Vector Field Visualization:
Introduction
What is a Vector Field?

A vector-valued function that assigns a vector (with direction and magnitude) to any given point.

It typically can be expressed as an ordinary differential equation, i.e., ODE.

\[ \frac{d\varphi(x)}{dt} = V(x) \]

Its solution gives rise to a “flow”, which consists of densely placed particle trajectories (i.e., the red curve shown in the example).
Why Is It Important?
Applications in Engineering and Science

Automotive design [Chen et al. TVCG07, TVCG08]

Weather study [Bhatia and Chen et al. TVCG11]

Oil spill trajectories [Tao et al. EMI2010]

Aerodynamics around missiles [Kelly et al. Vis06]
Applications in Computer Graphics

Texture Synthesis [Chen et al. TVCG12b]

Fluid simulation [Chenney SCA2004, Cao & Chen 2013]

Parameterization [Ray et al. TOG2006]

Smoke simulation [Shi and Yu TOG2005]

Painterly Rendering [Zhang et al. TOG2006]

Shape Deformation [von Funck et al. 2006]
Why Is It Challenging?

• Need to effectively visualize both magnitude + direction, often simultaneously

• Additional challenges:
  – large data sets
  – time-dependent data

*magnitude only*  

*direction only*
Classification of Visualization Techniques

- **Direct method**: overview of vector fields, minimal computation, e.g. glyphs, color mapping.
- **Texture-based**: covers domain with a convolved texture, e.g., Spot Noise, LIC, LEA, ISA, IBFV(S).
- **Geometric**: a discrete object(s) whose geometry reflects (e.g. tangent to) flow characteristics, e.g. integral curves.
- **Feature-based**: both automatic and interactive feature-based techniques, e.g. flow topology, vortex core structure, coherent structure, LCS, etc.
Flow Data

Data sources:

- **flow simulation:**
  - airplane- / ship- / car-design
  - weather simulation (air-, sea-flows)
  - medicine (blood flows, etc.)

- **flow measurement:**
  - wind tunnels, water channels
  - optical measurement techniques

- **flow models (analytic):**
  - differential equation systems (dynamic systems)
Flow Data

Simulation:
- flow: estimate (partial) differential equation systems (e.g. a physical model)
- set of samples (3/4-dims. of data), e.g., given on a curvilinear grid
- most important primitive: tetrahedron and hexahedron (cell)
- could be adaptive grids

Analytic:
- flow: analytic formula, differential equation systems $\frac{dx}{dt}$ (dynamical system)
- evaluated where-ever needed (e.g. making plots of flow in MatLab)

Measurement:
- vectors: taken from instruments, often estimated on a uniform grid
- optical methods + image recognition, e.g.: PIV (particle image velocimetry)
Notes on Computational Fluid Dynamics

- We often visualize Computational Fluid Dynamics (CFD) *simulation* data
- CFD is the discipline of predicting flow behavior, quantitatively
- data is (often) the result of a *simulation* of flow through or around an object of interest

some characteristics of CFD data:
- large, often gigabytes
- Unsteady, i.e. time-dependent
- unstructured, adaptive resolution grids
- Smooth field

Image source: Google images
Comparison with Reality

Experiment

Simulation

Really close but not exact
Types of the vector field data
2D vs. 2.5D Surfaces vs. 3D

2D flow visualization
• $R^2$ flows
• Planes, or flow layers (2D cross sections through 3D)

2.5D, i.e. surface flow visualization
• 3D flows around obstacles
• boundary flows on manifold surfaces (locally 2D)

3D flow visualization
• $R^3$ flows
• simulations, 3D domains
Steady vs. Time-dependent

Steady (time-independent) flows:
• flow itself constant over time
• \( \mathbf{v}(\mathbf{x}) \), e.g., laminar flows
• simpler case for visualization
• well understood behaviors and features

Time-dependent (unsteady) flows:
• flow itself changes over time
• \( \mathbf{v}(\mathbf{x},t) \), e.g., combustion flow, turbulent flow, wind field
• more complex cases
• *no unified theory to characterize them yet!*
Time-independent (steady) Data

- Dataset sizes over years (old data):

<table>
<thead>
<tr>
<th>Data set name and year</th>
<th>Number of vertices</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>McDonnell Douglas MD–80 ’89</td>
<td>230,000</td>
<td>13</td>
</tr>
<tr>
<td>McDonnell Douglas F/A–18 ’91</td>
<td>900,000</td>
<td>32</td>
</tr>
<tr>
<td>Space shuttle launch vehicle ’90</td>
<td>1,000,000</td>
<td>34</td>
</tr>
<tr>
<td>Space shuttle launch vehicle ’93</td>
<td>6,000,000</td>
<td>216</td>
</tr>
<tr>
<td>Space shuttle launch vehicle ’96</td>
<td>30,000,000</td>
<td>1,080</td>
</tr>
<tr>
<td>Advanced subsonic transport ’98</td>
<td>60,000,000</td>
<td>2,160</td>
</tr>
<tr>
<td>Army UH–60 Blackhawk ’99</td>
<td>100,000,000</td>
<td>~4,000</td>
</tr>
</tbody>
</table>
Time-dependent (unsteady) Data

• Dataset sizes over time:

<table>
<thead>
<tr>
<th>Data set name and year</th>
<th># vertices</th>
<th># time steps</th>
<th>size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tapered Cylinder ’90</td>
<td>131,000</td>
<td>400</td>
<td>1,050</td>
</tr>
<tr>
<td>McDonnell Douglas F/A–18 ’92</td>
<td>1,200,000</td>
<td>400</td>
<td>12,800</td>
</tr>
<tr>
<td>Descending Delta Wing ’93</td>
<td>900,000</td>
<td>1,800</td>
<td>64,800</td>
</tr>
<tr>
<td>Bell–Boeing V–22 tiltrotor ’93</td>
<td>1,300,000</td>
<td>1,450</td>
<td>140,000</td>
</tr>
<tr>
<td>Bell–Boeing V–22 tiltrotor ’98</td>
<td>10,000,000</td>
<td>1,450</td>
<td>600,000</td>
</tr>
</tbody>
</table>
Experimental Flow Visualization

Typically, optical Methods.

Understanding this experimental methods will help us understand why certain visualization approaches are adopted.
With Smoke or Dye

- Injection of dye, smoke, particles
- Optical methods:
  - transparent object with complex distribution of light refraction index
- Streaks, shadows
Large Scale Dying

Source: weathergraphics.com
Direct Methods
Direct FlowVis with Arrows

Properties:

• direct FlowVis
• frequently used!
• normalized arrows vs. velocity coding
• 2D: quite useful, 3D: often problematic
• often difficult to understand in complex cases, mentally integrate to reconstruct the flow

Image source: tms.org
Issues of Arrows in 3D

Common problems:

• Ambiguity
• Perspective shortening
• 1D objects generally difficult to grasp in 3D

Remedy:

• 3D-Arrows (are of some help)
Arrows in 3D – Examples

Compromise:
arrows only in layers

Vector arrows
Geometric-based Methods: Integral curves and surfaces
Direct vs. Geometric FlowVis

**Direct flow visualization:**
- overview of current state of flow
- visualization with vectors popular
- arrows, icons, glyph techniques

**Geometric flow visualization:**
- use of intermediate objects, e.g., after vector field integration over time
- visualization of development over time
- streamlines, stream surfaces
- *analogous to indirect (vs. direct) volume visualization*
Streamlines – Theory

• flow data $\mathbf{v}$: derivative information
  • $\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x})$;
    spatial points $\mathbf{x} \in \mathbb{R}^n$, Time $t \in \mathbb{R}$, flow
    vectors $\mathbf{v} \in \mathbb{R}^n$

• streamline $s$: integration over time, also
called trajectory, solution, curve
  • $s(t) = s_0 + \int_{0\leq u \leq t} \mathbf{v}(s(u)) \, du$;
    seed point $s_0$, integration variable $u$

• Property:
  • uniqueness

• difficulty: result $s$ also in the integral $\Rightarrow$
analytical solution usually impossible.
Streamlines – Practice

Basic approach:

• Mathematical expression: \( s(t) = s_0 + \int_{0 \leq u \leq t} v(s(u)) \, du \)

• practice: **numerical integration**

• **idea:**
  (very) locally, the solution is (approx.) linear

• Euler integration:
  follow the current flow vector \( v(s_i) \) from the current streamline point \( s_i \)
  for a very small time \( (dt) \) and therefore distance

Euler integration: \( s_{i+1} = s_i + v(s_i) \cdot dt \),
  integration of small steps \( (dt \text{ very small}) \)
Euler Integration – Example

2D model data:

\[ v_x = \frac{dx}{dt} = -y \]
\[ v_y = \frac{dy}{dt} = \frac{x}{2} \]

Sample arrows:

True solution: ellipses.
Euler Integration – Example

- Seed point \( \mathbf{s}_0 = (0|-1)^T \);
- current flow vector \( \mathbf{v}(\mathbf{s}_0) = (1|0)^T \);
- \( dt = \frac{1}{2} \)

\( v_x = \frac{dx}{dt} = -y \)

\( v_y = \frac{dy}{dt} = x/2 \)
Euler Integration – Example

New point \( s_1 = s_0 + v(s_0) \cdot dt = \left(\frac{1}{2} - 1\right)^T \); current flow vector \( v(s_1) = (1, 1/4)^T \);

\( v_x = \frac{dx}{dt} = -y \)

\( v_y = \frac{dy}{dt} = x/2 \)
Euler Integration – Example

- New point $s_2 = s_1 + v(s_1) \cdot dt = (1|-7/8)^T$;
  current flow vector $v(s_2) = (7/8|1/2)^T$;

$v_x = dx/dt = -y$

$v_y = dy/dt = x/2$
Euler Integration – Example

\[ \mathbf{s}_3 = (\frac{23}{16}|-\frac{5}{8})^T \approx (1.44|-0.63)^T; \]
\[ \mathbf{v}(\mathbf{s}_3) = (\frac{5}{8}|\frac{23}{32})^T \approx (0.63|0.72)^T; \]

\[ v_x = \frac{dx}{dt} = -y \]
\[ v_y = \frac{dy}{dt} = \frac{x}{2} \]
Euler Integration – Example

\[ \mathbf{s}_4 = \begin{pmatrix} \frac{7}{4} \\ -\frac{17}{64} \end{pmatrix}^T \approx \begin{pmatrix} 1.75 \\ -0.27 \end{pmatrix}^T; \]

\[ \mathbf{v}(s_4) = \begin{pmatrix} \frac{17}{64} \\ \frac{7}{8} \end{pmatrix}^T \approx \begin{pmatrix} 0.27 \\ 0.88 \end{pmatrix}^T; \]
Euler Integration – Example

$s_9 \approx (0.20|1.69)^T,$

$v(s_9) \approx (-1.69|0.10)^T;$
\begin{align*}
\mathbf{v}(s_{14}) & \approx (-3.22 | -0.10)^T; \\
\mathbf{s}_{14} & \approx (0.10 | -1.61)^T;
\end{align*}
Euler Integration – Example

\[ s_{19} \approx (0.75, -3.02)^T; \quad v(s_{19}) \approx (3.02, 0.37)^T; \]
clearly: large integration error, \( dt \) too large, 19 steps
Euler Integration – Example

- \( dt \) smaller (1/4): more steps, more exact.
  \( s_{36} \approx (0.04|-1.74)^T; \ v(s_{36}) \approx (1.74|0.02)^T; \)

- 36 steps
Comparison Euler, Step Sizes

Euler quality is proportional to $dt$
# Euler Example – Error Table

<table>
<thead>
<tr>
<th>$dt$</th>
<th>#steps</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>19</td>
<td>~200%</td>
</tr>
<tr>
<td>1/4</td>
<td>36</td>
<td>~75%</td>
</tr>
<tr>
<td>1/10</td>
<td>89</td>
<td>~25%</td>
</tr>
<tr>
<td>1/100</td>
<td>889</td>
<td>~2%</td>
</tr>
<tr>
<td>1/1000</td>
<td>8889</td>
<td>~0.2%</td>
</tr>
</tbody>
</table>
RK-2 – A Quick Round

RK-2: even with $dt = 1$ (9 steps) better than Euler with $dt = 1/8$ (72 steps)
RK-4 vs. Euler, RK-2

Even better: fourth order RK:

- four vectors \( a, b, c, d \)
- one step is a convex combination:
  \[
  s_{i+1} = s_i + \left( a + 2\cdot b + 2\cdot c + d \right)/6
  \]
- vectors:
  \[
  a = dt \cdot v(s_i) \quad \text{... original vector}
  
  b = dt \cdot v(s_i + a/2) \quad \text{... RK-2 vector}
  
  c = dt \cdot v(s_i + b/2) \quad \text{... use RK-2 ...}
  
  d = dt \cdot v(s_i + c) \quad \text{... and again}
  \]
Euler vs. Runge-Kutta

RK-4: pays off only with complex flows

Here approx. like RK-2
Summary:

• analytic determination of streamlines usually not possible
• hence: numerical integration
• various methods available (Euler, Runge-Kutta, etc.)
• Euler: simple, imprecise, esp. with small $dt$
• RK: more accurate in higher orders
• furthermore: adaptive methods, implicit methods, etc.
Streamline Placement

in 2D
Problem: Choice of Seed Points

Streamline placement:

• If regular grid used: very irregular result
Overview of Algorithm

Idea: streamlines should not lie too close to one another

Approach:
- choose a seed point with distance $d_{sep}$ from an already existing streamline
- forward- and backward-integration until distance $d_{test}$ is reached (or ...).
- two parameters:
  - $d_{sep}$ ... start distance
  - $d_{test}$ ... minimum distance
Algorithm – Pseudo-Code

• Compute initial streamline, put it into a queue
• current streamline = initial streamline
• WHILE not finished DO:
  
  TRY: get new seed point which is $d_{sep}$ away from current streamline
  
  IF successful THEN
    compute new streamline AND put to queue

  ELSE IF no more streamline in queue THEN
    exit loop

  ELSE next streamline in queue becomes current streamline
Streamline Termination

When to stop streamline integration:

• when distance to neighboring streamline ≤ $d_{\text{test}}$
• when streamline leaves flow domain
• when streamline runs into fixed point ($\mathbf{v} = 0$)
• when streamline gets too near to itself (loop)
• after a certain amount of maximal steps
New Streamlines
Different Streamline Densities

Variations of $d_{sep}$ relative to image width:

6%  3%  1.5%
$d_{sep}$ vs. $d_{test}$

$d_{test} = 0.9 \cdot d_{sep}$

$d_{test} = 0.5 \cdot d_{sep}$
Tapering and Glyphs

Thickness in relation to distance

Directional glyphs:
Literature

For more information, please see:


• Data Visualization: Principles and Practice, Chapter 6: Vector Visualization by A. Telea, AK Peters 2008
Acknowledgment

Thanks for the materials

• Prof. Robert S. Laramee, Swansea University, UK