VISUALIZATION

Scientific Visualization - I

Acknowledgement: ETH Zurich

Overview

- Basics
- Data
- Mapping Techniques
- Surface Rendering Shading
- Volume Visualization
- Vector Field Visualization

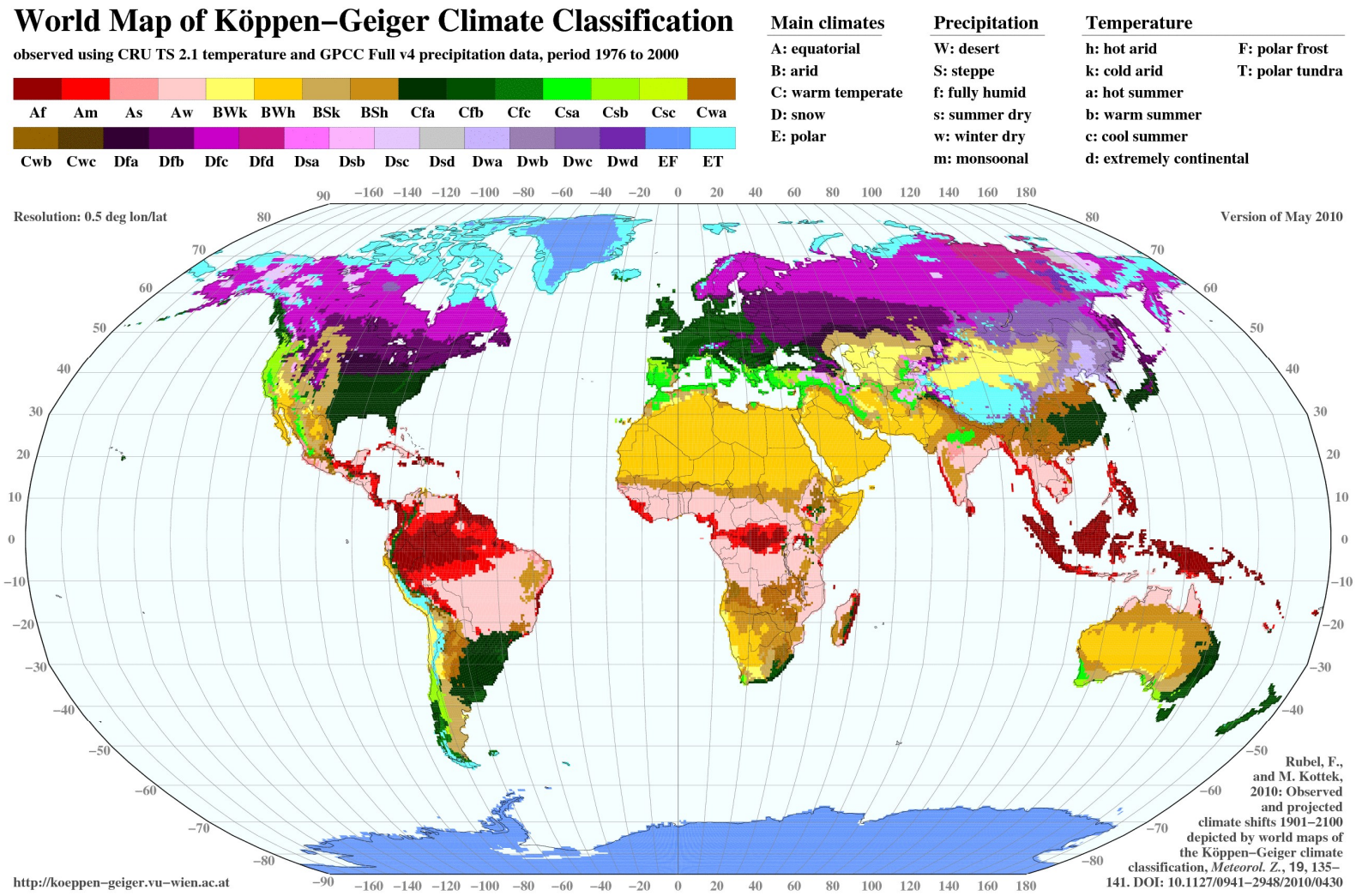
Overview

- Basics
- Data
- Mapping Techniques
- Surface Rendering Shading
- Volume Visualization
- Vector Field Visualization

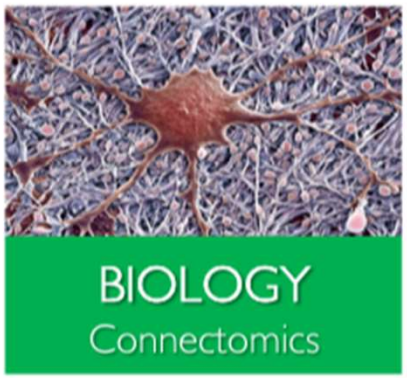
Scientific Visualization

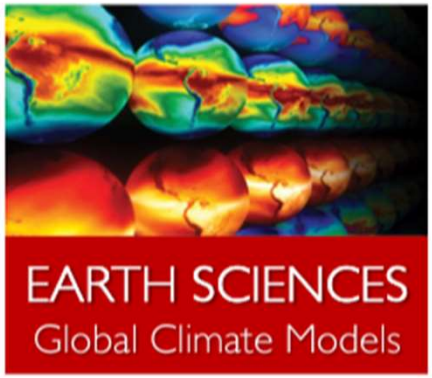
- Visualization of ‘scientific data’
- Datasets from simulation and physical measurements (quantitative)
  - Inherent spatial reference
- The capability of traditional presentation techniques is not sufficient for the increasing amount of data to be interpreted
- Data might come from any source with almost arbitrary size
- Techniques to efficiently visualize large-scale data sets and new data types need to be developed

Spatial Data



Applications

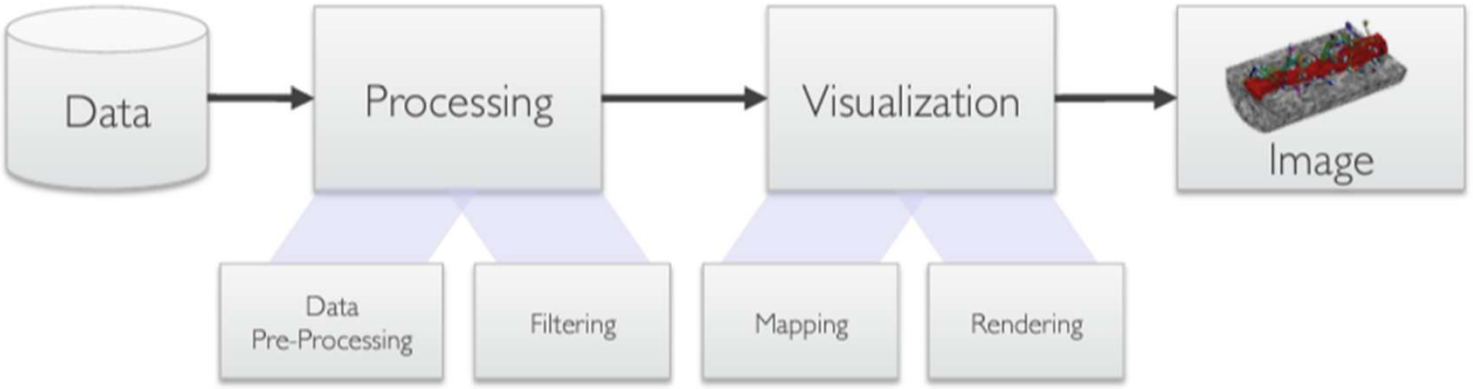


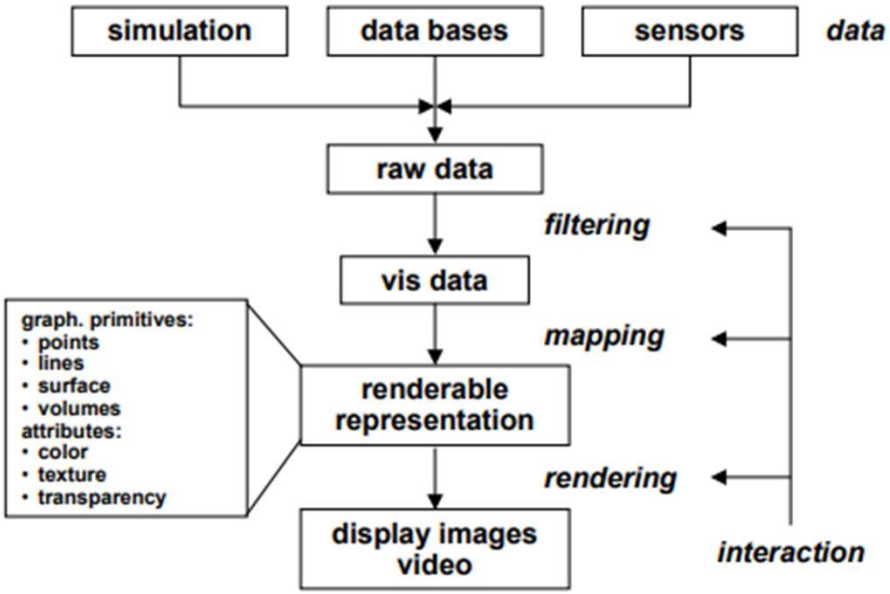


Challenges in Scientific Visualization

- Large data (storage, i/o, processing, rendering)
- Algorithms computationally expensive
  - Often require GPU processing for interactivity
- 3D
- Vectors
- Visualization methods depend heavily on dimensionality of domain
- Needs extensive background in Computer Graphics

Visualization Pipeline



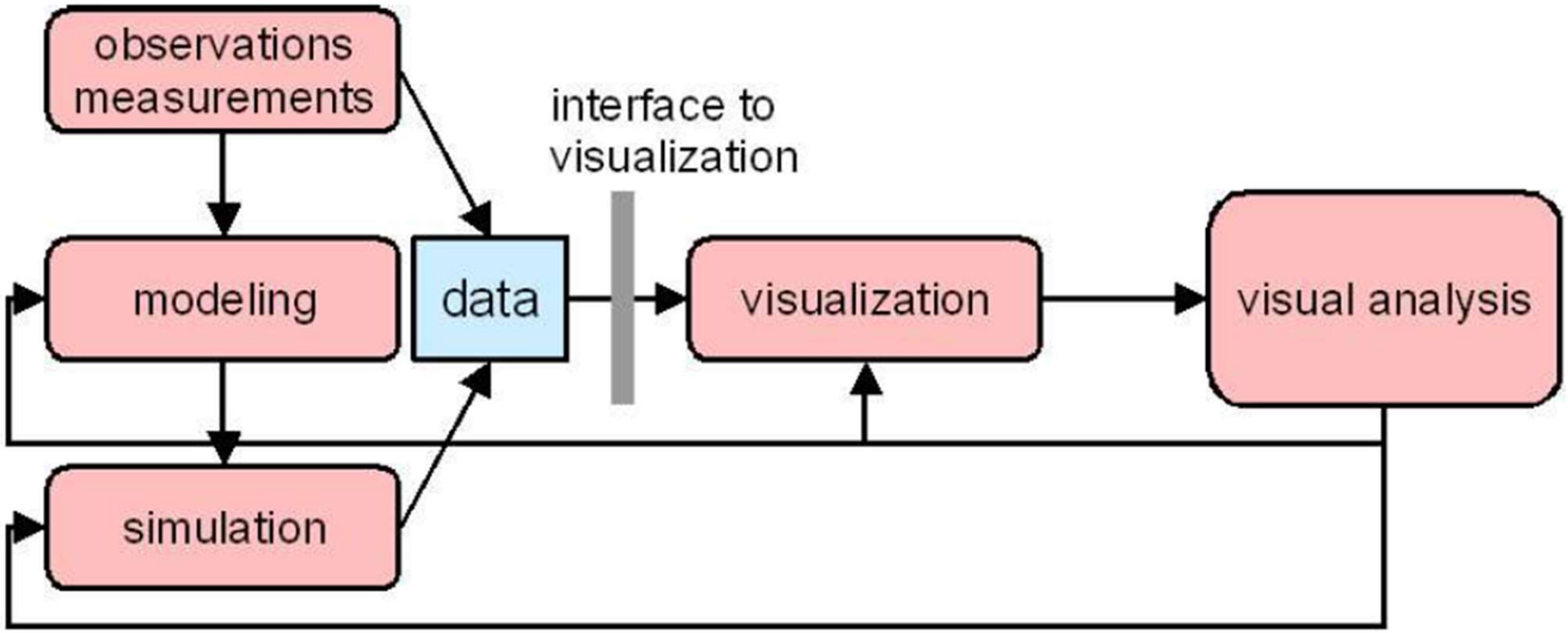


filtering
data -> data
- data format conversion
- clipping/cropping/denoising
- slicing
- resampling
- interpolation/approximation
- classification/segmentation

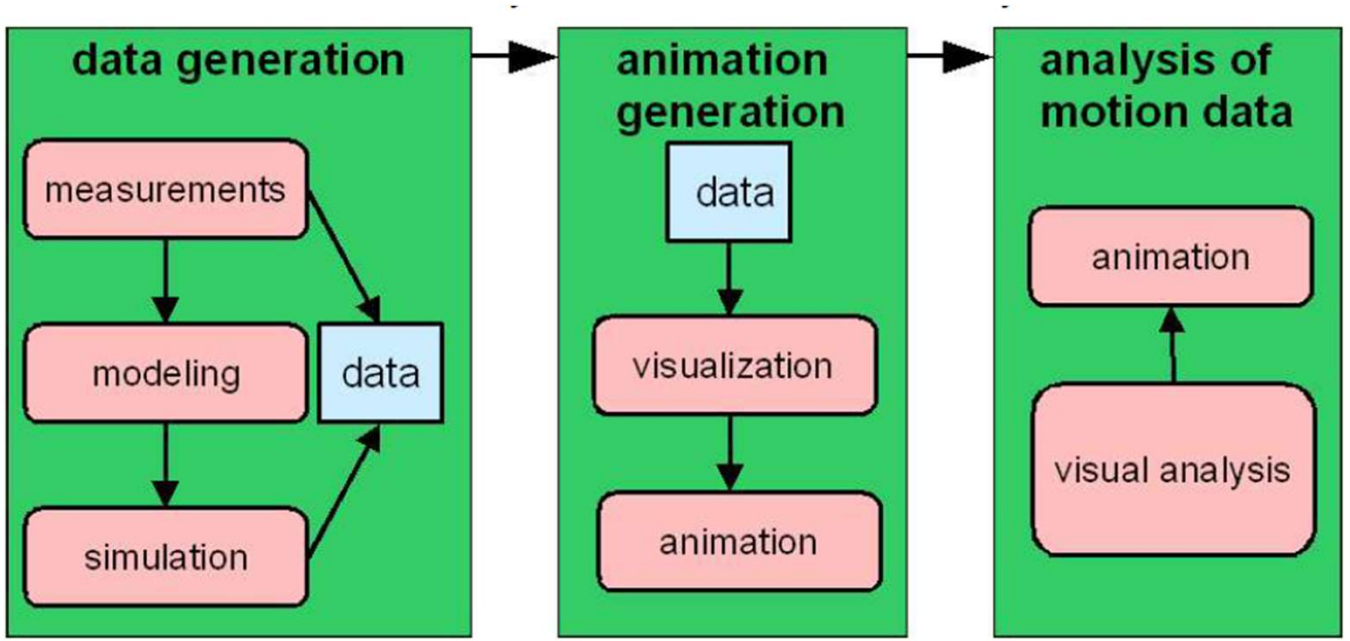
mapping
data ->graphical primitives
- scalar field ->isosurface
- 2D field ->height field
- vector field ->vectors
- tensor field ->glyphs
- 3D field -> volume

rendering:
- geometry/images/volumes
- "realism" – e.g. : shadows, lighting, shading

Visualization Analysis Cycle

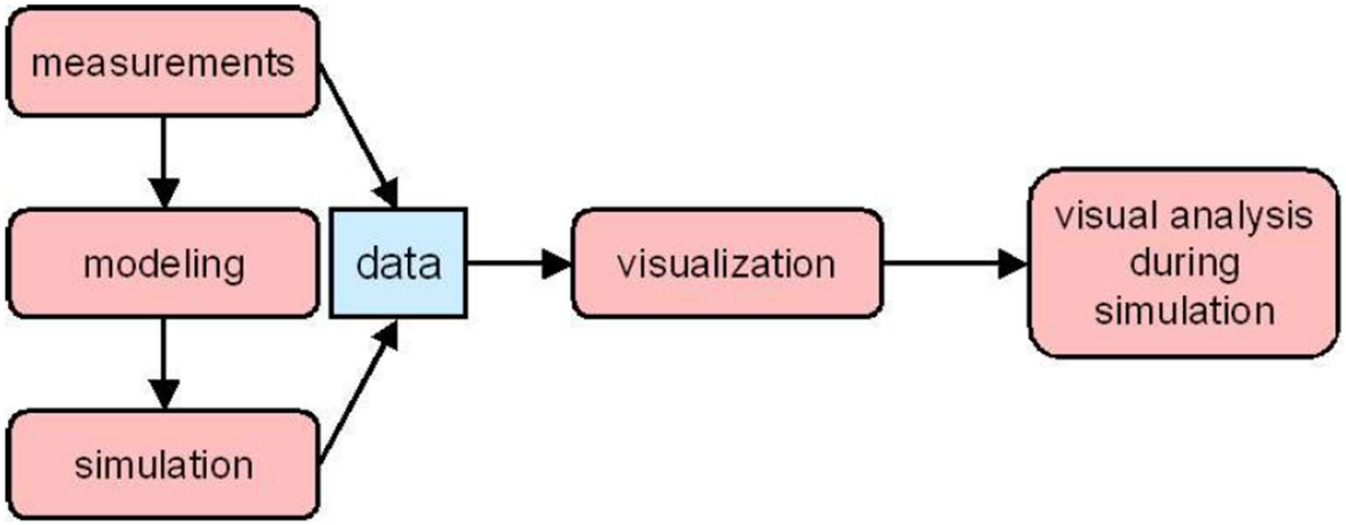


Visualization Scenario – Motion Mode



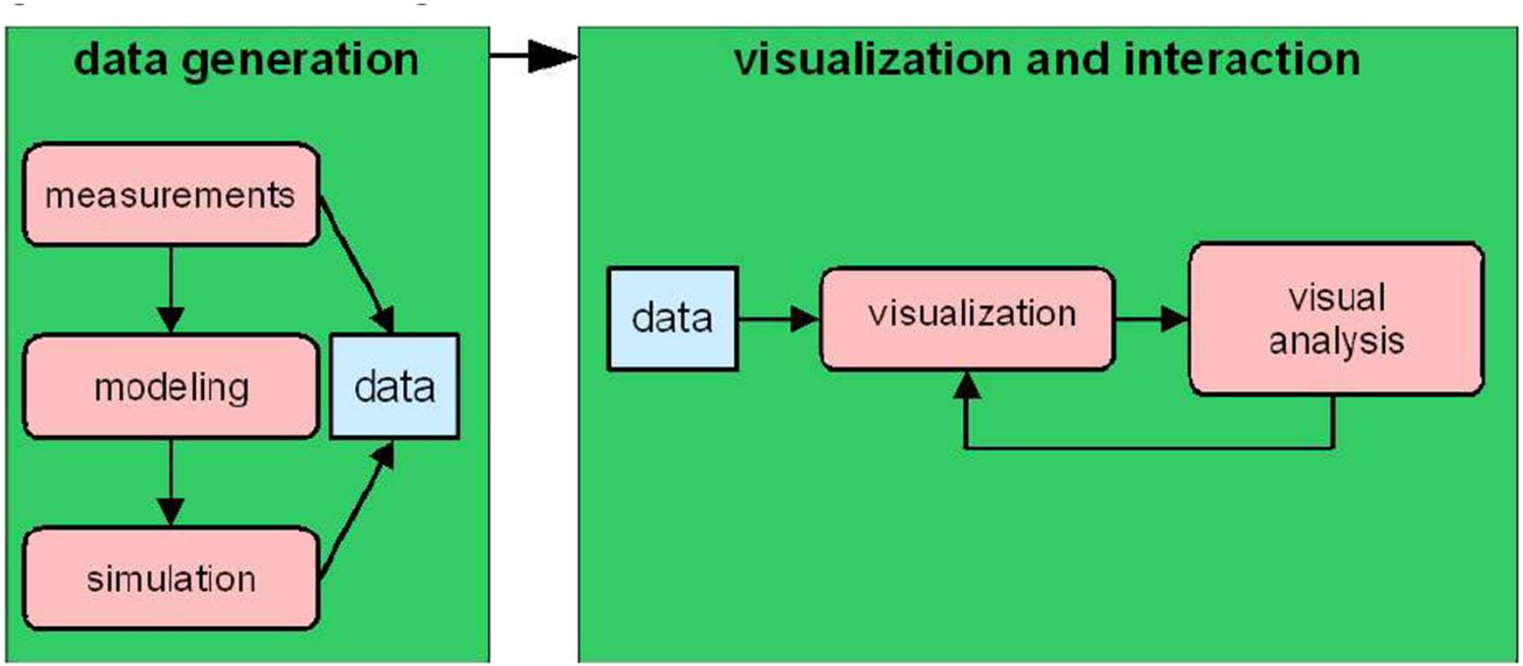
Scenario: video/movie mode – offline, no interaction

Visualization Scenario – Tracking Mode



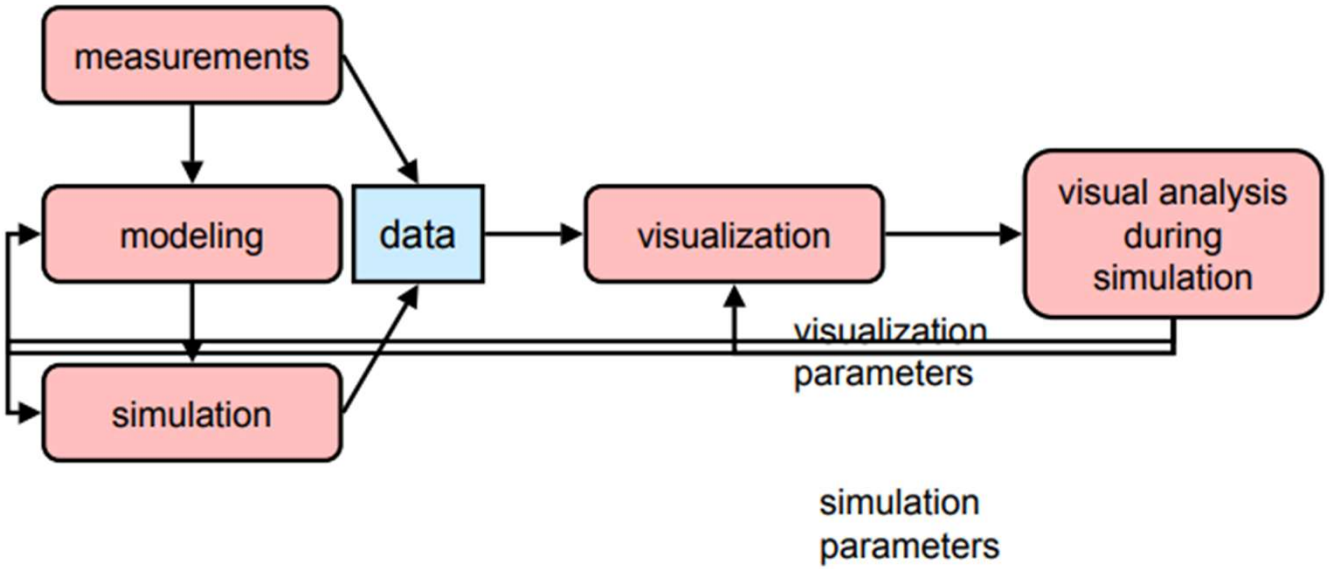
Scenario: tracking – online, no interaction

Visualization Scenario – Interactive Post Processing



Scenario: interactive post processing / visualization - offline

Visualization Scenario – Interactive Steering



Scenario: interactive steering / computational steering

Sources of Errors

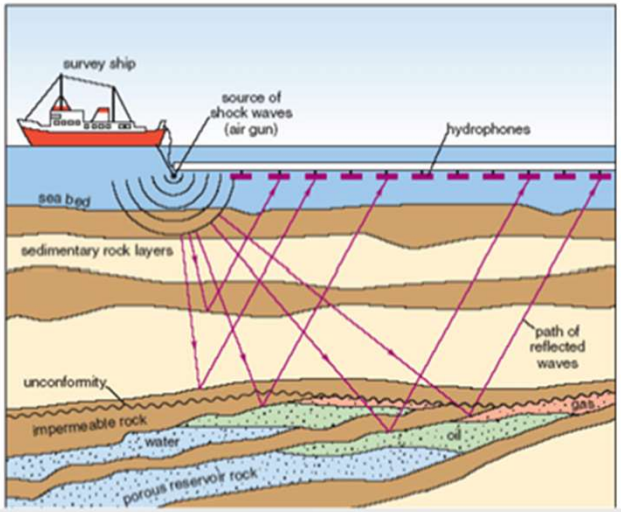
- Data acquisition:
  - Sampling: are we (spatially) sampling data with enough precision to get what we need out of it ?
  - Quantization: are we converting “real” data to a representation with enough precision to discriminate the relevant features ?
- Filtering:
  - Are we retaining/removing the “important/non-relevant” structures of the data ?
  - Selecting the “right” variable: Does this variable allow an analysis of the relevant features ?
  - Functional model for resampling: What kind of information do we introduce by interpolation and approximation ?
- Mapping:
  - Are we choosing the graphical primitives appropriately in order to depict the kind of information we want to get out of the data ?
- Rendering:
  - The need for interactive rendering often determines the chosen abstraction level.
  - Consider limitations of the underlying display technology: ergo, color quantization.
  - Carefully add “realism”: (the most realistic image is not necessarily the most informative one).

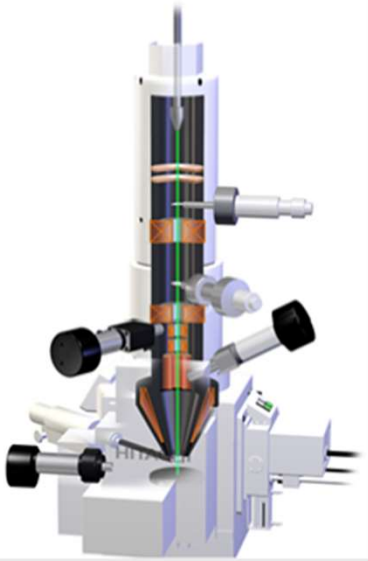
Overview

- Basics
- Data
- Mapping Techniques
- Surface Rendering Shading
- Volume Visualization
- Vector Field Visualization

Data Sources

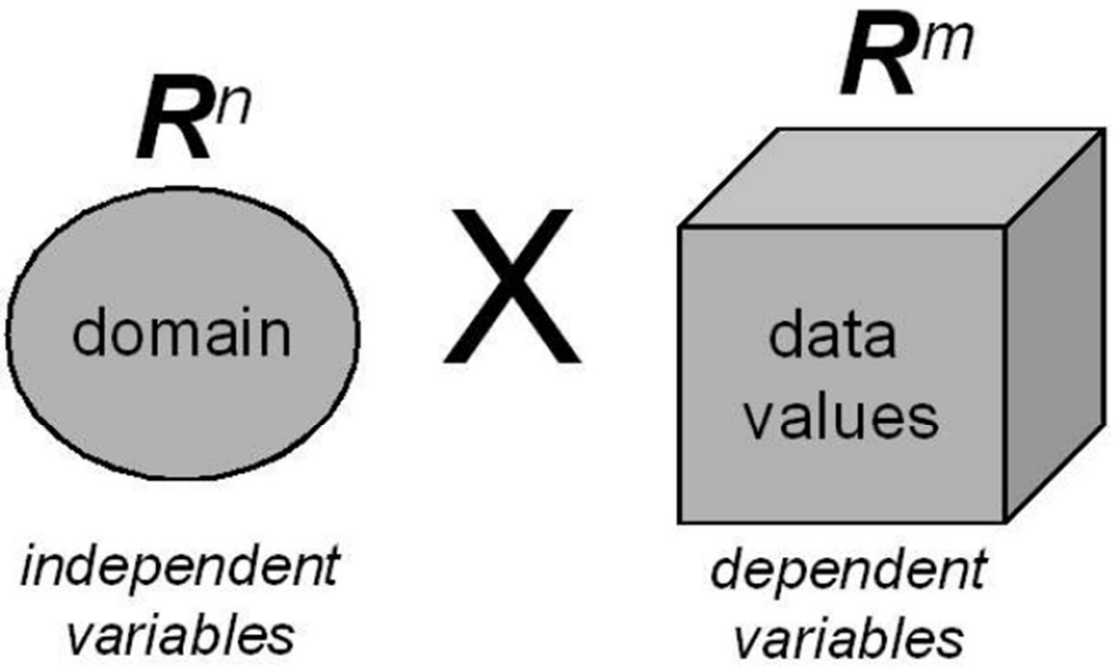
- Simulation
- Physical Measurements, Scientific Instruments





- Real world
  - Measurements and observation
- Theoretical world
  - Mathematical and technical models
- Artificial world
  - Data that is designed

Data Representation



scientific data $\subseteq R^{n+m}$

Data Representation

- Discrete representations
  - The objects we want to visualize are often ‘continuous’
  - But in most cases, the visualization data is given only at discrete locations in space and/or time
  - Discrete structures consist of samples, from which grids/meshes consisting of cells are generated

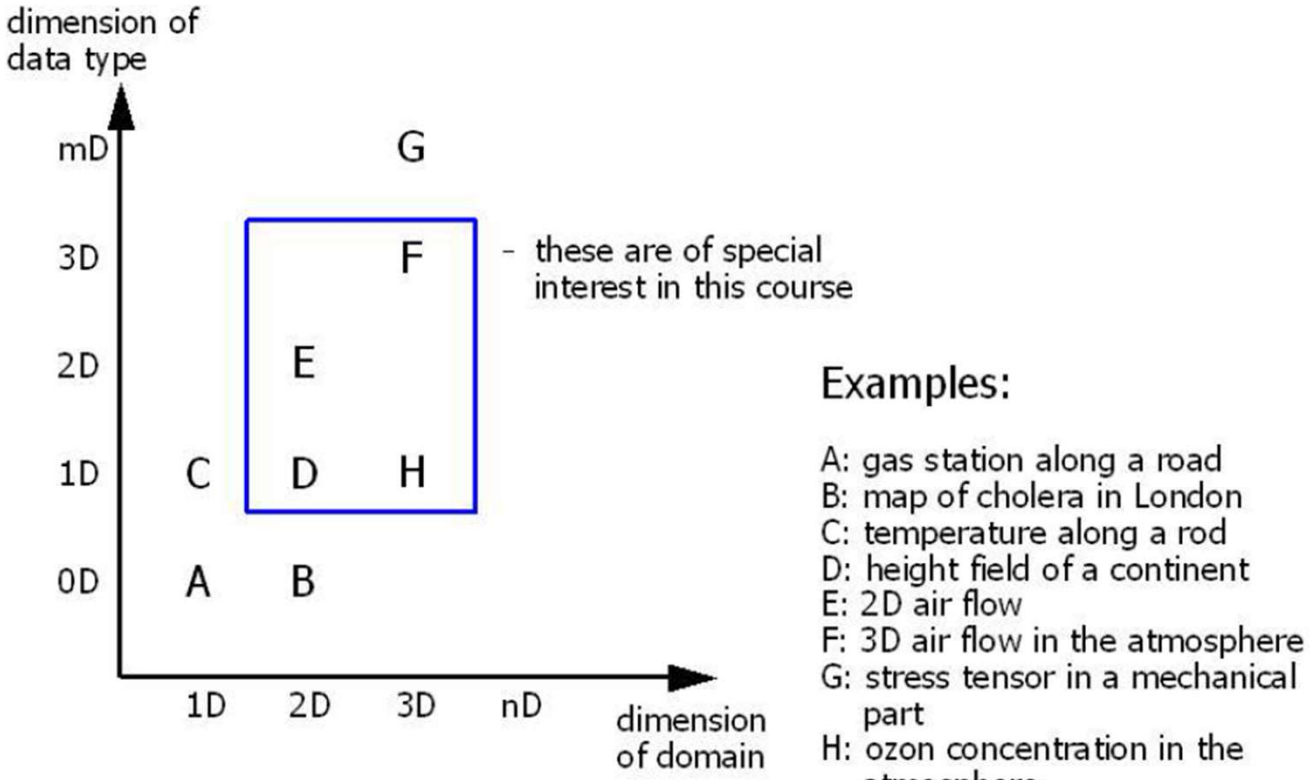
- Primitives in multi dimensions

dimension	cell	mesh
0D	points	
1D	lines (edges)	polyline(–gon)
2D	triangles, quadrilaterals (rectangles)	2D mesh
3D	tetrahedra, prisms, hexahedra	3D mesh

A mesh provides a discrete represenation of a geometric domain of interest,

Data Dimension

- Classification of visualization techniques according to:
  - Dimension of the domain of the problem (independent params)
  - Type and dimension of the data to be visualized (dependent params)



Domain

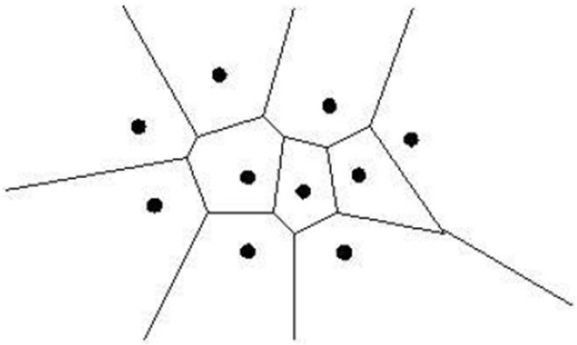
- The space in which the data is considered is called domain.
- The points for which data is available are called sample points.
- For the visualization the following characteristics are of interest:
  - Dimension:
    - The independent variables can be discrete or continuous.
    - If there are more than two dimensions a projection is necessary.
    - This may cause occlusion and ambiguity.
  - Influence
  - Structure:
    - Connection between sample points
- The (geometric) shape of the domain is determined by the positions of sample points.

Influence

- Values at sample points influence the data distribution in a certain region around these samples
- To reconstruct the data at arbitrary points within the domain, the distribution of all samples has to be calculated
  - Point influence: Only influence on point itself
  - Local influence : Only within a certain region
  - Global influence: Each sample might influence any other point within the domain

Voronoi Diagram

- The **Voronoi-diagram** is a decomposition of the domain.
- It is constructed by creating a region around each sample point, which can be considered as the area of influence of that sample point.
- Each region contains only a single sample point.
- A certain region contains all points that are closer to that sample than to every other sample.
- Each point in a certain region is assigned the value of the corresponding sample point



Data Structures - Requirements

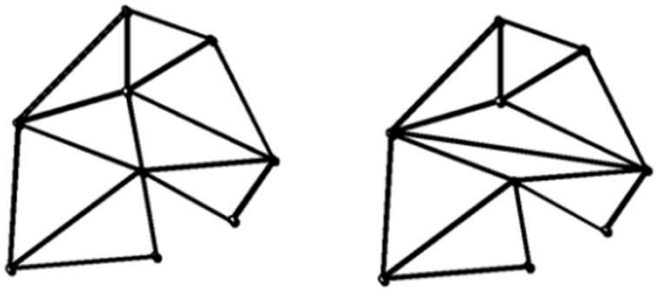
- Convenience of access
- Space efficiency
- Lossless vs. lossy
- Portability
  - binary – less portable, more space/time efficient
  - text – human readable, portable, less space/time efficient

Data Structures

- If points are arbitrarily distributed and no connectivity exists between them, the data is called scattered
- Otherwise, the data is composed of cells bounded by grid lines
- Topology specifies the structure (connectivity) of the data
- Geometry specifies the position of the data

Geometrical vs Topological Equivalence

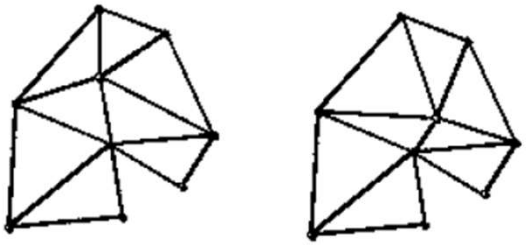
Geometrical Equivalence



Same geometry (vertex positions), different topology (connectivity)

Topological Equivalence

- Things that can be transformed into each other by stretching and squeezing, without tearing or sticking together bits which were previously separated



Topological equivalence

Fields

Attribute values associated with cells

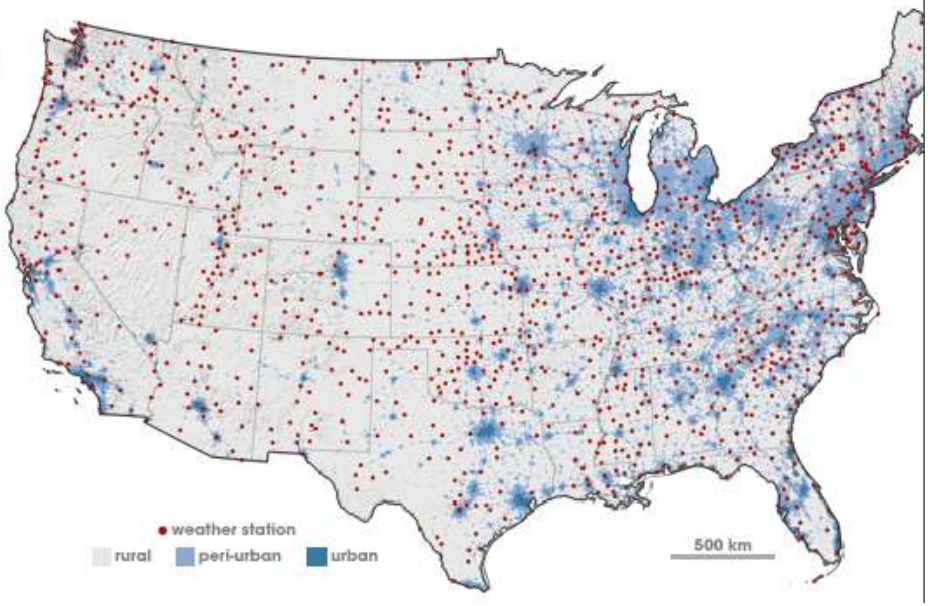
Cell contains data from continuous domain

- Temperature, pressure, wind velocity

Measured or simulated

Sampling & Interpolation

- Signal processing & stats



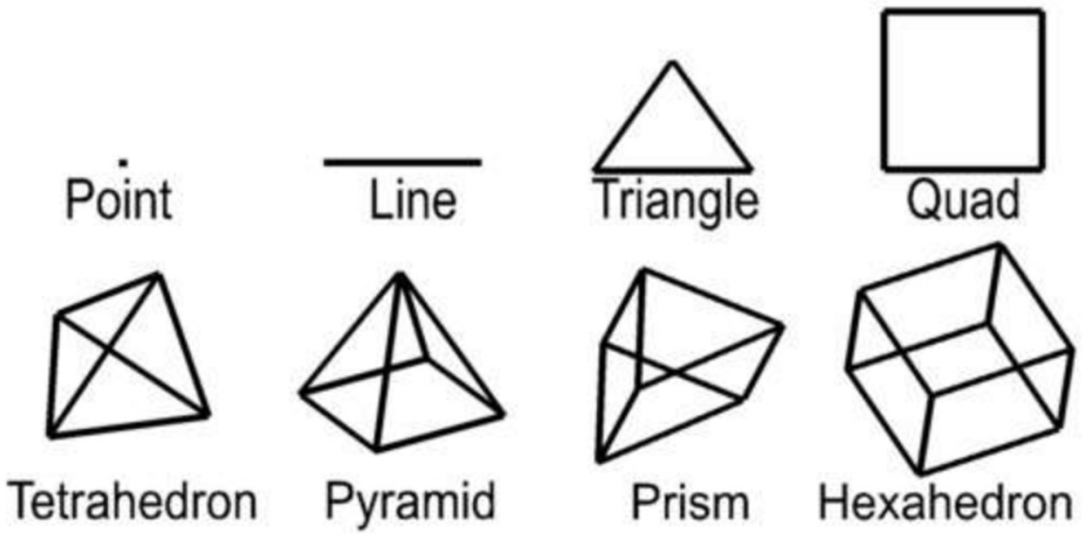
Grids

- Although there is great value in directly visualizing measured data; it does have many limitations.
- Unless sparse measured data are mapped to a grid, computation of contaminant areas or volumes is not possible.
- Further, the techniques available for visualizing the data will be very limited.
- Therefore a grid is created into which the data will be interpolated and extrapolated.

- A grid is defined as a collection of nodes and cells.
- Nodes are points in two or three-dimensions with coordinates and usually one or more data values.
- Cell refer to geometric objects. Cell values between grid points are resampled by interpolation
- The cell type and the nodes that comprise their vertices define these objects.

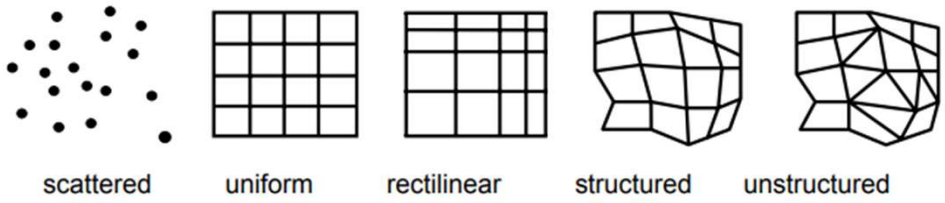
Cells

Cell Type	Number of Nodes	Dimensionality
Point	1	0
Line	2	1
Triangle	3	2
Quadrilateral	4	2
Tetrahedron	4	3
Pyramid	5	3
Prism	6	3
Hexahedron	8	3



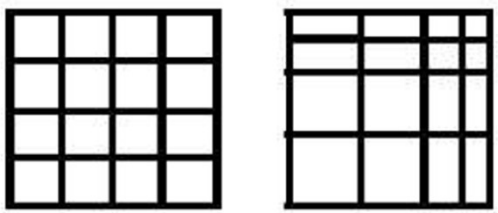
Grid Types

- Grids differ substantially in the simplicial elements (or cells) they are constructed from and in the way the inherent topological information is given



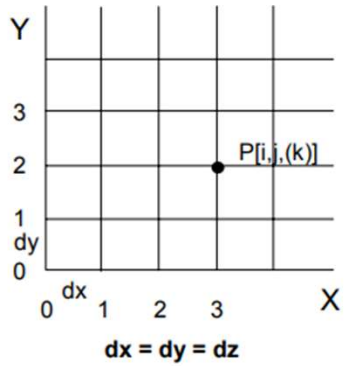
Structured Grids

- Have a regular topology and regular / irregular geometry
- A structured grid is often composed of sets of connected parallelograms (hexahedra), with cells being equal or distorted with respect to (non-linear) transformations.
- They may require more elements or badly shaped elements in order to precisely cover the underlying domain.
- Its topology is represented implicitly by an n-vector of dimensions.
- Its geometry is represented explicitly by an array of points.
- Every interior point has the same number of neighbors.



Cartesian Grid

- Structured grid
- Cells and points are numbered sequentially with respect to increasing X, then Y, then Z, or vice versa
- Number of points = Nx•Ny•Nz
- Number of cells = (Nx-1)•(Ny-1)•(Nz-1)
- Vertex positions are given implicitly from [i,j,k]:
  - P[i,j,k].x = origin + i • dx
  - P[i,j,k].y = origin + j • dy
  - P[i,j,k].z = origin + k • dz

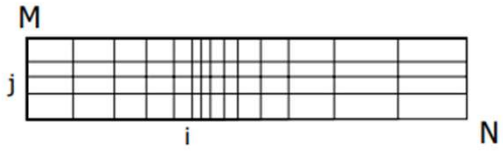


Uniform Grid

- Similar to Cartesian grids
- Consist of equal cells but with different resolution in at least one dimension (dx ≠ dy (≠ dz))
- Spacing between grid points is constant in each dimension -> same indexing scheme as for Cartesian grids
- Most likely to occur in applications where the data is generated by a 3D imaging device providing different sampling rates in each dimension

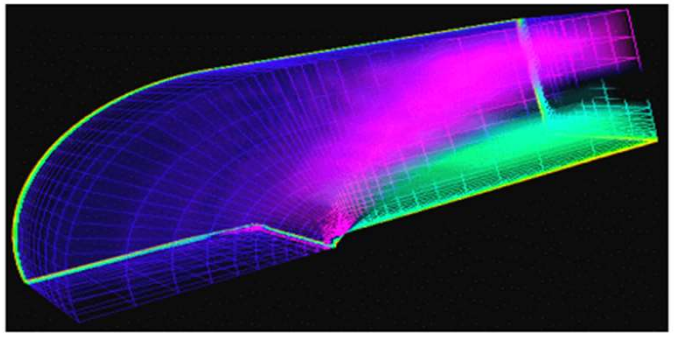
Rectilinear Grids

- Topology is still regular but irregular spacing between grid points
- Non-linear scaling of positions along either axis
- Spacing, x_coord[L], y_coord[M], z_coord[N], must be stored explicitly



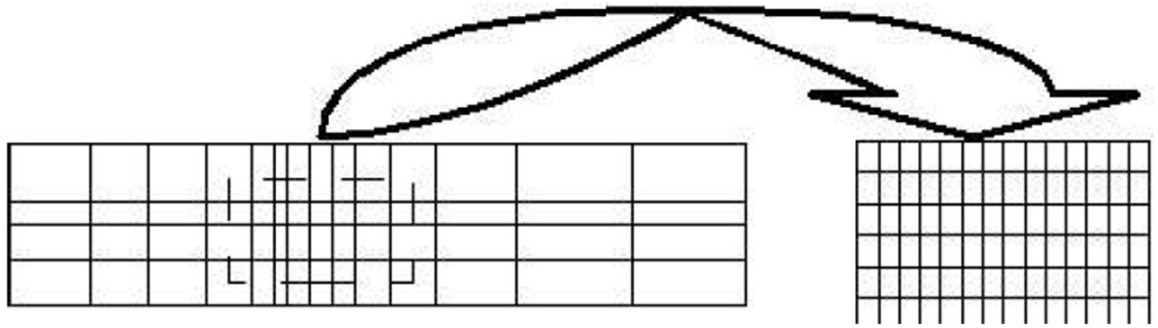
Curvilinear Grids

- Topology is still regular but irregular spacing between grid points
- Positions are non-linearly transformed
- Topology is still implicit, but vertex positions are explicitly stored
  - x_coord[L,M,N]
  - y_coord[L,M,N]
  - z_coord[L,M,N]
- Geometric structure might result in concave grids



Multigrids

- Multigrids are used to avoid wasted detail if the focus is only in some areas.
- In these regions of interest the grid is just finer than in the rest of it.

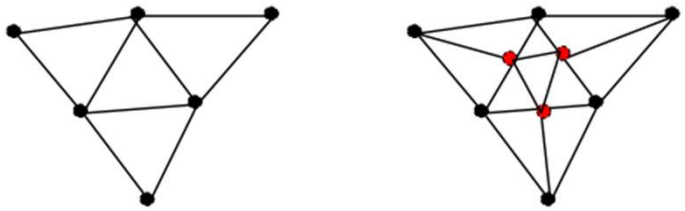


Unstructured Grids

- If no implicit topological (connectivity) information is given the grids are termed unstructured grids
- Unstructured grids are often computed using quadtrees (recursive domain partitioning for data clustering), or by triangulation of points sets
- The task is often to create a grid from scattered points
- Characteristics of unstructured grids
  - Grid point geometry and connectivity must be stored
  - Dedicated data structures needed to allow for efficient traversal and thus data retrieval
  - Often composed of triangles or tetrahedra
  - Less elements are needed to cover the domain

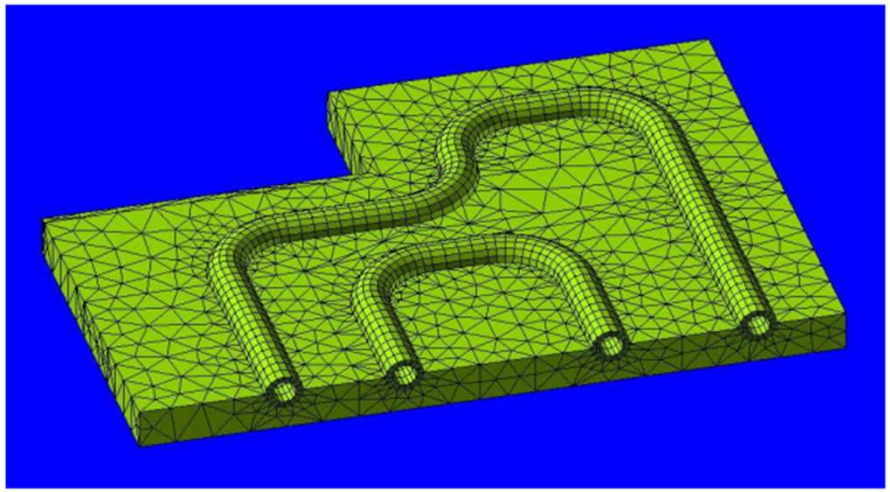
Unstructured Grids

- Composed of arbitrarily positioned and connected elements
- Can be composed of one unique element type or they can be hybrid (tetras, hexas, prisms)
- Triangle meshes in 2D and tetrahedral grids in 3D are most common
- Can adapt to local features (small vs. large cells)
- Can be refined adaptively
- Simple linear interpolation in simplices

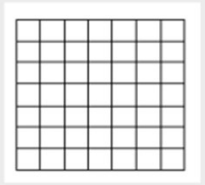


Hybrid Grids

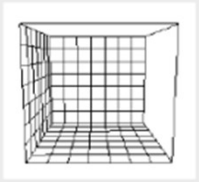
- Combination of different grid types



Grid - Examples



2D-Regular Grid

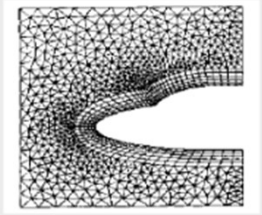


3D-Regular Grid

2D-Irregular Grid

2D-Structured Grid

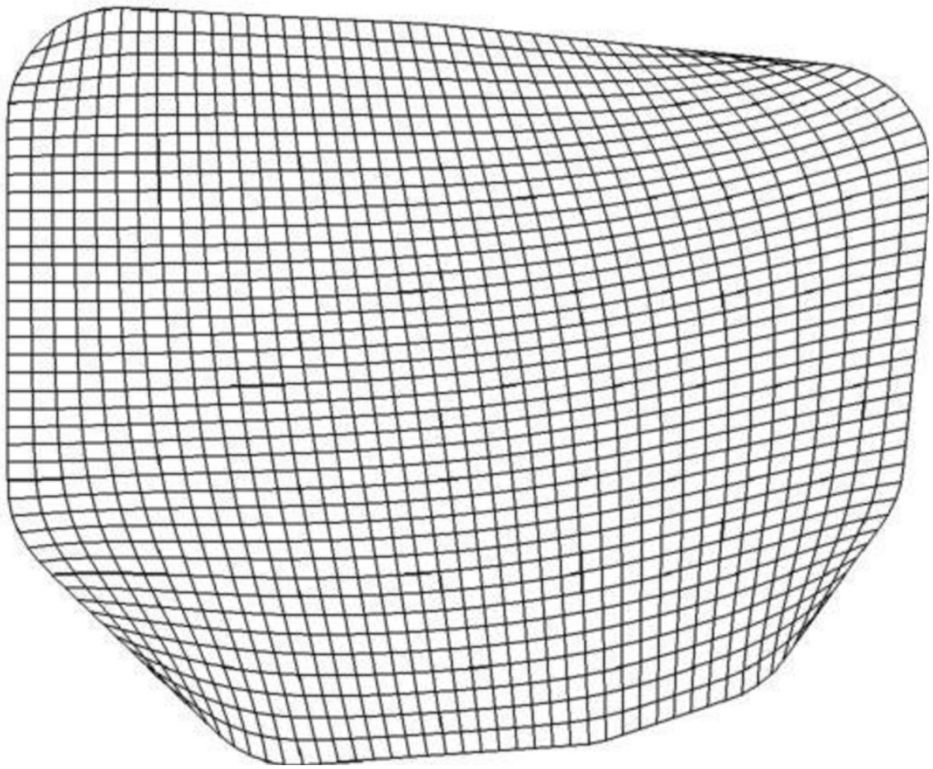
3D-Structured Grid



2D-Hybrid Grid

Convex Hull

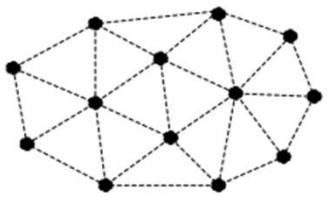
- The convex hull of a set of points in two-dimensional space is the smallest convex area containing the set.
- In the x-y plane, the convex hull can be visualized as the shape assumed by a rubber band that has been stretched around the set and released to conform as closely as possible to it.
- The convex hull best fits the spatial extent of the data.
- The convex hull defines an area. That area can be gridded in many ways.

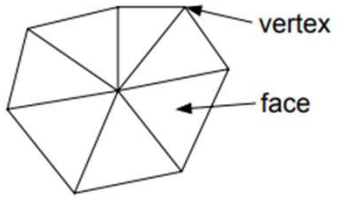


A two-dimensional example of a convex hull grid

Scattered Data

- Scattered data consists of irregularly distributed positions without connectivity information.
- To get a connectivity it is necessary to find a "good" triangulation
  - triangular/tetrahedral mesh with scattered points as vertices.
- For a set of points there are many possible triangulations.
- A measure for the quality of a triangulation is the aspect ratio of the so-defined triangles.
  - Long and thin triangles should be avoided.





Grid vs Mesh

- Grids are typically a set of simulation elements that have a well defined structure
- Meshes are often more general. They may be unstructured and use various shapes of elements, sometimes even mixing elements of different types in the same mesh.
- Sometimes they are considered equivalent.

Interpolation

- We have only a finite number of sample points, i.e., the continuous signal is only known at few points (data points).
- But in general data is also needed between this points.
- By interpolation we obtain a representation that matches the function at the data points.
- Interpolation helps to reconstruct the signal at points that are not sampled.

Scattered Data Interpolation

- The **scattered data interpolation** is a solution for local or global influence problems.
- At each point the weighted average of all sample points in the domain is computed
- Weighting functions determine the support of each sample point
- Radial basis functions simulate decreasing influence with increasing distance from samples

Spatial Interpolation

- Spatial interpolation methods are used to estimate measured data to the nodes in grids that do not coincide with measured points.
- The spatial interpolation methods differ in their assumptions, methodologies, complexity, and deterministic or stochastic nature.
- Inverse Distance Weighted
  - Inverse distance weighted averaging (IDWA) is a deterministic estimation method where values at grid nodes are determined by a linear combination of values at known sampled points.
  - IDWA makes the assumption that values closer to the grid nodes are more representative of the value to be estimated than samples further away.
  - Weights change according to the linear distance of the samples from the grid nodes.
  - The spatial arrangement of the samples does not affect the weights.
  - IDWA has also been shown to work well with noisy data.
  - The choice of power parameter in IDWA can significantly affect the interpolation results.
  - As the power parameter increases, IDWA approaches the nearest neighbor interpolation method where the interpolated value simply takes on the value of the closest sample point.
  - Optimal inverse distance weighting is a form of IDWA where the power parameter is chosen on the basis of minimum mean absolute error.

Data Type

- Scalar
- Vector
- Tensor: A **tensor** is an algebraic object that describes a multilinear relationship between sets of algebraic objects related to a vector space

- Volume
  - 3D array of point samples
  - Sample of images
  - Voxel (3D pixel)

Pixels & Voxels

- Pixel:
  - Picture element *a dot that represents the smallest graphic unit of display on the screen.*

- Voxel (3D pixel):
  - A **voxel** represents a value on a regular grid in three-dimensional space.
  - As with pixels in a 2D bitmap, voxels themselves do not typically have their position (i.e. coordinates) explicitly encoded with their values.
  - Instead, rendering systems infer the position of a voxel based upon its position relative to other voxels (i.e., its position in the data structure that makes up a single volumetric image).
  - Values are constant within a region around a grid point

Overview

- Basics
- Data
- Mapping Techniques
- Surface Rendering Shading
- Volume Visualization
- Vector Field Visualization

Mapping Techniques

- Mapping to geometry
  - Function Plots
  - Height fields
  - Isocontours (Isolines/ Isosurfaces)
- Color mapping

Function Plots

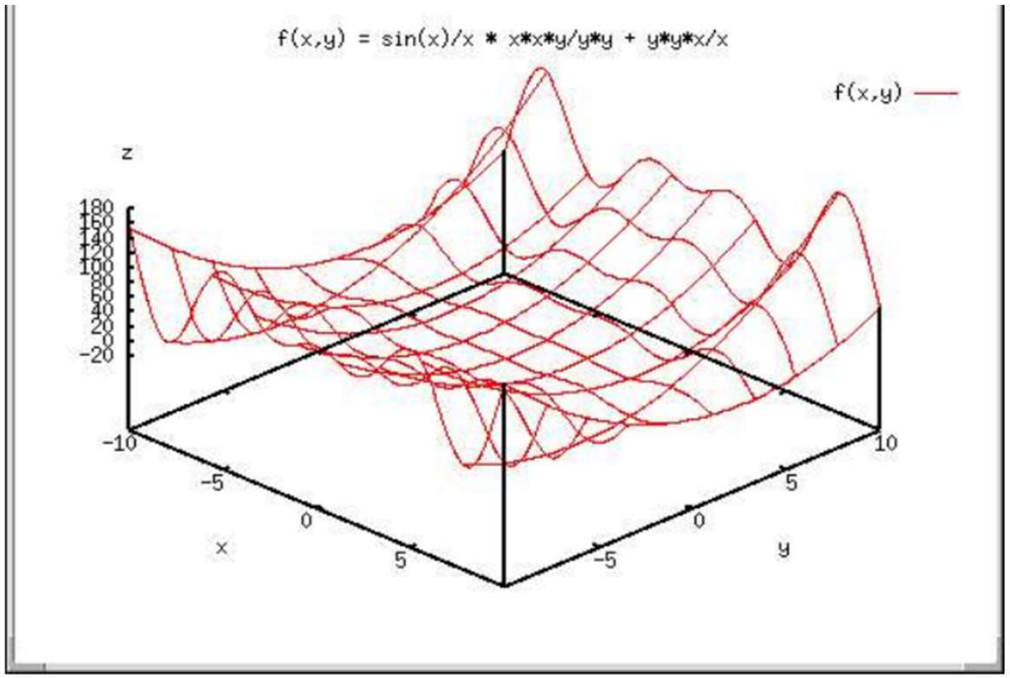
- Function plot for a 1D scalar field
  - Points $\{(s, f(s)) \mid s \in \mathbb{R}\}$
  - 1D manifold: line
- Function plot for a 2D scalar field
  - Points $\{(s, t, f(s,t)) \mid (s,t) \in \mathbb{R}^2\}$
  - 2D manifold: surface

Height Fields

- A height field is a visual representation of a function which takes as input a two-dimensional point and returns a scalar value ("height") as output.
- In other words, a height field is a two-dimensional array in X and Y direction. The array values represent the Z value of a surface at each point.
- Height fields are used to model any kind of surface, e.g. Landscapes
- There are three different types for the representation of surfaces:
  - Wireframe
  - Hidden Lines
  - Shaded surface

Wireframe Representation

- The wireframe representation requires the specification of viewing parameters.
- Every edge of the grid is drawn, even if it is hidden by a face lying in front.
- For the viewer the result appears as a transparent surface.

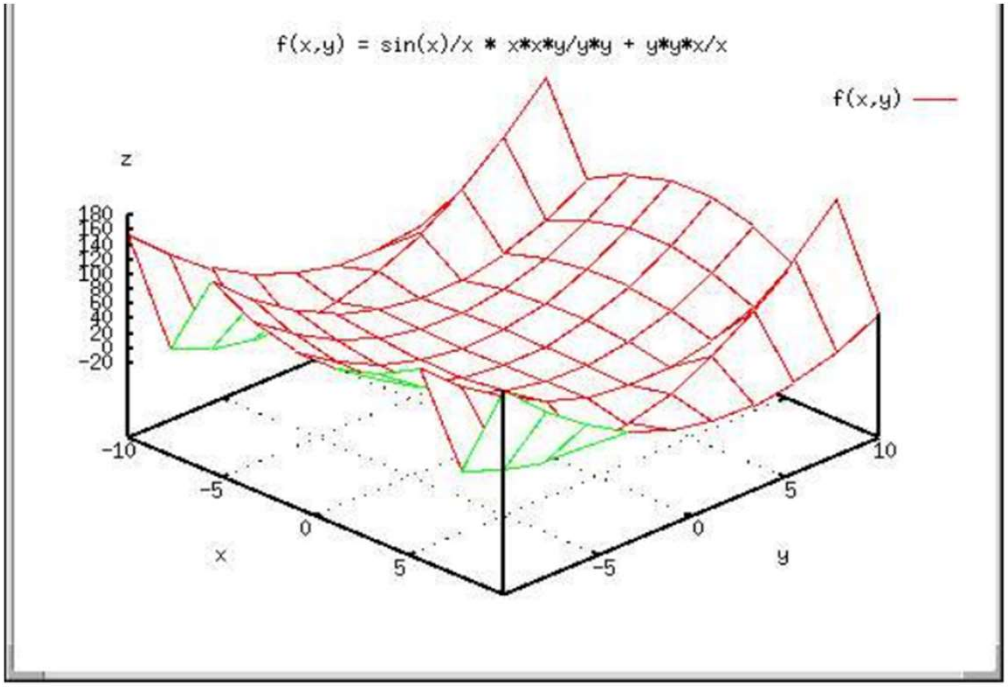


This graphic shows a 3D height field. The given plot represents the function

$$f(x,y)=\frac{\sin(x)}{x}+\frac{x^2 y}{y^2}+\frac{y^2 x}{x}$$

Hidden Lines representation

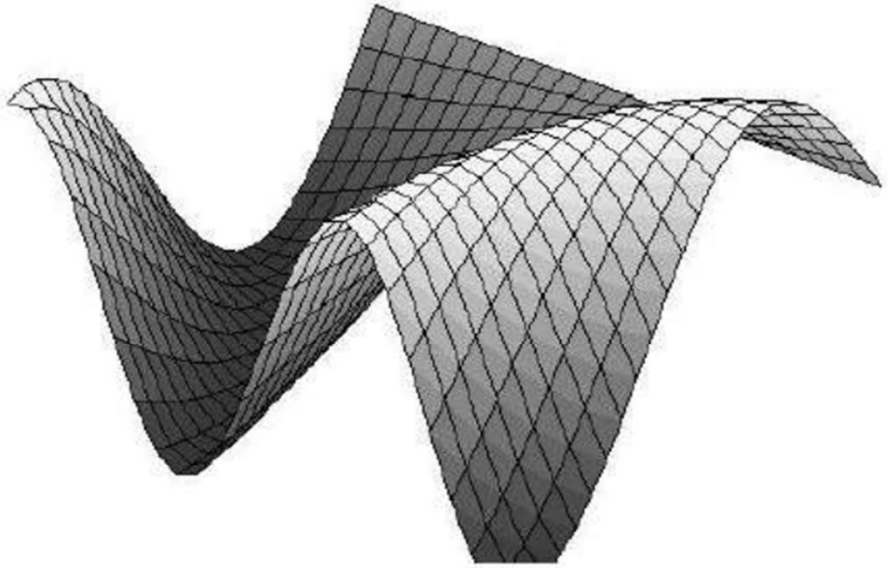
- Contrary to a wireframe the hidden lines representation removes all edges that belong to hidden faces.
- Due to this occlusion it gives the viewer a better spatial orientation



Plot of the function $f(x,y)=\frac{\sin(x)}{x}+\frac{x^2y}{y^2}+\frac{y^2x}{x}$ *with hidden lines.*

Shaded Surface Representation

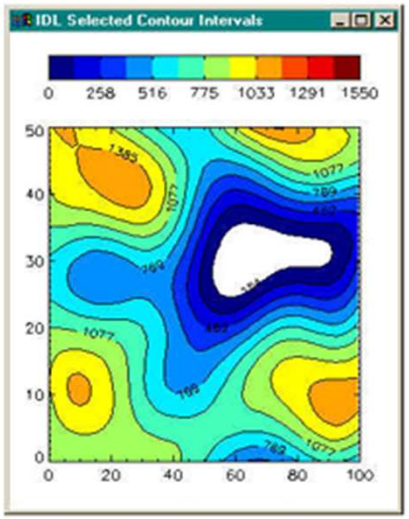
- The shaded surface is an extended version of the hidden lines representation.
- In addition to removed hidden faces it shades every polygon dependant on the influence of incident light.
- Therefore it requires the specification of a lighting/shading model



This picture shows a shaded surface with the light source in the top right corner.

Isocontours

- Contours: Set of points where the scalar field s has a given value c.
- Isocontours: Points in a scalar field that have a constant value
  - Isolines: 2D
  - Isosurfaces: 3D



isolines

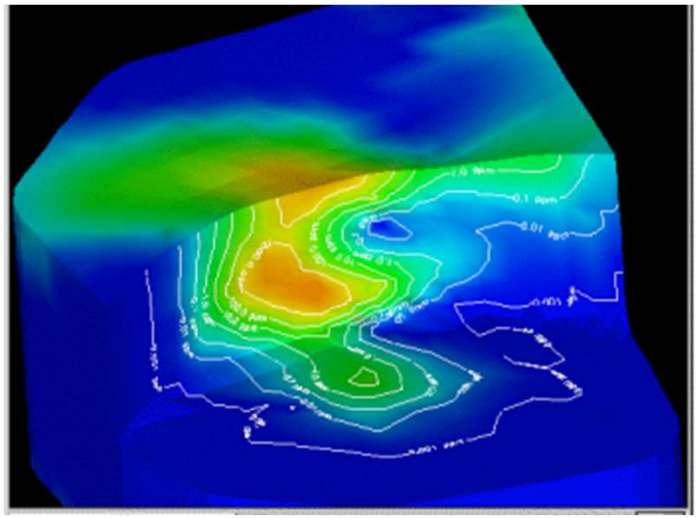
Isosurfaces

Isolines

- Visualization of 2D scalar fields
- Given a scalar function $f : \Omega \mapsto \mathbb{R}$

  and a scalar value $c \in \mathbb{R}$
- Isoline consists of points

$$\{(x, y) \mid f(x, y) = c\}$$



- Results in contour lines if connected.
- If f() is differentiable and grad(f) ≠ 0, then isolines are curves
- Isolines are often used for maps, weather charts, thermal diagrams or any other kind of 2 or 3 dimensional data representation.

Pixel by Pixel Contouring Algorithm

- Isolines can be implemented with a simple pixel by pixel contouring algorithm
- It is a straightforward approach, scanning all pixels for equivalence with a given isovalue.

Input:

$f:(1,\ldots,x_{max})\times(1,\ldots,y_{max})\rightarrow\mathbb{R}$

Isovalues $I_1,\ldots,I_n$ and isocolors $c_1,\ldots,c_n$

Algorithm:

for all $(x,y)\in(1,\ldots,x_{max})\times(1,\ldots,y_{max})$ do

for all $k\in\{1,\ldots,n\}$ do

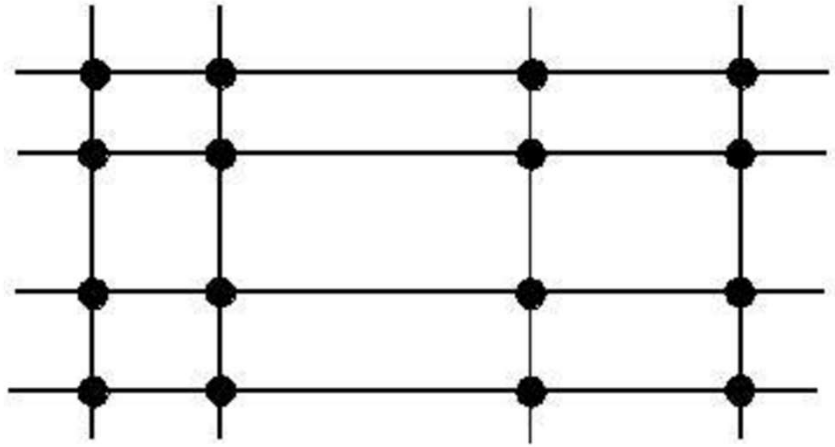
if $|f(x,y)-I_k|<\epsilon$ then

draw (x,y,c_k)

Marching Square Algorithm

- **Marching squares** is an algorithm that generates contours for a two-dimensional scalar field
- Used to compute the representation of a scalar function on a rectilinear grid
- The contours can be of two kinds:
  - *Isolines* – lines following a single data level, or *isovalue*.
  - *Isobands* – filled areas between isolines.

Marching Square Algorithm



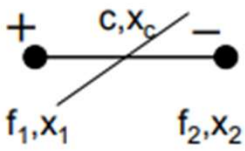
Rectilinear grid with scalar values represented as points.

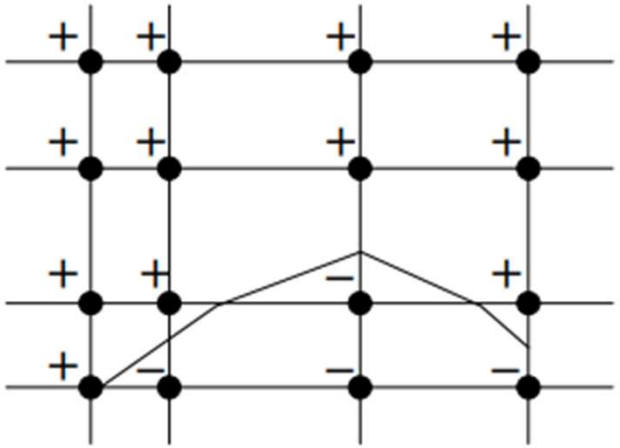
- Scalar values are given at each vertex f↔fij
- Take into account the interpolation within cells
- Divide and conquer: consider cells independently of each other

Marching Square Algorithm

- Which cells will be intersected ?
  - Initially mark all vertices by + or – , depending on the conditions $f_{ij} \geq c$, $f_{ij} < c$
- No isoline passes through cells (=rectangles) which have the same sign at all four vertices
  - So we only have to determine the edges with different signs
  - And find intersection by linear interpolation

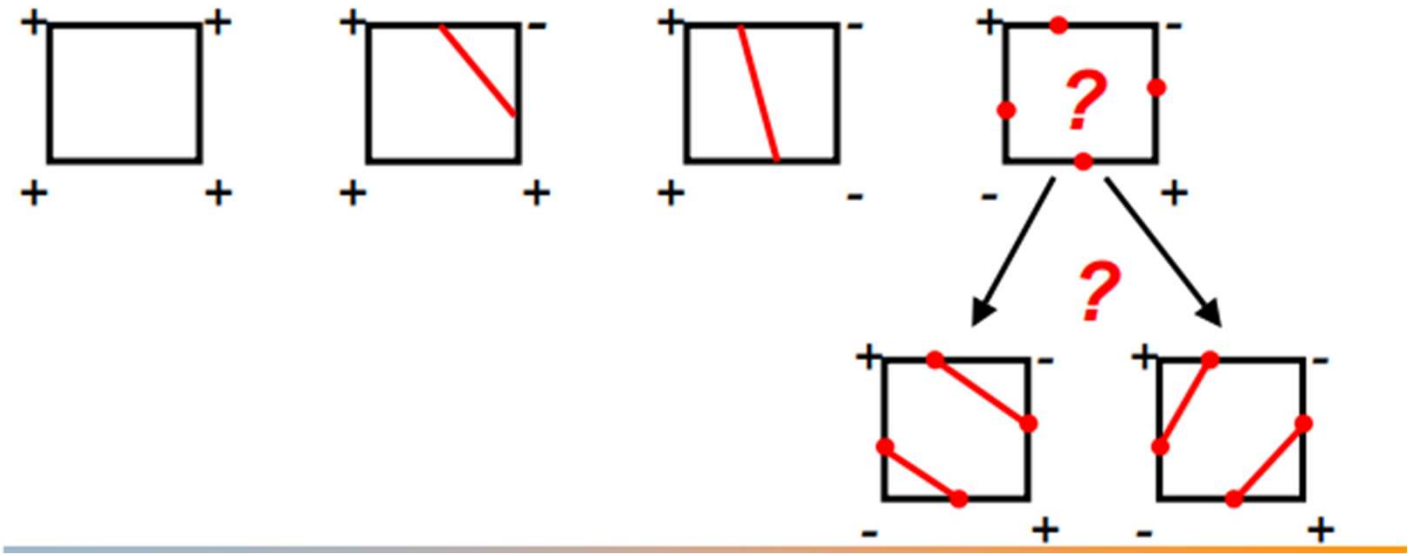
$$x_c = [(f_2 - c)x_1 + (c - f_1)x_2] / (f_2 - f_1)$$





Marching Square Algorithm

- Only 4 different cases (classes) of combinations of signs
- Symmetries: rotation, reflection, change + ↔ -
- Compute intersections between isoline and cell edge, based on linear interpolation along the cell edges

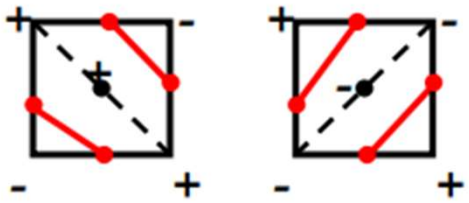


Marching Square Algorithm

- We can distinguish the cases by a decider
- Mid point decider
  - Interpolate the function value in the center

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

  - If $f_{center} < c$ we chose the right case, otherwise we chose the left case



  - Not always correct solution

Marching Square Algorithm – Cell Order Approach

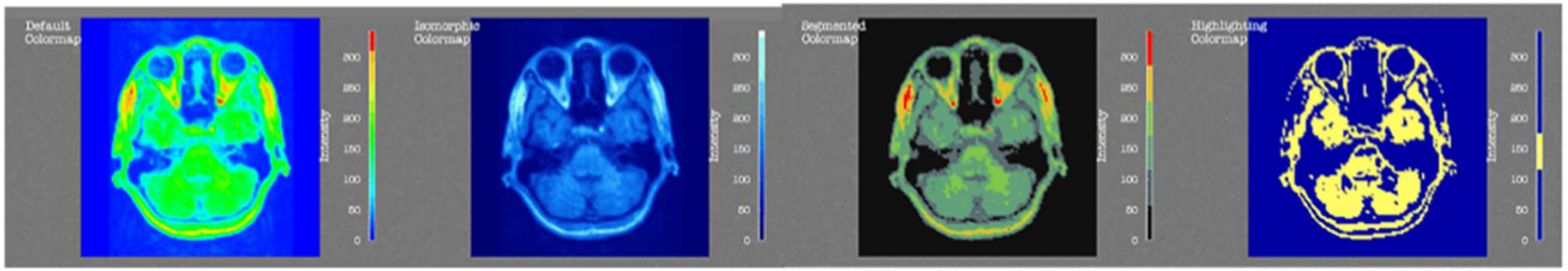
- Traverse all cells of the grid
- Apply marching squares technique to each single cell
- Disadvantage
  - Every vertex (of the isoline) and every edge in the grid is processed twice
  - The output is just a collection of pieces of isolines which have to be postprocessed to get (closed) polyline

Marching Square Algorithm – Contour Tracing Approach

- Start at a seed point of the isoline
- Move to the neighboring cell into which the isoline enters
- Trace isoline until:
  - Bounds of the domain are reached or
  - Isoline is closed
- Problem:
  - How to find seed points efficiently?
  - In a preprocessing step, mark all cells which have a sign change
  - Remove marker from cells which are traversed during contour tracing (unless there are 4 intersection edges!)

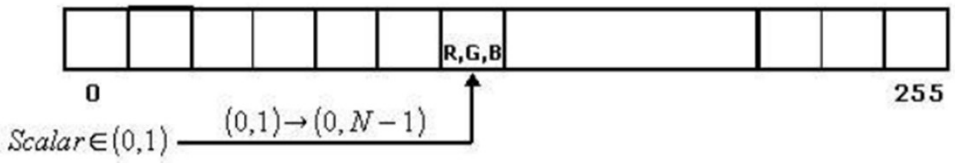
Color Mapping

- Map scalar value to color
  - Color table (e.g., array with RGB entries)
  - Procedural computation
  - With opacity: 1D *transfer function*



Color Coding for Scalar Data

- Assign to each scalar value a different color value
- Assign via transfer function T
  - T: scalar value -> color value
- Code color values into a color lookup table
  - Default range of a color lookup table is 256



Linear Transfer Function for Color Coding

- Specify color for f_{min} and for f_{max}
  - $(R_{min}, G_{min}, B_{min})$ and $(R_{max}, G_{max}, B_{max})$
- Linearly interpolate between them

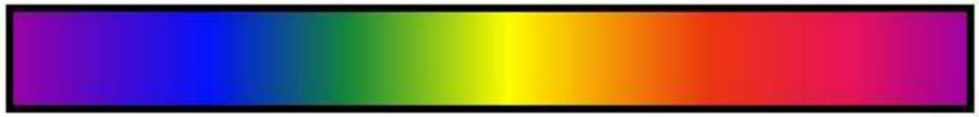
$$f \mapsto \frac{f - f_{min}}{f_{max} - f_{min}}(R_{min}, G_{min}, B_{min}) + \frac{f_{max} - f}{f_{max} - f_{min}}(R_{max}, G_{max}, B_{max})$$

Color Tables

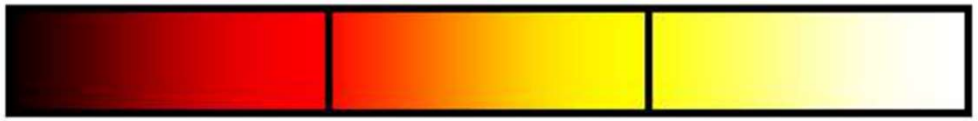
- Different color spaces lead to different interpolation functions
- In order to visualize (enhance/suppress) specific details, non-linear color lookup tables are needed

- **Gray scale color table**
  - Intuitive Ordering

- **Rainbow color table**
  - Less intuitive
  - HSV color mode
  - The rainbow color table has transitions from purple to blue, to green, to yellow, to orange, to red, to purple again.
  -



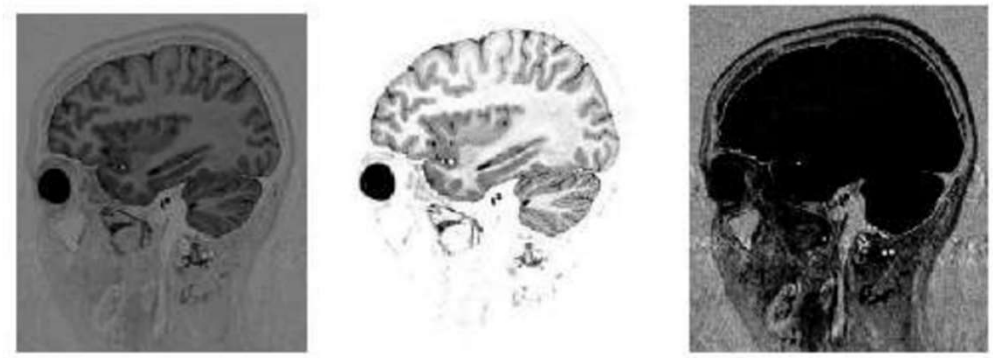
- **Temperature color table**
  - black – red – yellow - white



Color Table - Example

- Frequently transfer functions are specified interactive, in order to extract important characteristics

- Example: Special color table to visualize the brain tissue



Original, Brain, Tissue.

Readings

- Gregory M. Nielson, Hans Hagen, Heinrich Müller. Scientific visualization: overviews, methodologies, and techniques. IEEE Computer Society Press, 1997