Visual Data Analytics - Formulas

Adrian Castro

February 2021

1 General Knowledge

Visualization Pipeline:
- Data Acquisition
- Filtering/Enhancement (obtain useful data/cleaning)
- Visualization/Mapping (how to represent data)
- Rendering (generate 2D images/video)

Indipendent Variables: 2D/3D space, time
Dependent Variables: temperature, velocity
Characteristics of data values:
- Attribute types
- Domain
- Value range
- Dimension
- Error & Uncertainty
- Physical Interpretration

Qualitative data: categorical, non-measurable, discrete
Nominal data: no natural ordering, membership
Ordinal data: natural ordering
Categorical data: values from fixed number or categories
Scalar data: if we can map from higher dimensional data to a lower dimensional data ($f(x) : \mathbb{R}^n \to \mathbb{R}$)
Tensor data: multi-dimensional data

2 Interpolation

ISOContours: all points lying on the same line with the same values
Goal of interpolation: construct continuous function f which approximates given values

2.1 Radial Basis Functions

Each point (p_i, f_i) influences $f(x)$ based on distance:

$$r = \|p_i - x\|$$

$$f(x) = \sum_{i=1}^{N} f_i \cdot \varphi(\|p_i - x\|)$$

$$\varphi(r) = e^{-r^2}$$

Weighted radial basis functions
$f(p_j)$ interpolates the value f_j

$$f(p_j) = \sum_{i=1}^{N} w_i \cdot \varphi(\|p_i - p_j\|) = f_j$$

Yields a system of linear equations to be solved for w_i:

$$A = \begin{pmatrix} \varphi(\|p_1 - p_1\|) & \cdots & \varphi(\|p_N - p_1\|) \\ \vdots & \ddots & \vdots \\ \varphi(\|p_1 - p_N\|) & \cdots & \varphi(\|p_N - p_N\|) \end{pmatrix}$$

$$W = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \quad F = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

$$W = A^{-1} \cdot F$$

Drawbacks: global influence of every sample, adding a new point requires solving the equation system, computationally expensive
Inverse Distance Weighting
Assumption: Nearby points are more similar than those further away.

$$f(x) = \sum_{i=1}^{N} f_i \varphi(\|p_i - x\|)$$

$$d_i = \|p_i - x\| \qquad \varphi(r) = \frac{\frac{1}{r^2}}{\sum_{i=1}^{N} \frac{1}{d_i^2}}$$

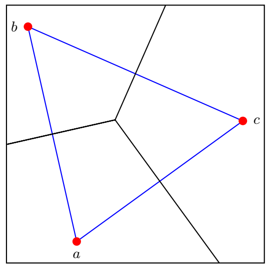
Adjusted formula:

$$\sum_{i=1}^{N} \frac{f_i}{\|p_i - x\|^2} \Big/ \sum_{i=1}^{N} \frac{1}{\|p_i - x\|^2}$$

Drawbacks: still costly, still global influence of every sample

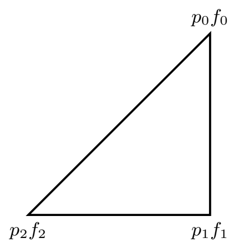
2.2 Triangulation

Giving up smooth, precise reconstruction in favor of speed
Generating a Triangulation
Goals: avoid long and thin triangles, maximize minimum angle in the triangulation, maximize $\frac{\text{radius of circle}}{\text{radius of circumcircle}}$
Delaunay Triangulation
1. Circumcircle does not contain another point of the set
2. Maximizes minimum angle of triangulation
3. Triangulation is unique for all but trivial cases
Voronoi Diagram



Every **Voronoi Sample** (a, b, c) is a vertex of a **Delaunay** triangulation

2.3 Interpolation Inside a Triangle



Find a function f that interpolates f_i at the point p_i such that:

$$f(p_i) = f_i \qquad i = 0, \cdots, N$$

Linear function:

$$f(x) = a + bx + cy$$

Where a, b, c can be obtained with:

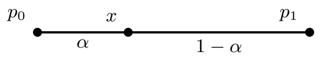
$$X = \begin{pmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{pmatrix} \quad A = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad F = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}$$

$$A = X^{-1} \cdot F$$

2.4 Baricentric Interpolation

We want to have a smooth, continuous interpolation $\forall \alpha \in [0,1]$

$$f(\alpha) = \alpha \cdot p_0 + (1-\alpha) \cdot p_0$$



$$\alpha_0 = A_0/A,\ \alpha_1 = A_1/A,\ \alpha_2 = A_2/A$$

Where $A =$ area of the triangle

$$x = \alpha_0 p_0 + \alpha_1 p_1 + \alpha_2 p_2$$

$$\alpha_0 + \alpha_1 + \alpha_2 = 1$$

If α_i are known, then $f(x)$ can be interpolated from values f_i at the vertices via:

$$f(x) = \alpha_0 f_0 + \alpha_1 f_1 + (1 - \alpha_0 - \alpha 1) \cdot f_2$$

Given a point Q we can find α by solving the linear system:

$$P = \begin{pmatrix} p_{0x} & p_{1x} & p_{2x} \\ p_{0y} & p_{1y} & p_{2y} \\ 1 & 1 & 1 \end{pmatrix}$$

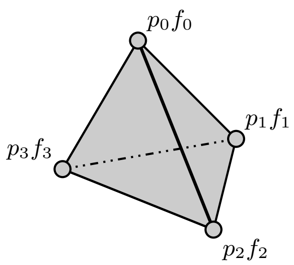
$$A = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix}\ X = \begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix}$$

$$A = P^{-1} \cdot X$$

2.5 Scalar Interpolation

Formula of tetrahedron:

$$f(x) = a + bx + cy + dz$$



Can get values a, b, c, d by solving linear system:

$$X = \begin{pmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{pmatrix}$$

$$A = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}\ F = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

$$A = X^{-1} \cdot F$$

Computing Gradient of Scalar Field

The gradient in a scalar field $f(x)$:

$$\nabla f(x) = \begin{pmatrix} \frac{\delta f(x)}{\delta x} \\ \frac{\delta f(x)}{\delta y} \\ \frac{\delta f(x)}{\delta z} \end{pmatrix}$$

In the case of a tetrahedron with function the gradient is always constant:

$$f(x) = a + bx + cy + dz$$

$$\nabla f(x) = \begin{pmatrix} \frac{\delta f(x)}{\delta x} \\ \frac{\delta f(x)}{\delta y} \\ \frac{\delta f(x)}{\delta z} \end{pmatrix} = \begin{bmatrix} b \\ c \\ d \end{bmatrix}$$

2.6 Piece-Wise Linear Interpolation

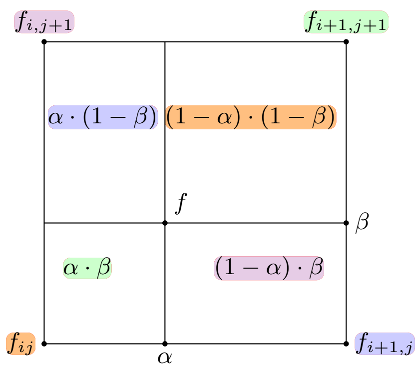
For data points $(x_0, y_0), \cdots, (x_N, y_N)$ Evaluate

$$f(x) = (1-\alpha) y_i + \alpha y_i$$

$$\text{where: } \alpha = \frac{x - x_1}{x_{i+1} - x_i}$$

Bilinear Interpolation

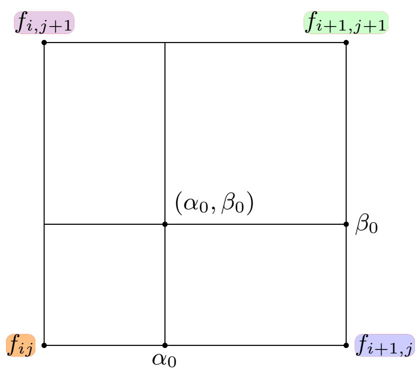
$$\begin{aligned} f(\alpha, \beta) = & f_{ij} \cdot (1-\alpha)(1-\beta) + \\ & f_{i+i,j} \cdot \alpha(1-\beta) + \\ & f_{i,j+1} \cdot (1-\alpha)\beta + \\ & f_{i+1,j+1} \cdot \alpha\beta \end{aligned}$$



Asymptotic Decider
Hyperbola form:

$$f(\alpha, \beta) = \gamma(\alpha - \alpha_0)(\beta - \beta_0) + \delta$$

where δ is the value at point (α_0, β_0)



Transform to **hyperbola form**:

$$f(\alpha, \beta) = A\alpha + B\beta + C\alpha\beta + D$$

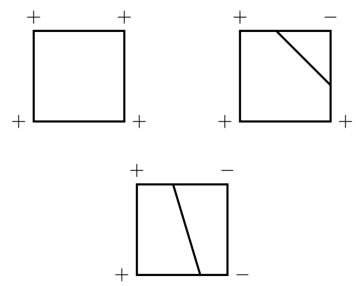
$$\begin{aligned} A &= f_{i+1,j} - f_{ij} \\ B &= f_{i,j+1} - f_{ij} \\ C &= f_{ij} - f_{i,j+1} - f_{i+1,j} + f_{i+1,j+1} \\ D &= f_{ij} \\ \delta &= (f_{ij} \cdot f_{i+1,j+1} - f_{i+1,j} \cdot f_{i,j+1}) \\ \alpha_0 &= -B/C \\ \beta_0 &= -A/C \\ \gamma &= C \end{aligned}$$

2.7 Marching Squares

Algorithm used to compute isolines on a 2D surface, given a grid of values.
Step 1: mark all the data with:

value >= isovalue: $+$
value < isovalue: $-$

Step 2: 16 different combinations, but we are interested in 4 base cases:

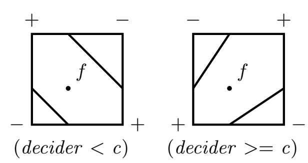


In the last case, we have ambiguity, which can be resolved in two ways:
• midpoint decider:

$$f_{center} = \frac{1}{4}(f_{ij} + f_{i+1,j} + f_{i,j+1} + f_{i+1,j+1})$$

• if midpoint decider is right in the middle, hence the ambiguity is still there, we can use the **asymptotic decider** previously defined as:

$$f(\alpha, \beta) = \gamma(\alpha - \alpha_0)(\beta - \beta_0) + \delta$$



3 Phong Illumination Model

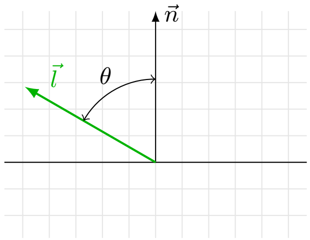
Components:
• **Ambient Light**: background light, constant everywhere
• **Diffuse Reflector**: reflects equally into all directions
• **Specular Reflector**: reflects mostly into the mirror direction

3.1 Lighting

$$C = K_a C_a O_d$$

K_a : ambient reflection coefficient $\in [0,1]$
C_a : color of the ambient light
O_a : object color

3.2 Diffuse Reflection



$$C = K_d C_p O_d \cdot \cos\theta$$

$$\cos\theta = \frac{\vec{n}\cdot\vec{l}}{|\vec{n}|\cdot|\vec{l}|}$$

K_d : diffuse reflection coefficient $\in [0,1]$
C_p : color of point of light
O_d : object color
$\cos\theta$: angle between light vector $\vec{l}$ and $\vec{n}$

Reminder:

$$|\vec{v}| = \sqrt{(v_0)^2 + \cdots + (v_n)^2}$$

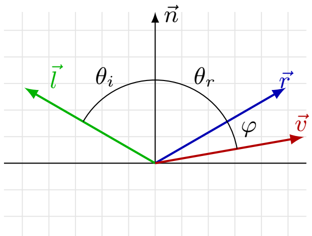
3.3 Specular Reflection

$$C = K_s C_p O_d \cdot \cos^n\varphi$$

$$\cos\varphi = \frac{\vec{r}\cdot\vec{v}}{|\vec{r}|\cdot|\vec{v}|}$$

$$\vec{r} = 2\cdot\left(\frac{\vec{n}\cdot\vec{l}}{|\vec{n}|\cdot|\vec{l}|}\right)\cdot\frac{\vec{n}}{|\vec{n}|} - \frac{\vec{l}}{|\vec{l}|}$$

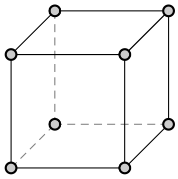
K_s : specular reflection coefficient $\in [0,1]$
C_p : color of point of light
O_d : object color
$\cos^n\varphi$: angle between reflected light ray $\vec{r}$ and the vector to the viewer $\vec{v}$
$(\cdots)^n$: shininess factor (extent of highlight)



Perfect Mirror: shininess factor n has to be very large, going to infinity.

4 Volume Visualization

- **Indirect** Volume Rendering (Marching Cubes, data is reduced to intermediate representation, which can then be rendered)
- **Direct** Volume Rendering (Ray Casting, data is considered semi-transparent with physical properties, and directly rendered)
- **Voxel**: point sample in 3D
- **Transfer Function**: maps data values to color & opacity



4.1 Transfer Function

Associates distinct materials (value range) to distinct properties (color and opacity). Maps a different color to each scalar value.

$$T : \texttt{scalar value} \rightarrow \mathit{color} + \alpha$$

4.2 Direct Volume Rendering

Each **voxel** emits light of the color assigned to it, and absorbs light according to its opacity.

4.3 Light Emission and Attenuation

Volume rendering integral:

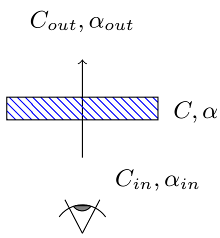
$$C(s) = \int_{s_0}^{s} C(s')\cdot e^{-\int_{s'}^{s}\alpha(t)\,dt}\,ds'$$

We are calculating the accumulated color until $C(s')$, with the absorption coefficient between the last color s' and the point of view s.

4.4 Ray Casting - Compositing

Front-to-back Strategy

$$C_{in} = (0,0,0), \alpha_{in} = 0$$
$$C_{out} = C_{in} + (a - \alpha_{in})\cdot\alpha\cdot C$$
$$\alpha_{out} = \alpha_{in} + (a - \alpha_{in})\cdot\alpha$$



4.5 Direct Volume Rendering: Phong Shading

We have to evaluate Phong's illumination model based on:
- **position** of current sample and light source
- sample's color emission assigned by transfer function
- sample's **normal/gradient** (central difference)

4.6 Gradient Estimation

We can estimate the gradient using finite differencing. However, this technique is not feasible with large amounts of data.

$$\nabla f(x,y,z) \approx$$
$$\approx \frac{1}{2h}\begin{pmatrix} f(x+h,y,z) - f(x-h,y,z) \\ f(x,y+h,z) - f(x,y-h,z) \\ f(x,y,z+h) - f(x,y,z-h) \end{pmatrix}$$

4.7 Compositing Schemes

- **Surface Rendering/First-Hit**: stop ray traversal if an ISOSurface is hit
- **Average**: simply accumulate colors, but ignore opacity
- **Maximum**: take maximum color along axis and display it

5 Flow Visualization

- Motion of fluids (gases, liquids)
- Geometric boundaries
- Conservation of mass, energy and momentum
- Velocity/Flow field $v(x,t)$

5.1 Vector Field Visualization

- Vector data representing **Direction** and **Magnitude**
- **Steady** (time-indipendent) flow:

$$v(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

- **Unsteady** (time dependent) flow:

$$v(x,t) : \mathbb{R}^n \times \mathbb{R}^1 \rightarrow \mathbb{R}^n$$

5.2 Vector and Scalar Functions

Scalar function:

$$\rho(x,t)$$

The gradient points to the direction of maximum change of $\rho(x,t)$:

$$\nabla\rho(x,t) = \begin{pmatrix} \frac{\delta\rho(x,t)}{\delta x} \\ \frac{\delta\rho(x,t)}{\delta y} \\ \frac{\delta\rho(x,t)}{\delta z} \end{pmatrix}$$

Jacobian matrix applied to vector field $v(x,t)$:

$$J = \nabla v(x,t) = \begin{pmatrix} \frac{\delta v_x}{\delta x} & \frac{\delta v_x}{\delta y} & \frac{\delta v_x}{\delta z} \\ \frac{\delta v_y}{\delta x} & \frac{\delta v_y}{\delta y} & \frac{\delta v_y}{\delta z} \\ \frac{\delta v_z}{\delta x} & \frac{\delta v_z}{\delta y} & \frac{\delta v_z}{\delta z} \end{pmatrix}$$

Divergence: Diagonal of Jacobian, describes flow into/out of a region:

$$div\ v(x,t) = \frac{\delta v_x}{\delta x} + \frac{\delta v_y}{\delta y} + \frac{\delta v_z}{\delta z}$$

$div\ v(x_0) > 0 : v$ has **source** in x_0

$div\ v(x_0) < 0 : v$ has **sink** in x_0

$div\ v(x_0) = 0 : v$ is **source-free** in x_0

Curl/vorticity: tells how fast the flow is rotating, and the axis around which it is rotating.

$$curl\ v(x,t) = \nabla v(x,t) = \begin{pmatrix} \frac{\delta v_z}{\delta y} - \frac{\delta v_y}{\delta z} \\ \frac{\delta v_x}{\delta z} - \frac{\delta v_z}{\delta x} \\ \frac{\delta v_y}{\delta x} - \frac{\delta v_x}{\delta y} \end{pmatrix}$$

5.3 Computing Characteristic Lines

Characteristic lines are **tangential** to the flow.

$$\frac{\delta x(t)}{\delta t} = v(x(t),t)$$

- do not intersect
- they are the solutions to the initial value problem

5.4 Euler Method

- First order method
- higher accuracy with smaller step size (Δt)

$$x(t+\Delta t) \approx x(t) + \Delta t \cdot v(x(t),t)$$

5.5 Midpoint Method

- Better, faster than Euler method

$$\Delta x = \Delta t \cdot v(x,t)$$
$$v_{mid} = v(x + \frac{\Delta x}{2}, t + \frac{\Delta t}{2})$$
$$x(t+\Delta t) = x(t) + \Delta t \cdot v_{mid}$$

5.6 Runge-Kutta 4th Order

- The most accurate

$$k_1 = \Delta t \cdot v(x,t)$$
$$k_2 = \Delta t \cdot v(x + \frac{k_1}{2}, t + \frac{\Delta t}{2})$$
$$k_3 = \Delta t \cdot v(x + \frac{k_2}{2}, t + \frac{\Delta t}{2})$$
$$k_4 = \Delta t \cdot v(x + k_3, t + \frac{\Delta t}{2})$$
$$x(t+\Delta t) = k + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$$

5.7 Vector Field Topology

- We want ot draw along the most important lines
- Draw topological skeleton of flow

5.8 Critical Points

- Singularities in vector fields such that:

$$v(x^*) = 0$$

- Points where magnitude of veectors goes to zero and direction of vectors is undefined

To find critical points: points where:

$$v(x^*) = 0$$

Classification First, we build the Jacobian matrix for $v(x^*)$

$$J = \begin{pmatrix} \frac{\delta v_x}{\delta x} & \frac{\delta v_x}{\delta y} \\ \frac{\delta v_y}{\delta x} & \frac{\delta v_y}{\delta y} \end{pmatrix}$$

Then, for each x^* calculate the **eigenvalues** λ_1, λ_2 of J.
To find the eigen values:

$$det(J-\lambda I) = det\begin{pmatrix} \frac{\delta v_x}{\delta x} - \lambda & \frac{\delta v_x}{\delta y} \\ \frac{\delta v_y}{\delta x} & \frac{\delta v_y}{\delta y} - \lambda \end{pmatrix}$$

Circulating Source

$Im(\lambda_1, \lambda_2) \neq 0$ and $Re(\lambda_1, \lambda_2) > 0$

Non-Circulating Source

$Im(\lambda_1, \lambda_2) = 0$ and $Re(\lambda_1, \lambda_2) > 0$

Center

$Im(\lambda_1, \lambda_2) \neq 0$ and $Re(\lambda_1, \lambda_2) = 0$

Circulating Sink

$Im(\lambda_1, \lambda_2) \neq 0$ and $Re(\lambda_1, \lambda_2) < 0$

Non-Circulating Sink

$Im(\lambda_1, \lambda_2) = 0$ and $Re(\lambda_1, \lambda_2) < 0$

Saddle Point

$Im(\lambda_1, \lambda_2) = 0$ and $\lambda_1 \cdot \lambda_2 < 0$

5.9 Dense Texture-Based Flow Visualization

- Better information coverage
- Critical point detection and classification

5.10 Line-Integral Convolution

- Global visualization technique
- Starts with random noise (white noise)
- *Smear out* the texture along trajectories of vector field
- Results in **low correlation** between **neighbouring lines** and **high correlation** along **flow direction**

Steps:
- Vector Field → (integration) Stream Line
- Stream Line → (convolution) Noise Texture
- Noise Texture → (result) Output Image

5.11 Filtering By Convolution

- Sliding a function $g(x)$ along a function $f(x)$:

$$s(x') = [f * g](x') = \int_{-\infty}^{\infty} f(x)g(x'-x)\,dx$$

- Function f is averaged with a weight function g
- $(x'-x)$ centers g around x'
- We use the single stream lines that pass through a single pixel to smear out the noise
- $\Phi_0(u)$ is the stream line containing the point (x_0, y_0)
- $T(x,y)$ is the randomly generated noise texture
- Compute the intensity at (x_0, y_0) as:

$$I(x_0, y_0) = \int_{-L}^{L} k(u) \cdot T(\Phi(u))\,du$$