10.

The optimization moves the vertices around on the sphere. The poles lose their special meanings, and all faces get equal area. The mapping converges to a state minimizing overall distortions. The same result is reached from all pole assignments (modulo rotations). Figure 12 presents the parametrization result for the same small object.

4.4. Elastically deformable Fourier surface models

Using the parametric surface description previously presented, the parametrized Fourier snake concept can be generalized from 2-D to 3-D. The concept is is similar to the technique proposed by Staib and Duncan (1992b). However, a surface can be arbitrarily sampled based on the variables θ and ϕ (there is no distinguished orientation or position of poles), and the surface energy function can similarly be defined and evaluated as in the 2-D case. The image potential can still use the complete (vector-valued) gradient information

$$E_{\mathrm{I}}(\boldsymbol{r}(\theta,\phi,\boldsymbol{p})) = \pm \int\int_A \nabla I\,(\boldsymbol{r}(\theta,\phi,\boldsymbol{p})) \cdot \boldsymbol{n}(\boldsymbol{r}(\theta,\phi,\boldsymbol{p}))\,\mathrm{d}A$$

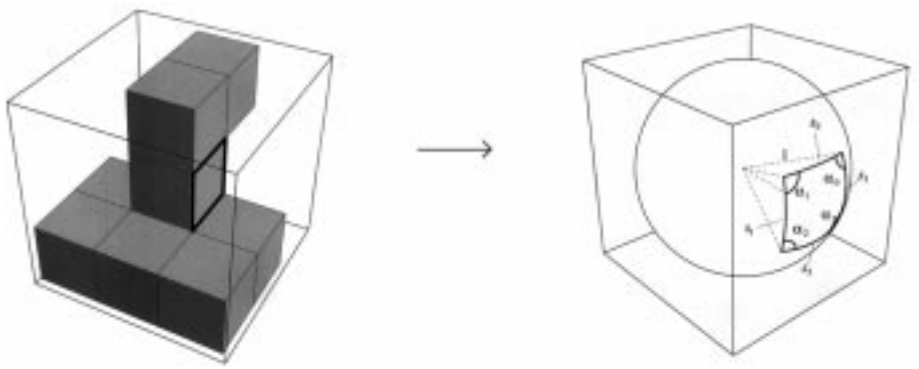


Figure 10. Every facet on the object surface maps to a spherical quadrilateral. Its sides are geodesic arcs on the sphere. On the unit sphere, a side s_i is equal to the corresponding centre angle. The quadrilateral in this illustration is special in that its sides $s_0 \ldots s_3$ are equal and its angles $\alpha_0 \ldots \alpha_3$ are equal: it is a spherical square.

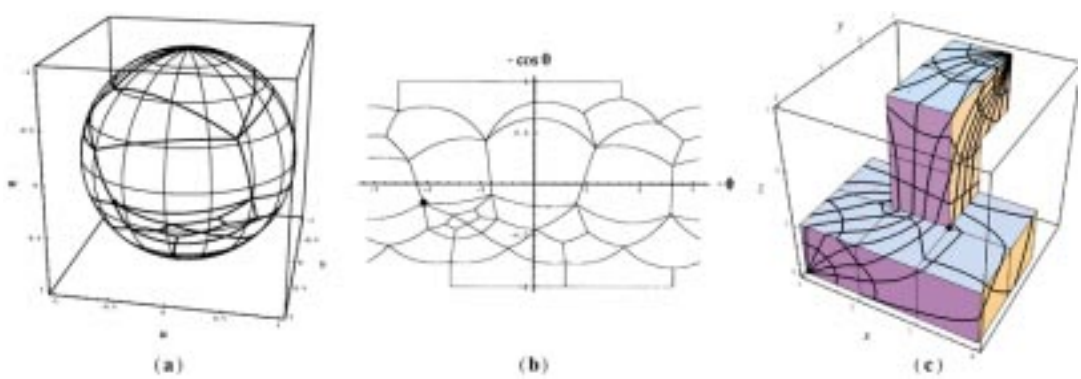


Figure 11. Optimization starts in this state, which is plotted in three ways. (**a**) The surface net is plotted in thick lines on the spherical parameter space. (**b**) ϕ and $\cos\theta$ are interpreted as Cartesian coordinates, giving a true-area cylinder projection. The horizontal lines at ± 1 are the poles. (**c**) Conversely, the globe coordinate grid is drawn over the object. For comparison, one vertex is marked with a black dot in all diagrams.

where $\mathrm{d}A$ is the surface element and $\boldsymbol{n}$ is the surface normal vector. Using $\boldsymbol{a} = \partial\boldsymbol{r}/\partial\phi \times \partial\boldsymbol{r}/\partial\theta = \boldsymbol{n}\,\mathrm{d}A/\mathrm{d}\phi\,\mathrm{d}\theta$, we derive

$$E_{\mathrm{I}}(\boldsymbol{r}) = \pm\frac{\int_0^{\pi}\int_0^{2\pi} \nabla I\,(\boldsymbol{r}(\theta,\phi,\boldsymbol{p}))\cdot\boldsymbol{a}(\boldsymbol{r}(\theta,\phi,\boldsymbol{p}))\,\mathrm{d}\phi\,\mathrm{d}\theta}{\int_0^{\pi}\int_0^{2\pi} \|\boldsymbol{a}(\boldsymbol{r}(\theta,\phi,\boldsymbol{p}))\|\,\mathrm{d}\phi\,\mathrm{d}\theta}.$$

We use the internal energy term $E_{\mathrm{D}}(\boldsymbol{r}(\theta,\phi,\boldsymbol{p}))$ to avoid sharp discontinuities in the surface normals. The curvature of the surface can be described by the principal curvatures κ_1 and κ_2, which are combined to

$$\kappa = \sqrt{\kappa_1^2 + \kappa_2^2},$$

creating a measure for the curvature at every point on the surface. As in the 2-D case we want to limit its derivative, resulting in the regularization term

$$E_{\mathrm{D}}(\boldsymbol{r}) = \int_0^{\pi}\int_0^{2\pi} \|\nabla\kappa(\phi,\theta,\boldsymbol{p})\|^2\cdot\|\boldsymbol{a}(\phi,\theta,\boldsymbol{p})\|^2\,\mathrm{d}\phi\,\mathrm{d}\theta.$$

After the definition of the total energy, the problem is completely analogous to the 2-D case: we have to determine the parameters that generate a surface which minimizes this energy. We used exactly the same minimization procedures as in the 2-D case.

For the average surface before and after the deformation, the mapping will not meet the constraint of constant density exactly, nor will it exactly minimize distortions. However, the mapping has been individually adapted to each of the training objects, which are chosen to span the whole range of expected biological variability. A smooth interpolation between them can be expected to reproduce close to uniform and optimal

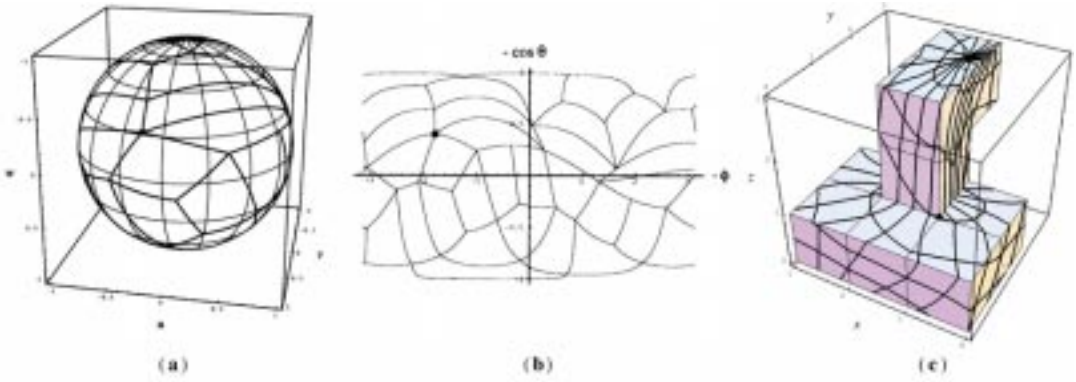


Figure 12. The parametrization achieved by the optimization, visualized as in Figure 11.

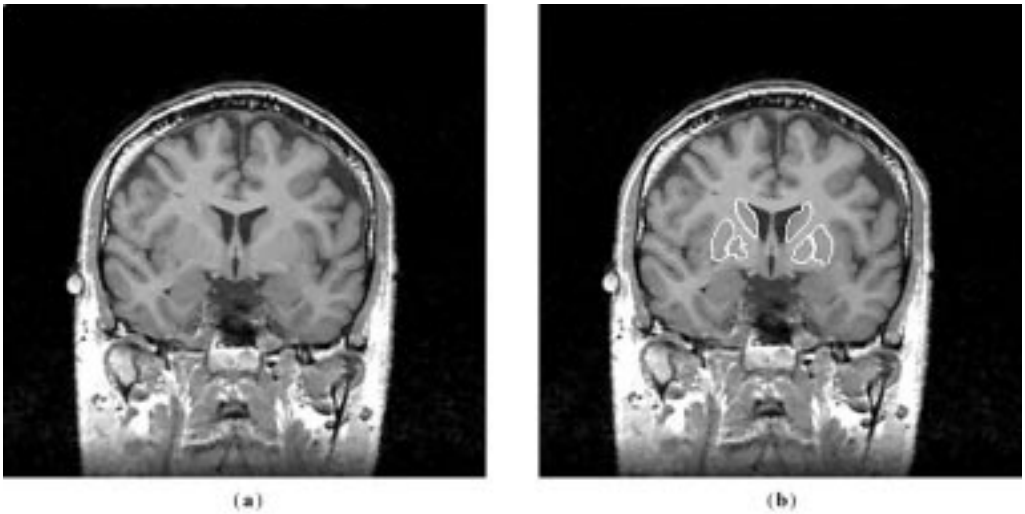


Figure 13. Manual segmentation of the putamen, the caudate nucleus and the globus pallidus. The images show a coronal slice from a 3-D data set (**a**) with overlay of the contours of segmented objects (**b**).

parametrization for the fitted surface. Exact uniformity and optimality is a means to establish correspondence among the objects of the training set, but it is not crucial for the success of the segmentation.

4.5. 3-D segmentation of deep grey matter structures

The training set consists of a collection of 30 3-D MRI data volumes of the human brain, where deep grey matter structures (putamen, caudate nucleus and globus pallidus) have been manually segmented. Figure 13 shows a coronal slice through one of the volumes and the outlines of the manually segmented objects. In the case of the putamen and globus pallidus one can see that there is practically no grey-level evidence to separate the two objects. Only *a priori* anatomical knowledge will allow their segmentation, which clearly demonstrates the need for model-based 3-D segmentation procedures.

The modules for 3-D surface parametrization and Fourier description, for the calculation of eigenmodes and for 3-D segmentation by restricted elastic deformations, are implemented and ready for tests and validation. Preliminary results demonstrate two different procedures of the complete segmentation system:

- Figure 14 illustrates the generation of parametric surface descriptions from binary segmentations. The surfaces of the putamen (left) and the caudate nucleus (right) use a spherical harmonic approximation up to degree 6 and 8, respectively.

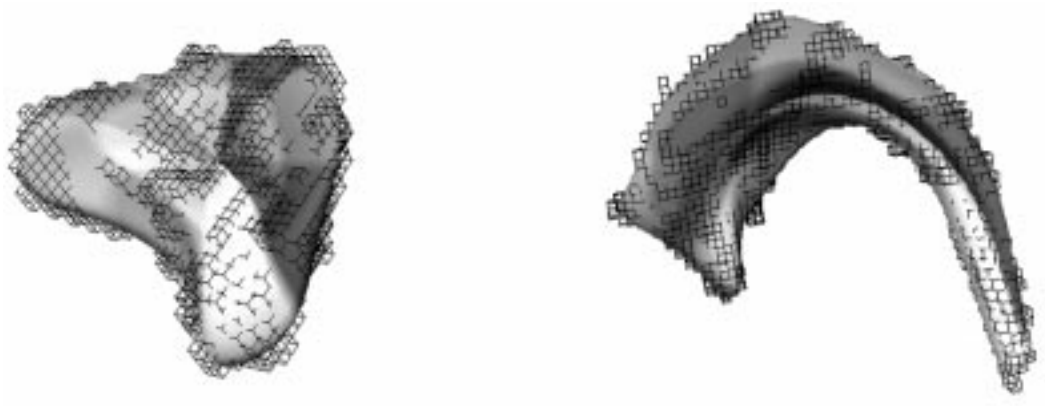


Figure 14. Parametrized description by spherical harmonics up to degree 8 of the surface of deep grey matter organs. The putamen is shown on the left, the caudate nucleus on the right. The original voxel object is overlaid as a wire-frame structure of the voxel edges.

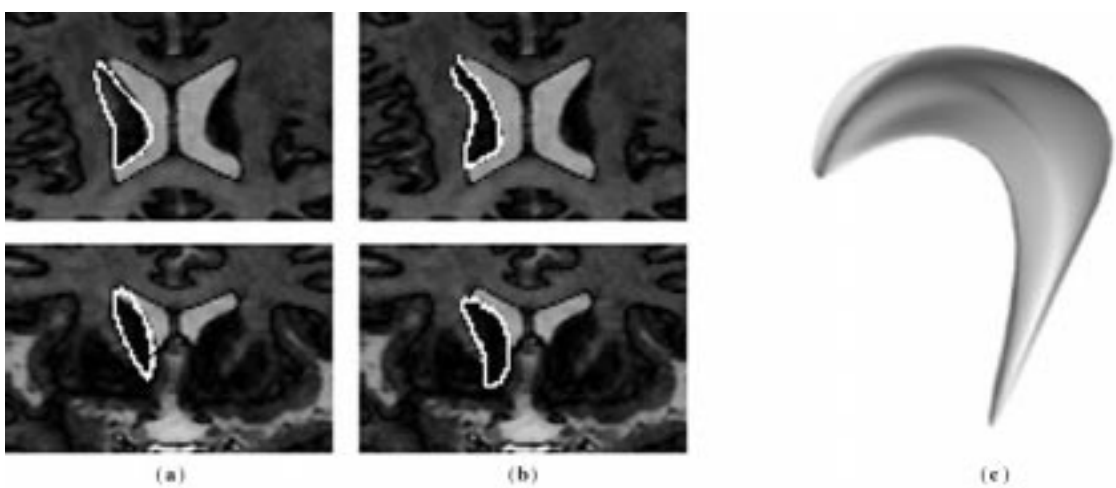


Figure 15. Segmentation of the left caudate nucleus by 3-D Fourier snakes. Images in panel **a** represent the initial placement of the 3-D model as axial (top) and coronal (bottom) cuts (with spherical harmonics up to degree 3 with 48 parameters). Panel **b** shows the final segmentation result. A graphical display of the elastically deformed model representing the result of the 3-D segmentation is shown to the right (**c**). The final optimization was based on spherical harmonics up to degree 5 (108 parameters).

- Figure 15 demonstrates the segmentation of the caudate nucleus from original grey-valued volume data by the 3-D Fourier snake procedure. Here, the initial placement of the model surface was performed automatically by calculating an initial placement of a spherical harmonic surface of low degree (degrees 0–3). The images in panels a and b show the initialization and final result after elastic deformation in axial and coronal cuts. For the elastic fit we used spherical harmonics up to degree 5. Panel c illustrates a surface rendering of the segmentation result. 15–20 minutes were needed for the elastic fit of the models on a Sparc 10/41 processor.

We are currently calculating the surface parametrization and the description by spherical harmonics of all the 30 segmentations. Based on these results, the deformation eigenmodes for these objects can be determined. The segmentation itself, i.e. the initial placement and restricted elastic deformation, will be carried out with the Fourier snake procedure as shown in Figure 15. However, optimization is performed in the subspace of a small set of eigenmodes rather than the original space of the parameters, as discussed in section 3 for the 2-D case.

5. CONCLUSIONS

Automated, robust segmentation of medical images most often needs *a priori* anatomical knowledge. Typical cases are the segmentation of healthy organs, which present restricted anatomical variability, or the segmentation of organs if only incomplete evidence for boundaries is given by the grey-valued images, requiring an 'intelligent guess' about the position of the object boundary.

Geometric organ models and the statistics of their *normal (expected) variation* seem to offer a promising solution to this problem. We proposed the use of the Fourier parametrization of our models, followed by a statistical analysis of a training set, providing mean organ models and their eigendeformations. Elastic fit of the mean model in the subspace of eigenmodes restricts possible deformations and finds an optimal match between the model surface and boundary candidates.

We demonstrated the complete procedure with the determination of the outline of the corpus callosum on a 2-D MRI slice of the human brain. We have shown that the selection of an anatomically defined reference coordinate system allows the inclusion of the variation of the spatial position and orientation into the description of the model variation. This procedure provides a homogeneous framework for the complete analysis, while adding only minimal, well defined user interaction to the procedure. The initial placement of the average model is performed by a coarse elastic deformation defined by only the largest eigenmodes, helping to drive the following optimization into the 'correct' local optimum.

In this paper we presented preliminary results of a full 3-D generalization of the Fourier snake procedure. While the modules are implemented and ready for analysis, we still need to spend effort into the 3-D model building. We demonstrated that manually segmented deep grey matter organs of the human brain can be described by spherical harmonic descriptors. We further demonstrated the applicability of an unrestricted 3-D Fourier snake procedure for the segmentation of these organs. In contrast to similar research projects, we could overcome the bottleneck of surface parametrization by applying a new parametrization technique which provides a homogenous parametrization of an arbitrarily shaped simply connected object. Further, the novel method represents a true 3-D extension of the 2-D procedure, which significantly differs from combining results of a stack of 2-D slices.

Our ultimate goal is to provide an automated 3-D segmentation procedure that needs only minimal user interaction. This manual interaction would consist of the selection of a few clearly defined landmarks and could therefore be carried out also by non-experts. The segmentation itself would then run fully automatically. Such an elastic deformation procedure restricted by prior knowledge about the deformation range would also find applications in *tracking* problems, where objects once defined with a relatively high expense could be automatically tracked in dynamic image sequences.

ACKNOWLEDGEMENTS

The authors thank Ron Kikinis (Brigham and Women's Hospital, Harvard Medical School, Boston) for organizing the MR acquisitions and for discussions in regard to the clinical needs for robust 3-D image segmentation. Martha Shenton (Brain Imaging Laboratory, Dept. of Psychiatry, Harvard Medical School, Boston) is kindly acknowledged for providing the MR data and the manually segmented 3-D data.

REFERENCES

Brechbühler, C., Gerig, G. and Kübler, O. (1992) Surface parametrization and shape description. In *Proc. VBC'92 Conf.*, pp. 80–89.

Brechbühler, C., Gerig, G. and Kübler, O. (1995) Parametrization of closed surfaces for 3-D shape description. *CVGIP: Image Understanding*, 61, 154–170.

Cohen, I., Cohen, L. D. and Ayache, N. (1992) Using deformable surfaces to segment 3-D images and infer differential structures. *CVGIP: Image Understanding*, 56, 242–263.

Cootes, T. F. and Taylor, C. J. (1992) Active shape models – 'smart snakes'. In *Proc. British Mach. Vision Conf. '92*, pp. 266–275, Springer.

Cootes, T. F., Hill, A., Taylor, C. J. and Haslam, J. (1993) The Use of Active Shape Models For Locating Structures in Medical Imaging. In *Proc. IPMI'93*, pp. 33–47, Flagstaff USA.

Gerig, G., Martin, J., Kikinis, R., Kübler, O., Shenton, M. and Jolesz, F. (1991) Automatic Segmentation of Dual-Echo MR Head Data. In *Proc. IPMI'91 (Wye)*, pp. 175–187.

Greiner, W. and Diehl, H. (1964) Theoretische Physik – Ein Lehr- und Übungsbuch für Anfangssemester, Band 3: Elektrodynamik, Verlag Harri Deutsch, Zürich und Frankfurt am Main, pp. 61–65.

Hermann, G. T. and Liu, H. K. (1979) Three-dimensional display of human organs from computer tomograms. *CGIP*, 9, 1–21.

Hill, A. and Taylor, C. J. (1992) Model-based image interpretation using genetic algorithms. *Image Vision Comp.*, 10, 295–300.

Hill, A., Cootes, T. F. and Taylor, C. J. (1992) A Generic System for Image Interpretation Using Flexible Templates. In *Proc. British Mach. Vision Conf.'92*, pp. 276–285, Springer.

Hill, A., Thornham, A. and Taylor, C. J. (1993) Model-Based Interpretation of 3-D Medical Images. In *Proc. British Mach. Vision Conf.'93*, pp. 239–348, BMVC Press.

Kass, M., Witkin, A. and Terzopoulos, D. (1988) Snakes: active contour models. *Int. J. Comp. Vision*, 1, 321–331.

Kuhl, F. P. and Giardina, C. R. (1982) Elliptic Fourier features of a closed contour. *CVGIP*, 18, 236–258.

Martin, J., Pentland , A. and Kikinis, R. (1994) Shape Analysis of Brain Structures Using Physical and Experimental Modes. In *Proc. CVPR'94*, pp. 752–755.

NAG Fortran Library Manual – Mark 13, (1988) 1st edn.

Nastar, C. and Ayache, N. (1994) Classification of Nonrigid Motion in 3-D Images using Physics-Based Vibration Analysis. In *Proc. IEEE Workshop on Biomed. Image Anal.*, pp. 61–69, Seattle, WA.

Neuenschwander, N., Fua, P., Székely, G. and Kübler, O. (1994) Initializing Snakes. In *Proc. CVPR'94, IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, June 21–23, Seattle, WA. pp. 658–663.

Pentland, A. and Sclaroff, A. P. (1991) Closed-form solutions for physically based shape modelling and recognition. *IEEE PAMI*, 13, 715–729

Persoon, E. and Fu, K. S. (1977) Shape discrimination using Fourier descriptors. *IEEE Trans. SMC*, 7, 170–179.

Reeves, C. R. (ed.) (1993) *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell, Oxford.

Sclaroff, S. and Pentland, A. P. (1994) On Modal Modelling for Medical Images: Underconstrained Shape Description and Data Compression. In *Proc. IEEE Workshop on Biomed. Image Anal.*, pp. 70–79, Seattle, WA.

Staib, L. H. and Duncan, J. S. (1992a) Boundary finding with parametrically deformable models. *IEEE PAMI* 14, 1061–1075.

Staib, L. H. and Duncan, J. S. (1992b) Deformable Fourier models for surface finding in 3-D images. In *Proc. VBC'92 Conf.*, pp. 90–194.

Terzopoulos, D. and Metaxas, D. (1991) Dynamic 3-D models with local and global deformations: deformable superquadrics. *IEEE PAMI*, 13, 703–714.

Terzopoulos, D., Witkin, A. and Kass, M. (1988) Symmetry-seeking models and 3-D object reconstruction. *Int. J. Comp. Vision*, 1, 211–221.

Vemuri, B. C. and Radisavljevic, A. (1994) Multiresolution stochastic hybrid shape models with fractal priors. *ACM Trans. Graphics*, 13, 177–20.

Parametrization of closed surfaces for 3-D shape description

Ch. Brechbühler, G. Gerig and O. Kübler

Communication Technology Laboratory
Image Science
Swiss Federal Institute of Technology (ETH)
CH-8092 Zurich, Switzerland

March 29, 1996

Abstract

This paper presents procedures for the explicit parametric representation and global description of surfaces of simply connected 3-D objects. The novel techniques overcome severe limitations of earlier methods (restriction to star-shaped objects [1], constraints on positioning and shape of cross-sections [2, 3], and nonhomogeneous distribution of parameter space).

We parametrize the surface by defining a continuous, one-to-one mapping from the surface of the original object to the surface of a unit sphere. The parametrization is formulated as a constrained optimization problem. Practicable starting values are obtained by an initial mapping based on a heat conduction model.

The parametrization enables us to expand the object surface into a series of spherical harmonic functions, extending the concept of elliptical Fourier descriptors for 2-D closed curves [4, 5]. Invariant, object-centered descriptors are obtained by rotating the parameter net and the object into standard positions.

The new methods are illustrated with simple 3-D test objects. Potential applications are recognition, classification and comparison of convoluted surfaces or parts of surfaces of 3-D shapes.

1 Introduction

With the proliferation of high quality volumetric image data, especially for the medical community, and new segmentation methods for multidimensional image data, 3-D objects become available and are ready for structural analysis. Most often, volumetric objects are represented by a binary voxel representation or by a triangulation of the surface. Although these representations allow a 3-D rendering for visually capturing the object properties, both lack descriptive power as they are based on huge lists of voxels or surface elements. Characterizing and understanding shape properties, however, requires a representation which captures global and local shape features with a small number of parameters. Such a concise description could be useful for a comparison of various objects, for finding dissimilarities, for matching objects to predefined models and for an efficient reconstruction and manipulation of objects.

A shape descriptor must be general enough to handle very different shapes, but should be capable of accurately representing global as well as local features of objects. Shape analysis favors object-centered volume- or surface descriptions, e.g. using polynomials, triangulation meshes, generalized cylinders, medial manifolds or spherical harmonics. Furthermore, the resulting description should be independent of the object position and orientation.

Some shape description methods based on mesh-like surface models rely on a robust and reproducible surface parametrization in a two-coordinate space. While tracking a contour in 2-D images is easily done, the extension to higher dimensions is non-trivial and requires the development of new concepts. So far, representation methods for mapping an object surface onto a sphere have been limited to represent only star-shaped or convex objects, as they start from an initial radial surface function $r(\theta, \phi)$ [1] [6]. Staib and

Duncan [3] discuss the use of a parameter space with torus topology, which can be deformed into a tube by squeezing the torus cross-section to a thin ribbon. Closed surfaces are obtained by considering tubes whose ends close up to a point. This approach clearly illustrates some principal difficulties which can also be found in other parametrization techniques.

- The idea of warping a torus to a closed surface poses the problem that the *parameters have different functions*. One parameter defines a kind of spine along which cross-sections are stacked up. The choice of the end-points of this spine decisively determines the solution and even determines whether an object can be parametrized or not.

- Squeezing a circle to a line (as done with the tubular torus cross-section) results in a *nonhomogeneous distribution of parameters* on the object surface. Although continuous, the representation of a line by harmonic functions results in a clustering of parameters near the edges where the tracing direction changes. Furthermore, closing a cylinder at both open ends causes further distortion to the parameter net.

- Warping a torus to a tube and finally to a closed surface shows that the parametrization does *not* result in a *one-to-one mapping* of surface points to parameters. The walls of the tube are traced up and down in order to avoid discontinuities at the open edges, visiting each surface point twice.

Our new approach tries to overcome these limitations. We present a new method that allows a uniform mapping of an object surface into a two-coordinate space with spherical topology. Our aim is the parametrization of arbitrarily shaped simply connected objects containing inclusions and protrusions. As a mapping of convoluted surface structures onto the surface of a sphere introduces distortions, optimization of the distribution of nodes in parameter space becomes necessary. This problem is solved by nonlinear optimization.

Parametrized surfaces can be expanded into spherical harmonics, hierarchically describing global and local shape properties by spatial frequency constituents [7]. A new method for the generation of descriptors which are invariant to translation, rotation and scaling is developed. Invariance is crucial for a comparative analysis of different structures.

This paper is organized as follows: section 2 describes the generation of a relational surface data structure, important both for the initial mapping by a diffusion algorithm and for the optimization step. In the section, 3 we present the new parametrization technique including an initial mapping to obtain good starting values and nonlinear optimization. Section 4 discusses the expansion of a 3-D shape into a set of spherical harmonics and the generation of invariant descriptors.

2 Surface data structure

A 3-D binary image containing a single simply 6-connected object represents our input data. We adopt the *cuberille* notion [8]. To allow working with the surface of the object, it must be represented as a data structure that reflects the geometry as well as topological relations. This data structure stores information about all the square faces separating object and background voxels and all the edges and vertices bounding these faces. This initial data structure represents a complete description of the object surface; similar structures are often used for 3-D display purposes [9].

For a n_x by n_y by n_z data volume, the voxel coordinates have the ranges $0 \leq x < n_x$, $0 \leq y < n_y$ and $0 \leq z < n_z$. We interpret each voxel as a cube extending from x to $x+1$, from y to $y+1$ and from z to $z+1$, where (x, y, z) are the nominal coordinates of the voxel. The voxel center is placed at $(x + \frac{1}{2}, y + \frac{1}{2}, z + \frac{1}{2})$ to retain integer coordinates for the corners.

The surface data structure is organized around the vertices. Each surface vertex is listed once in the structure, which gives it a unique identifying (id) number. The entry of a vertex specifies its Cartesian coordinates (x, y, z) and a list of its direct and diagonal neighbors. The neighbors are given in counter-clockwise sequence around the vertex when viewed from outside the object; therefore, direct and diagonal neighbors alternate (see Figure 1).

The following algorithm generates the surface data structure. Object voxels touching the border of the data volume would require a special treatment. Therefore, we make sure that all border voxels belong to

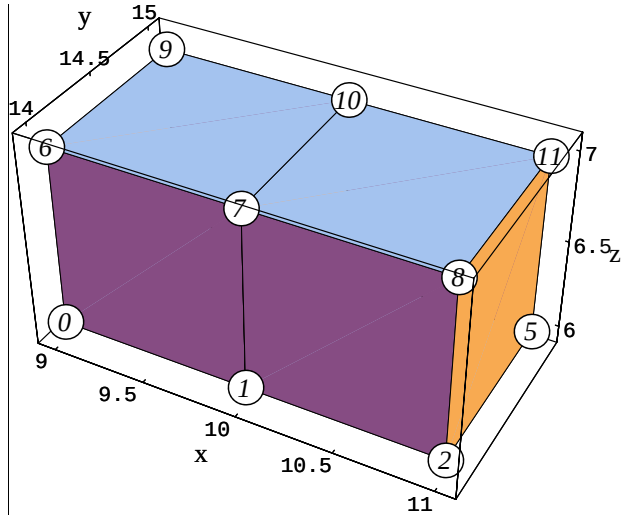


Figure 1: A two-voxel example object illustrating the surface data structure centered on the vertices. The numbers of the vertices are shown in circles (vertices 3 and 4 are hidden). The data structure entry for a vertex represents its Cartesian coordinates and a list of neighboring nodes. The entry for vertex 7 is $\{\{x$=`10, y`=`14, z`=`7`$\}$, *neighbors*=$\{$`6, 0, 1, 2, 8, 11, 10, 9`$\}\}$. Appendix A lists the complete surface data structure of this object.

the background. We define an *interior vertex* as the common corner of eight voxels of the data volume. In three nested loops – varying the coordinates x from 1 to $n_x - 1$, y from 1 to $n_y - 1$, and z from 1 to $n_z - 1$, respectively – we visit all the interior vertices. Let the z-loop be the outermost, so that z varies the slowest and numbers the current plane. If the eight incident voxels of a vertex are homogeneous, i.e. all in the object or all in the background, the vertex is ignored. Otherwise (if the incident voxels are mixed), a new surface vertex has to be entered in the data structure. A surface vertex in plane z may have neighboring surface vertices in the planes $z-1$, z and $z+1$. Therefore, surface vertices must be detected and assigned identifying numbers one plane ahead of time, and the id numbers of the current (z) and previous ($z-1$) plane must be kept at hand as well. For each surface vertex in the current plane, the Cartesian coordinates are put into the entry of the vertex. Starting at a 6-neighbor surface vertex, we cycle around the central vertex, going over all incident faces until we are back at the starting neighbor vertex. The list of the id numbers of all visited direct and diagonal neighbor vertices completes the entry of the central vertex.

Only vertices are stored in the data structure, but information about edges and faces of the surface is also present in the neighbor lists. For instance, vertex 7 in Figure 1 has the neighbor list $\{$`6, 0, 1, 2, 8, 11, 10, 9`$\}$. Taking every second number in this list – 6, 1, 8, 10 – reveals that edges to nodes 6, 1, 8 and 10 emanate from node 7. Overlapping triples of neighbors – $\{6, 0, 1\}$, $\{1, 2, 8\}$, $\{8, 11, 10\}$ and $\{10, 9, 6\}$ (by wrapping around to the first neighbor) – give the four faces $\{7, 6, 0, 1\}$, $\{7, 1, 2, 8\}$, $\{7, 8, 11, 10\}$ and $\{7, 10, 9, 6\}$, all written counterclockwise. In this way, every face of the surface is mentioned four times, once by each corner. Our algorithm, however, requires an additional table that lists every face exactly once. We generate it by visiting all vertices, putting a face in the table only when it is defined from the corner with the smallest id number, i.e. when it is mentioned for the first time.

The data structure generalizes in a very natural way to the case of open surfaces, or surface patches, where vertices on the border of the surface have an odd number of neighbors. All surface vertices that are not part of a border of the surface are called *inner vertices*. Their characteristic is an even number of neighbors.

The possibility of representing surface patches is not exploited here: as the object lies completely within our data volume, we always find a closed surface. All vertices in the list are inner vertices of the surface, and therefore, they all have an even number of neighbors.

The correspondence between the surface net and a graph becomes clear when a vertex is interpreted as a node in the graph, an edge as an arc and a face as a mesh (four-cycle)[10]. For a simply connected

object we get a planar graph with the following topological properties: Four edges bound each face, and each edge bounds two faces and is bounded by two vertices. Depending on the local connectivity, each vertex bounds three to six edges. There are exactly two more vertices than faces; this follows from Euler's relation: $n_{vert} = n_{face} + 2$.

A surface with its two degrees of freedom is characterized by a polygonal description based on vertex coordinates with three spatial coordinates. Seeking for an appropriate parametrization, however, would require a description based on two parameters.

3 Surface parametrization

The parametrization, i.e., the embedding of the object surface graph in the surface of the unit sphere, can be posed as a constrained optimization problem. The following paragraphs define the meaning of *variables*, *objective* (goal function), *constraints* and *starting values*.

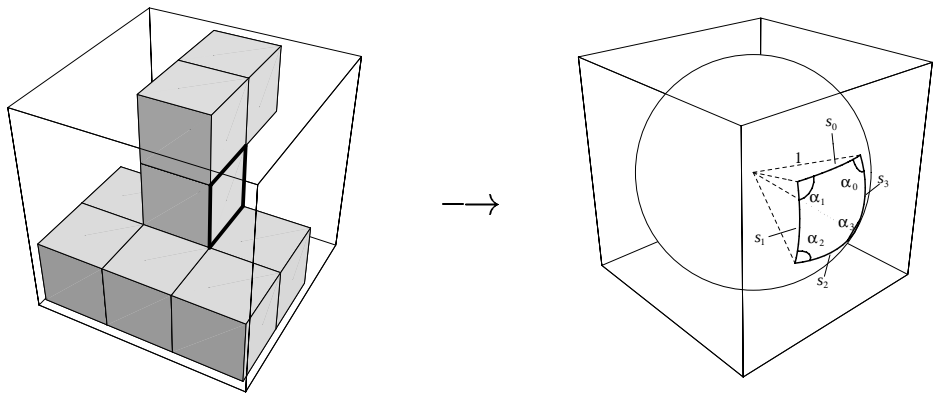


Figure 2: Every single face on the object's surface is mapped to a spherical quadrilateral. The sides of a spherical polygon are geodesic arcs on the sphere surface. As the sphere has unity radius, the length of a side s_i is equal to the corresponding center angle (in radian). The quadrilateral in this illustration is special in that its four sides $s_0 \cdots s_3$ are equal and its four angles $\alpha_0 \cdots \alpha_3$ are equal: it is the spherical analogue of a square.

The *variables* of the optimization problem are the coordinates of all vertices. Using two (e.g. spherical) coordinates per vertex would be the most economic representation with respect to storage space, but this would make the equal treatment of all spatial directions difficult and pose the problem of discontinuity and singularities in the parameter space. Therefore, we prefer Cartesian coordinates (u, v, w) for representing a location on the sphere, introducing one virtual degree of freedom per vertex.

There are three kinds of *constraints* on the variables. First, the Euclidean norm of the coordinates of any vertex must be 1. This constraint compensates for the virtual degree of freedom and forces every vertex to lie on the unit sphere in parameter space. Second, we ask for area preservation, which in our context means that any object surface region must map to a region of proportional area on the sphere. To satisfy this requirement, we include one constraint for each spherical quadrilateral (see Figure 2) that corresponds to an elementary face: its area must be 4π divided by the total number of faces. Third, no angle of any spherical quadrilateral must become negative or exceed π. In contrast to the first two kinds of constraints, these are inequalities. They can be stated as $\sin \alpha_i \geq 0$, $\forall i$. This adds four inequality constraints for each face.

The *objective* is to minimize the distortion of the surface net in the mapping. It is conceptually similar to angle preservation, and it must tend to make the shape of all the mapped faces as similar to their original square form as possible. To fulfill this goal perfectly, a facet should map to a "spherical square" (see Figure 2). This can never be reached exactly for all faces except when the object has a very special form, e.g., consists of one single voxel. There are several ways to trade off between the distortions made at

different vertices. We observe that the ideal shape of any face, a spherical square, minimizes circumference $\sum_{i=0}^{3} s_i$ of any spherical quadrilateral with a given area. At the same time it maximizes $\sum_{i=0}^{3} \cos s_i$. These two measures are similar, but not equivalent if summed over the whole net, as they trade off among distortions differently. The second measure punishes too long sides more and honors too short sides less than the first one, which is a desirable effect. It is also simpler to calculate; the cosine of a side - and of the respective central angle - is the dotproduct of the vectors from the sphere center to the endpoints of the side. In a sum over all four sides of all faces, each edge of the graph appears twice. We define the objective function as half that sum, which is the same as the sum of the cosines of the lengths of all edges.

The variables in our optimization are the positions on the unit sphere to which the vertices are mapped. Therefore, *starting values* in this context means an initial mapping of the object's surface to the sphere. It is important for the optimization algorithm that the sphere be completely covered with faces and none of them overlap. The following subsection describes the construction of an initial parametrization satisfying this requirement. An earlier version of the procedure is described in [11].

3.1 Initial parametrization

The first mapping or parametrization, respectively, is done in polar coordinates. The two polar coordinates θ and ϕ are determined for all vertices in two separate steps. Two vertices have to be selected as the poles for this process. The choice of these poles is not critical, as the optimization process removes all its influences except a rotation in parameter space. Selecting two poles which lie close together, however, results in a poor initial parametrization. The optimization will converge to the same solution, but it takes more iteration steps. We select the two vertices with maximal and minimal z coordinate in object space, respectively; the y and x coordinates are used to break ties. In our ordering of the surface vertices, they are the first and the last vertex.

A fully worked example with the "two-voxel" object of Figure 1 is presented in appendix A. It serves as a concrete and detailed illustration of the general concepts exposed below.

Latitude from diffusion. Latitude θ should grow smoothly from 0 at the north pole to π at the south pole. In this context, θ is not a free variable but rather an unknown function (of the location on the object) that we are looking for. To assign a latitude value with the desired property to every node, we formulate the corresponding continuous problem as Laplace's equation $\nabla^2\theta = 0$ (except at the poles), with Dirichlet conditions $\theta_{north} = 0$,$\theta_{south} = \pi$ for latitude θ[12]. A physical analogy is heat conduction: we heat the south pole up to temperature π , cool the north pole to temperature 0 and ask for the stationary temperature distribution on the heat-conducting surface. As usual in the discrete case, the Laplacian is approximated by finite second differences of the available direct neighbors, which in our case implies that every node's latitude (except the poles') must equal the average of its neighbors' latitudes. These conditions form a sparse set of linear equations, which can be written in the form $A'\boldsymbol{\theta}' = \mathbf{b}'$, where A' is a $n_{vert} \times n_{vert}$ matrix, $\boldsymbol{\theta}' := (\theta_0, \theta_1 \ldots \theta_{n_{vert}-1})^T$ and $\mathbf{b}'$ is an n_{vert} vector of constants. The border conditions supply two equations and the average property defines $n := n_{vert} - 2$ equations. Applying the border conditions $\theta_0 = \theta_{north}$ and $\theta_{n_{vert}-1} = \theta_{south}$ results in the reduced $n \times n$ system $A\boldsymbol{\theta} = \mathbf{b}$, where $A = (a_{1,1}, a_{1,2} \ldots a_{n,n})$ is symmetric and $\boldsymbol{\theta} := (\theta_1 \ldots \theta_n)^T$

The algorithms used to set up the matrix A and the right hand side vector $\mathbf{b}$ make use of the well-organized surface data structure, as illustrated below in pseudo-code notation. The sparsity of the matrix is exploited; only the few non-zero entries (4 to 7 per row) are stored. This saving is essential for the treatability of larger objects.

Set up matrix A:

```
for vertex = 1...n
    a_{vertex,vertex} := number of direct neighbors;
    for the direct neighbors of vertex
        if the neighbor is not a pole
            a_{vertex,neighbor} := -1;
```

Set up constant vector b:

```
set all entries of b to 0;
for the direct neighbors of south_pole
    b_neighbor := π;
```

This sparse symmetric linear system of equations is solved with the PILS package [13]. The solution has the important property that latitude θ varies monotonically between the poles, since there can be no local extremum by virtue of the maximum principle [14]. Figure 3a shows the resulting θ for a simple test object, "duck".

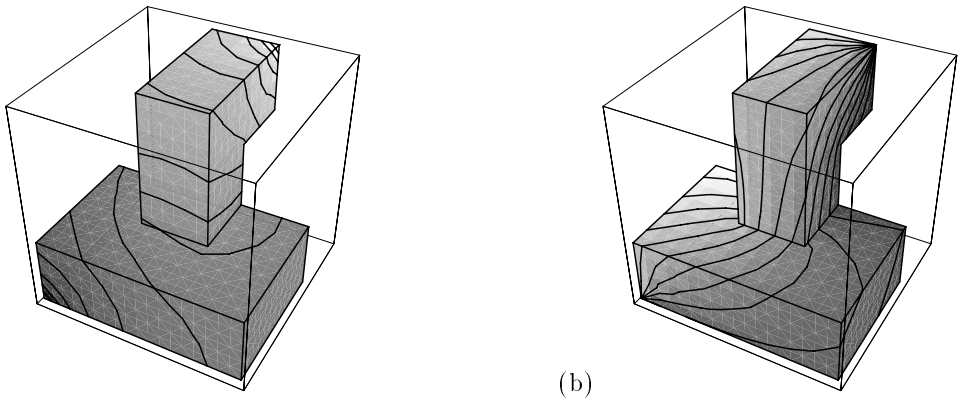


Figure 3: The simple object "duck" consisting of nine voxels is used for illustrating the initial parametrization. The north pole is at the lower left, the south pole at the upper right. Latitude is mapped on the object's surface as a grey value in *a*; iso-latitude lines are drawn every $\frac{\pi}{16}$. Longitude is shown in *b*; iso-longitude lines ("meridians") are $\frac{\pi}{8}$ apart.

Longitude from diffusion. Unlike latitude, longitude is a cyclic parameter: When we walk around a sphere counterclockwise (seen from the north), longitude keeps increasing monotonically all the time, but there must be a place where longitude leaps back by 2π. A global longitude parameter always has a discontinuous line running from pole to pole, and the step height is 2π. Consider, as an analogy, local time on every spot of the globe; the date line is a 24 hour discontinuity, but not a meridian. In our problem, the choice of the date line is immaterial: it just has to connect the two poles. The date line is chosen as a path with steepest latitude ascent in each of its nodes. The values crossing the date line from west to east are decremented by 2π, values propagated to the west are incremented by 2π. The poles and all links to them are removed from the net, making the topology of the net that of a tube. Longitude remains undefined for the poles. The cyclic Laplace equation $\nabla^2\phi = 0$ (with date line) again corresponds to a system of linear equations in the discrete case.

The new system of linear equations is structurally identical to the one for latitude. Typically, only a small part of the equations is different. The matrix for longitude ϕ differs from the one for latitude θ only by the values of six diagonal entries, corresponding to the three neighbors of each pole. This similarity simplifies setting up and solving the new system.

Due to the cyclic boundary conditions, the solution ϕ is defined only up to an additive constant. The linear equations are dependent, and the system is singular. To make it regular, we have to specify the longitude of one vertex. We arbitrarily set $2\phi_1 = 0$. This equation can be added to any row of the system. We add it to the first row.

The following portions of pseudo-code update the matrix and generate a new right hand side vector.

Modify matrix A:

```
for both poles
    for the direct neighbors of pole                    cut link to pole
        a_{neighbor,neighbor} -= 1;
a_{0,0} += 2;                                           for φ_1 = 0
```

Set up constant vector b:

```
for row := 1...n do
    b_row := 0;
previous := north pole;
here := 1;                                              any neighbor of north pole
maximum := 0.0;
```

```
while (here != south pole)                                  walk on date-line
    for the direct neighbors of here
        if θ_neighbor > maximum then
            maximum := θ_neighbor;
            nextpos := position of neighbor;
        if neighbor == previous then
            prevpos := position of neighbor;
    for the direct neighbors clockwise between prevpos and nextpos do
        add 2π to b_neighbor;
        subtract 2π from b_here;                            totally: 2π · # west_neighbors
    previous := here;
    here := neighbor of here indicated by nextpos;
```

In spherical coordinates, longitude is undefined at the poles. We arbitrarily set $\phi_{north} = \phi_{south} = 0$. Figure 3b illustrates the resulting ϕ for the "duck" test object.

For every node of the net, we now have computed a latitude θ and a longitude ϕ. This defines a continuous, unique mapping from the surface of the original object to the surface of a sphere. The spherical parameters ϕ and θ can also be used in a flat Cartesian coordinate system as shown in Figure 4b, which gives an overview of the whole unfolded net. The right border wraps around to the left border, and vice versa. The top and bottom borders stand for the poles. The θ axis points up to produce the same orientation as in the other figures; the south pole is at the top. The polar coordinates are transformed to Cartesian and yield starting values for the optimization.

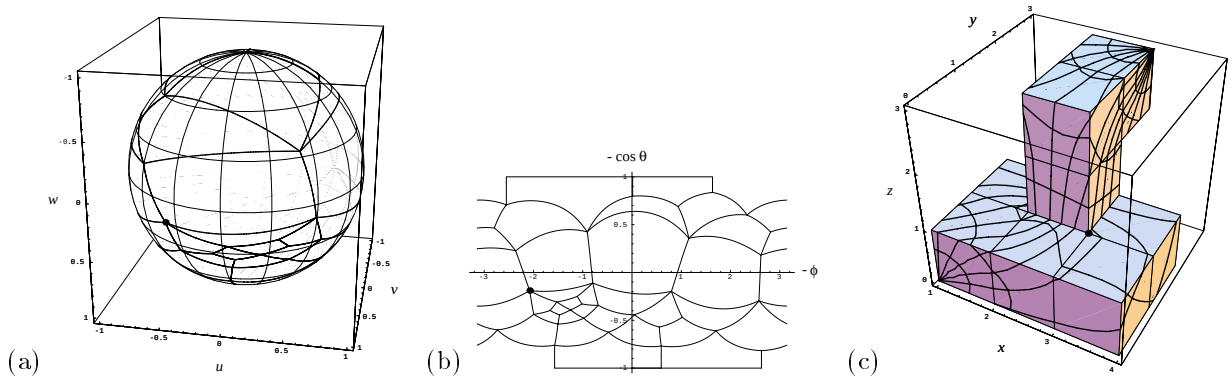


Figure 4: Diffusion yields the initial parametrization, which is visualized in three different ways. *(a)* The surface net is plotted on the spherical parameter space. The thick lines depict the edges of the original square faces. The equidistance for both θ and ϕ is $\frac{\pi}{8}$. *(b)* ϕ and $\cos\theta$ are interpreted as Cartesian coordinates. The monotonic cosine function is applied to give a true-area cylindrical projection. The horizontal lines at ± 1 are the poles. *(c)* Conversely, the globe coordinate grid is drawn over the object. For comparison, one vertex is marked with a black dot in all diagrams.

3.2 Optimization method

Powerful methods for nonlinear constrained minimization exist[15]. The commonly available optimization routines can not be used for larger objects because they are not suited for such a large problem, as they do not exploit its sparsity and information available about the constraints. A Newton-Lagrange algorithm [15] based on the sparse linear solver package PILS [13] is used to find the constrained maximum of the goal function. An active set strategy enforces the inequality constraints. A detailed discussion of the optimization algorithm would exceed the scope of this paper. In fact, any other constrained optimizer that could handle the size of the problem could be substituted.

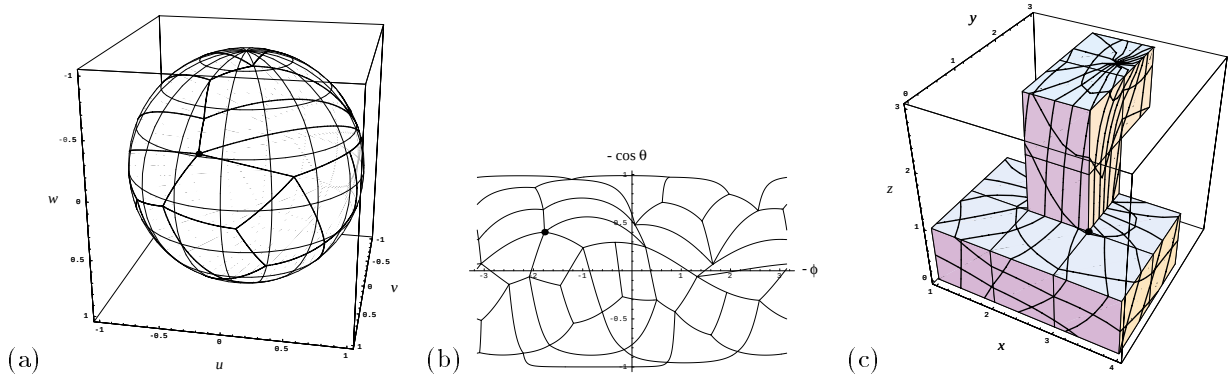


Figure 5: The result of the optimization, plotted in the same ways as in the previous figure. The areas of elementary facets in parameter space are now equal, and local distortions are minimized. The rotational position of the net on the sphere is arbitrary. *(c)* After the optimization, the former poles have lost their prominent role. They have now the same importance as any other point in parameter space and could lie anywhere on the surface. Only the use of polar coordinates for visualization gives them a conspicuous appearance.

The question arises if a solution exists at all and if it is unique. The goal function is a sum of cosines, which cannot exceed one. Thus the goal function is bounded by the number of edges in the graph, $2n_faces$. Any constraints can only lower the attainable maximum, but never make the goal function unbounded. Another problem could arise from incompatible constraints. From their definition - vector length, area content, angle - they are not inherently incompatible. There are $3\,n_{vert}$ degrees of freedom but only $n_{vert} + n_{face} - 1 = 2n_{vert} - 3$ equations to satisfy. At the solution, only few (typically less than five) inequalities are active. We have no indications that there might be any local maxima. Of course any rotated version of the solution yields the same values for the goal function and for all constraints, but this is not considered an essentially different solution.

Constraining all angles to the range $[0, \pi]$ makes sure none of the quadrilaterals can fold over. Thus the neighborhood relations are preserved during the optimization process.

As the goal function, the constraints, and their derivatives have to be calculated repeatedly, it is important that we can use an efficient data structure that holds information about adjacency (for the goal function and matrix sparsity) and surface facets (for the area constraints).

The solution of the nonlinear program defines the optimal parametrization of our object's surface. Figure 4 shows the starting point for the optimization; the right diagram is a combination of those in Figure 3. Figure 5 visualizes the result of the optimization in different ways. The same vertex as in Figure 4 is marked. Figure 6 shows several stages of the optimization process. The starting configuration, "0", corresponds to Fig. 4a. The final result is obtained after 64 iterations; it has the label "64" and is the same as Fig. 5a.

4 Spherical harmonic shape descriptors

Let x, y and z denote Cartesian object space coordinates and θ and ϕ polar parameter space coordinates. The parametrization gives us three explicit functions defining the object surface as follows:

$$\mathbf{x}(\theta, \phi) = \begin{pmatrix} x(\theta, \phi) \\ y(\theta, \phi) \\ z(\theta, \phi) \end{pmatrix}$$

When the free variables θ and ϕ run over the whole sphere (e.g. $\theta = 0 \ldots \pi, \phi = 0 \ldots 2\pi$), $\mathbf{x}(\theta, \phi)$ runs over the whole surface of our object. As our shape description method, we use the surface net representation to expand a 3-D shape into a complete set of basis functions. We use the spherical harmonics; Y_l^m denotes

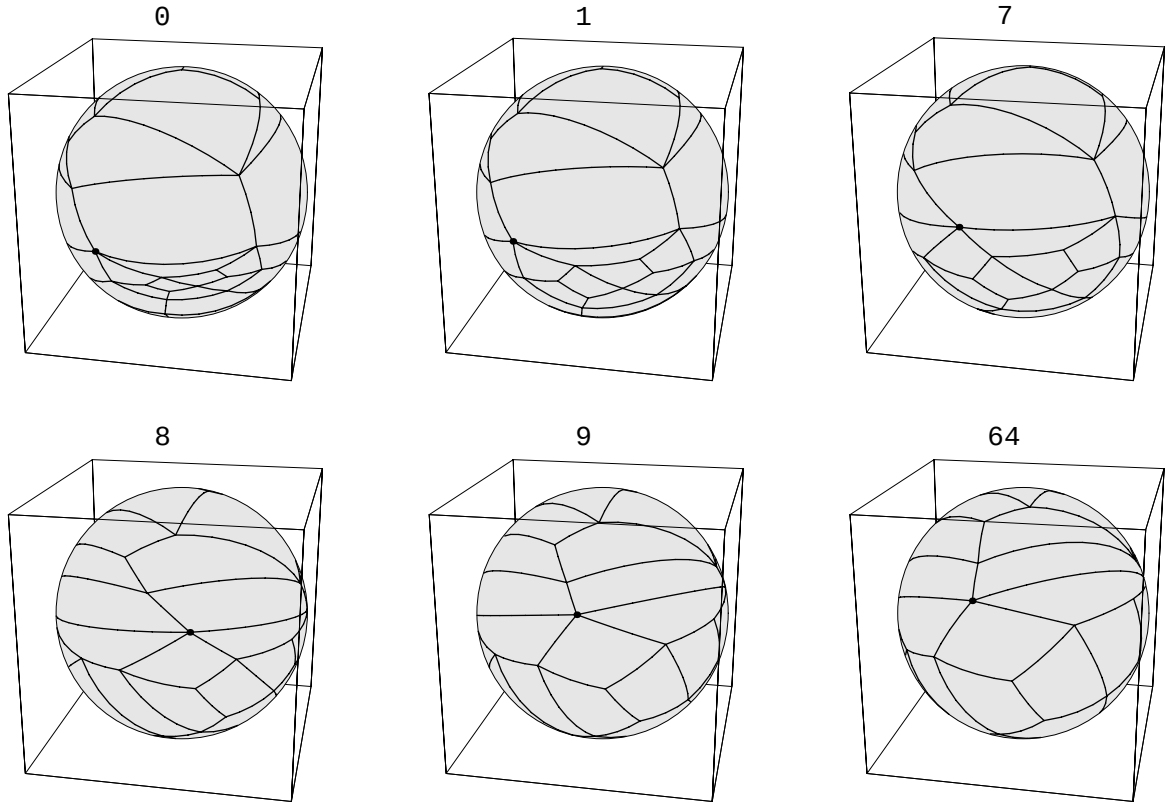


Figure 6: Stages from an optimization run. The labels indicate the number of iterations. "0" marks the starting configuration, where the areas of the facets vary considerably. "64" is the final result; all facets have now equal area, and distortions are minimized.

the function of degree l and order m [16]. Definitions are given in the appendix. The coefficients

$$\mathbf{c}_l^m = \begin{pmatrix} {c_x}_l^m \\ {c_y}_l^m \\ {c_z}_l^m \end{pmatrix}$$

in this series are three-dimensional vectors. Their components, ${c_x}_l^m$, ${c_y}_l^m$ and ${c_z}_l^m$, are usually complex numbers; they are real numbers for $m = 0$. The series takes the following form.

$$\mathbf{x}(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \mathbf{c}_l^m \; Y_l^m(\theta, \phi) \tag{1}$$

The coefficients of the spherical harmonic functions of different degrees provide a measure of the spatial frequency constituents that comprise the structure. Partial sums of the series (1) are plotted in Figure 7. The sums are truncated by limiting l to $0 \leq l < n_l$, where $n_l = 2, 4, 8$. As higher frequency components are included, more detailed features of the object appear.

The use of orthogonal basis functions is convenient for the calculation of the expansion coefficients. Formally, the coefficients are calculated by forming the inner product of $\mathbf{x}$ with the basis function in question:

$$\mathbf{c}_l^m \quad = \quad \langle \mathbf{x}(\theta, \phi), Y_l^m(\theta, \phi) \rangle = \int_0^{\pi} \int_0^{2\pi} \mathbf{x}(\theta, \phi) \, Y_l^m(\theta, \phi) \, d\phi \sin\theta d\theta \tag{2}$$

Note that the parametrization defines the function $\mathbf{x}(\theta, \phi)$ only for the parameter coordinates of the vertices. Let n_{vert} denote the number of vertices and i be the index of a vertex, $0 \leq i < n_{vert}$. Only $\mathbf{x}(\theta_i, \phi_i) = \mathbf{x}_i$

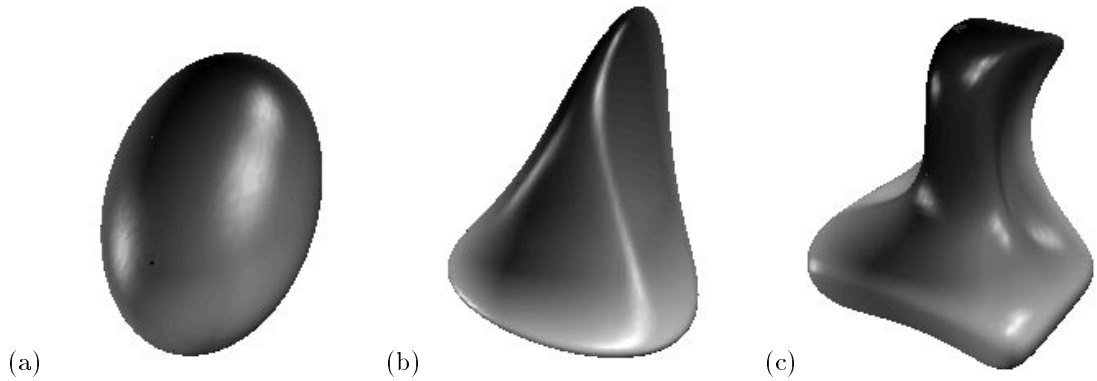


Figure 7: Global shape description by expansion into spherical harmonics: The figures illustrate the reconstruction of the partial spherical harmonic series, using coefficients up to degree 1 (a), to degree 3 (b) and 7 (c).

is defined. For the evaluation of the integral (2) we would have to define an interpolating function between these sample points; an adaptation of bilinear interpolation could be used for this purpose. But this would introduce an artificial sub-voxel resolution that has no base in the input data. On the other hand, the straightforward discretization of the integral,

$$\mathbf{c}_l^m \approx \frac{4\pi}{n_{vert}} \sum_{i=0}^{n_{vert}-1} \mathbf{x}_i \, Y_l^m(\theta_i, \phi_i) \tag{3}$$

does not give the precise coefficients of a series representing our object. The reason is that although the functions Y_l^m are orthonormal, their values evaluated at some set of parameter pairs (θ_i, ϕ_i) will generally not form an orthonormal set of vectors. We adopt some indexing scheme $j(l,m)$ that assigns a unique index j to every pair l, m, like e.g. $j(l,m) := l^2 + l + m$. When the degree of the spherical harmonics is limited, i.e. $0 \leq l < n_l$, j is also limited by $j < n_j = n_l^2$. We can arrange all needed values of our basis functions in a $n_{vert} \times n_j$ matrix B where $b_{i,j(l,m)} = Y_l^m(\theta_i, \phi_i)$. In the usual case where n_j is significantly smaller than n_{vert}, the columns of B are approximately orthogonal. We further arrange the object space coordinates of all vertices in an $n_{vert} \times 3$ matrix $X = (\mathbf{x}_0, \mathbf{x}_1, \ldots \mathbf{x}_{n_{vert}-1})^T$ and all coefficients in the $n_j \times 3$ matrix $C = (\mathbf{c}_0^0, \mathbf{c}_1^{-1}, \mathbf{c}_1^0 \ldots)^T$. The equations (3) for all l and m take the compact form $C \approx \frac{4\pi}{n_{vert}} B^T X$. But what we really want is a spherical harmonic series that passes near the real positions of our vertices, i.e. $X = BC + E$ where the error matrix E should be small. These so-called normal equations are solved with least square sums over the columns of E by

$$C = (B^T B)^{-1} B^T X \ . \tag{4}$$

The global approximation error is the square of the Frobenius norm of $E = BC - X$, which is also minimized. This is not too different from (3) because the symmetric $n_j \times n_j$ matrix $\frac{4\pi}{n_{vert}} B^T B$ is close to the identity matrix.

4.1 Rotation independent descriptors

The coefficients obtained so far still depend on the relative position of the parameter net of the object surface and on the orientation of the object in space (Figure 5). We can get rid of these dependencies by rotating the object to canonical positions in parameter space and object space. This needs three rotations in parameter space and three rotations in object space, when rotations are described by Euler angles. All rotations result in new linear combinations of the components of the harmonic descriptors.

The relations between the Cartesian and the spherical coordinates of the parameter space are $u = \sin\theta \cos\phi$, $v = \sin\theta \sin\phi$ and $w = \cos\theta$. To define a standard position we consider only the contribution

of the spherical harmonics of degree $l = 1$ in equation (1), an ellipsoid (see Figure 7).

$$\mathbf{x}_1(\theta,\phi) = \sum_{m=-1}^{1} \mathbf{c}_1^m \, Y_1^m(\theta,\phi) \tag{5}$$

This sum involves the basis functions $Y_1^{-1} = \frac{\sqrt{3}}{2\sqrt{2\pi}}(u - iv)$, $Y_1^0 = \frac{\sqrt{3}}{2\sqrt{\pi}} w$ and $Y_1^1 = -\frac{\sqrt{3}}{2\sqrt{2\pi}}(u + iv)$. Any three real valued linear combinations of these (i. e. $({c_x}_1^{-1})^* = -{c_x}_1^1$ and ${c_x}_1^0 \in \Re$ and similarly for y and z), interpreted as Cartesian coordinates in the object space, will always describe an ellipsoid. We rotate the parameter space so that the north pole ($\theta = 0$, on the w axis) will be at one end of the shortest main axis of this first order ellipsoid and the point where the Greenwich meridian ($\phi = 0$) crosses the equator ($\theta = \frac{\pi}{2}$, on the u axis) is at one end of the longest main axis.

At the three main axes, the length of the vector $\mathbf{x}_1(\theta,\phi)$ is stationary: it reaches a maximum, a saddle point, and a minimum, respectively. Measuring Euclidean lengths becomes simpler when we transform the component vectors to a Euclidean, real valued form. A short calculation yields

$$\mathbf{x}_1(\mathbf{u}) \;=\; A\mathbf{u} \;=\; A \begin{pmatrix} u \\ v \\ w \end{pmatrix} \;=\; \mathbf{a}_1\, u + \mathbf{a}_2\, v + \mathbf{a}_3\, w \quad , \tag{6}$$

where

$$A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \;=\; \frac{\sqrt{3}}{2\sqrt{2\pi}} \left(\mathbf{c}_1^{-1} - \mathbf{c}_1^1, i(\mathbf{c}_1^{-1} + \mathbf{c}_1^1), \sqrt{2}c_1^0 \right) \quad . \tag{7}$$

We are looking for the unit vectors $\hat{\mathbf{u}}_1$, $\hat{\mathbf{u}}_2$ and $\hat{\mathbf{u}}_3$ that maximize or minimize the length of the vector. The solutions are the eigenvectors of $A^T A$, with nonnegative eigenvalues $l_1^2 > l_2^2 > l_3^2$. Their roots l_1, l_2 and l_3 represent half the lengths of the main axes of the ellipsoid. At the middle eigenvector, $\hat{\mathbf{u}}_2$, $\|\mathbf{x}_1\|$ has a saddle-point rather than an extremum. The normalized eigenvectors form the rotation matrix $R_{uvw}^T = (\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3)$, which is applied to the parameters $\mathbf{u}_i$ associated with each vertex i: $\mathbf{u}_i' = R_{uvw}^T \mathbf{u_i}$. This new parametrization, results in new coefficients $\mathbf{c}_l^{m\prime}$ and hence in the new coefficient matrix $A' = A\, R_{uvw}$. Its three column vectors $\mathbf{a}'_1$, $\mathbf{a}'_2$ and $\mathbf{a}'_3$ are the main axes of the first order ellipsoid in object space.

All rotations are determined based on the values of $\mathbf{c}_1^i$ ($i = -1 \ldots 1$) of the ellipsoid only, but they are applied to all components of the descriptor $\{\mathbf{c}_l^m\}$. The parameter space rotations result in a different description of the same object in the same position, just parametrized in a standard way.

Now, the ellipsoid is rotated in the object space to make its main axes coincide with the coordinate axes, putting the longest ellipsoid axis along x and the shortest one along z. The object space rotations require only the matrix multiplication $\mathbf{c}_l^{m\prime\prime} = R_{xyz}\, \mathbf{c}_l^{m\prime}$. The object space rotation matrix is $R_{xyz} = \text{diagonal}(\frac{1}{l_1}, \frac{1}{l_2}, \frac{1}{l_3}) \cdot {A'}^T$. It rotates the main axes of the ellipsoid into an axis-parallel position and makes the coefficient matrix $A'' = R_{xyz}\, A' = R_{xyz}\, A\, R_{uvw}$ diagonal. The elements of the diagonal are the lengths of the main axes of the ellipsoid.

The descriptors $\mathbf{c}''$ are now invariant under rotation of the object, except mirrorings (rotations by π). Including information from higher degree coefficients could eventually disambiguate these cases. Ignoring $\mathbf{c}_0^0$ results in translation invariance. Scaling invariance can be achieved by dividing all descriptors by l_1, the length of the longest main axis.

4.2 Importance of uniform parametrization

The report so far presumed that a homogeneous density and a minimal distortion of the parameter net would be important for shape characterization, especially for obtaining an invariant description. Similarly, the 2-D expansion of contours $s(t)$ into series of harmonics [5] was based on the model of tracing a curve with constant velocity, i.e. assigning same lengths ΔL to equivalent parameter steps Δt. A non-uniform distribution of parameters on an object surface, e.g. by clustering at certain locations, seems to be suboptimal with respect to a uniform representation of the whole surface. One would expect an over-representation of some parts to the disadvantage of others, resulting in distorted shape descriptions.

The importance of a parametrization with minimal distortion can be demonstrated with an experiment. The expansion into a series of spherical harmonics is calculated for both the non-uniform initial parametrization (bypassing the optimization step for this part of the experiment) and the result after optimization.

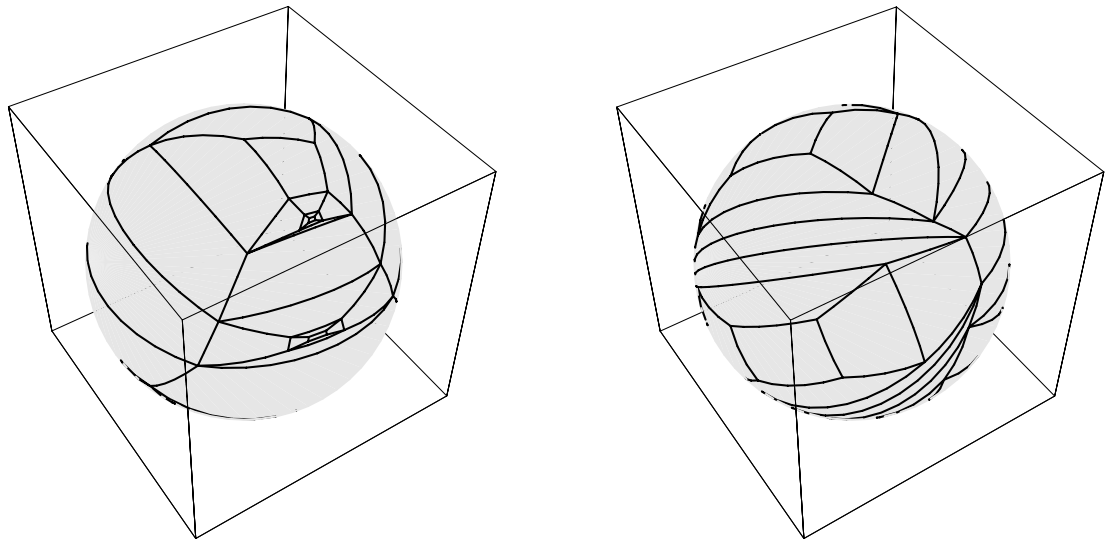


Figure 8: Two different parametrizations of the "E" object. *Left:* The initial parametrization is the starting point of the optimization. *Right:* The optimized parametrization.

A manifestly non-star-shaped form was chosen: the original object consists of 11 voxels and is shaped like the character E. Its initial and optimized parametrizations are given in Figure 8 for comparison. The diagrams correspond to Figures 4a and 5a.

Figure 9 illustrates the expansion in a spherical harmonic series up to degree ten and the truncated reconstruction up to degree one (top), four (middle) and ten (bottom) for the initial (left) and optimized (right) parametrization. (A five-fold oversampling was applied to the surface to represent it accurately. This may be viewed as a rough form of numerical integration.) Comparing the expressive difference, one can conclude that a uniform parametrization is absolutely essential to obtain useful spherical harmonic descriptors. Even from the distorted initial parametrization, descriptors can be derived that are "optimal" in the least squares sense, but the series of harmonics does not reflect the shape properties of the surface. Using the optimized parametrization, the reconstructed objects demonstrate the most desirable behavior that coefficients of higher degree add information about details of higher spatial frequency. The first degree harmonic approximation (Figure 9b) covers the whole object and comprises information about the major size and elongation, whereas the three "legs" of the E-shape appear in the reconstruction using harmonics up to degree four (Figure 9d,f).

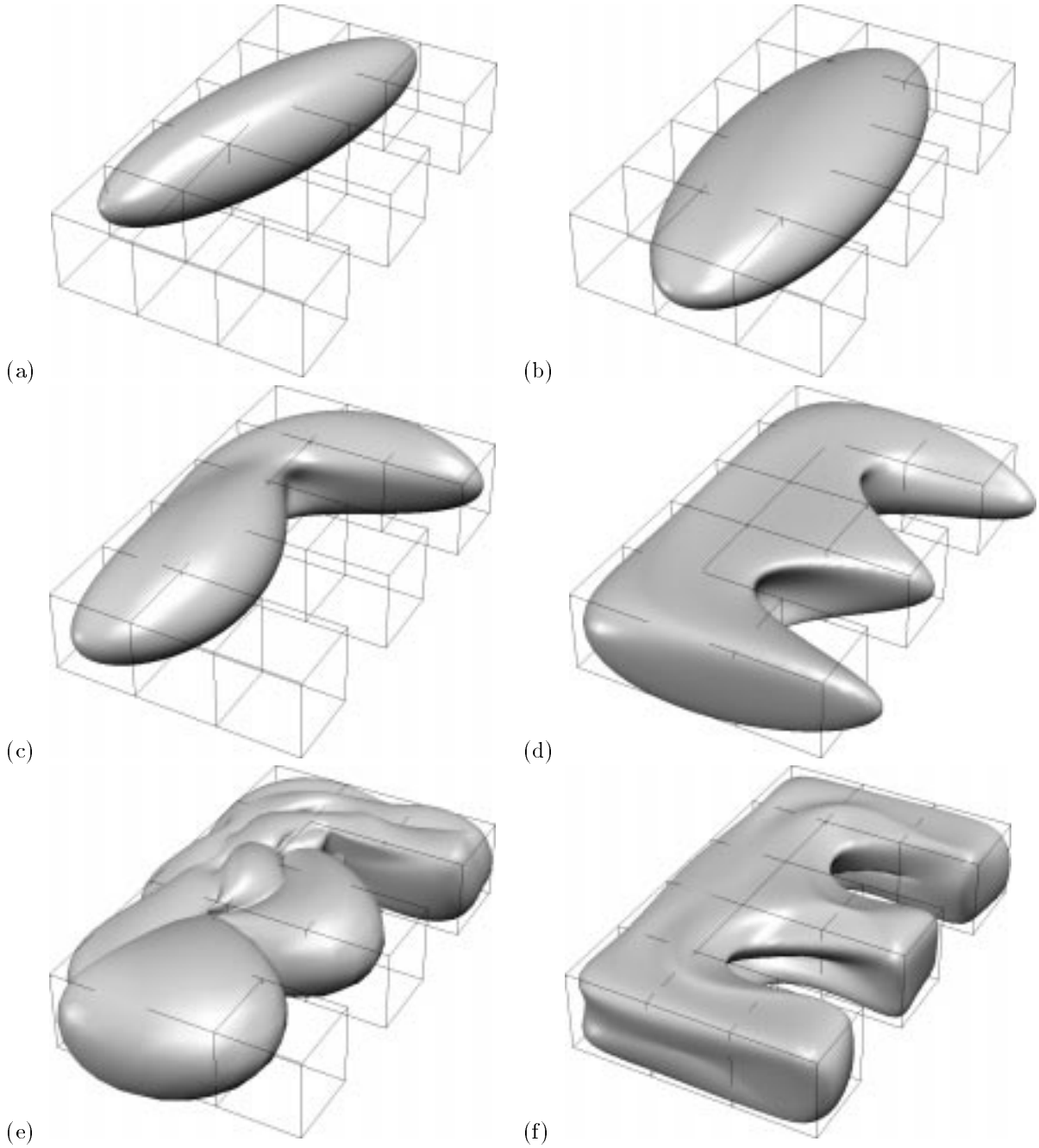


Figure 9: Experiment demonstrating the importance of a homogeneous parameter distribution for shape description. The original E-shaped object surface, indicated by a wireframe, is expanded into a series of spherical harmonics once starting from a non-uniform parametrization and second from an optimized parametrization. The series are truncated at degree one (a,b), four (c,d), and ten (e,f). Shaded surfaces depict the reconstructions. The images in the left column (a,c,e) clearly illustrate the poor shape representation based on non-uniform parametrization. A significant improvement is achieved by using the optimized parametrization (b,d,f), reflecting the hierarchical nature of a harmonic approximation regarding shape details at different scales. Quantitatively, $\|E\|_F$ measures the error, which is divided by the square root of the number of rows of E to give the RMS distance in pixel units. The values for the individual reconstructions are 1.143 (a), 0.884 (b), 0.729 (c), 0.250 (d), 0.313 (e), and 0.102 (f).

4.3 Experimental results

The surfaces of the following test objects are parametrized, and their shapes are expressed through spherical harmonic descriptors.

- *c4* A "c"-shaped polyhedron made up from five 4x4x4 voxel cubes.
- *c8* The same object, magnified by a factor of two in all coordinate directions.
- *box A* The voxels fill a rectangular box, which is not aligned with the coordinate axes.
- *box B* Rotating the box A results in a completely different sampling.
- *patella* The patella was extracted from a segmented CT scan data set of a human knee.
- *ventricle* The ventricular system has been segmented from a MRI data set of a hydrocephalus patient[1]. We selected one lateral ventricle. The data has been interpolated to compensate for the aspect ratio of 1:1:6.4 of the original data.

For each object, Figure 10 presents the cuberille interpretation of the input data, a spread-out graph of the parametrization, and the reconstruction from spherical harmonic descriptors. The cylindrical projection we chose for drawing the parametrization shows the true area ratios. The smooth surface of the reconstruction is shaded in a pattern that allows the estimation of the parameter values. These latter parameter values do not coincide with the ones in the middle diagram. They rather differ by the rotation in parameter space that makes the descriptors rotation invariant. The object space rotation is suppressed in the diagrams to show the spatial relation of the original data – shown as a wireframe – with the reconstruction from the descriptor, up to degree 8. In the case of the ventricle, this shows an insufficient degree of detail, but the same value was chosen for comparability.

Table 1 summarizes the sizes and differences of the various test objects. Virtually all of the processing time for an object is spent in the optimization. The figure for computation time must be interpreted with caution; it qualifies only the optimization program, which is not necessarily as efficient as possible, and which might be substituted with an out-of-the box optimizer. Times are measured on a HP 9000/735. The number of vertices, n_{vert} indicates the size of the problem: the optimization has $3\,n_{vert}$ variables, $2\,n_{vert} - 3$ equality constraints and $4\,n_{vert} - 8$ inequality constraints.

The distance between the descriptors appears to be a valid rough measure of shape dissimilarity. The matrix of distances is symmetric by definition. The two "c"s are most similar to each other. The two boxes are also quite similar. Both these examples illustrate the translation, rotation and scale invariance of the descriptors. The patella is more similar to a box than to a "c", whereas the ventricle is more similar to a "c" than to any of the other objects.

			distance to					
name	n_{vert}	time	box A	box B	c4	c8	patella	ventricle
box A	628	33 s	0	0.0241	0.2370	0.2378	0.0673	0.3850
box B	902	267 s	0.0241	0	0.2796	0.2808	0.0859	0.4555
c4	354	26 s	0.2370	0.2796	0	0.0002	0.2309	0.2175
c8	1410	338 s	0.2378	0.2808	0.0002	0	0.2299	0.2143
patella	2182	536 s	0.0673	0.0859	0.2309	0.2299	0	0.2623
ventricle	37654	28 h	0.3850	0.4555	0.2175	0.2143	0.2623	0

Table 1: Comparison of the six test objects, including the squared Euclidean distances between their descriptors.

[1] Data courtesy Ron Kikinis, M. D., Surgical Planning Lab, Department of Radiology, Brigham and Women's hospital and Harvard Medical School, Boston

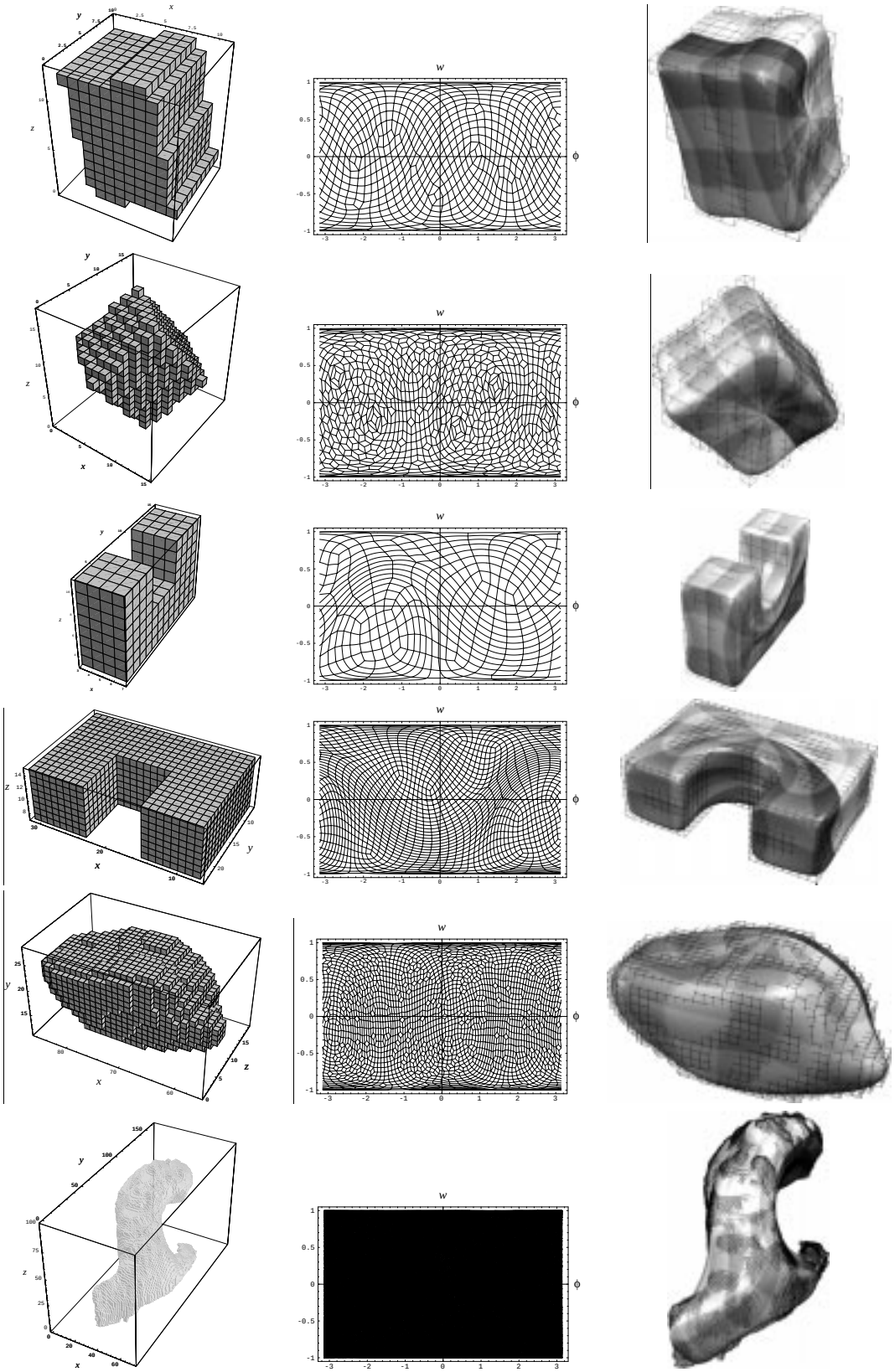


Figure 10: Test objects and results. From Top to bottom: objects "box A", "box B", "c4", "c8", "patella", and "ventricle". From left to right in each row: The cuberille interpretation of the input data, the parametrization drawn as a flat net spread out in $(\phi, w = \cos\theta)$ parameter space, and the reconstruction superimposed with a wireframe of the original data.

5 Conclusions

This paper presents new techniques to generate explicit parametric representations of *convoluted* object surfaces with minimal distortion and to characterize 3-D surfaces by *invariant* spherical harmonic shape descriptors. We intended to overcome traditional limitations of expressing an object surface by explicit parametric representations, which are the restriction to star-shaped objects, the non-uniform spacing of parameters on the object surface, and often a specific choice of the new coordinate system with respect to the object geometry. The parametrization technique is described for closed surfaces of simply connected objects, but it generalizes naturally to the unfolding or flattening of complex surface patches (open surfaces with one edge). Other classes of simple 3-D surfaces with different topology, for example tori and tubes, are not considered.

The following paragraphs summarize the properties of the novel surface analysis techniques.

Parametrization: The unfolding and optimization procedure maps nodes onto a sphere, each of which can be expressed by two parameters and is associated with a voxel vertex. The procedure imposes no restrictions regarding the geometry of objects and is suitable for surfaces of arbitrary complexity. An initial diffusion of "temperature" on the object's surface achieves a continuous mapping by assigning latitude and longitude to each surface vertex. The position of poles and the geometry of the object produce a clustering of parameters at certain regions of the object surface. This non-uniform distribution of parameter density on the object surface is corrected by a nonlinear optimization technique, which preserves the area of original surface elements in parameter space and minimizes their distortion. The latter is formulated as the goal function of the nonlinear optimization problem and the former as its constraints. The resulting arrangement of the vertex nodes on the sphere (parameter space) reflects the geometry of the original shape and achieves similar parametrizations for similar shapes, but is free to rotate around any axis as no surface points are kept fixed. It must be pointed out that the polar coordinate system for spherical surfaces is only used for the sake of visualization. This coordinate system itself determines a non-uniform tessellation of the sphere and may give a misleading visual impression of parameter densities (see e.g. Figure 5). Alternative display techniques could be found by tessellating the sphere into small cells, for example by regular or semiregular polyhedra.

The parametrization of object surfaces forms an intermediate representation with the following properties:

- The surface of arbitrarily shaped (but simply connected) objects can be parametrized. Objects are not restricted to a limited family of shapes; even protrusions and intrusions are appropriately dealt with.
- The surface is explicitly represented by the variation of two parameters, expressing the properties of local surface neighborhoods as well as of the global shape.
- The parametrization results in a continuous (no overlap of elements), one-to-one mapping of surface vertices to a sphere. While varying the two parameters θ and ϕ over the parameter range, each point of the surface is visited exactly once.
- The optimization results in a unique, reproducible solution (except for rotation).
- The parametrization preserves areas exactly and minimizes local distortions which cannot be avoided when mapping an object with corners to a sphere (see location marked by a black dot in Figures 5 a, b and c). The uniformity of the parametrization is important for a subsequent shape description, as illustrated in Figure 9.

The parametrization technique is potentially interesting for applications where a mapping of convoluted object surfaces to a simple surface – like the sphere – is required. The unfolding, or flattening, process with minimization of distortions generates a representation which could serve as a useful intermediate surface description for many structure analysis processes. The only restriction, i.e. the presence of the closed surface of a simply connected object, highlights the generality of the approach.

As discussed previously, the unfolding is in principle not restricted to closed surfaces. Ongoing developments focus on a similar technique for flattening parts of surfaces onto planar charts. This procedure could be interesting for the comparative analysis and description of convoluted surface patches. Practical

applications can be found in brain research, for example, where regional cortical patterns of the human brain are qualitatively and quantitatively analyzed.

Applications are still constrained by the efficiency of the nonlinear optimization. Although the method itself poses no restriction on the maximum number of object vertices, the commonly available optimization routines cannot be applied for a large number of vertices (exceeding several hundreds). In real applications, e.g. the analysis of volume data in medicine, one can expect to deal with object surfaces with up to one million voxel vertices. We have recently developed an optimization technique which takes into account the sparsity of the problem and the specific nature of the local constraints.

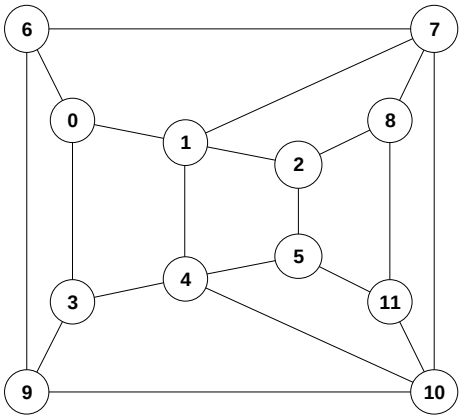
Shape description: The new parametrization allows representation of object surfaces of arbitrary complexity. As one possible approach to global shape analysis, it enables us to expand an object surface into a series of spherical harmonic functions. The numerical coefficients in the Fourier series represent an object-centered, surface-oriented descriptor of the object's form. Surface description with harmonic descriptors is no longer restricted to star-shaped objects but can now be applied to a broad class of shapes. Invariance has to be considered as one of the most important properties of shape description, as it only allows a comparative analysis between different objects or a match between objects and models. With the development of new scale and rotation independent descriptors we obtain a *global, object-centered shape description* which is invariant to standard transformations (rotation, translation and scaling). The invariant positioning of the object and of the parameter net are based on the analysis of harmonic descriptors up to the first degree, defining the three main axes of the ellipsoid. The symmetry of this low frequency representation determines a general 3-D object only up to four different positions. Including coefficients of higher degree, for example at the extremal points of the description up to degree one, could disambiguate the different cases and avoid a matching using four different object descriptions.

Applications of global object representation and description in computer vision and image analysis are imminent. Generality with respect to object complexity, invariance to standard transformations, and descriptive power in terms of object geometry are the critical issues for shape-based categorization and comparison of 3-D objects. Robot vision and medical image analysis, e.g. are dealing with recovering the global shape characteristics of objects. Whereas the former most often deals with a small number of views of objects and hence only a partial surface description, modern scanning techniques in medicine can provide full 3-D images. Mapping of convoluted surface structures and high level 3-D shape descriptions of anatomical objects (e.g. the heart cavities, the ventricular system or cortical substructures of the brain) will play a significant role in the analysis of shape dissimilarities, morphological deformations and in the comparison of malformed with "normal" shape structures. The overall shape is captured by a small number of parameters, expressing structural details at various scales with coefficients of different degrees. The continuous analytical description of the approximated surface permits to compute local differential characteristics, e.g. principal curvature [17]. Inferring the differential structure would result in a characterization of important landmarks, e.g., for registration of different 3-D objects.

Appendix A: An example of the initial parametrization

The "two-voxel" object (cf. Figure 1) is used throughout the example. The surface data structure of this object and a flat diagram of the surface net are given below for reference.

node number		x	y	z	neighbors
0	{{{	9,	14,	6},	{1,7,6,9,3,4}},
1	{{	10,	14,	6},	{0,3,4,5,2,8,7,6}},
2	{{	11,	14,	6},	{1,4,5,11,8,7}},
3	{{	9,	15,	6},	{4,1,0,6,9,10}},
4	{{	10,	15,	6},	{3,9,10,11,5,2,1,0}},
5	{{	11,	15,	6},	{4,10,11,8,2,1}},
6	{{	9,	14,	7},	{7,10,9,3,0,1}},
7	{{	10,	14,	7},	{6,0,1,2,8,11,10,9}},
8	{{	11,	14,	7},	{7,1,2,5,11,10}},
9	{{	9,	15,	7},	{10,4,3,0,6,7}},
10	{{	10,	15,	7},	{9,6,7,8,11,5,4,3}},
11	{{	11,	15,	7},	{10,7,8,2,5,4}}}



Determining latitude

The border conditions

$$\begin{aligned} \theta_0 &\equiv \theta_{north} = 0 \\ \theta_{11} &\equiv \theta_{south} = \pi \end{aligned}$$

and the average properties

$$\begin{aligned} 4\theta_1 &= \theta_0 + \theta_4 + \theta_2 + \theta_7 \\ 3\theta_2 &= \theta_1 + \theta_5 + \theta_8 \\ 3\theta_3 &= \theta_4 + \theta_0 + \theta_9 \\ 4\theta_4 &= \theta_3 + \theta_{10} + \theta_5 + \theta_1 \\ &\vdots \\ 4\theta_{10} &= \theta_9 + \theta_7 + \theta_{11} + \theta_4 \end{aligned}$$

can be arranged in matrix notation.

$$\begin{pmatrix} 1 & 0 & & 0 & & & 0 & & & & & \\ -1 & 4 & -1 & & -1 & & & -1 & & & & \\ & -1 & 3 & & & -1 & & & -1 & & & \\ -1 & & & 3 & -1 & & & & & -1 & & \\ & -1 & & -1 & 4 & -1 & & & & & -1 & \\ & & -1 & & -1 & 3 & & & & & & -1 \\ -1 & & & & & & 3 & -1 & & -1 & & \\ & -1 & & & & & -1 & 4 & -1 & & -1 & \\ & & -1 & & & & & -1 & 3 & & & -1 \\ & & & -1 & & & -1 & & & 3 & -1 & \\ & & & & -1 & & & -1 & & -1 & 4 & -1 \\ & & & & & 0 & & & 0 & & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \\ \theta_9 \\ \theta_{10} \\ \theta_{11} \end{pmatrix} = \begin{pmatrix} \theta_{north} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \theta_{south} \end{pmatrix} \tag{8}$$

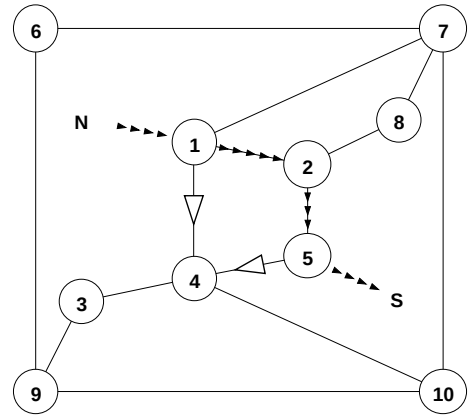
Applying the first and the last row (boundary conditions) leads to the following reduced, symmetric system.

$$\begin{pmatrix} 4 & -1 & & -1 & & & -1 & & & \\ -1 & 3 & & & -1 & & & -1 & & \\ & & 3 & -1 & & & & & -1 & \\ -1 & & -1 & 4 & -1 & & & & & -1 \\ & -1 & & -1 & 3 & & & & & \\ & & & & & 3 & -1 & & -1 & \\ -1 & & & & & -1 & 4 & -1 & & -1 \\ & -1 & & & & & -1 & 3 & & \\ & & -1 & & & -1 & & & 3 & -1 \\ & & & -1 & & & -1 & & -1 & 4 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \\ \theta_9 \\ \theta_{10} \end{pmatrix} = \begin{pmatrix} \theta_{north} \\ 0 \\ \theta_{north} \\ 0 \\ \theta_{south} \\ \theta_{north} \\ 0 \\ \theta_{south} \\ 0 \\ \theta_{south} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \pi \\ 0 \\ 0 \\ \pi \\ 0 \\ \pi \end{pmatrix} \tag{9}$$

The solution is $\boldsymbol{\theta} = (\frac{2\pi}{5}, \frac{3\pi}{5}, \frac{3\pi}{10}, \frac{\pi}{2}, \frac{7\pi}{10}, \frac{3\pi}{10}, \frac{\pi}{2}, \frac{7\pi}{10}, \frac{2\pi}{5}, \frac{3\pi}{5})^T$, or $\boldsymbol{\theta}' = (0, \frac{2\pi}{5}, \frac{3\pi}{5}, \frac{3\pi}{10}, \frac{\pi}{2}, \frac{7\pi}{10}, \frac{3\pi}{10}, \frac{\pi}{2}, \frac{7\pi}{10}, \frac{2\pi}{5}, \frac{3\pi}{5}, \pi)^T$, where the first and last value correspond to the poles.

Determining longitude

The path 0, 1, 2, 5, 11 is used as the date line; it is indicated by a row of small black triangles. Links extending from the date line to the west are $1 \to 4$ and $5 \to 4$; they are marked with white triangles. The poles, "N" (vertex 0) and "S" (vertex 11), are no longer part of the net. The following equations result.



$$
\begin{aligned}
3\phi_1 &= (\phi_4 - 2\pi) + \phi_2 + \phi_7 \\
3\phi_2 &= \phi_1 + \phi_5 + \phi_8 \\
2\phi_3 &= \phi_4 + \phi_9 \\
4\phi_4 &= \phi_3 + \phi_{10} + (\phi_5 + 2\pi) + (\phi_1 + 2\pi) \\
&\vdots \\
3\phi_{10} &= \phi_9 + \phi_7 + \phi_4
\end{aligned}
$$

These equations, together with $2\phi_1 = 0$, can be put into matrix notation as follows.

$$
\begin{pmatrix}
5 & -1 & & -1 & & & -1 & & & \\
-1 & 3 & & & -1 & & & -1 & & \\
 & & 2 & -1 & & & & & -1 & \\
-1 & & -1 & 4 & -1 & & & & & -1 \\
 & -1 & & -1 & 2 & & & & & \\
 & & & & & 2 & -1 & & -1 & \\
-1 & & & & & -1 & 4 & -1 & & -1 \\
 & -1 & & & & & -1 & 2 & & \\
 & & -1 & & & -1 & & & 3 & -1 \\
 & & & -1 & & & -1 & & -1 & 3
\end{pmatrix}
\begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10} \end{pmatrix}
=
\begin{pmatrix} -2\pi \\ 0 \\ 0 \\ 4\pi \\ -2\pi \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\tag{10}
$$

The solution is $\boldsymbol{\phi} = (0, 0, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{-\pi}{4}, \frac{3\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4}, \pi, \pi)^T$

Appendix B: Spherical harmonic functions

The following definitions are used in this paper [12].
Legendre polynomials

$$
P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} (x^2 - 1)^l \tag{11}
$$

Associated Legendre polynomials

$$
P_l^m(x) = (-1)^m (1 - x^2)^{\frac{m}{2}} \frac{d^m}{dx^m} P_l(x) = \frac{(-1)^m}{2^l l!} (1 - x^2)^{\frac{m}{2}} \frac{d^{m+l}}{dx^{m+l}} (x^2 - 1)^l \tag{12}
$$

Spherical harmonic functions

$$
Y_l^m(\theta, \phi) = \sqrt{\frac{2l+1}{4} \frac{(l-m)!}{(l+m)!}} \; P_l^m(\cos\theta) \; e^{im\phi} \tag{13}
$$

$$
Y_l^{-m}(\theta, \phi) = (-1)^m Y_l^{m*}(\theta, \phi) \tag{14}
$$

A spherical harmonic function of degree l can be written as a homogeneous polynomial of degree l in $u = \sin\theta\,\cos\phi$, $v = \sin\theta\,\sin\phi$ and $w = \cos\theta$.

References

[1] D. H. Ballard and Ch. M. Brown, *Computer Vision*, Prentice-Hall Inc, Englewood Cliffs, New Jersey, 1981.

[2] F. Solina and R. Bajcsy, Recovery of parametric models from range images: The case for superquadrics with global deformations, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**(2):131–147, February 1990.

[3] Lawrence H. Staib and James S. Duncan, Deformable fourier models for surface finding in 3d images, In Richard A. Robb, editor, *Visualization in biomedical computing 1992*, volume Proc. SPIE 1808, pp. 90–104, 1992.

[4] E. Persoon and K.S. Fu, Shape discrimination using fourier descriptors, *IEEE Trans. Systems, Man and Cybernetics SMC*, **7**(3):388–397, May 1977.

[5] F. P. Kuhl and Ch. R. Giardina, Elliptic fourier features of a closed contour, *Computer Graphics and Image Processing*, **18**(3):236–258, March 1982.

[6] D. N. Kennedy, J. Sacks, P. A. Filipek, and V. S. Caviness, Three-dimensional fourier shape analysis in magnetic resonance imaging, In Peder C. Pedersen and Banu Onaral, editors, *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 12/1, pp. 78–79, Philadelphia, PA, USA, October 1990.

[7] Christian Brechbühler, Guido Gerig, and Olaf Kübler, Surface parametrization and shape description, In Richard A. Robb, editor, *Visualization in biomedical computing 1992*, volume Proc. SPIE 1808, pp. 80–89, 1992.

[8] Gabor T. Herman and Hsun Kao Liu, Three-dimensional display of human organs from computer tomograms, *Computer Graphics and Image Processing*, **9**(1):1–21, January 1979.

[9] D. Gordon and J. K. Udupa, Fast surface tracking in three-dimensional binary images, *Computer Vision, Graphics, and Image Processing*, **45**(2):196–214, February 1989.

[10] Jayaram K. Udupa and Venkatramana G. Ajjangadde, Boundary and object labelling in three-dimensional images, *Computer Vision, Graphics, and Image Processing*, **51**(3):355–369, September 1990.

[11] Christian Brechbühler, Guido Gerig, and Olaf Kübler, Towards representation of 3d shape: Global surface parametrization, In C. Arcelli, L. P. Cordella, and G. Sanniti di Baya, editors, *Visual Form: Analysis and Recognition*, pp. 79–88, New York and London, 1992. Plenum Press.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. Vetterling, *Numerical recipes in C - The art of scientific computing*, Cambridge University Press, Cambridge, 1988.

[13] C. Pommerell and W. Fichtner, PILS: An iterative linear solver package for ill-conditioned systems, In *Proceedings of the Supercomputing '91, Albuquerque, New Mexico, November 18-22*, pp. 588–599, 1991.

[14] L. Nirenbarg, A strong maximum principle for parabolic equations, *Commun. Pure Appl. Math., vol. VI*, pp. 167–177, 1953.

[15] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*, Academic Press, London a.o., 1981.

[16] W. Greiner and H. Diehl, *Theoretische Physik - Ein Lehr- und Übungsbuch für Anfangssemester*, volume 3: Elektrodynamik, Verlag Harri Deutsch, Zürich und Frankfurt am Main, 1986.

[17] P. T. Sander and S. W. Zucker, Inferring surface trace and differential structure from 3-d images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**(9):833–854, September 1990.

Isosurfaces and Level-Set Surface Models

Ross T. Whitaker
Scientific Computing and Imaging Institute
School of Computing
University of Utah

1 Introduction

1.1 Motivation

This chapter addresses mechanisms for analyzing and processing volumes in a way that deals specifically with *isosurfaces*. The underlying philosophy is to use isosurfaces as a modeling technology that can serve as an alternative to parameterized models for a variety of important applications in visualization and computer graphics. This paper presents the mathematics and numerical techniques for describing the geometry of isosurfaces and manipulating their shapes in prescribed ways. We start with a basic introduction into the notation and fundamental concepts and then presents the geometry of isosurfaces. We describe the method of level sets, i.e., moving isosurfaces, and present the mathematical and numerical methods they entail. This paper concludes with some application examples and describes *VISPACK*, a *C++*, object-oriented library the performs volume processing and level-set modeling.

1.2 Isosurfaces

1.2.1 Modeling Surfaces With Volumes

When considering surface models for graphics and visualization, one is faced with a staggering variety of options including meshes, spline-based patches, constructive solid geometry, implicit blobs, and particle systems. These options can be divided into two basic classes — explicit (parameterized) models and implicit models. With an implicit model, one specifies the model as a *level set* of a scalar function,

$$\phi : \begin{matrix} U \\ x, y, z \end{matrix} \mapsto \begin{matrix} \mathbb{R} \\ k \end{matrix}, \tag{1}$$

where $U \subset \mathbb{R}^3$ is the domain of the volume (and the *range* of the surface model). Thus, a surface $\mathcal{S}$ is

$$\mathcal{S} = \{\boldsymbol{x} | \phi(\boldsymbol{x}) = k\} . \tag{2}$$

The choice of k is arbitrary, and ϕ is sometimes called the *embedding*. Notice that surfaces defined in this way divide U into a clear inside and outside—such surfaces are always closed wherever they do not intersect the boundary of the domain.

Choosing this implicit strategy begs the question of how to represent ϕ. Historically, implicit models are represented using linear combinations of *basis* functions. These basis or potential functions usually have several degrees of freedom including 3D position, size, and orientation. By combining these functions, one can create complex objects. Typical models might contain several hundred to several thousands of such primitives. This is the strategy behind the "blobby" models proposed by Blinn [1].

While such an implicit modeling strategy offers a variety of new modeling tools, it has some limitations. In particular, the global nature of the potential functions limits ones ability to model *local* surface deformations. Consider a point $\boldsymbol{x} \in \mathcal{S}$ where $\mathcal{S}$ is the level surface associated with a model $\phi = \sum_i \alpha_i$, and α_i is one of the individual potential functions that comprise that model. Suppose one wishes to move the surface at the point $\boldsymbol{x}$ in a way that maintains continuity with the surrounding neighborhood. With multiple, global basis functions one must decide which basis function or combination of basis functions to alter and at the same time control the effects on other parts of the surface. The problem is generally ill posed — there are many ways to adjust the basis functions so that $\boldsymbol{x}$ will move in the desired direction and yet it may be impossible to eliminate the effects of those movements on other disjoint parts of the surface. These problems can be overcome, however they usually entail heuristics that tie the behavior of the surface deformation to the choice of representation [2].

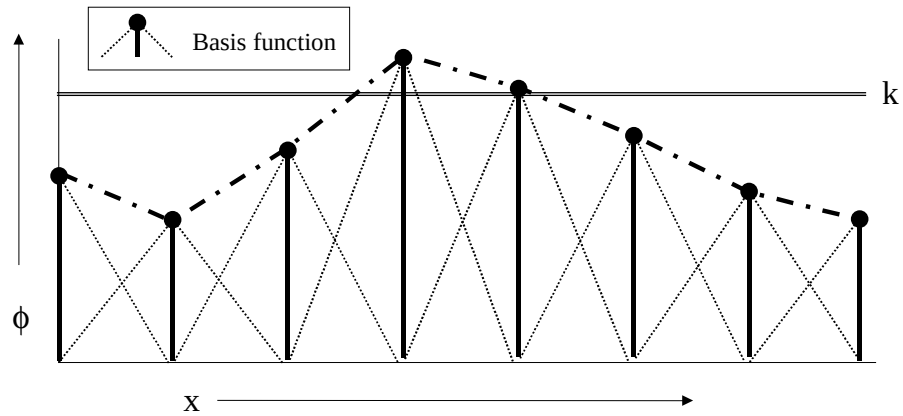


Figure 1: A volume can be considered as an implicit model with a large number of local basis functions.

An alternative to using a small number of *global* basis functions is to use a relatively large number of *local* basis functions. This is the principle behind using a volume as an implicit model. A volume is a discrete sampling of the embedding ϕ. It is also an implicit model with a very large number of basis functions, as shown in Figure 1. The total number of basis functions is fixed, as are their positions (grid points) and extent. One can change only the magnitude of each basis function, i.e., each basis function has only one degree of freedom. A typical volume of size $128 \times 128 \times 128$ contains over a million such basis functions. The shape of each basis function is open to interpretation — it depends on how one interpolates the values between the grid points. A trilinear interpolation, for instance, implies a basis function that is a piece-wise cubic polynomial with a value of one at the grid point and zero at neighboring grid points. Another advantage of using volumes as implicit models, is that for the purposes of analysis we can treat the volume as a continuous function whose values can be *set* at each point according to the application. Once the continuous analysis is complete we can map the algorithm into the discrete domain using standard methods of numerical analysis. The sections that follow discuss how to compute the geometry of surfaces that are represented as volumes and how to manipulate the shapes of those surfaces by changing the gray-scale values in the volume.

1.2.2 Isosurface Extraction and Visualization

This paper addresses the question of how to use volumes as surface models. Depending on the application, however, a 3D grid of data (i.e. a volume) may not be a suitable model representation. For instance, if the goal is make measurements of an object or visualize its shape, an explicit model might be necessary. In such cases it is beneficial to convert between volumes and other modeling technologies.

For instance, the literature proposes several methods for scan converting polygonal meshes or solid models [3, 4]. Likewise a variety of methods exist for extracting parametric models of isosurfaces from volumes. The most prevalent method is to locate isosurface crossings along grid lines in a volume (between voxels along the 3 cardinal directions) and then to link these points together to form triangles and meshes. This is the strategy of “marching cubes” [5] and other related approaches. However, extracting a parametric surface is not essential for visualization, and a variety of direct methods [6, 7] are now computationally feasible and arguably superior in quality. This chapter does not address

the issue of extracting or rendering isosurfaces, but rather studies the geometry of isosurfaces and how to manipulate them directly by changing the grey-scale values in the underlying volume. Thus, we propose volumes as a mechanism for studying and deforming surfaces, regardless of the ultimate form of the output. Their are many ways of rendering or visualizing them and and these techniques are beyond the scope of this discussion.

2 Surface Normals

The surface normal of an isosurface is given by the normalized gradient vector. Typically, we identify a surface normal with a point in the volume domain D. That is

$$\boldsymbol{n}(\boldsymbol{x}) = \frac{\nabla\phi(\boldsymbol{x})}{|\nabla\phi(\boldsymbol{x})|} \text{ where } \boldsymbol{x} \in D. \tag{3}$$

The convention regarding the direction of this vector is arbitrary; the negative of the normalized gradient magnitude is also normal to the isosurface. The gradient vector points toward that side of the isosurface which has greater values (i.e. brighter). When rendering, the convention is to use *outward pointing* normals, and the sign of the gradient must be adjusted accordingly. However, for most applications any consistent choice of normal vector will suffice. On a discrete grid, one must also decide how to approximate the gradient vector (i.e., first partial derivatives). In many cases central differences will suffice. However, in the presence of noise, especially when volume rendering, it is sometimes helpful to compute first derivatives using some smoothing filter (e.g., convolution with a Gaussian). When using the normal vector to solve certain kinds of partial differential equations, it is sometimes necessary to approximate the gradient vector with discrete, one-sided differences, as discussed in successive sections.

Note that a single volume contains families nested isosurfaces, arranged like the layers of an onion. We specific the normal to an isosurface as a function of the position within the volume. That is, $\boldsymbol{n}(\boldsymbol{x})$ is the normal of the (single) isosurface that passes through the point $\boldsymbol{x}$. The k value associated with that isosurface is $\phi(\boldsymbol{x})$.

3 Second-Order Structure

In differential geometric terms, the second-order structure of a surface is characterized by a quadratic patch that shares first- and second-order contact with the surface at a point (i.e., tangent plane and osculating circles). The *principal directions* of the surface are those associated with the quadratic approximation, and the *principal curvatures*, k_1, k_2, are the curvatures in those directions.

The second-structure of the isosurface can be computed from the first- and second-order structure of the embedding, ϕ. All of the isosurface shape information is contained field of normals given by $\boldsymbol{n}(\boldsymbol{x})$. The 3×3 matrix of derivatives of this vector,

$$N = -\left[\boldsymbol{n}_x \;\; \boldsymbol{n}_y \;\; \boldsymbol{n}_z\right] \tag{4}$$

describes the second-order structure of the surface. This matrix has (typically) rank two, and the two nonzero eigenvalues of this matrix give the principle curvatures. That is,

$$e_1 = k_1, e_2 = k_2, e_3 = 0. \tag{5}$$

The *mean curvature* is the mean of the two principal curvatures, which is one half of the trace of $N(\boldsymbol{x})$ [8]:

$$\begin{aligned} H &= \frac{k_1 + k_2}{2} = \frac{1}{2}\mathrm{Tr(N)} \\ &= \frac{\phi_x^2(\phi_{yy} + \phi_{zz}) + \phi_y^2(\phi_{xx} + \phi_{zz}) + \phi_z^2(\phi_{xx} + \phi_{yy}) - 2\phi_x\phi_y\phi_{xy} - 2\phi_x\phi_z\phi_{xz} - 2\phi_y\phi_z\phi_{yz}}{2(\phi_x^2 + \phi_y^2 + \phi_z^2)^{3/2}} \end{aligned} \tag{6}$$

The *Gaussian curvature* is the product of the principal curvatures:

$$K = k_1 k_2 = e_1 e_2 + e_1 e_3 + e_2 e_3 = 2\mathrm{Tr(N)}^2 - \frac{1}{2}||\mathrm{N}|| \tag{7}$$

$$= \frac{\begin{array}{c}\phi_z^2(\phi_{xx}\phi_{yy} - \phi_{xy}\phi_{xy}) + \phi_y^2(\phi_{xx}\phi_{zz} - \phi_{xz}\phi_{xz}) + \phi_x^2(\phi_{yy}\phi_{zz} - \phi_{yz}\phi_{yz}) \\ + 2(\phi_x\phi_y(\phi_{xz}\phi_{yz} - \phi_{xy}\phi_{zz}) + \phi_x\phi_z(\phi_{xy}\phi_{yz} - \phi_{xz}\phi_{yy}) + \phi_y\phi_z(\phi_{xy}\phi_{xz} - \phi_{yz}\phi_{xx}))\end{array}}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^2}.$$

The total curvature, also called the deviation from flatness, D, is the root sum of squares of the two principal curvatures, which is the Euclidean norm of the matrix N.

Notice, these measures exist at every point in U, and at each point they describe the geometry of the particular isosurface that passes through that point. All of these quantities can be computed on a discrete volume using finite differences, as described in successive sections.

4 Deformable Surfaces

This section begins with mathematics for describing surface deformations on parametric models. The result is an evolution equation for a surface. Each of the terms in this evolution equation can be re-expressed in a way that is independent of the parameterization. Finally, the evolution equation for a parametric surface gives rise to an evolution equation (differential equation) on a volume, which encodes the shape of that surface as a level set.

4.1 Surface Deformation

A regular surface $\mathcal{S} \subset \mathbb{R}^3$ is a collection of points in 3D that can be be represented *locally* as a continuous function. In geometric modeling a surface is typically represented as a two-parameter object in a three-dimensional space, i.e., a surface is local a mapping $\boldsymbol{S}$:

$$\boldsymbol{S} : \underset{r}{V} \times \underset{s}{V} \mapsto \underset{x,y,z}{\mathbb{R}^3}, \tag{8}$$

where $V \times V\mathbb{R}^2$, and the bold notation refers specifically to a parameterized surface (vector-valued function). A deformable surface exhibits some motion over time. Thus $\boldsymbol{S} = \boldsymbol{S}(r, s, t)$, where $t \in \mathbb{R}^+$. We assume second-order-continuous, orientable surfaces; therefore at every point on the surface (and in time) there is surface normal $\boldsymbol{N} = \boldsymbol{N}(r, s, t)$. We use $\mathcal{S}_t$ to refer to the entire set of points on the surface.

Local deformations of $\boldsymbol{S}$ can be described by an evolution equation, i.e., a differential equation on $\boldsymbol{S}$ that incorporates the position of the surface, local and global shape properties, and responses to other forcing functions. That is,

$$\frac{\partial \boldsymbol{S}}{\partial t} = \boldsymbol{G}\left(\boldsymbol{S}, \boldsymbol{S}_r, \boldsymbol{S}_s, \boldsymbol{S}_{rr}, \boldsymbol{S}_{rs}, \boldsymbol{S}_{ss}, \ldots\right), \tag{9}$$

where the subscripts represent partial derivatives with respect to those parameters. The evolution of $\boldsymbol{S}$ can be described by a sum of terms that depends on both the geometry of $\boldsymbol{S}$ and the influence of other functions or data.

There are a variety of differential expressions that can be combined for different applications. For instance, the model could move in response to some directional "forcing" function [9, 10], $\boldsymbol{F} : U \mapsto \mathbb{R}^3$, that is

$$\frac{\partial \boldsymbol{S}}{\partial t} = \boldsymbol{F}(\boldsymbol{S}). \tag{10}$$

Alternatively, the surface could expand and contract with a spatially-varying speed. For instance,

$$\frac{\partial \boldsymbol{S}}{\partial t} = G(\boldsymbol{S})\boldsymbol{N} \tag{11}$$

where $G : \mathbb{R}^3 \mapsto \mathbb{R}$ is a signed speed function. The evolution might also depend on the surface geometry itself. For instance,

$$\frac{\partial \boldsymbol{S}}{\partial t} = \boldsymbol{S}_{rr} + \boldsymbol{S}_{ss} \tag{12}$$

describes a surface that moves in way that is becomes more *smooth* with respect to its own parameterization. This motion can be combined with the motion of Equation 10 to produce a model that is pushed by a forcing function but maintains a certain smoothness in its shape and parameterization. There are myriad terms that depend on both the differential geometry of the surface and outside forces or functions to control the evolution of a surface.

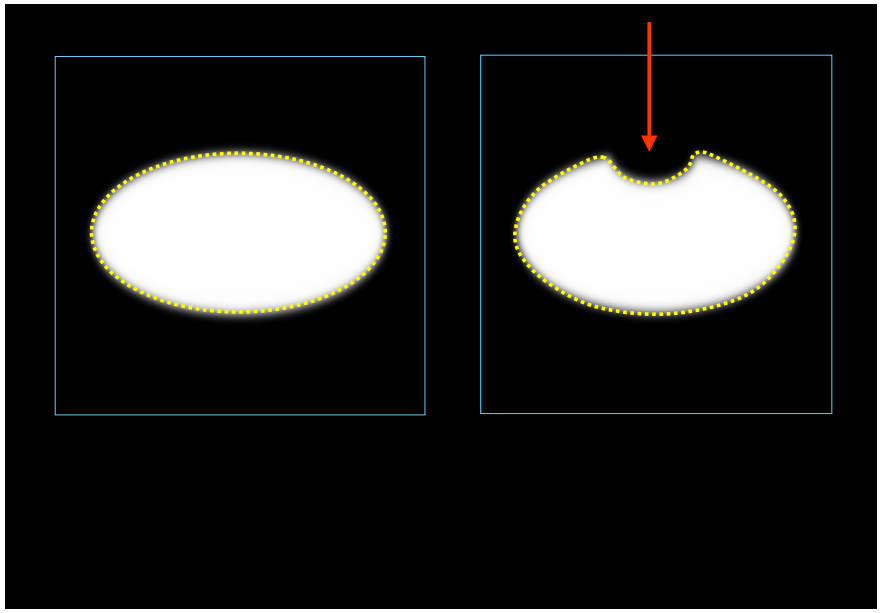


Figure 2: Level-set models represent curves and surfaces implicitly using greyscale images: a) an ellipse is represented as the level set of an image, b) to change the shape we modify the greyscale values of the image.

5 Deformation: The Level Set Approach

The method of level-sets, proposed by Osher and Sethian [11] and described extensively in [8], provides the mathematical and numerical mechanisms for computing surface deformations as time-varying iso-values of ϕ by solving a partial differential equation on the 3D grid. That is, the level-set formulation provides a set of numerical methods that describe how to manipulate the greyscale values in a volume, so that the isosurfaces of ϕ move in a prescribed manner (shown in Figure 2).

We denote the movement of a point on a surface as it deforms as $\mathrm{d}\boldsymbol{x}/\mathrm{d}t$, and we assume that this motion can be expressed in terms of the position of $\boldsymbol{x} \in U$ and the geometry of the surface at that point. In this case, there are generally two options for representing such surface movements implicitly:

Static: A single, static $\phi(\boldsymbol{x})$ contains a family of level sets corresponding to surfaces as different times t. That is,

$$\phi(\boldsymbol{x}(t)) = k(t) \quad \Rightarrow \quad \nabla\phi(\boldsymbol{x}) \cdot \frac{\partial \boldsymbol{x}}{t} = \frac{\mathrm{d}k(t)}{\mathrm{d}t}. \tag{13}$$

To solve this static method requires constructing a ϕ that satisfies equation 13. This is a boundary value problem, which can be solved somewhat efficiently starting with a single surface using the fast marching method of Sethian [12]. This representation has some significant limitations, however, because (by definition) a surface cannot pass back over itself over time, i.e., motions must be strictly monotonic — inward or outward.

Dynamic: The approach is to use a one-parameter *family* of embeddings, i.e., $\phi(\boldsymbol{x}, t)$ changes over time, $\boldsymbol{x}$ remains on the k level set of ϕ as it moves, and k remains constant. The behavior of ϕ is obtained by setting the total derivative of $\phi(\boldsymbol{x}(t), t) = k$ to zero. Thus,

$$\phi(\boldsymbol{x}(t), t) = k \quad \Rightarrow \quad \frac{\partial \phi}{\partial t} = -\nabla\phi \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}. \tag{14}$$

This approach can accommodate models that move forward and backward and cross back over their own paths (over time). However, to solve this requires solving the initial value problem (using finite forward differences)

on $\phi(\boldsymbol{x}, t)$ — a potentially large computational burden. The remainder of this discussion focuses on the dynamic case, because of its superior flexibility.

All surface movements depend on position and geometry, and the level-set geometry is expressed in terms of the differential structure of ϕ. Therefore the dynamic formulation from equation 14 gives a general form of the partial differential equation on ϕ:

$$\frac{\partial \phi}{\partial t} = -\nabla \phi \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = -\nabla \phi \cdot \boldsymbol{F}(\boldsymbol{x}, \mathrm{D}\phi, \mathrm{D}^2\phi, \ldots), \tag{15}$$

where $D^n\phi$ is the set of order-n derivatives of ϕ evaluated at $\boldsymbol{x}$. Because this relationship applies to every level-set of ϕ, i.e. all values of k, this equation can be applied to all of U, and therefore the movements of *all* the level-set surfaces embedded in ϕ can be calculated from Equation 15.

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

Such level-set methods are well documented in the literature [11, 13] for applications such as computational physics [14], image processing [15, 16], computer vision [17, 18], medical image analysis [19, 18], and 3D reconstruction [20, 21]. For instance, in computational physics level-set methods are a a powerful tool for modeling moving interfaces between different materials (see Osher and Fedkiw [14] for a nice overview of recent results). Examples are water-air and water-oil. In such cases, level-set methods can be used to compute deformations that minimize surface area while preserving volumes for materials that *split and merge* in arbitrary ways. The method can be extended to multiple, non-overlapping objects.

Level-set methods have also been shown to be effective in extracting surface structures from biological and medical data. For instance Malladi *et al.* [18] propose a method in which the level-sets form an expanding or contracting contour which tends to "cling" to interesting features in 2D angiograms. At the same time the contour is also influenced by its own curvature, and therefore remains smooth. Whitaker *et al.* [19, 22] have shown that level sets can be used to simulate conventional deformable surface models, and demonstrated this by extracting skin and tumors from *thick-sliced* (e.g. clinical) MR data, and by reconstructing a fetal face from 3D ultrasound. A variety of authors [23, 24, 16, 25] have presented variations on the method and presented results for 2D and 3D data. Sethian [8] gives several examples of level-set curves and surface for segmenting CT and MR data.

5.1 Deformation Modes

In the case of parametric surfaces, one can choose from a variety of different expressions to construct an evolution equation that is appropriate for a particular application. For each of those parametric expressions, there is a corresponding expression that can be formulated on ϕ, the volume in which the level-set models are embedded. In constructing evolutions on levels sets, there can be no reference to the underlying surface parameterization (terms depending on r and s in Equations 8 through 12). This has two important implications: 1) only those surface movements that are normal to the surface are represented—any other movement is equivalent to a reparameterization 2) all of the derivatives with respect to surface parameters r and s must be expressed in terms of invariant surface properties that can be derived without a parameterization.

Consider the term $\boldsymbol{S}_{rr} + \boldsymbol{S}_{ss}$ from equation 12. If r, s is an orthonormal parameterization, the effect of that term is based purely on surface shape, not on the parameterization, and the expression $\boldsymbol{S}_{rr} + \boldsymbol{S}_{ss}$ is twice the *mean curvature*, H, of the surface. The corresponding level-set formulation is given by Equation 6.

Table 1 shows a list of expressions used in the evolution of parameterized surfaces and their equivalents for level-set representations. Also given are the assumptions about the parameterization that give rise to the level-set expressions.

6 Numerical Methods

By taking the strategy of embedding surface models in volumes, we have converted equations that describe the movement of surface points to nonlinear, partial differential equations defined on a volume, which is generally a rectilinear

	Effect	Parametric Evolution	Level-Set Evolution	Parameter Assumptions
1	External force	$\boldsymbol{F}$	$\boldsymbol{F} \cdot \nabla\phi$	None
2	Expansion/ contraction	$G(\boldsymbol{x})\boldsymbol{N}$	$G(\boldsymbol{x})\lvert\nabla\phi(\boldsymbol{x},t)\rvert$	None
3	Mean curvature	$S_{rr} + S_{ss}$	$H\lvert\nabla\phi\rvert$	Orthonormal
4	Gauss curvature	$S_{rr} \times S_{ss}$	$K\lvert\nabla\phi\rvert$	Orthonormal
5	Second order	S_{rr} or S_{ss}	$\left(H \pm \sqrt{H^2 - K}\right)\lvert\nabla\phi\rvert$	Principal curvatures

Table 1: A list of evolution terms for parametric models has a corresponding expression on the embedding, ϕ, associated with the level-set models.

grid. The expression $u^n_{i,j,k}$ refers to the nth time step at position i, j, k, which has an associated value in the 3D domain of the continuous volume $\phi(x_i, y_j, z_k)$. The goal is to solve the differential equation consisting of terms from Table 5.1 on the discrete grid $u^n_{i,j,k}$.

The discretization of these equations raises two important issues. First is the availability of accurate, stable numerical schemes for solving these equations. Second is the problem of computational complexity and the fact that we have converted a *surface* problem to a *volume* problem, increasing the dimensionality of the domain over which the evolution equations must be solved.

The level-set terms in Table 1 are combined, based on the needs of the application, to create a partial differential equation on $\phi(\boldsymbol{x}, t)$. The solutions to these equations are computed using finite differences. Along the time axis solutions are obtained using finite *forward* differences, beginning with an initial model (i.e., volume) and stepping sequentially through a series of discrete times steps (which are denoted as superscripts on u). Thus the update equation is:

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t \Delta u^n_{i,j,k}, \tag{16}$$

The term $\Delta u^n_{i,j,k}$ is a discrete approximation to $\partial\phi/\partial t$, which consists of a weighted sum of terms such as those in Table 5.1. Those terms must, in turn, be approximated using finite differences on the volume grid.

6.1 Up-wind Schemes

The terms in Table 1 fall into two basic categories: the first-order terms (items 1 and 2 in Table 1) and the second-order terms (items 3 through 5). The first-order terms describe a moving wave front with a space-varying velocity (expression 1) or speed (expression 2). Equations of this form cannot be solved with a simple finite forward difference scheme. Such schemes tend to overshoot, and they are unstable. To address this issue Osher and Sethian [26] have proposed an *up-wind* scheme. The up-wind method relies on a one-sided derivative that looks in the up-wind direction of the moving wave front, and thereby avoids the over-shooting associated with finite forward differences.

We denote the type of discrete difference using superscripts on a difference operator, i.e., $\delta^{(+)}$ for forward differences, $\delta^{(-)}$ for backward differences, and δ for central differences. For instance, differences in the x direction on a discrete grid, $u_{i,j,k}$, with domain X and uniform spacing h are defined as

$$\delta^{(+)}_x u_{i,j,k} \triangleq (u_{i+1,j,k} - u_{i,j,k})/h, \tag{17}$$

$$\delta^{(-)}_x u_{i,j,k} \triangleq (u_{i,j,k} - u_{i-1,j,k})/h, \text{ and} \tag{18}$$

$$\delta_x u_{i,j,k} \triangleq (u_{i+1,j,k} - u_{i-1,j,k})/(2h), \tag{19}$$

$$\tag{20}$$

where we have left off the time superscript for conciseness. Second-order terms are computed using the *tightest-fitting*

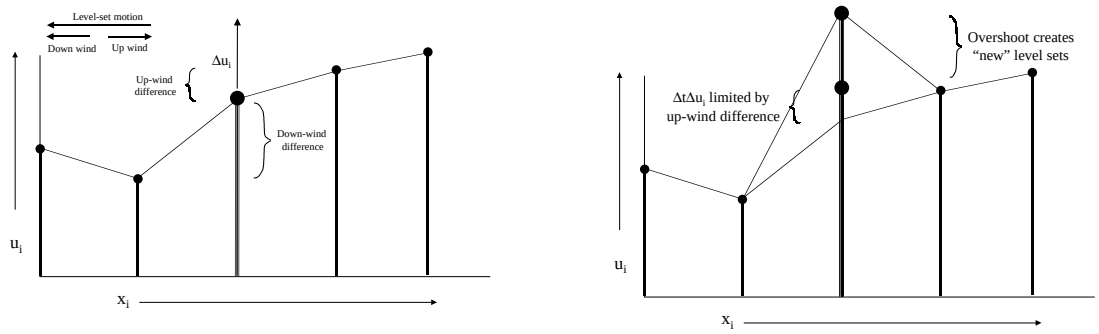


Figure 3: The up-wind numerical scheme uses one-sided derivatives to prevent overshooting and the creation of new level sets.

central difference operators. For example,

$$\delta_{xx} u_{i,j,k} \triangleq (u_{i+1,j,k} + u_{i-1,j,k} - 2u_{i,j,k}) / h^2, \tag{21}$$

$$\delta_{zz} u_{i,j,k} \triangleq (u_{i,j,k+1} + u_{i,j,k-1} - 2u_{i,j,k}) / h^2, \text{ and} \tag{22}$$

$$\delta_{xy} u_{i,j,k} \triangleq \delta_x \delta_y u_{i,j,k} \tag{23}$$

The discrete approximation to the first-order terms of in Table 5.1 are computed using the up-wind proposed by Osher and Sethian [11]. This strategy avoids overshooting by approximating the gradient of ϕ using a one-sided differences in the direction that is up-wind of the moving level-set thereby ensuring that no *new* contours are created in the process of updating $u^n_{i,j,k}$ (as depicted in Figure 3). The scheme is separable along each axis (i.e., x, y, and z).

Consider Term 1 in Table 5.1. If we use superscripts to denote the vector components, i.e.,

$$\boldsymbol{F}(x,y,z) = (F^{(x)}(x,y,z), F^{(y)}(x,y,z), F^{(z)}(x,y,z)), \tag{24}$$

the up-wind calculation for a grid point $u^n_{i,j,k}$ is

$$\boldsymbol{F}(x_i,y_i,z_i) \cdot \nabla\phi(x_i,y_j,z_k,t) \approx \sum_{q\in\{x,y,z\}} F^{(q)}(x_i,y_i,z_i) \begin{cases} \delta^+_q u^n_{i,j,k} & F^{(q)}(x_i,y_i,z_i) > 0 \\ \delta^i_q u^n_{i,j,k} & F^{(q)}(x_i,y_i,z_i) < 0 \end{cases} \tag{25}$$

The time steps are limited—the fastest moving wave front can move only one grid unit per iteration. That is

$$\Delta t_{\boldsymbol{F}} \leq \frac{1}{\sum_{q\in\{x,y,z\}} \sup_{i,j,k\in X}\{|\nabla F^{(q)}(x_i,y_j,z_k)|\}}. \tag{26}$$

For Term 2 in Table 5.1 the direction of the moving surface depends on the normal, and therefore the same up-wind strategy is applied in a slightly different form.

$$\begin{aligned} &G(x_i,y_j,z_k)|\nabla\phi(x_i,y_j,z_k,t)| \approx \\ &\sum_{q\in\{x,y,z\}} G(x_i,y_i,z_i) \begin{cases} \max^2(\delta^+_q u^n_{i,j,k},0) + \min^2(\delta^-_q u^n_{i,j,k},0) & G(x_i,y_i,z_i) > 0 \\ \min^2(\delta^+_q u^n_{i,j,k},0) + \max^2(\delta^-_q u^n_{i,j,k},0) & G(q)(x_i,y_i,z_i) < 0 \end{cases} \end{aligned} \tag{27}$$

The time steps are, again, limited by the fastest moving wave front:

$$\Delta t_G \leq \frac{1}{3\sup_{i,j,k\in X}\{|\nabla G(x_i,y_j,z_k)|\}} \tag{28}$$

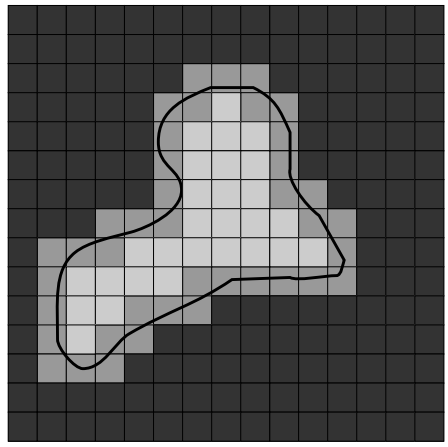


Figure 4: A level curve of a 2D scalar field passes through a finite set of cells. Only those grid points nearest to the level curve are relevant to the evolution of that curve.

To compute approximation the update to the second-order terms in Table 5.1 requires only central differences . Thus, the mean curvature is approximated as:

$$\begin{aligned} H^n_{i,j,k} &= \frac{1}{2}\left(\left(\delta_x u^n_{i,j,k}\right)^2 + \left(\delta_y u^n_{i,j,k}\right)^2 + \left(\delta_z u^n_{i,j,k}\right)^2\right)^{-1}\Big[\left(\left(\delta_y u^n_{i,j,k}\right)^2 + \left(\delta_z u^n_{i,j,k}\right)^2\right)\delta_{xx}u^n_{i,j,k} \\ &+ \left(\left(\delta_z u^n_{i,j,k}\right)^2 + \left(\delta_x u^n_{i,j,k}\right)^2\right)\delta_{yy}u^n_{i,j,k} + \left(\left(\delta_x u^n_{i,j,k}\right)^2 + \left(\delta_y u^n_{i,j,k}\right)^2\right)\delta_{zz}u^n_{i,j,k} \\ &-2\delta_x u^n_{i,j,k}\delta_y u^n_{i,j,k}\delta_{xy}u^n_{i,j,k} - 2\delta_y u^n_{i,j,k}\delta_z u^n_{i,j,k}\delta_{yz}u^n_{i,j,k} - 2\delta_z u^n_{i,j,k}\delta_x u^n_{i,j,k}\delta_{zx}u^n_{i,j,k}\Big] \end{aligned} \tag{29}$$

Such curvature terms can be computing by using a combination of forward and backward differences as described in [27]. In some cases this is advantageous—but the details are beyond the scope of this paper.

The time steps are limited, for stability, to

$$\Delta t_H \leq \frac{1}{6}. \tag{30}$$

When combining terms, the maximum time steps for each terms is scaled by one over the weighting coefficient for that term.

6.2 Narrow-Band Methods

If one is interested in only *a single level set*, the formulation described previously is not efficient. This is because solutions are usually computed over the entire domain of ϕ. The solutions, $\phi(x, y, z, t)$ describe the evolution of an embedded family of contours. While this dense family of solutions might be advantageous for certain applications, there are other applications that require only a single surface model. In such applications the calculation of solutions over a dense field is an unnecessary computational burden, and the presence of contour families can be a nuisance because further processing might be required to extract the level set that is of interest.

Fortunately, the evolution of a single level set, $\phi(\boldsymbol{x}, t) = k$, is not affected by the choice of embedding. The evolution of the level sets is such that they evolve independently (to within the error introduced by the discrete grid). Furthermore, the evolution of ϕ is important only in the vicinity of that level set. Thus, one should perform calculations for the evolution of ϕ only in a neighborhood of the surface $\mathcal{S} = \{\boldsymbol{x}|\phi(\boldsymbol{x}) = k\}$. In the discrete setting, there is a particular subset of grid points whose values control a particular level set (see Figure 4). Of course, as the surface moves, that subset of grid points must change to account for its new position.

Adalsteinson and Sethian [28] propose a *narrow-band* approach which follows this line of reasoning. The narrow-band technique constructs an embedding of the evolving curve or surface via a signed distance transform. The distance transform is truncated, i.e, computed over a finite width of only m points that lie within a specified distance to the level set. The remaining points are set to constant values to indicate that they do not lie within the narrow band, or *tube*

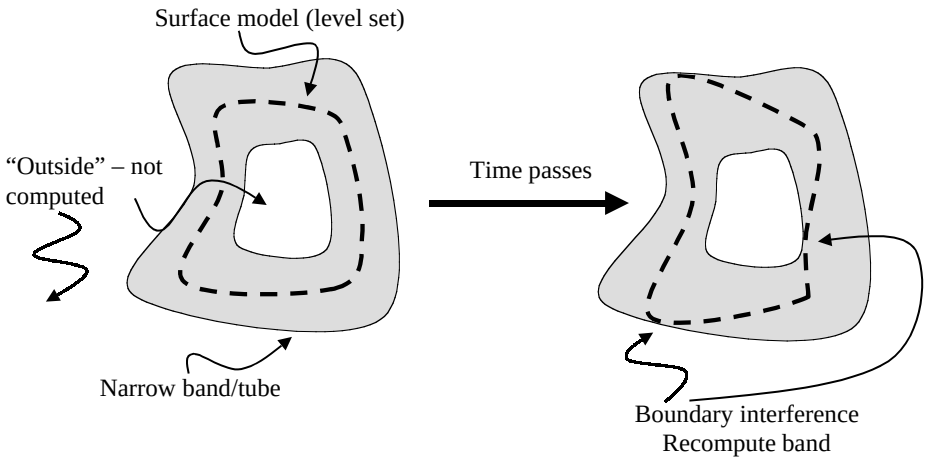


Figure 5: The narrow band scheme limits computation to the vicinity of the specific level set. As the level-set moves near the edge of the band the process is stopped and the band recomputed.

as they call it. The evolution of the surface (they demonstrate it for curves in the plane) is computed by calculating the evolution of u only on the set of grid points that are within a fixed distance to the initial level set, i.e. within the narrow band. When the evolving level set approaches the edge of the band (see Figure 5), they calculate a new distance transform and a new embedding, and they repeat the process. This algorithm relies on the fact that the embedding is not a critical aspect of the evolution of the level set. That is, the embedding can be transformed or recomputed at any point in time, so long as such a transformation does not change the position of the kth level set, and the evolution will be unaffected by this change in the embedding.

Despite the improvements in computation time, the narrow-band approach is not optimal for several reasons. First it requires a band of significant width ($m = 12$ in the examples of [28]) where one would like to have a band that is only as wide as necessary to calculate the derivatives of u near the level set (e.g. $m = 2$). The wider band is necessary because the narrow-band algorithm trades off two competing computational costs. One is the cost of stopping the evolution and computing the position of the curve and distance transform (to sub-cell accuracy) and determining the domain of the band. The other is the cost of computing the evolution process over the entire band. The narrow-band method also requires additional techniques, such as smoothing, to maintain the stability at the boundaries of the band, where some grid points are undergoing the evolution and nearby neighbors are static.

6.3 The Sparse-Field Method

The basic premise of the narrow band algorithm is that computing the distance transform is so costly that it cannot be done at every iteration of the evolution process. The strategy proposed here is to use an approximation to the distance transform that makes it feasible to recompute the neighborhood of the level-set model at each time step. Computation of the evolution equation is computed on a band of grid points that is only on point wide. The embedding is extended from the active points to a neighborhood around those points that is precisely the width needed at each time. This extension is done via a fast distance transform approximation.

This approach has several advantages. First, the algorithm does precisely the number of calculations needed to

compute the next position of the level curve. It does not require explicitly recalculating the positions of level sets and their distance transforms. Because the number of points being computed is so small, it is feasible to use a linked-list to keep track of them. Thus, at each iteration the algorithm visits only those points adjacent to the k-level curve. For large 3D data sets, the very process of incrementing a counter and checking the status of all of the grid points is prohibitive.

The *sparse-field* algorithm is analogous to a locomotive engine that lays down tracks before it and picks them up from behind. In this way the number of computations increases with the surface area of the model rather than the resolution of the embedding. Also, the sparse-field approach identifies a single level set with a specific set of points whose values control the position of that level set. This allows one to compute external forces to an accuracy that is better than the grid spacing of the model, resulting in a modeling system that is more accurate for various kinds of "model fitting" applications.

The sparse-field algorithm takes advantage of the fact that a k-level surface, S, of a discrete image u (of any dimension) has a set of cells through which it passes, as shown in Figure 4. The set of grid points adjacent to the level set is called the *active set*, and the individual elements of this set are called *active points*. As a first-order approximation, the distance of the level set from the center of any active point is proportional to the value of u divided the gradient magnitude at that point. Because all of the derivatives (up to second order) in this approach are computed using nearest neighbor differences, only the active points and their neighbors are relevant to the evolution of the level-set at any particular time in the evolution process. The strategy is to compute the evolution given by equation 15 on the active set and then update neighborhood around the active set using a fast distance transform. Because active points must be adjacent to the level-set model, their positions lie within a fixed distance to the model. Therefore the values of u for locations in the active set must lie within a certain range. When active-point values move out of this *active range* they are no longer adjacent to the model. They must be removed from the set and other grid points, those whose values are moving into the active range, must be added to take their place. The precise ordering and execution of these operations is important to the operation of the algorithm.

The values of the points in the active set can be updated using the up-wind scheme for first-order terms and central differences for the mean-curvature flow, as described in the previous sections. In order to maintain stability, one must update the neighborhoods of active grid points in a way that allows grid points to enter and leave the active set without those changes in status affecting their values. Grid points should be removed from the active set when they are no longer the nearest grid point to the zero crossing. If we assume that the embedding u is a discrete approximation to the distance transform of the model, then the distance of a particular grid point, $x_m = (i, j, k)$, to the level set is given by the value of u at that grid point. If the distance between grid points is defined to be unity, then we should remove a point from the active set when the value of u at that point no longer lies in the interval $[-\frac{1}{2}, \frac{1}{2}]$ (see Figure 6). If the neighbors of that point maintain their distance of 1, then those neighbors will move into the active range just x_m is ready to be removed.

There are two operations that are significant to the evolution of the active set. First, the values of u at active points change from one iteration to the next. Second, as the values of active points pass out of the active range they are removed from the active set and other, neighboring grid points are added to the active set to take their place. In [21] the author gives some formal definitions of active sets and the operations that affect them, which show that active sets will always form a boundary between positive and negative regions in the image, even as control of the level set passes from one set off active points to another.

Because grid points that are near the active set are kept at a fixed value difference from the active points, active points serve to control the behavior of non-active grid points to which they are adjacent. The neighborhoods of the active set are defined in *layers*, $L_{+1}, \ldots L_{+N}$ and $L_{-1}, \ldots L_{-N}$, where the i indicates the distance (city block distance) from the nearest active grid point, and negative numbers are used for the outside layers. For notational convenience the active set is denoted L_0.

The number of layers should coincide with the size of the footprint or neighborhood used to calculate derivatives. In this way, the inside and outside grid points undergo no changes in their values that affect or distort the evolution of the zero set. Most of the level-set work relies on surface normals and curvature, which require only second-order derivatives of ϕ. Second-order derivatives are calculated using a $3 \times 3 \times 3$ kernel (city-block distance 2 to the corners). Therefore only five layers are necessary (2 inside layers, 2 outside layers, and the active set). These layers are denoted L_1, L_2, L_{-1}, L_{-2}, and L_0.

The active set has grid point values in the range $[-\frac{1}{2}, \frac{1}{2}]$. The values of the grid points in each neighborhood layer are kept 1 unit from the next layer closest to the active set (as in Figure 6). Thus the values of layer L_i fall in the interval $[i - \frac{1}{2}, i + \frac{1}{2}]$. For $2N + 1$ layers, the values of the grid points that are totally inside and outside are $N + \frac{1}{2}$

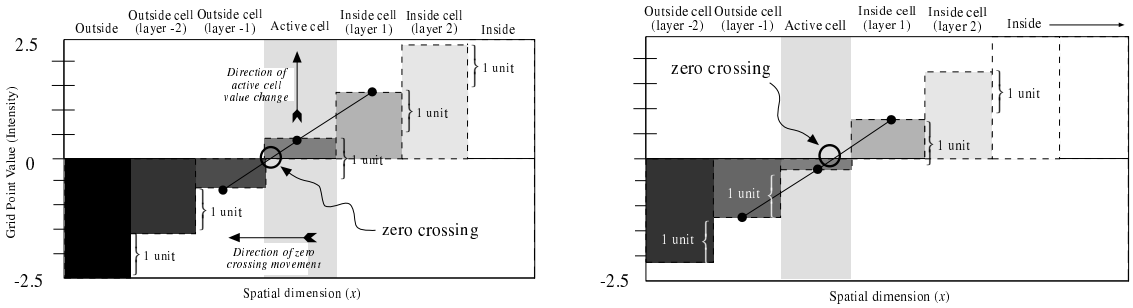


Figure 6: The status of grid points and their values at two different points in time show that as the zero crossing moves, *activity* is passed one grid point to another.

and $-N - \frac{1}{2}$, respectively. The procedure for updating the image and the active set based on surface movements is as follows:

1. For each active grid point, $x_m = (i, j, k)$, do the following:
   (a) Calculate the local geometry of the level set.
   (b) Compute the net change of u_{x_m}, based on the internal and external forces, using some stable (e.g., upwind) numerical scheme where necessary.
2. For each active grid point x_j add the change to the grid point value and decide if the new value $u_{x_m}^{n+1}$ falls outside the $[-\frac{1}{2}, \frac{1}{2}]$ interval. If so, put x_m on lists of grid points that are changing status, called the *status list*; S_1 or S_{-1}, for $u_{x_m}^{n+1} > 1$ or $u_{x_m}^{n+1} < -1$, respectively.
3. Visit the grid points in the layers L_i in the order $i = \pm 1, \ldots \pm N$, and update the grid point values based on the values (by adding or subtracting one unit) of the next inner layer, $L_{i \mp 1}$. If more than one $L_{i \mp 1}$ neighbor exists then use the neighbor that indicates a level curve closest to that grid point, i.e., use the maximum for the outside layers and minimum for the inside layers. If a grid point in layer L_i has no $L_{i \mp 1}$ neighbors, then it gets demoted to $L_{i \pm 1}$, the next level away from the active set.
4. For each status list $S_{\pm 1}, S_{\pm 2}, \ldots, S_{\pm N}$ do the following:
   (a) For each element x_j on the status list S_i, remove x_j from the list $L_{i \mp 1}$, and add it to the L_i list, or, in the case of $i = \pm(N+1)$, remove it from all lists.
   (b) Add all $L_{i \mp 1}$ neighbors to the $S_{i \pm 1}$ list.

This algorithm can be implemented efficiently using linked-list data structures combined with arrays to store the values of the grid points and their states as shown in Figure 7. This requires only those grid points whose values are changing, the active points and their neighbors, to be visited at each time step. The computation time grows as m^{n-1}, where m is the number of grid points along one dimension of u (sometimes called the resolution of the discrete sampling). Computation time for dense-field approach increases as m^n. The m^{n-1} growth in computation time for the sparse-field models is consistent with conventional (parameterized) models, for which computation times increase with the resolution of the domain, rather than the range.

Another important aspect of the performance of the sparse-field algorithm is the larger time steps that are possible. The time steps are limited by the speed of the "fastest" moving level curve, i.e., the maximum of the force function. Because the sparse-field method calculates the movement of level sets over a subset of the image, time steps are bounded from below by those of the dense-field case, i.e.,

$$\sup_{x \in \mathcal{A} \subset X} (g(x)) \leq \sup_{x \in X} (g(x)), \tag{31}$$

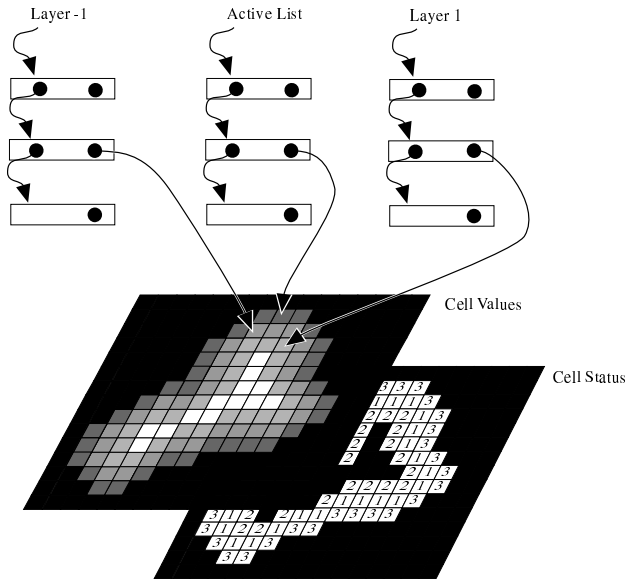


Figure 7: Linked-list data structures provide efficient access to those grid points with values and status that must be updated.

where $g(x)$ is the space varying speed function and $\mathcal{A}$ is the active set.

Results from previous work [21] have demonstrated several important aspects of the sparse-field algorithm. First, the manipulations of the active set and surrounding layers allow the active set to "track" the deformable surface as it moves. The active set always divides the inside and outside of the objects it describes (i.e., it stays closed). Empirical results show significant increases in performance relative to both the computation of full domain and the narrow-band method, as proposed in the literature. Empirical results also show that the sparse-field method is about as accurate as both the full, discrete solution, and the narrow-band method. Finally, because the method positions level sets to sub-voxel accuracy it avoids aliasing problems and is more accurate then these other methods when it comes to *fitting* level-set models to other surfaces. This sub-voxel accuracy is important aspect of the implementation, and will significantly impact the quality of the results for the applications that follow.

7 Applications

This section describes several examples of how level-set surface models can be used to address problems in graphics, visualization, and computer vision. These examples are a small selection of those available in the literature. All of these examples where implemented using the sparse-field algorithm and the VISPack library, which is described in the section that follows.

7.1 Surface Morphing

This section summarizes the work of [29], which describes the use of level-set surface models to perform 3D shape metamorphosis. The *morphing* of 3D surfaces is the process of constructing a series of 3D models that constitute a smooth transition from one shape to another (i.e., a homotopy). Such a capability is interesting for creating animations and as a tool for geometric modeling. There is not yet a single, general method for generating such transitional shapes. However, there are several desirable aspects of morphing algorithms that allow us to compare the adequacy of different approaches to surface morphing. Several desirable properties of 3D surface morphing are:

1. The transition process should begin with an *initial* surface and end with a specified *target* surface.

2. The morphing algorithm should apply to a wide range of shapes and topologies.

3. Intermediate surfaces should undergo continuous 3D transitions (rather than continuity only in the image space).

4. A 3D morphing algorithm should incorporate user input easily but should degrade gracefully without it.

5. Transitional shapes should depend only on the surface geometry of the two input shapes and user input.

These requirements are not exhaustive, but they capture many of the practical aspects of 3D morphing.

In this section we show how level-set models provide an algorithm for 3D morphing which meets most of these criteria and compare favorably with existing algorithms. Furthermore, this algorithm is a natural extension of the mathematical principles discussed in previous sections. The strategy is to allow a free-form deformation of one surface (called the *initial* surface) using the signed distance transform of a second surface (the *target* surface). This free-form deformation is combined with an underlying coordinate transformation that gives either a rough global alignment of the two surfaces, or one-to-one relationships between a finite set of landmarks on both the initial and target surfaces. The coordinate transformation can be computed automatically or using user input (as in [30]).

Much of the previous 3D morphing work has focused on morphing parametric models [31, 32] and applies to only very limited classes of shapes and topologies. Several authors have described volumetric techniques. Hughes [33] demonstrates how volumes can provide topological flexibility in surface morphing. Lerios et al. [30] followed up with a volume-based scheme which incorporates user input via underlying coordinate transformations (a known generalization the image warping technique that is often used in image morphing). Neither of these approaches have dealt with the deeper issue of deforming the level sets of a volume, but rather rely on the properties of the embedding. Payne and Toga [34] as well as Cohen-Or *et al.* [35] fix the embedding problem by using a signed distance transform to create volumes from surfaces. However, interpolating distance transforms can introduce artifacts that violate the previously stated properties, and both of these methods use a discrete distance transform which introduces volume aliasing.

7.1.1 Free-Form Deformations

The distance transform gives the nearest Euclidean distance to a set of points, curve, or surface. For closed surfaces in 3D, the signed distance transform gives a positive distance for points inside and negative for points outside (one can also choose the opposite sign convention).

If two connected shapes overlap then the initial surface can expand or contract using the distance transform of the target. The steady state of such a deformation process is a shape consisting of the zero set of the distance transform of the target. That is, the initial object becomes the target. This is the basis of the proposed 3D morphing algorithm.

Let $D(\boldsymbol{x})$ be the signed distance transform of the target surface, B, and let A be the initial surface. The evolution process which takes a model S from A to B is defined by

$$\frac{\partial \boldsymbol{x}}{\partial t} = \boldsymbol{N}\ D(\boldsymbol{x}), \tag{32}$$

where $\boldsymbol{x}(t) \in \mathcal{S}_t$ and $\mathcal{S}_{t=0} = A$. The free-form deformations can be combined with an underlying coordinate transformation. The strategy is to use a coordinate transformation (for instance a translation and rotation) to position the two surfaces near each other. These transformations can capture gross similarities in shape as well as user input. A coordinate transformation is given by

$$\boldsymbol{x}' = T(\boldsymbol{x}, \alpha), \tag{33}$$

where $0 \leq \alpha \leq 1$ parameterizes a continuous family of these transformations that begins with identity, i.e. $\boldsymbol{x} = T(\boldsymbol{x}, 0)$. The evolution equation for a parametric surface is

$$\frac{\partial \boldsymbol{x}}{\partial t} = \boldsymbol{N}\ D(T(\boldsymbol{x}, 1)), \tag{34}$$

and the corresponding level-set equation is

$$\frac{\partial \Phi(\boldsymbol{x}, t)}{\partial t} = |\nabla \Phi(\boldsymbol{x}, t)|\ D(T(\boldsymbol{x}, 1)). \tag{35}$$

This process produces a series of transition shapes (parameterized by t). The coordinate transformation can be a global rotation, translation, or scaling, or it might be a *warping of the underlying 3D space* as was used by [30].

Figure 8: A 3D model of a jet that was built using Clockworks, a CSG modeling system.

Incorporating user input is important for any surface morphing technique, because in many cases finding the best set of transition surfaces depends on context. Only users can apply semantic considerations to the transformation of one object to another. However, this underlying coordinate transformation can, in general, achieve only some finite similarity between the "warped" initial model and the target, and even this may require a great deal of user input. In the event that a user is not able or willing to define every important correspondence between two objects, some other method must "fill in" the gaps remaining between the initial and target surface. In [30] they propose alpha blending to achieve that smooth transition—really just a fading from one surface to the other. We are proposing the use of the free-form deformations, implemented with level-set models, to achieve a continuous transition between the shapes that result from the underlying coordinate transformation. We have also experimented with ways of automatically orienting and scaling objects, using 3D moments, in order to achieve a significant correspondence between two objects.

Figure 8 shows a 3D model of a jet that was built using Clockworks [36], a CSG modeling system. Lerios et al. [30] demonstrate the transition of a jet to a dart, which was accomplished using 37 user-defined correspondences, roughly a hundred user-defined parameters. Figure 9 shows the use of level-set models to construct a set of transition surfaces between a jet and a dart. The triangle mesh is extracted from the volume using the method of marching cubes [5]. These results are obtained without any user input. Distance transforms on the CSG models are computed near the level surface using an analytical description and extended into the volume using a level-set method [37].

The application in this section shows how level-set models moving according to the first-order term given in expression 2 in Table 1 can "fit" other objects by moving with a speed that depends on the signed distance transform of the target object. The application in the next section relies on expression 5 of Table 1, a second-order flow that depends on the principal curvatures of the surface itself.

7.2 Filleting and Blending Solid Objects

The construction of blending surfaces is an important tool in solid modeling. Geometric solid primitives and their intersections often produce sharp corners or creases that are often not consistent with the real-world objects that they are intended to represent. This section shows how blending can be described as a deformation process, where surfaces move under a geometric flow that can add or remove material based on local curvature information. The result is a method for solid object blending that does not depend on any particular model representation. Thus this method is not restricted to a specific class of shapes or topologies. Additionally, the results are invariant; they do not depend on arbitrary choices of coordinate systems or bases. The only requirement is that the blended objects must be closed surfaces with some known inside-outside function.

Surface blending techniques are typically tied very closely to the choice of geometric primitives. For instance,

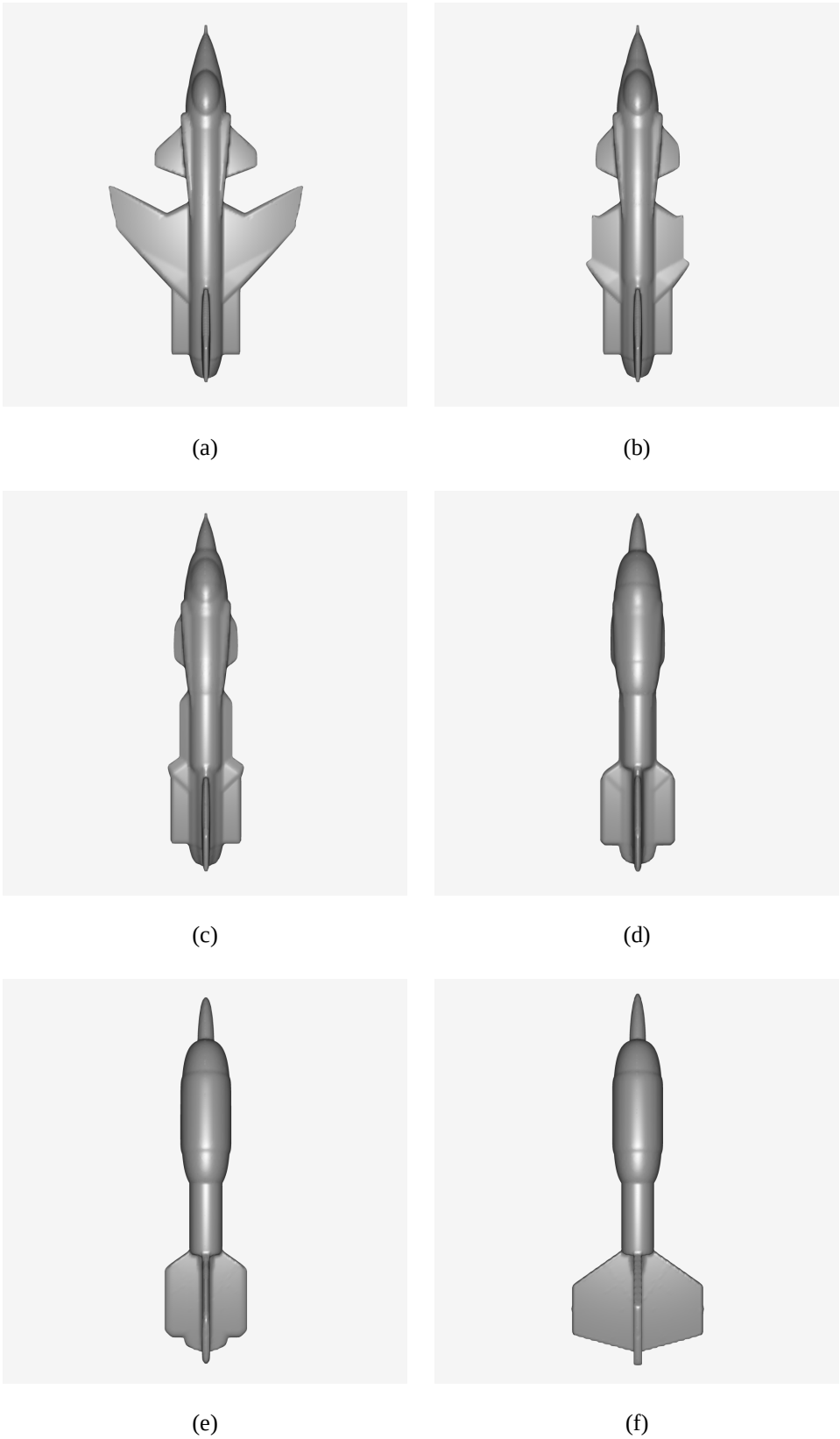


Figure 9: The deformation of the jet to a dart using a level-set model moving with a speed defined by the signed distance transform of the target object.

Middleditch and Sears [38] propose a set-theoretic method for blending solids which relies on low-order algebraic primitives. A fillet at the joint of two tori requires the solution of a degree 32 polynomial. Bloomenthal and Shoemake [39] propose a modeling system based on convolutions, which relies on a skeletonized representation of objects. In general the use of convolution to achieve deformations on implicit shapes results in shapes that reflect both the shape of the model and the embedding, Φ.

The blending method proposed in this section implements an interative smoothing scheme that smooths only along the level set; the final result is independent of the embedding. Consider the case of fillets. We propose that a fillet can be constructed from a process of "filling in" material in places of high curvature. The curvature of a level-set model can be calculated from the embedding, and the deformation of the level set is well defined by the curvature terms in Table 1.

The strategy is to construct a curvature term, k_p, that consists of only positive curvatures. [1] The principal curvatures of the level sets of Φ are functions of Φ and its derivatives. For a specific Φ the principal curvatures are functions of 3-space $k_1(\boldsymbol{x})$ and $k_2(\boldsymbol{x})$. For *adding* material the joint between two objects, we consider only the positive curvature components, i.e.,

$$\frac{\partial \Phi}{\partial t} = |\nabla \Phi| k_p = |\nabla \Phi| k_1^+ + |\nabla \Phi| k_2^+, \tag{36}$$

where k^+ consists of only the positive parts of k and is defined as zero elsewhere. Because the use of separate curvature terms can cause over-shooting, the up-wind scheme (treating k_p as a space-varying velocity in the normal direction) is used for this evolution.

Figure 10 shows how the positive-curvature flow can be used to construct fillets. No knowledge of the underlying models is necessary. The fillets grow larger as more time passes. The physical extent or position of the fillet can be controlled by either specifying a region of action or by placing a small blob of deformable material in the joint that requires a fillet. Figure 11 shows how such a blending capability can be useful in animation. In this case a pair of superquadrics undergo a rigid transformation that controls their relative positions. Level-set models with a positive-curvature flow are used to create a smooth joint between these two primitives. Notice that the positive curvature method does not suffer from the growth or expansion artifacts that are often associated with distance-based blending methods [40].

Thus, a second-order flow can create smooth blends between objects in a way that does not require specific knowledge of the shapes or topologies of the object involved. The application in the next section, 3D scene reconstruction, shows how a combination of first-order and second-order terms from Table 1 are combined to create technique that fits models to data while maintaining certain smoothness constraints and thereby offsetting the effects of noise.

7.3 3D Reconstruction from Multiple Range Maps

Level-set models are useful for problems related to 3D reconstruction. Previous work has presented level-set results derived from noisy 3D data such as MRI [19] and ultrasound [41]. In [42] we have shown how the reconstruction of objects from multiple range maps can be formulated as a problem of finding the surface that optimizes the posterior probability given a set of measurements (noisy range maps) and some information about the a-priori probability of different kinds of surfaces. That optimization problem can be expressed as a volume integral which can be solved with level-set models. This section presents the mathematical expressions that result from those formulations and presents some new results: the reconstruction of entire scenes by fitting level-set models to the data from a scanning LADAR (laser ranging and detection) system.

A *range map* is a collection of range measurements taken along different directions (lines of sight) but from a single point of view. Range maps could come from any number of different sources including laser scanners, structured light depth systems, shape from stereo, or shape from motion. We assume that such range maps are noisy and uncertain. The goal is to combine a number of range maps from different points of view to create a 3D structure that reflects the collective confidence and depth measures.

Several examples in the literature have applied parametric models to this task. Turk and Levoy [43], for instance, "zip" together triangle meshes in order to construct 3D objects from sequences of range maps from a laser range finder. They perform minor adjustments to the surface position in order account for ambiguity in the range maps. Their approach assumes very little noise in the input, which is reasonable given the high quality of their range maps. Chen and Medioni [44] use a parametric (triangle mesh) model which expands inside a sequence of range maps.

[1] The sign of curvature is defined by the direction of the normals— in this work normals point into the volume enclosed by the object.

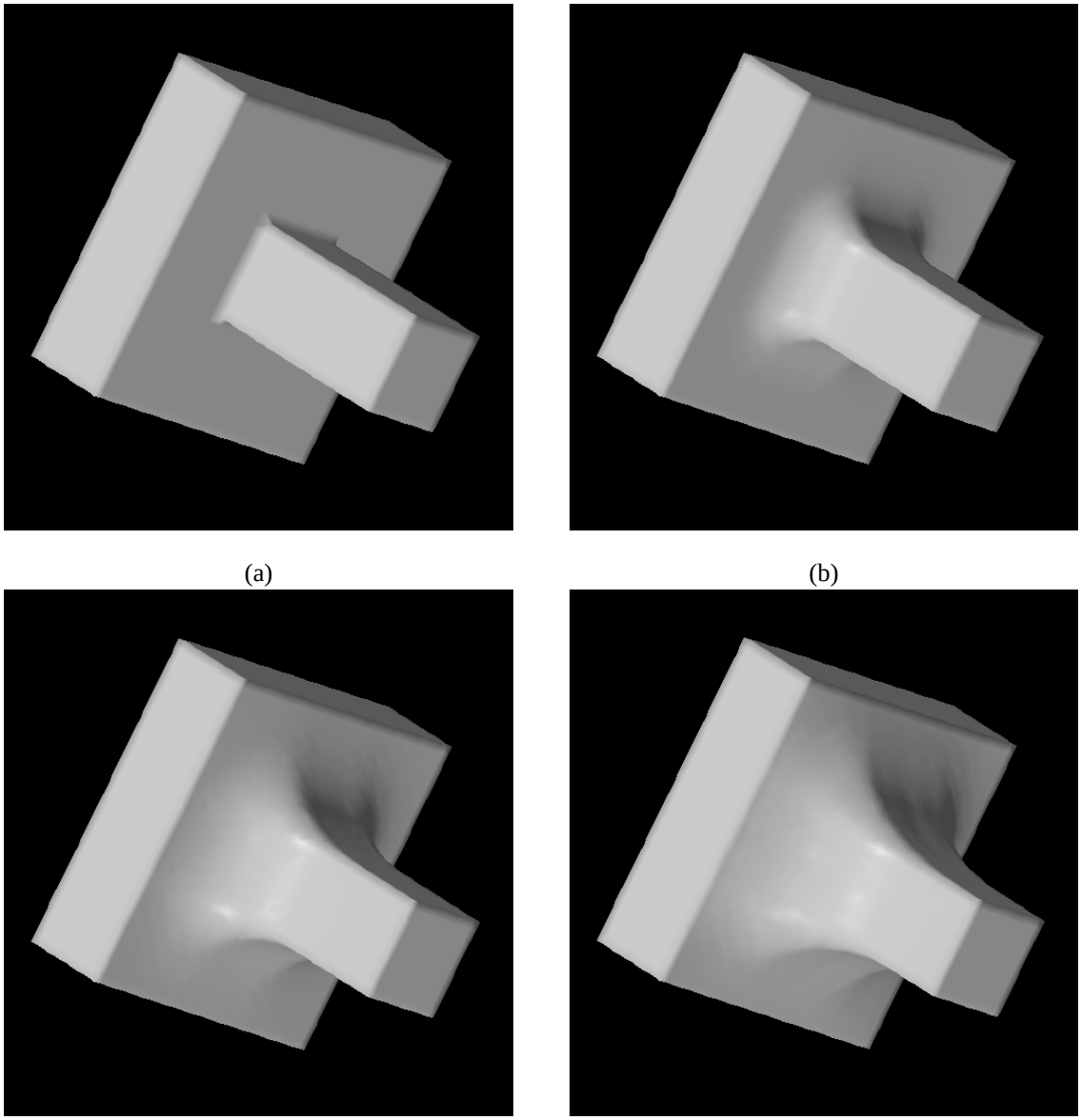


Figure 10: Two rectangular solid models are joined by a volumetric fillet that is created from a positive curvature flow.

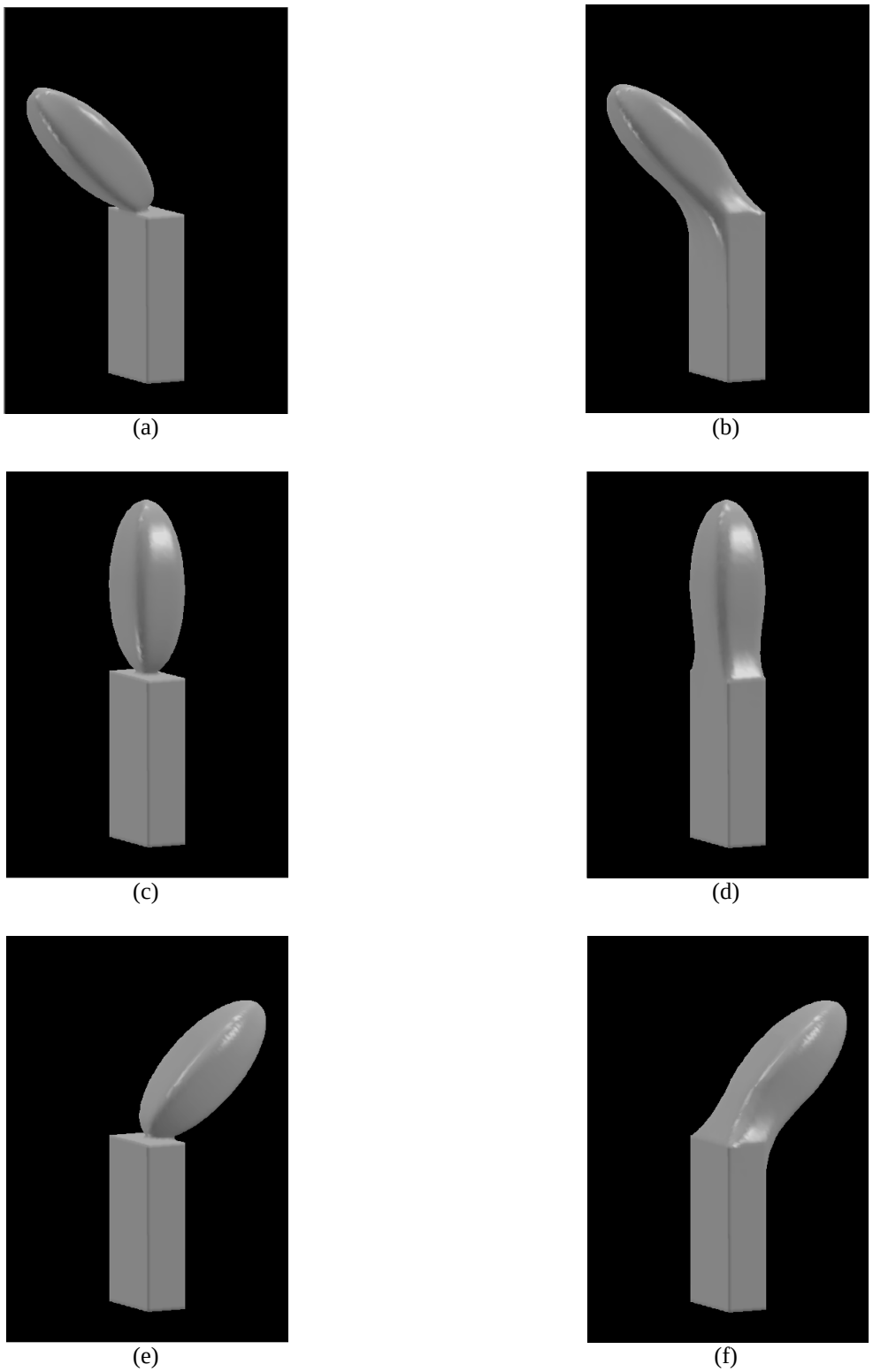


Figure 11: A short animation is created by specifying the relative motion between two superquadric components of an object. A positive-curvature flow (applied frame by frame to the joint between the two 3D models) creates a smooth, flexible object.

Curless and Levoy [45] describe a volume-based technique for combining range data. They use the signed distance transform to encode volume elements with data that represent the averages (with some allowance for outliers) of multiple measurements. Surfaces of objects are the level sets of volumes. Related approaches are given in [46, 47]. Bajaj et. al. [48] use a Delaunay triangulation to impose a topology on a set of unordered 3D points and then fit trivariate Bernstein-Bezier patches—i.e. a higher-order implicit model—to the data. Muraki [2] uses implicit or blobby models to reconstruct objects from range data. The individual blobs are spherically symmetric 3D potentials that are combined linearly so that they blend together. The resulting models, with approximately 400 primitives are quite coarse.

This work differs from previous work in two ways. First, rather than heuristics, our reconstruction strategy is based on a strategy that solves for the optimal surface estimate. This optimal estimate includes information about one's expectations of the likelihood of different surfaces. The result is not a closed-form solution, but an iterative process that seeks to fit a level-set model to the data while enforcing a kind of smoothness on the data.

7.3.1 Objective function for multiple range maps

The evolution equation for the estimation of optimal surfaces is shown in [42] to consist of two parts:

$$\frac{\partial \boldsymbol{x}}{\partial t} = -G(\boldsymbol{x})\boldsymbol{N} + \rho(\mathcal{S}). \tag{37}$$

This first part, $-G(\boldsymbol{x})\boldsymbol{N}$, is the data term, which is a movement with variable speed (as in expression 2 from Table 1) that is the cumulative effect from all of the individual range maps. The second part is the prior, which describes the likelihood of the surface independent of the data. The data term is

$$G(\boldsymbol{x}) = \sum_j c^{(j)}(\boldsymbol{x})\; D^{(j)}(\boldsymbol{x})\; \omega\left(D^{(i)}(\boldsymbol{x})\right)\; \gamma^{(j)}(\boldsymbol{x}), \tag{38}$$

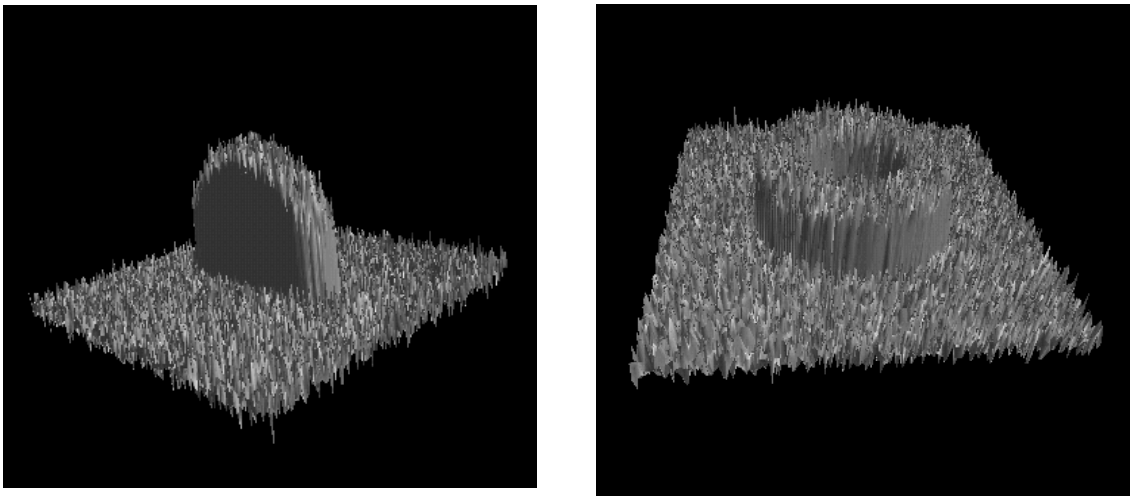
where D_j is the signed distance along the line of sight from a range measurement in range map j associated passing through $\boldsymbol{x}$. The function $\omega : \mathbb{R} \mapsto \mathbb{R}$ is a windowing function that limits the penalty of any one range measurement, and $c(\cdot)$ is a confidence function, which is inversely proportional to the level of noise in the range measurement associated with the same line of sight. The term $\gamma(\cdot)$ is an integration constant that takes into account the curvilinear coordinate system of the range scanner.

Thus, a set of range maps creates a scalar function of 3D, which describes the movement of a surface model as it seeks the optimal surface position. In the absence of a prior, $\rho = 0$, the zero set of this function is the final position (steady state) of that evolving surface. Thus, in the absence of a prior, one could sample $g(\boldsymbol{x})$ and obtain an approximation to the optimal surface estimate. This strategy results in an algorithm that is very much like that of [45].

There are several reasons for going to an iterative scheme for finding optimal solutions. First is the use of a prior. In surface reconstruction, even a very low level of noise can degrade the quality of the rendered surfaces in the final result, and in such cases better reconstructions can be obtained by introducing a prior. Second is aliasing. Discretizing $g(\boldsymbol{x})$ and finding the zero crossings will cause aliasing in those places where the transition from positive to negative is particularly steep. A deformable model can place the surface much more precisely. The third reason for going to an iterative scheme is that despite the windowing function $\omega(\boldsymbol{x})$ there is interference between different range maps at places of high curvature. This problem is addressed by introducing a nonlinearity which is solved in an iterative scheme given by equation 37. In the work described in [21], the solution of the linear problem, the zero set of $g(\boldsymbol{x})$, serves as the initial estimate for the nonlinear, iterative optimization strategy that results from the inclusion of a prior and a nonlinear term that compensates for lack of any explicit model of self occlusions.

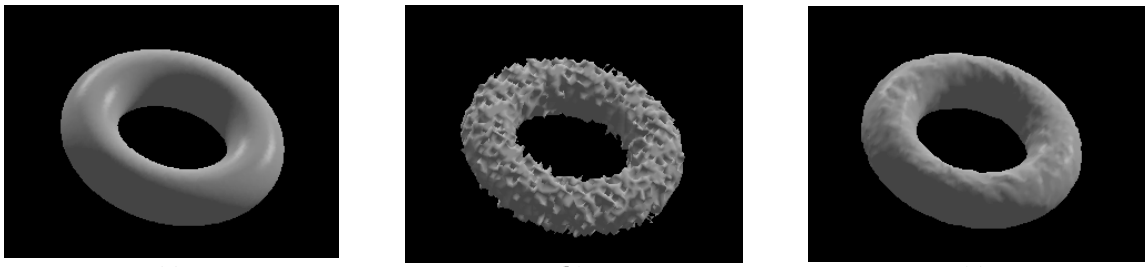
Equation 37 includes a *prior*, which is a likelihood function on surface shape. A reasonable choice of prior is one that models objects with less surface area as more likely than objects with more surface area. Alternatively, one could say that given a set of surfaces that are near the data, the algorithm should choose a surface that has less area. Often, but not always, this will be the smoother surface. The $\rho(\mathcal{S})$ that results from this prior is the mean curvature. Therefore the evolution of the surface, using the level-set formulation, that seeks to maximize the posterior probability (given a set of range maps and a prior that penalizes surface area) is

$$\frac{\partial \Phi(\boldsymbol{x},t)}{\partial t} = |\nabla\Phi(\boldsymbol{x})| \sum_j \left(D^{(j)}(\boldsymbol{x})\; \omega\left(D^{(i)}(\boldsymbol{x})\right) \times\; \gamma^{(j)}(\boldsymbol{x})\; c^{(j)}(\boldsymbol{x}) \frac{\left(\nabla\Phi\cdot\boldsymbol{n}^{(j)}(\boldsymbol{x})\right)^{+}}{\nabla\Phi\cdot\boldsymbol{n}^{(j)}(\boldsymbol{x})} \right) + \beta H, \tag{39}$$



(a) (b)

Figure 12: Range maps: Synthetic range data 200×200 pixels with 20% Gaussian white noise of a torus end (a) and side (b).



(a) (b) (c)

Figure 13: (a) An analytically-defined model of a torus. (b) An initial model (80×80×40 voxels) is constructed by combining six points of view of a torus and solving for $g(\boldsymbol{x}) = 0$. (c) The model, which is attracted to the range data but subject to internal forces, evolves and settles into a smoother steady state.

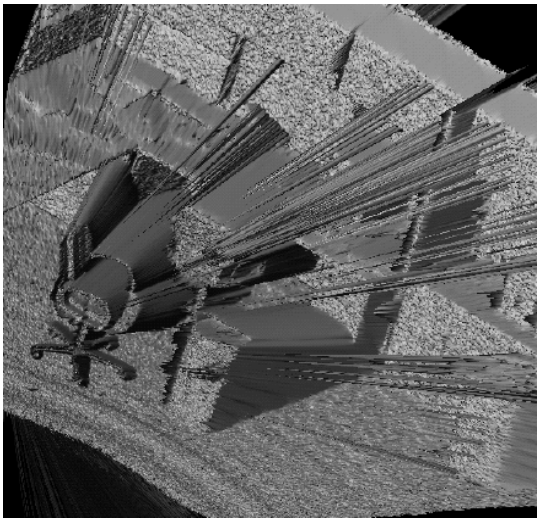
where $\boldsymbol{n}^{(j)}(\boldsymbol{x})$ is the line of sight from a range finder to a 3D point, $\boldsymbol{x}$, β is a free parameter that controls the level of smoothing in the model, and H is the expression for the mean curvature given in equation 6.

Figure 12 shows a pair of simulated range maps constructed from an analytical description of a torus. These 200×200 pixel range maps are corrupted with additive Gaussian noise that has a standard deviation of 20% (as a function of the smaller of the two radii). Six synthetic noise-corrupted viewpoints of a torus are combined to create a level-set reconstruction of a torus. Figure 13(a) shows the initial model (80×80×40 voxels) used for fitting a level-set models to the range data. Figure 13(b) shows the result of the level-set models that uses 13(a) as an initial state and has a value of β equal to 0.5. The result is a reasonable reconstruction of the noiseless model (Figure 13(c)) which combines the six points of view and the smoothing function.

Figure 14(a) shows a range map taken with the Perceptron model P5000, an infra-red, time-of-flight laser range finder with a pan-tilt mechanism. Figure 14(b) shows the amplitudes associated with the return signal (an *intensity*), and 14(c) shows a surface plot of the range map to demonstrate the degree of noise (additive and outliers). Figure 14(d) shows the confidence values associated with those range measurements. These confidence values are derived from empirical data about the level of noise in the range finder (which depends on the return amplitude), and some analysis, from first principles, about the effects of uncertainty in the 3D positions of the scans and the model — which results in the lower confidence at edges as described in [42]. We combined twelve such views from different locations

(a)

(b)



(c)

(d)

Figure 14: (a) One of twelve range maps (b) The associated amplitude map (c) A surface plot of the range data to show the level of noise. (d) The confidence measures associated with those range values.

in the room to generate the results that follow.

Figure 15(a) shows the initial estimate based on the zero crossings of $g(\boldsymbol{x})$, and 15(b) shows the result of 32 iterations with the prior term and the correction for the surface normal direction. The size of the volume is $300 \times 150 \times 180$ voxels, and the resolution is 1.8 cm/voxel. These results show the ability of the statistically-based approach to overcome the noise in the scanner, and they show that the inclusion of iterative, model-fitting scheme helps create more accurate reconstructions. The resolution of the model falls below that of the scans, because it was limited by the random-access-memory available on our workstation. Some small features, such as the arm rests of the chairs, are lost because of the inaccuracies in the registration of the individual range maps.

8 VISPACK

8.1 Introduction

VISPACK is a set of *C++*, object-oriented libraries for image processing, volume processing, and level-set surface modeling. It consists of five libraries: Matrix, Image, Volume, Util, and Voxmodel (level-set modeling). These libraries can be used separately or together when creating applications.

VISPACK incorporates eight basic design attributes. These are

Data Handles/Copy on Write: VISPack is an object-oriented library, and as such we allow the objects to handle memory management, and relieve the programmer (in most cases) from having to worry pointers and the corresponding memory allocation/deallocation problems. For this we use the data handles with a *copy on write protocol*. Copy constructors perform a shallow copy with reference counting until a *non const* operation on the underlying buffers forces a deep copy. Thus deep copies are performed only when necessary, but all memory is maintained by the objects and objects behave as "variables" rather than pointers.

Modified Data Hiding: Access to data in objects is generally through access methods, however, pointers to buffers for fast implementations are available.

Templates: VISPack utilizes the templating construct of C++ virtually throughout. Many of the objects, including images, volumes, lists, and arrays, are intended to support a wide range of data types. Thus, via templating programmers can define the pixels of different images of different types, such as floating point, 24-bit color, and 16-bit greyscale.

Use of Standard File Formats: When appropriate VISPack uses standard file formats. We choose formats that are well known and have publicly available libraries that can be distributed with our libraries. The matrix library uses a simple text format. The image library uses TIFF and FITS file formats. Because no standard format exists for saving volumes of data we do use a *raw* file format.

Operator Overloading: Proper use of operator overloading gives users a convenient way to execute operations on an object. When compined with the copy-on-write convention, operator overloading allows programmers to treat many heavy-weight objects (e.g. images and volumes) as variables. For instance, the following code computes non-maximal edges in a on a filtered volume.

```
Volume<float> dx, dy, dz;
Volume<float> vol_gauss = vol.gauss(0.5);
Volume<float> vol_out = (((dx = vol_gauss.dx()).power(2)
          *vol_gauss.dx(2)
      + ((dy = vol_gauss.dy()).power(2)*vol_gauss.dy(2)
      + ((dz = vol_gauss.dz()).power(2)*vol_gauss.dz(2)
      + dx*dy*(dx).dy() + dx*dz*(dx).dz())
      + dy*dz*(dy).dz()) )).zeroCrossings()
          && ((dx.power(2) + dy.power(2)) > T*T));
```

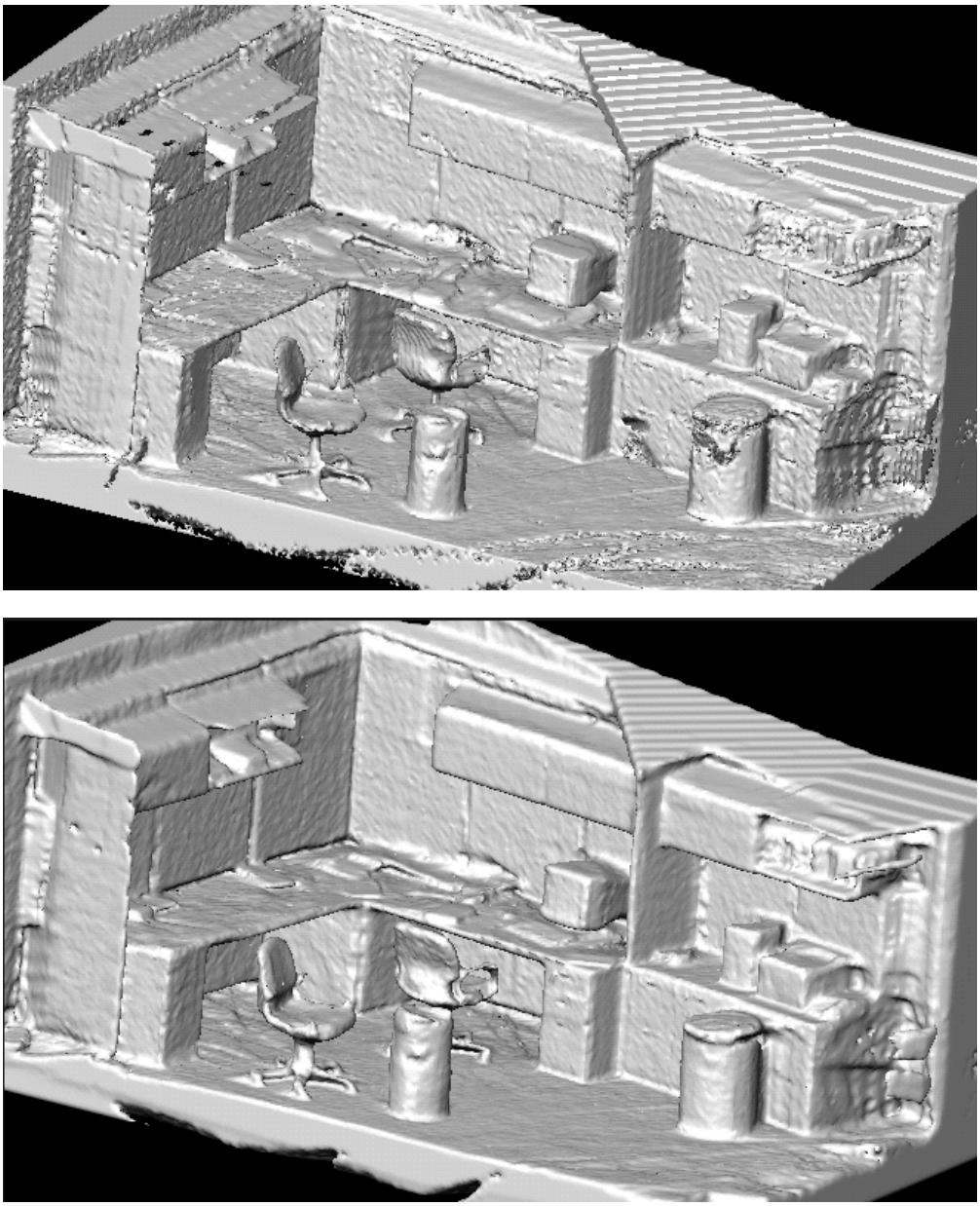


Figure 15: (top) The 3D reconstruction resulting from the zero crossings of $g(\boldsymbol{x})$ gives some averaging, but includes no prior. (bottom) The result of 32 iterations with the iterative scheme includes the prior and excludes influences of data on surfaces that face away from the scanner.

8.2 Level-Set Surface-Modeling Library

The Level-Set Surface-Modeling (LSSM) Library is an implementation of the level-set technique [11, 13] specifically for deforming surface models embedded in volumes. The implementation uses the sparse-field method described in [20]. The library implements all of the basic numerical algorithms and handles all of the data structures required to perform LSSM. The strategy for using this library is to subclass the object `VoxModel`, set some parameters, define a set of simple virtual functions that control the deformation process, initialize the model, and then direct the model to iteratively deform according to those equations. This section describes the relationship between the mathematics of previous sections and the VISPack library. Its also presents an example of using VISPack libarary to do 3D shape metamorphosis as described in Section 7.1.

8.2.1 Surface Deformation

The LSSM library allows one to solve for surface deformations, as a function of time, for general level-set surface movements of the form:

$$\frac{\partial \boldsymbol{x}}{\partial t} = \alpha \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{N}(\boldsymbol{x})) + \beta G(\boldsymbol{x}, \boldsymbol{N}(\boldsymbol{x}))\boldsymbol{N}(\boldsymbol{x}) + \gamma \boldsymbol{N}(\boldsymbol{x}) + \eta E\left(k_1(\boldsymbol{x}), k_2(\boldsymbol{x})\right), \tag{40}$$

where $\boldsymbol{x}$ is a point on the surface. This equation is solved by representing the surface as the kth level set of an implicit function $\phi(\boldsymbol{x}, t) : \mathbb{R}^3 \times \mathbb{R}^+ \mapsto \mathbb{R}$. This gives

$$\frac{\partial \phi}{\partial t} = \alpha \boldsymbol{F}(\boldsymbol{x}, \nabla\phi)) \cdot \nabla\phi + \beta G(\boldsymbol{x}, \nabla\phi)|\nabla\phi| + \gamma|\nabla\phi| + \eta E(D\phi, \mathrm{D}^2\phi), \tag{41}$$

where $\mathrm{D}\phi$ and $\mathrm{D}^2\phi$ are collections first and second derivatives of ϕ, respectively. This equation is solved on a discrete grid using an *up-wind* scheme gradient calculations, central differences for the curvature, and forward finite differences in time. The LSSM library uses the *sparse-field* method described in Section 6.3 and in [21].

Thus, the LSSM library offers the following capabilities:

1. Creates an initial model (with associated active set) from a volume.
2. Calculates $\Delta u^n_{i,j,k}$ and Δt using virtual functions (defined by subclasses) that describe $\boldsymbol{F}$ and G, and parameters (values set by the subclass) α, β, γ, and η.
3. Performs an update on the values of $u^n_{i,j,k}$.
4. Maintains the list of active grid points and updates the *layers* around those points in order to maintain a neighborhood from which to calculate subsequent updates.
5. Provides access to the volume that defines $u^n_{i,j,k}$ and the linked list of active grid points.

Given the volume defining $u^n_{i,j,k}$, one can then rely on the functionality of the volume library for subsequent processing, file I/O, or surface extraction.

8.2.2 Structure and Philosophy of the LSSM Library

The library is organized (mostly for ease of development) into a base class, `LevelSetModel`, and a derived class, `VoxModel`. The base class does all of the book keeping associated with the active set and surrounding *layers*, the link lists associated with those sets, and initializing the model. Thus it adds and removes voxels from the active set (and surrounding layers) in response to an update operation. The base class assumes that the subclasses know how to update individual voxels. Applications are built by subclassing `VoxModel` and redefining a small set of virtual functions that control the movement of the model.

The subclass, `VoxModel`, performs update on the grid points in the active set of the form given in Equation 16, using functions $\boldsymbol{F}$ and G and parameters α, β, γ, and η. It also calculates the maximum Δt that ensures stability. Thus a user who wishes to perform a surface deformation using the LSSM library, would create subclass of `VoxModel` and define the appropriate virtual functions and set the parameters to achieve the desired behavior.

8.2.3 The `LevelSetModel` Object

The `LevelSetModel` contains a volume of values, a volume of status flags, five lists (one active list, two inside lists, and two outside lists), and three parameters that determine the origin of the coordinate system form which the model performs its calculations.

There are two constructors, `LevelSetModel()` and `LevelSetModel( const VISVolume<float> &)`. The first simply initializes the data structure, and the second also set the values of the model volume (`_values`) to the input. Once the values have been set, one can create an initial volume from those values by calling `constructLists()`, which can also take a floating-point argument that controls the scaling of the input relative to a local distance transform near the zero set.

The list that keeps track of the active set, called `_active_list`, keeps track of the location of those grid points and a single floating-point value, which stores the change in their values from one iteration to the next.

Another important methods for users of this object is `update(float)`, which changes the grey-scale values of the grid for the active set according to the values stored in `_active_list`, and updates the status of elements on the active list as well as the values and status of nearby layers (2 inside and 2 outside). The floating point argument is the value of Δt from Equation 16, and the return value is the maximum change that occurred on the active set. Finally, the method `iterate()` calls the virtual method `calculate_change`, a virtual function which sets the values of $\Delta u^n_{i,j,k}$ and returns the maximum value of Δt for stability, and then calls `update`. For this object the function `calculate_change` performs some trivial (i.e., useless) operation.

8.2.4 The `VoxModel` Object

The `VoxModel` object is a subclass of `LevelSetModel`, and it add three things to the base class.

1. `calculate_change()` is redefined to implement the surface deformation described in Equation 41.
2. The virtual functions are declared for F (called `force`) and G (called `grow`). These functions are defined to return zero for this object.
3. The parameters that control the relative influence of the various terms are read from file by a routine `load_params`.
4. A method `rescale(float)` is defined, which resamples the volume of grid-point values into a new volume with different resolution and redefines the lists (and thereby the model) in this new volume. This method is for performing coarse-to-fine deformation procedures.

8.3 Example: 3D Shape Metamorphasis

The `Morph` object allows one to construct a sequence of volumes or surface meshes using the 3D shape metamorphasis technique described in Section 7.1, which was first proposed by Whitaker and Breen [20]. This technique relies distance transforms for both the source and target objects and uses a LSSMs to manipulate the shape of the source so that it coincides with the target. The surface deformation that describes this behavior is

$$\frac{\partial \boldsymbol{x}}{\partial t} = \beta G\left(T(\boldsymbol{x})\right) \boldsymbol{N}(\boldsymbol{x}), \tag{42}$$

where $G(\boldsymbol{x})$ is simply the distance transform (or some monotonic function thereof) of the target, and T is a coordinate transformation that aligns the source and target objects. The level-set formulation of this is

$$\frac{\partial \phi(\boldsymbol{x}, t)}{\partial t} = \beta G\left(T(\boldsymbol{x})\right) |\nabla \phi|. \tag{43}$$

The morphing process consists of several steps:

1. Read in distance transforms (in the form of volumes) for both source and target.
2. Initialize the LSSM by fitting it to the zero set of the source distance transform.

3. Update the LSSM according to Equation 43.

4. Save intermediate volumes/surfaces at regular intervals.

The remainder of this section lists the code and comments for three files, morph.h (which declares the `Morph` object), morph.C (which defines the methods) and main.C (which performs all of the I/O and uses the `Morph` object to construct a sequence of shapes.

8.4 Morph.h

```
//
// morph.h
//
//

#ifndef iris_morph_h
#define iris_morph_h

#include "voxmodel/voxmodel.h"
#include "matrix/matrix.h"

#define INIT_STATE 0
#define MORPH_STATE 1
//
// This is the morph object.  It uses all of the machinery of the base
// class to manipulate level sets.  It needs to have an initial volume
// and a final volume (which would typically be the distance transform,
// it might need a 3D transformation, and it needs to redefine the
// virtual function "grow", which takes 6 floats as input, the position
// followed by the normal vectors (all will calculated and passed into
// this method by the base class).  It might also have a state, that
// indicates whether or not it’s been initialized.
//
// Functions not defined here should be defined in "morph.C"
//
class Morph: public VoxModel
{
  protected:
    VISVolume<float> _dist_source;
    VISVolume<float> _dist_target;
    VISMatrix _transform;
//
// This is the function that is used by the base class to manipulate the
level
// set.  You can define it to by anything you want.  For this object, it
will
// return a value from the distance transform of the target.
//
    virtual float grow(float x, float y, float z,
         float nx, float ny, float nz);

// There are two states.  In the first state, the model is trying to fit
// to the input data.  In this way the models starts by looking just like
```

```
// the input data
    int _state;

  public:

    Morph(const Morph& other)
    {
 _dist_target = other._dist_target;
 _initial = other._initial;
 _state = MORPH_STATE;
 _transform = VISVISMatrix(3, 3);
 _transform.identity();
// initialize();
    }

    Morph(VISVolume<float> init, VISVolume<float> d)
 :VoxModel()
    {
 _dist_target = d;
 _initial = init;
 _state = MORPH_STATE;
 _transform = VISVISMatrix(3, 3);
 _transform.identity();
// initialize();
    }

    void initialize();

// for this object I assume that the transform is just a matrix.
// but it could be anything
    void transform(const VISVISMatrix& t)
    { _transform = t;}

    const VISVISMatrix& transform()
    { return(_transform);}

    void distance(const VISVolume<float> d)
    { _dist_target = d;}
    VISVolume<float> distance()
    { return(_dist_target);}

};
#endif
```

8.5 Morph.C

```
#include "morph.h"
#include "util/geometry.h"
#include "util/mathutil.h"


//
// this is the virtual function, that  is the guts of it all.
```

```
//

float Morph::grow(float x, float y, float z,
    float nx, float ny, float nz)
{

// this says you are in the morph state (things have been initialized)
    if (_state == MORPH_STATE)
 {
 float xx, yy, zz;
 VISPoint p(4u);
 p.at(0) = x;
 p.at(1) = y;
 p.at(2) = z;
 p.at(3) = 1;
 VISPoint p_tmp;
// this is where you could put some other transform.
 p_tmp = _transform*p;

 xx = p_tmp.x();
 yy = p_tmp.y();
 zz = p_tmp.z();

// make sure you are not out of the bounds
// of your distance volume.
 if (_dist_target.checkBounds(xx, yy, zz))
// if not, get the distance (use trilinear interpolation).
     return(_dist_target.interp(xx, yy, zz));
 else
     return(0.0f);
    }
else
    {
// if you are still initializing, then move toward the zero set of
// your initial case
 if (_initial.checkBounds(x, y, z))
     return(_initial.interp(x, y, z));
 else
     return(0.0f);
    }
}

// this makes the model look like the input.
#define INIT_ITERATIONS 5
void Morph::initialize()
{
    _values = _initial;
    int state_tmp = _state;
    _state = INIT_STATE;
    construct_lists(DIFFERENCE_FACTOR);
// these couple of iterations are required to make sure that the zero
// sets of the model match the zero sets of the
//
```

```
    for (int i = 0; i < INIT_ITERATIONS; i++)
 {
// limit the dt to 1.0 so that the model settles in to a solution
     update(::min(calculate_change(), 1.0f));
 }
    _state = state_tmp;
}
```

8.6 Main.C

```
#include "vol/volume.h"
#include "vol/volumefile.h"
#include "image/imagefile.h"
#include "morph.h"
#include <string.h>


const int V_HEIGHT = (40);
const int V_WIDTH = (40);
const int V_DEPTH = (40);

#define XY_RADIUS (12)  // this matches the 2.5D data generated in
torus.C
#define T_RADIUS (4)  // this matches the 2.5D data generated in torus.C
#define S_RADIUS (12)  // radius of a sphere

#define B_WIDTH (20.0f)
#define B_HEIGHT (60.0f)
#define B_DEPTH (20.0f)

#define B_CENTER_X (12.0f)
#define B_CENTER_Y (32.0f)
#define B_CENTER_Z (12.0f)

float sphere(unsigned x, unsigned y, unsigned z);
float torus(unsigned x, unsigned y, unsigned z);
float cube(unsigned x, unsigned y, unsigned z);


// This is a program that does the morph.  If you give it two
// arguments, it reads the initial model and the dist trans for the
// final model from the two file names given, otherwise, it makes a
sphere
// and deforms it into a torus

main(int argc, char** argv)
{

VISVolume<float> vol_source, vol_target;
VISVolumeFile vol_file;
int i;
char fname[80];
```

```
vol_source = VISVolume<float>(25,65,25);
vol_source.evaluate(cube);

if (argc > 2)
    {
// read in the sourceing model
 vol_source = VISVolume<float>(vol_file.read_float(argv[1]));
// read in the dist trans of the final model
     vol_target = VISVolume<float>(vol_file.read_float(argv[2]));
    }
else
// make up some volumes
    {
 vol_source = VISVolume<float>(V_WIDTH, V_HEIGHT, V_DEPTH);
 vol_source.evaluate(sphere);
 vol_target = VISVolume<float>(V_WIDTH, V_HEIGHT, V_DEPTH);
 vol_target.evaluate(torus);
    }

// create morph object
Morph morph(vol_source, vol_target);
// loads in some parameters (for morphing these are all zero but one)
// i.e.
//
//
//
//
morph.load_parameters("morph_params");
morph.initialize();
vol_file.write_float(morph.values(), "morph0.flt");

float dt;

// do 150 iterations for your model to get from start to finish
// probably don't need this many iterations

for (i = 0; i < 150; i++)
    {
 dt = morph.calculate_change();
// limit dt to 0.5 so that model never overshoots goal
 dt = min(dt, 0.5f);
 morph.update(dt);

 printf("iteration %d dt %f\n", i, dt);

 if (((i + 1)%10) == 0)
 {
// save every tenth volume
     sprintf(fname, "morph_out.%d.dat", i + 1);
     vol_file.write_float(morph.values(), fname);
 }
    }
```

```
// save a surface model (i.e. marching cubes).
vol_file.march(0.0f, morph.values(), ‘‘morph_final.iv’’);

printf("done\n");

}
```

References

[1] J. Blinn, “A generalization of algebraic surface drawing,” *ACM Trans. on Graphics*, vol. 1, pp. 235–256, March 1982.

[2] S. Muraki, “Volumetric shape description of range data using “blobby model”,” in *SIGGRAPH ’91 Proceedings* (T. W. Sederberg, ed.), pp. 227–235, July 1991.

[3] G. Taubin, “An accurate algorithm for rasterizing algebraic curves and surfaces,” *IEEE Computer Graphics & Applications*, March 1994.

[4] D. Breen, S. Mauch, and R. Whitaker, “3d scan conversion of csg models into distance, closest-point and colour volumes,” in *Volume Graphics* (M. Chen, A. Kaufman, and R. Yagel, eds.), pp. 135–158, London: Springer, 2000.

[5] W. Lorenson and H. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1982.

[6] M. Levoy, “Display of surfaces from volume data,” *IEEE Computer Graphics and Applications*, vol. 9, no. 3, pp. 245–261, 1990.

[7] R. A. Drebin, L. Carpenter, and P. Hanrahan, “Volume rendering,” in *SIGGRAPH ’88 Proceedings*, pp. 65–74, August 1988.

[8] J. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge: Cambridge University Press, second ed., 1999.

[9] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, pp. 321–323, 1987.

[10] D. Terzopoulos and K. Fleischer, “Deformable models,” *The Visual Computer*, vol. 4, pp. 306–331, December 1988.

[11] S. Osher and J. Sethian, “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations,” *Jrnl. of Comp. Phys.*, vol. 79, pp. 12–49, 1988.

[12] J. A. Sethian, “A fast marching level set method for monotonically advancing fronts,” *Proc. Nat. Acad. Sci.*, vol. 93, no. 4, pp. 1591–1595, 1996.

[13] J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Sciences*. Cambridge University Press, 1996.

[14] S. Osher and R. Fedkiw, “Level set methods: An overview and some recent results,” Tech. Rep. 00-08, UCLA Center for Applied Mathematics, Department of Mathematics, University of California, Los Angeles, 2000.

[15] L. Alvarez and J.-M. Morel, “A morphological approach to multiscale analysis: From principles to equations,” in *Geometry-Driven Diffusion in Computer Vision* (B. M. ter Haar Romeny, ed.), pp. 4–21, Kluwer Academic Publishers, 1994.

[16] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," in *Fifth Int. Conf. on Comp. Vision*, pp. 694–699, IEEE, IEEE Computer Society Press, 1995.

[17] B. B. Kimia and S. W. Zucker, "Exploring the shape manifold: the role of conservation laws.," in *Shape in Picture: the mathematical description of shape in greylevel images* (Y.-L. O, A. Toet, H. Heijmans, D. H. Foster, and P. Meer, eds.), Springer-Verlag, 1992.

[18] R. Malladi, J. A. Sethian, and B. C. Vemuri, "Shape modeling with front propagation: A level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 158–175, 1995.

[19] R. T. Whitaker and D. T. Chen, "Embedded active surfaces for volume visualization," in *SPIE Medical Imaging 1994*, (Newport Beach, California), 1994.

[20] R. Whitaker and D. Breen, "Level-set models for the deformation of solid objects," in *The Third International Workshop on Implicit Surfaces*, pp. 19–35, Eurographics, 1998.

[21] R. T. Whitaker, "A level-set approach to 3D reconstruction from range data," *Int. Jrnl. of Comp. Vision*, vol. October, no. 3, pp. 203–231, 1998.

[22] R. T. Whitaker, "Algorithms for implicit deformable models," in *Fifth Intern. Conf. on Comp. Vision*, IEEE, IEEE Computer Society Press, 1995.

[23] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contour models," in *Fifth Int. Conf. on Comp. Vision*, pp. 810–815, IEEE, IEEE Computer Society Press, 1995.

[24] A. Yezzi, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum, "A geometric snake model for segmentation of medical imagery," *IEEE Transactions on Medical Imaging*, vol. 16, pp. 199–209, April 1997.

[25] L. Lorigo, O. Faugeraus, W. Grimson, R. Keriven, and R. Kikinis, "Segmentation of bone in clinical knee MRI using texture-based geodesic active contours," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI '98)* (W. Wells, A. Colchester, and S. Delp, eds.), pp. 1195–1204, October 1998.

[26] S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Jrnl. of Comp. Phys.*, vol. 79, pp. 12–49, 1988.

[27] X. Xue and R. Whitaker, "Variable-conductance, level-set curvature for image denoising," in *IEEE International Conference on Image Processing*, p. To Appear., October 2001.

[28] D. Adalstein and J. A. Sethian, "A fast level set method for propagating interfaces," *Jrnl. of Comp. Phys.*, pp. 269–277, 1995.

[29] D. Breen and R. Whitaker, "A level-set approach to 3D shape metamorphosis," *IEEE Transactions on Visualization and Computer Graphics*, p. To Appear., 2001.

[30] A. Lerios, C. D. Garfinkle, and M. Levoy, "Feature-Based volume metamorphosis," in *SIGGRAPH '95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 449–456, Addison Wesley, Aug. 1995.

[31] J. Kent, W. Carlson, and R. Parent, "Shape transformation for polyhedral objects," in *SIGGRAPH '92 Proceedings*, pp. 47–54, July 1992.

[32] J. Rossignac and A. Kaul, "AGRELS and BIPs: Metamorphosis as a bezier curve in the space of polyhedra," *Computer Graphics Forum (Eurographics '94 Proceedings)*, vol. 13, pp. C–179–C–184, September 1994.

[33] J. F. Hughes, "Scheduled Fourier volume morphing," in *Computer Graphics (SIGGRAPH '92 Proceedings)* (E. Catmull, ed.), vol. 26, pp. 43–46, July 1992.

[34] B. Payne and A. Toga, "Distance field manipulation of surface models," *IEEE Computer Graphics and Applications*, vol. 12, no. 1, pp. 65–71, 1992.

[35] D. Cohen-Or, D. Levin, and A. Solomivici, "Three-dimensional distance field metamorphosis," *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 116–141, 1998.

[36] P. Getto and D. Breen, "An object-oriented architecture for a computer animation system," *The Visual Computer*, vol. 6, pp. 79–92, March 1990.

[37] D. Breen, S. Mauch, and R. Whitaker, "3D scan conversion of CSG models into distance volumes," in *Proceedings of the 1998 Symposium on Volume Visualization*, pp. 7–14, ACM SIGGRAPH, October 1998.

[38] A. Middleditch and K. Sears, "Blend surfaces for set theoretic volume modeling systems," in *SIGGRAPH '85 Proceedings*, pp. 161–170, July 1985.

[39] J. Bloomenthal and K. Shoemake, "Convolution surfaces," in *SIGGRAPH '91 Proceedings* (T. W. Sederberg, ed.), pp. 251–257, July 1991.

[40] M. Desbrun and M.-P. Gascuel, "Animating soft substances with implicit surfaces," in *SIGGRAPH '95 Proceedings*, pp. 287–290, August 1995.

[41] R. T. Whitaker, "Volumetric deformable models: Active blobs," in *Visualization In Biomedical Computing 1994* (R. A. Robb, ed.), (Mayo Clinic, Rochester, Minnesota), pp. 122–134, SPIE, 1994.

[42] R. T. Whitaker, "A level-set approach to 3D reconstruction from range data," *Int. Jrnl. of Comp. Vision*, vol. October, no. 3, pp. 203–231, 1998.

[43] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proc. of SIGGRAPH '94*, pp. 311–318, ACM SIGGRAPH, August 1994.

[44] Y. Chen and G. Médioni, "Fitting a surface to 3-D points using an inflating ballon model," in *Second CAD-Based Vision Workshop* (A. Kak and K. Ikeuchi, eds.), vol. 13, pp. 266–273, IEEE, 1994.

[45] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. of SIGGRAPH '96*, pp. 303–312, ACM SIGGRAPH, August 1996.

[46] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[47] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt, "Reliable surface reconstruction from multiple range images," in *Euro. Conf. on Comp. Vision*, Springer-Verlag, 1996.

[48] C. Bajaj, F. Bernardini, and G. Xu, "Automatic reconstruction of surfaces and scalar fields from 3D scans," in *SIGGRAPH '95 Proceedings*, pp. 109–118, August 1995.

Isosurfaces and Level-Sets for Volume Processing

Ross T. Whitaker

SCI Institute School of Computing

University of Utah

For Consideration

"There are two *kinds of people in this world... Those who like to solve partial differential equations, and those who don't."*

– anonymous

Overview

- Introduction
- Geometry
- Deformable Surfaces
- Level-Set Surface Models
- Numerical Methods
- Applications
- The *VISPack* Library

Introduction

Motivation

- Framework for Volume Processing/Filtering
  - Level-set methods for nonlinear image/volume filtering
  - Denoising & Reconstruction
- Surface-Modeling Technology
  - Many degrees of freedom
  - Topologically flexible
  - Limits resolution rather than shape

Introduction

Isosurfaces

- An Implicit Representation

$$\phi : \begin{array}{ll} U & \mapsto \mathbb{R} \\ x,y,z & k \end{array}$$

- $U \subset \mathbb{R}^3$ — domain of the volume/range of surface model.
- A surface S is a set of ponts

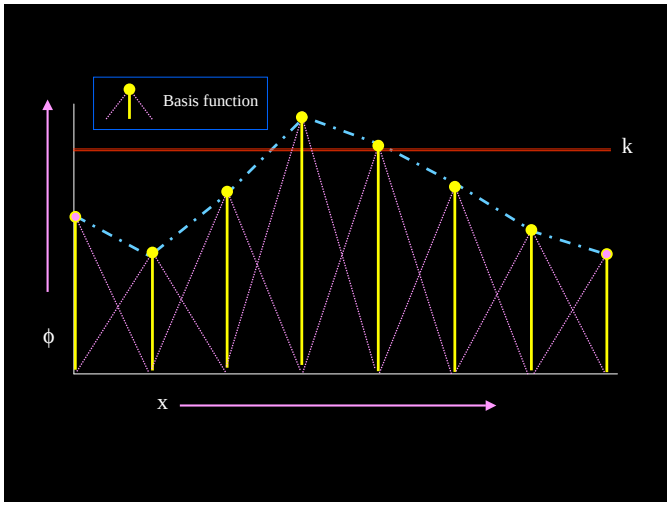
$$S = \{\bar{x} | \phi(\bar{x}) = k\}$$

How Do We Represent ϕ

- Linear combination of global basis functions
  - "Blobby" models — *Blinn 1982*
  - Deformation by modifying size, position, number, etc. — *Muraki 1991*
- Linear combination of local local basis functions
  - Can define local deformations in terms of nearest neighbors
  - Many degrees of freedom
  - Well-defined relationship between surface movements and voxel values

Local Basis Functions

A Volume-Based Representation (1D)



Local Basis Functions

- Geometry defined by *local* operations
- Continuous mathematics on ϕ
- Voxel manipulations determined by well-defined numerical methods

Isosurface Visualization

- Triangulation (mesh extraction)
  - E.g. *marching cubes* (*Lorensen et. al. 87*)
  - Other related approaches
- "Direct" visualization
  - E.g. volume rendering

- Note: not the domain of this talk

The Geometry of Isosurfaces

- Surface normals
- Curvature
- Goal: express the geometry of the isosurface in terms of derivatives of ϕ

Surface Normals

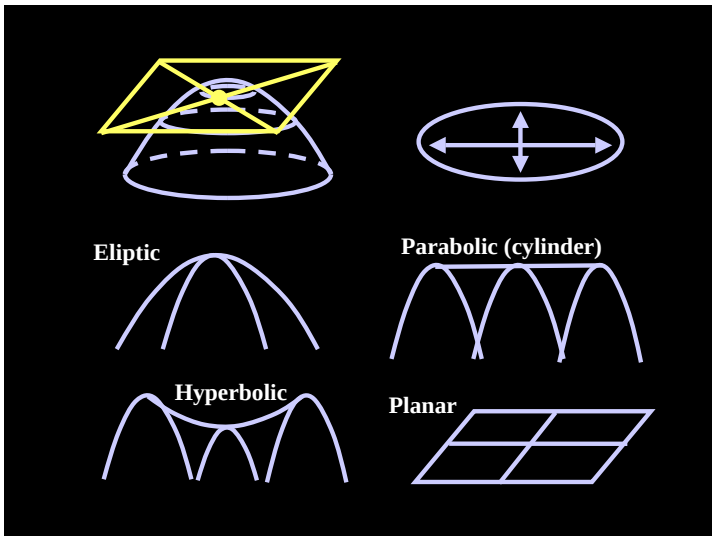
- Exists for every point in D.

$$\bar{n}(\bar{x}) = \frac{\nabla\phi(\bar{x})}{|\nabla\phi(\bar{x})|} \quad \text{where} \quad \bar{x} \in D.$$

- Gives normal to level set passing through that point
- Covention — inside or out (be consistent)
- How to compute? (e.g. central differences)

Second-Order Structure

A Short Course In Differential Geometry



Differential Geometry

Things to Remember

- Surface shape is locally quadratic
- Principal directions — $\bar{e}_1, \bar{e}_2$
- Principal curvatures — k_1, k_1
- Mean curvature — $H = \frac{k_1+k_2}{2}$
- Gaussian curvature — $K = k_1 \times k_2$
- Total curvature — $D^2 = k_1^2 + k_2^2 = 4H^2 - 2K$
- Note: $k_{1,2} = H \pm \sqrt{H^2 - K}$

Second-Order Structure of The Level Sets

- Gradient of the normal $\longrightarrow$ matrix:

$$N = [\bar{n}_x \;\; \bar{n}_y \;\; \bar{n}_z]$$

- Second-order geometry of level set $\longrightarrow$ invariants of N
- Eigenvalues:
  - $e_1 = k_1$
  - $e_2 = k_2$
  - $e_3 = 0$

Second-Order Structure of The Level Sets

Mean Curvature

$$H(\bar{x}) = \frac{k_1 + k_2}{2} = \frac{1}{2}\mathrm{Tr}(\mathrm{N}) = \frac{1}{2}\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$$

$$H(\bar{x}) = \frac{\begin{array}{c}\phi_x^2(\phi_{yy}+\phi_{zz}) + \phi_y^2(\phi_{xx}+\phi_{zz}) + \\ \phi_z^2(\phi_{xx}+\phi_{yy}) - 2\phi_x\phi_y\phi_{xy} - \\ 2\phi_x\phi_z\phi_{xz} - 2\phi_y\phi_z\phi_{yz}\end{array}}{2(\phi_x^2+\phi_y^2+\phi_z^2)^{3/2}}$$

Second-Order Structure of The Level Sets

Total Curvature

$$D^2(\bar{x}) = k_1^2 + k_2^2 = ||N||$$

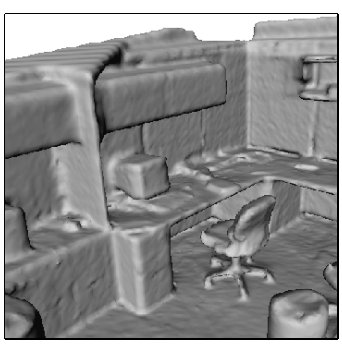
Second-Order Structure of The Level Sets

Gaussian Curvature

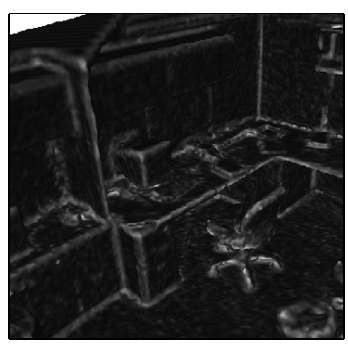
$$K(\bar{x}) = k_1 k_2 = e_1 e_2 + e_1 e_3 + e_2 e_3 = 2H^2(\bar{x}) - \frac{1}{2} D^2(\bar{x})$$

$$K(\bar{x}) = \frac{\begin{array}{c} \phi_z^2(\phi_{xx}\phi_{yy} - \phi_{xy}\phi_{xy}) + \phi_y^2(\phi_{xx}\phi_{zz} - \phi_{xz}\phi_{xz}) \\ +\phi_x^2(\phi_{yy}\phi_{zz} - \phi_{yz}\phi_{yz}) + 2\left[\phi_x\phi_y(\phi_{xz}\phi_{yz} - \phi_{xy}\phi_{zz})\right. \\ \left. + \phi_x\phi_z(\phi_{xy}\phi_{yz} - \phi_{xz}\phi_{yy}) + \phi_y\phi_z(\phi_{xy}\phi_{xz} - \phi_{yz}\phi_{xx})\right] \end{array}}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^2}$$

Example: Total Curvature of An Isosurface



Isosurface of greyscale volume



Total curvature (lighter is greater)

Deformable Models

- Basic terminology/mathematics
- Deformation processes (i.e. types)
- Level-Set Surface Models

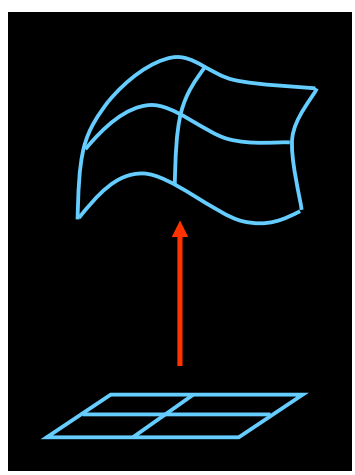
Deformable Models

Regular Surfaces

- Model only local deformation in terms of local surface structure
- Surface — $\mathscr{S} \subset \mathbb{R}^3$
- Local mapping

$$\bar{S} : \underset{r}{V} \times \underset{s}{V} \mapsto \underset{x,y,z}{\mathbb{R}^3},$$

where $V \times V \subset \mathbb{R}^2$



Deformable Models

Regular Surfaces

- Locally parametric and changing over time — $\bar{S} = \bar{S}(r,s,t)$
- Surface normals $\bar{n} = \bar{n}(r,s,t)$
- Parameterization is just a place holder
  - Geometry will be reexpressed without
  - It will be "forgotten" as the model deforms

Deformable Surfaces

- Points on surface, $\bar{x} = \bar{S}(r,x,t)$, move in a prescribed manner
- Depends on position and local surface structure of $\bar{S}$.

$$\frac{\partial \bar{x}}{\partial t} = \bar{G}(\bar{x}, \bar{n}, \bar{S}_r, \bar{S}_s, \bar{S}_{rr}, \bar{S}_{rs}, \bar{S}_{ss}, \ldots),$$

- Note: independence from parameterization $\longrightarrow$ assume (r,s) is an orthnormal parameterization.

Surface Deformation Examples

- Forcing function

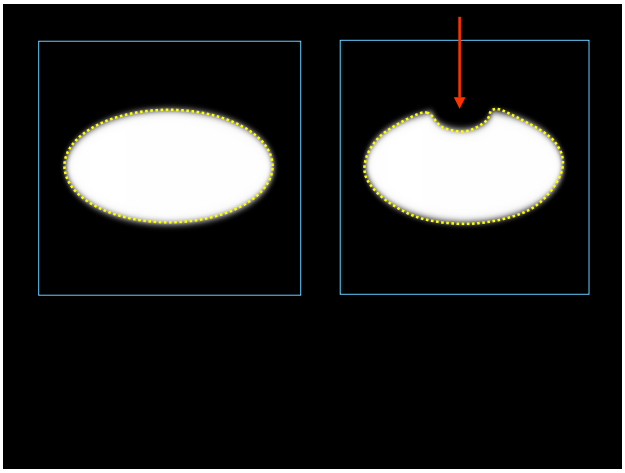
$$\frac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x})$$

- Expand/Contract

$$\frac{\partial \bar{x}}{\partial t} = G(\bar{x})\bar{n}$$

Level-Set Surface Deformation

- Represent a surface/contour implicit
- Change shape by modifying greyscale values



Level-Set Surface Deformation

Two approaches

- Static:

$$\phi(\bar{x}(t)) = k(t) \implies \nabla\phi(\bar{x}) \cdot \frac{\partial \bar{x}}{\partial t} = \frac{\mathrm{d}k(t)}{\mathrm{d}t}$$

  Eikonal Equation

- Dynamic:

$$\phi(\bar{x}(t), t) = k \implies \frac{\partial \phi}{\partial t} = -\nabla\phi \cdot \frac{\mathrm{d}\bar{x}}{\mathrm{d}t}$$

Sethian, 1999

Level-Set Surface Models

Dynamic Approach

- Embed the surface motion in ϕ

$$\frac{\partial \phi}{\partial t} = -\nabla\phi \cdot \frac{\mathrm{d}\bar{x}}{\mathrm{d}t} = -\nabla\phi \cdot \bar{F}(\bar{x}, \mathrm{D}\phi, \mathrm{D}^2\phi, \ldots),$$

  where $D^n\phi$ are nth-order derivatives of ϕ

- Applies to *all* level sets of ϕ

Deformation Modes

	Effect	Parametric Evolution	Level-Set Evolution	Parameter Assumptions
1	External force	$\bar{F}$	$\bar{F} \cdot \nabla\phi$	None
2	Expansion/ contraction	$G(\bar{x})\bar{N}$	$G(\bar{x})\lvert\nabla\phi(\bar{x},t)\rvert$	None
3	Mean curvature	$S_{rr} + S_{ss}$	$H\lvert\nabla\phi\rvert$	Orthonormal
4	Gauss curvature	$S_{rr} \times S_{ss}$	$K\lvert\nabla\phi\rvert$	Orthonormal
5	Second order	S_{rr} or S_{ss}	$\left(H \pm \sqrt{H^2 - K}\right)\lvert\nabla\phi\rvert$	Principal curvatures

Level-Set Surface Deformation Examples

- Forcing function

$$\frac{\partial \bar{x}}{\partial t} = \bar{F}(\bar{x}). \implies \phi_t = -\bar{F} \cdot \nabla\phi$$

- Expand/contract

$$\frac{\partial \bar{x}}{\partial t} = G(\bar{x})\bar{n} \implies \phi_t = G\lvert\nabla\phi\rvert$$

Numerical Methods

- Finite differences
- Up-wind schemes
- Narrow-band and sparse-field methods
- Notation
  - $u^n_{i,j,k}$ — nth time step, i, j, k on grid
  - $\overline{x}_{i,j,k}$ — 3D position of i, j, k grid point
  - $u^n_{i,j,k}$ discrete sampling of $\phi(\overline{x}_{i,j,k}, t_n)$
  - assume w.o.l.g. grid spacing is 1 unit

Finite Differences

- Time — forward differences

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

- Space (to approxmiate $\Delta u^n_{i,j,k}$ — depends
  - Second-order terms — central differences
  - First-order terms — up-wind schemes

First-Order Operators

$$
\begin{aligned}
\delta_x^{(+)} u_{i,j,k} &\triangleq (u_{i+1,j,k} - u_{i,j,k}) \\
\delta_x^{(-)} u_{i,j,k} &\triangleq (u_{i,j,k} - u_{i-1,j,k}) \\
\delta_x u_{i,j,k} &\triangleq \frac{1}{2}(u_{i+1,j,k} - u_{i-1,j,k})
\end{aligned}
$$

Second-Order Operators

Tightest Fitting

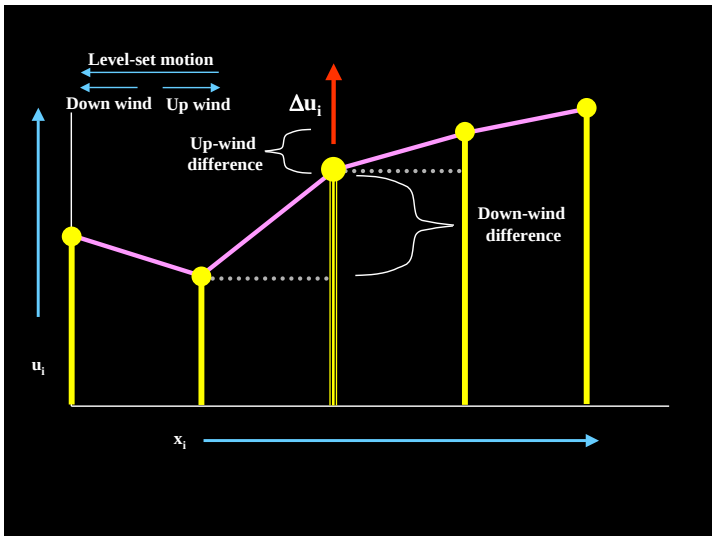
$$
\begin{aligned}
\delta_{xx} u_{i,j,k} &\triangleq \delta_x^{(-)} \delta_x^{(+)} u_{i+1,j,k} = u_{i+1,j,k} + u_{i-1,j,k} - 2u_{i,j,k} \\
\delta_{zz} u_{i,j,k} &\triangleq \delta_z^{(-)} \delta_z^{(+)} u_{i+1,j,k} = u_{i,j,k+1} + u_{i,j,k-1} - 2u_{i,j,k} \\
\delta_{xy} u_{i,j,k} &\triangleq \delta_x \delta_y u_{i,j,k}
\end{aligned}
$$

Up-Wind Scheme

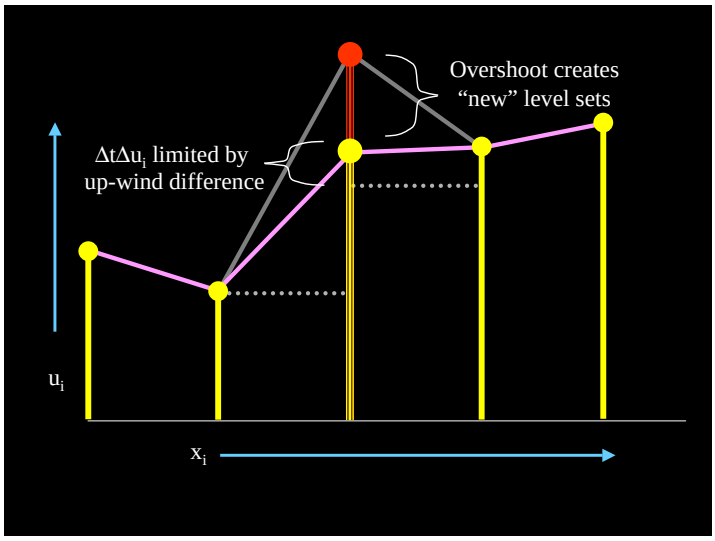
Osher and Sethian, 1988

- "Non-oscillatory" scheme
- Use *one-sided* difference operators in gradient of ϕ
- Avoid overshoot — create no new contours
- Note: choice of derivatives depends on data

Up-Wind Scheme



Up-Wind Scheme



Up-Wind Scheme

- Let

$$\bar{F}(\bar{x}) = (F^{(1)}(\bar{x}), F^{(2)}(\bar{x}), F^{(3)}(\bar{x}))$$

- Then

$$\bar{F}(\bar{x}_{i,j,k}) \cdot \nabla\phi(\bar{x}_{i,j,k}, t) \approx \sum_{q=1}^{2} F^{(q)}(\bar{x}_{i,j,k}) \begin{cases} \delta_q^+ u_{i,j,k}^n & \text{for } F^{(q)}(\bar{x}_{i,j,k}) > 0 \\ \delta_q^- u_{i,j,k}^n & \text{for } F^{(q)}(\bar{x}_{i,j,k}) < 0 \end{cases}$$

Second-Order Terms

Example: Mean Curvature

$$H^n_{i,j,k} = \frac{\begin{array}{c}\left(\left(\delta_y u^n_{i,j,k}\right)^2 + \left(\delta_z u^n_{i,j,k}\right)^2\right)\delta_{xx}u^n_{i,j,k} \\ +\left(\left(\delta_z u^n_{i,j,k}\right)^2 + \left(\delta_x u^n_{i,j,k}\right)^2\right)\delta_{yy}u^n_{i,j,k} + \left(\left(\delta_x u^n_{i,j,k}\right)^2 + \left(\delta_y u^n_{i,j,k}\right)^2\right)\delta_{zz}u^n_{i,j,k} \\ -2\delta_x u^n_{i,j,k}\delta_y u^n_{i,j,k}\delta_{xy}u^n_{i,j,k} - 2\delta_y u^n_{i,j,k}\delta_z u^n_{i,j,k}\delta_{yz}u^n_{i,j,k} - 2\delta_z u^n_{i,j,k}\delta_x u^n_{i,j,k}\delta_{zx}u^n_{i,j,k}\end{array}}{\left(\delta_x u^n_{i,j,k}\right)^2 + \left(\delta_y u^n_{i,j,k}\right)^2 + \left(\delta_z u^n_{i,j,k}\right)^2}$$

Time Steps

Limited For Stability

Up-wind scheme:

$$\Delta t_G \leq \frac{1}{3\sup_{i,j,k\in X}\{|\nabla G(x_i, y_j, z_k)|\}}$$

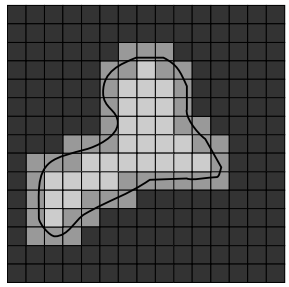
Second-order terms:

$$\Delta t_H \leq \frac{1}{6}$$

Narrow-Band Methods

Evolution of level set, $\phi(\bar{x}, t) = k$, not impacted by choice of embedding

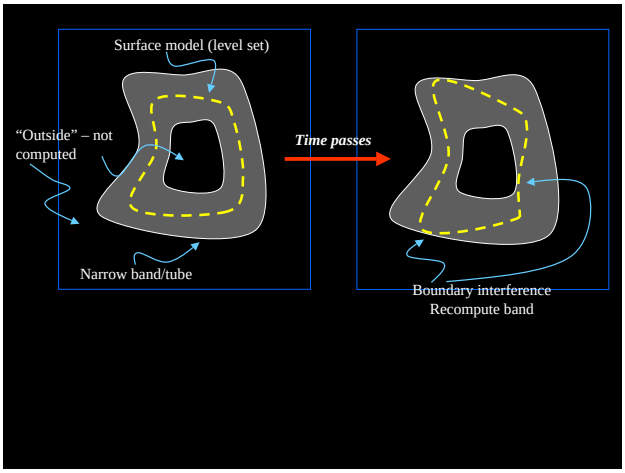
Perform calculations for surface evolution only in a neighborhood of the surface $\mathscr{S} = \{\bar{x} | \phi(\bar{x}) = k\}$



Narrow-Band Algorithm

Adalsteinson and Sethian, 1995

Restablish the neighborhood (embedding) when necessary

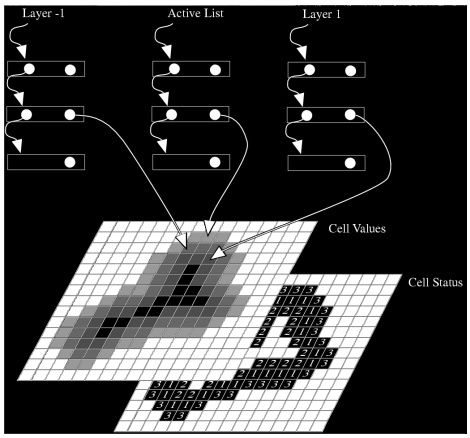


Sparse-Field Algorithm

Whitaker 1998

Maintain a thin band of pixels around an *active set*

Update active set and then consecutive layers



Sparse-Field Algorithm

Behavior

Fast: execution time grows with surface elements

Accurate: active set avoids shocks—sub voxel surface fitting

Fitting of surface model to toriodal data (dist. trans.)

Conventional — Shocks

Sparse-Field

Applications

- Surface morphing
- Filleting
- 3D reconstruction
- Antialiasing binary volumes
- Other

Surface Morphing

Whitaker and Breen, 1998; Breen and Whitaker 2000

Two parts of morphing:

1. Warping — coordinate transformation
2. Blending — "filling in" details ← level-set surface models
   - Fourier domain — *Hughes 1992*
   - Linear interpolation — *Lerios et al. 1995*
   - Distance trans. — *Cohen-Or et al. 1998*

Surface Morphing

Strategy

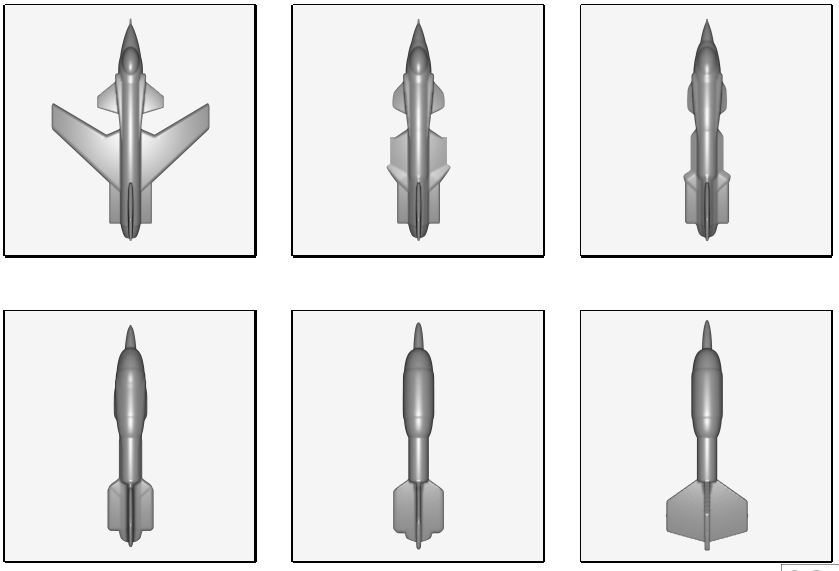
Expansion/contraction — depending on signed-distance transform to "target"

$$\frac{\mathrm{d}\bar{x}}{\mathrm{d}t} = \gamma_B\left(\bar{x}(t)\right)\bar{n}\left(\bar{x}(t)\right)\ \forall\,\bar{x}(t)\ \in \mathscr{S}_t$$

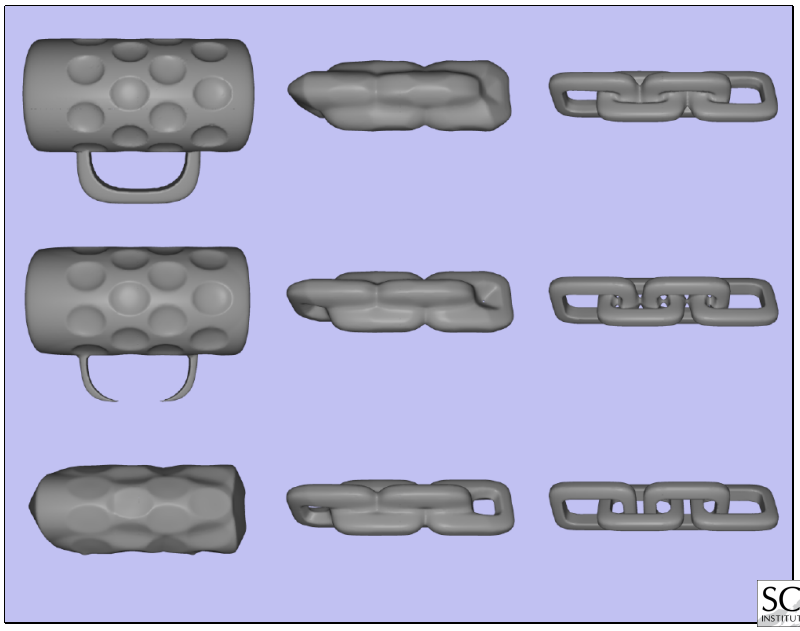
$$\Updownarrow$$

$$\frac{\partial\phi(\bar{x})}{\partial t} = |\nabla\phi(\bar{x})|\gamma_B(\bar{x})$$

Example



Example



Filleting and Blending

Whitaker and Breen, 1998

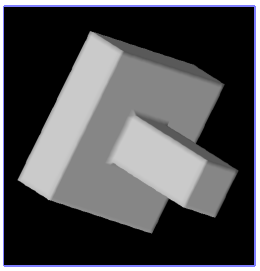
Modify mean curvature flow to “filling in” material

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[\max(k_1, 0) + \max(k_2, 0) \right]$$

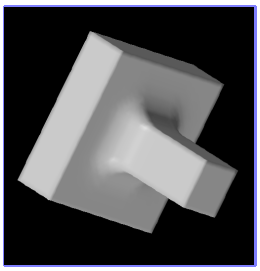
Can also limit to prescibed curvature κ

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[\max(k_1, \kappa) + \max(k_2, \kappa) \right]$$

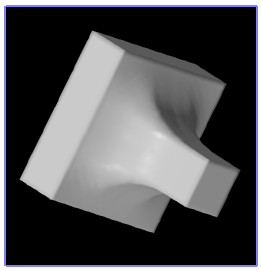
Filleting



Solid Models

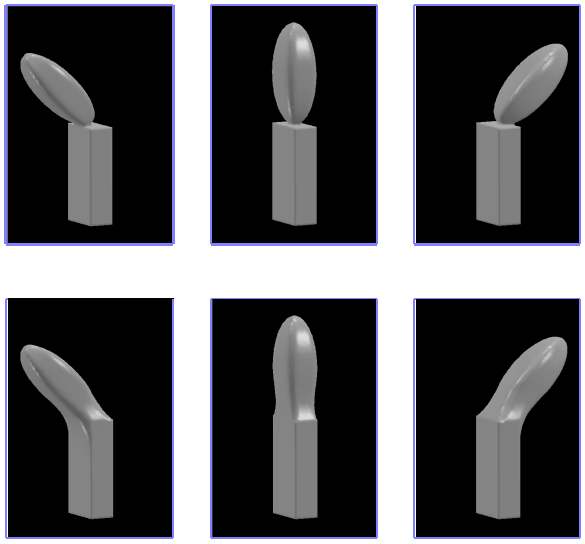


t = 1.0



t = 4.0

Blending



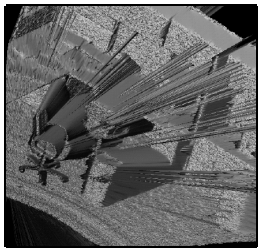
3D Reconstruction From Range Data

Whitaker, 1998

Estimate 3D surfaces from noisy range measurements taken from different points of view

A range image

Amplitude data



Surface rendering

Model Fitting To Range Data

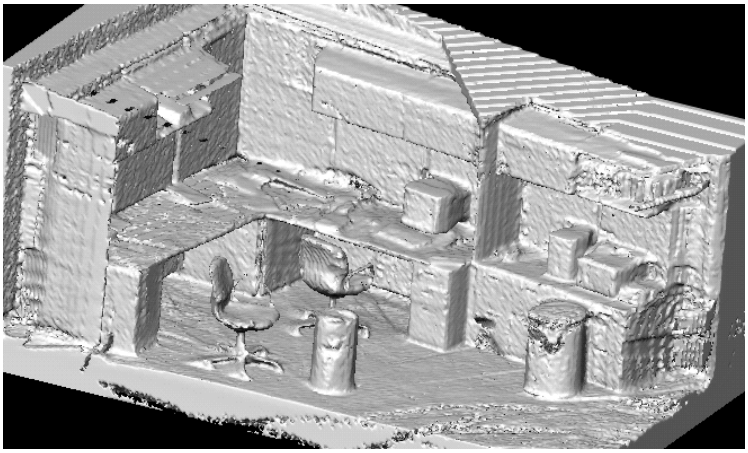
Combine *data term* with prior (surface area penalty)

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| G(\bar{x}) + H(\bar{x})$$

The data term, $G(\bar{x})$, includes inference from a *set* of registered range maps

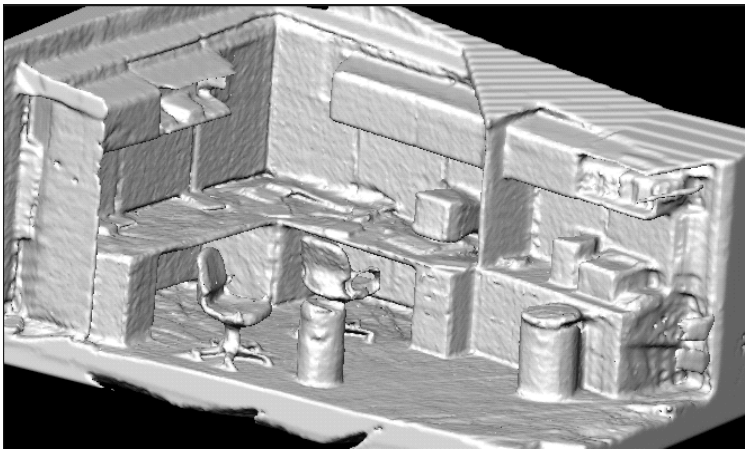
Initial Model

No prior
No deformation/fitting



Final Estimate

Prior
Deformation/fitting



Antialiasing Binary Volumes

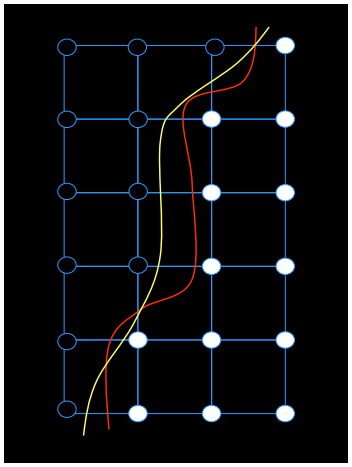
Whitaker, 2000

Treat the binary volume problem as an undercontrained *estimation problem*
Solution must:

- Enclose all of the inside points
- Enclose none of the outside points

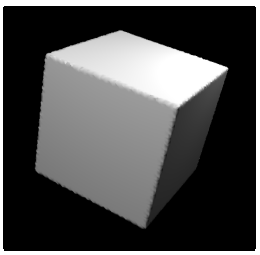
Resolve ambiguity by imposing some other condition

- Find a "preferred" solution
- E.g. minimal surface area

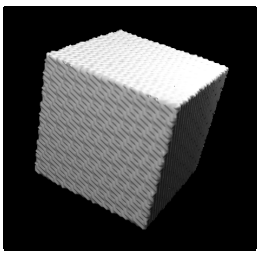


Antialiasing Binary Volumes

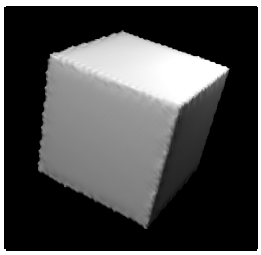
Example



"Ideal" volume



Binary volume



Surface Estimate

Other Applications

- 2D/3D image segmentation *(Whitaker, 95; Casselles et al. 95; Malladi et al., 95; etc.)*
- Image/volume filtering *(Osher & Rudin, 90; Alvarez & Morel, 1994; Malladi & Sethian, 96; etc.)*
- Image blending *(Whitaker, 00)*
- Computer vision *(Kimia & Zucker, 92, 94)*
- Physical simulation *(Osher & Fedkiw, 00)*
- See also: Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 1999

VISPack

Volume-Image-Surface Package

Capabilities

- Image processing
- Volume processing
- Level-set surface modeling

Design

- *C++* — object oriented
- Data handles/copy on write
- Templates
- Functional interface — operator overloading
- File I/O (extensible) — no GUI

Operator Overloading

3D Edge Detector

```
Volume<float> dx, dy, dz;
Volume<float> vol_gauss = vol.gauss(0.5);
Volume<float> vol_out
   = (((dx = vol_gauss.dx()).power(2)*vol_gauss.dx(2)
   + ((dy = vol_gauss.dy()).power(2)*vol_gauss.dy(2)
   + ((dz = vol_gauss.dz()).power(2)*vol_gauss.dz(2)
   + dx*dy*(dx).dy() + dx*dz*(dx).dz())
   + dy*dz*(dy).dz()) )).zeroCrossings()
     && ((dx.power(2) + dy.power(2)) > T*T));
```

Level-Set Surface Modeling

$$\begin{aligned}\frac{\partial \phi}{\partial t} &= \alpha \bar{F}(\bar{x}, \nabla\phi) \cdot \nabla\phi + \beta G(\bar{x}, \nabla\phi)|\nabla\phi| \\ &\quad + \gamma|\nabla\phi| + \eta E(\mathrm{D}\phi, \mathrm{D}^2\phi)\end{aligned}$$

VoxModel Object

1. Creates a model from a volume
2. Calculates $\Delta u^n_{i,j,k}$, Δt, etc.
3. Virtual functions (subclasses) define $\bar{F}$, G
4. Parameters (α, β, γ, η) set by the subclass
5. Performs an update on the values of $u^n_{i,j,k}$
6. Maintains active set and updates the *layers*
7. Provides access to $u^n_{i,j,k}$ and list of active grid points.

Example: Shape Morphing Object

Subclass Of *VoxModel*

1. *Source* and *target* volumes
2. Inialize to *source*
3. Define virtual method:
```
grow(float x, float y, float z,
    float nx, float ny, float nz)
```
4. call `iterate()` and save volumes/models at regular intervals

Conclusions

- Isosurface — representing and *manipulating* solid shapes
- Level set surface models — wide range of applications for processing volumes and surfaces
- Analytical — surface motion $\longrightarrow$ pde's on volume
- Lots of potential
  - Applications
  - New technologies

Thanks

- Office of Naval Research
- National Library of Medicine
- National Science Foundation