Oct 29 2015

VIS2015

VAST * INFOVIS * SCIVIS

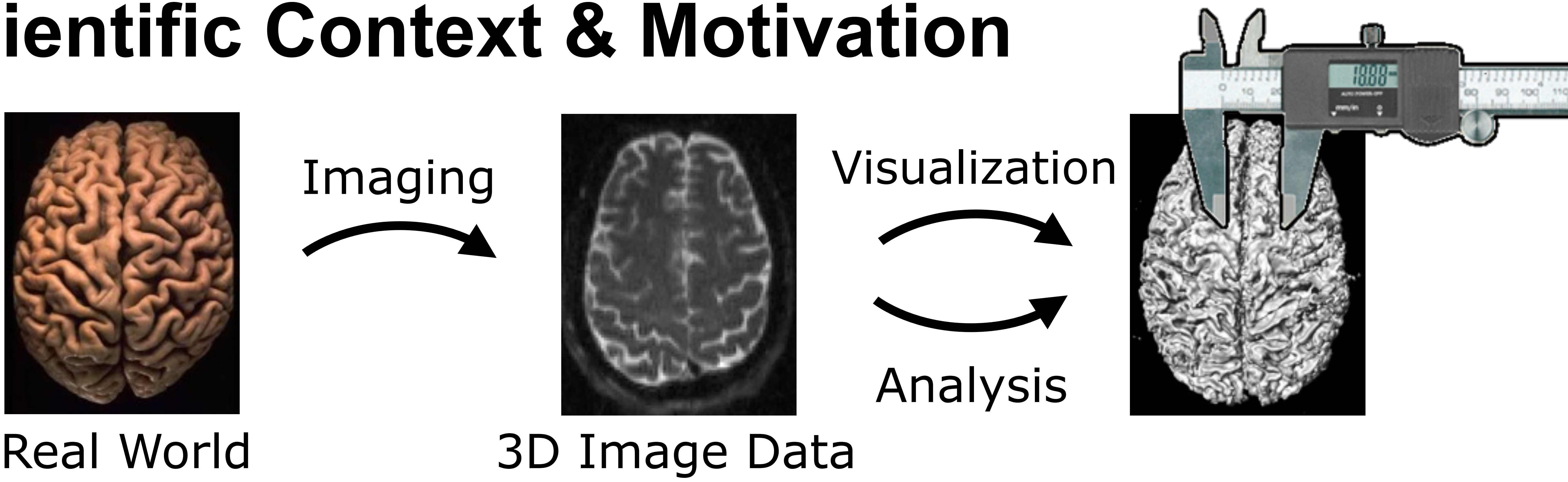
Diderot: a Domain-Specific Language for Portable Parallel Scientific Visualization and Image Analysis

Gordon Kindlmann, Charisee Chiw, Nicholas Seltzer, Lamont Samuels, John Reppy

THE UNIVERSITY OF CHICAGO COMPUTER SCIENCE

Computation Institute

Scientific Context & Motivation



- Scientists need software to show and measure structure in large complex image datasets
- Creating new visualization/analysis tools is an essential part of the scientific process

Creating vis/analysis tools is hard to do

Increasing range of:

Imaging **modalities**

Imaging **applications**

Vis & analysis **algorithms**

Scientists need to **rapidly** implement variety of new programs

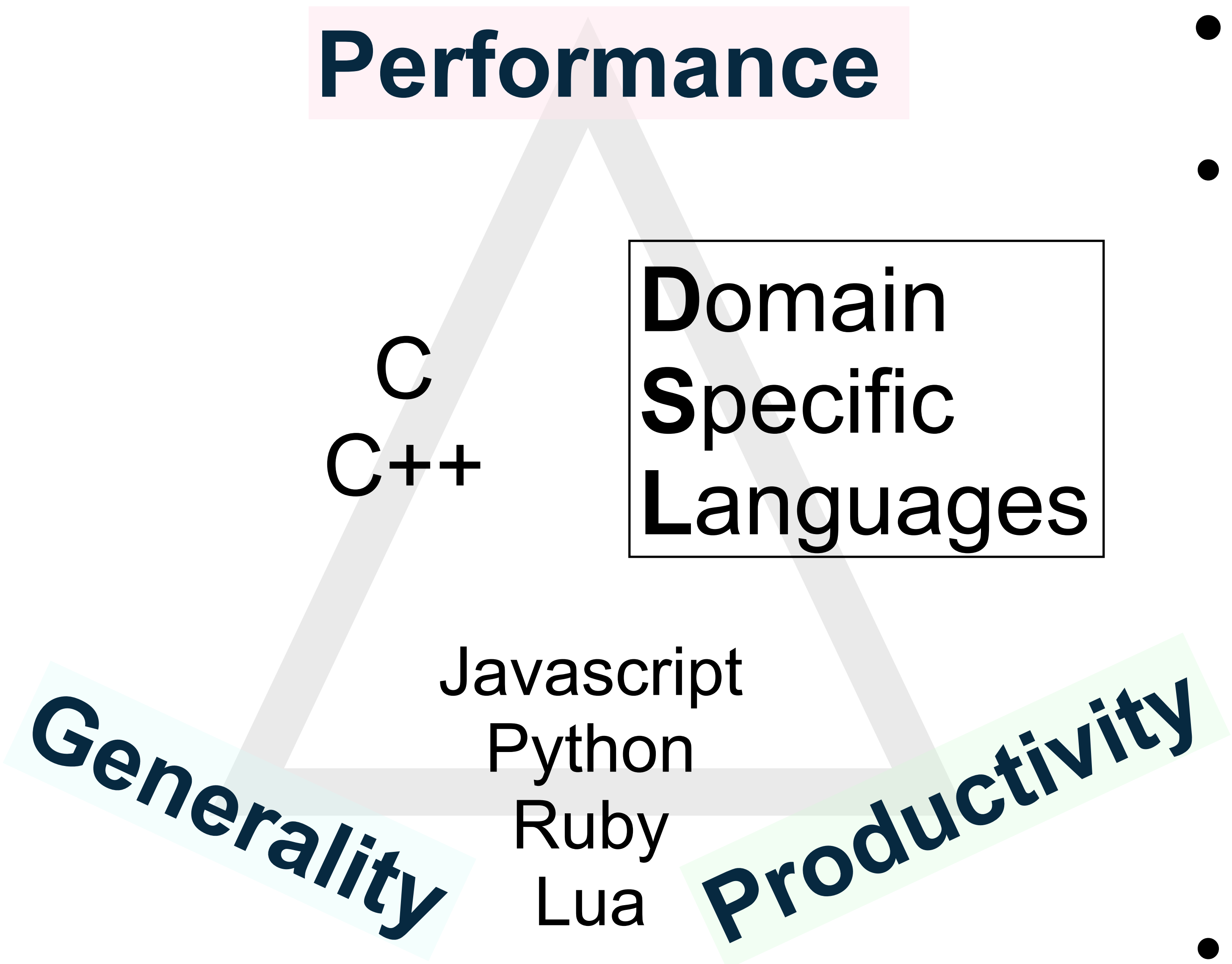
Goal: speed the development of portable parallel methods of 3D scientific visualization and analysis

Programmers want **portable** parallel languages

Increasing **data size** → Need **parallel** computing

Rapidly shifting parallel computing architectures

Triangle of language strengths (courtesy Pat Hanrahan)

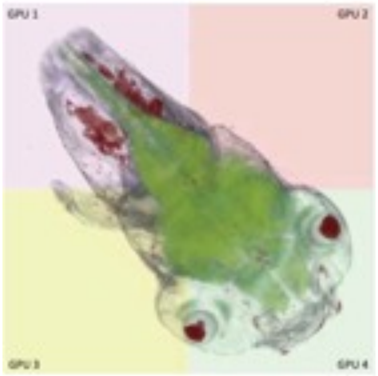


- **“Why not write a library?”**
- DSL advantages:
  1. Code can be concise, idiomatic (types, syntax, operations)
  2. Compiler analysis, optimizations
  3. Express parallel execution apart from OS, hardware (CPU/GPU)
- Expert C/C++ coders like libraries

Goal: Open up Sci Vis research to a larger user community

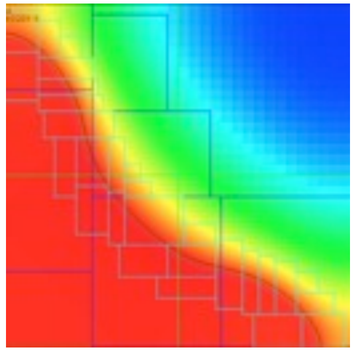
Related DSL research

- **Vivaldi** [Choi-VIS-2014]: Volume rendering, processing in Python-like DSL on distributed GPU clusters



- **ViSlang** [Rautek-VIS-2014]: Slangs (procedural, declarative, functional) interactively combined

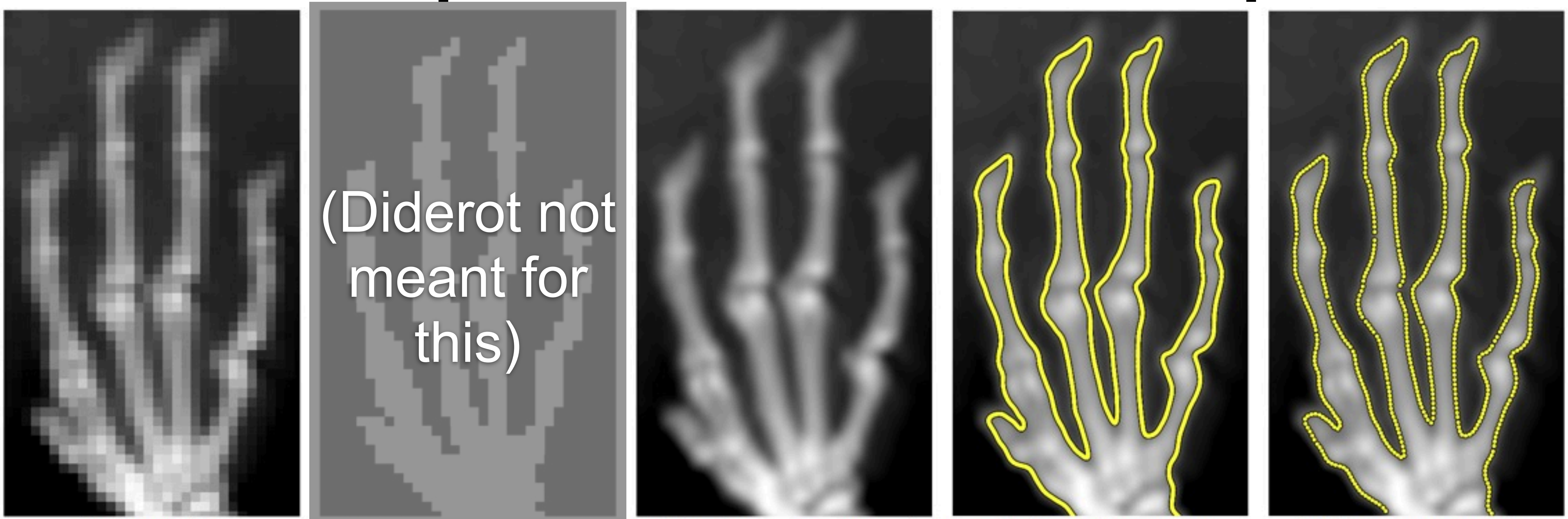
- **Scout** [McCormick-VIS-2004] [McCormick-JPC-2007] [Jablin-IPDPS-2011] [McCormick-WOLFHPC-2014]: compile data- or task-parallel programs on grids, using LLVM toolchain



- Other DSLs discussed in paper
- Diderot's strength: **idiomatic mathematical abstractions**

Data: University of Utah SCI group, NIH NIGMS grant P41GM103545

Diderot computes on fields, not samples



(a) Image data (b) Discrete thresholding (c) Continuous reconstruction (d) Dense isocontour (e) Isocontour particles

- Convolve image data (a) with kernel to get continuous field (c)
- `field#1(2)[] F = ctmr ⊛ image("hand.nrrd");`
- `field#N(D)[S]`: C^N continuous field: $\mathbb{R}^D \rightarrow$ tensors shape S
- []: scalar, [3]: 3-vector, [3,3]: 3x3 matrix (**Appendix A** gives grammar)

Example complete program: isocontour sampling

```
field#1(2)[] F = c4hexic ⊛ image("hand.nrrd");
input int size0; input int size1;
input int stepsMax = 10;
input real epsilon = 0.0001;
input vec2 dir0; input vec2 dir1;
input vec2 orig;
strand isofind(vec2 pos0) {
  output vec2 pos = pos0;
  int steps = 0;
  update {
    // Stop after too many steps or leaving field
    if (steps > stepsMax || !inside(pos, F))
      die;
    // one Newton-Raphson iteration
    vec2 delta = -normalize(∇F(pos)) * F(pos)/|∇F(pos)|;
    pos += delta;
    if (|delta| < epsilon)
      stabilize;
    steps += 1;
  }
}
initially { isofind(orig + ui*dir0 + vi*dir1) |
            vi in 0..(size1-1), ui in 0..(size0-1) };
```

Globals are immutable; used for program inputs

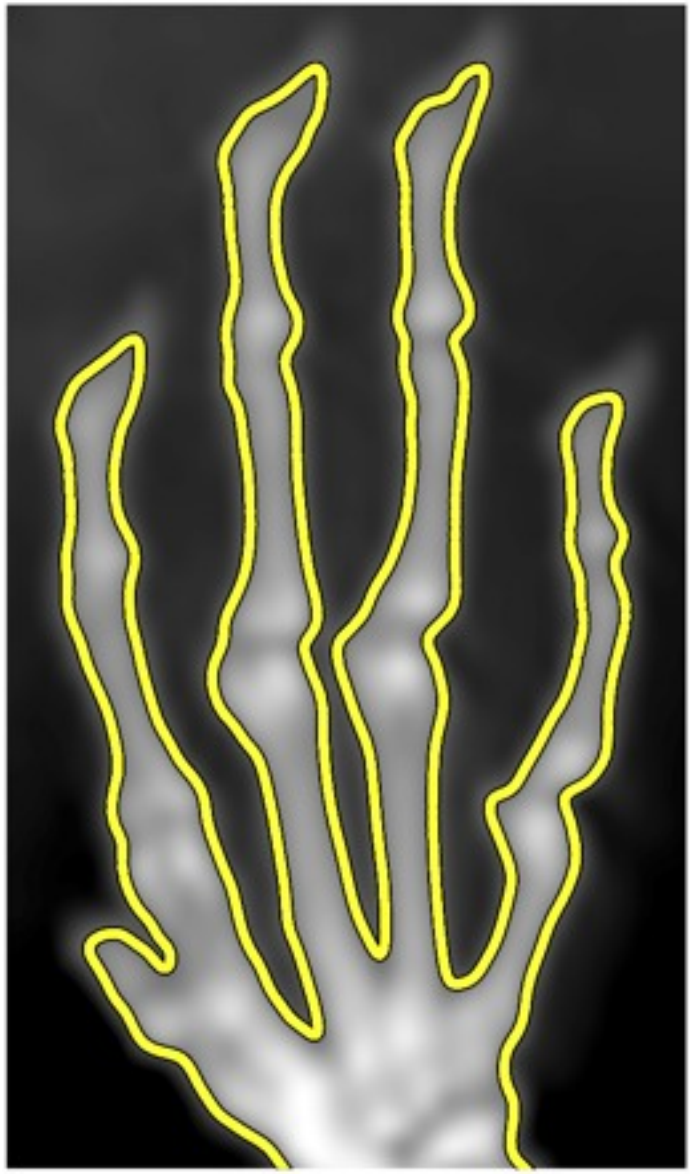
Strands are bulk synchronous

Strand state, including output

update method implements algorithm

Legible math!

Initialization of collection of strands with comprehension notation



Volume rendering soft isosurfaces

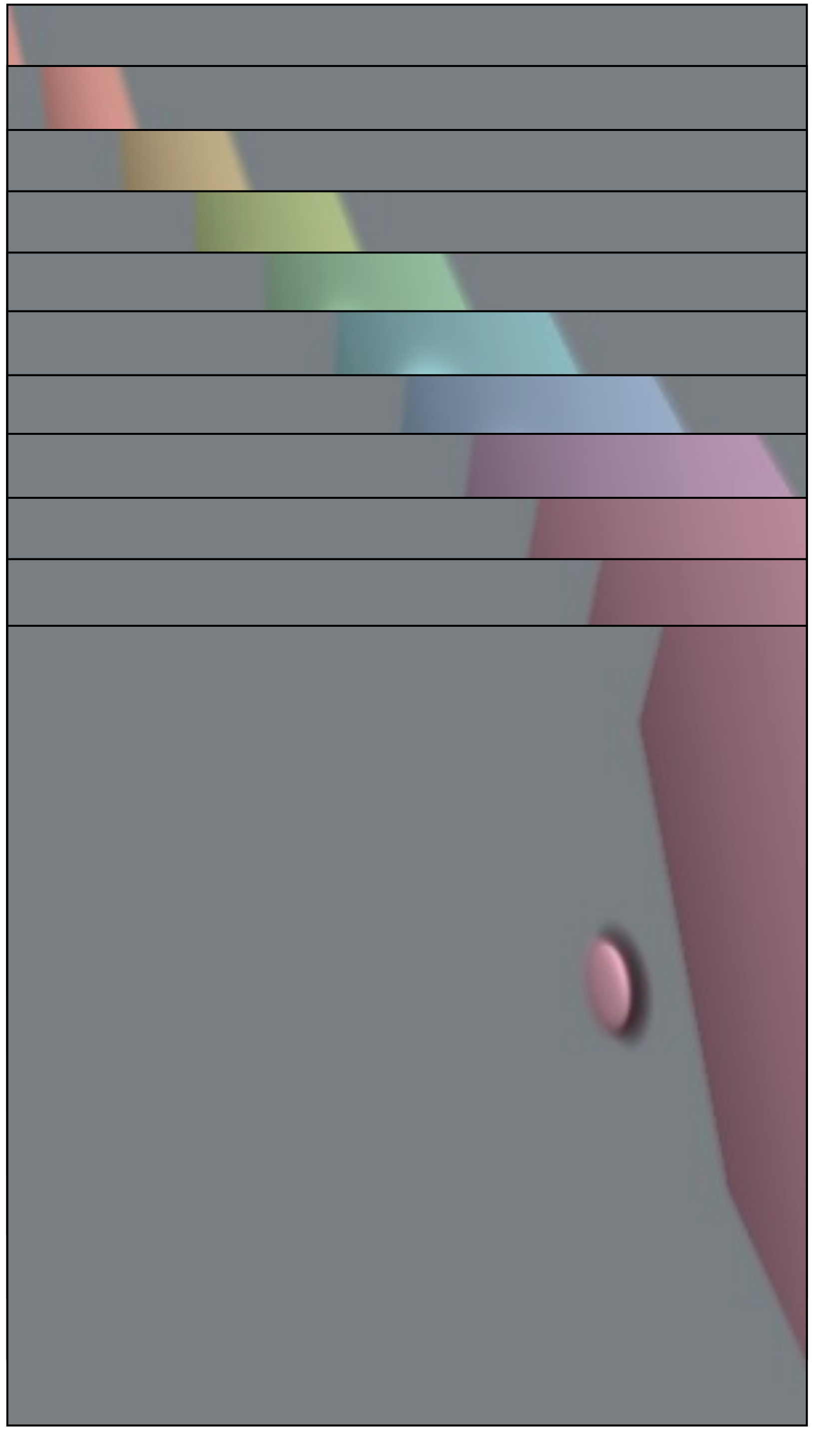
```
field#0(1)[3] cmap = tent ⊛ image("isobow.nrrd");
field#4(3)[] V = bspln5 ⊛ image("canny.nrrd");
field#4(3)[] F = V - isoval;
...
function real alpha(real v, real g) = max(0, 1 - |v|/(g*thick));
...
strand raycast(int ui, int vi) {
  real transp = 1;
  vec3 rgb = [0,0,0]; output vec4 rgba = [0,0,0,0];
  update {
    if (rayN > camVspFar) { stabilize; }
    real val = F(x);
    vec3 grad = -∇F(x);
    real a = alpha(val, |grad|);
    real shade = max(0, normalize(grad)•light);
    rgb += transp*a*(0.2 + 0.8*shade)*color(x);
    transp *= 1 - a;
  }
  stabilize {
    real a = 1-transp;
    if (a > 0) rgba = [rgb[0]/a, rgb[1]/a, rgb[2]/a, a];
  }
}
initially [ raycast(ui, vi)
          | vi in 0..iresV-1, ui in 0..iresU-1 ];
```

Isosurface is zero level-set

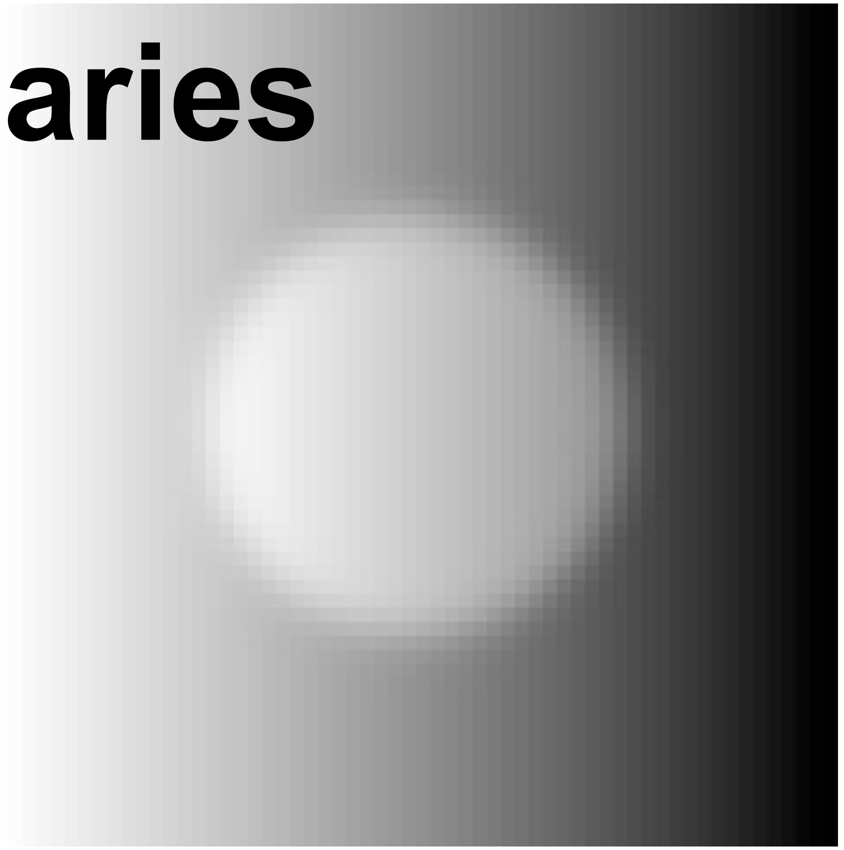
[Levoy-CGnA-1988]

Over operator with pre-multiplied alphas

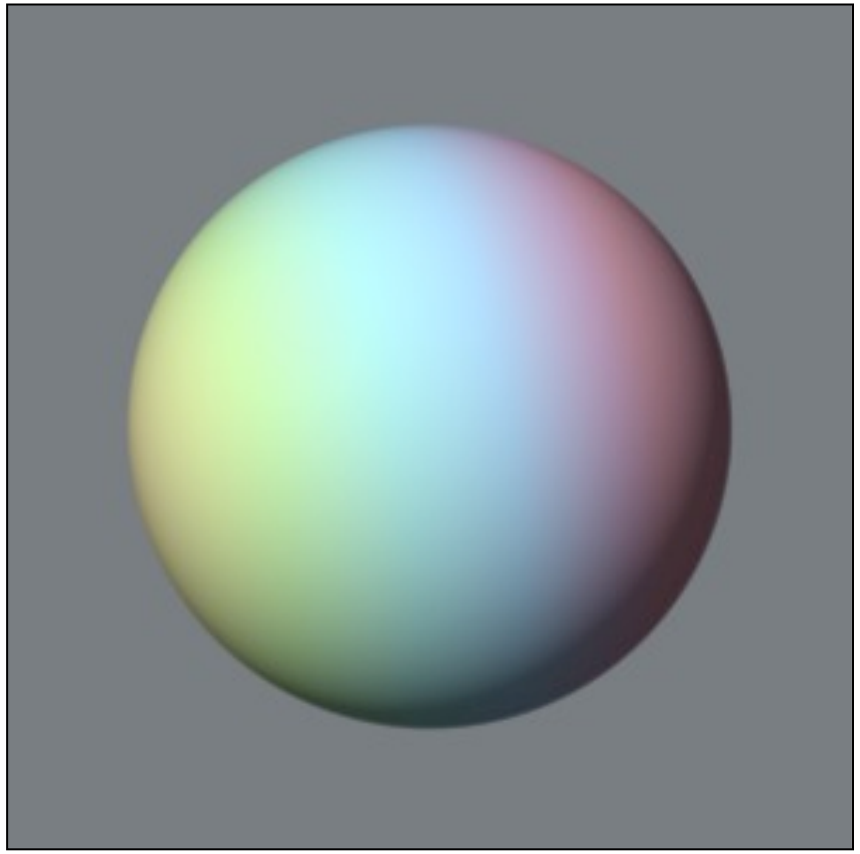
set final output rgba



Volume rendering material boundaries



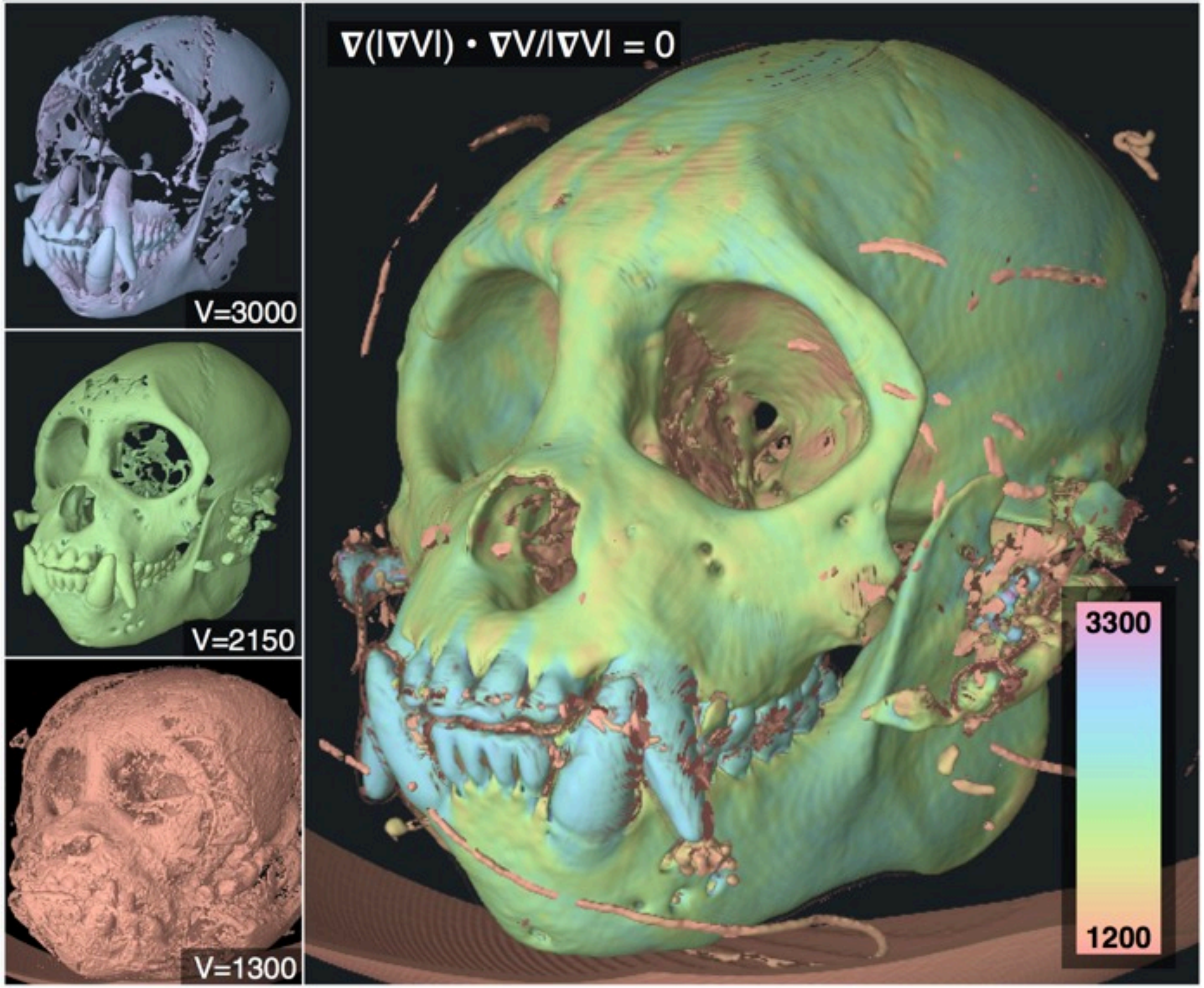
- How to show material boundaries?
- Canny edge [Canny-PAMI-1986]:
  - $|\nabla v|$ maximal w.r.t motion along $\nabla v/|\nabla v|$
  - $\Rightarrow \nabla|\nabla v| \cdot \nabla v/|\nabla v| == 0$
- Change one line of Diderot code:
  - **`field#4(3)[] F = V - isoval;`**
  - **`field#2(3)[] F = ∇|∇V| • ∇V/|∇V|;`**
- For shading, Diderot computes ∇F
  - involves 3rd derivatives (!)



Canny edges in real CT scan

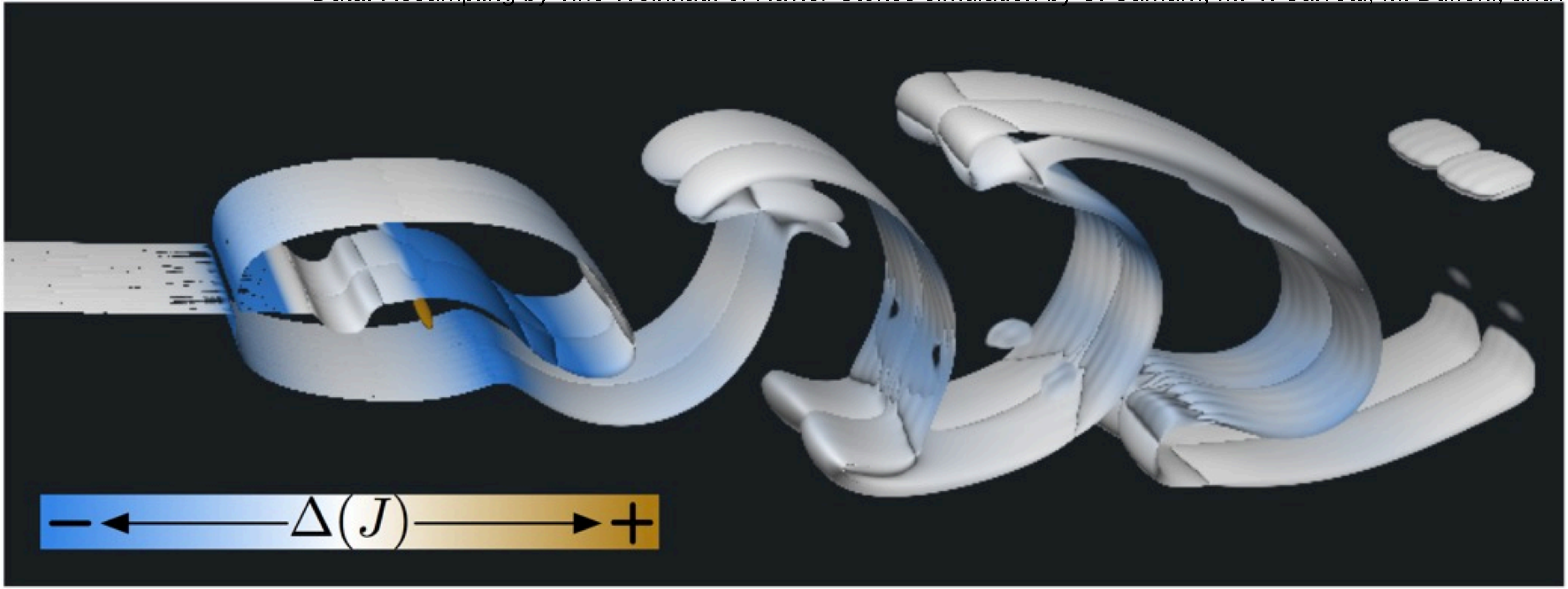
Data: Callum Ross, University of Chicago

- There is no isosurface that captures the bone surface
- Canny edge surface shows underlying value (novel vis)



Rendering flow field structure

Data: Resampling by Tino Weinkauf of Navier-Stokes simulation by S. Camarri, M.-V. Salvetti, M. Buffoni, and A. Iollo



- `field#4(3)[3] V = bspln5 ⊛ image("flow.nrrd");`
- `field#3(3)[] F = (V/|V|) • (∇×V/|∇×V|);`
  - Normalized Helicity [Degani-AIAAJ-1990]

Rendering anisotropy of diffusion tensor field

```
field#4(3)[3,3] V = bspln5 ⊛ image("dti.nrrd");
field#4(3)[3,3] E = V - trace(V)*identity[3]/3;
field#4(3)[] F = sqrt(3.0/2.0)*|E|/|V| - isoval;
```

Compare with original definition [Basser-JMRB-1996]

$$\underline{\boldsymbol{D}} = \underline{\mathbf{D}} - \langle \mathrm{D} \rangle \underline{\mathbf{I}}.$$

$$\mathrm{FA} = \sqrt{\frac{3}{2}} \frac{\sqrt{\underline{\boldsymbol{D}} : \underline{\boldsymbol{D}}}}{\sqrt{\underline{\mathbf{D}} : \underline{\mathbf{D}}}}.$$

Not just for volume rendering!

Data: Centre for Functional MRI of the Brain, John Radcliffe Hospital, Oxford University

Streamlines in flow field

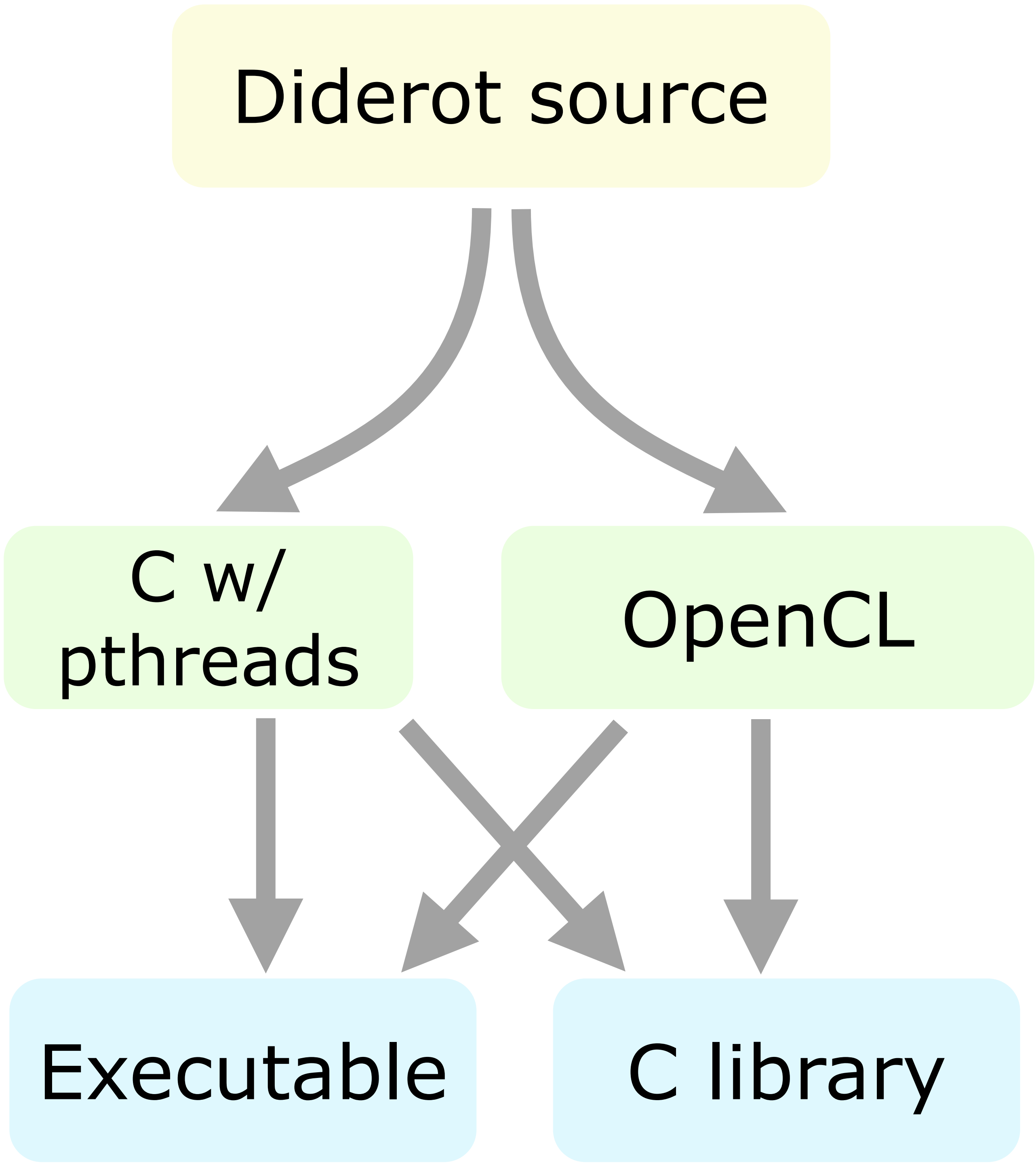
Data: Wolfgang Kollmann, UC Davis

```
vec2{} x0s = load("seeds.txt"); // list of seedpoints
real h = 0.02;
int stepNum = 200;
field#1(2)[2] V = bspln3 ⊛ image("flow.nrrd");
real arrow = 0.1;   // scale from |V(x)| to arrow size
strand sline(vec2 x0) {
  int step = 0;
  vec2 x = x0;
  output vec2{} p = {x0}; // start streamline at seed
  update {
    if (inside(x, V)) {
      x += h*V(x + 0.5*h*V(x));  // Midpoint method
      p = p @ x;    // append new point to streamline
    }
    step += 1;
    if (step == stepNum) {
      // finish streamline with triangular arrow head
      vec2 a = arrow*V(x);    // length of arrow head
      vec2 b = 0.4*[-a[1],a[0]]; // perpendicular to a
      p = p@(x-b); p = p@(x+a); p = p@(x+b); p = p@x;
      stabilize;
    }
  }
}
initially [ sline(i, x0s{i}) | i in 0..length(x0s)-1 ];
```

Output is set of sequence of points

Legible integration

Compilation

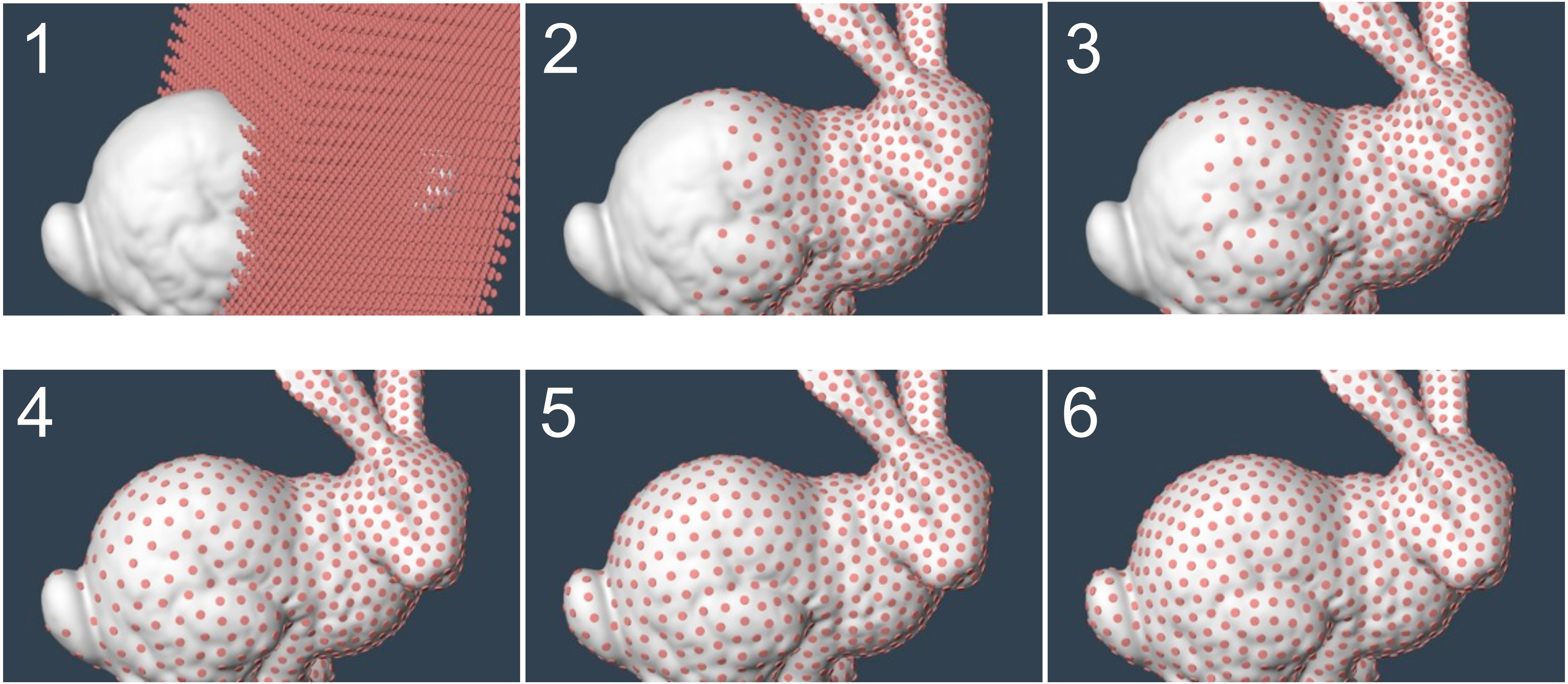


- Compiler written in SML/NJ
- Three stages of intermediate representation (IR)
  - "EIN" IR is like lambda calculus meets Einstein summation notation
  - Produces identities:
    - $\nabla \bullet (\nabla \times V) = 0$
    - $\mathrm{Trace}(u \otimes v) = u \bullet v$
  - Section 5.1 of paper
- Use **`clang`** to compile executable or C library

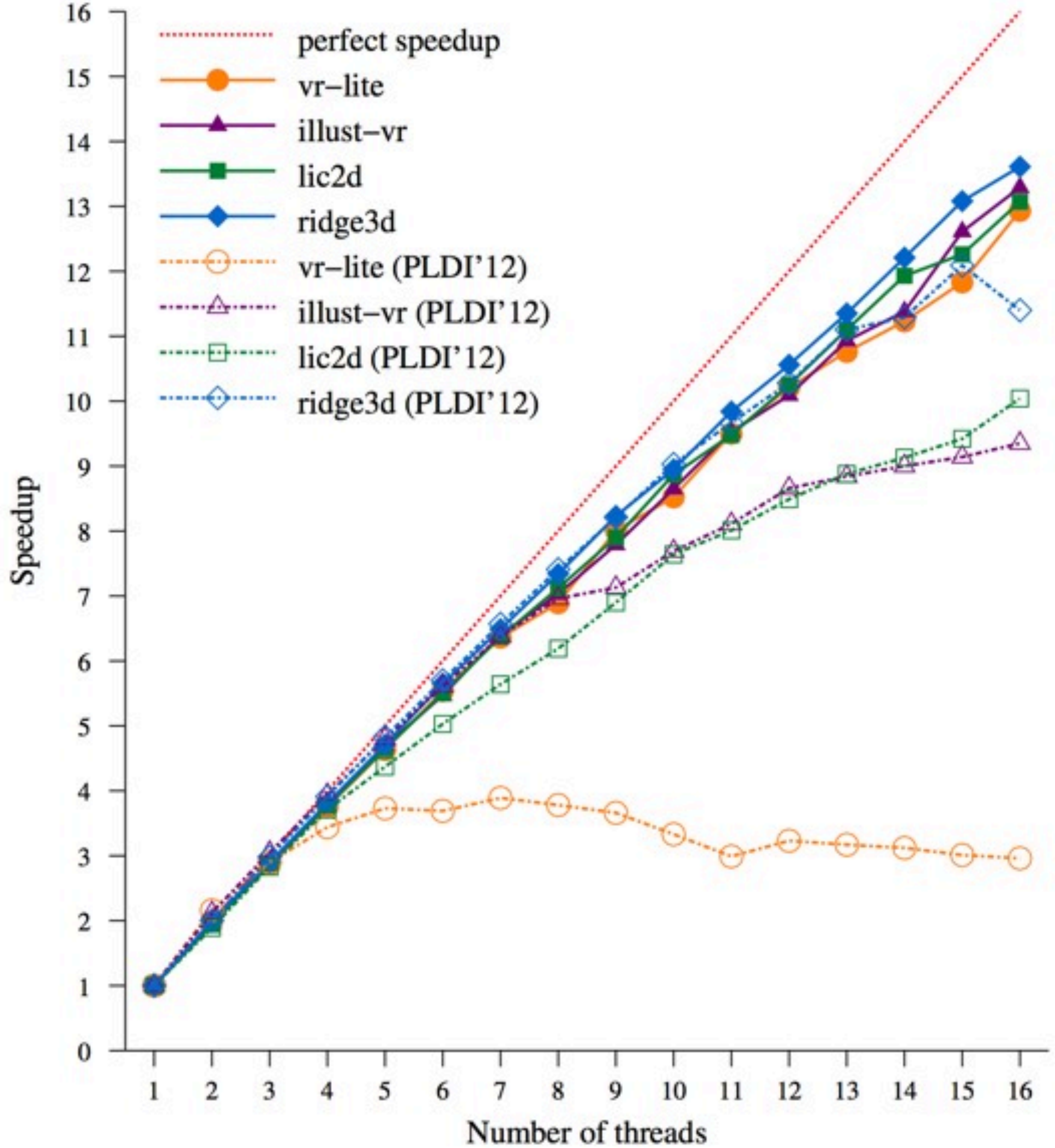
Compile to executable or C Library

- Stand-alone executable w/ command-line interface
  - each input has corresponding option
    - `input real isoval = 10;` ⇒ ... `-isoval 10` ...
- Compile to library, with API for
  - Setting inputs, retrieving outputs
    - `ISO_InVarSet_isoval(ISO_World_t *wrld, float v);`
    - `ISO_OutputGet_pos(ISO_World_t *wrld, Nrrd *data);`
  - Initializing, stepping through computation
- **Appendix B**: 2D particle system example
- Let’s watch 3D particle system go ...

(snapshots from interactive demo shown during talk)



Speedup curves (on CPU)



- Significant improvement in speedup relative to previous 2012 paper in Programming Language Design and Implementation (PLDI)

Performance numbers

Program	Teem	Diderot (PLDI '12) Seq.	1P	6P	12P	16P	Diderot (this paper) Seq.	1P	6P	12P	16P	OpenCL
vr-lite	19.93	8.63	9.51	2.57	2.94	3.20	7.46	7.52	1.36	0.74	0.59	1.43
illust-vr	86.16	44.30	48.55	8.65	5.61	5.19	38.12	38.28	7.00	3.79	2.88	4.32
lic2d	3.03	1.59	1.64	0.33	0.19	0.16	1.56	1.51	0.28	0.15	0.12	1.09
ridge3d	7.92	5.96	6.36	1.12	0.62	0.56	5.22	5.26	0.93	0.50	0.39	1.77

Execution times in seconds, averaged over 10 runs

- "Teem" = hand-coded C, not parallel (no pthreads)
- Intel Xeon E5-2687W (16 cores), Ubuntu 12.04.
- OpenCL w/ NVIDIA Tesla K20c, using NVIDIA's CUDA 6.0 driver
- **Appendix C** compares with hand-written OpenCL

Ongoing Work

- Stronger math abstractions
  - Declarative mathematical statement of algorithm
  - Time-varying fields (time as special dimension)
- Better computing
  - New backends: CUDA and MPI (for larger datasets)
  - Better GPU performance through OpenCL
  - New fields: (higher-order) Finite Element Meshes
- Better usability: debugger, GUI generation

Conclusions

- Good progress on an ambitious goal
- Diderot good for:
  - Writing **legible** vis programs that run in parallel
  - Trying new sci vis methods in terms of fields, tensors
- Diderot not (yet) good for:
  - Working directly on grids (e.g. Marching Cubes, level-set segmentation, per-pixel classification)
  - Fast execution on big data essential, rather than fast implementation

Works cited

- [Choi-VIS-2014] H. Choi, W. Choi, T. M. Quan, D. G. C. Hildebrand, H. Pfister, and W.- K. Jeong. Vivaldi: A domain-specific language for volume processing and visualization on distributed heterogeneous systems. IEEE Trans. Vis. Comp. Graph. (Proc. SciVis), 20(12):2407–2416, Dec. 2014.
- [Rautek-VIS-2014] P. Rautek, S. Bruckner, M. E. Gröller, and M. Hadwiger. ViSlang: A system for interpreted domain-specific languages for scientific visualization. IEEE Trans. Vis. Comp. Graph. (Proc. SciVis), 20(12):2388–2396, Dec. 2014
- [McCormick-VIS-2004] P. McCormick, J. Inman, J. P. Ahrens, C. Hansen, and G. Roth. Scout: A hardware-accelerated system for quantitatively driven visualization and analysis. In Proceedings of IEEE Visualization 2004, pages 171–178, 2004
- [McCormick-JPC-2007] P. McCormick, J. Inman, J. Ahrens, J. Mohd-Yusof, G. Roth, and S. Cummins. Scout: A data-parallel programming language for graphics processors. J. Par. Comp., 33:648–662, Nov. 2007.
- [Jablin-IPDPS-2011] J. Jablin, P. McCormick, and M. Herlihy. Scout: High-performance heterogeneous computing made simple. In Proceedings of IEEE International Symposium on Parallel and Distributed Processing, pages 2093–2096, 2011
- [McCormick-WOLFHPC-2014] P. McCormick, C. Sweeney, N. Moss, D. Prichard, S. K. Gutierrez, K. Davis, J. Mohd-Yusof. Exploring the Construction of a Domain-Aware Toolchain for High-Performance Computing. Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing (WOLFHPC). pages 1–10, 2014.
- [Levoy-CGnA-1988] Display of surfaces from volume data. IEEE Computer Graphics & Applications, 8(5):29–37, 1988.
- [Canny-PAMI-1986] A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell., 8(6):679–714, 1986.
- [Degani-AIAAJ-1990] D. Degani, Y. Levy, and A. Seginer. Graphical visualization of vortical flows by means of helicity. AIAA Journal, 28:1347–1352, Aug. 1990.
- [Basser-JMRB-1996] P. J. Basser and C. Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI. J. Mag. Res., B, 111:209–219, 1996.

Thank you

- Anonymous reviewers for constructive comments
- National Science Foundation CCF-1446412
- Data: University of Utah SCI group, NIH NIGMS grant P41GM103545 | Callum Ross, University of Chicago | Resampling by Tino Weinkauf of Navier-Stokes simulation by S. Camarri, M.-V. Salvetti, M. Buffoni, and A. Iollo | Xavier Tricoche, Purdue University | Centre for Functional MRI of the Brain, John Radcliffe Hospital, Oxford University | Wolfgang Kollmann, UC Davis
- Reproducibility! (Even before a release...)
- Example programs from this talk will be here:

  `https://github.com/Diderot-Language/examples`
- **Google Group: `https://goo.gl/kXpxhV`**
- Questions?