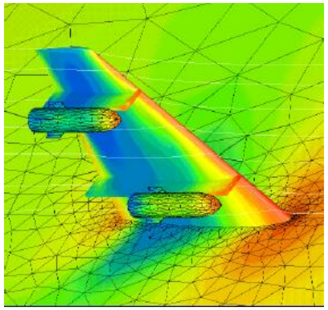
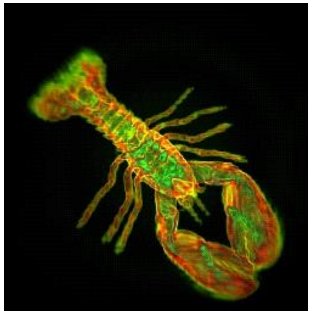
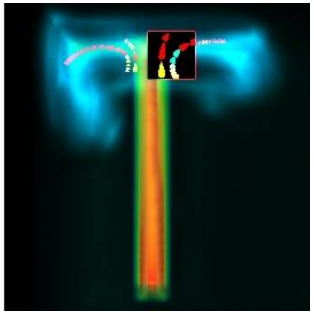
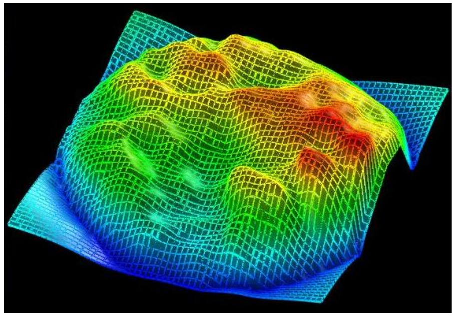
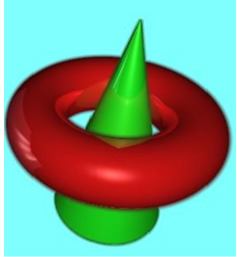
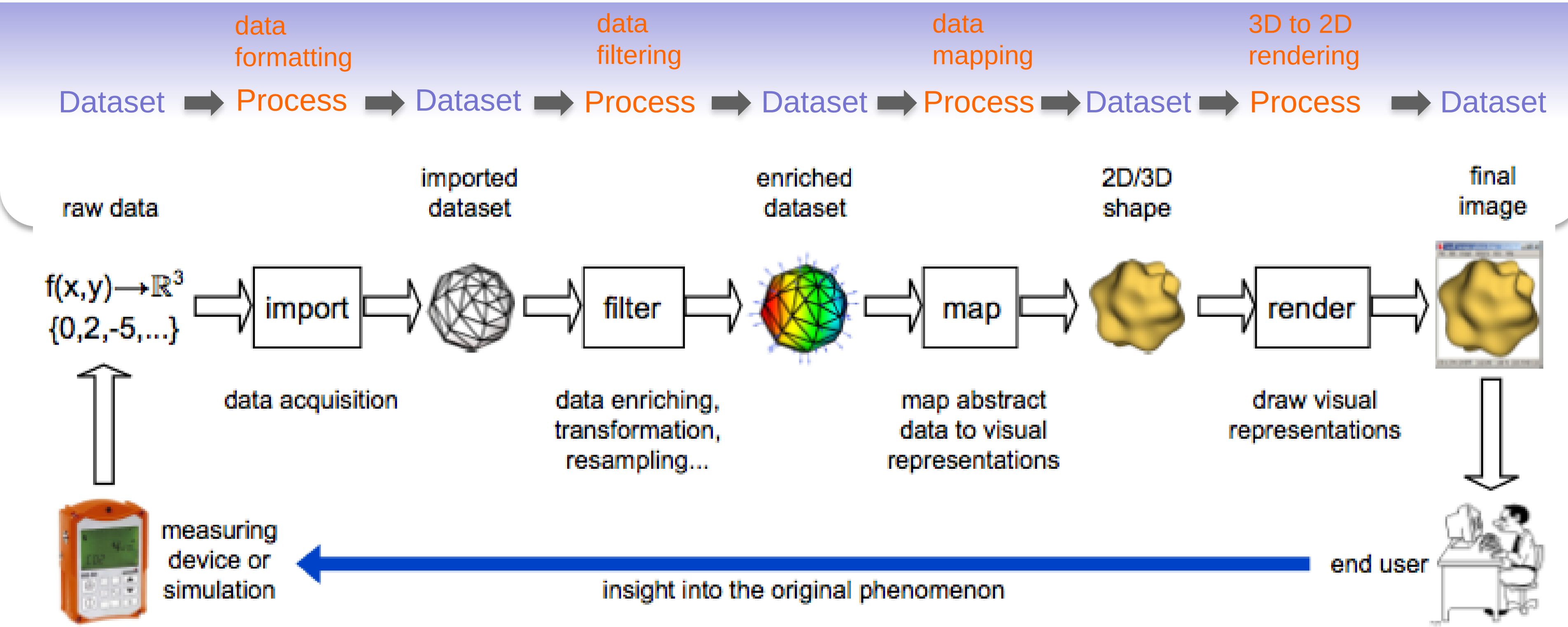
Vector Algorithms

The Visualization Pipeline - Recall



Vector algorithms (Chapter 6)

1. **Scalar derived quantities**
   - divergence, curl, vorticity
2. **0-dimensional shapes**
   - hedgehogs and glyphs
   - color coding
3. **1-dimensional and 2-dimensional shapes**
   - displacement plots
   - stream objects
4. **Image-based algorithms**
   - image-based flow visualization in 2D, curved surfaces, and 3D

Basic problem

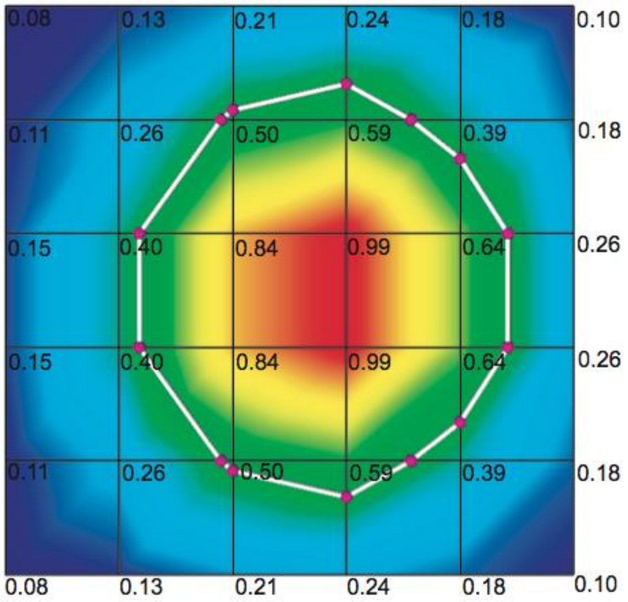
Input data

•vector field $v : D \rightarrow \mathbf{R}^n$

•domain D 2D planar surfaces, 2D surfaces embedded in 3D, 3D volumes

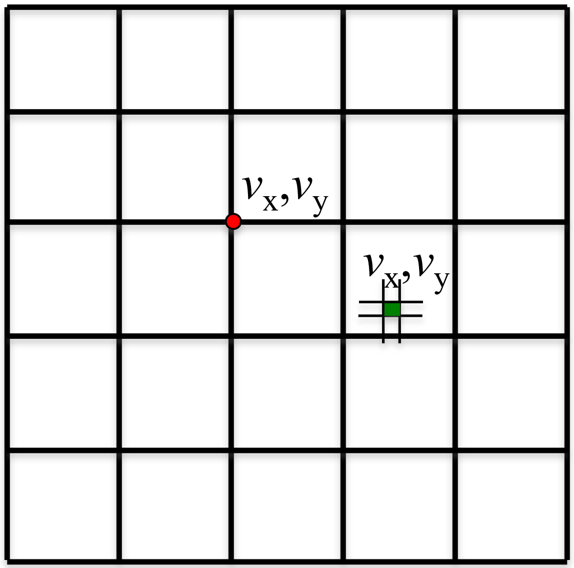
•variables n=2 (fields tangent to 2D surfaces) or n=3 (volumetric fields)

Challenge: comparison with scalar visualization



Scalar visualization

- challenge is to map D to 2D screen
- after that, we have 1 pixel per scalar value



Vector visualization

- challenge is to map D to 2D screen
- after that, we have 1 pixel for 2 or 3 scalar values!

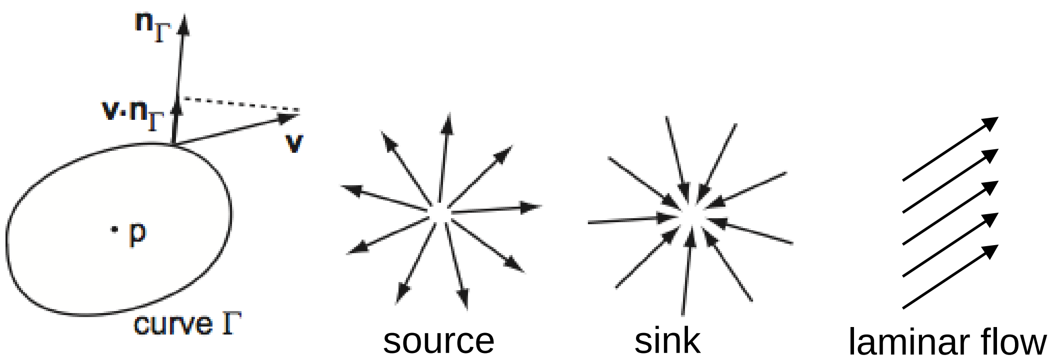
First solution: Reuse scalar visualization

- compute derived scalar quantities from vector fields
- use known scalar visualization methods for these

1. Divergence

- think of vector field as encoding a fluid flow
- intuition: amount of mass (air, water) created, or absorbed, at a point in D
- given a field $\mathbf{v} : \mathbf{R}^3 \rightarrow \mathbf{R}^3$, $\mathrm{div}\ \mathbf{v} : \mathbf{R}^3 \rightarrow \mathbf{R}$ is

$$\mathrm{div}\ \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \quad \text{equivalent to} \quad \mathrm{div}\ \mathbf{v} = \lim_{\Gamma \to 0} \frac{1}{|\Gamma|} \int_\Gamma (\mathbf{v} \cdot \mathbf{n}_\Gamma) \mathrm{d}s$$



source
div $\mathbf{v}$>0

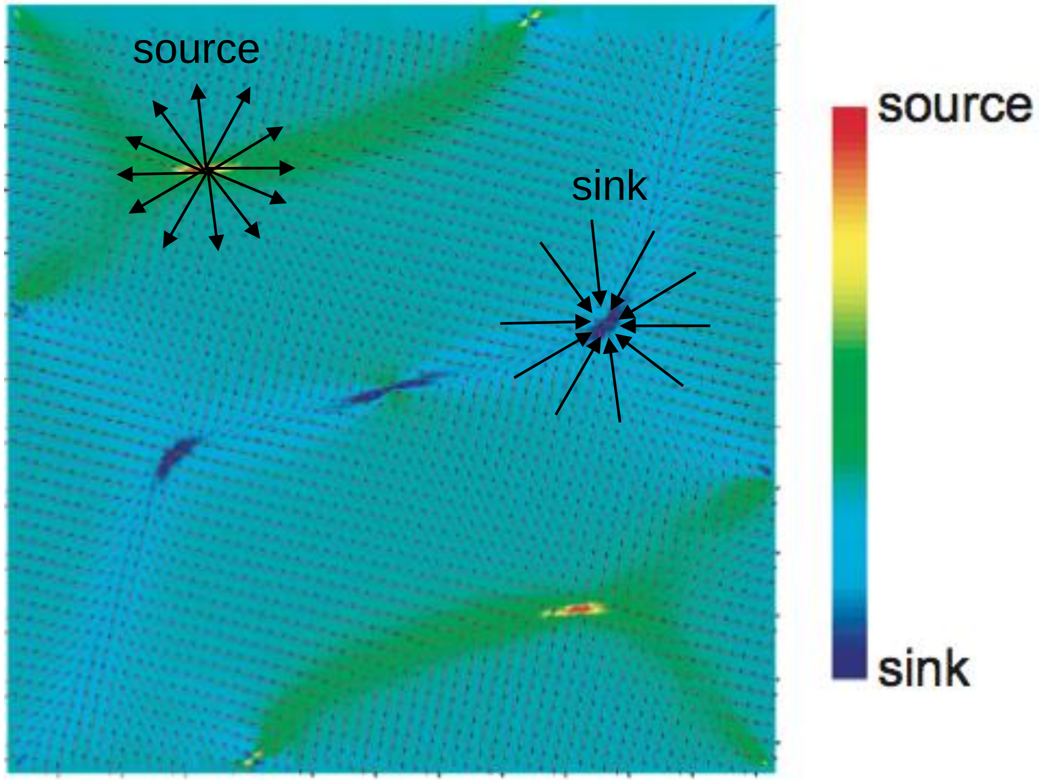
sink
div $\mathbf{v}$<0

laminar flow
div $\mathbf{v}$=0

div $\mathbf{v}$ is sometimes denoted as $\nabla \cdot v$

Divergence

•compute using definition with partial derivatives
•visualize using e.g. color mapping



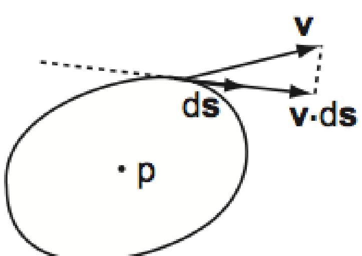
•gives a good impression of where the flow 'enters' and 'exits' some domain

Curl

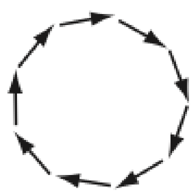
2. Curl (also called rotor)

- consider again a vector field as encoding a fluid flow
- intuition: how quickly the flow 'rotates' around each point?
- given a field $\mathbf{v} : \mathbf{R}^3 \to \mathbf{R}^3$, $\text{rot } \mathbf{v} : \mathbf{R}^3 \to \mathbf{R}^3$ is

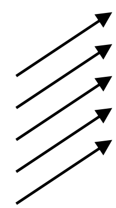
$$\text{rot } \mathbf{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right) \quad \text{equivalent to} \quad \text{rot } \mathbf{v} = \lim_{\Gamma \to 0} \frac{1}{|\Gamma|} \int_\Gamma \mathbf{v} \cdot d\mathbf{s}$$



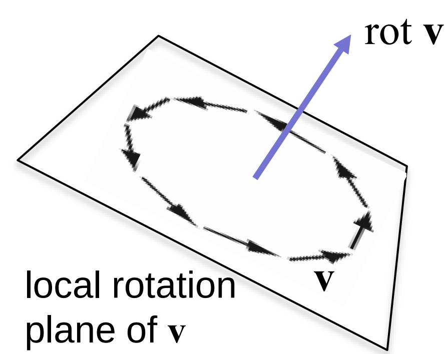
curve Γ



rotational flow
$||\text{rot } \mathbf{v}|| > 0$



laminar flow
$\text{rot } \mathbf{v} = \mathbf{0}$



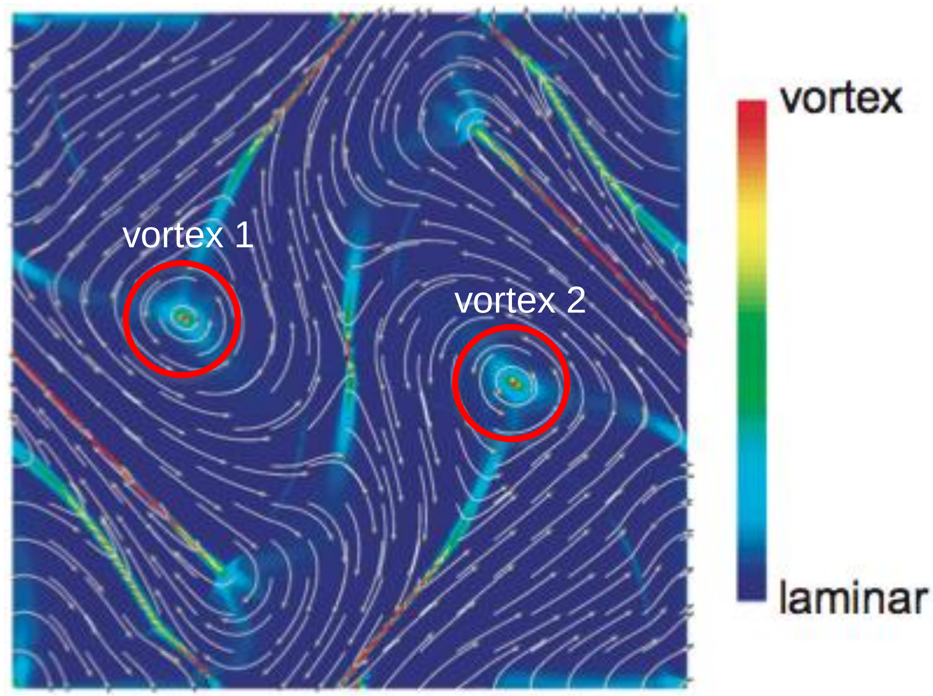
local rotation
plane of **v**

- $\text{rot } \mathbf{v}$ is locally perpendicular to plane of rotation of $\mathbf{v}$
- its magnitude: 'tightness' of rotation – also called vorticity

$\text{rot } \mathbf{v}$ is sometimes denoted as $\nabla \times v$

Curl

- compute using definition with partial derivatives
- visualize magnitude $\|\mathrm{rot}\ \mathbf{v}\|$ using e.g. color mapping

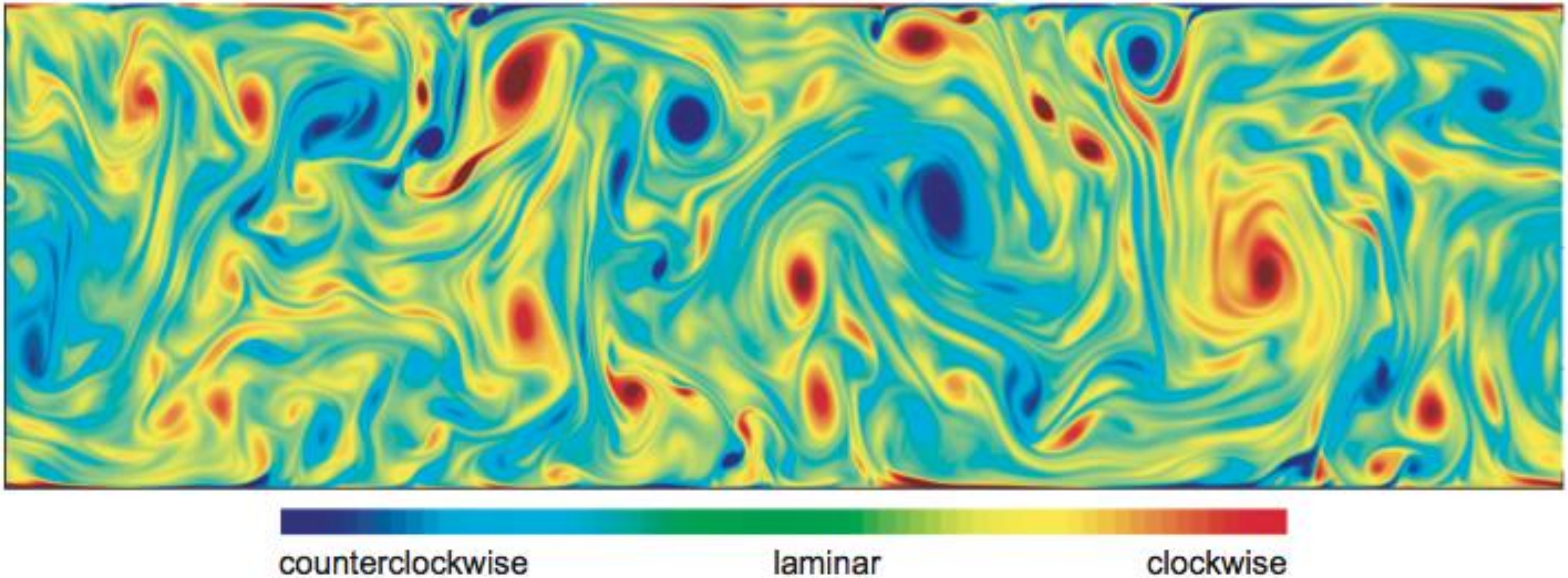


- very useful in practice to find vortices = regions of high vorticity
- these are highly important in flow simulations (aerodynamics, hydrodynamics)

Curl

Example of vorticity

- 2D fluid flow
- simulated by solving Navier-Stokes equations
- visualized using vorticity



Observations

- vortices appear at different scales
- see the ‘pairing’ of vortices spinning in opposite directions
- what happens with the flow close to the boundary? Why

Vector glyphs

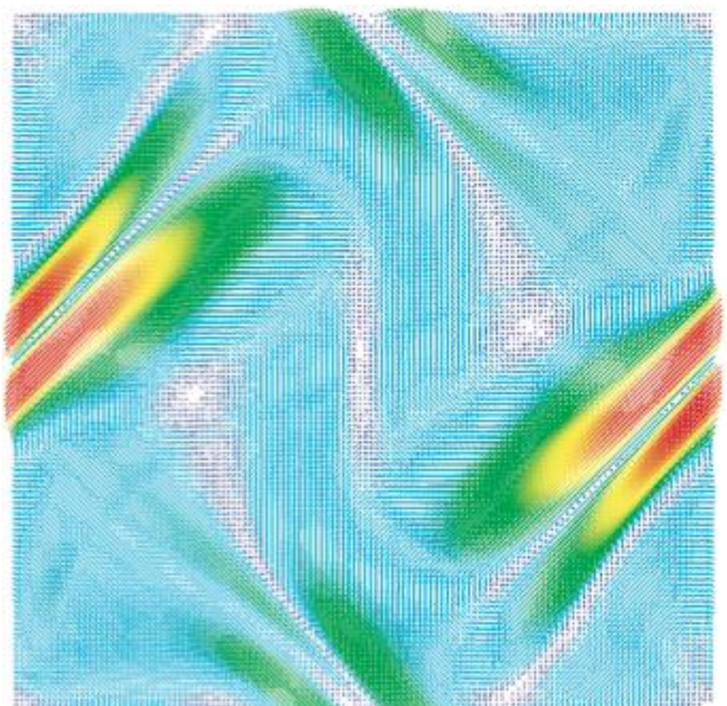
Icons, or signs, for visualizing vector fields

•placed by (sub)sampling the dataset domain
•attributes (scale, color, orientation) map vector data at sample points

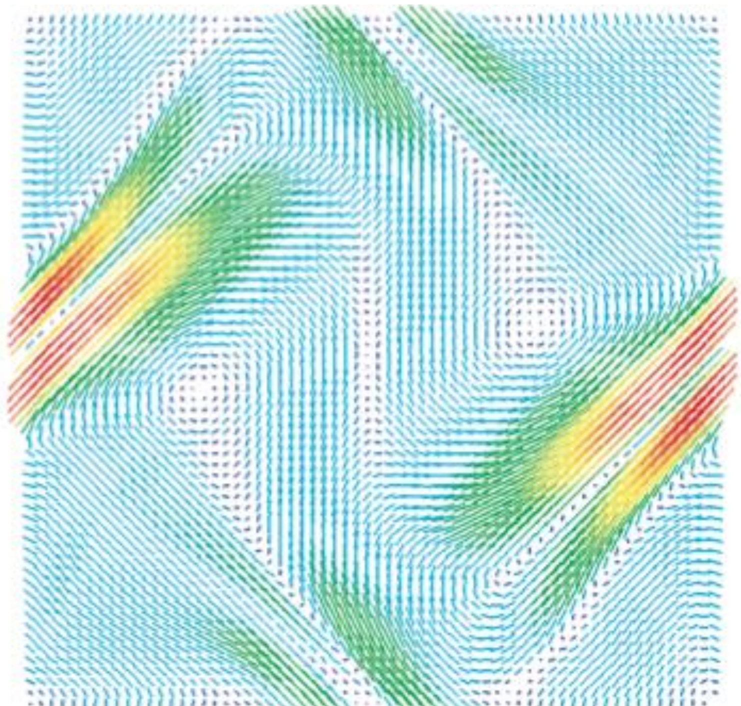
Simplest glyph: Line segment (hedgehog plots)

•for every sample point $x \in D$

- draw line $(x, x + k\mathbf{v}(x))$
- optionally color map $\|\mathbf{v}\|$ onto it



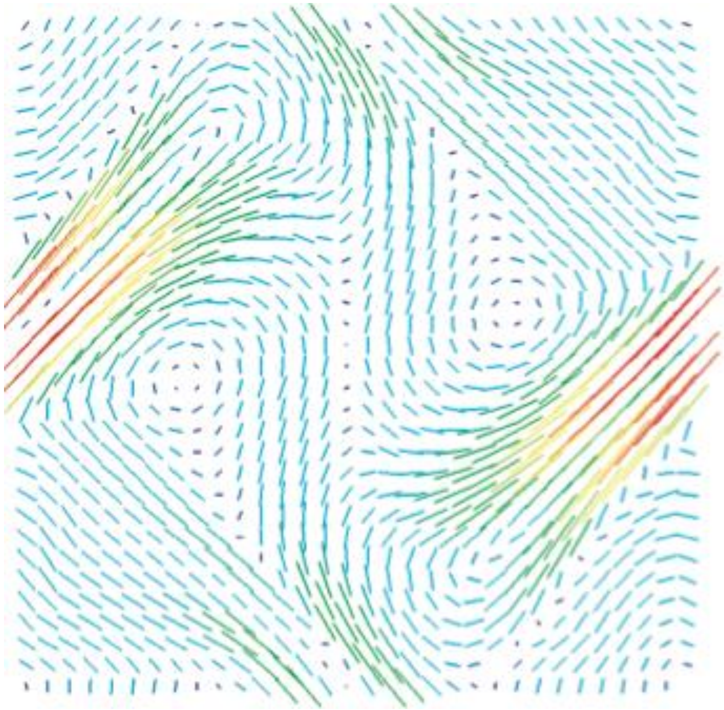
128^2 glyph grid



64^2 glyph grid

MHD simulation
256^2 grid

Vector glyphs



32^2 glyph grid

32^2 glyph grid, no line scaling

MHD simulation
256^2 grid

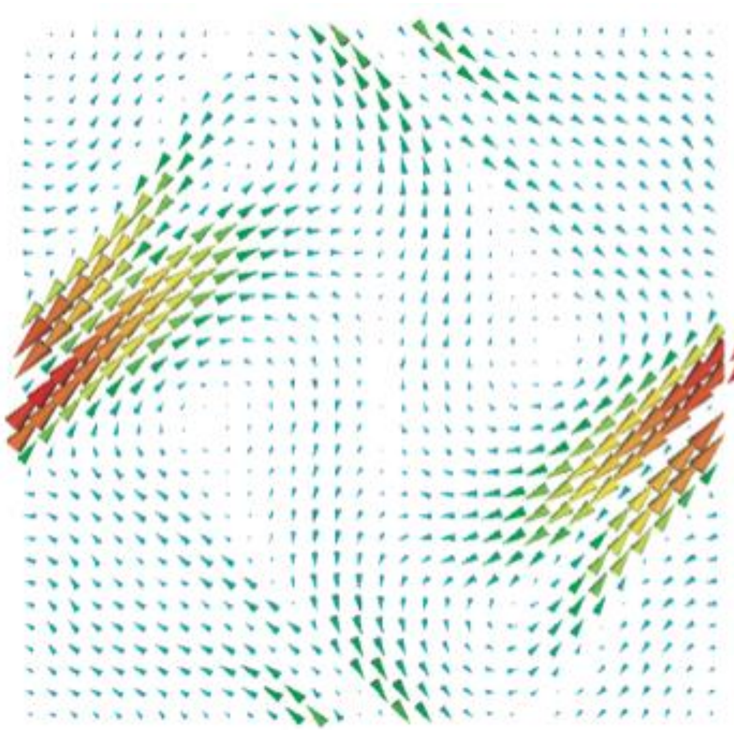
Observations

•trade-offs

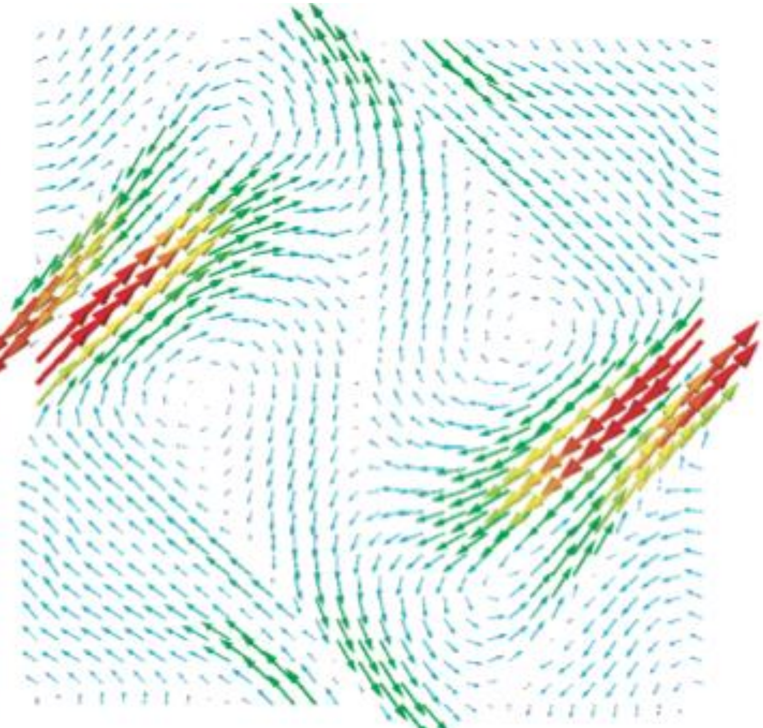
- more samples: more data points depicted, but more potential clutter
- less samples: less data points depicted, but higher clarity
- more line scaling: easier to see high-speed areas, but more clutter
- less line scaling: less clutter, but harder to perceive directions

Can you observe other pro's and con's of line glyphs?

Vector glyphs



3D cone glyphs



3D arrow glyphs

MHD simulation
256^2 grid

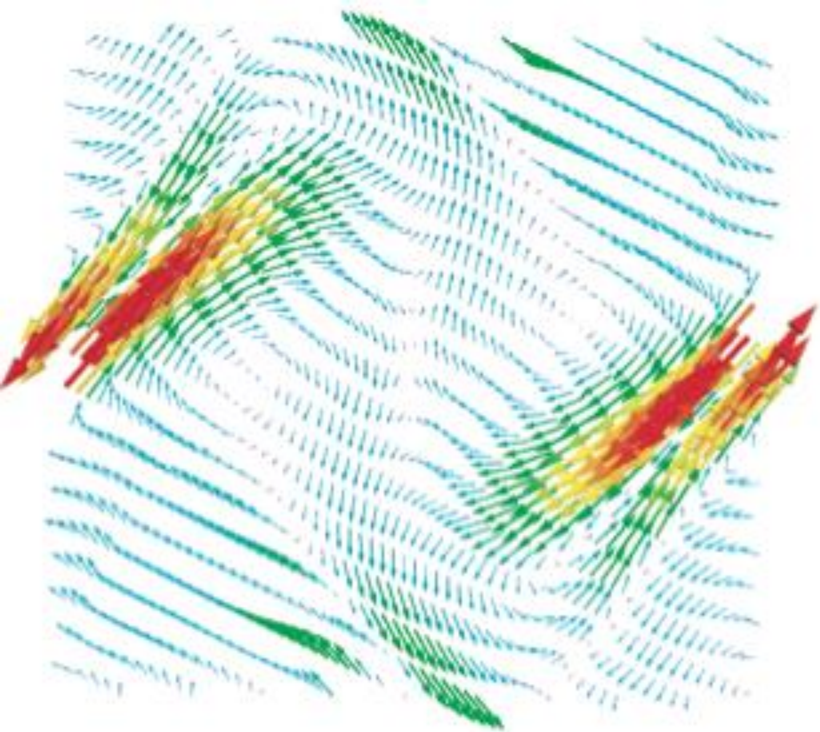
Variants

•cones, arrows, …

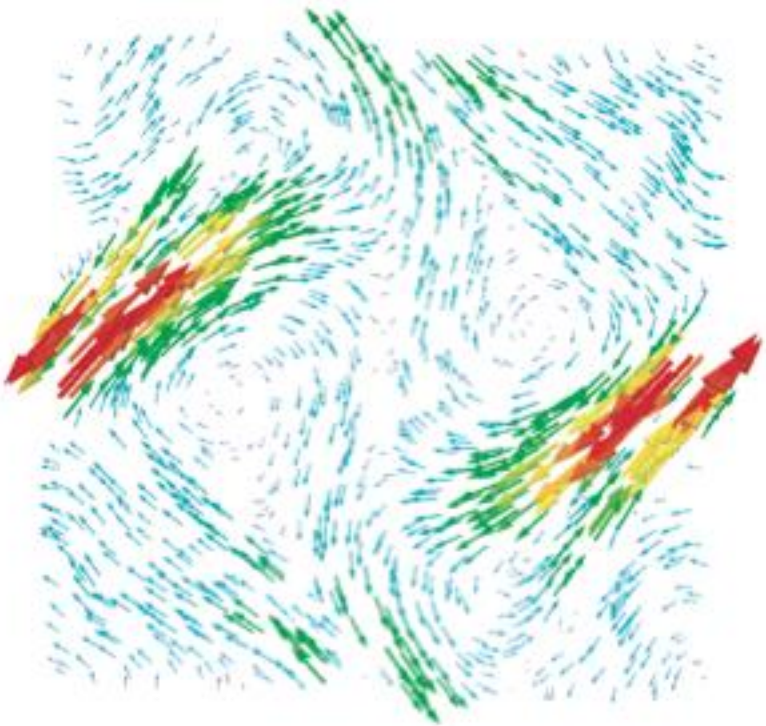
- show *orientation* better than lines
- but take more space to render
- shading: good visual cue to separate (overlapping) glyphs

Can you observe other pro's and con's of cone or arrow glyphs?

Vector glyphs



samples on a rotated grid



random samples, quasi-uniform density

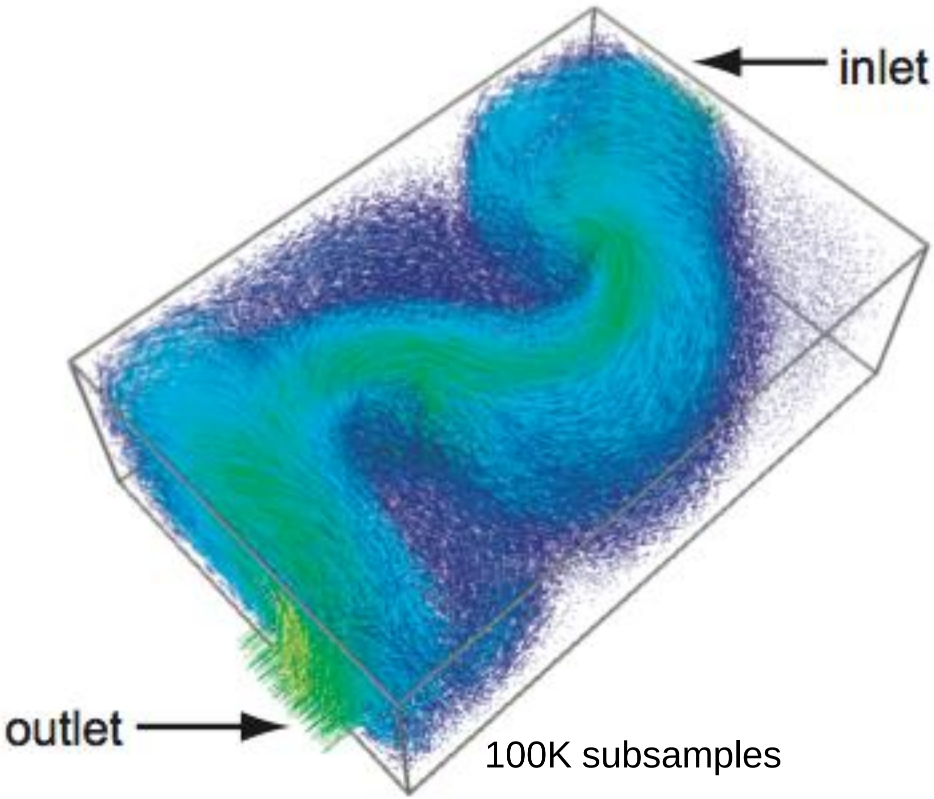
How to choose sample points

•avoid uniform grids!

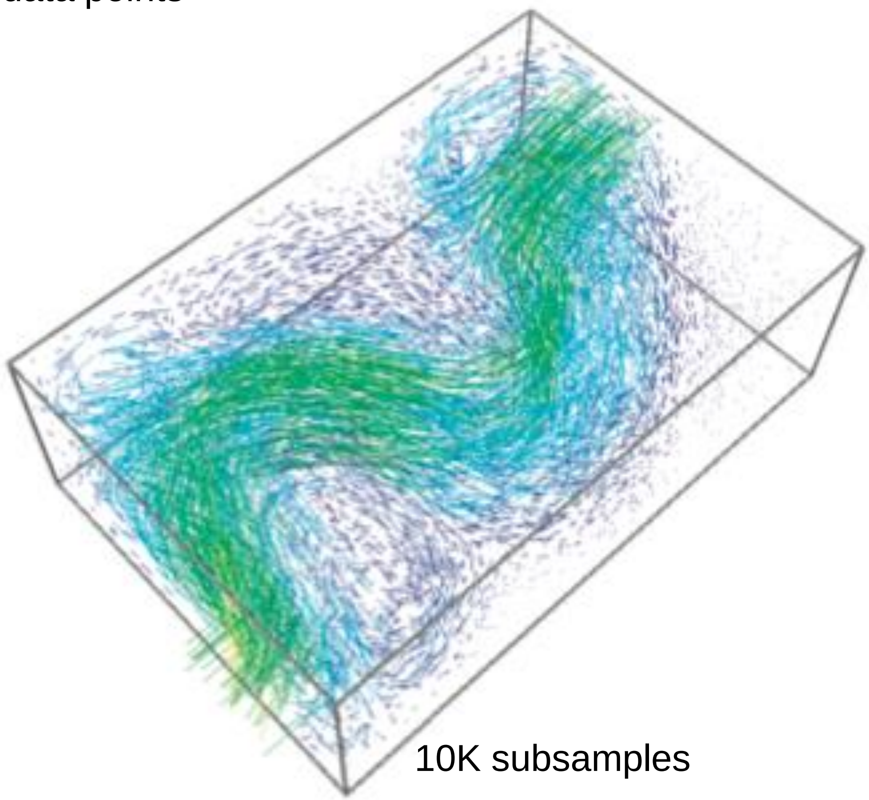
•random sampling: generally OK

3D vector glyphs

128x85x42 volume field
456960 data points



100K subsamples

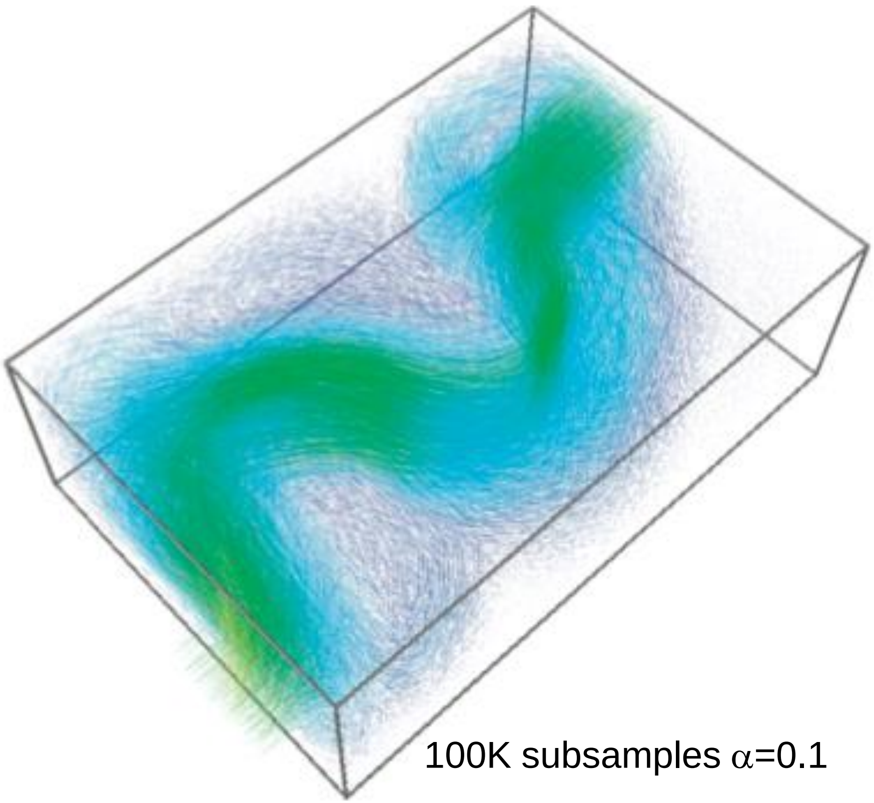


10K subsamples

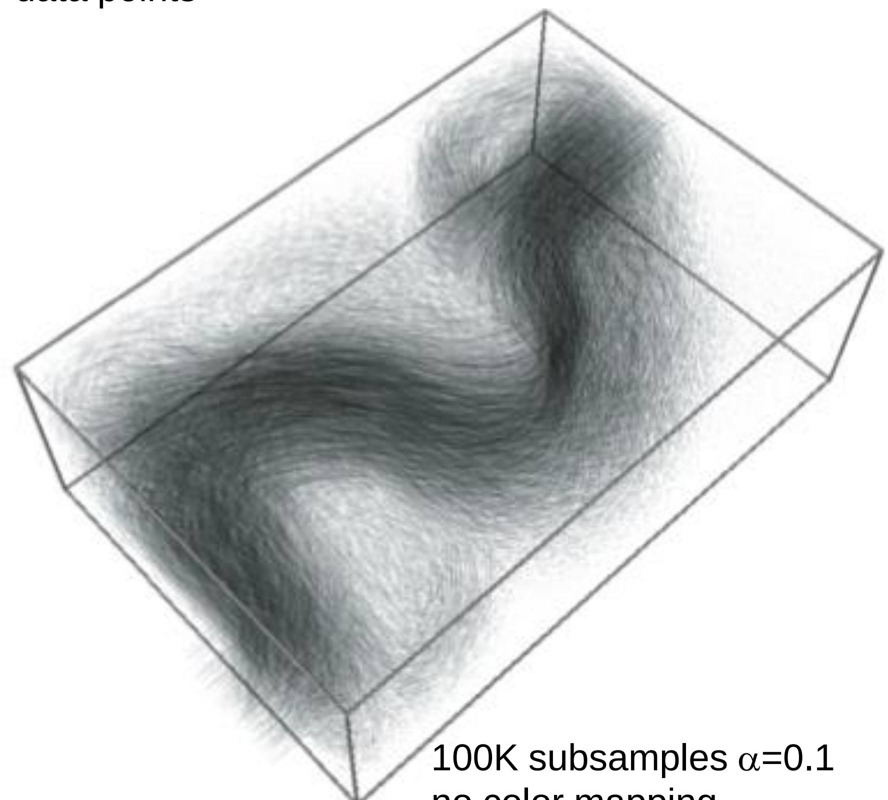
- same idea/technique as 2D vector glyphs
- 3D additional problems
  - more data, same screen space
  - occlusion
  - perspective foreshortening
  - viewpoint selection

3D vector glyphs

128x85x42 volume field
456960 data points



100K subsamples α=0.1

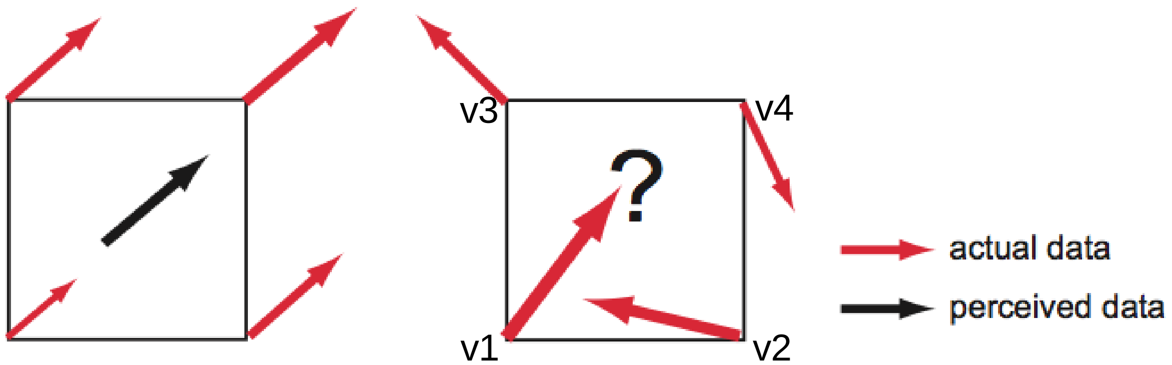


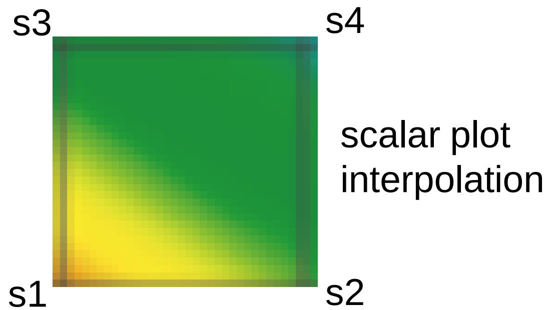
100K subsamples α=0.1
no color mapping

Alpha blending

- extremely simple and powerful tool
- reduce *perceived* occlusion
    - low-speed zones: highly transparent
    - high-speed zones: opaque and highly coherent (why?)

Glyph problem revisited





Recall the 'inverse mapping' proposal

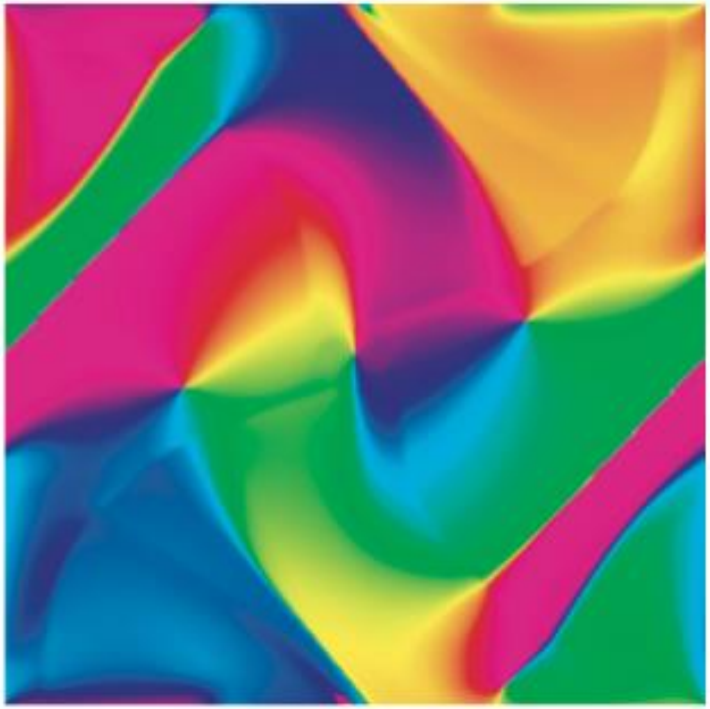
- •we render something…
- •…so we can visually map it to some data/phenomenon

Glyph problems

- •**no interpolation** in glyph space (unlike for scalar plots with color mapping!)
- •a glyph takes more space than a pixel
- •we (humans) aren't good at visually interpolating arrows…
- •scalar plots are **dense**; glyph plots are **sparse**
  - this is why glyph positioning (sampling) is extra important

Vector color coding


magnitude=luminance


constant luminance (direction coding only)


direction
color wheel

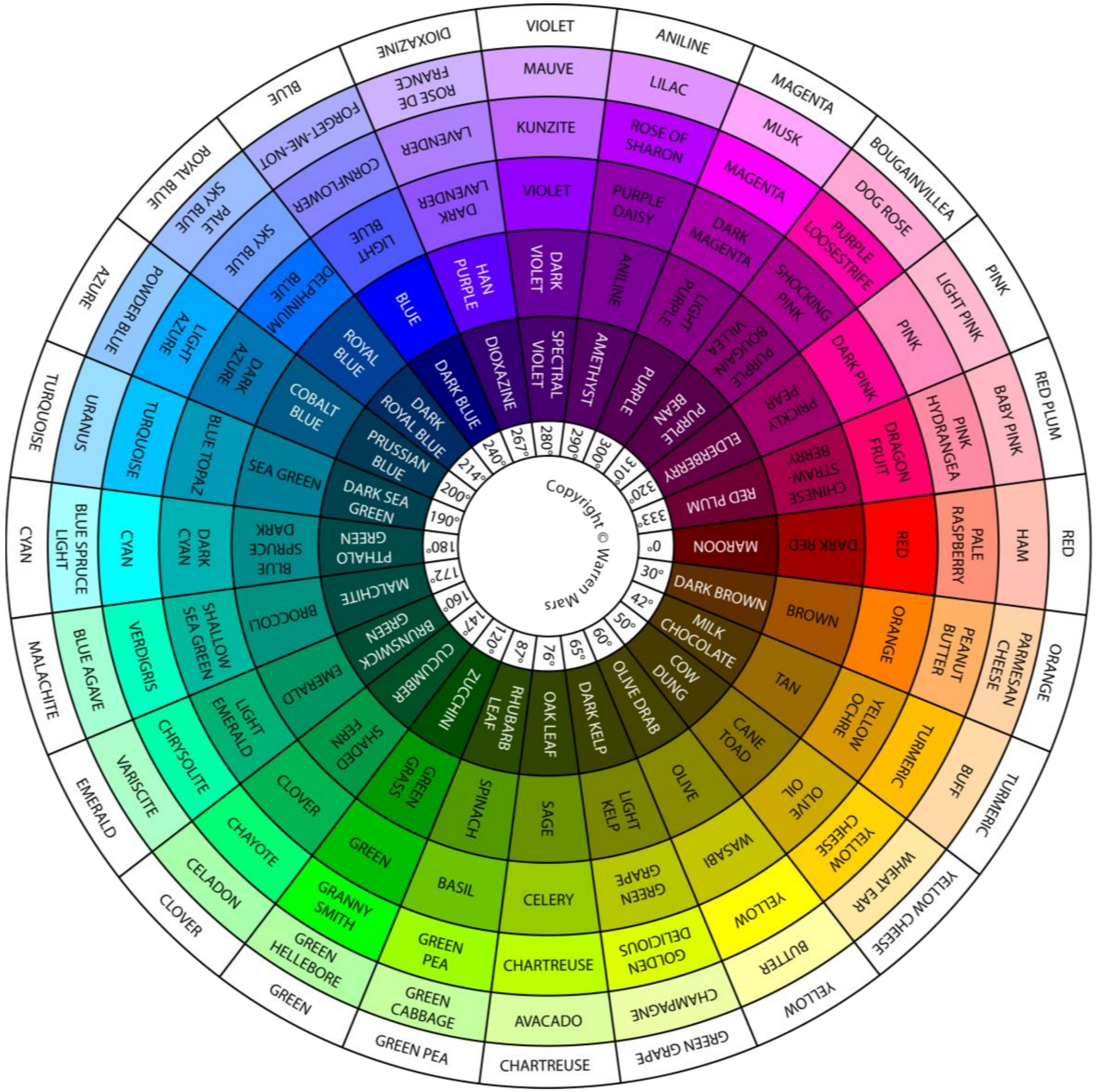
Reduce vector data to scalar data (using HSV color model)

•direction = hue

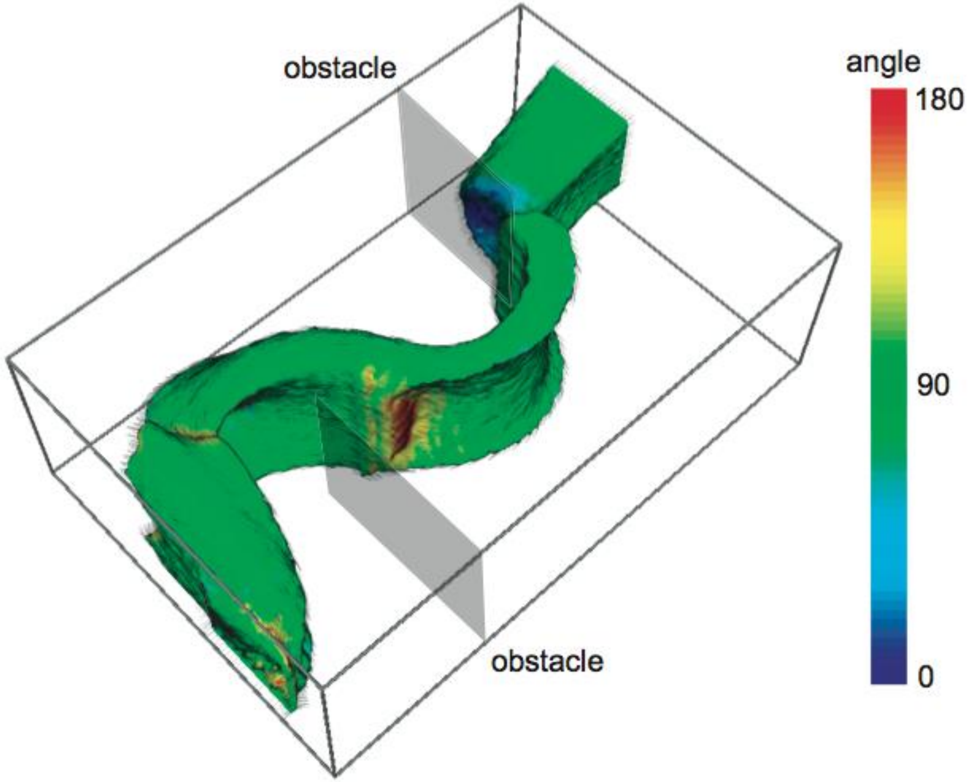
•magnitude = luminance (optional)

•no occlusion/interpolation problems…

•…but images are highly abstract (recall: we don’t naturally *see* directions)


VIOLET
ANILINE
MAGENTA
BOUGAINVILLEA
PINK
RED PLUM
RED
ORANGE
TURMERIC
YELLOW CHEESE
YELLOW
GREEN GRAPE
CHARTREUSE
GREEN PEA
GREEN
CLOVER
EMERALD
MALACHITE
CYAN
TURQUOISE
AZURE
ROYAL BLUE
BLUE
DIOXAZINE
MAUVE
LILAC
MUSK
DOG ROSE
LIGHT PINK
BABY PINK
HAM
PARMESAN CHEESE
BUFF
WHEAT EAR
BUTTER
CHAMPAGNE
AVACADO
GREEN CABBAGE
GREEN HELLEBORE
CELADON
VARISCITE
BLUE AGAVE
BLUE SPRUCE LIGHT
URANUS
POWDER BLUE
PALE SKY BLUE
FORGET-ME-NOT
ROSE DE FRANCE
KUNZITE
ROSE OF SHARON
MAGENTA
PURPLE LOOSESTRIFE
PINK
PINK HYDRANGEA
PALE RASPBERRY
PEANUT BUTTER
TURMERIC
YELLOW CHEESE
YELLOW
GOLDEN DELICIOUS
CHARTREUSE
GREEN PEA
GRANNY SMITH
CHAYOTE
CHRYSOLITE
VERDIGRIS
CYAN
TURQUOISE
LIGHT AZURE
SKY BLUE
CORNFLOWER
LAVENDER
VIOLET
PURPLE DAISY
DARK MAGENTA
SHOCKING PINK
DARK PINK
DRAGON FRUIT
RED
ORANGE
YELLOW OCHRE
OLIVE OIL
WASABI
GREEN GRAPE
CELERY
BASIL
GREEN
CLOVER
LIGHT EMERALD
SHALLOW SEA GREEN
DARK CYAN
BLUE TOPAZ
DARK AZURE
DELPHINIUM BLUE
LIGHT BLUE
DARK LAVENDER
DARK VIOLET
ANILINE
LIGHT PURPLE
PURPLE BOUGAIN-VILLEA
PRICKLY PEAR
CHINESE STRAW-BERRY
DARK RED
BROWN
TAN
CANE TOAD
OLIVE
LIGHT KELP
SAGE
SPINACH
GREEN GRASS
SHADED FERN
EMERALD
BROCCOLI
BLUE SPRUCE DARK
SEA GREEN
COBALT BLUE
ROYAL BLUE
BLUE
HAN PURPLE
SPECTRAL VIOLET
AMETHYST
PURPLE
PURPLE BEAN
ELDERBERRY
RED PLUM
MAROON
DARK BROWN
MILK CHOCOLATE
COW DUNG
OLIVE DRAB
DARK KELP
OAK LEAF
RHUBARB LEAF
ZUCCHINI
CUCUMBER
BRUNSWICK GREEN
MALCHITE
PTHALO GREEN
DARK SEA GREEN
PRUSSIAN BLUE
DARK ROYAL BLUE
DARK BLUE
DIOXAZINE
280°
290°
300°
310°
320°
333°
0°
30°
42°
50°
60°
65°
76°
87°
120°
147°
160°
172°
180°
190°
200°
214°
240°
267°
Copyright © Warren Mars

Vector color coding



color = angle between vector field and normal of some given surface

See if vectors are tangent to some given surface

- color-code angle between vector and surface normal
- easily spot
    - tangent regions (flow stays on surface, green)
    - inflow regions (flow enters surface, red)
    - outflow regions (flow exits surface, blue)