Chapter 4 – Data Transformation and Reconstruction

4.1 Motivation

A multidimensional data set L_m^n consist of

- *m independent variables* representing the data domain (usually space, time).

- *n dependent variables* defined over the domain (e.g. scalar, vector and/or tensor data).

Motivation (cont'd)

Two Problems:

- Data can exist in various forms (analytic functions, sampled, implicitly) but many visualization methods require a particular representation

  ⇒Transform independent variables (e.g. interpolation, sampling)

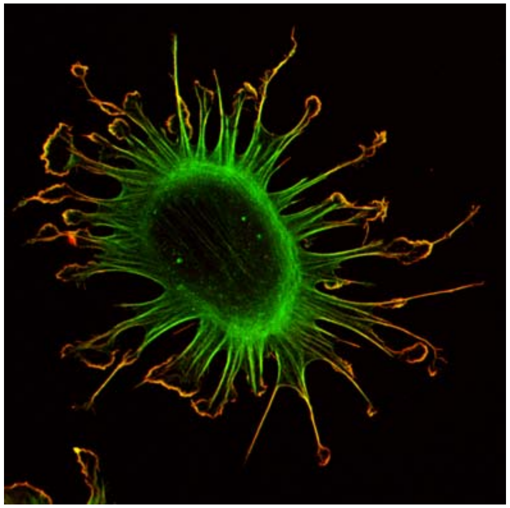
- Data attributes contain not enough information or information is to complex

  ⇒Transform the dependent variable (data reduction, data enrichment, data modification)

4.2 Sources of Volume Data

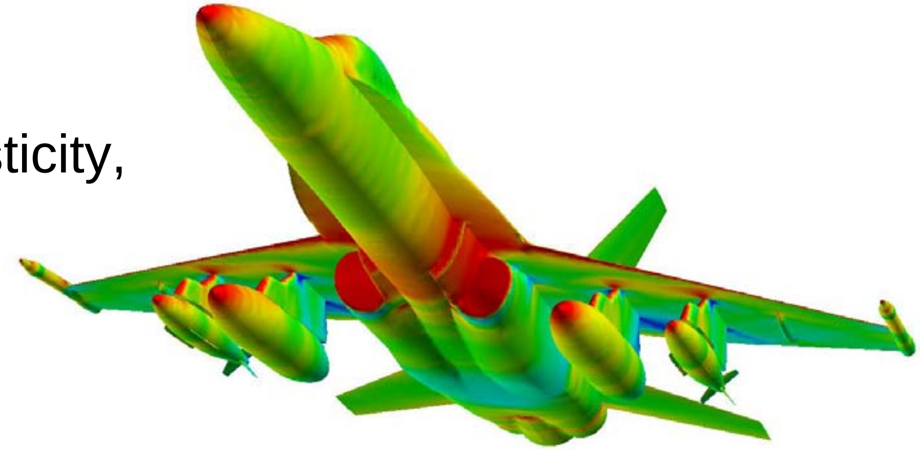
We are predominantly interested in (time varying) volume data (*m*=3 or 4). Widespread in Science and Engineering, but here are just a few common examples:

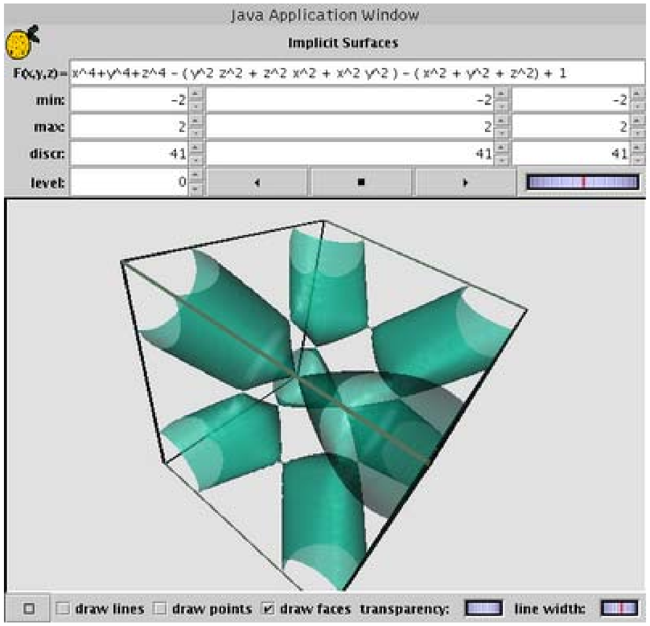
- Medicine and Biology ("biomedical imaging")
  - CT (“Computed Tomography”) Scans
  - MRI (“Magnetic Resonance Imaging”) Scans
  - PET (“Positron Emission Tomography”) Scans
  - Ultrasonography (ultrasonic echo-sounding)
  - Confocal microscopy



Sources of Volume Data (cont'd)

- Physics/Engineering
  - Measurements of density, pressure, elasticity, etc. over some volume
  - Computer simulations
    - Computational fluid dynamics (CFD)
    - Stress analysis (FEM – Finite Element Modelling)
    - Numerical models (e.g. of Earth's magnetic field)
- Graphics
  - Implicit surfaces as a modelling tool
    - Convolutional smoothing of polyhedra
    - Model by sculpting volume data

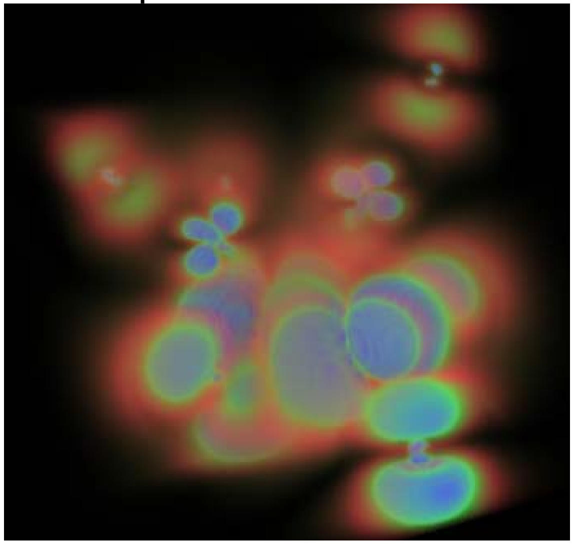




Some Volume Visualization Examples

Microscience:

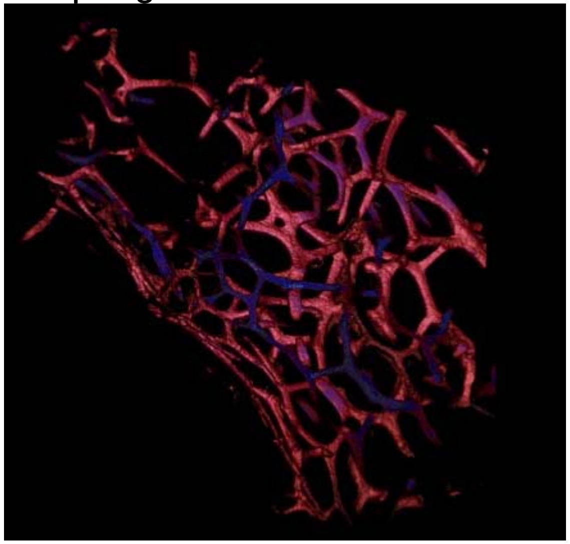
Molecular structure of an iron protein



Data source: Kitware Inc.

Biology:

Cellular structure of a sea sponge



Data source: BIRU – University of Auckland

Medicine:

MRI Data of the human brain

Data source: University of Erlangen-Nürnberg

Terminology

- Start with some function *f(x, y, z)* or *f(x, y, z,t)*
  - Called a *field* over R^3 (or R^4)
  - We deal only in 3-D and 4-D fields, but can have arbitrary *n*-D fields
- Have various types of fields
  - **Scalar**, i.e. *f* of type *real*
    - e.g. CT scan, density measurements, …
  - **Vector**, i.e. *f* is an *n*-vector (commonly *n* = 3)
    - e.g. fluid velocity in a CFD simulation
    - force fields (magnetic, electric, gravitational)
  - **Tensor**, i.e. *f* is a matrix
    - e.g. stress and strain in FEM modelling

4.3 Transformation of the Independent Variable

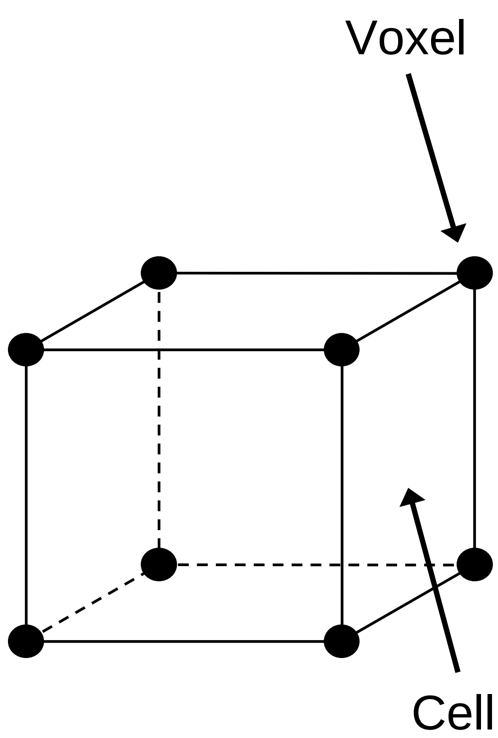
- Sampling
  - Continuous data → discrete (sampled) data
  - Usually use regular sampling:

$$f_{ijk} = f(x_i, y_j, z_k) \quad \text{where} \quad \begin{cases} x_\mathrm{i} = x_0 + i\Delta x \\ y_\mathrm{j} = y_0 + j\Delta y \\ z_\mathrm{k} = z_0 + k\Delta z \end{cases}$$

- Interpolation
  - Discrete (sampled) data → continuous data
  - Explained in the following slides
- Resampling
  - Discrete (sampled) data → discrete (sampled) data
  - Interpolation followed by sampling

Sampled Volume Data

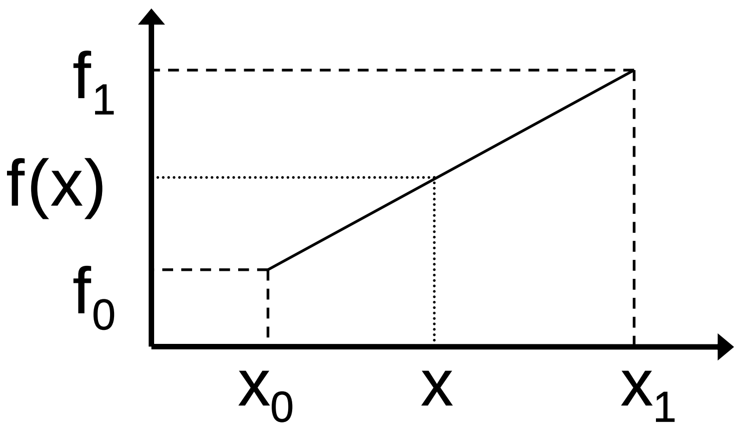
- *Sampled volume data* is defined only at a particular set of (x,y,z)
  - Most commonly on a cartesian grid
  - Sample values are called *voxels*
  - A cuboidal region with voxels at all 8 vertices is called a *cell*
    - DON'T CONFUSE THESE TWO TERMS!!



Volumes as Fields

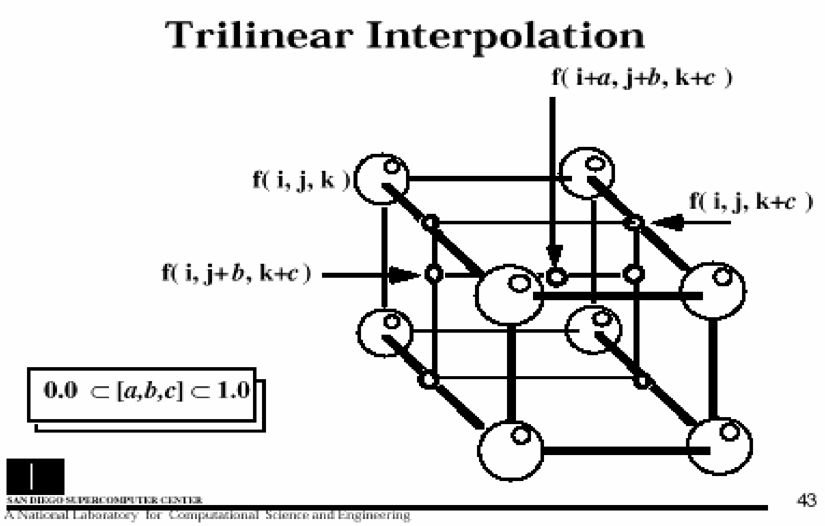
- Important to remember that samples represent a field, i.e. a continuum
- Process of defining the underlying field from a set of samples is called *interpolation* or *reconstruction*
  - Formally defined by convolution with a reconstruction filter (see next section)
  - *Trilinear interpolation* is the most common reconstruction method
  - *Tricubic interpolation* (e.g. B-spline) used for high-quality work

Trilinear interpolation



- Linear interpolation

$$f(x) = f(x_0) + \frac{x - x_0}{x_1 - x_0}\left(f(x_1) - f(x_0)\right)$$
$$= (1-t)\, f_0 + t\, f_1$$
$$\text{where } t = \frac{x - x_0}{x_1 - x_0},\ f_0 = f(x_0),\ f_1 = f(x_1)$$

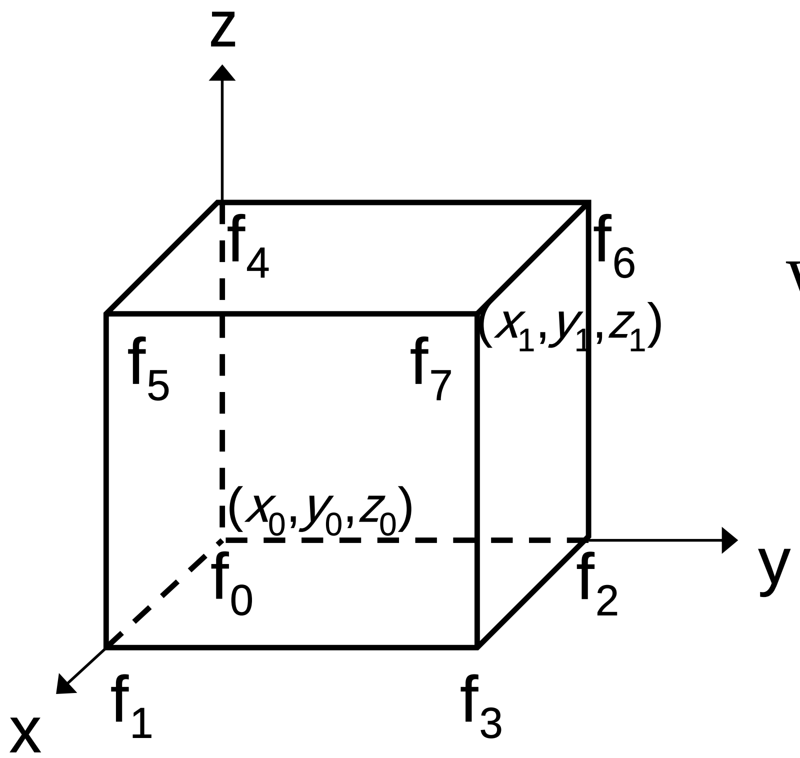


- Trilinear interpolation can be regarded as 7 linear interpolations

Trilinear interpolation (cont'd)

$$f(x,y,z) = (1-r)(1-s)(1-t)\,f_0 + r\,(1-s)(1-t)\,f_1$$
$$+ (1-r)s(1-t)\,f_2 + r\,s\,(1-t)\,f_3$$
$$+ (1-r)(1-s)t\;f_4 + r\,(1\text{-}s)\,t\;f_5$$
$$+ (1-r)s\,t\;f_6 + r\,s\,t\;f_7$$

$$\text{where } r = \frac{x-x_0}{x_1-x_0},\; s = \frac{y-y_0}{y_1-y_0},\; t = \frac{z-z_0}{z_1-z_0}$$



Gradients

- Many algorithms require the *gradient* of *f*
- The gradient is a vector perpendicular to the isocontours of the field
  - i.e. is in the direction of steepest ascent
  - magnitude is rate of change of value in that direction
- Defined as

$$\text{gradient}(f) = \text{grad } f = \nabla f = \begin{pmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \\ \dfrac{\partial f}{\partial z} \end{pmatrix}$$

Gradients (cont'd)

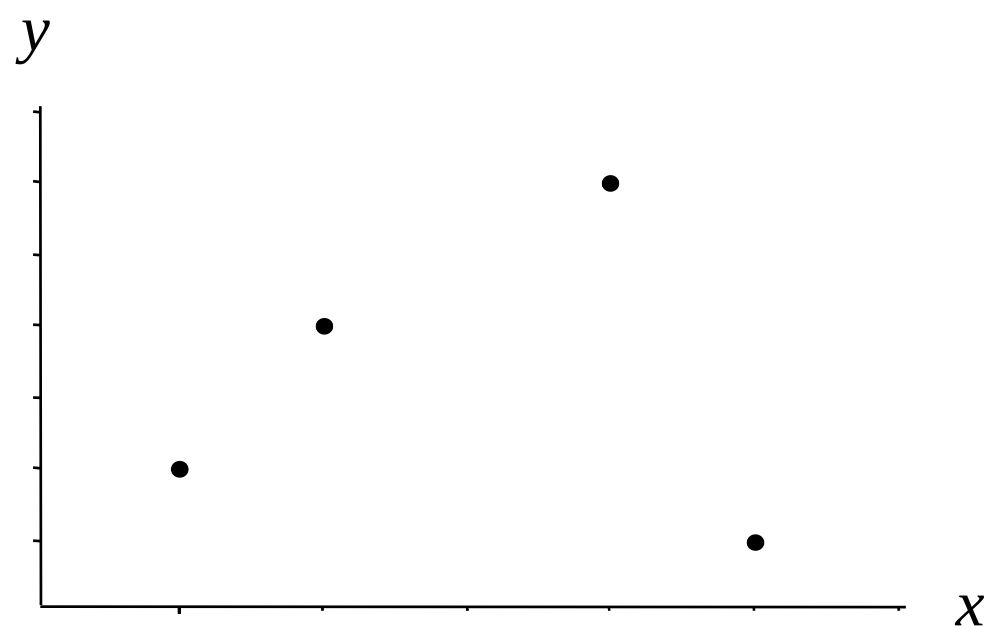
- A trilinearly reconstructed field has discontinuous gradients at cell boundaries, so rather than computing true gradients we normally compute a smooth approximation:
  - Use central differences to compute gradients at voxels

$$\nabla f(x,y,z) = \begin{pmatrix} \dfrac{f(x+\Delta x, y, z) - f(x-\Delta x, y, z)}{2\Delta x} \\ \dfrac{f(x, y+\Delta y, z) - f(x, y-\Delta y, z)}{2\Delta y} \\ \dfrac{f(x, y, z+\Delta z) - f(x, y, z-\Delta z)}{2\Delta z} \end{pmatrix}$$

  - Trilinearly interpolate those to get **grad** *f* (*x*, *y*, *z*)

4.4 Reconstruction Filters

Consider a sequence of uniformly-spaced samples, y_0, y_1, ... How do we interpolate to get a smooth function?

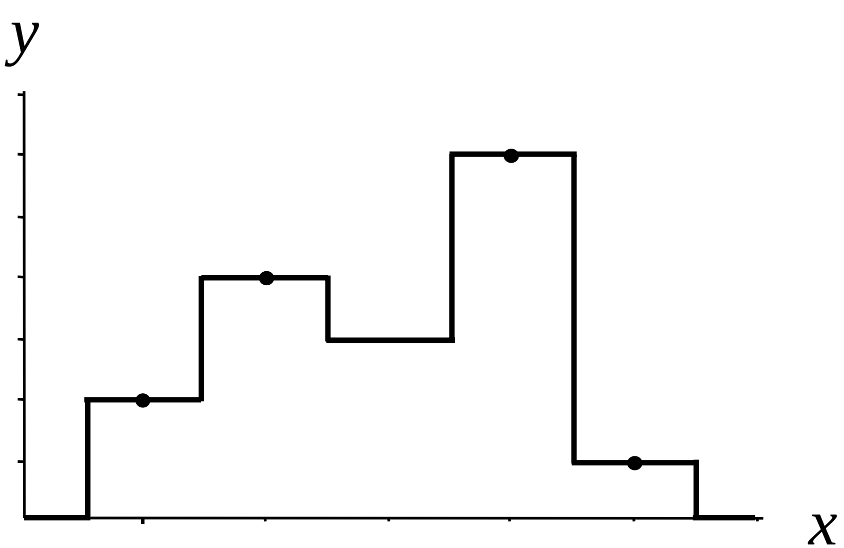


Piecewise Constant Interpolation

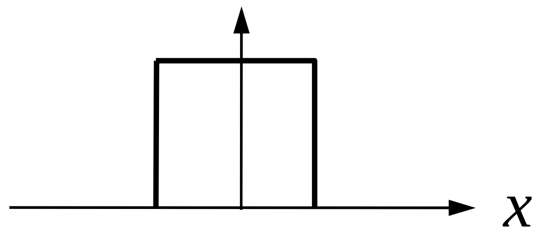
("Nearest Neighbour" interpolation/ box filtering)

$$y(x) = \sum y_i \; U(x-i)$$

$$\text{where } U(x) = \begin{cases} 1 & -0.5 \leq x \leq 0.5 \\ 0 & \text{otherwise} \end{cases}$$



[The unit "square pulse" function]



Convolutional Smoothing

- Piecewise constant is not smooth enough
- Common smoothing technique is “convolutional smoothing”
  - Smoothed value at any point is the average of the input function in the vicinity of the point
  - Unweighted average over a fixed interval is called “running mean”
  - Generally have a weight function or *filter* function, $h(x)$
  - Box filtering is convolutional smoothing with square pulse, $h = U$

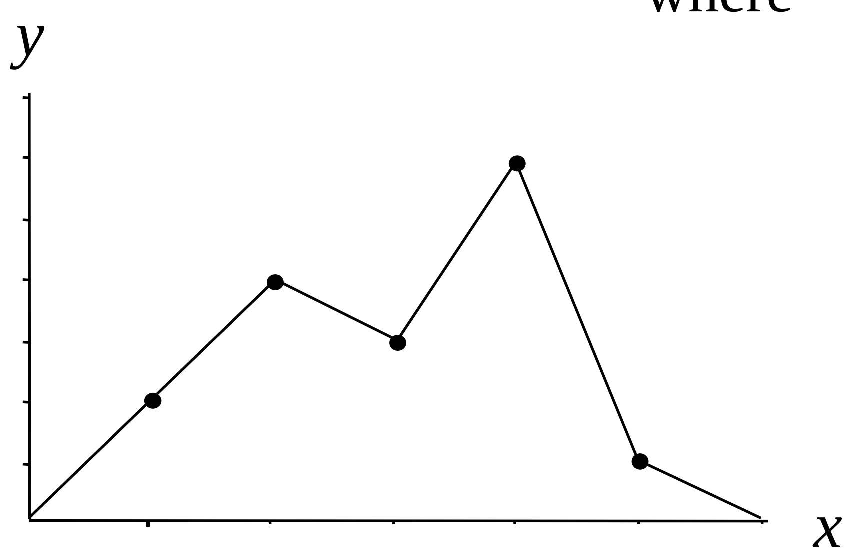
$$f_{smooth}(x) = f * h = \int_{-\infty}^{\infty} f(u)h(x-u)\,du$$

Piecewise Linear Interpolation

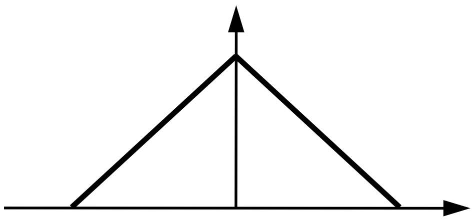
Obtained by "box filtering" nearest-neighbour plot.

$$y(x) = \sum y_i \, L(x-i)$$

$$\text{where} \quad L(x) = U(x) * U(x) = \begin{cases} 1+x & -1 \le x < 0 \\ 1-x & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

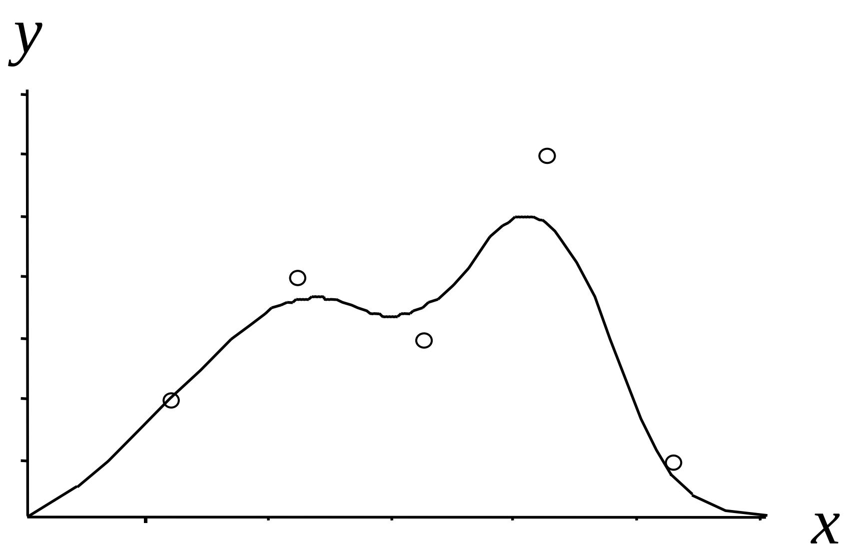


[The "tent" function = linear b-spline]

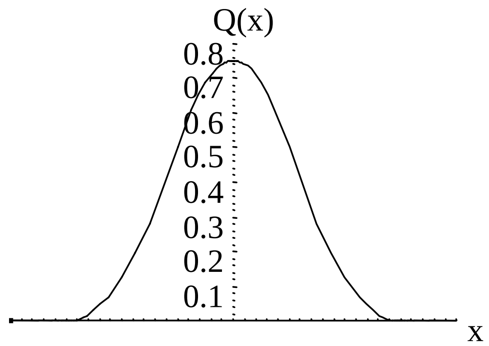


Piecewise Quadratic Interpolation

$$\text{where}\quad Q(x) = L(x) * U(x) = \begin{cases} \dfrac{(2x+3)^2}{8} & -\dfrac{3}{2} \le x < -\dfrac{1}{2} \\ \dfrac{3}{4} - x^2 & -\dfrac{1}{2} \le x < \dfrac{1}{2} \\ \dfrac{(2x-3)^2}{8} & \dfrac{1}{2} \le x < \dfrac{3}{2} \\ 0 & \text{otherwise} \end{cases}$$



[The “Quadratic B-Spline” function]



Volume Reconstruction

- Reconstruction filters can be extended to 3D and be used to interpolate (reconstruct) sample volumes.

$$f_{smooth}(\mathbf{x}) = f * h = \int_{-\infty}^{\infty} f(\mathbf{u})h(\mathbf{x}-\mathbf{u})d\mathbf{u}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\mathbf{u})h(\mathbf{x}-\mathbf{u})du_x du_y du_z$$

$$where \quad \mathbf{x} = (x, y, z)$$

$$\mathbf{u} = (u_x, u_y, u_z)$$

$$f(\mathbf{u}) = f_{ijk} \text{ iff } (i, j, k) \text{ is the sample point closest to } \mathbf{u}$$

Volume Reconstruction (cont'd)

- Separable filters can be written $h(x,y,z) = h_s(x)h_s(y)h_s(z)$
- Examples are:

Trilinear filter:

$$h_s(x) = \begin{cases} 1-|x| & \text{if } |x|<1 \\ 0 & \text{otherwise} \end{cases}$$

Tricubic filters: Tricubic B-Spline (B=1,C=0)

Catmull-Rom Spline (B=0, C=0.5)

$$h_s(x) = \frac{1}{6}\begin{cases} (12-9B-6C)|x|^3 + (-18+12B+6C)|x|^2 + (6-2B) & \text{if } |x|<1 \\ (-B-6C)|x|^3 + (6B+30C)|x|^2 + (-12B-48C)|x| + (8B+24C) & \text{if } 1 \le |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

Volume Reconstruction (cont'd)

(Truncated) Gaussian filters:

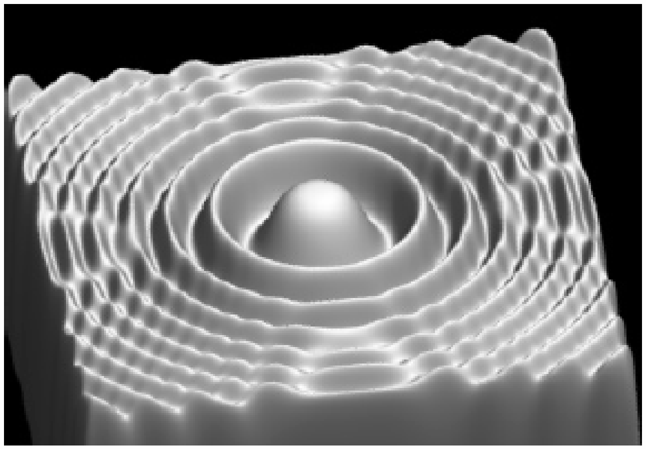
$$h_s(x) = \begin{cases} e^{-x^2/2\sigma^2} & \text{if } |x| < x_m \\ 0 & \text{otherwise} \end{cases}$$

Windowed sinc filters:

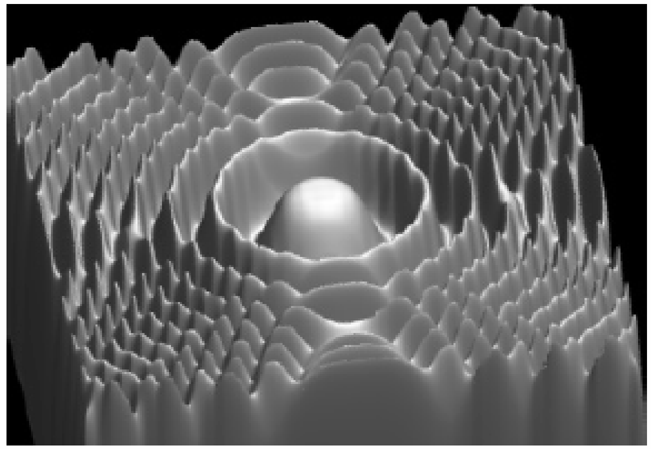
$$h_s(x) = \begin{cases} \left(1+\cos(\pi x / x_m)\right)\text{sinc}(4x / x_m) & \text{if } |x| < x_m \\ 0 & \text{otherwise} \end{cases}$$

Volume Reconstruction (cont'd)

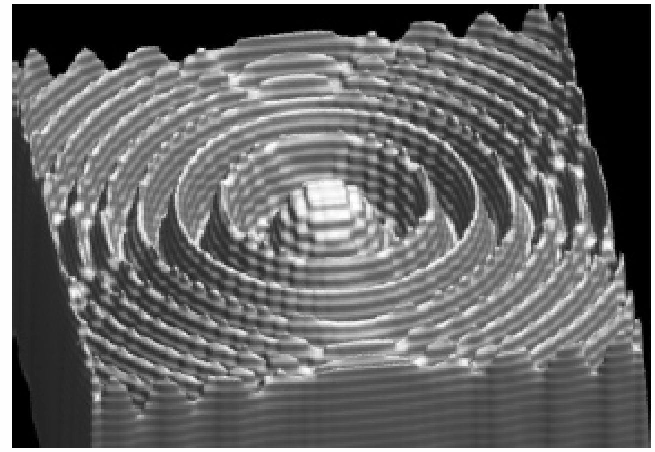
© 1994, Stephen R. Marschner, Richard J. Lobb, *An Evaluation of Reconstruction Filters for Volume Rendering*, Proceedings of IEEE Visualization '94, pp. 100-107.



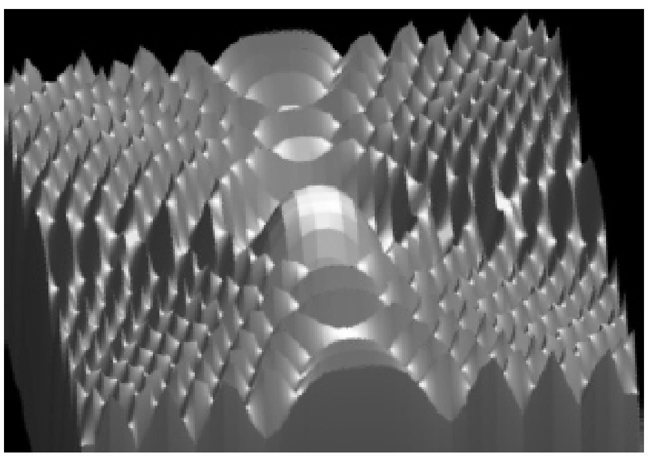
(a) B-spline



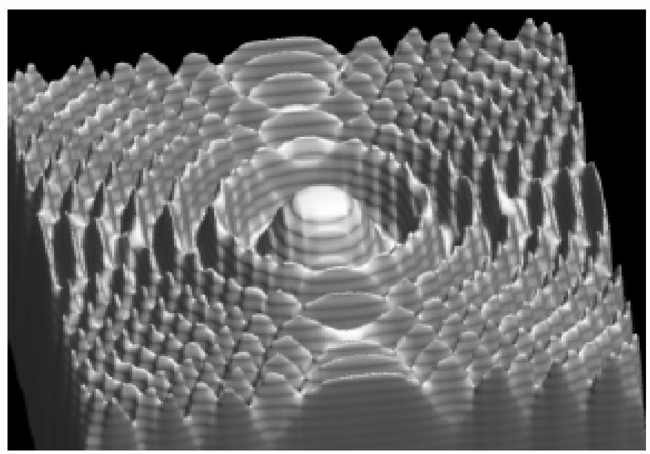
(b) Catmull-Rom



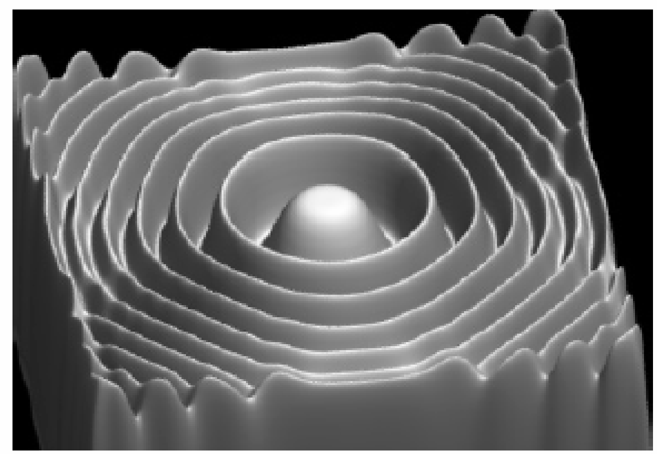
(c) Cubic ($B = 0.5$, $C = 0.85$)



(d) Trilinear



(e) Cubic ($B = 0.26$, $C = 0.1$)



(f) Windowed sinc ($r = 4.8$)

4.5 Transformations of the Dependent Variable

- We are predominantly interested in
  - Scalars
  - Vectors
  - Symmetric (2^{nd} order) tensors

- Possible Transformations of data are
  - Data reduction
  - Data modification
  - Data expansion

Transformations for Scalar Data

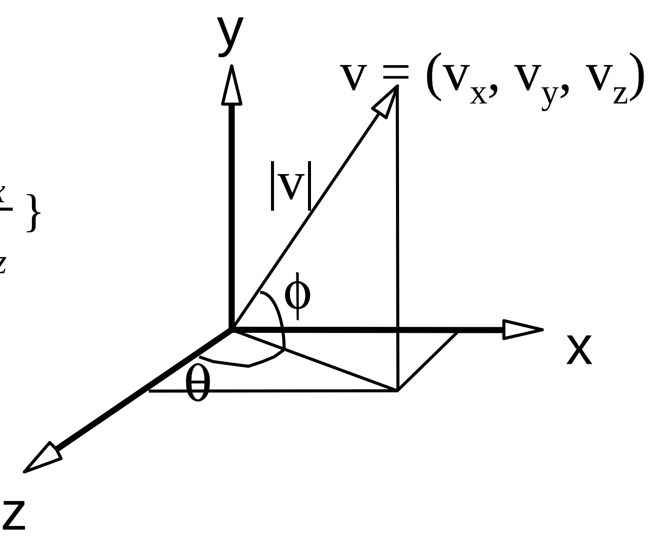
- Compute gradient
- Image Processing Algorithms
  - smoothing, sharpening, edge detection, …
- Statistical techniques for multivariate data

Transformations for Vector Data

- Vector magnitude $$|\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$
- Vector direction

$$\phi = \tan^{-1} \frac{u_y}{\sqrt{u_x^2 + u_z^2}}$$

$$\theta = atan2(u_x, u_z) \text{ \{i.e. a 4-quadrant } \tan^{-1} \frac{u_x}{u_z} \text{\}}$$

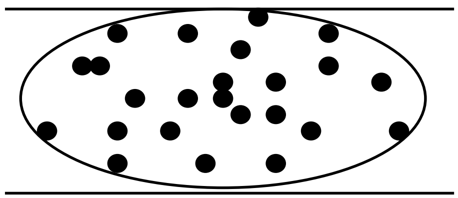


- Jacobian

$$\mathbf{J}_{\mathbf{v}} = \begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{pmatrix}$$

Transformations for Tensor Data

- We only deal with *n*-D symmetric 2nd-order tensors which are represented by an $n \times n$ matrix.
- Example: Diffusion Tensor
  - Water molecules move randomly due to Brownian motion (diffusion)
  - In inhomogeneous materials diffusion speed is different in each direction
  - Water molecules originating at fixed location form ellipsoidal shape
  - Shape described by a tensor



Eigenvalues and Eigenvectors

Any *n*-dimensional symmetric tensor T always has *n* eigenvalues λ_i and *n* mutually perpendicular eigenvectors $\mathbf{v}_i$ such that

$$\mathbf{T}\mathbf{v}_i = \lambda_i \mathbf{v}_i \qquad i = 1, \ldots, n$$

The eigenvectors and eigenvalues of the diffusion tensor give the direction and length of the principal axes of the diffusion ellipsoid

Coordinate transformations

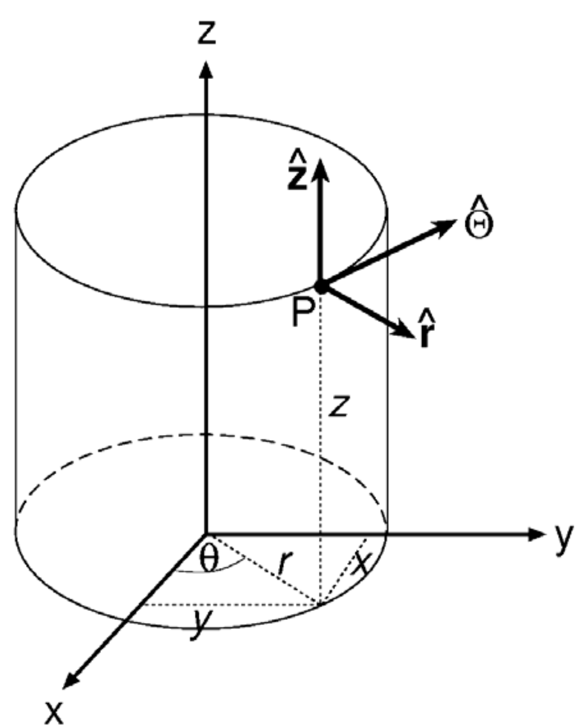
- In many cases it’s convenient to define a new coordinate systems (material coordinate systems) **q**(*x*,*y*,*z*) which better represents the shape of the modelled object.
- Example: Cylindrical coordinates for modelling a tube

Transform world coordinates to material coordinates

$$\begin{pmatrix} r \\ \theta \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2} \\ \cos^{-1}\dfrac{x}{\sqrt{x^2+y^2}} \\ z \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2} \\ \sin^{-1}\dfrac{y}{\sqrt{x^2+y^2}} \\ z \end{pmatrix}$$

Transform material coordinates to world coordinates

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\cos\theta \\ r\sin\theta \\ z \end{pmatrix}$$



Coordinate transformations (cont'd)

Let $\mathbf{J} = \begin{pmatrix} \frac{\partial q_1}{\partial x} & \frac{\partial q_1}{\partial y} & \frac{\partial q_1}{\partial z} \\ \frac{\partial q_2}{\partial x} & \frac{\partial q_2}{\partial y} & \frac{\partial q_2}{\partial z} \\ \frac{\partial q_3}{\partial x} & \frac{\partial q_3}{\partial y} & \frac{\partial q_3}{\partial z} \end{pmatrix}$ be the Jacobian of the coordinate transformation.

Then the representations of a vector $\mathbf{v}$ in world coordinates and a vector $\hat{\mathbf{v}}$ in material coordinates are converted into each other by

$$\mathbf{v} = \mathbf{J}\hat{\mathbf{v}} \quad \text{and} \quad \hat{\mathbf{v}} = \mathbf{J}^{-1}\mathbf{v}$$

Similarly the representations of a tensor are converted into each other by

$$\mathbf{T} = \mathbf{J}\hat{\mathbf{T}}\mathbf{J}^T \quad \text{and} \quad \hat{\mathbf{T}} = \mathbf{J}^{-1}\mathbf{T}(\mathbf{J}^{-1})^T$$

4.6 References

- T. Todd Elvins, *Introduction to Volume Visualization*, SIGGRAPH 94, Course Notes #10.
- Stephen R. Marschner, Richard J. Lobb, *An Evaluation of Reconstruction Filters for Volume Rendering*, Proceedings of IEEE Visualization ’94, pp. 100-107.
- Burkhard Wünsche, *Scientific Visualization*, chapter 4, In “A Toolkit for the Visualization of Tensor Fields in Biomedical Finite Element Models”, PhD Thesis, University of Auckland, 2004.
- Rosenblum et al. (editors), *Scientific Visualization – Advances and Challenges*, Academic Press, 1994.