

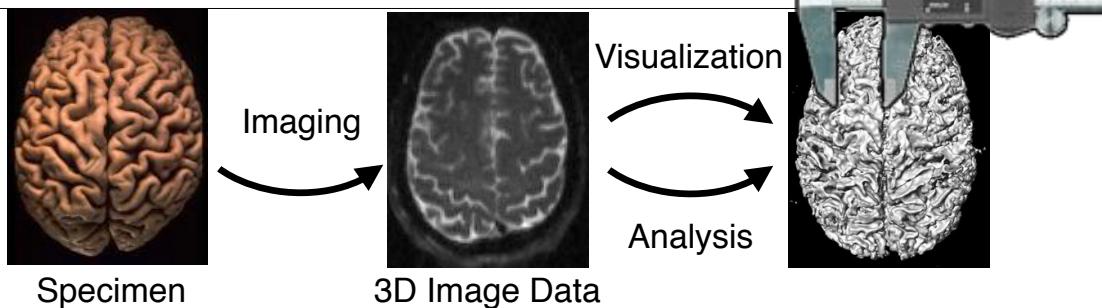
# Discovering Stable Features in Scientific Images (and a bit about Diderot)

Gordon Kindlmann

<http://people.cs.uchicago.edu/~glk/>  
glk@uchicago.edu



## Context



- Goal: geometric models of **meaningful features** for quantitative analysis of large image datasets
  - “Feature” = proxy for structure under scientific study
- Want: **generality** WRT feature co-dimension
- Method: particle systems for feature **sampling**
- But: how do we know if some **mathematical** feature is plausible as an **anatomic** feature?

# Outline

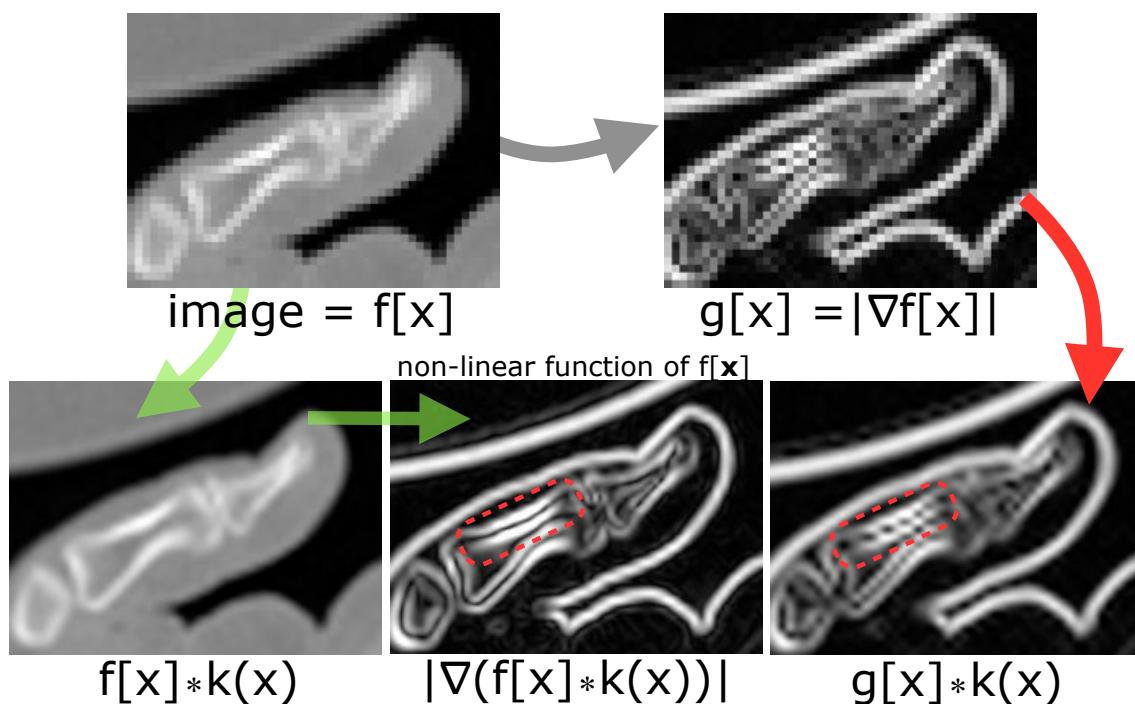
---

- Features
  - Particles
  - Scale-space
  - Stability with respect to scale
  - Results
  - Discussion
- 

By “Image” I mean a continuous field

---

Specimen position arbitrary; want fine structure



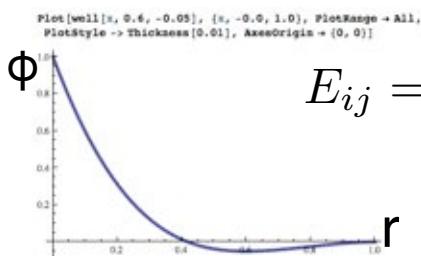
# What I mean by mathematical “feature”

- In a continuous & differentiable field  $f(\mathbf{x})$  created by convolution:
- Feature = positions  $\mathbf{x}$  satisfying feature equation
  - e.g. Isocontours:  $f(\mathbf{x}) = f_0$
  - e.g. Laplacian zero-crossing edges:  $\nabla^2 f = 0$
- (No models, priors, blend of data & smoothing terms..)
- ( $\mathbf{g} = \nabla f$ ;  $\mathbf{H} = \nabla \otimes \nabla f$ ;  $\mathbf{H} = \sum_i \lambda_i \mathbf{e}_i \otimes \mathbf{e}_i$ ;  $\lambda_1 \geq \lambda_2 \geq \lambda_3$ )
- Combinations of maxima and minima WRT  $\mathbf{e}_i$ 
  - Ridge Surface (Eberly '96):  $\mathbf{g} \cdot \mathbf{e}_3 = 0$ ;  $\lambda_3 < 0$
  - Ridge Line:  $\mathbf{g} \cdot \mathbf{e}_3 = \mathbf{g} \cdot \mathbf{e}_2 = 0$ ;  $\lambda_3 < 0$ ;  $\lambda_2 < 0$
  - Iterative update scheme (Newton optimization) to move closer to feature if near it: we can sample features

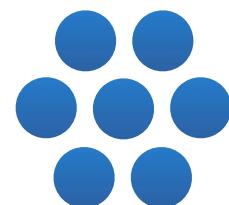
[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Dynamic Particle Systems

- GL Kindlmann, R San Jose Estepar, SM Smith, C-F Westin, **Sampling and Visualizing Creases with Scale-Space Particles**. IEEE Trans. Vis. Comp. Graph, 15(6):1415-1424 (2009)
- Set of points subject to (particle-image) feature constraint and (interparticle) energy minimization
$$\operatorname{argmin}_{\mathbf{x}_i, N} \mathcal{E} = \operatorname{argmin}_{\mathbf{x}_i, N} \sum_{i,j=1}^N E_{ij}$$
- Hard constraint of particles to feature; no energy



$$E_{ij} = \Phi \left( \frac{|\mathbf{x}_i - \mathbf{x}_j|}{\sigma_r} \right)$$

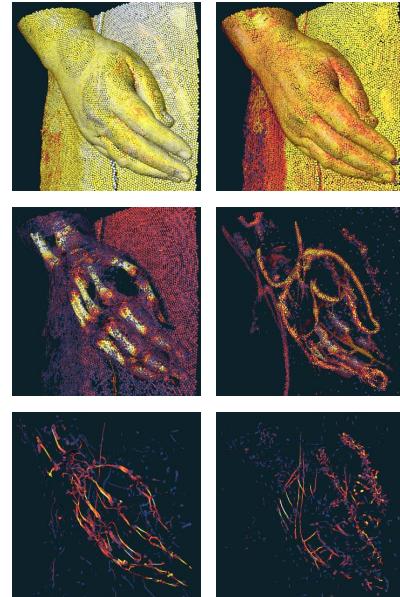


(demo)

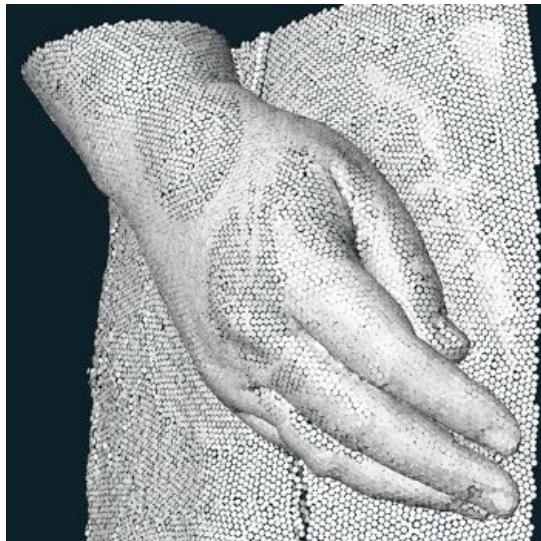
[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Visual survey of feature types

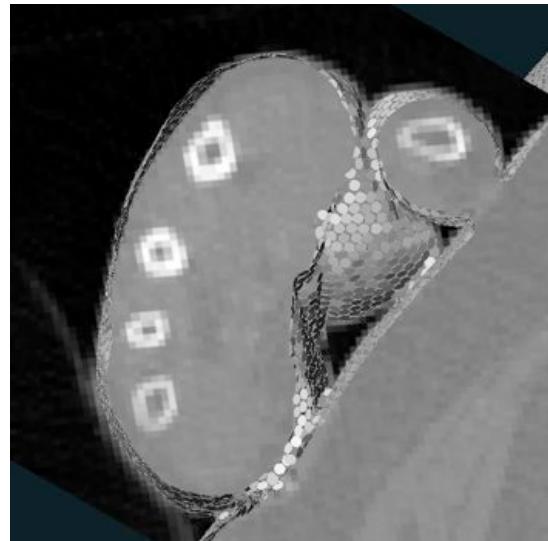
- Illustrated with hand from Visible Human, Female CT
- Feature types
  - Isosurface
  - Laplacian zero-crossing
  - Ridges & Valleys (“creases”)
    - surfaces or lines
- **Same** particles, many features
  - each little glyph = one particle
  - show local feature ingredients
- Can sample features, but
  - Issues: **scale, meaningfulness**



## Isosurface



Outside 3D view

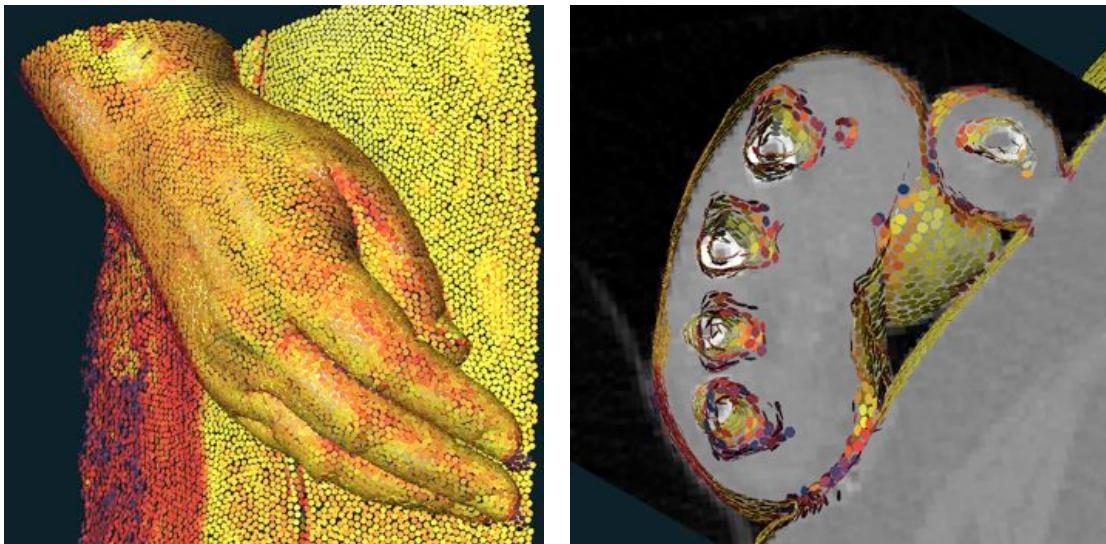


Different, Cropped 3D view w/ 2D plane

- AKA isocline, isophote, isocontour, level set
- $f(\mathbf{x}) = v_0$

# Laplacian 0-crossing

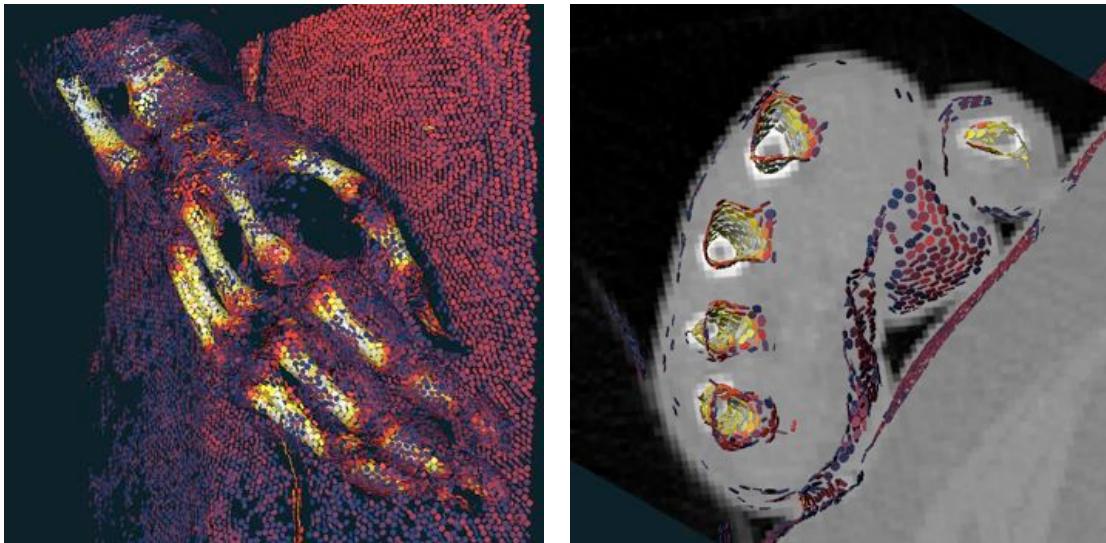
---



- Classical definition of edge
- $\nabla^2 f(\mathbf{x}) = 0$ ; strength =  $|\nabla f(\mathbf{x})|$

# Ridge Surface

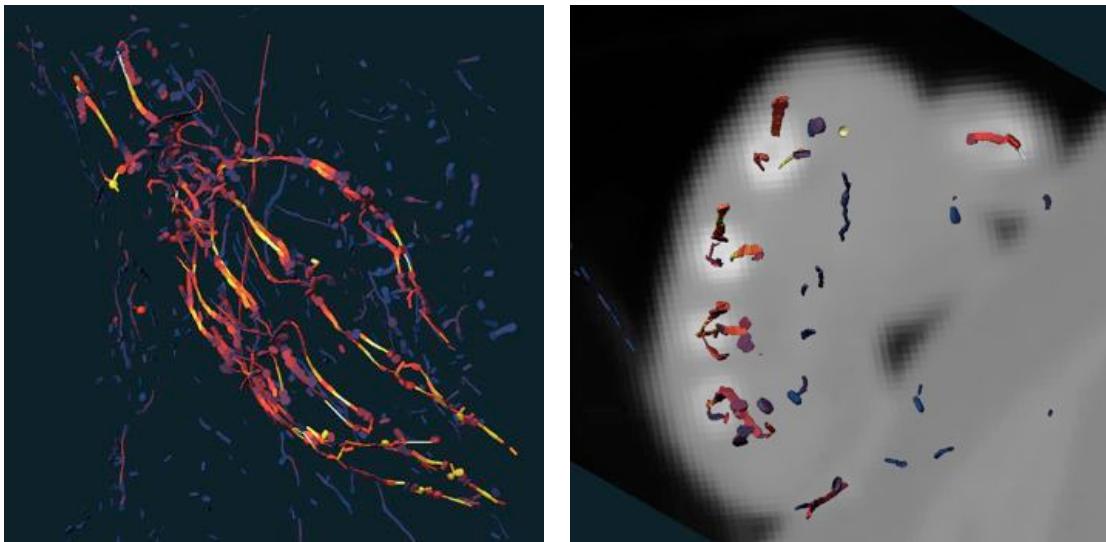
---



- Maximal surface wrt Hessian minor eigenvector  $\mathbf{e}_3$
- $\nabla f(\mathbf{x}) \cdot \mathbf{e}_3(\mathbf{x}) = 0$ ; strength =  $-\lambda_3$

# Ridge Line

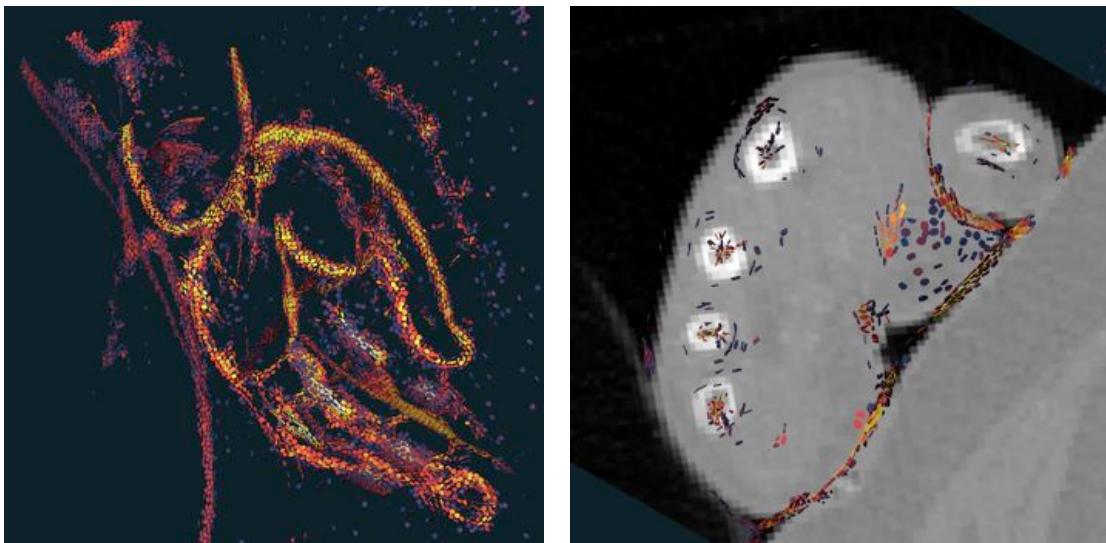
---



- Maximal curve wrt Hessian minor,medium eigenvectors
- $\nabla f(\mathbf{x}) \cdot \mathbf{e}_3(\mathbf{x}) = 0, \nabla f(\mathbf{x}) \cdot \mathbf{e}_2(\mathbf{x}) = 0$ ; strength =  $-\lambda_2$

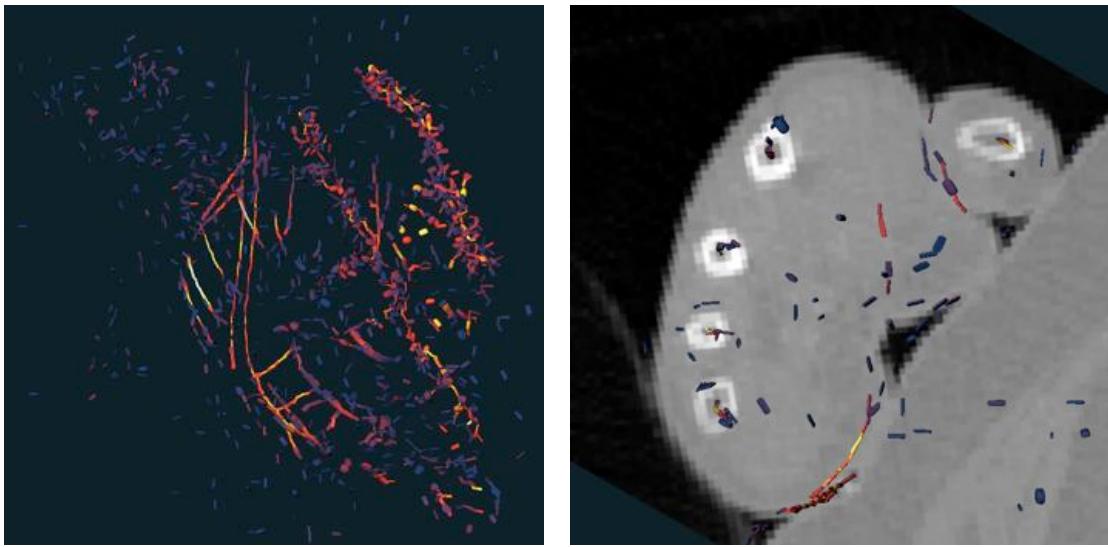
# Valley Surface

---



- Minimal surface wrt Hessian major eigenvector  $\mathbf{e}_1$
- $\nabla f(\mathbf{x}) \cdot \mathbf{e}_1(\mathbf{x}) = 0$ ; strength =  $\lambda_1$

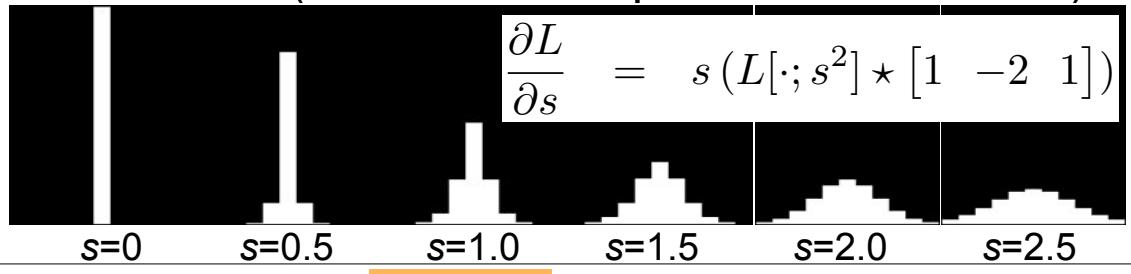
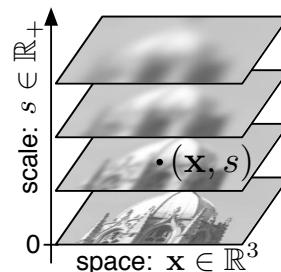
# Valley Line



- Minimal curve wrt Hessian major, medium eigenvectors
- $\nabla f(\mathbf{x}) \cdot \mathbf{e}_1(\mathbf{x}) = 0, \nabla f(\mathbf{x}) \cdot \mathbf{e}_2(\mathbf{x}) = 0$ ; strength =  $\lambda_2$

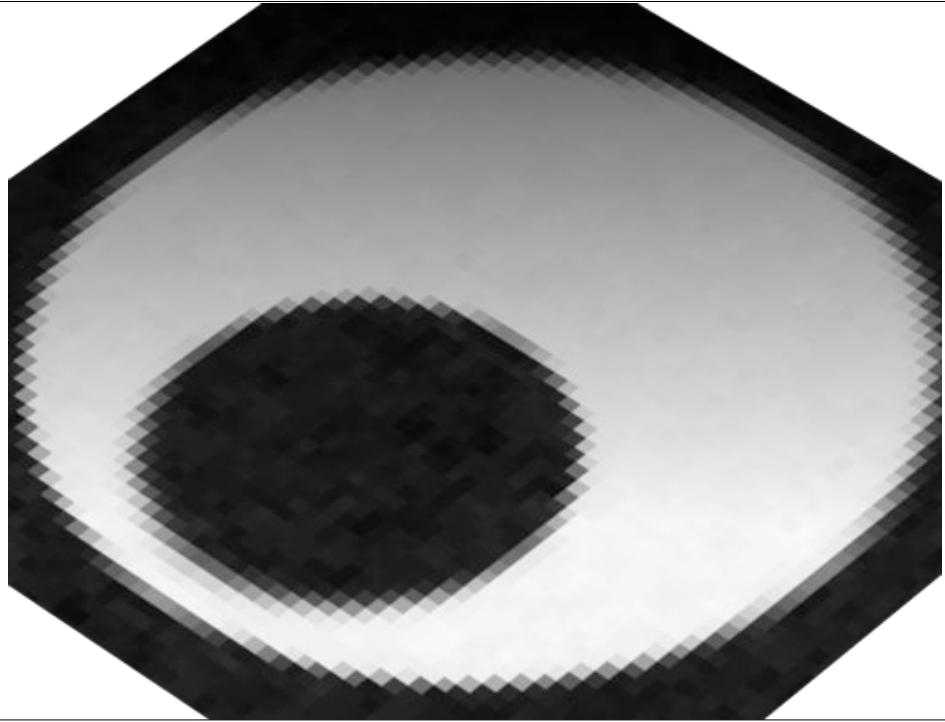
## Scale Space

- Image + continuous family of blurrings
  - feature scale not always known a priori
- Practical requirements
  - Probe at arbitrary points in scale space
  - Efficiently handle real-world 3D datasets
- Scale interpolation based on Lindeberg's "Gaussian" (soln. to heat eq. in discrete domain)



# Sampling scale-space feature

---

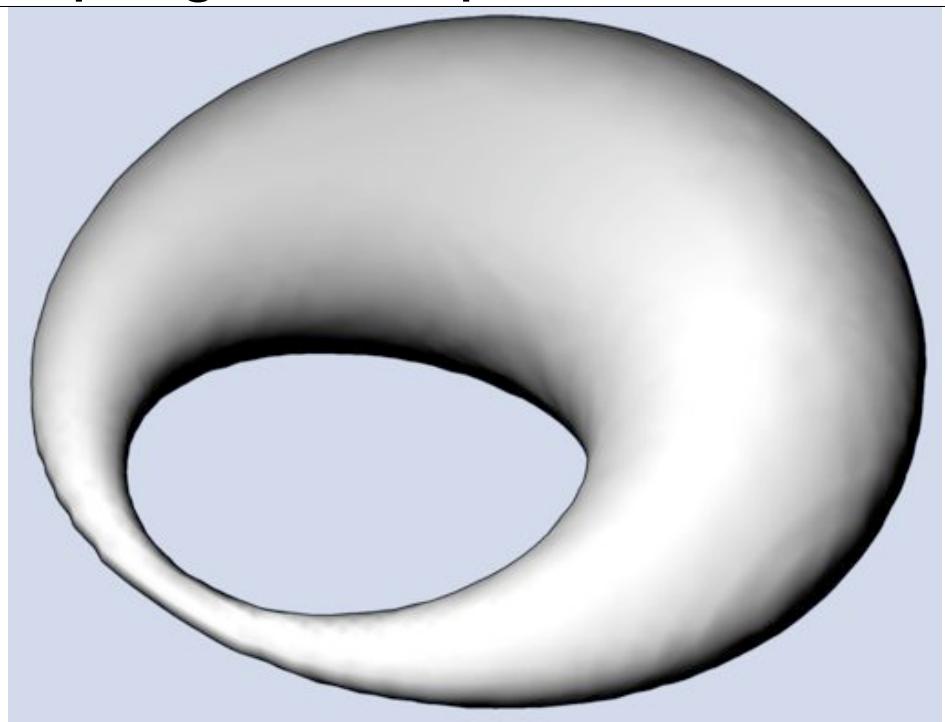


---

[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Sampling scale-space feature

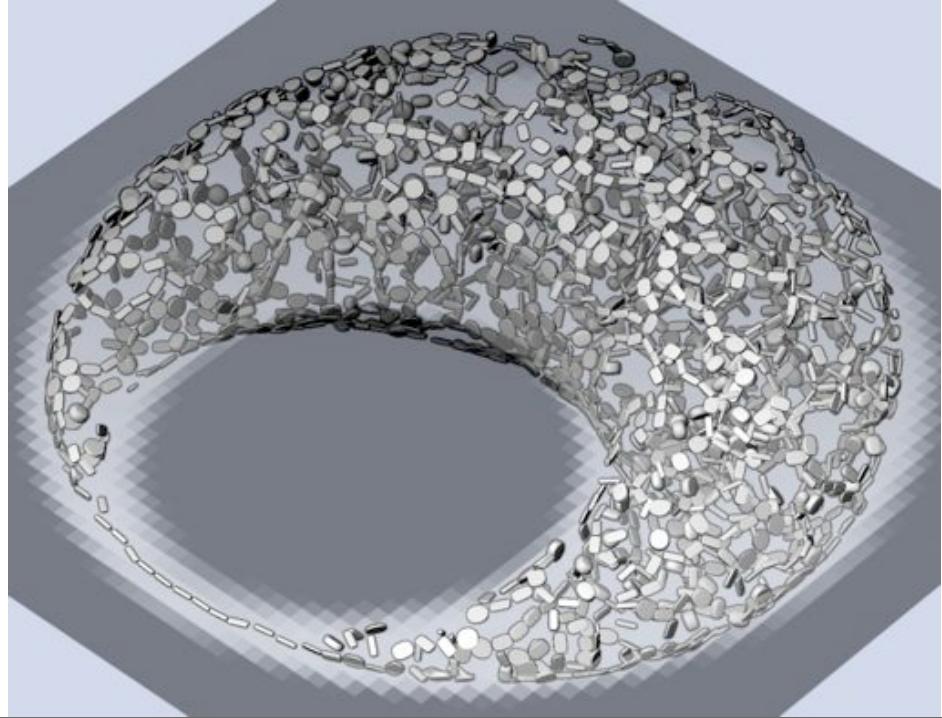
---



---

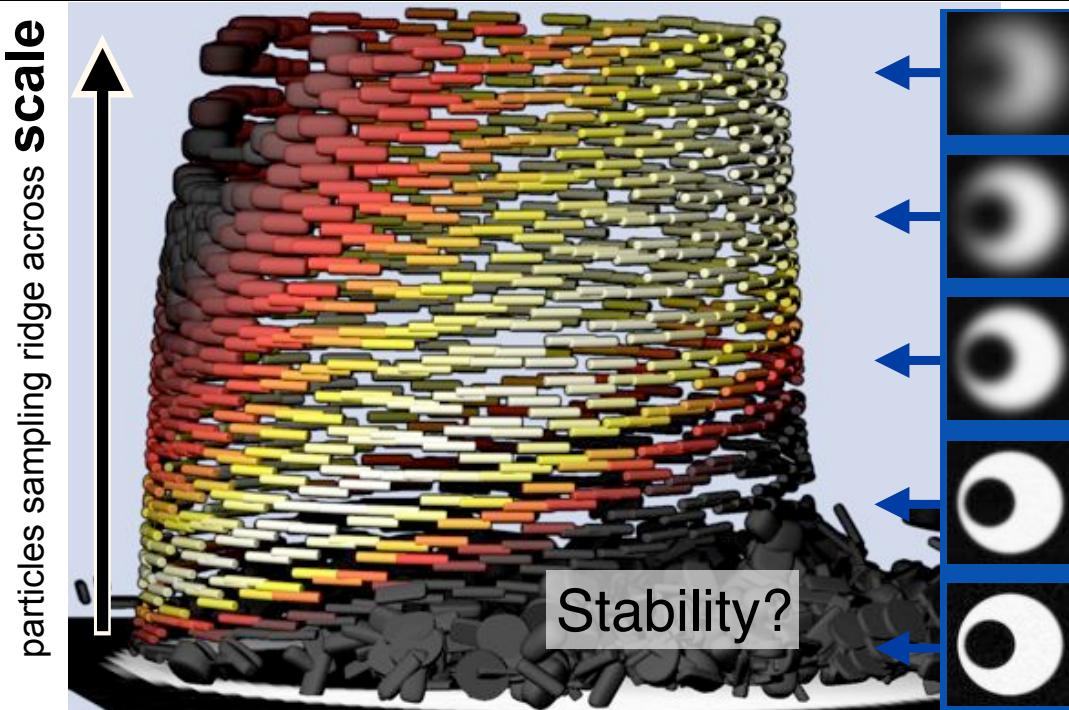
[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Sampling scale-space feature



[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

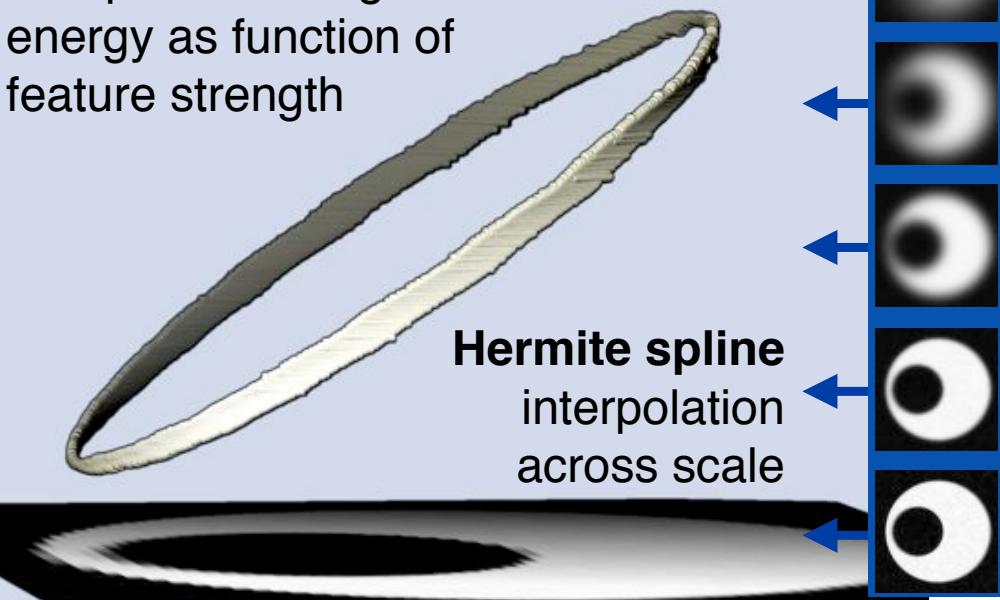
# Sampling scale-space feature



[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Sampling scale-space feature

**Localize in scale**  
with particle-Image  
energy as function of  
feature strength

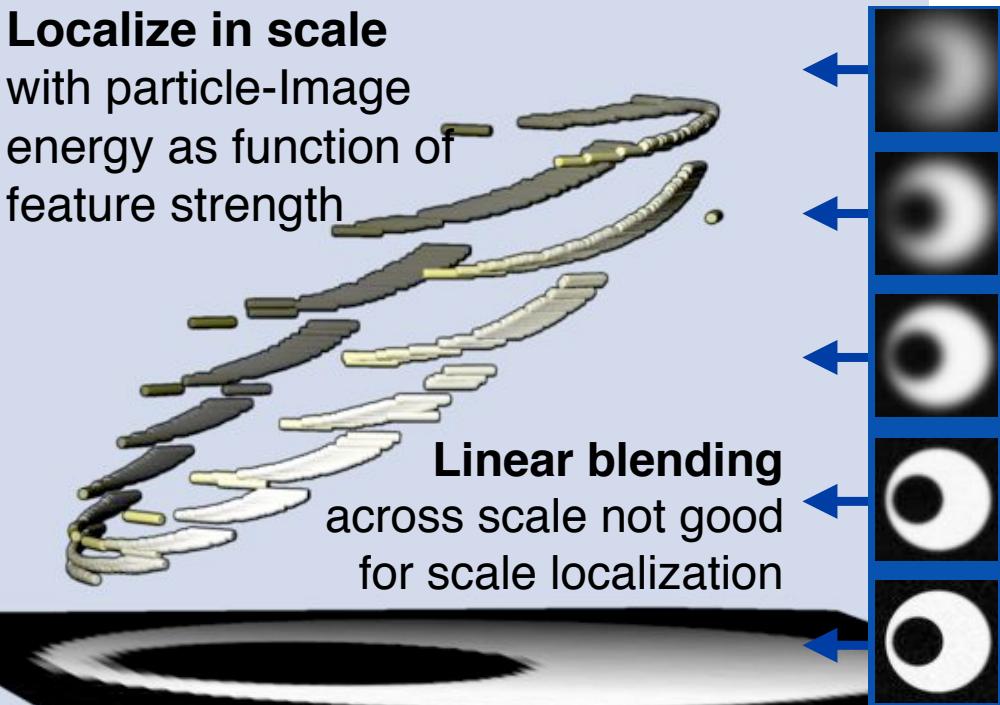


**Hermite spline**  
interpolation  
across scale

[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Sampling scale-space feature

**Localize in scale**  
with particle-Image  
energy as function of  
feature strength



**Linear blending**  
across scale not good  
for scale localization

[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

The purpose is not images, it is  
feature sampling

Glyphs  
displaying  
scale

Feature  
localization and  
sampling in  
space and scale

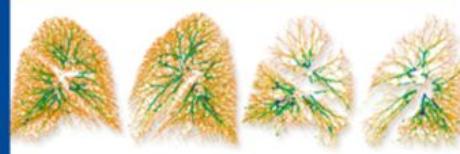
## Applications

### Computed Tomographic Measures of Pulmonary Vascular Morphology in Smokers and Their Clinical Implications



Raúl San José Estépar<sup>1,2,3</sup>, Gregory L. Kinney<sup>4</sup>, Jennifer L. Black-Shinn<sup>4</sup>, Russell P. Bowler<sup>5</sup>, Gordon L. Kindermann<sup>6</sup>, Blaz<sup>7</sup>, Michael H. Cho<sup>7,9</sup>, D. Crapo<sup>12</sup>, J. Michael Wells<sup>11</sup>, and the Lung Health Study

#### AMERICAN JOURNAL OF Respiratory and Critical Care Medicine.



VOLUME 184  
NUMBER 2  
JULY 15 2011  
[www.atsjournals.org](http://www.atsjournals.org)

Division of Pulmonary and Critical Care, and <sup>1</sup>Boston, Massachusetts; <sup>2</sup>Department of Pulmonary and Critical Care Medicine, and <sup>3</sup>Divine, University of Chicago, Chicago, Illinois; <sup>4</sup>Divisions of Medicine, University of Pittsburgh/UPMC, Pittsburgh, Pennsylvania; <sup>5</sup>Department of Pulmonary and Critical Care, University of Colorado, Denver, Colorado; and <sup>12</sup>Division of Pulmonary

**Aim:** Angiographic investigation of pulmonary vascular remodeling in smokers is limited.

**Objective:** Using volumetric computed tomography of the chest we sought to quantitatively assess pulmonary vascular remodeling and its clinical associations.

**Methods:** Pulmonary vessels were identified, segmented, and measured. Total aggregate vessel volume for each lobe was calculated and normalized to the TBV of that lobe and expressed as a percentage ( $\% \text{TV}$ ) to calculate lobe-specific density measures of vessels. A Hounsfield unit threshold of  $-950$

- IN THIS ISSUE**
- ARTICLES**
- Markers of Vascular Perturbation Correlate with Airway Structural Change in Asthma *Marc W. Johnson, et al.*
  - Effects of Allergic Phenotype on Respiratory Symptoms and Exacerbations in Patients with COPD *Daniel B. Johnson, et al.*
  - Nonlinear Exercise Training in Advanced COPD Is Superior to Traditional Exercise Training: A Randomized Trial *Peter Krapf, et al.*
  - Diaphragm Dysfunction on Admission to the ICU: Prevalence, Risk Factors, and Prognostic Impact—A Prospective Study *Alexandre Demoule, et al.*
  - Computed Tomographic Measures of Pulmonary Vascular Morphology in Smokers and Their Clinical Implications *Aad Sem Jint Epelman, et al.*
  - Chronic Intermittent Hypoxia Induces Atherosclerosis via Activation of Adipose Angiotensin-like 4 *Luxem J. Dzau, et al.*

#### EDITORIAL

##### on the Subject

ation suggests that pulmonary vascular remodeling is characterized by distal pruning of vessels. Computed tomographic imaging may allow quantification of this process.

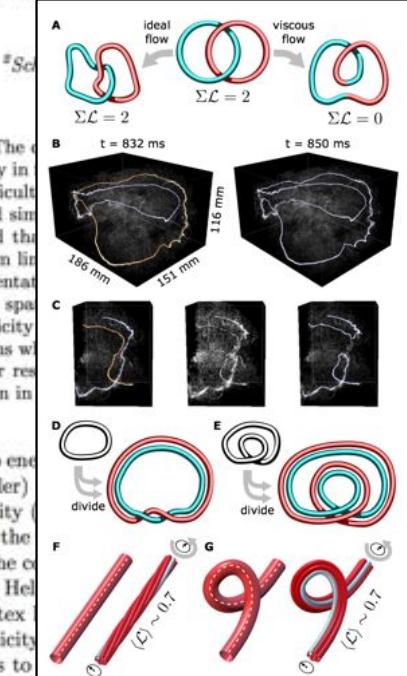
#### to the Field

pulmonary vascular remodeling in smokers may be associated with increased oxygenation of the intraparenchymal blood

# Applications

## Helicity conservation by flow across scales in reconnecting vortex links and knots

Martin W. Scheeler,<sup>1,\*</sup> Dustin Kleckner,<sup>1,\*†</sup> Davide Proment,<sup>2</sup> Gordon L. Kindermann,<sup>3</sup> and William T.M. Irvine<sup>1,‡</sup>



*Department of Physics,  
Urbana, Illinois 60637, USA.  
St Anglia, NR4 7TJ Norwich, Norfolk, UK.  
Department of Computer Science,  
Urbana, Illinois 60637, USA.*

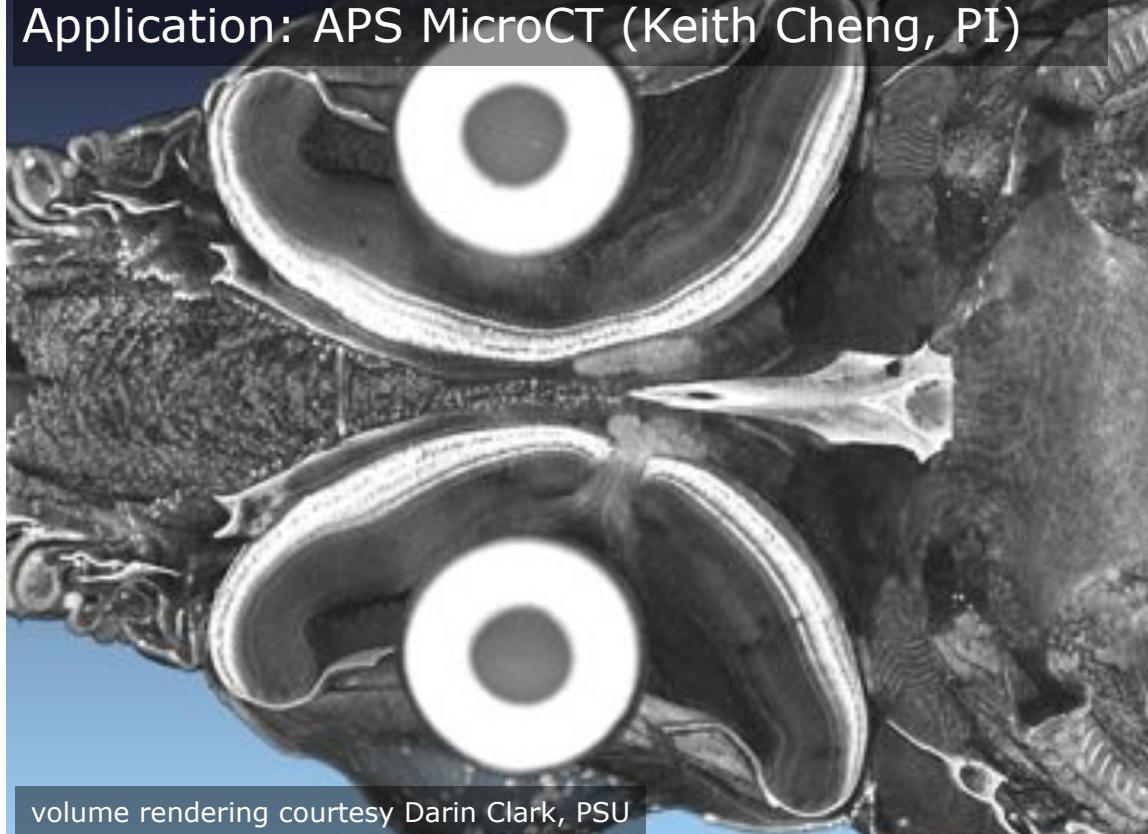
A fundamental conserved quantity has a rich history in the presence of dissipation has proven we create vortex knots and links in viscous fluids through topology changing reconnections. We a reconnection enables the transfer of helicity is remarkably efficient, owing to the anti-parallel vortices. Using a new method for quantifying We also infer the presence of geometric deformations twist, where it may ultimately be dissipated. is an important role in fluids and related fields,

## PNAS 2014 (to appear)

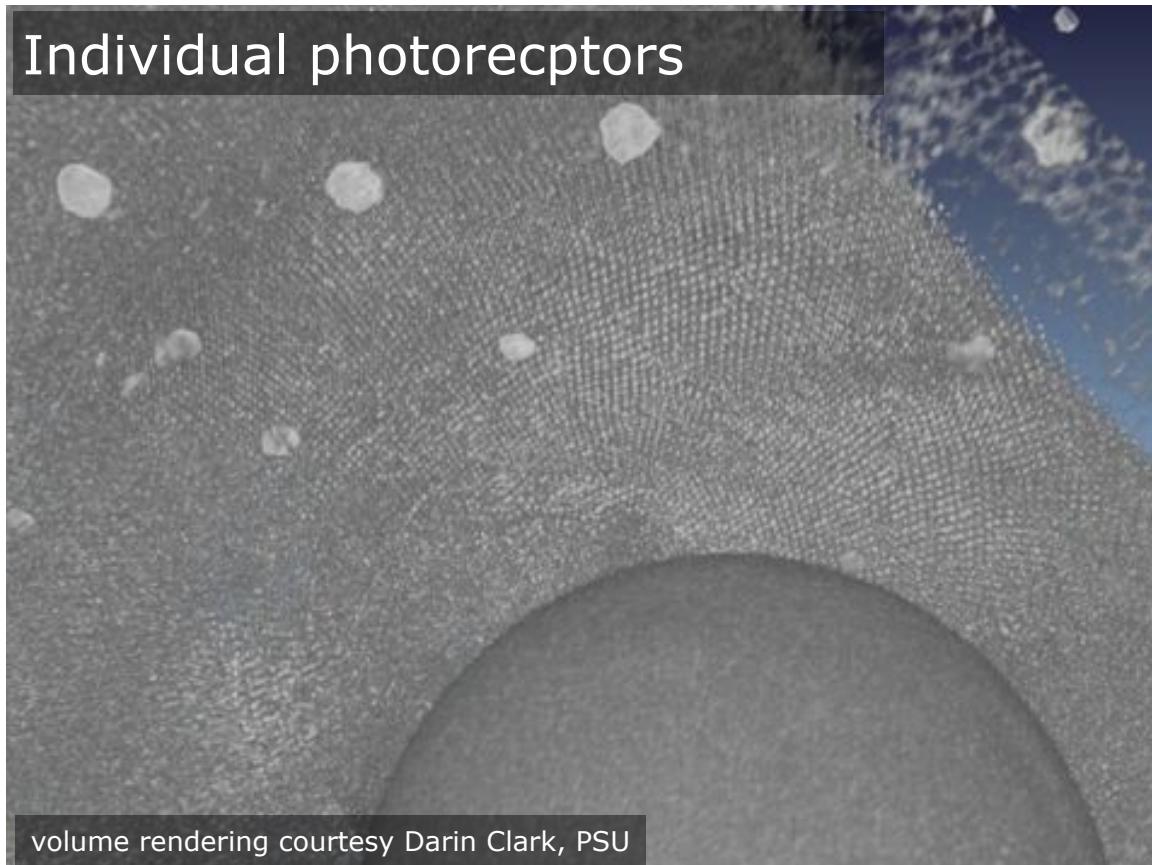
have been hindered by the lack of techniques to create vortices with topological structure. Thanks to a recent advance [19], this is finally possible.

By performing experiments on linked and knotted vortices in water, as well as numerical simulations of Bose-Einstein condensates (a compressible superfluid [20]) and Biot-Savart vortex evolution, we investigate the conservation of helicity, in so far as it can be inferred from the centerlines of reconnecting vortex tubes. We describe a

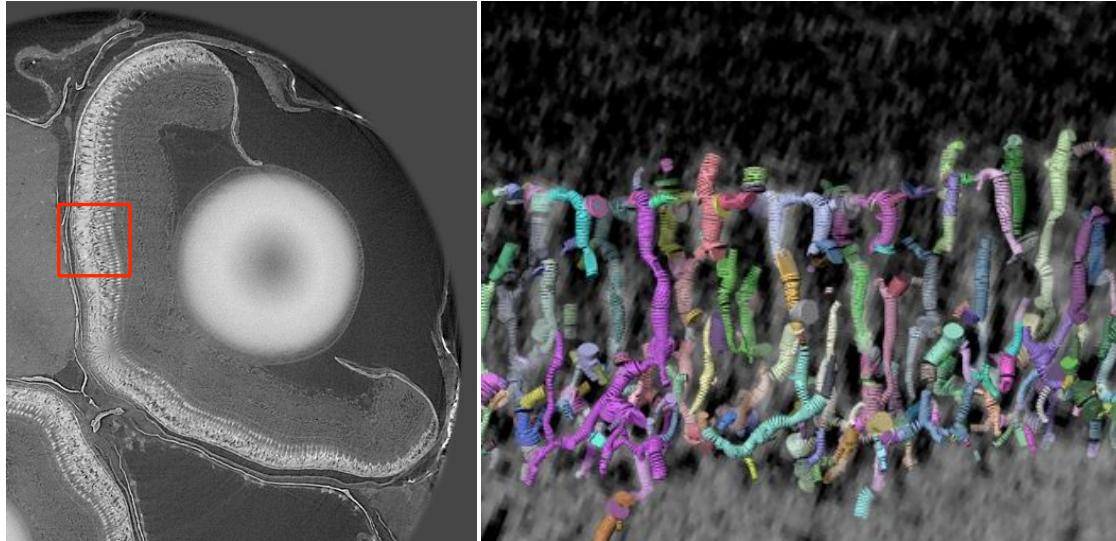
## Application: APS MicroCT (Keith Cheng, PI)



## Individual photoreceptors



## Extraction of micro-anatomy



- Extraction of individual photoreceptors from microCT
  - With scale-space particles
  - Lots of structure to be discovered at other scales!

## But how do you know if a feature is **meaningful**?

---

- A math feature (edge, ridge) may exist in image, but does it represent a physical or real structure?
- Basic idea: its real if it doesn't move when you blur a little bit (stable with respect to scale)
- Precedent:
  - SIFT: Scale-invariant Feature Transform
  - Marr-Hildreth edge detection

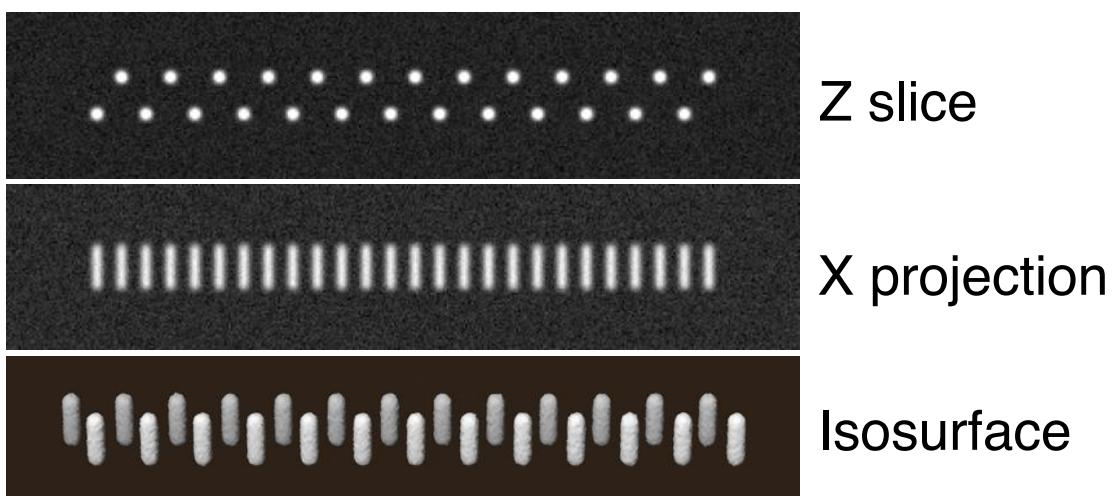
---

[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

## Testing stability with respect to scale

---

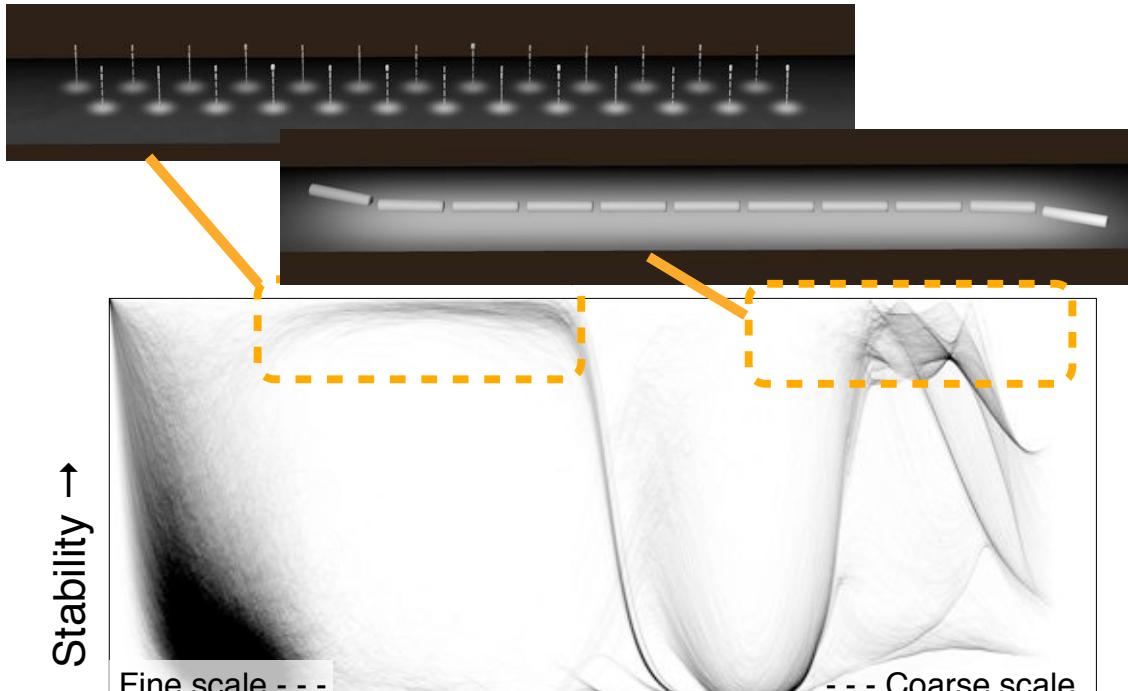
Synthetic data for testing



---

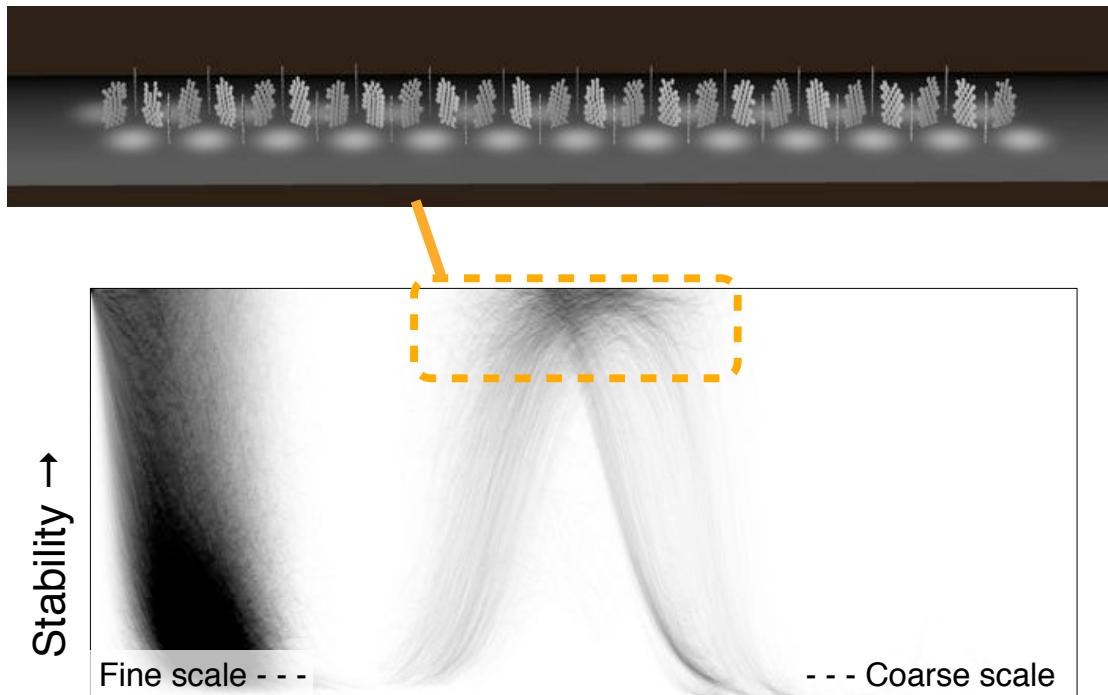
[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Stability of ridge lines



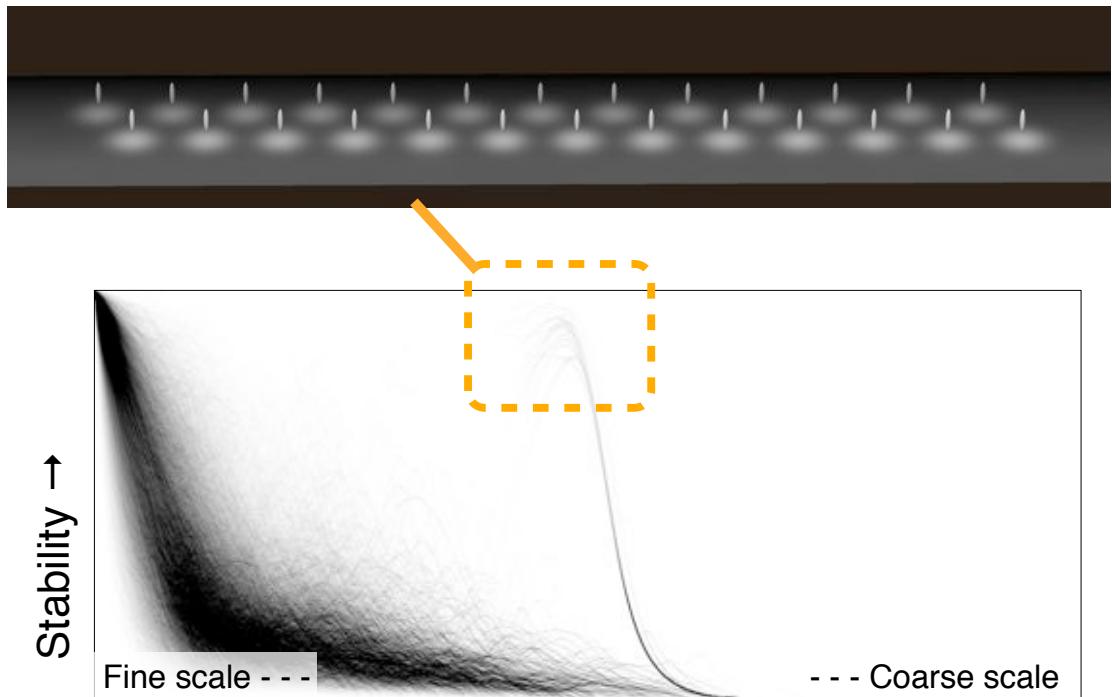
[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Stability of valley surfaces



[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

# Stability of valley surfaces



[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

## Discussion

- New imaging modalities, contrast mechanisms, and their combination => proliferation of possibly useful features; this may help show the way
- “Finding” features in 2 senses: **where**, and **which**
- From image samples to feature samples
- Future work:
  - Points into polyline trees, polygonal meshes
  - Shape modeling and statistics

[Features](#) | [Particles](#) | [Scale-space](#) | [Scale stability](#) | [Results](#) | [Discussion](#)

(switching gears)

## Creating new tools is hard

Increasing range of:

Imaging modalities

Imaging applications

Vis & analysis questions —> algorithms

Want to **rapidly implement** variety of programs  
(the “inner loops”)



Diderot helps rapidly develop portably parallel methods of image visualization and analysis

One example problem....

Want **portably** parallel implementations

Increasing **data size**



Need **parallel** computing

Rapidly shifting parallel computing architectures

# Digital Light Sheet Microscopy

- PI: Kevin White, Institute for Genomics and Systems Biology, U of Chicago

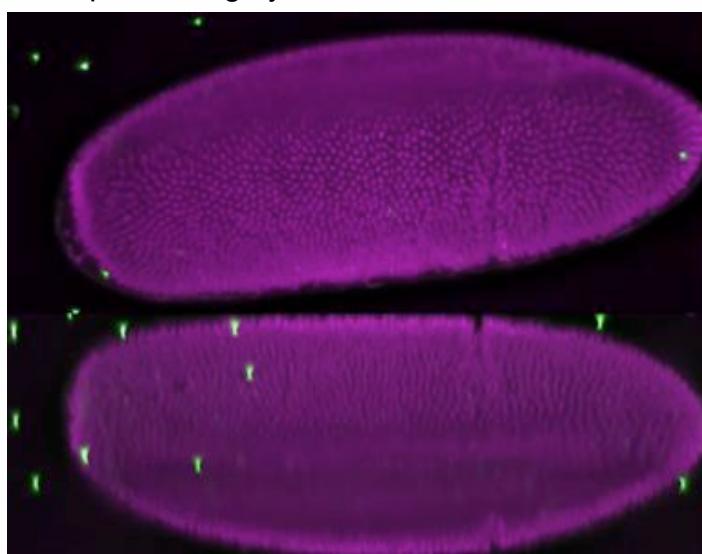


- Drosophila embryogenesis imaged over 20 hours
- movie shows 2 projections of 3D data

# Digital Light Sheet Microscopy

- Data: 2 channels x 1700x1700 (XY) x 500 (Z) x ~500 (time) x ~100 genes

Data processing by Luke Peeler, BS/MS

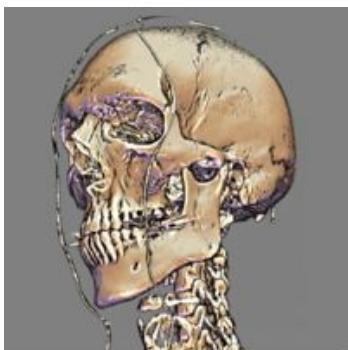


- New Zeiss Z-1 (UChicago KCBD)
- **What is the spatial/temporal structure of gene expression, at the resolution of single cells?**
- Fancy device, but almost no code!

- Domain-Specific Language for portably parallel analysis and visualization of continuous fields (scalar/vector/tensor)
- Gain **programmer efficiency** and **parallel performance** at cost of algorithmic **generality**
- **Portably** parallel: compiles to multi-core CPUs (pthreads), GPUs (OpenCL)
- High-level notation supports rapid development and mathematically legible code ("from whiteboard to executable")
- C Chiw, G Kindlmann, J Reppy, L Samuels, and N Seltzer. **Diderot: A Parallel DSL for Image Analysis and Visualization**. In Proceedings of the 33rd ACM SIGPLAN PLDI, pages 111–120, June 2012.

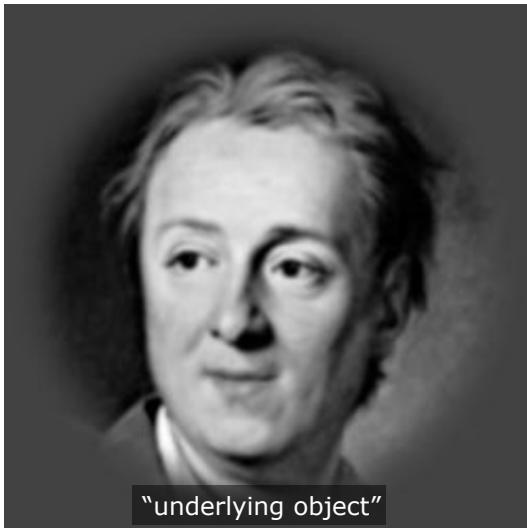
## What is Diderot best at?

- Algorithms with large number of (mostly) independent computations with local properties of continuous fields, e.g.:
  - Direct Volume Rendering
  - Streamlines, Fiber Tractography
  - Particle Systems for Image Feature Sampling



## Objects versus images (again)

- Measurements of objects produce **images**



"underlying object"

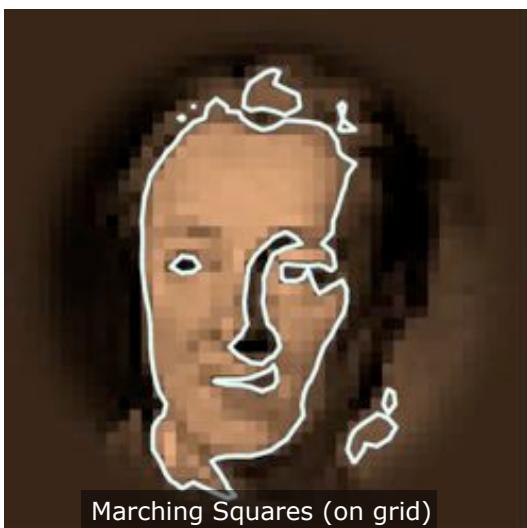


"measured image"

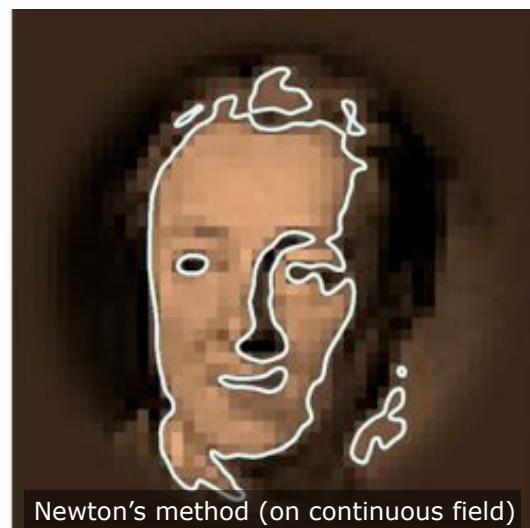
- Goal of scientific vis & analysis is to make statements about the underlying **objects**

## Objects versus images (again)

- Grid orientation/spacing is property of **image**



Marching Squares (on grid)



Newton's method (on continuous field)

- Continuous fields (in Diderot) help get away from grid details towards object properties

# Diderot program structure

- Computation decomposed into collection of mostly autonomous *strands*
- Each strand has state and an **update** method
- **update** implements one iteration of algorithm
  - strands can **stabilize**, **die**, **new**
- Abstractions:
  - Fields: convolution & differentiation of discrete data
  - Parallel computation (CPU vs GPU)
  - Strand communication

## Minimal example

Square roots of numbers 1..1000 by Heron's method

```
// Global definitions


Globals are immutable;  
used for program inputs



Strands are bulk synchronous



Input parameters for initialization



Strand state, including output



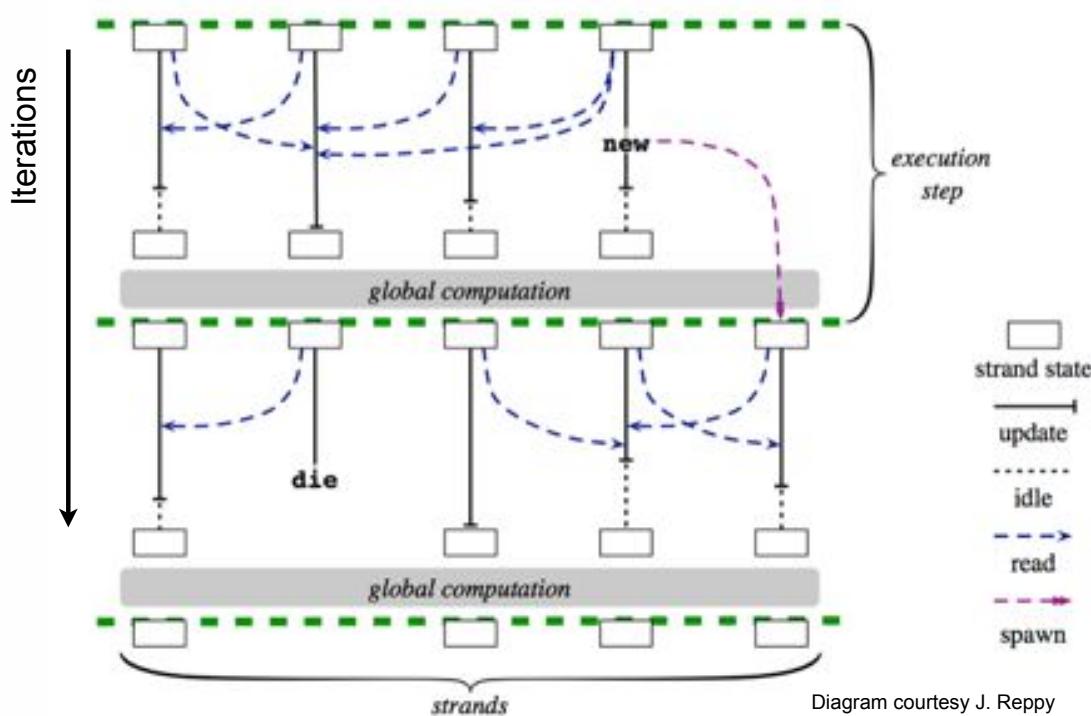
Update method implements algorithm



Initialization of collection of strands with  
comprehension notation


```

# Execution model

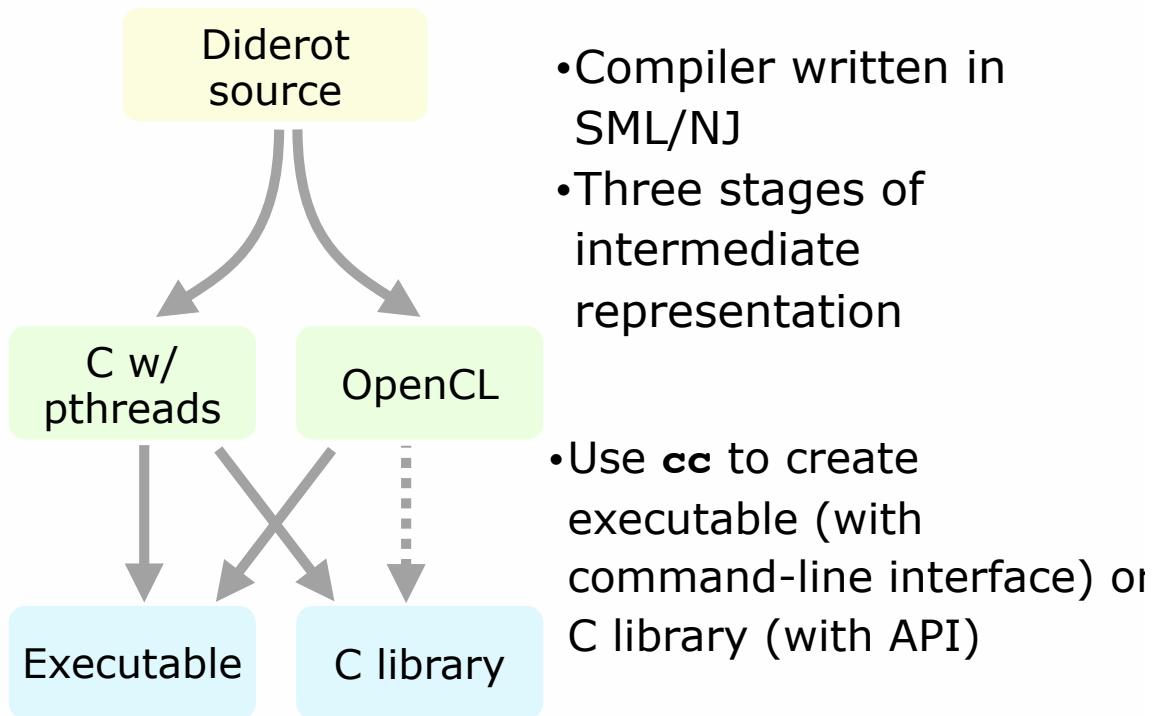


## Example: sampling isosurfaces

```
1 input real isoval = 0.4;
2 field#1(2)[] F = ctmr @ image("ddro.nrrd") - isoval;
3 int grid = 150;
4 int stepsMax = 10;
5 real epsilon = 0.000001;
6 strand FindZero(vec2 x0) {
7     output vec2 x = x0;
8     int steps = 0;
9     update {
10         if (!inside(x, F) || steps > stepsMax)
11             die; // Stop outside domain or after many steps
12         if (|∇F(x)| == 0)
13             die; // Can't proceed with zero derivative
14         // the Newton-Raphson step
15         vec2 dx = normalize(∇F(x)) * F(x)/|∇F(x)|;
16         x -= dx;
17         if (|dx| < epsilon)
18             stabilize; // Converged when step small enough
19         steps += 1;
20     }
21 }
22 initially { FindZero([lerp(0, 1, -0.5, ui, grid-0.5),
23                         lerp(0, 1, -0.5, vi, grid-0.5)])
24             | vi in 0..(grid-1), ui in 0..(grid-1) };
```



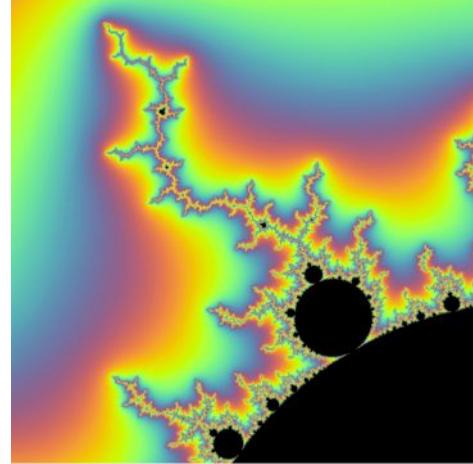
# Compilation



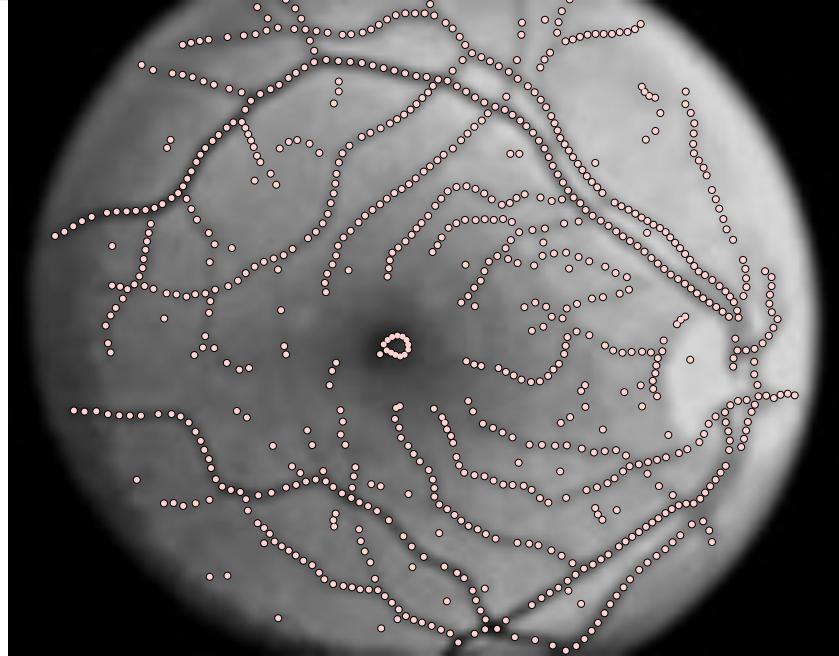
## Mandelbrot set

```
// Global definitions
input int reso = 2000;
input real escape = 4.0;
input int maxiter = 1000;
input vec2 center = [0,0];
input real fov = 2;
field#0(1)[3] cmap = tent ⊕ image("colormap.nrrd");
// Strand definition
strand mandel(vec2 c) {
    vec2 z = c;
    int iter = 0;
    output vec3 rgb = [0, 0, 0];
    update {
        // z = z^2 + c
        z = [z[0]^2 - z[1]^2, 2*z[0]*z[1]] + c;
        if (|z| > escape) {
            // point escaped; color based on iter and |z|
            real time = iter - log2(log(|z|)/log(escape));
            rgb = cmap(fmod(log(time), 1));
            stabilize;
        }
        iter += 1;
        if (iter > maxiter) {
            rgb = [0, 0, 0];
            stabilize;
        }
    }
}

// Strand initialization
initially [ mandel([lerp(center[0]-fov, center[0]+fov,
                           1, realIdx, reso),
                      lerp(center[1]-fov, center[1]+fov,
                           1, compIdx, reso)])
           | compIdx in 1..reso, realIdx in 1..reso ];
```

A vibrant, colorful visualization of the Mandelbrot set. The fractal is rendered using a color map where black represents points that did not escape (i.e., they remain bounded), while other colors represent the number of iterations required for escape. The main body of the fractal is black, while its intricate, branching filaments are rendered in a spectrum of colors including red, orange, yellow, green, and blue. The background is a smooth gradient from dark blue to white.

# Blood vessel sampling w/ particles



Particles repel each other w/ potential function  
(strand communication by Lamont Samuels)

## Finding valley lines

```
strand valleyline(vec3 initpos) {
    output vec3 x = initpos;
    update {
        vec3{3} ev = evecs( $\nabla \otimes \nabla F(x)$ );
        vec3 dir = normalize((ev{3}  $\otimes$  ev{3}
            + ev{2}  $\otimes$  ev{2})  $\cdot$   $\nabla F(x)$ );
        real fdd =  $\nabla F(x) \cdot$  dir;
        real sdd = dir  $\cdot$   $\nabla \otimes \nabla F(x) \cdot$  dir;
        vec3 delta = dir * fdd / sdd; // Newton Optimization
        if (|delta| < epsilon) {
            stabilize;
        }
        x -= delta;
    }
}
```

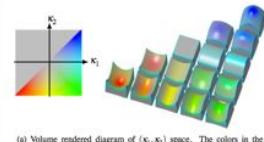
Lung airways (chest CT)



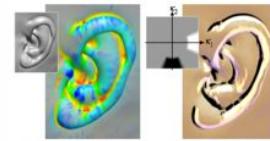
# Example visualization method

- Curvature-based transfer functions (Vis'03)

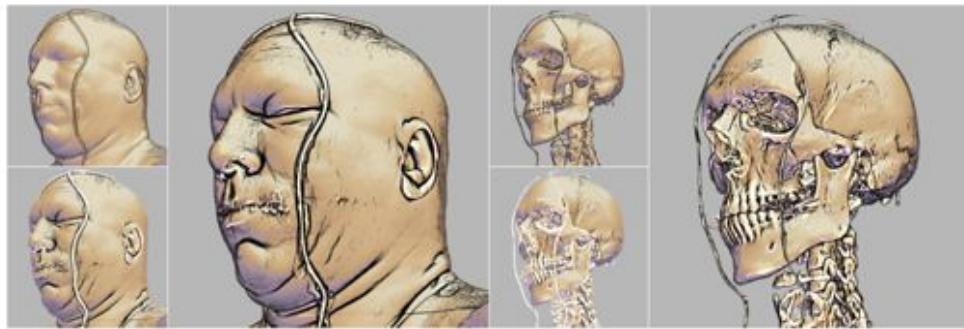
$$\begin{aligned}
 \nabla n^T &= -\nabla \left( \frac{\mathbf{g}^T}{|\mathbf{g}|} \right) = -\left( \frac{\nabla \mathbf{g}^T}{|\mathbf{g}|} - \frac{\mathbf{g} \nabla^T |\mathbf{g}|}{|\mathbf{g}|^2} \right) \\
 &= -\frac{1}{|\mathbf{g}|} \left( \mathbf{H} - \frac{\mathbf{g} \nabla^T (\mathbf{g}^T \mathbf{g})^{1/2}}{|\mathbf{g}|} \right) = -\frac{1}{|\mathbf{g}|} \left( \mathbf{H} - \frac{\mathbf{g} \nabla^T (\mathbf{g}^T \mathbf{g})}{2|\mathbf{g}|(\mathbf{g}^T \mathbf{g})^{1/2}} \right) \\
 &= -\frac{1}{|\mathbf{g}|} \left( \mathbf{H} - \frac{\mathbf{g} (2\mathbf{g}^T \mathbf{H})}{2|\mathbf{g}|^2} \right) = -\frac{1}{|\mathbf{g}|} \left( \mathbf{I} - \frac{\mathbf{g} \mathbf{g}^T}{|\mathbf{g}|^2} \right) \mathbf{H} \\
 &= -\frac{1}{|\mathbf{g}|} (\mathbf{I} - \mathbf{n} \mathbf{n}^T) \mathbf{H}.
 \end{aligned}$$



(a) Volume rendered diagram of  $(x_1, x_2)$  space. The colors in the  $(x_1, x_2)$  transfer function domain are mapped onto the patches with corresponding surface curvature.



(b) Left: Visualization of ear curvature using transfer function from (a); Right: ridge and valley emphasis implemented with inset transfer function, combined with Gooch shading



## The C code to implement that

```

#define DOT_4(a,b) ((a)[0]*(b)[0]+(a)[1]*(b)[1]+(a)[2]*(b)[2]+(a)[3]*(b)[3])
#define VL_4(i, axis) DOT_4(fw0 + (axis)*4, iv##axis + i*4)
#define D1_4(i, axis) DOT_4(fw1 + (axis)*4, iv##axis + i*4)
#define D2_4(i, axis) DOT_4(fw2 + (axis)*4, iv##axis + i*4)

if (/* x0 */ 
    ivY[ 0] = VL_4( 0,X);
    ivY[ 1] = VL_4( 1,X);
    ivY[ 2] = VL_4( 2,X);
    ivY[ 3] = VL_4( 3,X);
    ivY[ 4] = VL_4( 4,X);
    ivY[ 5] = VL_4( 5,X);
    ivY[ 6] = VL_4( 6,X);
    ivY[ 7] = VL_4( 7,X);
    ivY[ 8] = VL_4( 8,X);
    ivY[ 9] = VL_4( 9,X);
    ivY[10] = VL_4(10,X);
    ivY[11] = VL_4(11,X);
    ivY[12] = VL_4(12,X);
    ivY[13] = VL_4(13,X);
    ivY[14] = VL_4(14,X);
    ivY[15] = VL_4(15,X);
    /* x0y0 */
    ivZ[ 0] = VL_4( 0,Y);
    ivZ[ 1] = VL_4( 1,Y);
    ivZ[ 2] = VL_4( 2,Y);
    ivZ[ 3] = VL_4( 3,Y);
    T : /* x0y0z0 */
        if (doV) {
            N : *val = VL_4( 0,Z);
            /* F */
            D : if (!(!doD1 || !doD2))
                return;
            D : /* x0y0z1 */
            k1| if (doD1) {
                gvec[2] = D1_4( 0,Z);
            }
            k2| if (doD2) {
                /* x0y0z2 */
                hess[8] = D2_4( 0,Z);
            }
            /* x1 */
            ivY[ 0] = D1_4( 0,X);
            ivY[ 1] = D1_4( 1,X);
            ivY[ 2] = D1_4( 2,X);
            ivY[ 3] = D1_4( 3,X);
            ivY[ 4] = D1_4( 4,X);
            ivY[ 5] = D1_4( 5,X);
            ivY[ 6] = D1_4( 6,X);
            ivY[ 7] = D1_4( 7,X);
            ivY[ 8] = D1_4( 8,X);
            ivY[ 9] = D1_4( 9,X);
            ivY[10] = D1_4(10,X);
            ivY[11] = D1_4(11,X);
            ivY[12] = D1_4(12,X);
            ivY[13] = D1_4(13,X);
            ivY[14] = D1_4(14,X);
            ivY[15] = D1_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D1_4( 0,Y);
            ivZ[ 1] = D1_4( 1,Y);
            ivZ[ 2] = D1_4( 2,Y);
            ivZ[ 3] = D1_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D1_4( 0,Z);
            ivZ[ 1] = D1_4( 1,Z);
            ivZ[ 2] = D1_4( 2,Z);
            ivZ[ 3] = D1_4( 3,Z);
            /* x0y1z0 */
            if (doD1) {
                gvec[1] = VL_4( 0,Z);
            }
            if (doD2) {
                /* x0y1z1 */
                hess[5] = hess[7] = D1_4( 0,Z);
                /* x0y2 */
                ivZ[ 0] = D2_4( 0,Y);
                ivZ[ 1] = D2_4( 1,Y);
                ivZ[ 2] = D2_4( 2,Y);
                ivZ[ 3] = D2_4( 3,Y);
                /* x0y2z0 */
                hess[4] = VL_4( 0,Z);
                /* h_yy */
            }
            /* x1 */
            ivY[ 0] = D1_4( 0,X);
            ivY[ 1] = D1_4( 1,X);
            ivY[ 2] = D1_4( 2,X);
            ivY[ 3] = D1_4( 3,X);
            ivY[ 4] = D1_4( 4,X);
            ivY[ 5] = D1_4( 5,X);
            ivY[ 6] = D1_4( 6,X);
            ivY[ 7] = D1_4( 7,X);
            ivY[ 8] = D1_4( 8,X);
            ivY[ 9] = D1_4( 9,X);
            ivY[10] = D1_4(10,X);
            ivY[11] = D1_4(11,X);
            ivY[12] = D1_4(12,X);
            ivY[13] = D1_4(13,X);
            ivY[14] = D1_4(14,X);
            ivY[15] = D1_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D1_4( 0,Y);
            ivZ[ 1] = D1_4( 1,Y);
            ivZ[ 2] = D1_4( 2,Y);
            ivZ[ 3] = D1_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D1_4( 0,Z);
            ivZ[ 1] = D1_4( 1,Z);
            ivZ[ 2] = D1_4( 2,Z);
            ivZ[ 3] = D1_4( 3,Z);
            /* x0y1z1 */
            hess[6] = hess[8] = D1_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D2_4( 0,Y);
            ivZ[ 1] = D2_4( 1,Y);
            ivZ[ 2] = D2_4( 2,Y);
            ivZ[ 3] = D2_4( 3,Y);
            /* x0y2z1 */
            hess[7] = hess[9] = D2_4( 0,Z);
            /* x0y3 */
            ivZ[ 0] = D3_4( 0,Y);
            ivZ[ 1] = D3_4( 1,Y);
            ivZ[ 2] = D3_4( 2,Y);
            ivZ[ 3] = D3_4( 3,Y);
            /* x0y3z0 */
            hess[8] = D3_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D3_4( 0,X);
            ivY[ 1] = D3_4( 1,X);
            ivY[ 2] = D3_4( 2,X);
            ivY[ 3] = D3_4( 3,X);
            ivY[ 4] = D3_4( 4,X);
            ivY[ 5] = D3_4( 5,X);
            ivY[ 6] = D3_4( 6,X);
            ivY[ 7] = D3_4( 7,X);
            ivY[ 8] = D3_4( 8,X);
            ivY[ 9] = D3_4( 9,X);
            ivY[10] = D3_4(10,X);
            ivY[11] = D3_4(11,X);
            ivY[12] = D3_4(12,X);
            ivY[13] = D3_4(13,X);
            ivY[14] = D3_4(14,X);
            ivY[15] = D3_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D3_4( 0,Y);
            ivZ[ 1] = D3_4( 1,Y);
            ivZ[ 2] = D3_4( 2,Y);
            ivZ[ 3] = D3_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D3_4( 0,Z);
            ivZ[ 1] = D3_4( 1,Z);
            ivZ[ 2] = D3_4( 2,Z);
            ivZ[ 3] = D3_4( 3,Z);
            /* x0y1z2 */
            hess[9] = hess[11] = D3_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D4_4( 0,Y);
            ivZ[ 1] = D4_4( 1,Y);
            ivZ[ 2] = D4_4( 2,Y);
            ivZ[ 3] = D4_4( 3,Y);
            /* x0y2z2 */
            hess[10] = D4_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D4_4( 0,X);
            ivY[ 1] = D4_4( 1,X);
            ivY[ 2] = D4_4( 2,X);
            ivY[ 3] = D4_4( 3,X);
            ivY[ 4] = D4_4( 4,X);
            ivY[ 5] = D4_4( 5,X);
            ivY[ 6] = D4_4( 6,X);
            ivY[ 7] = D4_4( 7,X);
            ivY[ 8] = D4_4( 8,X);
            ivY[ 9] = D4_4( 9,X);
            ivY[10] = D4_4(10,X);
            ivY[11] = D4_4(11,X);
            ivY[12] = D4_4(12,X);
            ivY[13] = D4_4(13,X);
            ivY[14] = D4_4(14,X);
            ivY[15] = D4_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D4_4( 0,Y);
            ivZ[ 1] = D4_4( 1,Y);
            ivZ[ 2] = D4_4( 2,Y);
            ivZ[ 3] = D4_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D4_4( 0,Z);
            ivZ[ 1] = D4_4( 1,Z);
            ivZ[ 2] = D4_4( 2,Z);
            ivZ[ 3] = D4_4( 3,Z);
            /* x0y1z3 */
            hess[11] = hess[13] = D4_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D5_4( 0,Y);
            ivZ[ 1] = D5_4( 1,Y);
            ivZ[ 2] = D5_4( 2,Y);
            ivZ[ 3] = D5_4( 3,Y);
            /* x0y2z3 */
            hess[12] = D5_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D5_4( 0,X);
            ivY[ 1] = D5_4( 1,X);
            ivY[ 2] = D5_4( 2,X);
            ivY[ 3] = D5_4( 3,X);
            ivY[ 4] = D5_4( 4,X);
            ivY[ 5] = D5_4( 5,X);
            ivY[ 6] = D5_4( 6,X);
            ivY[ 7] = D5_4( 7,X);
            ivY[ 8] = D5_4( 8,X);
            ivY[ 9] = D5_4( 9,X);
            ivY[10] = D5_4(10,X);
            ivY[11] = D5_4(11,X);
            ivY[12] = D5_4(12,X);
            ivY[13] = D5_4(13,X);
            ivY[14] = D5_4(14,X);
            ivY[15] = D5_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D5_4( 0,Y);
            ivZ[ 1] = D5_4( 1,Y);
            ivZ[ 2] = D5_4( 2,Y);
            ivZ[ 3] = D5_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D5_4( 0,Z);
            ivZ[ 1] = D5_4( 1,Z);
            ivZ[ 2] = D5_4( 2,Z);
            ivZ[ 3] = D5_4( 3,Z);
            /* x0y1z4 */
            hess[13] = hess[15] = D5_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D6_4( 0,Y);
            ivZ[ 1] = D6_4( 1,Y);
            ivZ[ 2] = D6_4( 2,Y);
            ivZ[ 3] = D6_4( 3,Y);
            /* x0y2z4 */
            hess[14] = D6_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D6_4( 0,X);
            ivY[ 1] = D6_4( 1,X);
            ivY[ 2] = D6_4( 2,X);
            ivY[ 3] = D6_4( 3,X);
            ivY[ 4] = D6_4( 4,X);
            ivY[ 5] = D6_4( 5,X);
            ivY[ 6] = D6_4( 6,X);
            ivY[ 7] = D6_4( 7,X);
            ivY[ 8] = D6_4( 8,X);
            ivY[ 9] = D6_4( 9,X);
            ivY[10] = D6_4(10,X);
            ivY[11] = D6_4(11,X);
            ivY[12] = D6_4(12,X);
            ivY[13] = D6_4(13,X);
            ivY[14] = D6_4(14,X);
            ivY[15] = D6_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D6_4( 0,Y);
            ivZ[ 1] = D6_4( 1,Y);
            ivZ[ 2] = D6_4( 2,Y);
            ivZ[ 3] = D6_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D6_4( 0,Z);
            ivZ[ 1] = D6_4( 1,Z);
            ivZ[ 2] = D6_4( 2,Z);
            ivZ[ 3] = D6_4( 3,Z);
            /* x0y1z5 */
            hess[15] = hess[17] = D6_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D7_4( 0,Y);
            ivZ[ 1] = D7_4( 1,Y);
            ivZ[ 2] = D7_4( 2,Y);
            ivZ[ 3] = D7_4( 3,Y);
            /* x0y2z5 */
            hess[16] = D7_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D7_4( 0,X);
            ivY[ 1] = D7_4( 1,X);
            ivY[ 2] = D7_4( 2,X);
            ivY[ 3] = D7_4( 3,X);
            ivY[ 4] = D7_4( 4,X);
            ivY[ 5] = D7_4( 5,X);
            ivY[ 6] = D7_4( 6,X);
            ivY[ 7] = D7_4( 7,X);
            ivY[ 8] = D7_4( 8,X);
            ivY[ 9] = D7_4( 9,X);
            ivY[10] = D7_4(10,X);
            ivY[11] = D7_4(11,X);
            ivY[12] = D7_4(12,X);
            ivY[13] = D7_4(13,X);
            ivY[14] = D7_4(14,X);
            ivY[15] = D7_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D7_4( 0,Y);
            ivZ[ 1] = D7_4( 1,Y);
            ivZ[ 2] = D7_4( 2,Y);
            ivZ[ 3] = D7_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D7_4( 0,Z);
            ivZ[ 1] = D7_4( 1,Z);
            ivZ[ 2] = D7_4( 2,Z);
            ivZ[ 3] = D7_4( 3,Z);
            /* x0y1z6 */
            hess[17] = hess[19] = D7_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D8_4( 0,Y);
            ivZ[ 1] = D8_4( 1,Y);
            ivZ[ 2] = D8_4( 2,Y);
            ivZ[ 3] = D8_4( 3,Y);
            /* x0y2z6 */
            hess[18] = D8_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D8_4( 0,X);
            ivY[ 1] = D8_4( 1,X);
            ivY[ 2] = D8_4( 2,X);
            ivY[ 3] = D8_4( 3,X);
            ivY[ 4] = D8_4( 4,X);
            ivY[ 5] = D8_4( 5,X);
            ivY[ 6] = D8_4( 6,X);
            ivY[ 7] = D8_4( 7,X);
            ivY[ 8] = D8_4( 8,X);
            ivY[ 9] = D8_4( 9,X);
            ivY[10] = D8_4(10,X);
            ivY[11] = D8_4(11,X);
            ivY[12] = D8_4(12,X);
            ivY[13] = D8_4(13,X);
            ivY[14] = D8_4(14,X);
            ivY[15] = D8_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D8_4( 0,Y);
            ivZ[ 1] = D8_4( 1,Y);
            ivZ[ 2] = D8_4( 2,Y);
            ivZ[ 3] = D8_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D8_4( 0,Z);
            ivZ[ 1] = D8_4( 1,Z);
            ivZ[ 2] = D8_4( 2,Z);
            ivZ[ 3] = D8_4( 3,Z);
            /* x0y1z7 */
            hess[19] = hess[21] = D8_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D9_4( 0,Y);
            ivZ[ 1] = D9_4( 1,Y);
            ivZ[ 2] = D9_4( 2,Y);
            ivZ[ 3] = D9_4( 3,Y);
            /* x0y2z7 */
            hess[20] = D9_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D9_4( 0,X);
            ivY[ 1] = D9_4( 1,X);
            ivY[ 2] = D9_4( 2,X);
            ivY[ 3] = D9_4( 3,X);
            ivY[ 4] = D9_4( 4,X);
            ivY[ 5] = D9_4( 5,X);
            ivY[ 6] = D9_4( 6,X);
            ivY[ 7] = D9_4( 7,X);
            ivY[ 8] = D9_4( 8,X);
            ivY[ 9] = D9_4( 9,X);
            ivY[10] = D9_4(10,X);
            ivY[11] = D9_4(11,X);
            ivY[12] = D9_4(12,X);
            ivY[13] = D9_4(13,X);
            ivY[14] = D9_4(14,X);
            ivY[15] = D9_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D9_4( 0,Y);
            ivZ[ 1] = D9_4( 1,Y);
            ivZ[ 2] = D9_4( 2,Y);
            ivZ[ 3] = D9_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D9_4( 0,Z);
            ivZ[ 1] = D9_4( 1,Z);
            ivZ[ 2] = D9_4( 2,Z);
            ivZ[ 3] = D9_4( 3,Z);
            /* x0y1z8 */
            hess[21] = hess[23] = D9_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D10_4( 0,Y);
            ivZ[ 1] = D10_4( 1,Y);
            ivZ[ 2] = D10_4( 2,Y);
            ivZ[ 3] = D10_4( 3,Y);
            /* x0y2z8 */
            hess[22] = D10_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D10_4( 0,X);
            ivY[ 1] = D10_4( 1,X);
            ivY[ 2] = D10_4( 2,X);
            ivY[ 3] = D10_4( 3,X);
            ivY[ 4] = D10_4( 4,X);
            ivY[ 5] = D10_4( 5,X);
            ivY[ 6] = D10_4( 6,X);
            ivY[ 7] = D10_4( 7,X);
            ivY[ 8] = D10_4( 8,X);
            ivY[ 9] = D10_4( 9,X);
            ivY[10] = D10_4(10,X);
            ivY[11] = D10_4(11,X);
            ivY[12] = D10_4(12,X);
            ivY[13] = D10_4(13,X);
            ivY[14] = D10_4(14,X);
            ivY[15] = D10_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D10_4( 0,Y);
            ivZ[ 1] = D10_4( 1,Y);
            ivZ[ 2] = D10_4( 2,Y);
            ivZ[ 3] = D10_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D10_4( 0,Z);
            ivZ[ 1] = D10_4( 1,Z);
            ivZ[ 2] = D10_4( 2,Z);
            ivZ[ 3] = D10_4( 3,Z);
            /* x0y1z9 */
            hess[23] = hess[25] = D10_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D11_4( 0,Y);
            ivZ[ 1] = D11_4( 1,Y);
            ivZ[ 2] = D11_4( 2,Y);
            ivZ[ 3] = D11_4( 3,Y);
            /* x0y2z9 */
            hess[24] = D11_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D11_4( 0,X);
            ivY[ 1] = D11_4( 1,X);
            ivY[ 2] = D11_4( 2,X);
            ivY[ 3] = D11_4( 3,X);
            ivY[ 4] = D11_4( 4,X);
            ivY[ 5] = D11_4( 5,X);
            ivY[ 6] = D11_4( 6,X);
            ivY[ 7] = D11_4( 7,X);
            ivY[ 8] = D11_4( 8,X);
            ivY[ 9] = D11_4( 9,X);
            ivY[10] = D11_4(10,X);
            ivY[11] = D11_4(11,X);
            ivY[12] = D11_4(12,X);
            ivY[13] = D11_4(13,X);
            ivY[14] = D11_4(14,X);
            ivY[15] = D11_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D11_4( 0,Y);
            ivZ[ 1] = D11_4( 1,Y);
            ivZ[ 2] = D11_4( 2,Y);
            ivZ[ 3] = D11_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D11_4( 0,Z);
            ivZ[ 1] = D11_4( 1,Z);
            ivZ[ 2] = D11_4( 2,Z);
            ivZ[ 3] = D11_4( 3,Z);
            /* x0y1z10 */
            hess[25] = hess[27] = D11_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D12_4( 0,Y);
            ivZ[ 1] = D12_4( 1,Y);
            ivZ[ 2] = D12_4( 2,Y);
            ivZ[ 3] = D12_4( 3,Y);
            /* x0y2z10 */
            hess[26] = D12_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D12_4( 0,X);
            ivY[ 1] = D12_4( 1,X);
            ivY[ 2] = D12_4( 2,X);
            ivY[ 3] = D12_4( 3,X);
            ivY[ 4] = D12_4( 4,X);
            ivY[ 5] = D12_4( 5,X);
            ivY[ 6] = D12_4( 6,X);
            ivY[ 7] = D12_4( 7,X);
            ivY[ 8] = D12_4( 8,X);
            ivY[ 9] = D12_4( 9,X);
            ivY[10] = D12_4(10,X);
            ivY[11] = D12_4(11,X);
            ivY[12] = D12_4(12,X);
            ivY[13] = D12_4(13,X);
            ivY[14] = D12_4(14,X);
            ivY[15] = D12_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D12_4( 0,Y);
            ivZ[ 1] = D12_4( 1,Y);
            ivZ[ 2] = D12_4( 2,Y);
            ivZ[ 3] = D12_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D12_4( 0,Z);
            ivZ[ 1] = D12_4( 1,Z);
            ivZ[ 2] = D12_4( 2,Z);
            ivZ[ 3] = D12_4( 3,Z);
            /* x0y1z11 */
            hess[27] = hess[29] = D12_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D13_4( 0,Y);
            ivZ[ 1] = D13_4( 1,Y);
            ivZ[ 2] = D13_4( 2,Y);
            ivZ[ 3] = D13_4( 3,Y);
            /* x0y2z11 */
            hess[28] = D13_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D13_4( 0,X);
            ivY[ 1] = D13_4( 1,X);
            ivY[ 2] = D13_4( 2,X);
            ivY[ 3] = D13_4( 3,X);
            ivY[ 4] = D13_4( 4,X);
            ivY[ 5] = D13_4( 5,X);
            ivY[ 6] = D13_4( 6,X);
            ivY[ 7] = D13_4( 7,X);
            ivY[ 8] = D13_4( 8,X);
            ivY[ 9] = D13_4( 9,X);
            ivY[10] = D13_4(10,X);
            ivY[11] = D13_4(11,X);
            ivY[12] = D13_4(12,X);
            ivY[13] = D13_4(13,X);
            ivY[14] = D13_4(14,X);
            ivY[15] = D13_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D13_4( 0,Y);
            ivZ[ 1] = D13_4( 1,Y);
            ivZ[ 2] = D13_4( 2,Y);
            ivZ[ 3] = D13_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D13_4( 0,Z);
            ivZ[ 1] = D13_4( 1,Z);
            ivZ[ 2] = D13_4( 2,Z);
            ivZ[ 3] = D13_4( 3,Z);
            /* x0y1z12 */
            hess[29] = hess[31] = D13_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D14_4( 0,Y);
            ivZ[ 1] = D14_4( 1,Y);
            ivZ[ 2] = D14_4( 2,Y);
            ivZ[ 3] = D14_4( 3,Y);
            /* x0y2z12 */
            hess[30] = D14_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D14_4( 0,X);
            ivY[ 1] = D14_4( 1,X);
            ivY[ 2] = D14_4( 2,X);
            ivY[ 3] = D14_4( 3,X);
            ivY[ 4] = D14_4( 4,X);
            ivY[ 5] = D14_4( 5,X);
            ivY[ 6] = D14_4( 6,X);
            ivY[ 7] = D14_4( 7,X);
            ivY[ 8] = D14_4( 8,X);
            ivY[ 9] = D14_4( 9,X);
            ivY[10] = D14_4(10,X);
            ivY[11] = D14_4(11,X);
            ivY[12] = D14_4(12,X);
            ivY[13] = D14_4(13,X);
            ivY[14] = D14_4(14,X);
            ivY[15] = D14_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D14_4( 0,Y);
            ivZ[ 1] = D14_4( 1,Y);
            ivZ[ 2] = D14_4( 2,Y);
            ivZ[ 3] = D14_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D14_4( 0,Z);
            ivZ[ 1] = D14_4( 1,Z);
            ivZ[ 2] = D14_4( 2,Z);
            ivZ[ 3] = D14_4( 3,Z);
            /* x0y1z13 */
            hess[31] = hess[33] = D14_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D15_4( 0,Y);
            ivZ[ 1] = D15_4( 1,Y);
            ivZ[ 2] = D15_4( 2,Y);
            ivZ[ 3] = D15_4( 3,Y);
            /* x0y2z13 */
            hess[32] = D15_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D15_4( 0,X);
            ivY[ 1] = D15_4( 1,X);
            ivY[ 2] = D15_4( 2,X);
            ivY[ 3] = D15_4( 3,X);
            ivY[ 4] = D15_4( 4,X);
            ivY[ 5] = D15_4( 5,X);
            ivY[ 6] = D15_4( 6,X);
            ivY[ 7] = D15_4( 7,X);
            ivY[ 8] = D15_4( 8,X);
            ivY[ 9] = D15_4( 9,X);
            ivY[10] = D15_4(10,X);
            ivY[11] = D15_4(11,X);
            ivY[12] = D15_4(12,X);
            ivY[13] = D15_4(13,X);
            ivY[14] = D15_4(14,X);
            ivY[15] = D15_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D15_4( 0,Y);
            ivZ[ 1] = D15_4( 1,Y);
            ivZ[ 2] = D15_4( 2,Y);
            ivZ[ 3] = D15_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D15_4( 0,Z);
            ivZ[ 1] = D15_4( 1,Z);
            ivZ[ 2] = D15_4( 2,Z);
            ivZ[ 3] = D15_4( 3,Z);
            /* x0y1z14 */
            hess[33] = hess[35] = D15_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D16_4( 0,Y);
            ivZ[ 1] = D16_4( 1,Y);
            ivZ[ 2] = D16_4( 2,Y);
            ivZ[ 3] = D16_4( 3,Y);
            /* x0y2z14 */
            hess[34] = D16_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D16_4( 0,X);
            ivY[ 1] = D16_4( 1,X);
            ivY[ 2] = D16_4( 2,X);
            ivY[ 3] = D16_4( 3,X);
            ivY[ 4] = D16_4( 4,X);
            ivY[ 5] = D16_4( 5,X);
            ivY[ 6] = D16_4( 6,X);
            ivY[ 7] = D16_4( 7,X);
            ivY[ 8] = D16_4( 8,X);
            ivY[ 9] = D16_4( 9,X);
            ivY[10] = D16_4(10,X);
            ivY[11] = D16_4(11,X);
            ivY[12] = D16_4(12,X);
            ivY[13] = D16_4(13,X);
            ivY[14] = D16_4(14,X);
            ivY[15] = D16_4(15,X);
            /* x1y0 */
            /* hess */
            ivZ[ 0] = D16_4( 0,Y);
            ivZ[ 1] = D16_4( 1,Y);
            ivZ[ 2] = D16_4( 2,Y);
            ivZ[ 3] = D16_4( 3,Y);
            /* x0y1 */
            ivZ[ 0] = D16_4( 0,Z);
            ivZ[ 1] = D16_4( 1,Z);
            ivZ[ 2] = D16_4( 2,Z);
            ivZ[ 3] = D16_4( 3,Z);
            /* x0y1z15 */
            hess[35] = hess[37] = D16_4( 0,Z);
            /* x0y2 */
            ivZ[ 0] = D17_4( 0,Y);
            ivZ[ 1] = D17_4( 1,Y);
            ivZ[ 2] = D17_4( 2,Y);
            ivZ[ 3] = D17_4( 3,Y);
            /* x0y2z15 */
            hess[36] = D17_4( 0,Z);
            /* h_zz */
            /* x1 */
            ivY[ 0] = D17_4( 0,X);
            ivY[ 1] = D17_4( 1,X);
            ivY[ 2] = D17_4( 2,X);
            ivY[ 3] = D17_4( 3,X);
            ivY[ 4] = D17_4( 4,X);
            ivY[ 5] = D17_4( 5,X);
            ivY[ 6] = D17_4( 6,X);
            ivY[ 7] = D17_4( 7,X);
            ivY[ 8] = D17_4( 8,X);
            ivY[ 9] = D17_4( 9,X);
            ivY[10] = D17_4(10,X);
            ivY[11] = D17_4(11,X);
            ivY[12] = D17_4(12,X);
            ivY[13] = D17_4(13,X);
            ivY[14] = D17_4(14,X);
            ivY[15] = D17_4
```

# OpenCL code (for GPUs)

```

float4 computeGradient(image3d_t sampler, float4 gradPos, const float gradOffset)
{
    //central differences gradient
    return (float4){0};

    read_image(sampler, linearSampler, (float4)(gradPos.x+gradOffset, gradPos.y, gradPos.z, 0.f)).x-
    read_image(sampler, linearSampler, (float4)(gradPos.x-gradOffset, gradPos.y, gradPos.z, 0.f)).x-
    read_image(sampler, linearSampler, (float4)(gradPos.x, gradPos.y+gradOffset, gradPos.z, 0.f)).x-
    read_image(sampler, linearSampler, (float4)(gradPos.x, gradPos.y-gradOffset, gradPos.z, 0.f)).x-
    read_image(sampler, linearSampler, (float4)(gradPos.x, gradPos.y, gradPos.z+gradOffset, 0.f)).x-
    read_image(sampler, linearSampler, (float4)(gradPos.x, gradPos.y, gradPos.z-gradOffset, 0.f)).x;
    0.1;
}

float2 computeCurvature(
    image3d_t sampler,
    float4 gradPos,
    const float gradOffset
)
{
    float4 gradient = computeGradient(
        sampler,
        gradPos,
        gradOffset);
    float4 gradient1 = computeGradient(
        sampler,
        gradPos+(float4)(gradOffset,0.f,0.f,0.f) ,
        gradOffset);
    float4 gradient2 = computeGradient(
        sampler,
        gradPos-(float4)(gradOffset,0.f,0.f,0.f) ,
        gradOffset);
    float4 gradient3 = computeGradient(
        sampler,
        gradPos+(float4)(0.f,gradOffset,0.f,0.f) ,
        gradOffset);
    float4 gradient4 = computeGradient(
        sampler,
        gradPos-(float4)(0.f,gradOffset,0.f,0.f) ,
        gradOffset);
    float4 gradient5 = computeGradient(
        sampler,
        gradPos+(float4)(0.f,0.f,gradOffset,0.f) ,
        gradOffset);
    float4 gradient6 = computeGradient(
        sampler,
        gradPos-(float4)(0.f,0.f,gradOffset,0.f) ,
        gradOffset);

    gradient1 = fast_normalize(gradient1);
    gradient2 = fast_normalize(gradient2);
    gradient3 = fast_normalize(gradient3);
    gradient4 = fast_normalize(gradient4);
    gradient5 = fast_normalize(gradient5);
    gradient6 = fast_normalize(gradient6);

    float l = fast_length(gradient);
    if (l <= 0.0f)
        return (float2)(0.0f,0.0f);

    float4 n = -gradient;
    float P[3][3];
    P[0][0] = 1.0f*x*x;
    P[0][1] = x*x*y;
    P[0][2] = x*x*z;
    P[1][0] = y*y*x;
    P[1][1] = 1.0f*y*y;
    P[1][2] = y*y*z;
    P[2][0] = z*z*x;
    P[2][1] = z*z*y;
    P[2][2] = 1.0f*z*z;

    float hessian[3][3];
    hessian[0][0] = gradient1.x - gradient2.x;
    hessian[0][1] = gradient1.y - gradient2.y;
    hessian[0][2] = gradient1.z - gradient2.z;
    hessian[1][0] = gradient3.x - gradient4.x;
    hessian[1][1] = gradient3.y - gradient4.y;
    hessian[1][2] = gradient3.z - gradient4.z;
    hessian[2][0] = gradient5.x - gradient6.x;
    hessian[2][1] = gradient5.y - gradient6.y;
    hessian[2][2] = gradient5.z - gradient6.z;

    float T[3][3];
    float G[3][3];// = -P'*hessian*P;
    T[0][0] = P[0][0]*hessian[0][0] + P[1][0]*hessian[0][1] + P[2][0]*hessian[0][2];
    T[1][0] = P[0][1]*hessian[1][0] + P[1][1]*hessian[1][1] + P[2][1]*hessian[1][2];
    T[2][0] = P[0][2]*hessian[2][0] + P[1][2]*hessian[2][1] + P[2][2]*hessian[2][2];
    T[0][1] = P[0][0]*hessian[0][1] + P[1][0]*hessian[0][0] + P[2][0]*hessian[0][2];
    T[1][1] = P[0][1]*hessian[1][1] + P[1][1]*hessian[1][0] + P[2][1]*hessian[1][2];
    T[2][1] = P[0][2]*hessian[2][1] + P[1][2]*hessian[2][0] + P[2][2]*hessian[2][2];
    T[0][2] = P[0][0]*hessian[0][2] + P[1][0]*hessian[0][1] + P[2][0]*hessian[0][0];
    T[1][2] = P[0][1]*hessian[1][2] + P[1][1]*hessian[1][1] + P[2][1]*hessian[1][0];
    T[2][2] = P[0][2]*hessian[2][2] + P[1][2]*hessian[2][1] + P[2][2]*hessian[2][0];

    G[0][0] = (T[0][0]*P[0][0]) + T[1][0]*P[1][0] + T[2][0]*P[2][0];
    G[1][0] = (T[0][0]*P[0][1]) + T[1][0]*P[1][1] + T[2][0]*P[2][1];
    G[2][0] = (T[0][0]*P[0][2]) + T[1][0]*P[1][2] + T[2][0]*P[2][2];
    G[0][1] = (T[0][1]*P[0][0]) + T[1][1]*P[1][0] + T[2][1]*P[2][0];
    G[1][1] = (T[0][1]*P[0][1]) + T[1][1]*P[1][1] + T[2][1]*P[2][1];
    G[2][1] = (T[0][1]*P[0][2]) + T[1][1]*P[1][2] + T[2][1]*P[2][2];
    G[0][2] = (T[0][2]*P[0][0]) + T[1][2]*P[1][0] + T[2][2]*P[2][0];
    G[1][2] = (T[0][2]*P[0][1]) + T[1][2]*P[1][1] + T[2][2]*P[2][1];
    G[2][2] = (T[0][2]*P[0][2]) + T[1][2]*P[1][2] + T[2][2]*P[2][2];

    float k1 = G[0][0]+G[1][1]+G[2][2];
    float f = sqrt(G[0][0]*G[0][0]+G[1][1]*G[1][1]+G[2][2]*G[2][2]);
    float k2 = (t - sqrt(2.0f*k1*k1))/2.0f;
    return (float2)(k1,k2);
}

```

Courtesy Klaus Engel, Siemens

## Example: curvature measurement

```

// volume dataset
field#2(3) F = bspln3 @ load("quads.img");

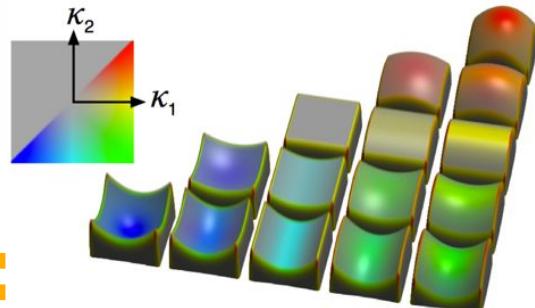
// RGB colormap of (kappa1,kappa2)
field#0(2)[3] RGB = tent @ load("rgb.img");

```

```

vec3 grad = -∇F(pos);
vec3 norm = normalize(grad);
// begin curvature computation
tensor[3,3] H = ∇⊗∇F(pos);
tensor[3,3] P = identity[3] - norm⊗norm;
tensor[3,3] G = -(P·H·P)/|grad|;
real disc = max(0.0, sqrt(2.0*|G|^2 - trace(G)^2));
real k1 = (trace(G) + disc)/2.0;
real k2 = (trace(G) - disc)/2.0;
// find material RGBA
vec3 matRGB = RGB([clamp(-1.0, 1.0, 6.0*k1),
                    clamp(-1.0, 1.0, 6.0*k2)]);

```



Direct (coordinate-free) notation encourages and basis-independent code (eventually, dimension-independent code)

# Much to do (still) ...

---

- Mathematically idiomatic field definitions
  - **lifting**:  $\text{field}\#1(3)[3] V = \text{ctmr} \circledast \text{image}(\text{vec.nrrd}) \rightarrow \text{field}\#1(3)[] M = |V|$  (Charisee Chiw)
  - **composition**:  $V = W \circ F$ , and then differentiation by chain rule
- Fields from point clouds, Finite Element Meshes
- Many possibilities for GUI / IDE:
  - Sliders for all **input** variables
  - Nicer compiler error messages
  - Stepping debugger
- More parallel targets: MPI, CUDA
- Virtual memory for big datasets

## Summary

---

- Harder and harder to extract knowledge from scientific imaging
- Biologists are most in need of practical parallel tools, and least empowered to create them for their work
- Diderot is (ambitious) work-in-progress
- Is open-source (and undocumented)
  - <http://diderot-language.cs.uchicago.edu>
- Thanks for your attention! glk@uchicago.edu