CPSC 446/546

Data and Info Visualization

Lecture #18 : Spatial Data/Network Data

Outline

- Announcements
- Spatial Data
- Network Data

Announcements

- Assignment 4 available, due Nov. 9
- Exams with printed answer key will be returned at end of class (scores on Post'em with midterm status info.)

17	31-Oct	Arrange Spatial Data		Munzner chap 8
18	2-Nov	Networks and Trees		Munzner chap 9
19	7-Nov	GeoVis -- Miriam Olivares		
20	9-Nov	Manipulate View/Facet into Multiple Views	**Assign 4 due**,Assign 5 available,	Munzner chap 11,12
21	14-Nov	Visualization in Medical Research-- Kim Blenman		
22	16-Nov	Reduce Items and Attributes	Assign 6 available	Munzner chap 13
	21-Nov	THANKSGIVING BREAK		
		THANKSGIVING BREAK		
23	28-Nov	Embed Focus+ Context	**Assign 5 due**	Munzner chap 14
24	30-Nov	Guest Speaker- Anjali Singhvi		online articles
25	5-Dec	Review		
26	7-Dec	**Midterm Exam 2**		
	15-Dec	end of reading period	**Assign 6 due**	

For Guest Speaker Lectures:

- Attendance required
- No slides will be posted -- come and take notes!
- There will be questions (easy if you attend the class) on the second midterm about each of the guest lectures (Nov 7, Nov 14, Nov 30)

Arrange Spatial Data

Arrange Spatial Data

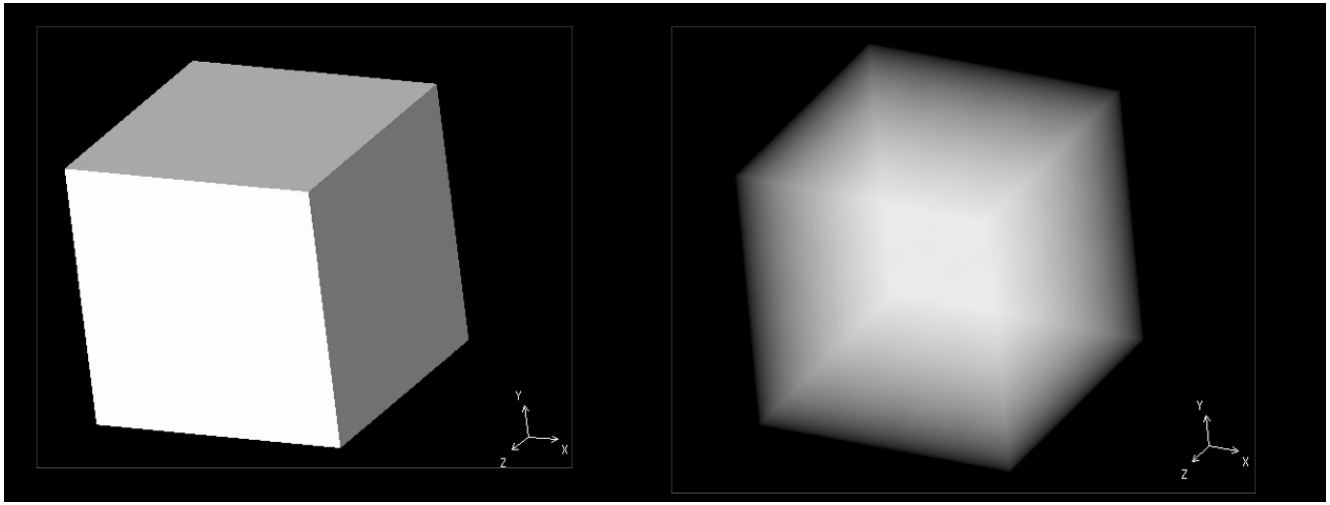
- Use Given
  - Geometry
    - *Geographic*
    - *Other Derived*

  - Spatial Fields
    - *Scalar Fields (one value per cell)*
      - *Isocontours*
      - *Direct Volume Rendering*

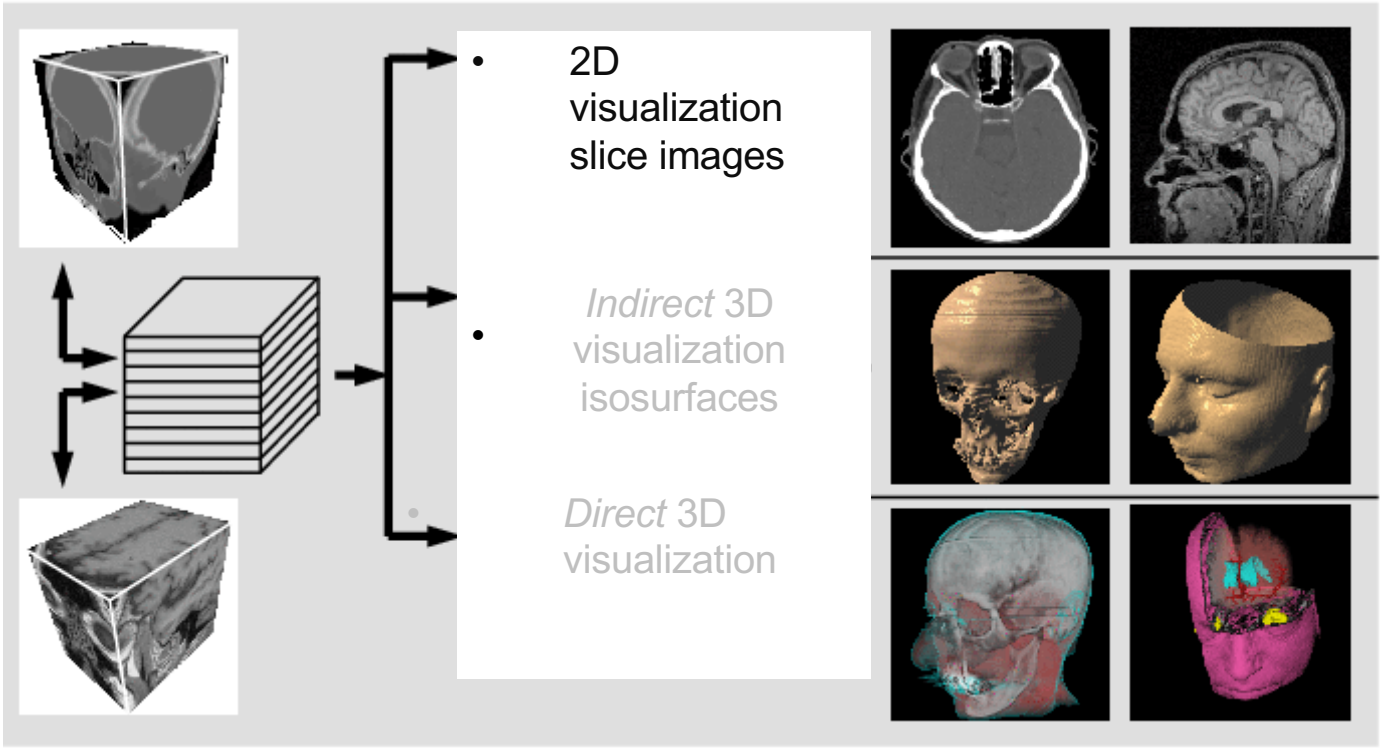
    - *Vector and Tensor Fields (many values per cell)*
      - *Flow Glyphs (local)*
      - *Geometric (sparse seeds)*
      - *Textures (dense seeds)*
      - *Features (globally derived)*

Scalar Fields

- A volume filled with a continues value – e.g. density within a cloud.

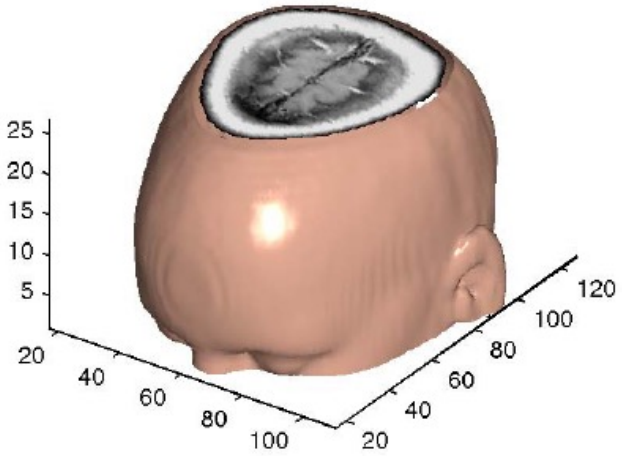


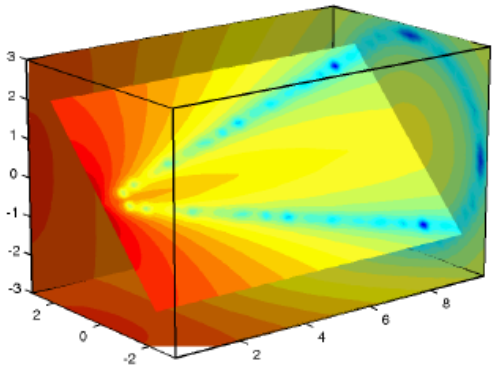
Visualizing Volume (3D) Data



[© Weiskopf/Machiraju/Möller]

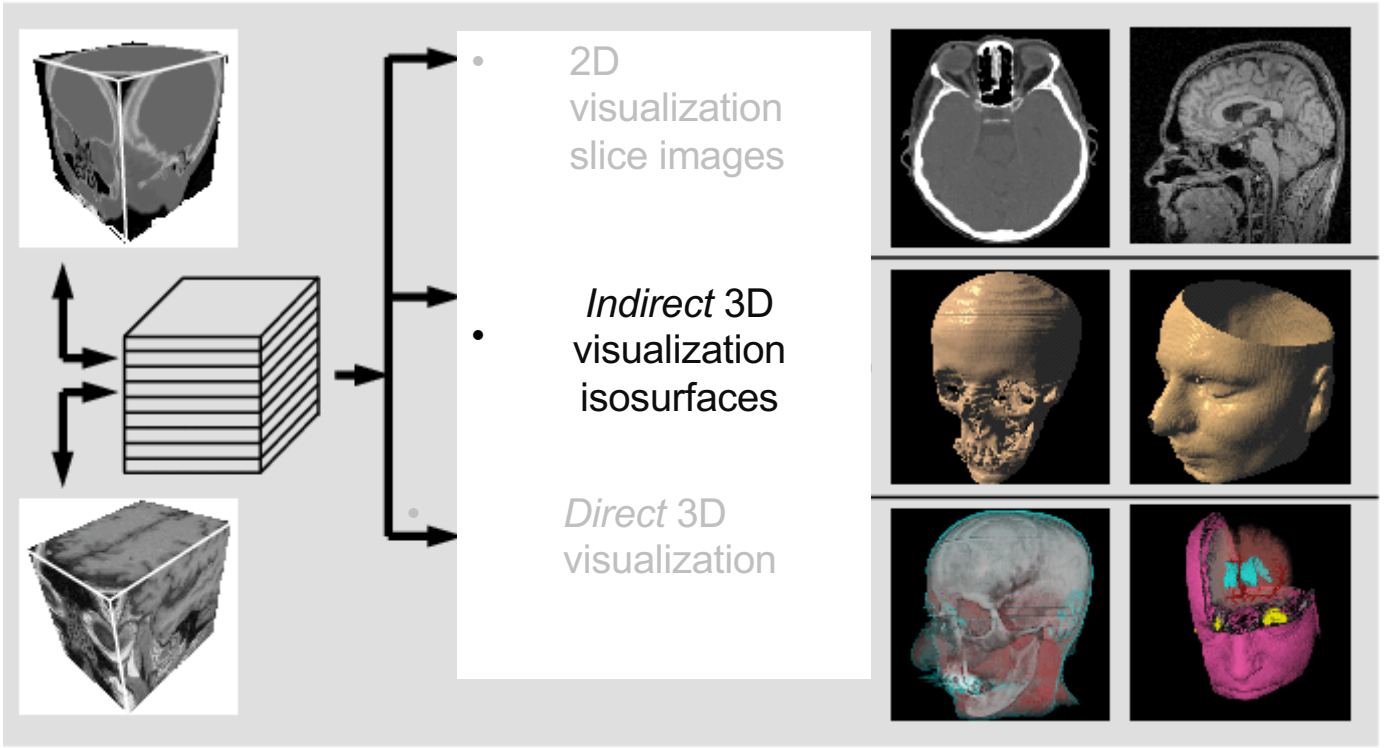
Scalar Fields -- Slices





Reduce to coded values on plane – show the plane in context

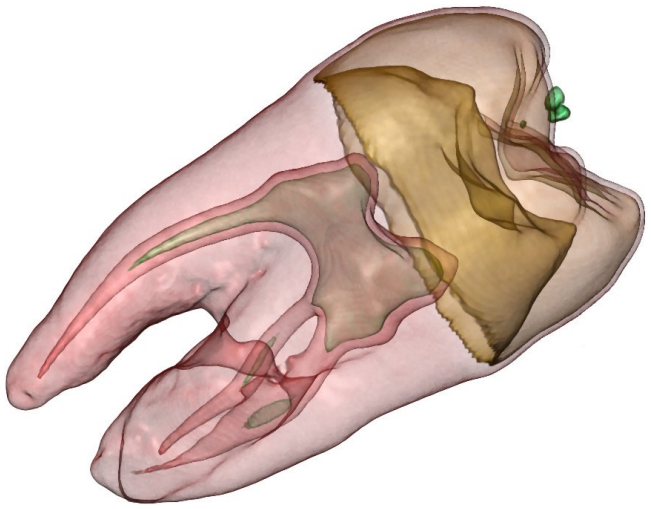
Visualizing Volume (3D) Data



[© Weiskopf/Machiraju/Möller]

Isosurfacing

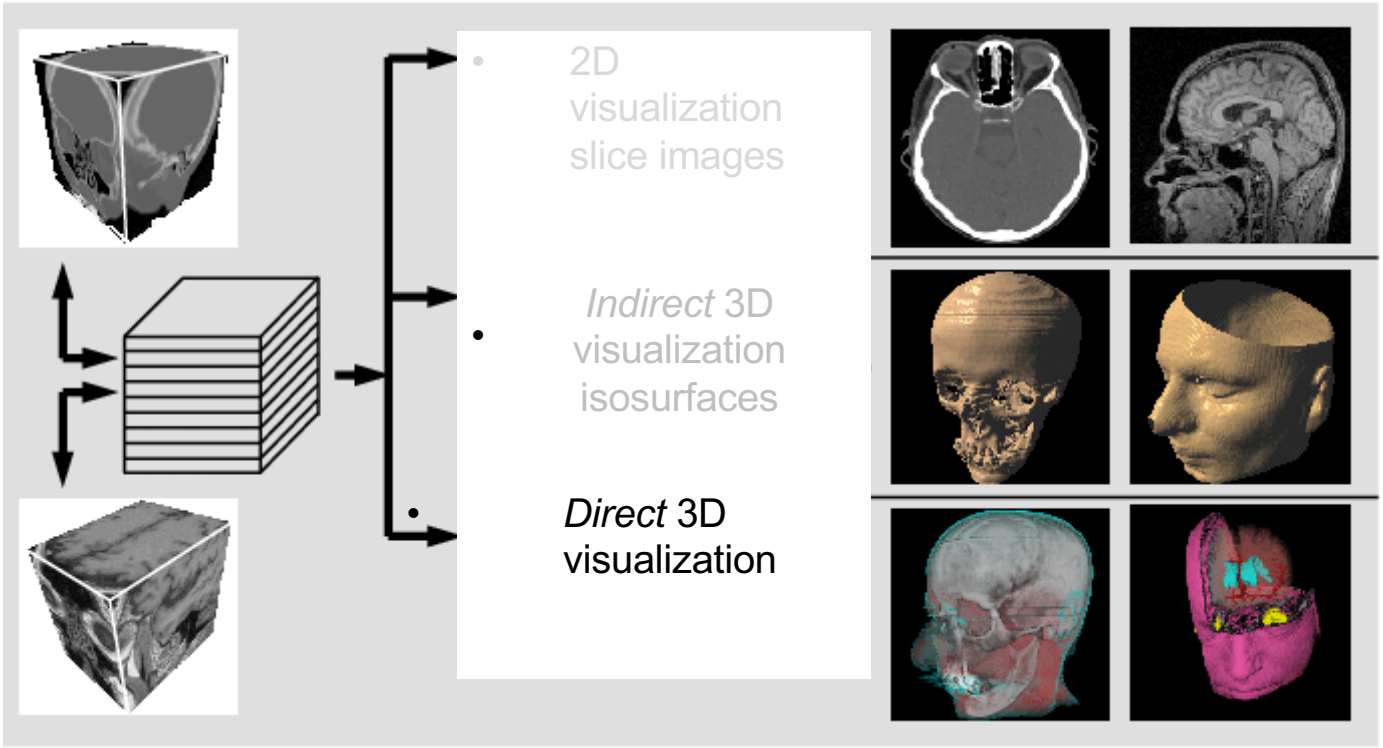
(a) An isosurfaced tooth.



(b) Multiple isosurfaces.

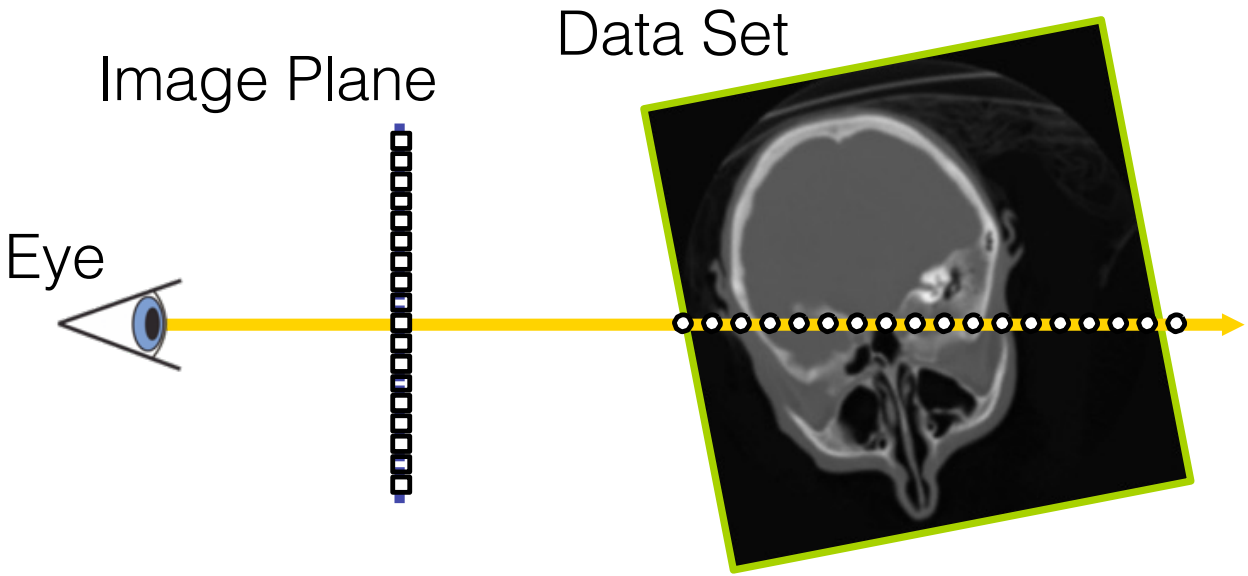
[J. Kniss, 2002]

Visualizing Volume (3D) Data



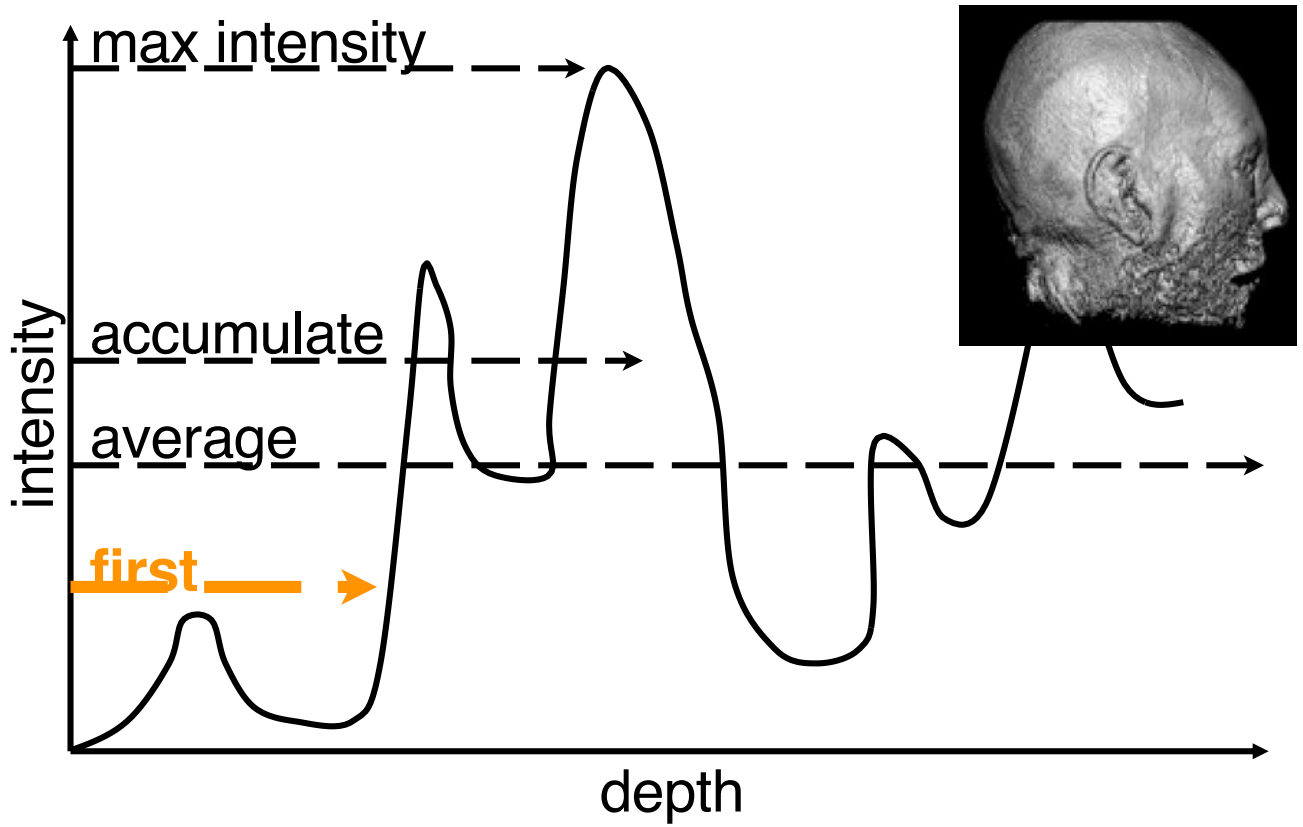
[© Weiskopf/Machiraju/Möller]

Volume Ray Casting

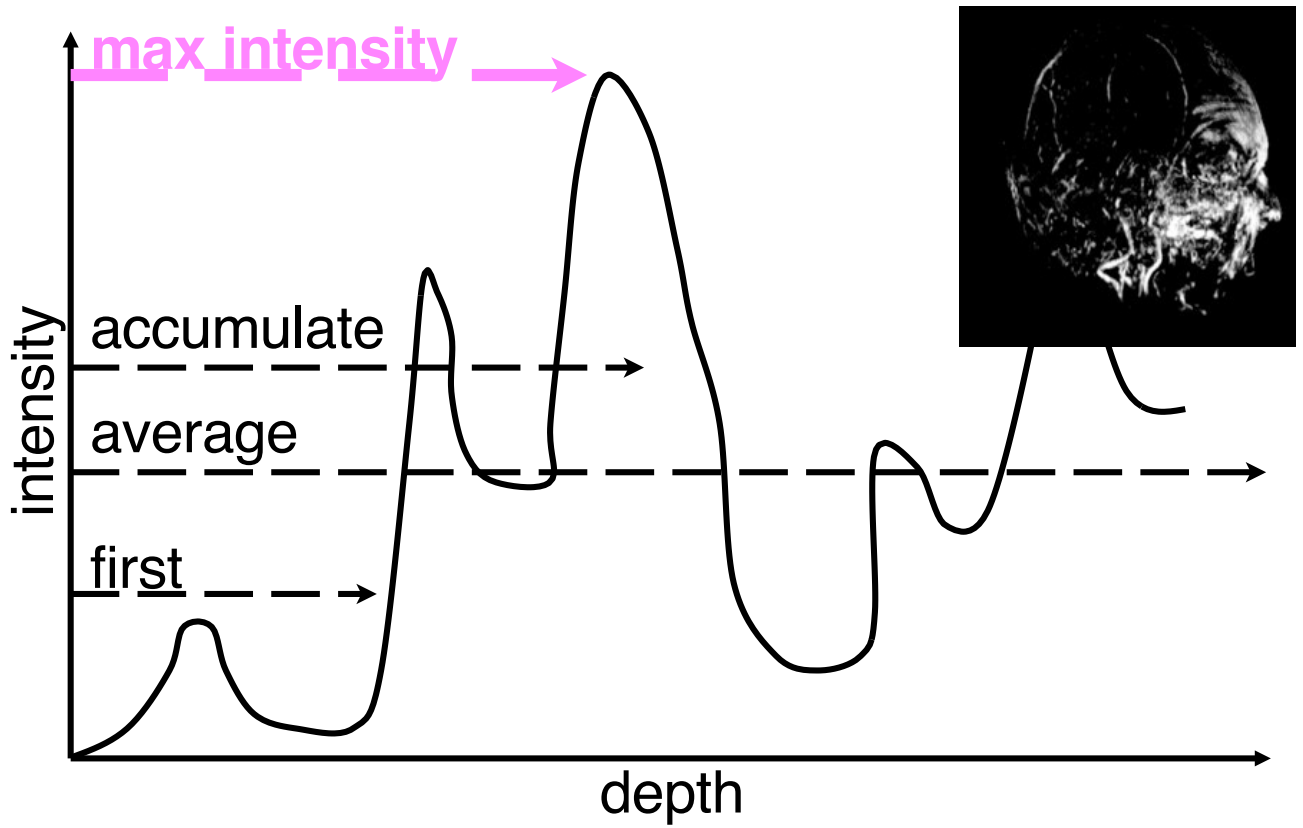


Use orthographic projection in Scivis

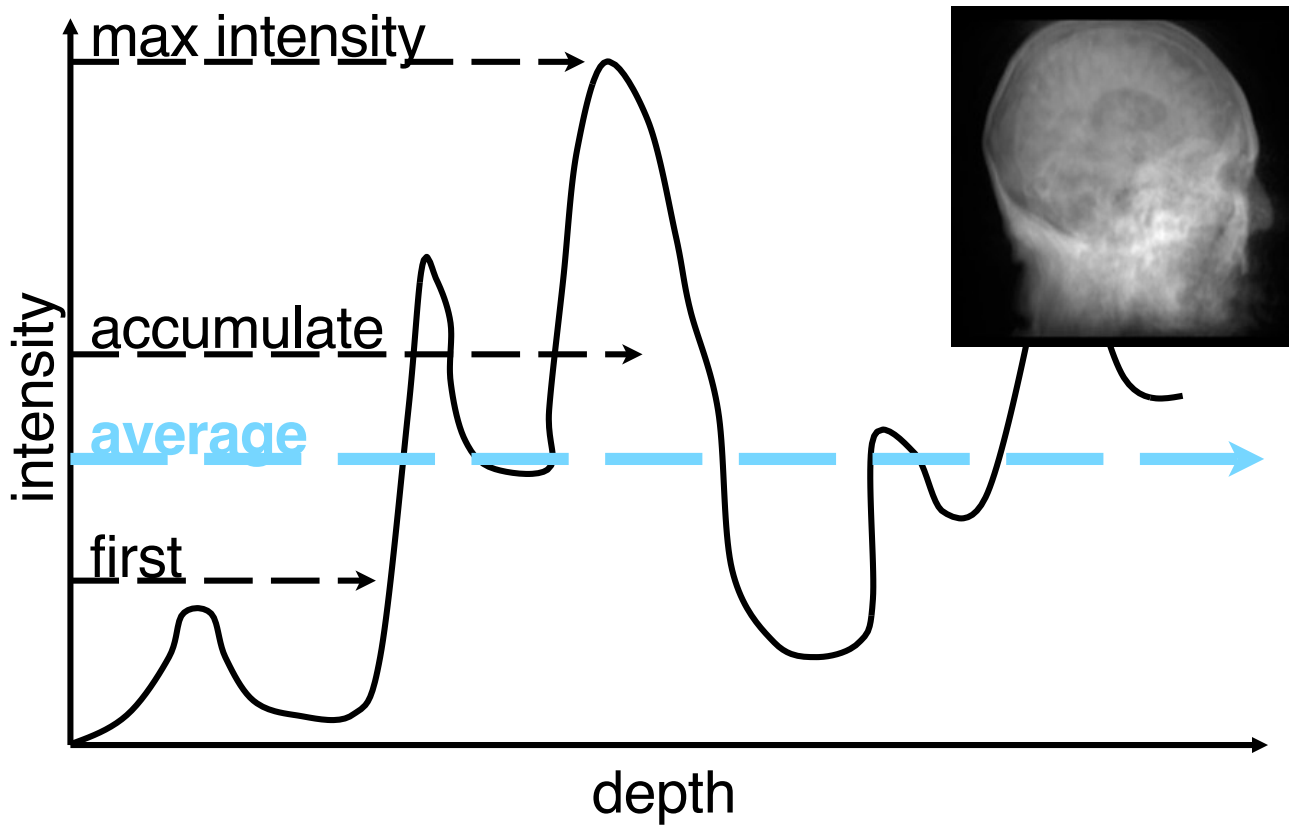
Types of Compositing



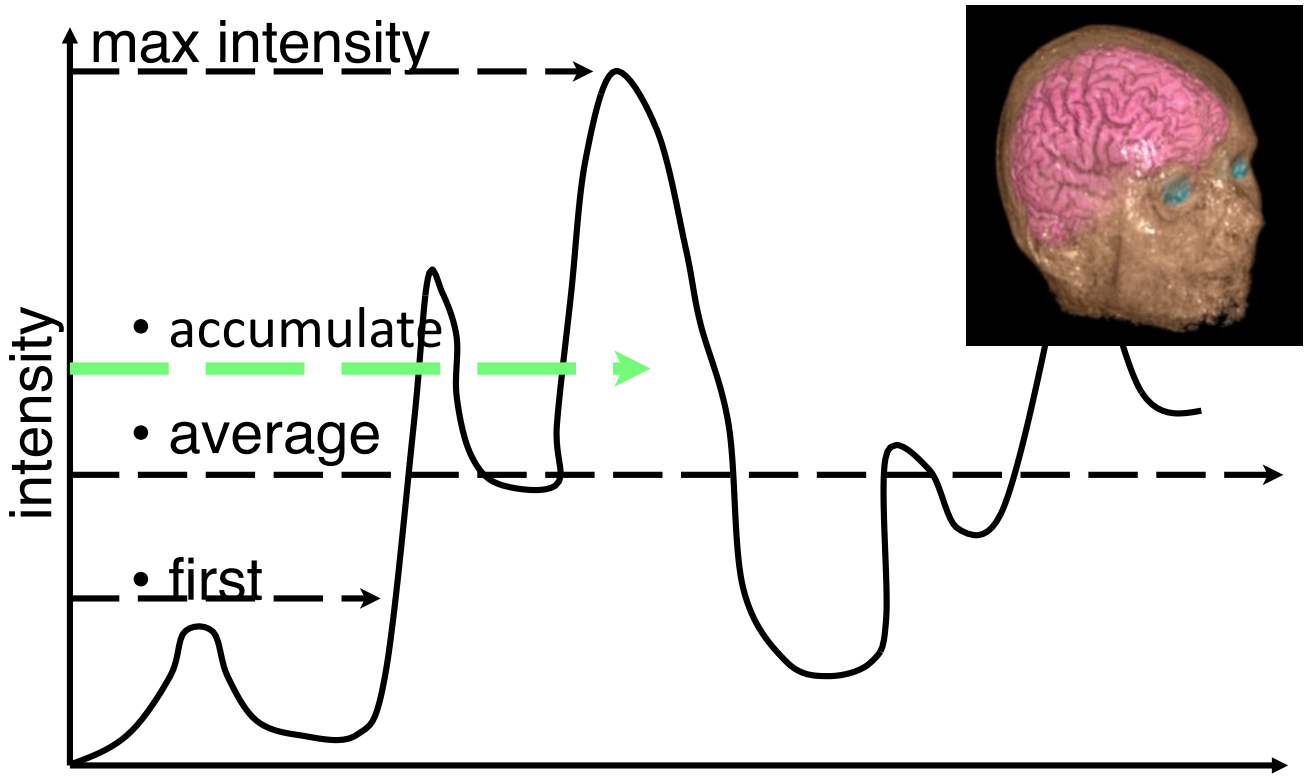
Types of Compositing



Types of Compositing

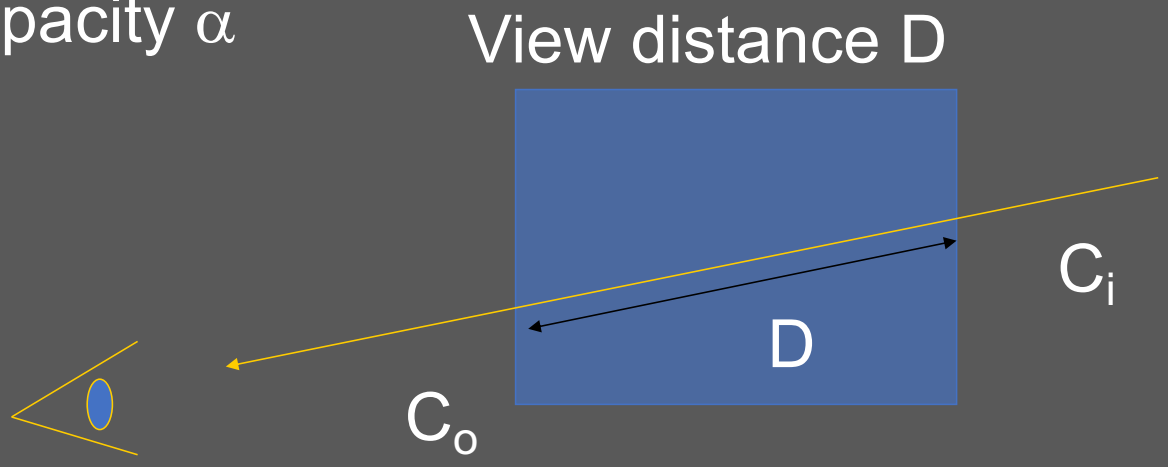


Types of Compositing



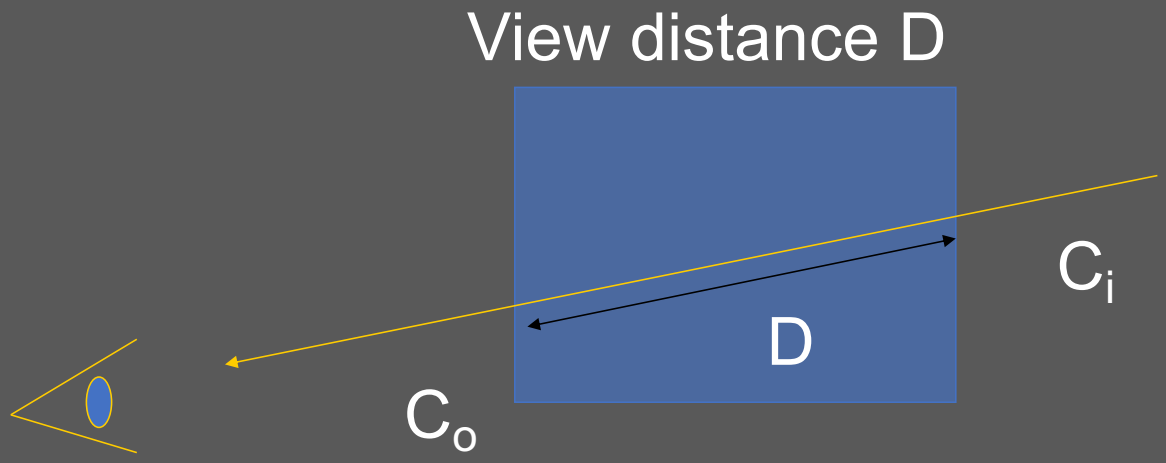
Compositing for Volume Visualization

Voxel has color C
And opacity α



$$C_o = C_i (1-\alpha D) + C\alpha D$$

Compositing for Volume Visualization

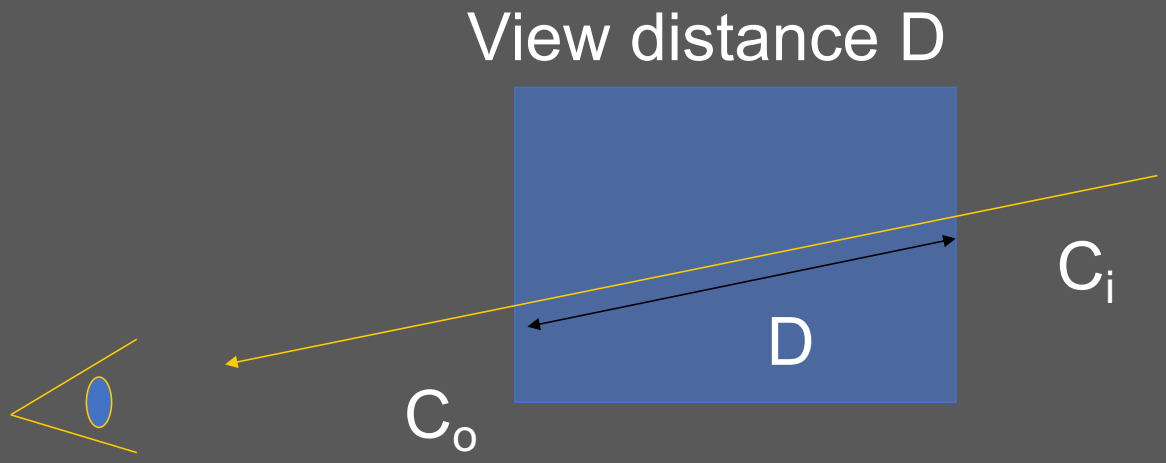


Parameter defined so that $0 < \alpha D < 1$

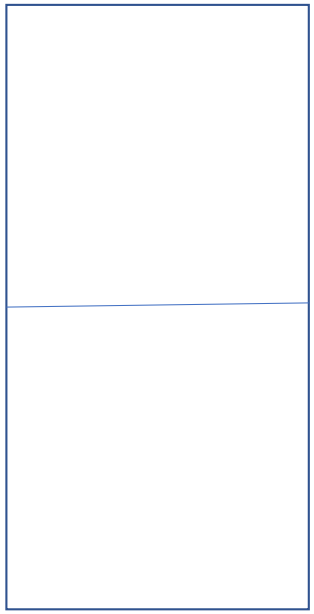
$$C_o = C_i (1-\alpha D) + C\alpha D$$

If α is large, only see color of voxel, if α is very small, voxel nearly invisible

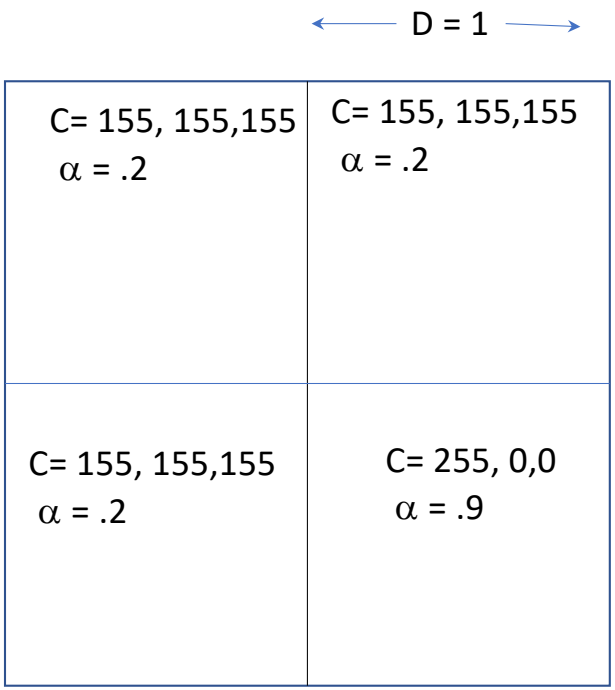
Compositing for Volume Visualization



C_i Is either black, or the color accumulated from passing through other volumes



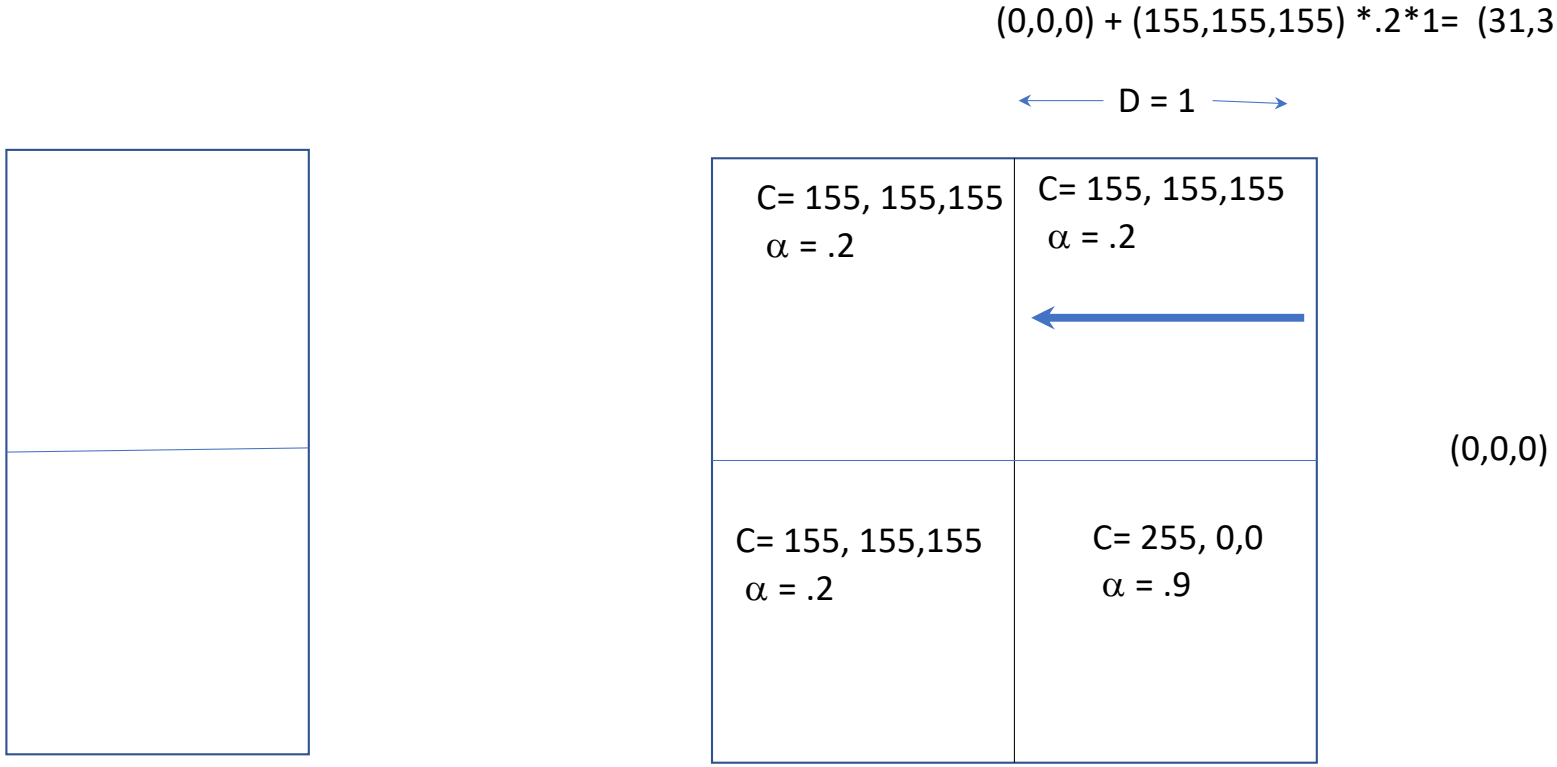
Pixels to compute



Side view one slice of volume

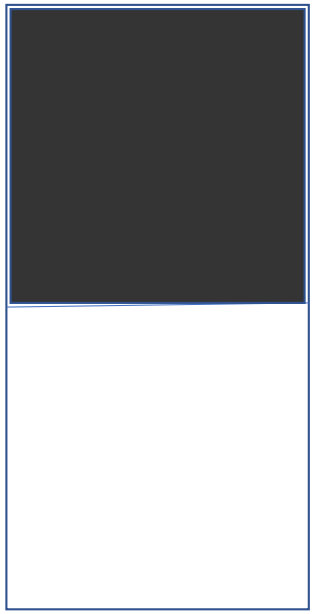
Initial Color Is

(0,0,0) + (155,155,155) *.2*1= (31,31,31)



$$C_o = C_i (1-\alpha D) + C\alpha D$$

(31,31,31)* .8 + (155,155,155) *.2*1= (52,52,52)



D = 1

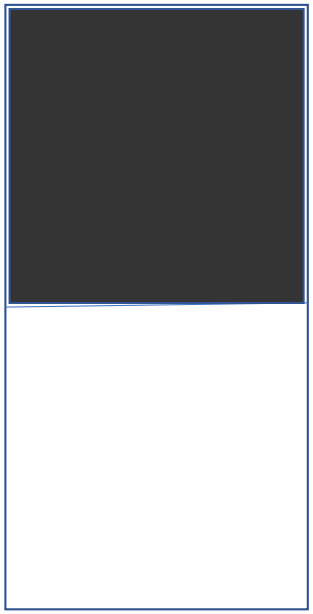
C= 155, 155,155 α = .2	C= 155, 155,155 α = .2
C= 155, 155,155 α = .2	C= 255, 0,0 α = .9

(0,0,0)

$$C_o = C_i (1-\alpha D) + C\alpha D$$

(31,31,31)* .8 + (155,155,155) *.2*1= (52,52,52)

D = 1



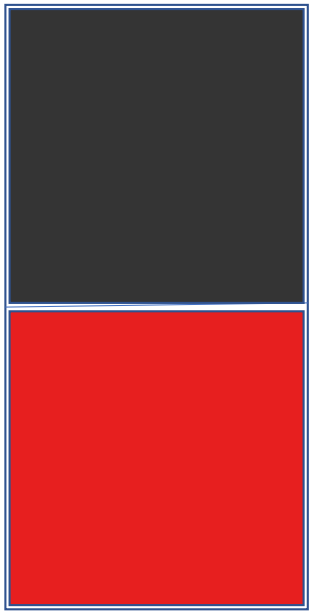
C= 155, 155,155 α = .2	C= 155, 155,155 α = .2
C= 155, 155,155 α = .2	C= 255, 0,0 α = .9

(0,0,0)

(255,0,0) *.9 = (230,0,0)

$$C_o = C_i (1-\alpha D) + C\alpha D$$

(31,31,31)* .8 + (155,155,155) *.2*1= (52,52,52)



D = 1

C= 155, 155,155 α = .2	C= 155, 155,155 α = .2
C= 155, 155,155 α = .2	C= 255, 0,0 α = .9

(0,0,0)

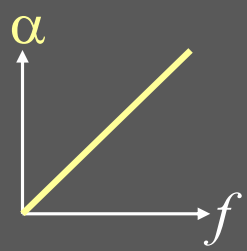
(230,0,0)*.8 + (155,155,155)*.2 = (213,31,31)

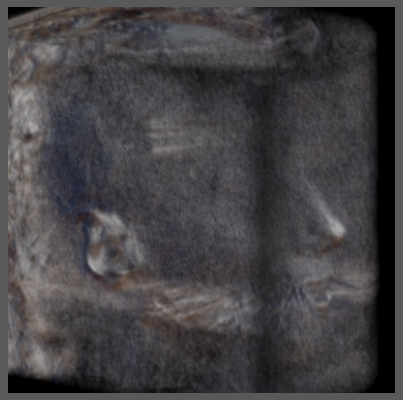
$$C_o = C_i (1-\alpha D) + C\alpha D$$

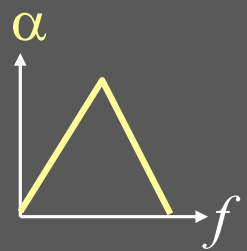
Transfer Function Design

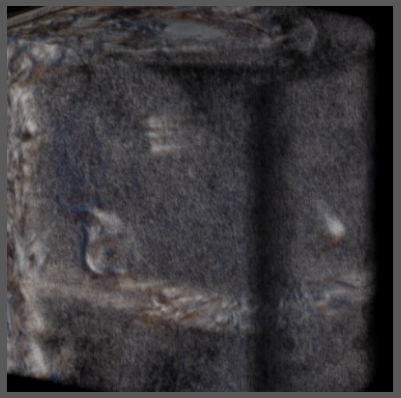
- Transfer function (α and rgb for each scalar value) **design** is non-trivial!
- Lots of tools to help visualization designers to create good transfer functions

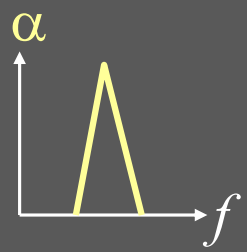
Using different functions for the opacity give different views of data

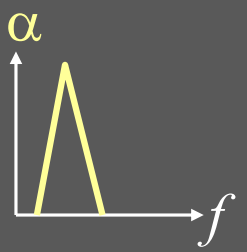


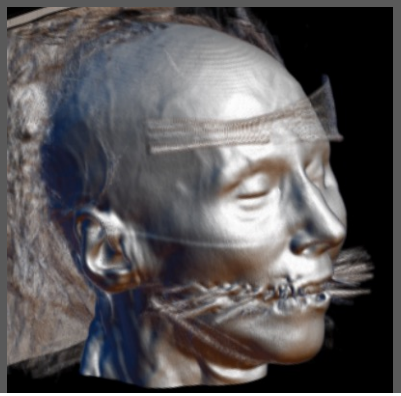






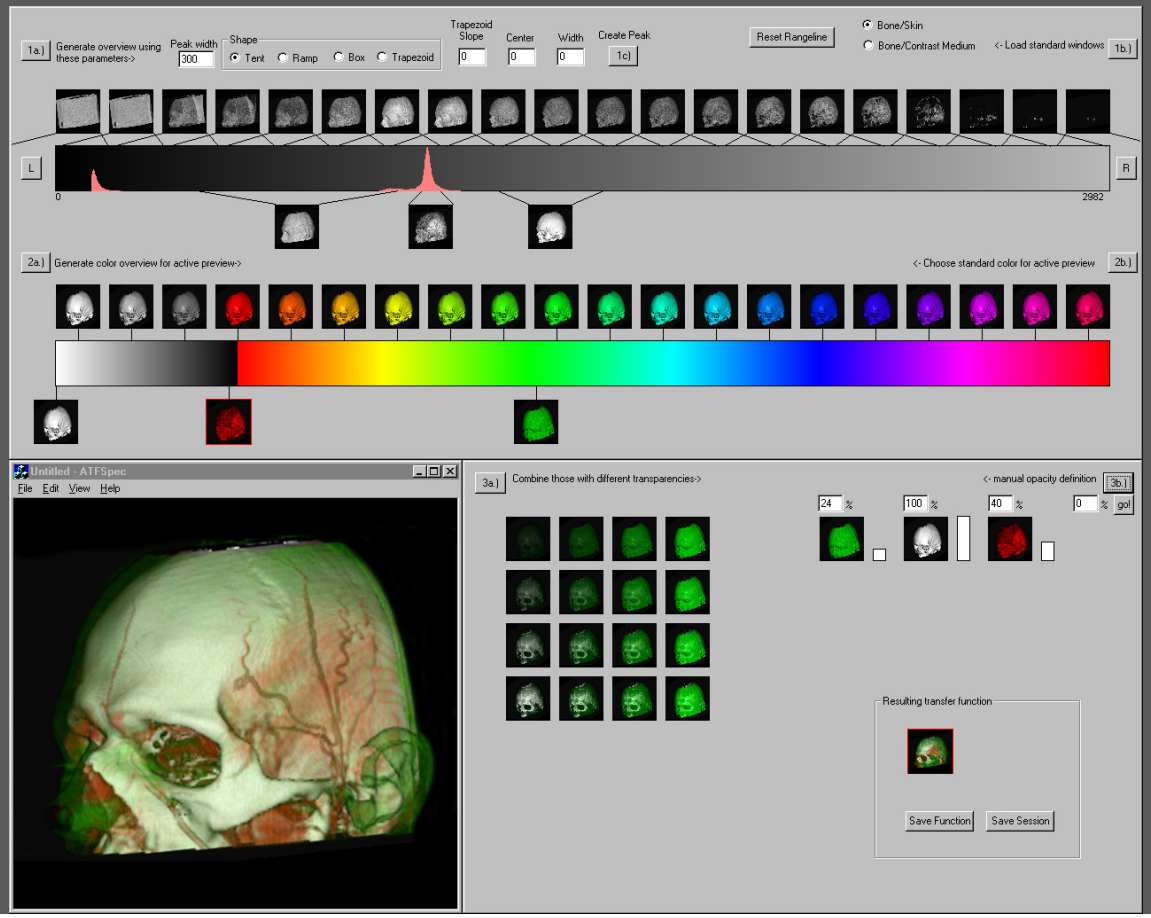






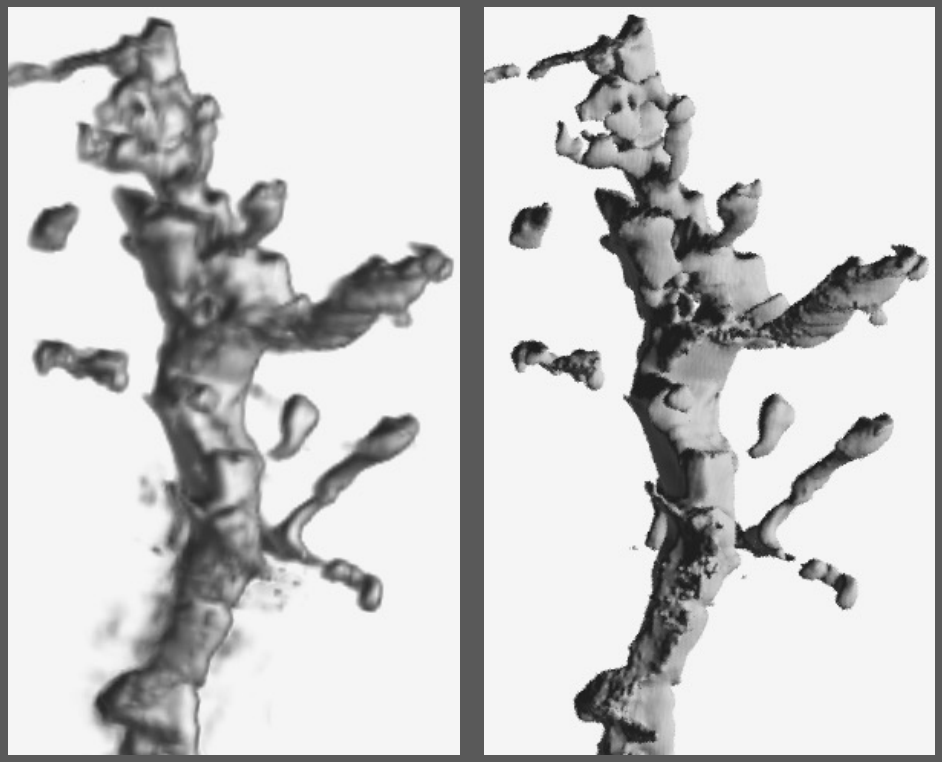
Lots of possible color/opacity

"Mastering Transfer Function Specification Using VolumePro Technology", König, Gröller: Spring Conference on Computer Graphics '01



Multidimensional Transfer Functions

Volume Rendering vs. Isosurfacing

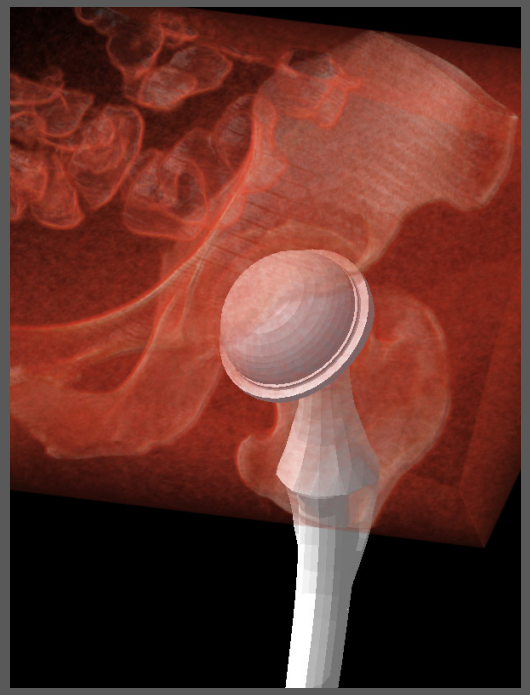


(a) Direct volume rendered

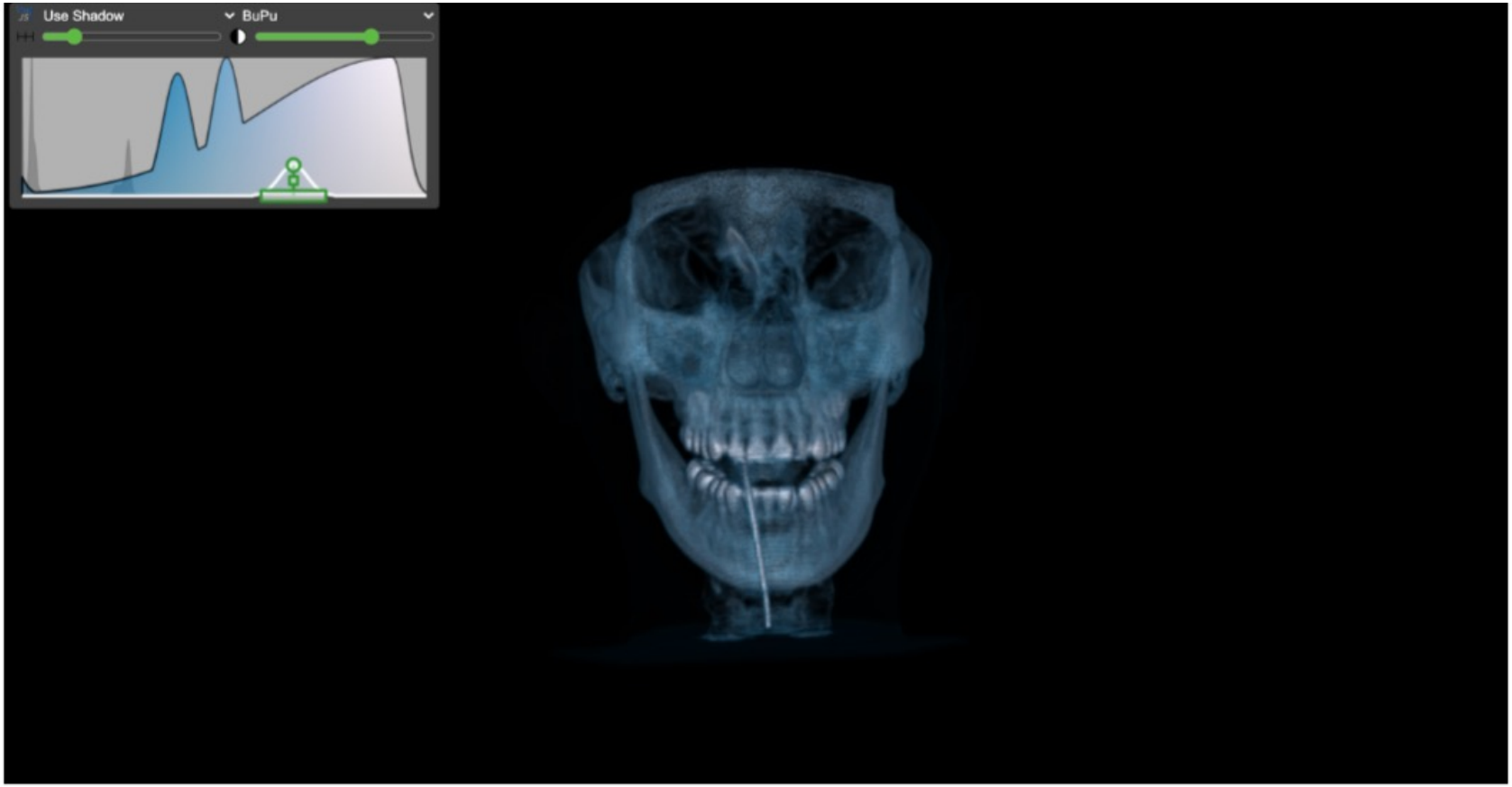
(b) Isosurface rendered

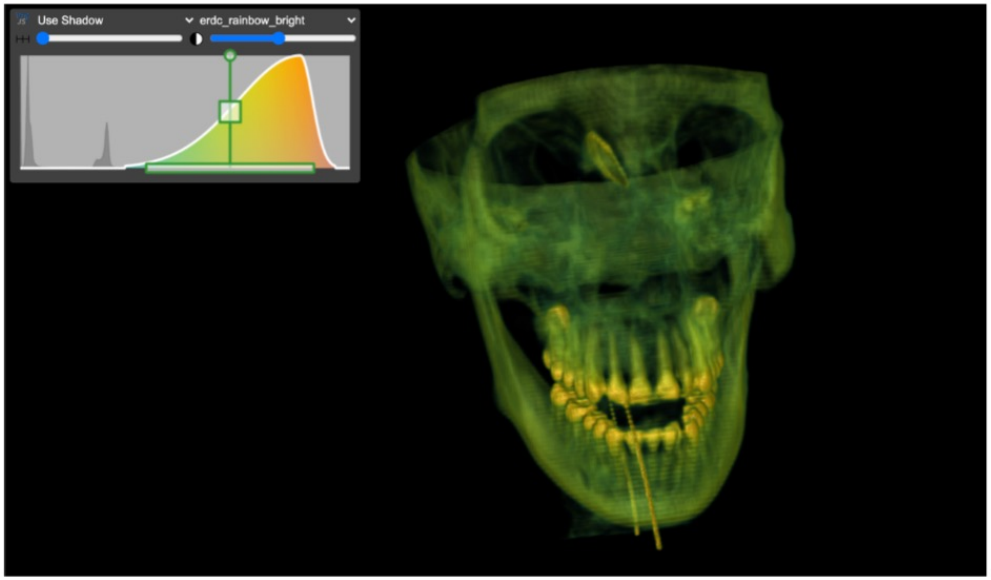
[Kindlmann, 1998]

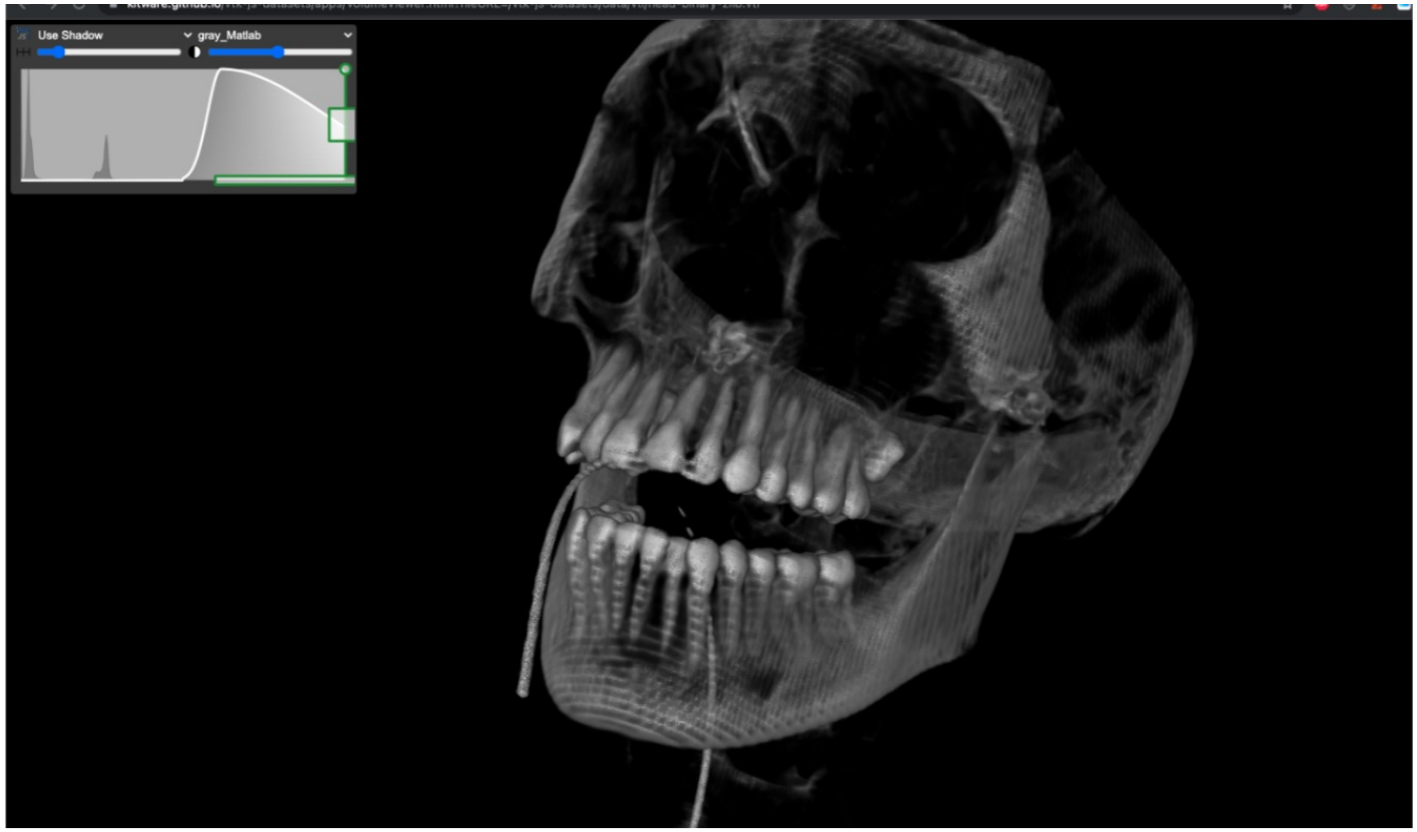
Viewing volume data as a translucent cloud provides context – view of hip replacement



http://www.jch.com/volumes/index.htm

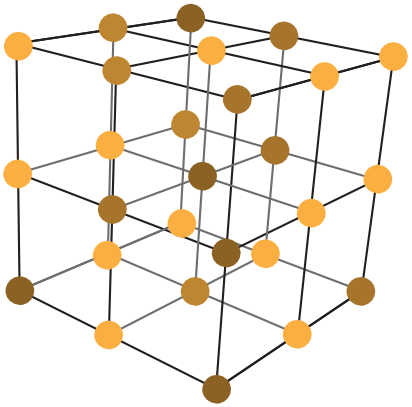

Use Shadow
BuPu


Use Shadow
erdc_rainbow_bright

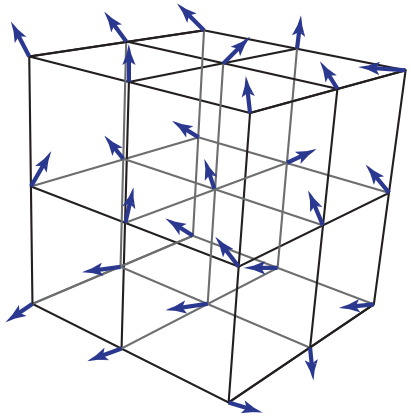

Use Shadow
gray_Matlab

http://randomfootage.homestead.com/pumpkinctscan.html

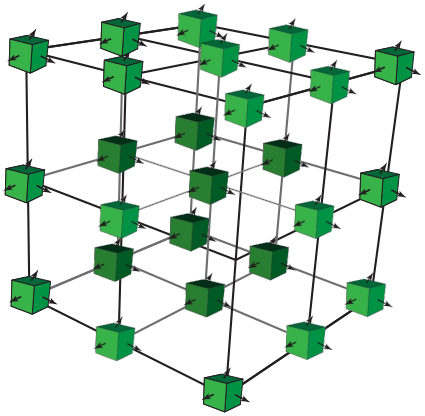
Fields in Visualization



Scalar Fields



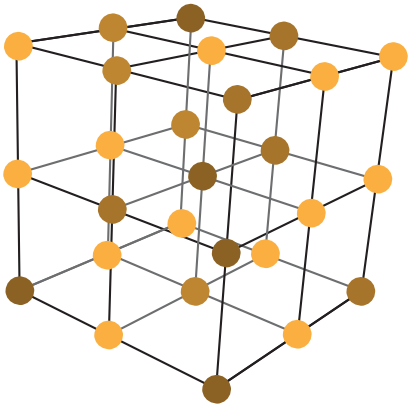
Vector Fields



Tensor Fields

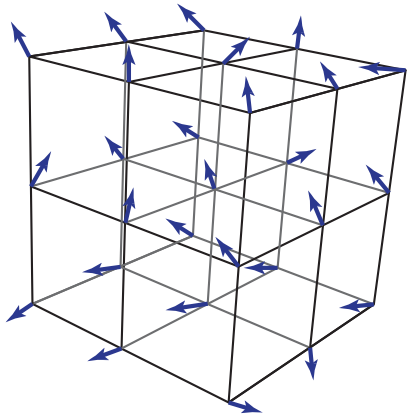
Each point in space has an associated...

Fields in Visualization



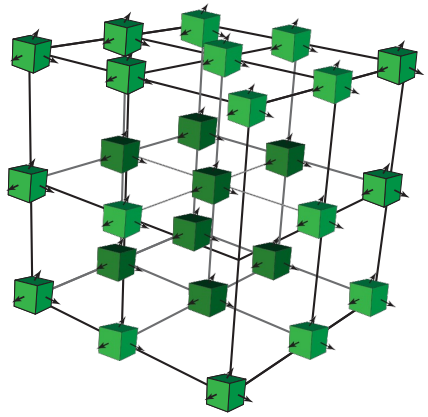
Scalar Fields

(Order-0 Tensor Fields)



Vector Fields

(Order-1 Tensor Fields)



Tensor Fields

(Order-2+)

Each point in space has an associated...

s_0

Scalar

v_0

v_1

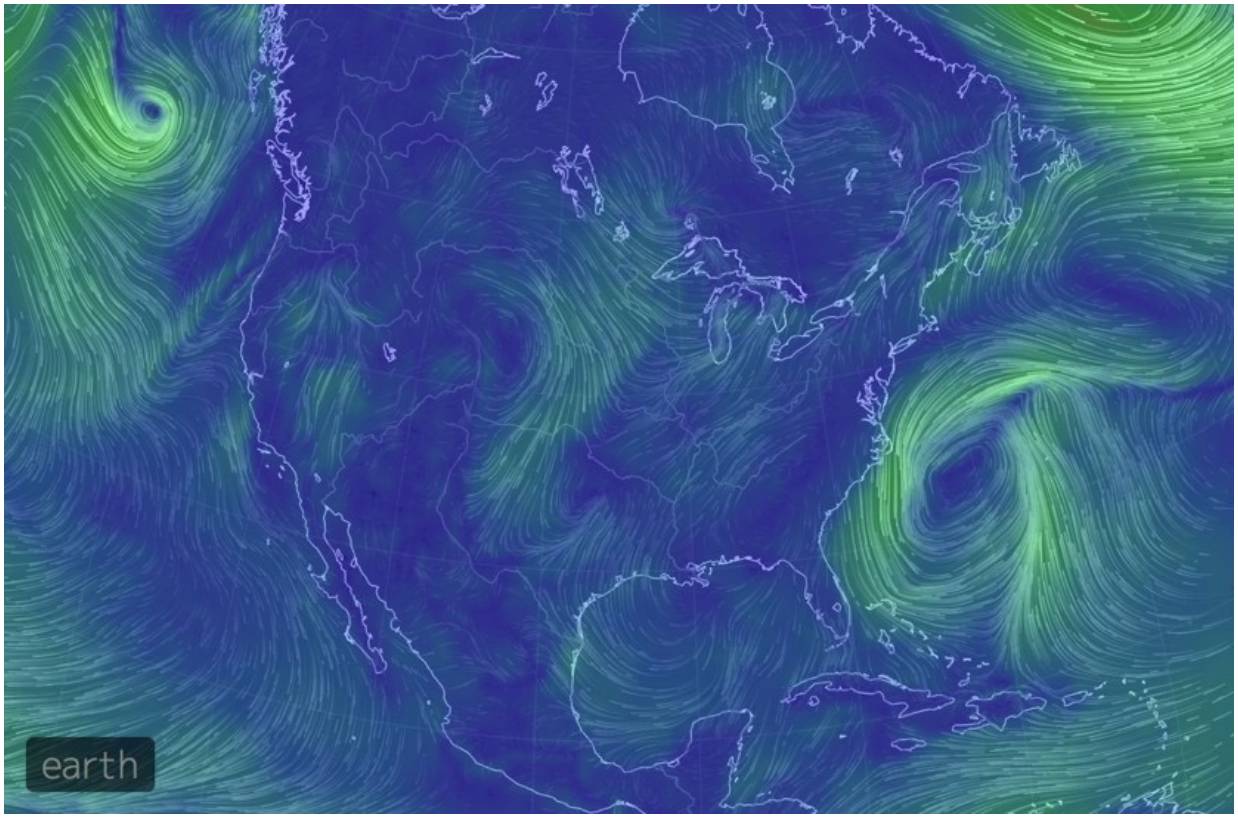
v_2

Vector

$$\begin{matrix} 00 & 01 & 02 \\ 10 & 11 & 12 \\ 20 & 21 & 22 \end{matrix}$$

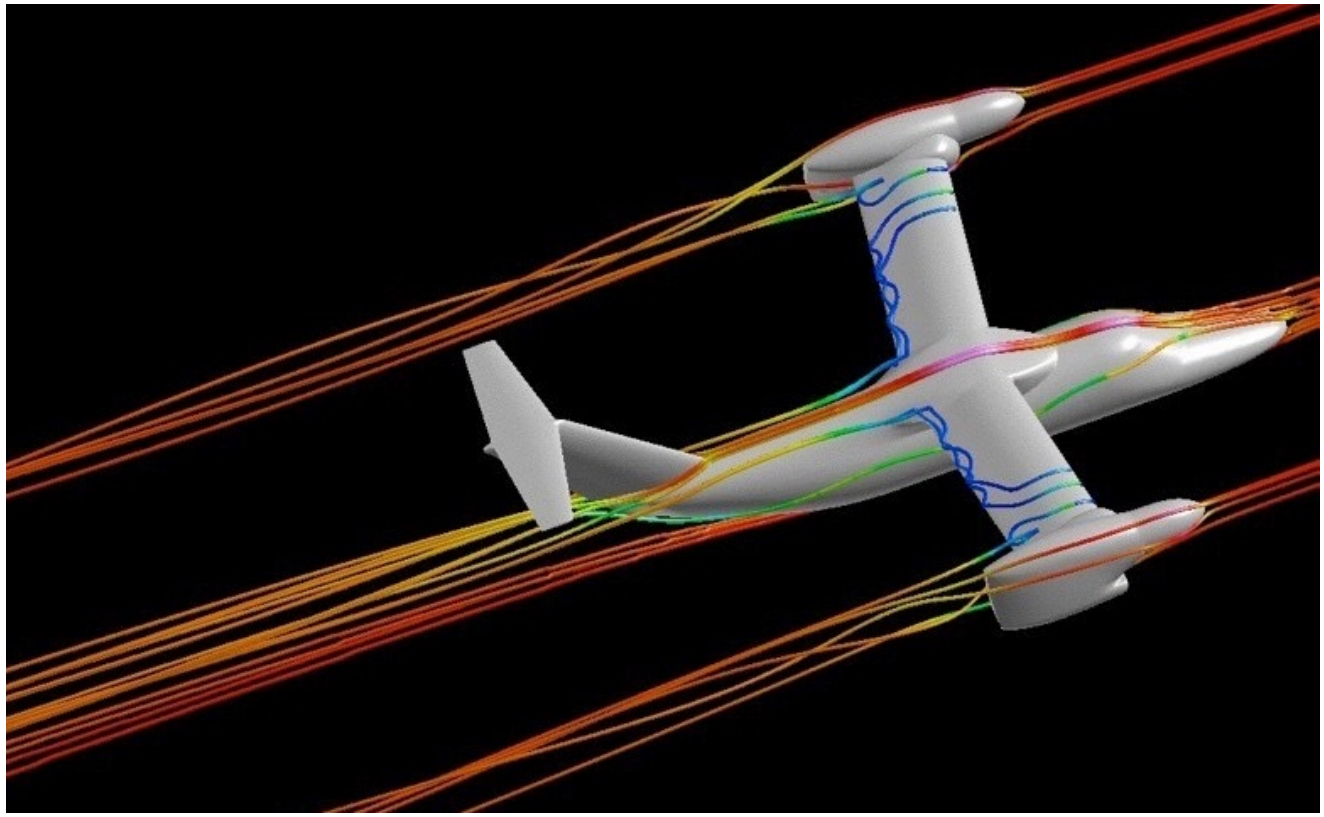
Tensor

Examples of Vector Fields



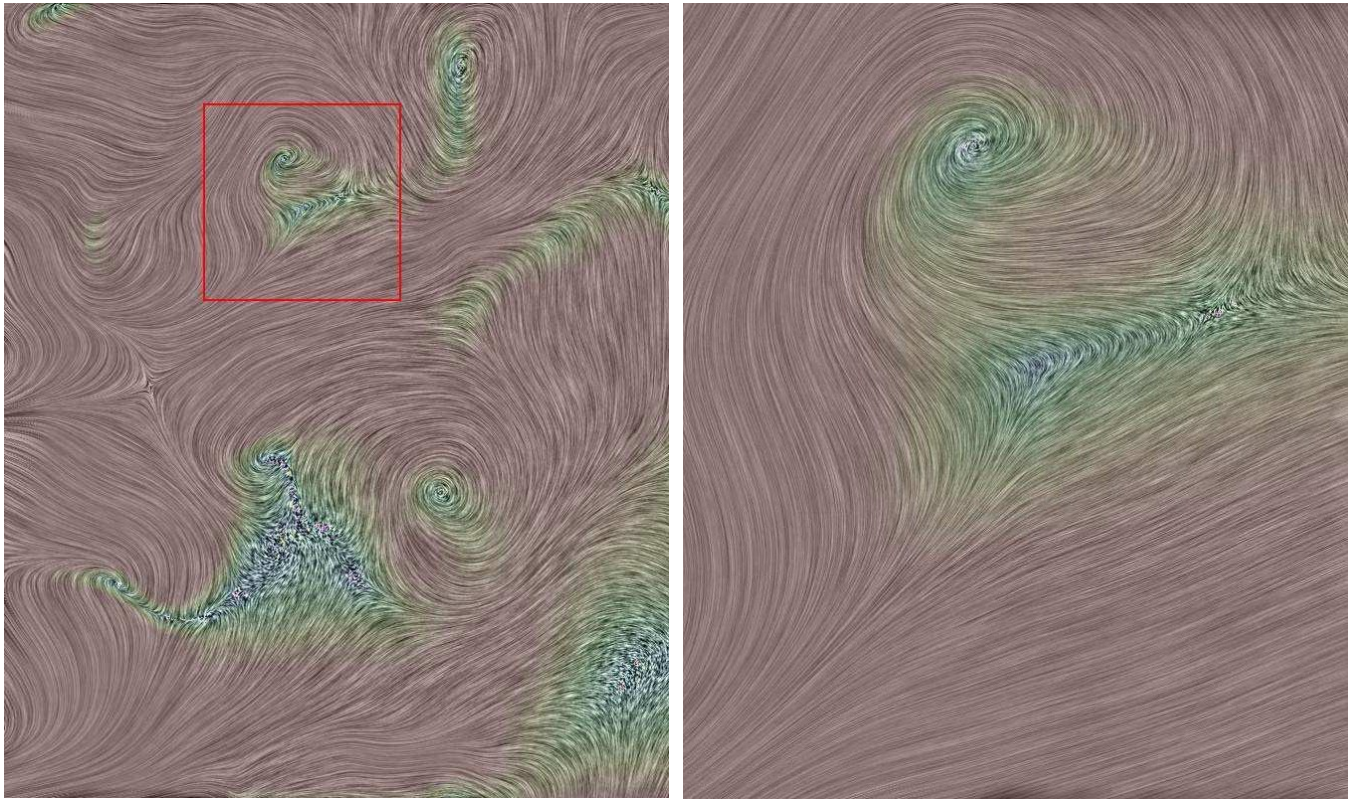
Wind [earth.nullschool.net, 2014]

Examples of Vector Fields



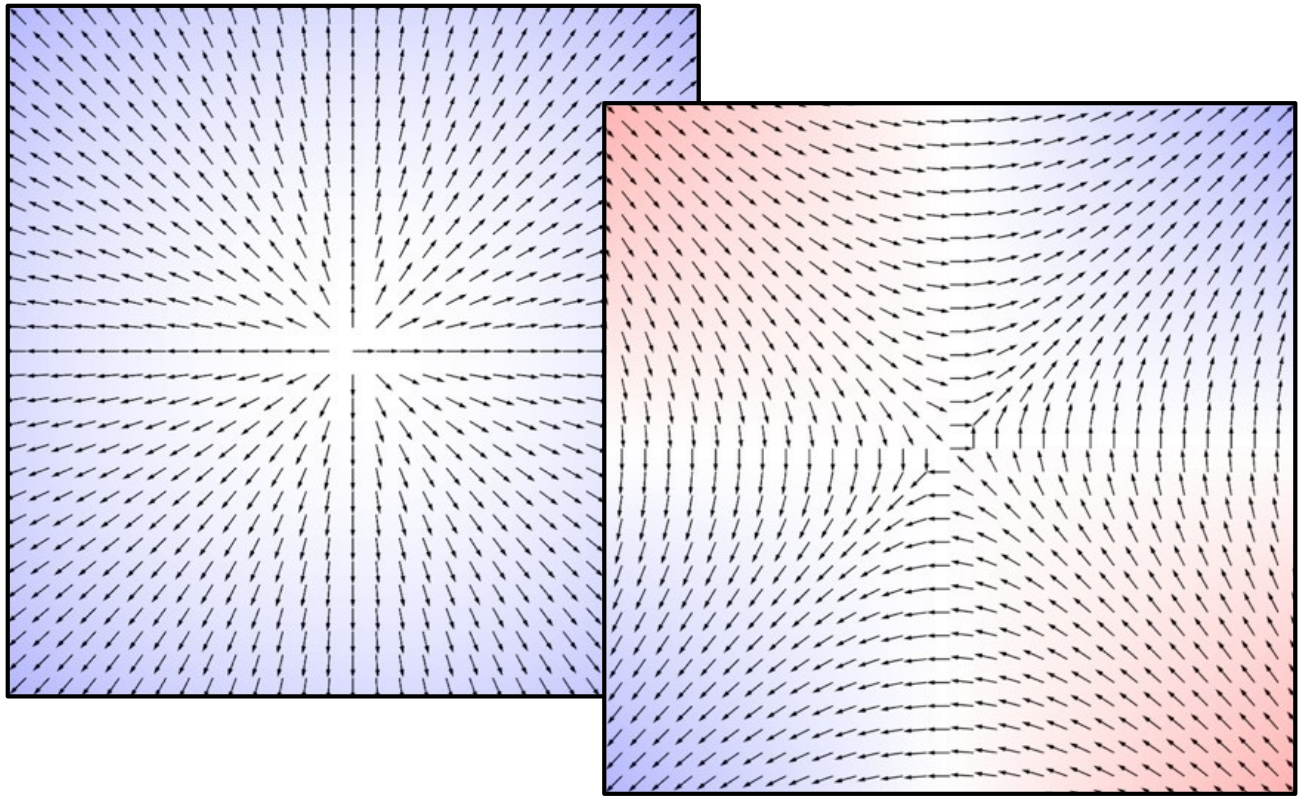
Computational Fluid Dynamics [newmerical]

Examples of Vector Fields



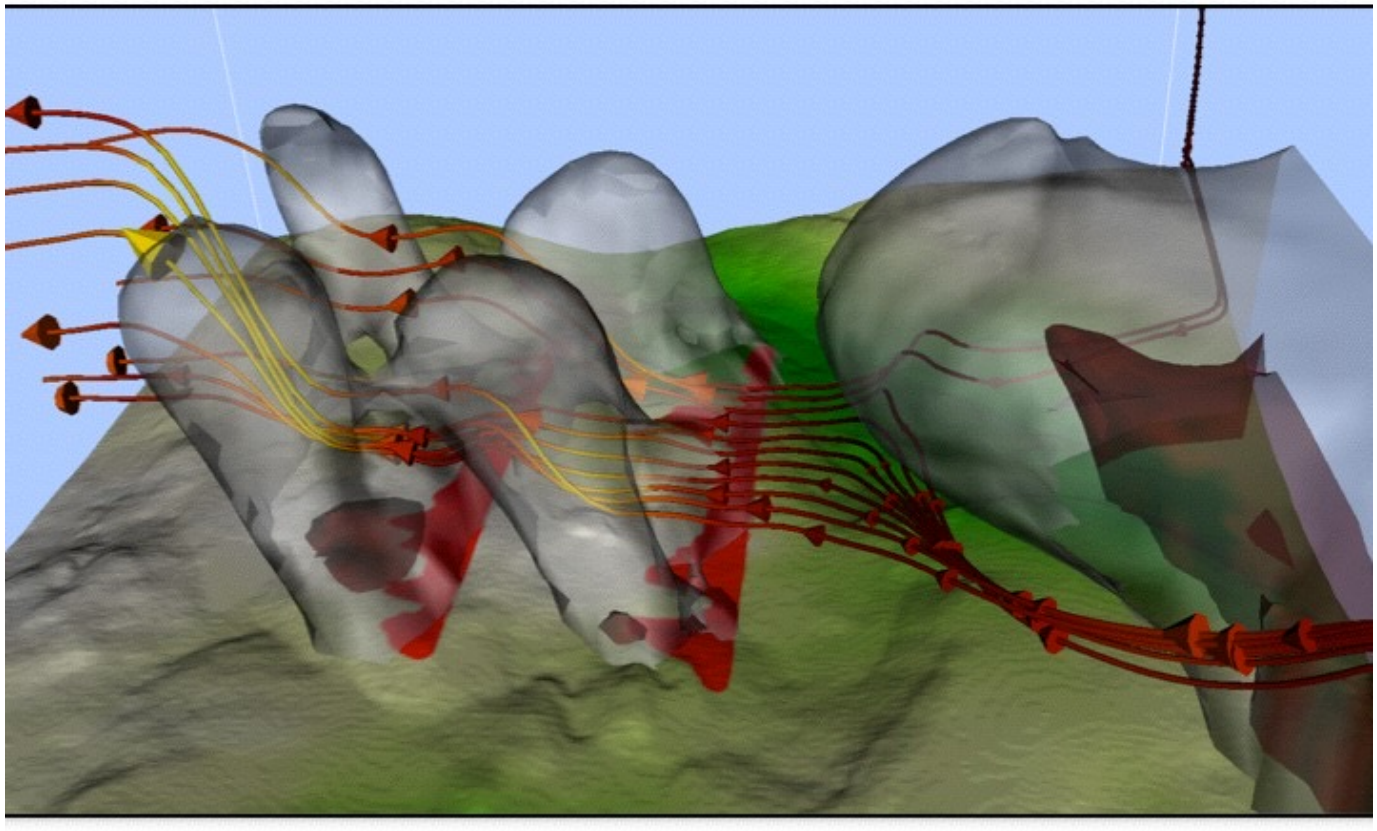
Earthquake Ground Surface Movement [H. Yu et. al., SC2004]

Examples of Vector Fields



Gradient Vector Fields

Examples of Vector Fields



Wildfire Modeling [E. Anderson]

Visualizing Vector Fields

- Direct: Glyphs, Render statistics as scalars
- Geometry: Streamlines and variants
- Textures: Line Integral Convolution (LIC)
- Topology: Extract relevant features and draw them

Glyphs

- Represent each vector with a symbol
- Hedgehogs are primitive glyphs (glyph is a line)

Glyphs

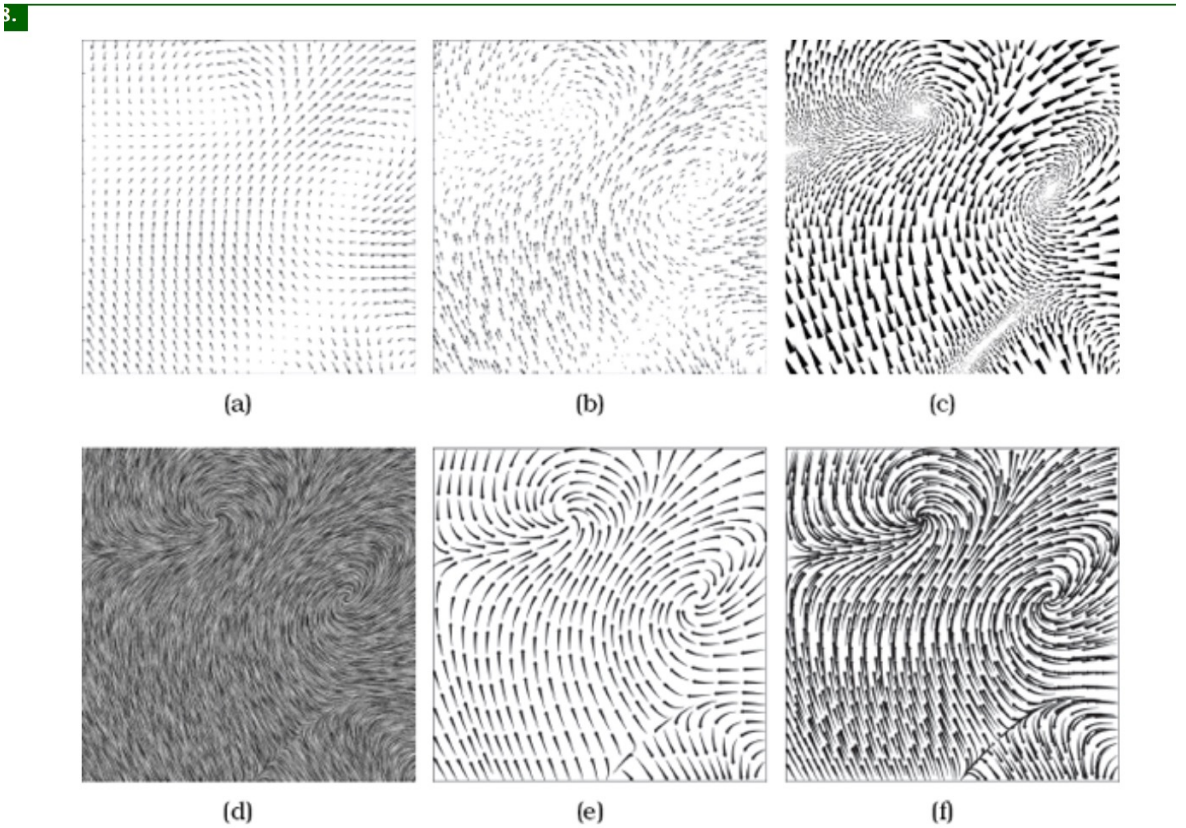
- Represent each vector with a symbol
- Hedgehogs are primitive glyphs (glyph is a line)
- Glyphs that show direction and/or magnitude can convey more information
- If we have a separate scalar value, how might we encode that?
- Clutter issues

Credit: TAKESHI TAKAHARA / SCIENCE PHOTO LIBRARY

Caption: A prototype car being tested in a wind tunnel at Mazda, Japan. The smoke streams indicate the general behaviour of the air flow around the car and are used to assess the coefficient of drag (CD), which expresses the resistance of the car moving through the air. Tiny woolen strings are also attached to the frame. These are used to show the air flow patterns on a smaller scale close to specific points such as the windscreens, air intakes for cooling

Vector Fields -- Flows

Using Glyphs

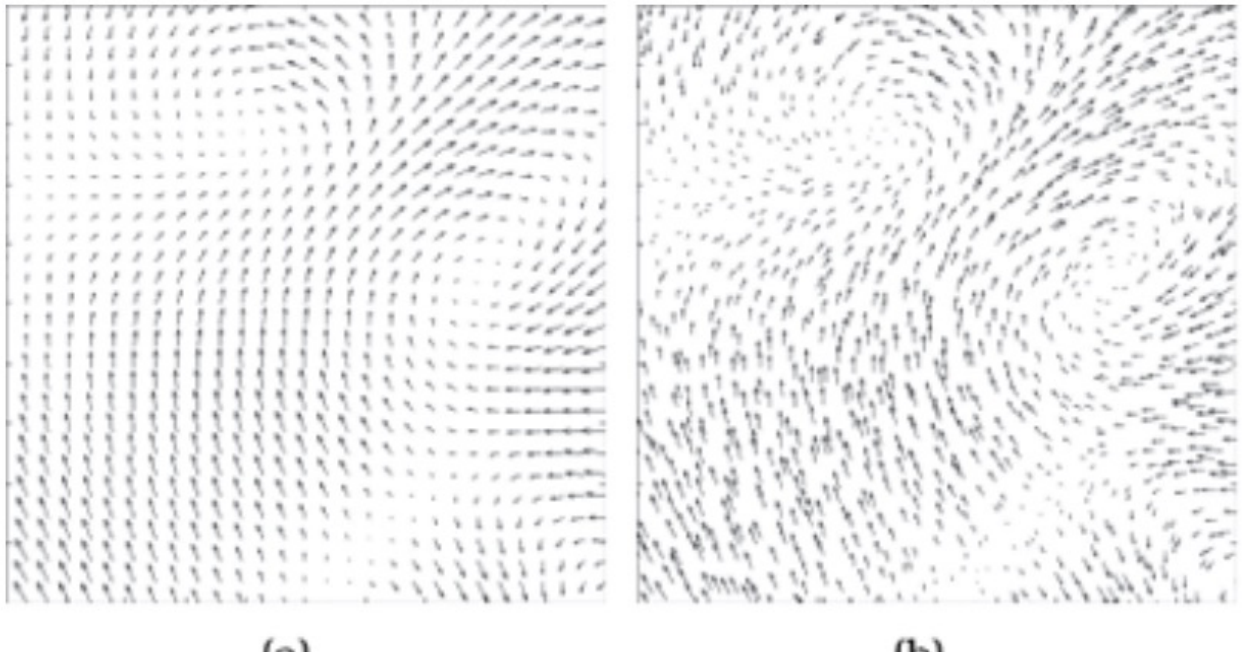


Jitter --Recall

From
https://www.reddit.com/r/dataisbeautiful/comments/qrj0ur/titanic_survival_by_gender_and_class_learning_r/
There are three columns used: class, age, and survival.
Here because there are so many passengers in each passenger class "jitter" is used to spread them out. horizontal position here is determined by class, and then a random number (positive or negative) is added to that position to spread them out. The points can also be made semi-transparent to further show where there are high densities of points.

Jitter

Add small number to x and y before evaluating vector – reduces visual impact of sampling pattern.

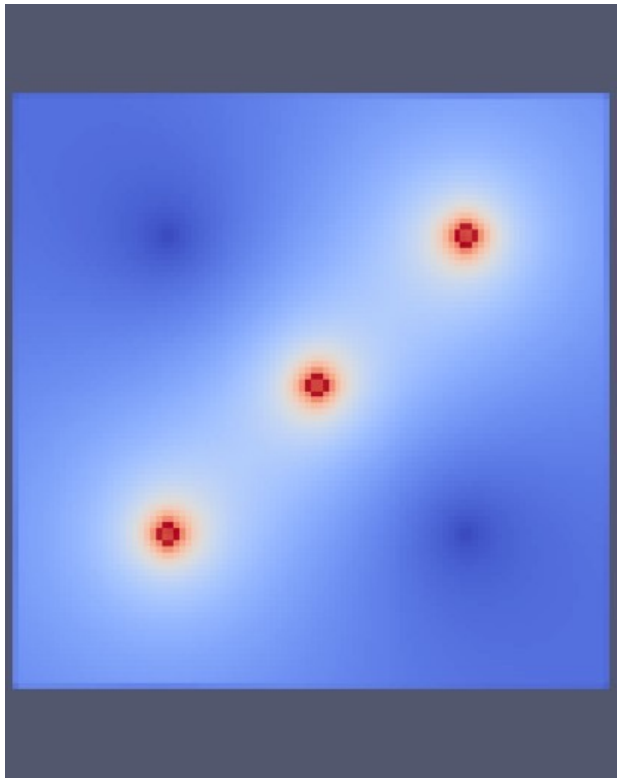


Glyphs

- For vector fields, can encode
  - Direction
  - Magnitude
  - Scalar value
- Good:
  - Show precise local measures
  - Can encode scalar information as color
- Bad:
  - Possible sampling issues
  - Clutter (Occlusion): Can remove some points to help
  - Clutter is worse in higher dimensions

Rendering Vector Field Statistics as Scalars

- Many statistics we can compute for vector fields:
  - Magnitude
  - Vorticity
  - Curvature
- These are scalars, can color with our scalar field visualization techniques (e.g. volume rendering)

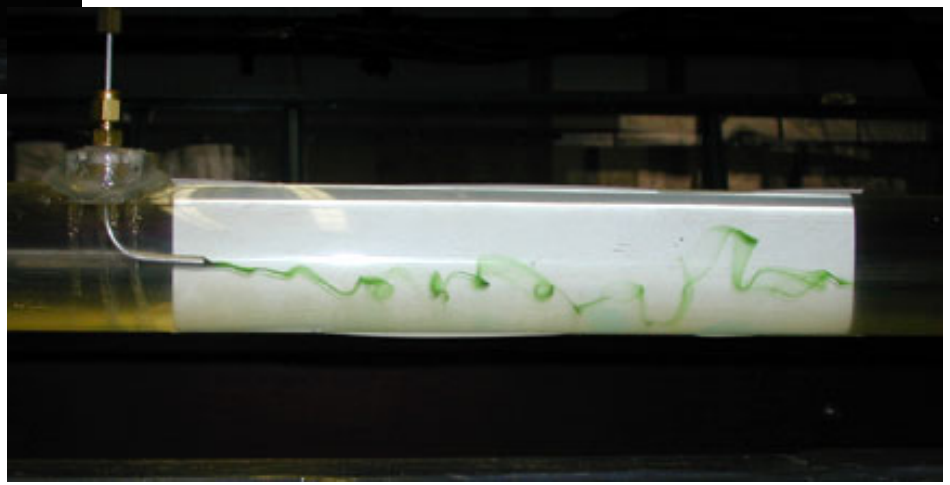


[Color indicates vector magnitude]

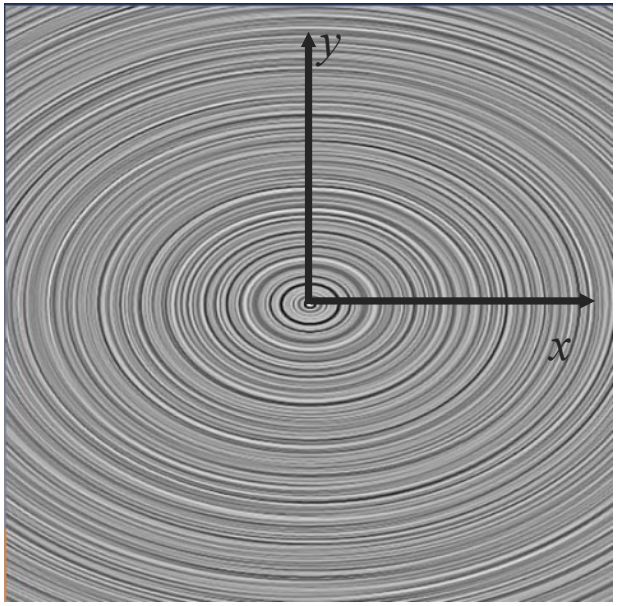
Streamlines & Variants

- Trace a line along the direction of the vectors
- Streamlines are always tangent to the vector field
- Basic Particle Tracing:
  1. Set a starting point (seed)
  2. Take a step in the direction of the vector at that point
  3. Adjust direction based on the vector where you are now
  4. Go to Step 2 and Repeat

Visualization before/without computers



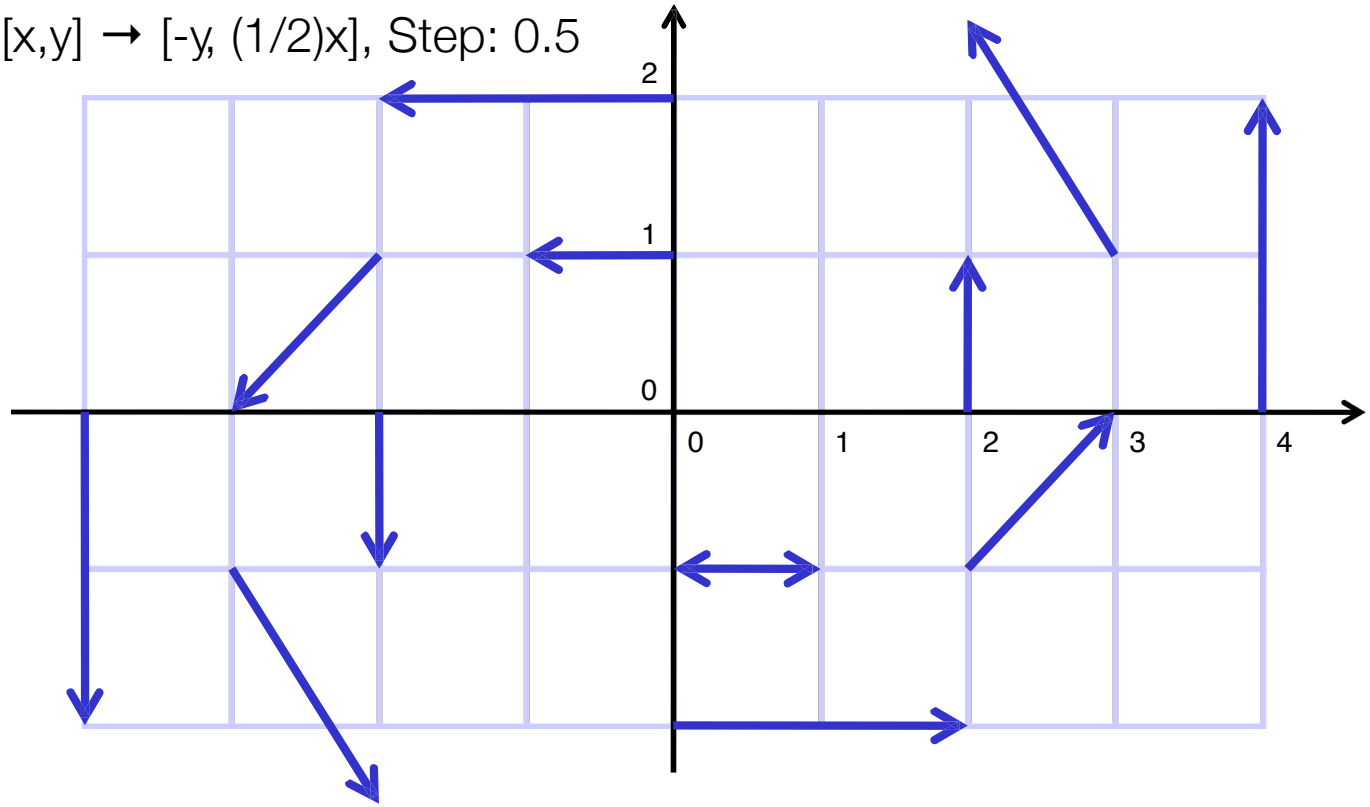
- Elliptical path
- Suppose we have the actual equation
- Given point (x,y), the vector is at that point is [v_x, v_y] where
  - $v_x = -y$
  - $v_y = (1/2)x$
- Want a streamline starting at (0,-1)



[LIC (**not streamlines!**) via Levine]

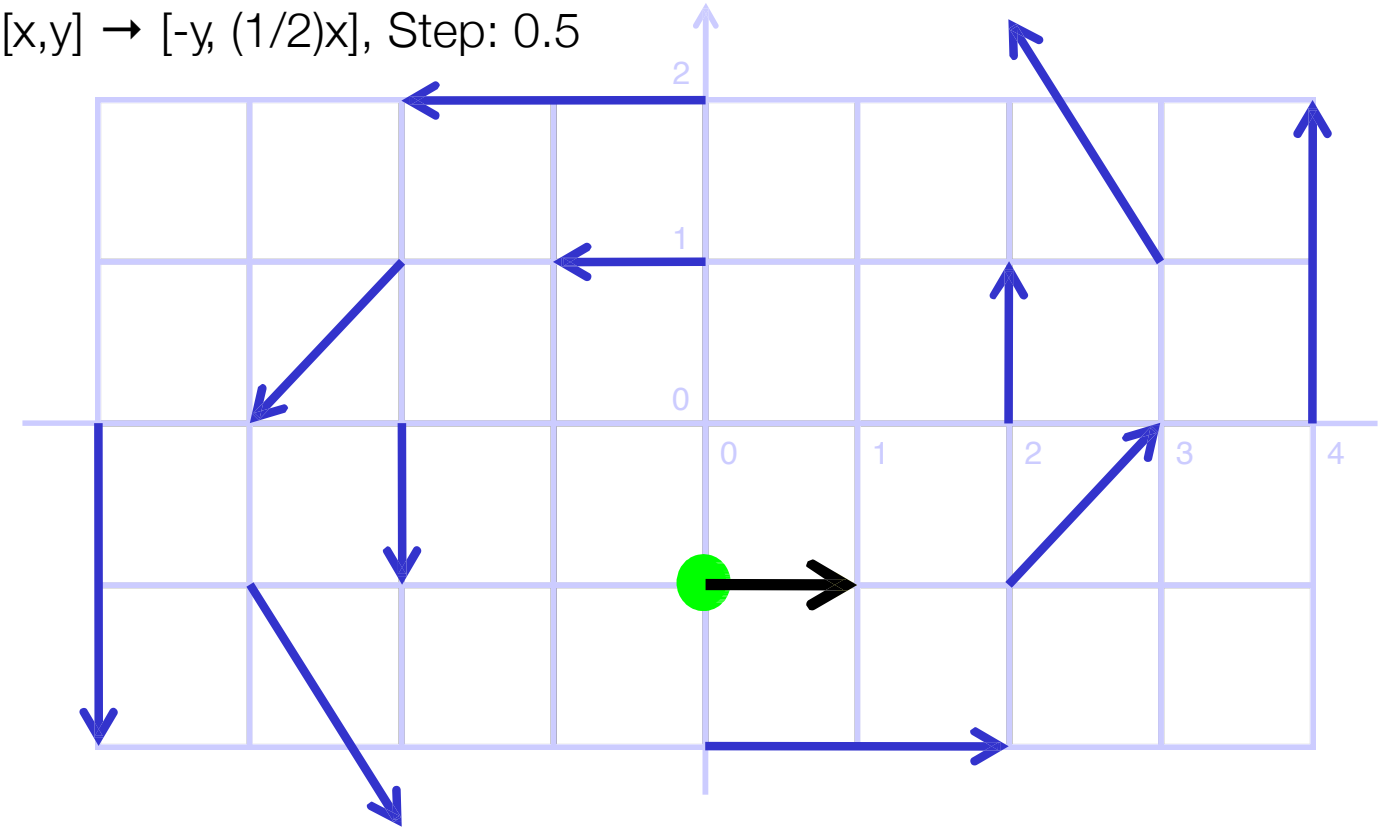
Some Glyphs

[x,y] → [-y, (1/2)x], Step: 0.5



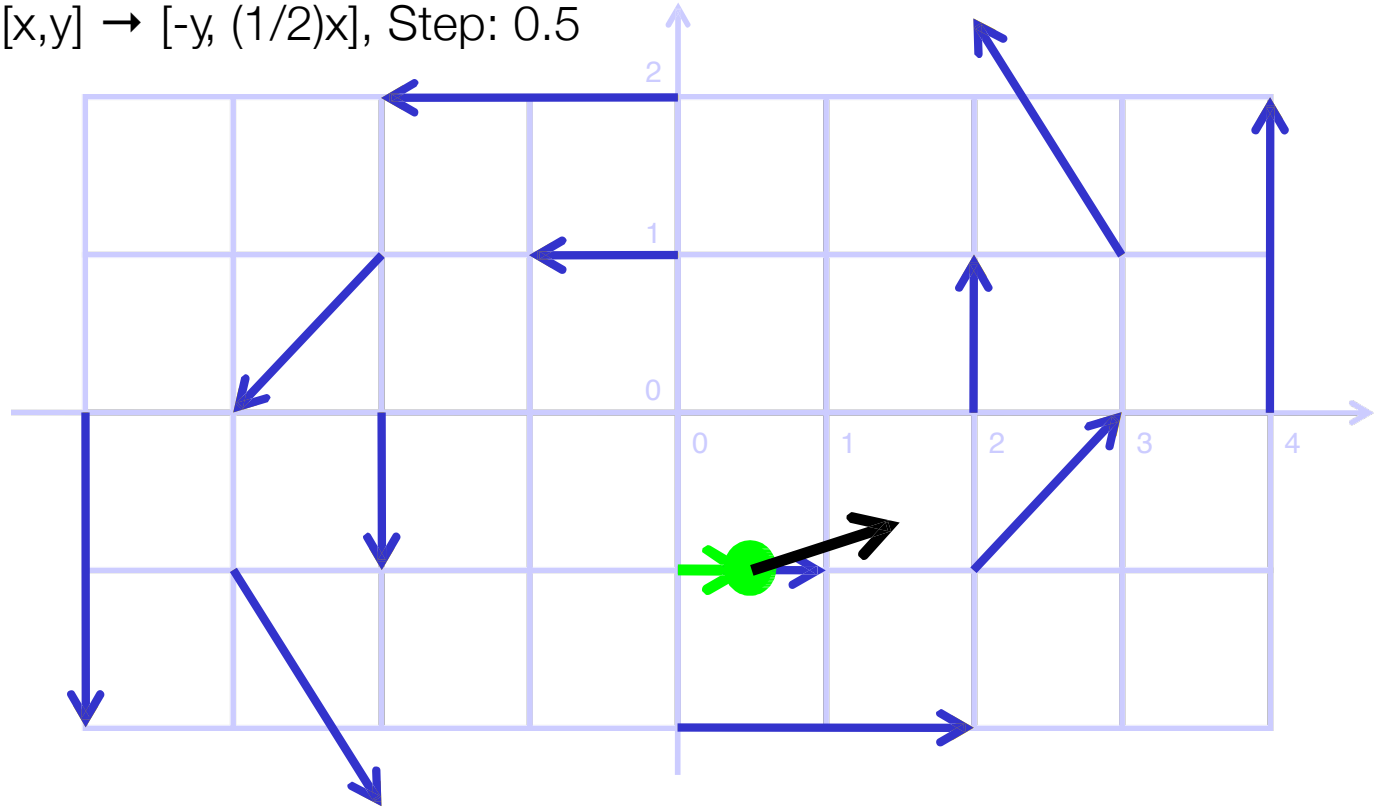
Streamlines (Step 1)

[x,y] → [-y, (1/2)x], Step: 0.5



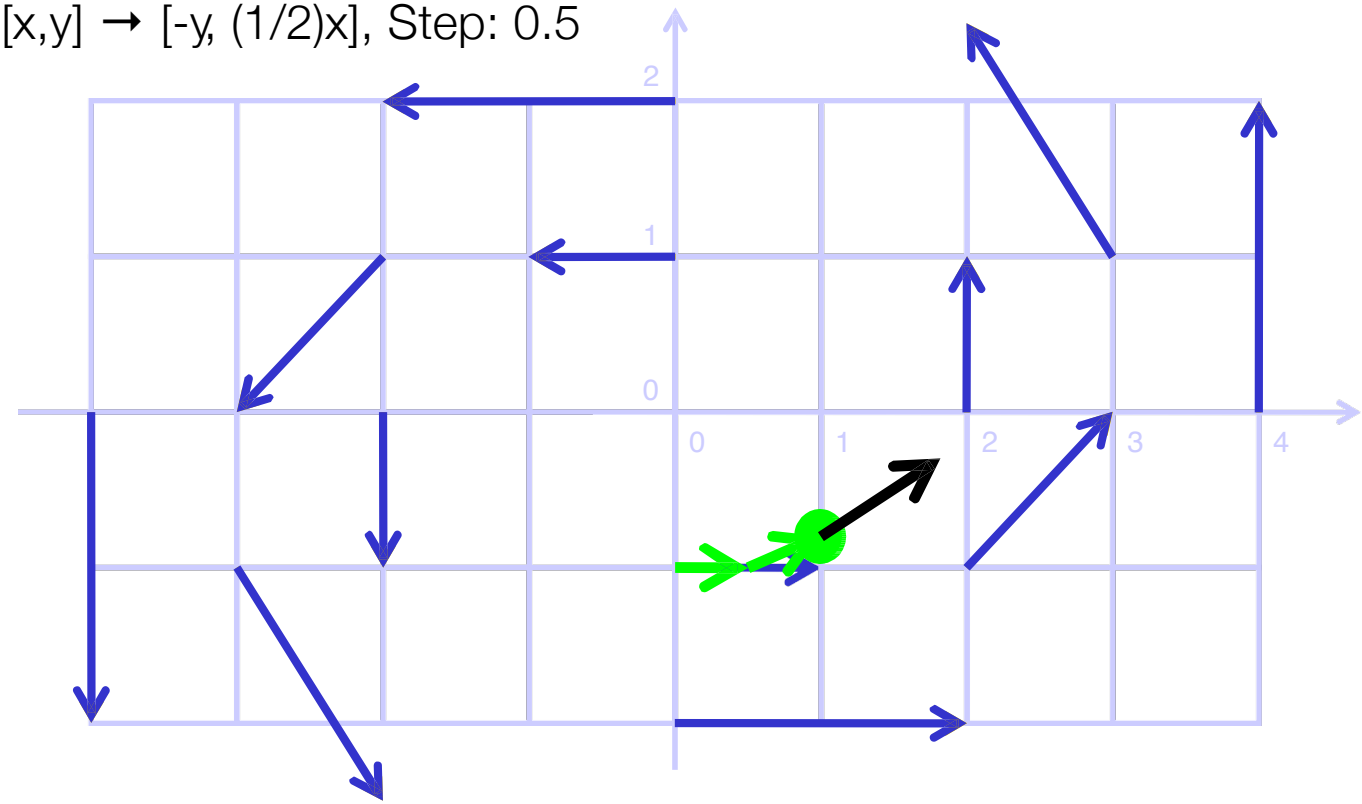
Streamlines (Step 2)

[x,y] → [-y, (1/2)x], Step: 0.5



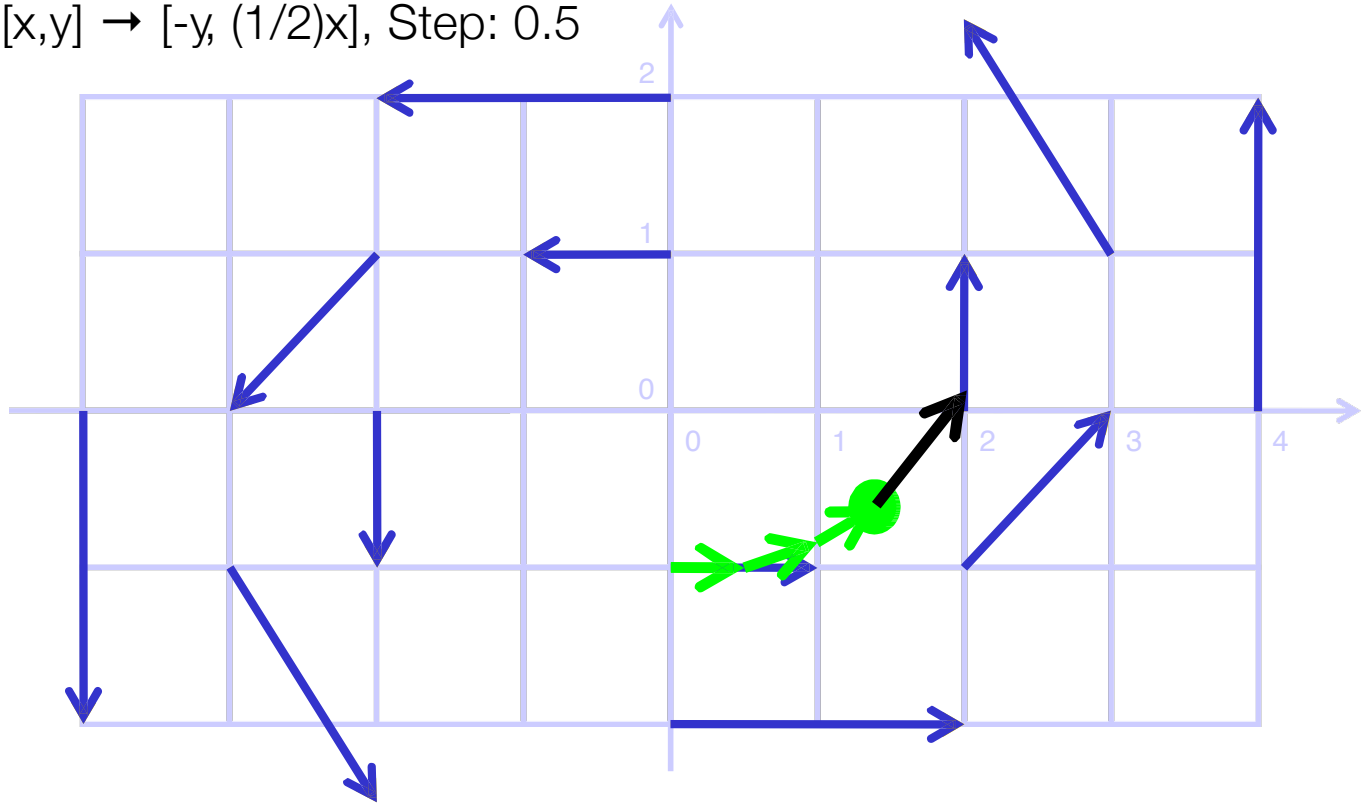
Streamlines (Step 3)

[x,y] → [-y, (1/2)x], Step: 0.5



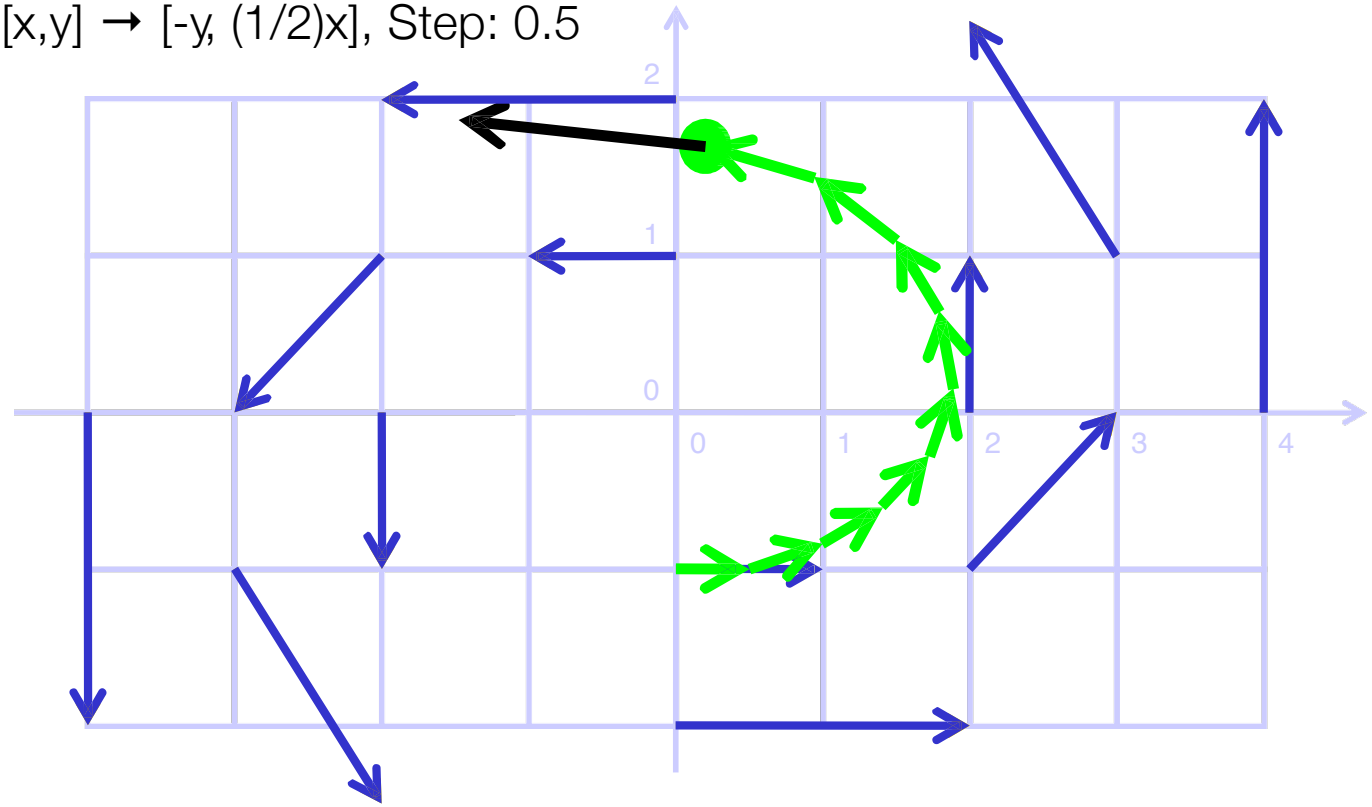
Streamlines (Step 4)

[x,y] → [-y, (1/2)x], Step: 0.5



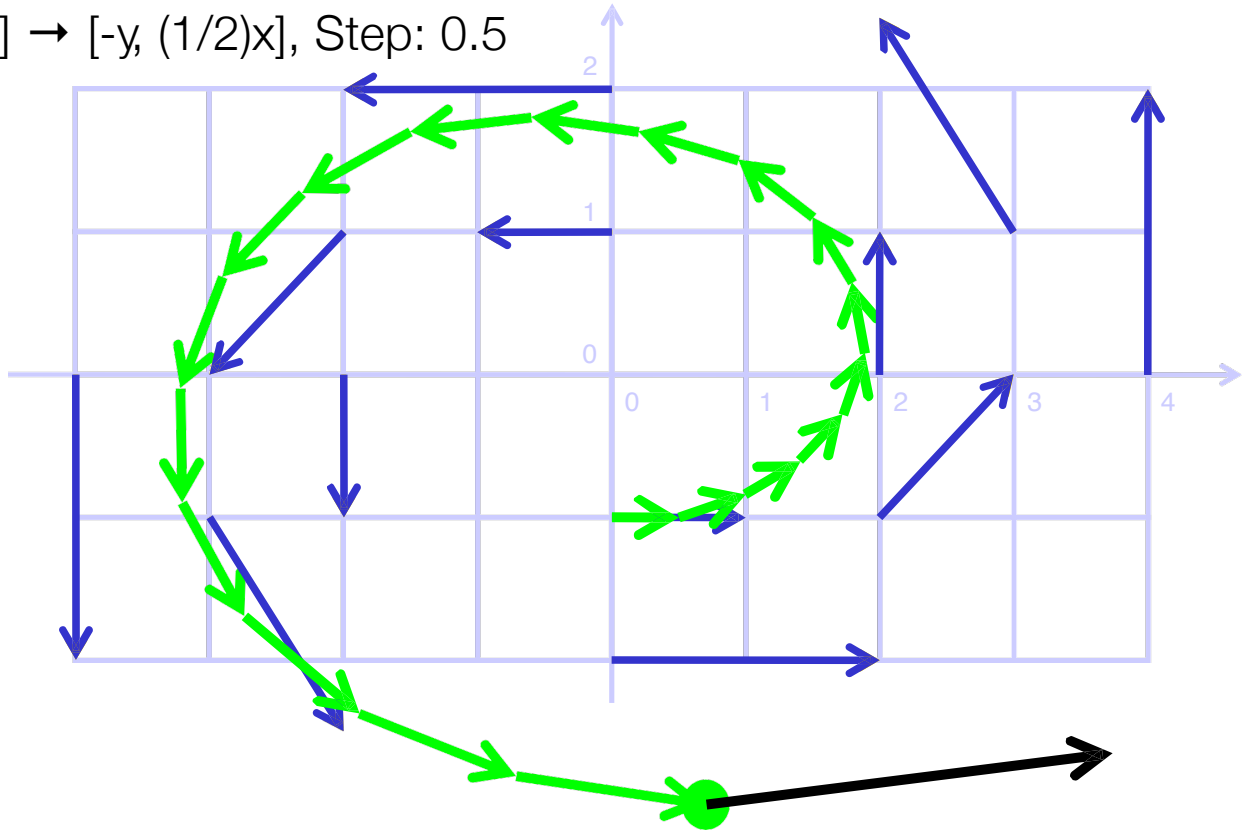
Streamlines (Step 10)

[x,y] → [-y, (1/2)x], Step: 0.5



Streamlines (Step 19)

[x,y] → [-y, (1/2)x], Step: 0.5



Euler Method

- Seeking to approximate integration of the velocity over time
- Euler method is the starting point for approximating this
- Problems?

Euler Method

- Seeking to approximate integration of the velocity over time
- Euler method is the starting point for approximating this
- Problems?
  - Choice of step size is important

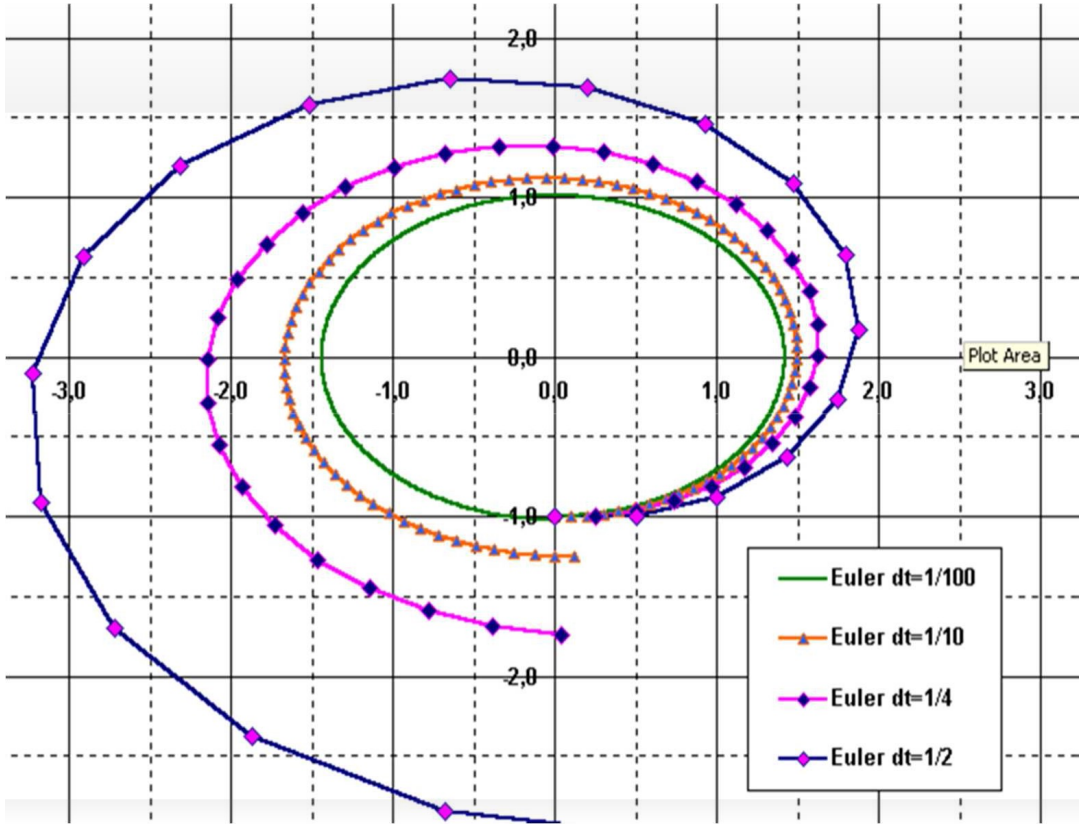
Euler Method

- Seeking to approximate integration of the velocity over time
- Euler method is the starting point for approximating this
- Problems?
  - Choice of step size is important
  - Choice of seed points are important

Euler Method

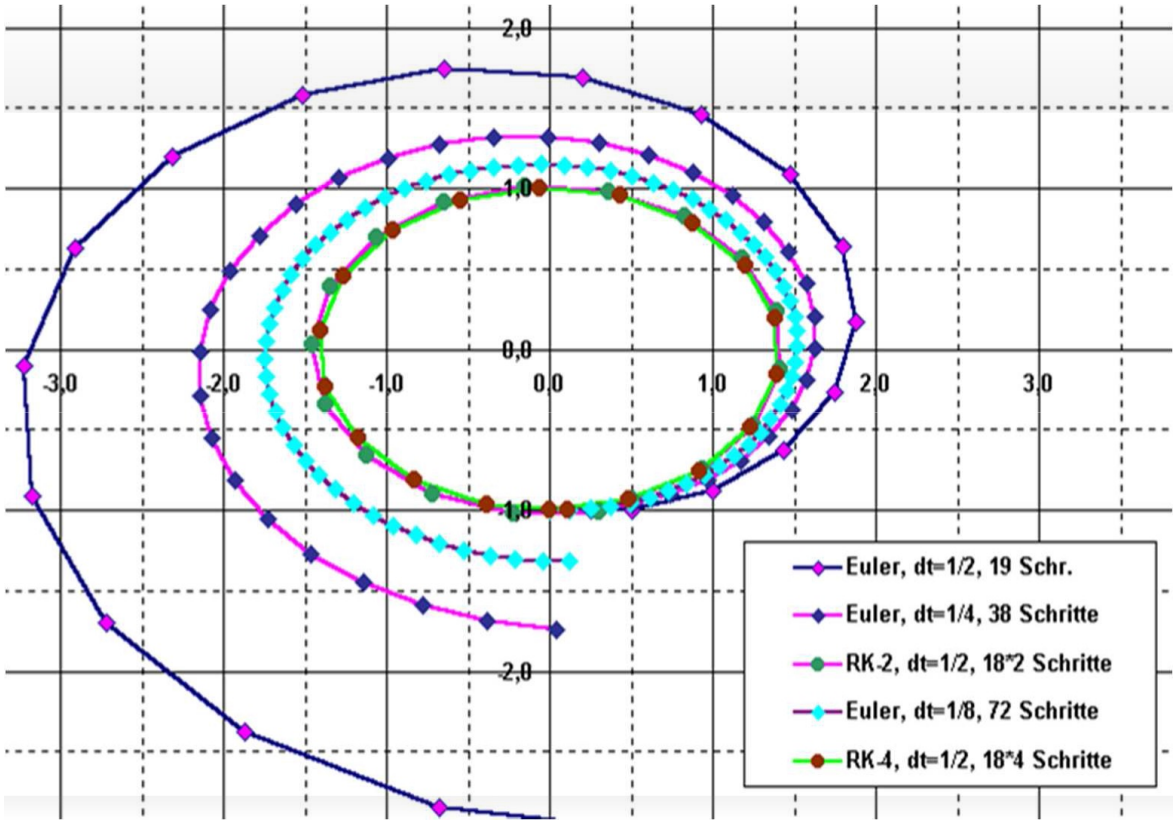
- Seeking to approximate integration of the velocity over time
- Euler method is the starting point for approximating this
- Problems?
  - Choice of step size is important
  - Choice of seed points are important
- Also remember that we have a field—we don't have measurements at every point (interpolation)

Euler Quality by Step Size



[via Levine]

Higher-Order Comparison

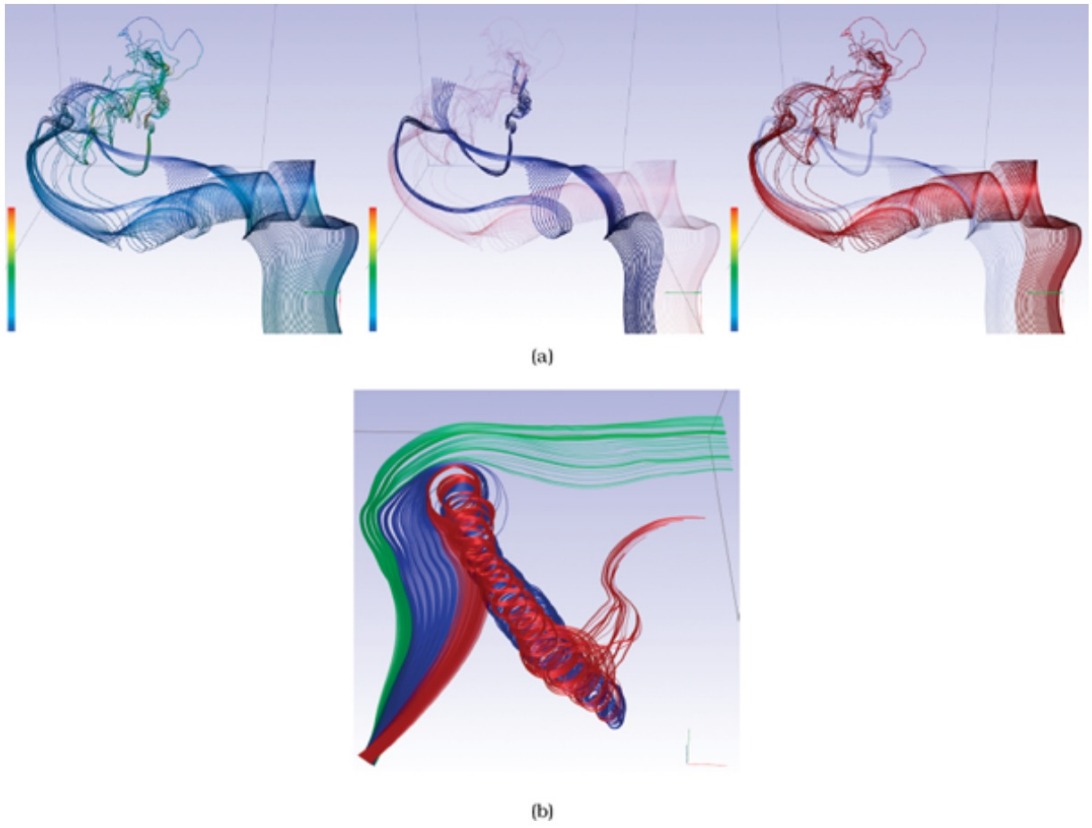


[UMass Dartmouth/ Levine]

Vector Fields

Streamlines, pathlines show what happen to passive particles in the flow – derived geometry.

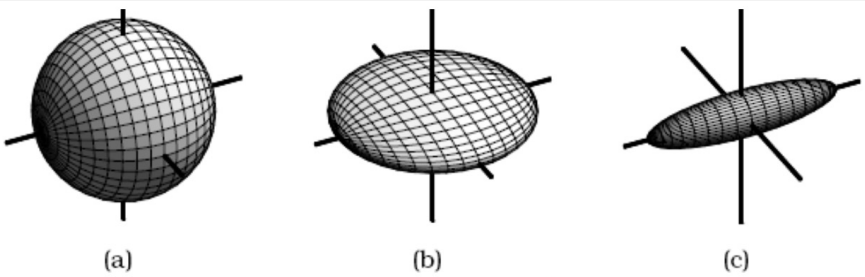
Figure 8.9.



(a)

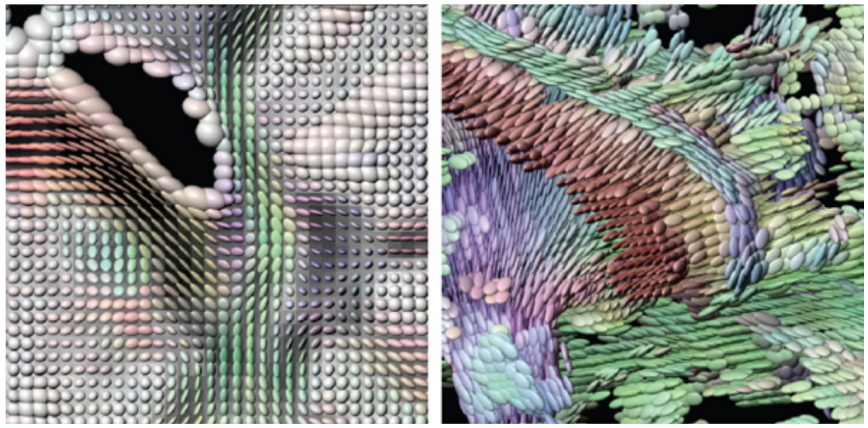
(b)

Tensor Fields – A Matrix at Each Point



Ellipsoid glyphs can show three basic shapes. (a) Isotropic: sphere. (b) Partially anisotropic: planar. (c) Fully anisotropic: linear. From [Kindlmann 04, Figure 1].

Figure 8.12.

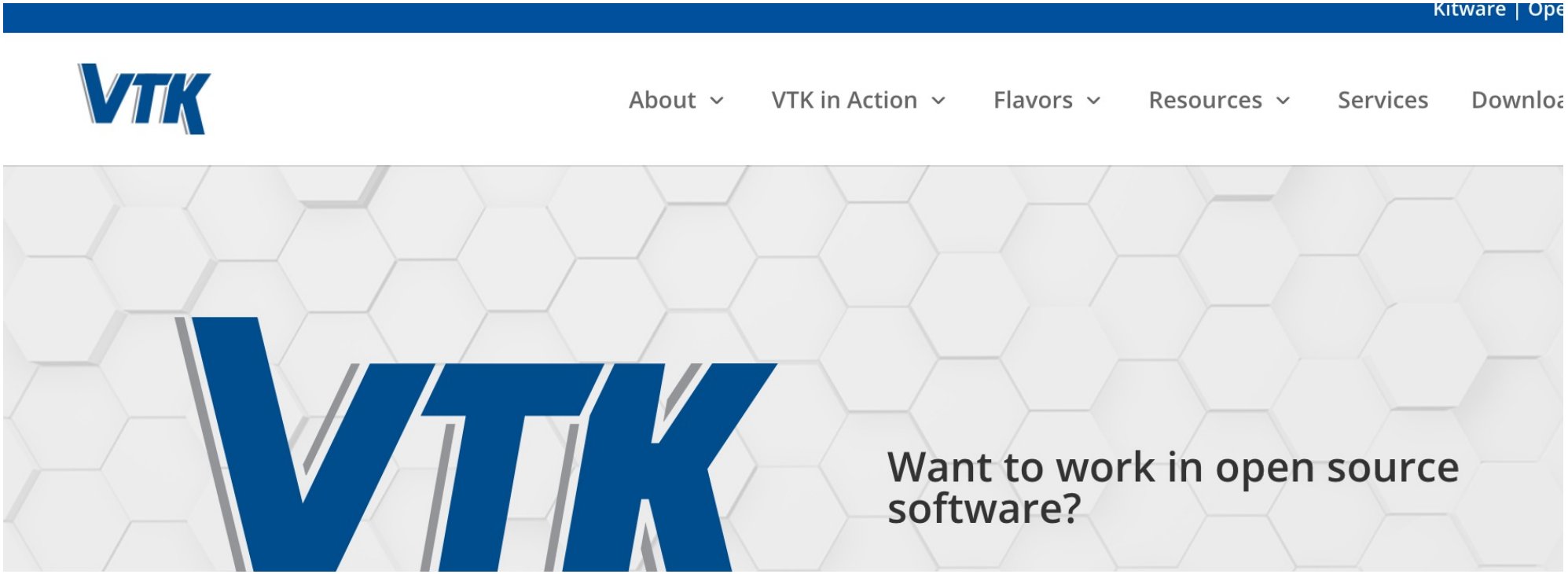


Programming – Visualizing Grid and Field Data

With the exception of glyph (like the arrows in HW4) for regularly sampled value in the plane visualization doesn't fit the D3 model

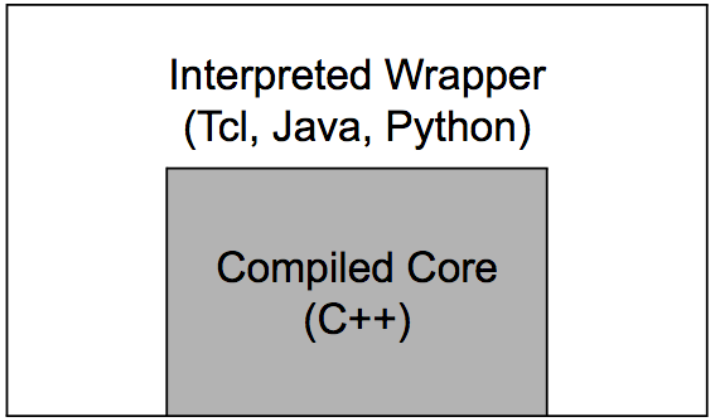
-- Need interpolation between values
-- Need to extract geometric structures
-- Need to specify view and 3D ->2D projection

Many applications built with VTK: https://vtk.org/

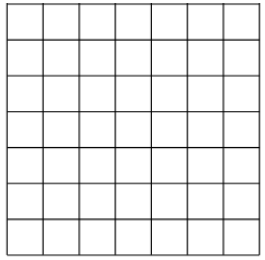


VTK was originally part of the textbook The Visualization Toolkit An Object-Oriented Approach to 3D Graphics. Will Schroeder, Ken Martin, and Bill Lorensen—three graphics and visualization researchers—wrote the book and companion software on their own time, beginning in December 1993, with legal permission from their then-employer, GE R&D. The motivation for the book was to collaborate with other researchers and develop an open framework for creating leading-edge visualization and graphics applications.

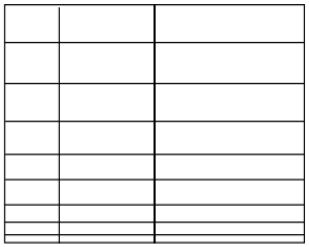
The Visualization Toolkit consists of two basic subsys-tems: a compiled C++ class library and an “inter-preted” wrapper layer



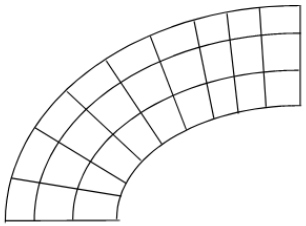
Dataset types



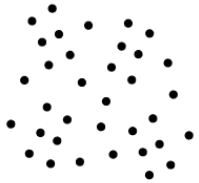
(a) Image Data
(vtkImageData)



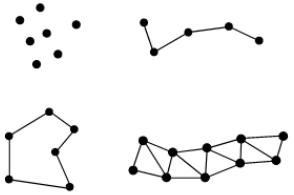
(b) Rectilinear Grid
(vtkRectilinearGrid)



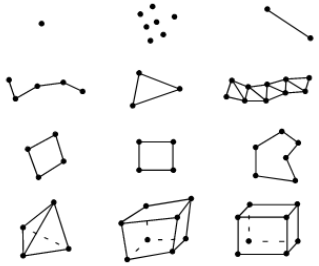
(c) Structured Grid
(vtkStructuredGrid)



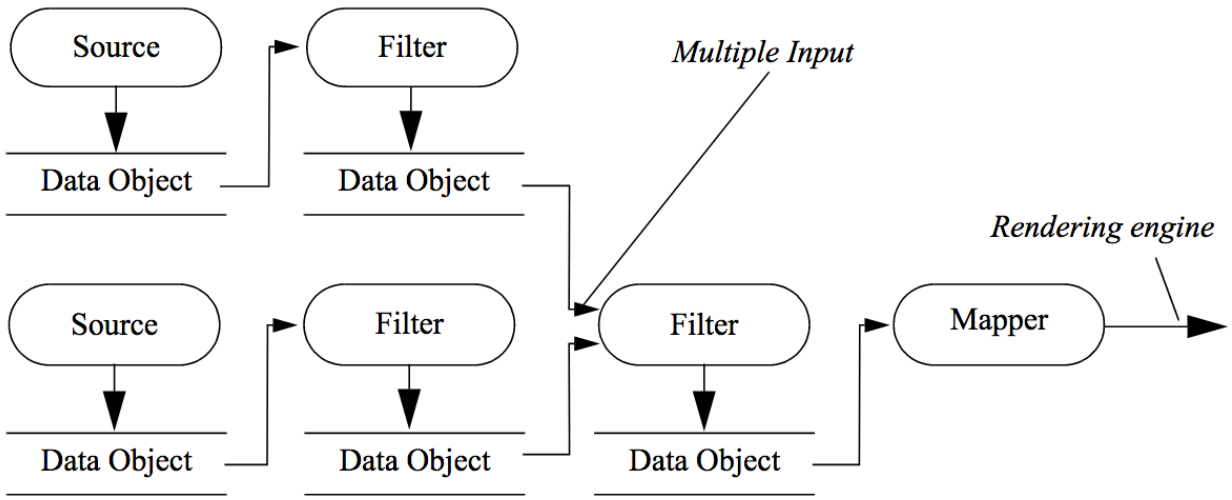
(d) Unstructured Points
(use vtkPolyData)



(e) Polygonal Data
(vtkPolyData)

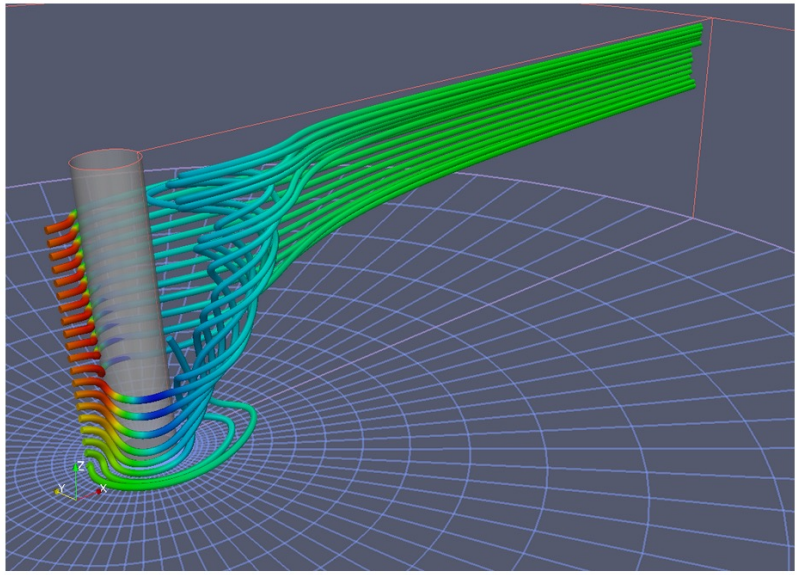


(f) Unstructured Grid
(vtkUnstructuredGrid)


Source
Filter
Multiple Input
Data Object
Data Object
Rendering engine
Source
Filter
Filter
Mapper
Data Object
Data Object
Data Object

- Develop an open-source, multi-platform visualization application.
- Support distributed computation models to process large data sets.
- Create an open, flexible, and intuitive user interface.
- Develop an extensible architecture based on open standards.

The History of ParaView

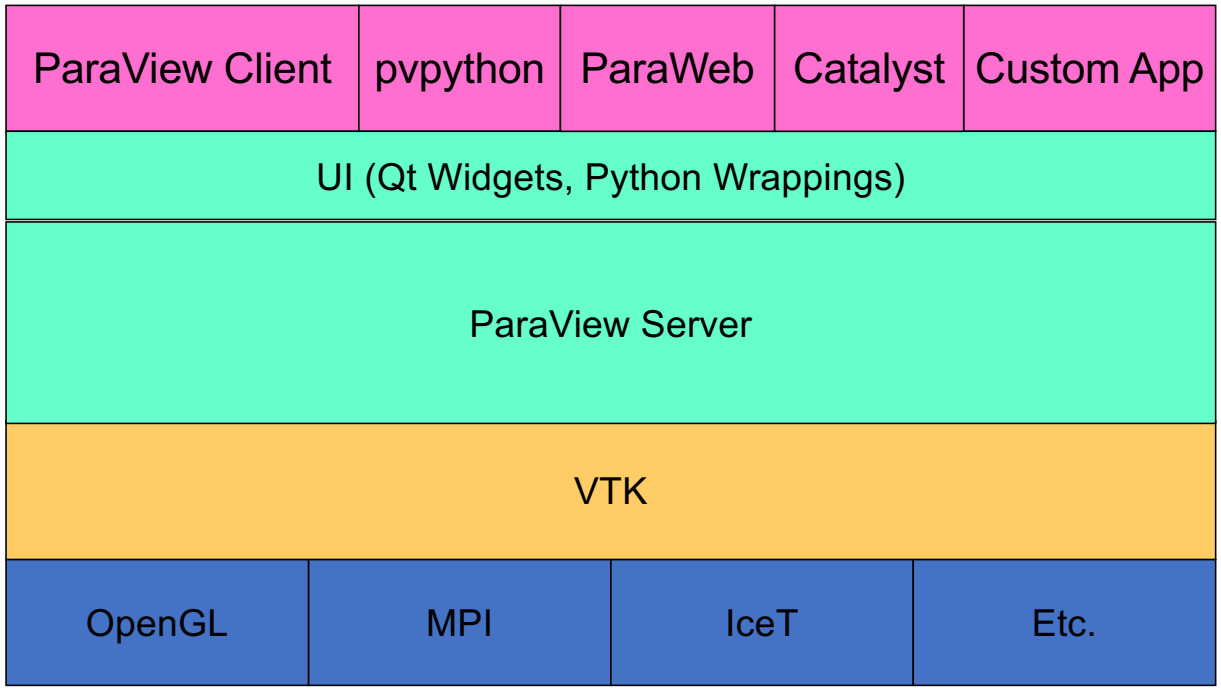


The ParaView project started in 2000 as a collaborative effort between Kitware Inc. and Los Alamos National Laboratory. The initial funding was provided by a three-year contract with the US Department of Energy ASCI Views program. The first public release, ParaView 0.6, was announced in October 2002.

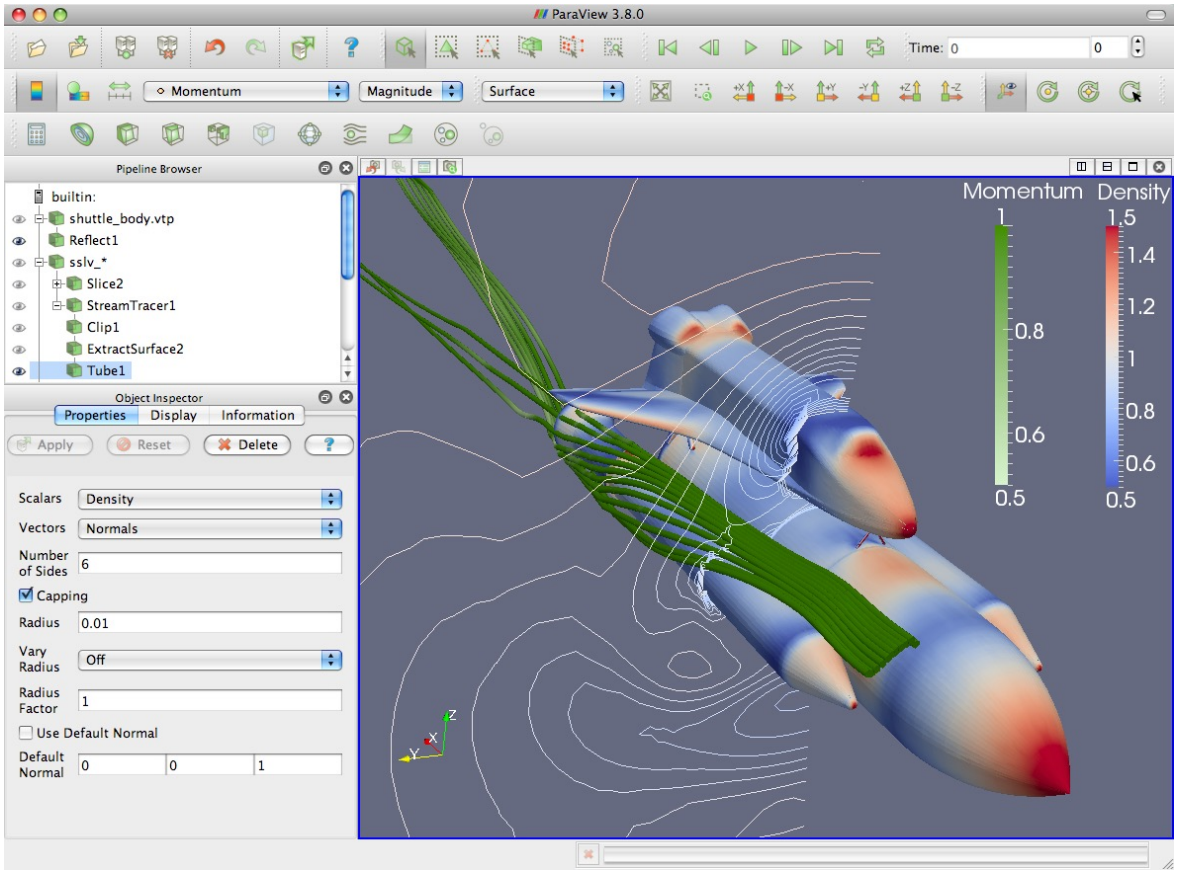
Independent of ParaView, Kitware started developing a web-based visualization system in December 2001. This project was funded by Phase I and II SBIRs from the US Army Research Laboratory and eventually became the ParaView Enterprise Edition. PVEE significantly contributed to the development of ParaView's client/server architecture.

Since the beginning of the project, Kitware has successfully collaborated with Sandia, Los Alamos National Laboratories, the Army Research Laboratory and various other academic and government

ParaView Application Architecture



ParaView Application Architecture



Also see : https://www.youtube.com/watch?v=Y1RATo2swM8

vtk.js

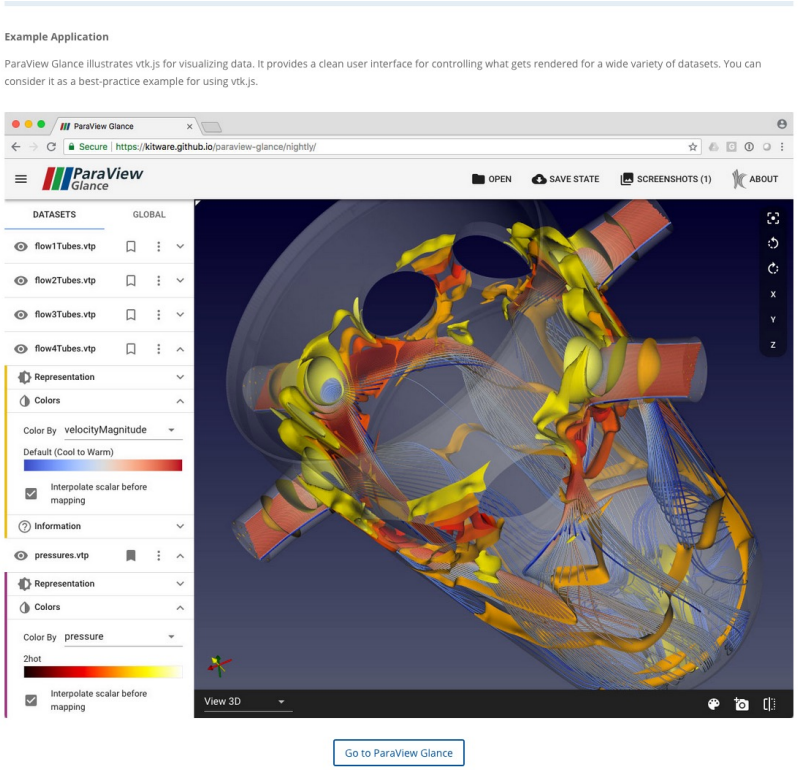
vtk.js is a re-implementation of VTK/C++ in JavaScript that leverages WebGL and focuses on geometry rendering and volume rendering.

https://kitware.github.io/vtk-js/examples/MultiSliceImageMapper.html

http://kitware.github.io/vtk-js/examples/VolumeContour.html

https://kitware.github.io/paraview-glance/app/

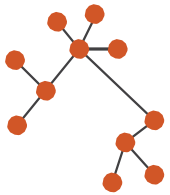
Paraview Glance uses vtk.js



Arrange networks and trees

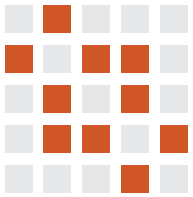
Node–Link Diagrams
Connection Marks

✔ NETWORKS ✔ TREES



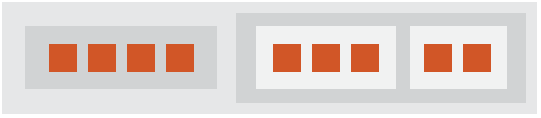
Adjacency Matrix
Derived Table

✔ NETWORKS ✔ TREES



Enclosure
Containment Marks

✘ NETWORKS ✔ TREES

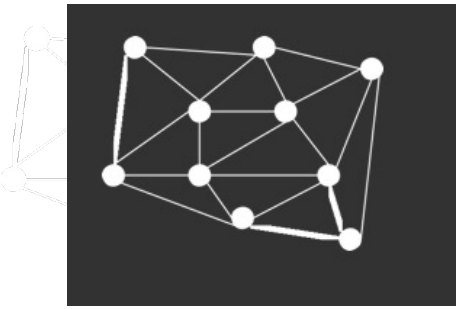


Graphs and Trees

Graphs

Model relations among data

Nodes and *edges*

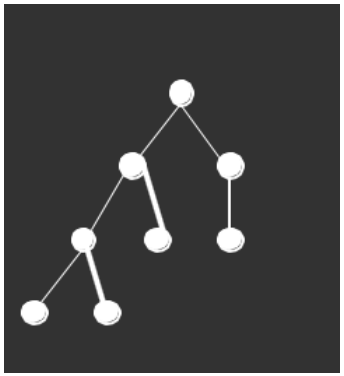


Trees

Graphs with hierarchical structure

Connected graph with N-1 edges

Nodes as *parents* and *children*



Slides based on lectures by J. Heer and T. Munzner

Spatial Layout

- A primary concern of tree/graph drawing is the spatial arrangement of nodes and edges.
- Often (but not always) the goal is to effectively depict the graph structure:

  Connectivity, path-following
  Topological distance
  Clustering / grouping
  Ordering (e.g., hierarchy level)

Applications

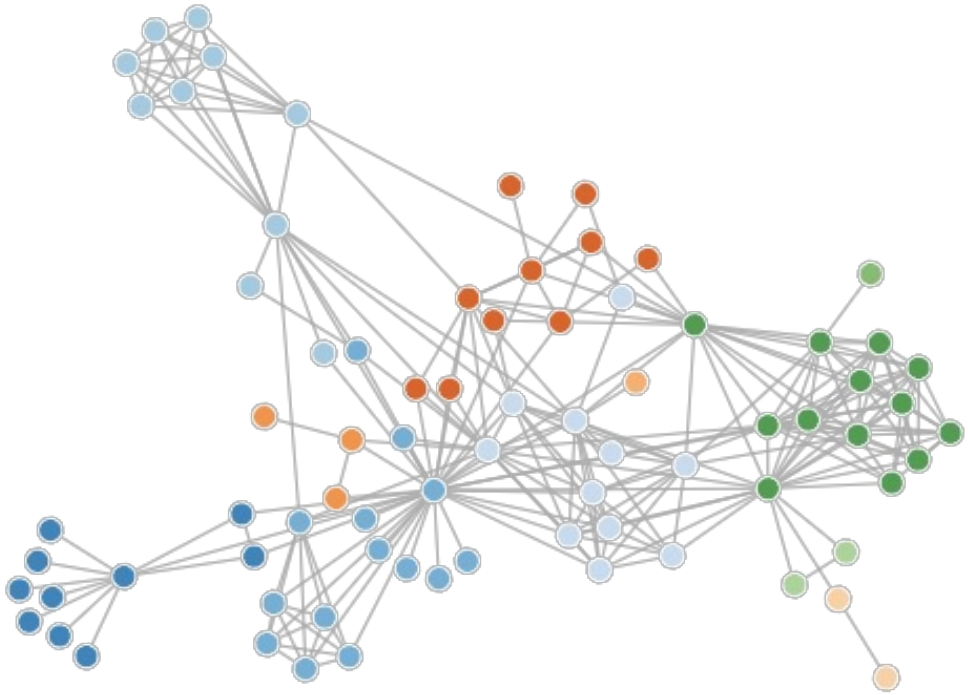
Tournaments
Organization Charts
Genealogy Diagramming
Biological Interactions
(Genes, Proteins)
Computer Networks
Social Networks
Simulation and Modeling
Integrated Circuit Design

Topics

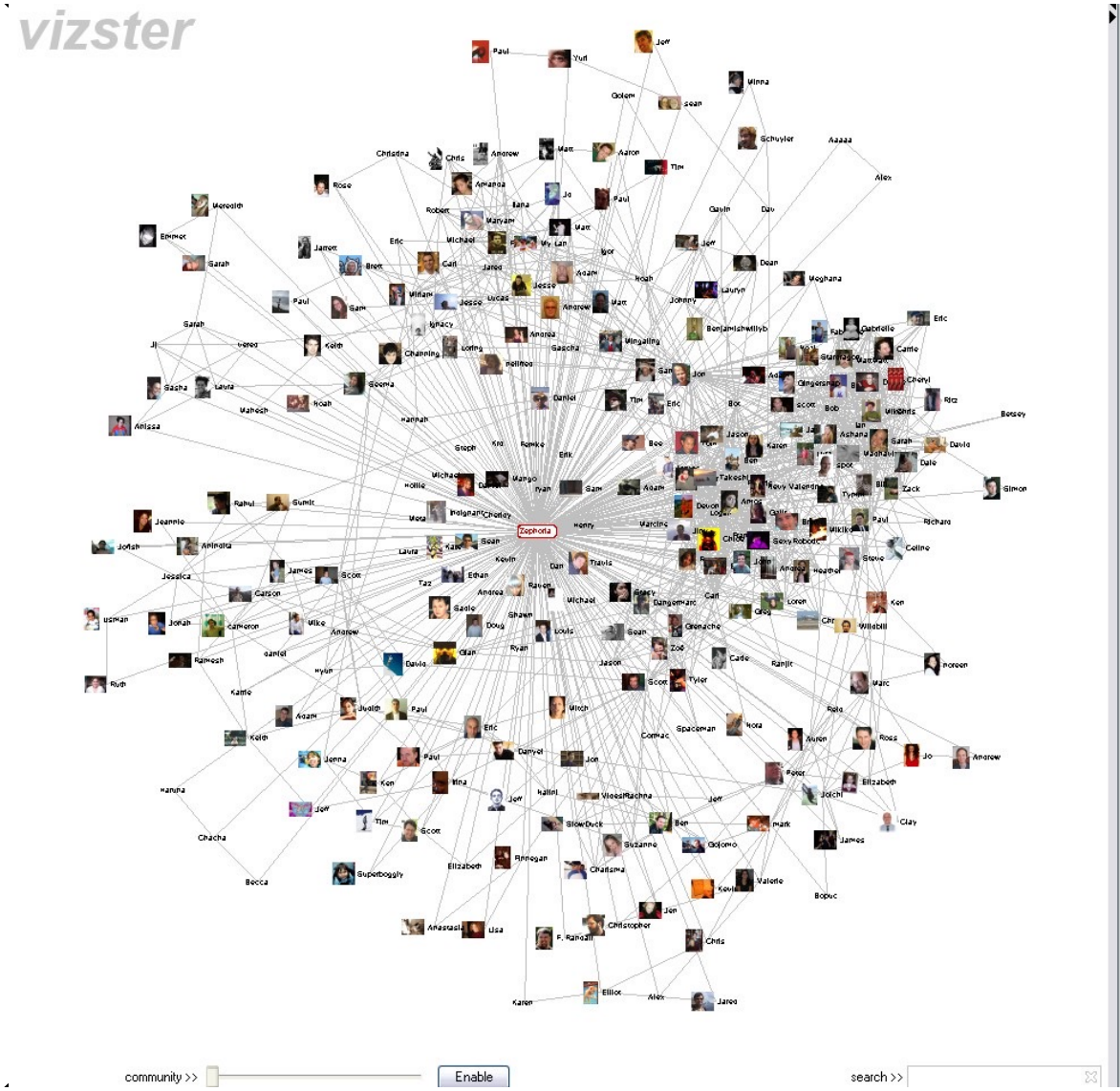
- Graph Layout: Node-Link Diagrams
  - Force-Directed Layout
  - Sugiyama-Style Layout
  - Alternatives to Node-Link
- Diagrams
  - Matrix Diagrams
  - Attribute-Driven Layout
- Tree Visualization

Node-Link Graph Layout

Force-Directed Layout



Interactive Example: Configurable Force Layout



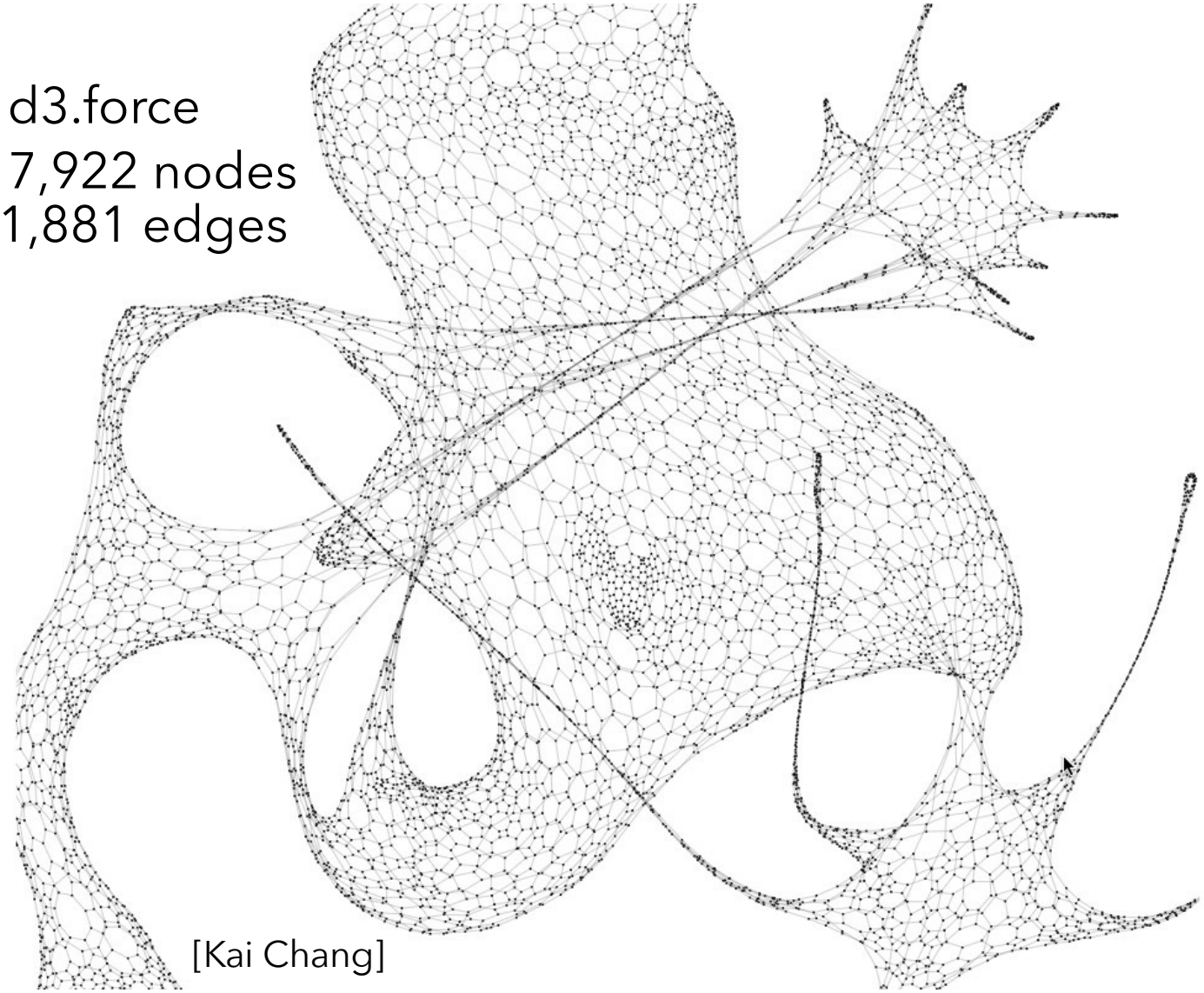
Zephoria

User ID	21721
Friends	266
Age	??
Gender	Female
Status	Single
Location	San Francisco, CA
Hometown	Lancaster, PA
Occupation	researcher: social networks, identity, context
Interests	apophenia, observing people, culture, questioning power, reading, buddhism, ipseity, computer-mediated communication, social networks, technology, anthropology, stomping
Music	psytrance/goa/trance [Infected Mushroom, Son Kite... Iboga/Digital Structures], Ani Difranco, downtempo, Thievery Corporation, Beth Orton, Morcheeba, Ween, White Stripes
Books	Authors: Erving Goffman, Stanley Milgram, Jeanette Winterson, Eric Schlosser, Leslie Feinberg, Dorothy Allison, Italo Calvino, Hermann Hesse
TV Shows	??
Movies	Koyaanisqatsi, Amelie, Waking Life, Tank Girl, The Matrix, Clockwork Orange, American Beauty, Fight Club, Boys Don't Cry
Member Since	??
Last Login	2003-10-21
Last Updated	2003-10-21
About	[Some know me as danah...] I'm a geek, an activist and an academic, fascinated by people and society. I see life as a very large playground and enjoy exploring its intricacies. I revel in life's chaos, while simultaneously providing my own insane element. My musings: http://www.zephoria.org/thoughts/
Want to Meet	Someone who makes life's complexities seem simply elegant

Use the Force!

http://mbostock.github.io/d3/talk/20110921/

d3.force
7,922 nodes
11,881 edges



[Kai Chang]

Force-Directed Layout

Nodes = charged particles — $F = q_i * q_j / d_{ij}^2$

with air resistance — $F = -b * v_i$

Edges = springs — $F = k * (L - d_{ij})$

Each node has charge q_i, and velocity v_i, distance between nodes is d_{ij}

Each edge has spring length L (no force if distance is L)

air resistance b and spring constant k given for all nodes and edges

Force-Directed Layout

Nodes = charged particles $F = q_i * q_j / d_{ij}^2$

with air resistance $F = -b * v_i$

Edges = springs $F = k * (L - d_{ij})$

At each timestep, calculate forces acting on nodes.

Integrate for updated velocities and positions.

D3's force layout uses **velocity Verlet** integration.

Assume uniform mass ***m*** and timestep ***Δt***:

$F = ma \rightarrow F = a \rightarrow F = \Delta v / \Delta t \rightarrow F = \Delta v$

Forces simplify to velocity offsets!