VISUALIZATION

Scientific Visualization - II

Acknowledgement: ETH Zurich

Overview

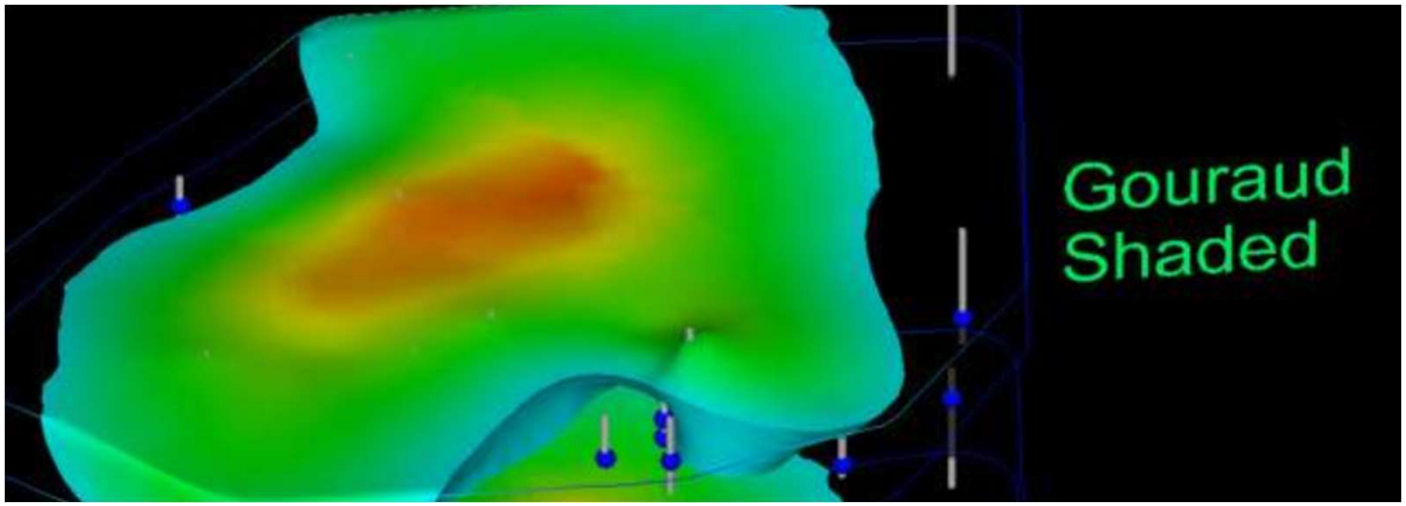
- Basics
- Data
- Mapping Techniques
- Surface Rendering Shading
- Volume Visualization
- Vector Field Visualization

Surface Rendering Shading Techniques

- There are various Surface Rendering shading techniques:
  - Gouraud Shading
  - Flat Shading
  - Background Shading
- The choice of surface rendering shading technique has a dramatic impact on the visualizations.
- Transparency could be applied to any of the surface rendering techniques except background shading.
- Transparency provides a means to see features or objects inside while still providing the basic shape.
- Objects inside a colored transparent object will have altered colors and the colors of the transparent object are affected by the color of the background and any other objects inside or behind.

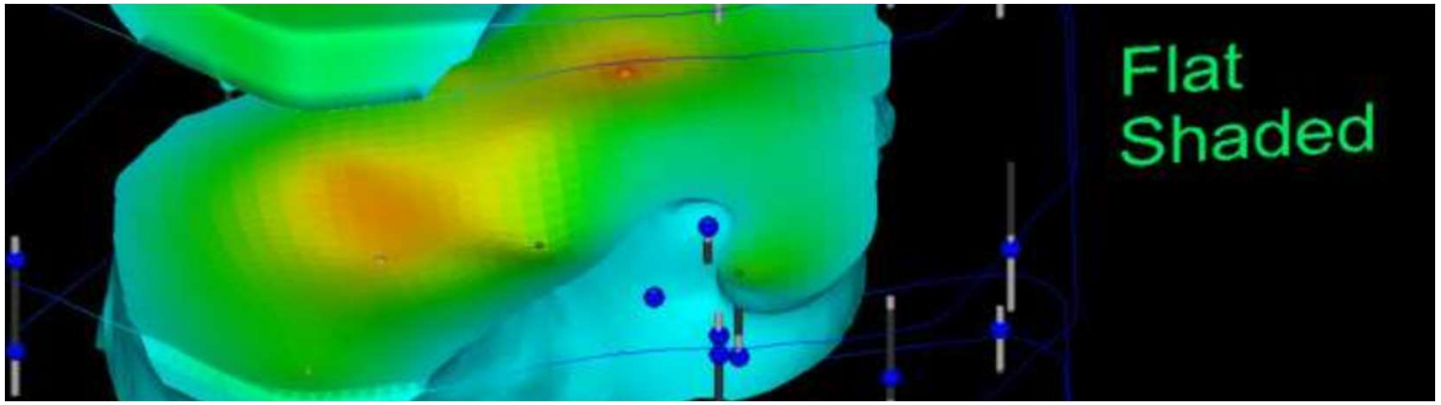
Gouraud Shading

- Gouraud shading is a method to simulate the differing effects of light and color across the surface of an object.
- Intensity- interpolation is used to develop Gouraud shading.
- By intensity interpolation, the intensity of each and every pixel is calculated.
- Gouraud shading shades a polygon by linearly interpolating intensity values across the surface.
- By this, if we know the intensities of two points then we can able to find the intensity of any point in between them.



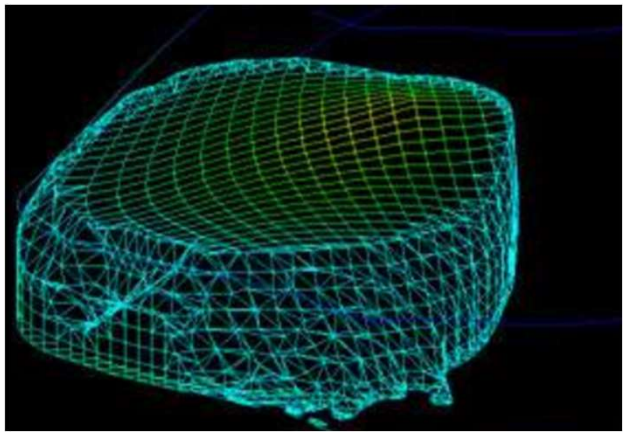
Flat Shading

- In flat shading, no textures are used and only one color tone is used for the entire object, with different amounts of white or black added to each face of the object to simulate shading.
- It assigns a single color to each polygon of a 3D object, based on the angle between the polygon's normal and the light source.
- Flat shading creates sharp edges and discontinuities between polygons, which can be useful for stylized or low-poly graphics, but not for realistic or smooth graphics.



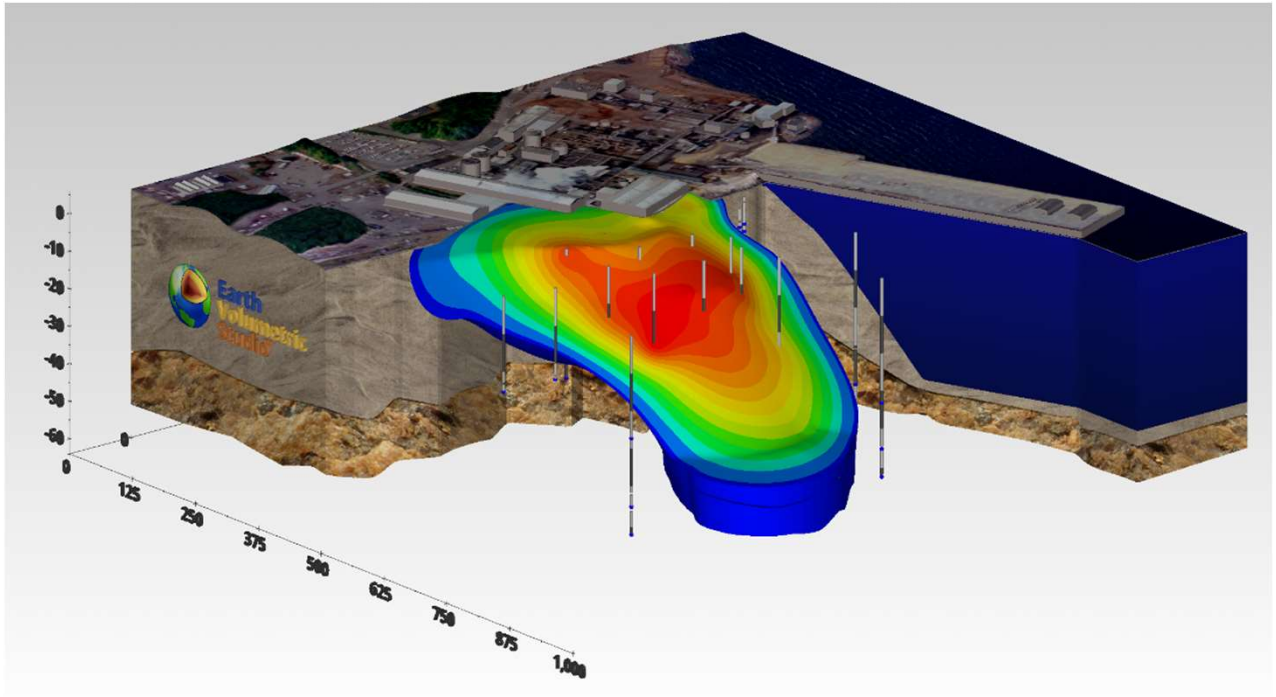
Background Shading

- In Background shading each cell of the plume is colored the same color as the background.
- This makes the cell invisible,
- However the cell is still opaque.
- Objects that are behind the background shaded cells are not visible.
- Background shading of the surfaces provides a "hidden line" rendering where the cells behind are not shown.



Texture Mapping

- Texture mapping is a process of projecting a raster image onto one or more surfaces.
- The images should be geo-referenced to ensure that the image's features are placed in the correct spatial location.
- In figure below a chlorinated hydrocarbon contaminant plume is shown at an industrial facility on the coast. (A plume is a vertical body of one fluid moving through another)
- Sand and rock geologic layers are displayed below the ocean layer.
- A color aerial photograph of the actual site was used to texture map and render the geologic layer that represents the ocean and was also applied to the three-dimensional representations of the site buildings as well as the ground surface.



Overview

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

Volumetric Data

- Volume data are 3D entities that may have information inside them
- They are obtained by sampling, simulation, or modeling techniques.
  - Medical: A sequence of 2D slices obtained from Magnetic Resonance Imaging (MRI), Computed Tomography (CT), or ultrasonography is 3D reconstructed into a volume model and visualized for diagnostic purposes or for planning of treatment or surgery.
  - Earth Science: Seismic Measurements
  - Chemistry: Electron Density Maps
- Volumetric data is typically a set S of samples (*x, y, z, v*), representing the value v of some property of the data at a 3D location (x, y, z).
- If *v* is simply a 0 or a 1, with 0 indicating background and 1 indicating the object, the data is called binary data.
- The data may instead be multivalued, with *v* representing some measurable property of the data, such as density, color, heat, or pressure.
- The value *v* may even be a vector, representing, for example, velocity at each location.

Characteristics of Volume Data

- Essential information in the interior
- But geometric representation of fire, clouds or gaseous phenomena can not be described
- Have to distinguish between shape (given by the geometry of the grid) and appearance (given by the scalar values or color, texture, etc.)
- Even if the data could be described geometrically, there are, in general, too many primitives to be represented

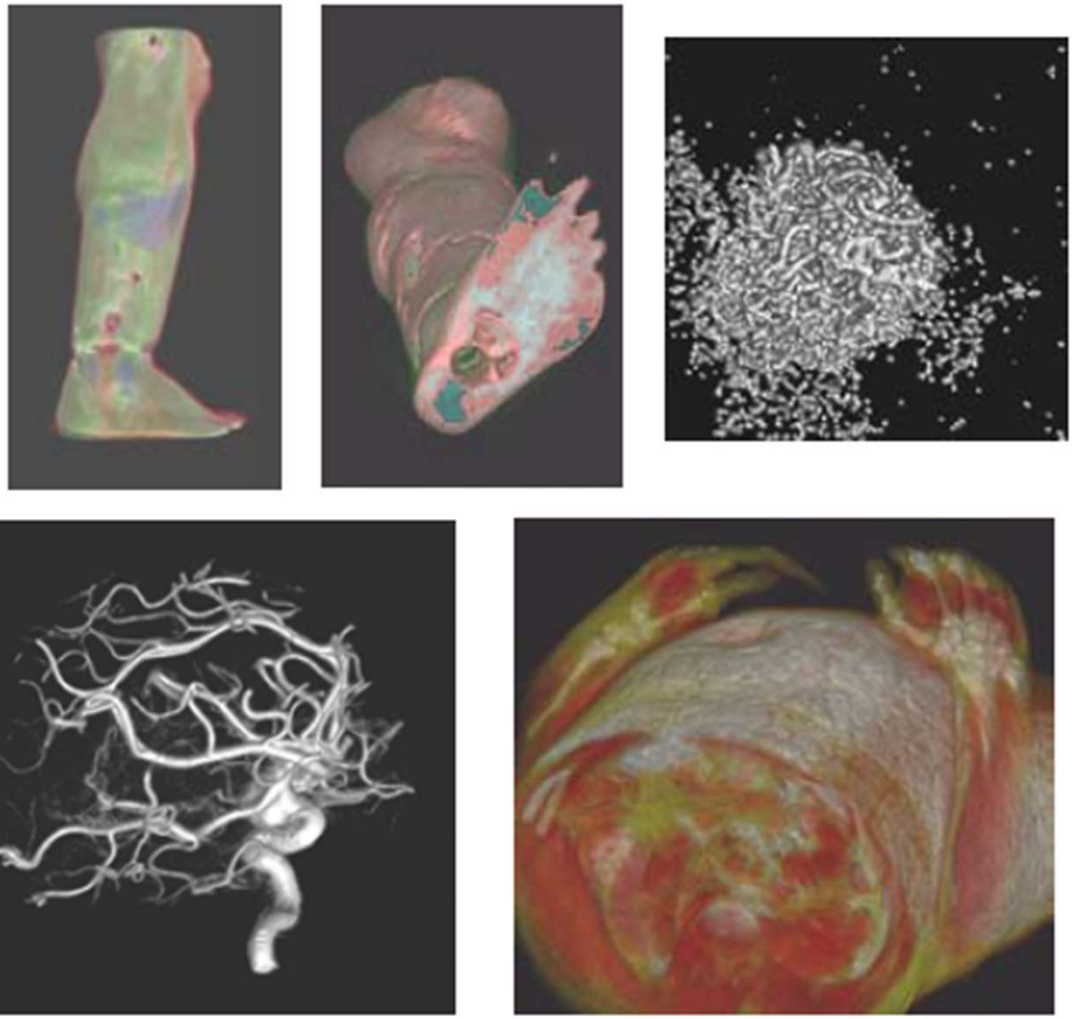
Volume Visualization

- Over the years many techniques have been developed to visualize volumetric data.
- Most of the early methods involved approximating a surface contained within the data using geometric primitives.
- When volumetric data are visualized using a surface rendering technique, a dimension of information is essentially lost.
- In response to this, volume rendering techniques were developed that attempt to capture the entire 3D data in a 2D image by projecting a 2D image directly from the 3D volumetric data.
- Volume rendering convey more information than surface rendering images, but at the cost of increased algorithm complexity, and consequently increased rendering times.

Volume Visualization

- Volume visualization is used to create two-dimensional graphical representations from scalar datasets that are defined on three-dimensional grids.
- Standard surface modeling only defines the opaque outer surface of an object, so you can not see inside of it.
- The basic idea of volume visualization is to make the boundaries of an
- object transparent, so that one can see inside.
- Examples of 3D data range:
  - Medical applications like CT, MRI scans, confocal microscopy, ultrasound
  - Seismic data
  - Fluid dynamics

Applications - Medical



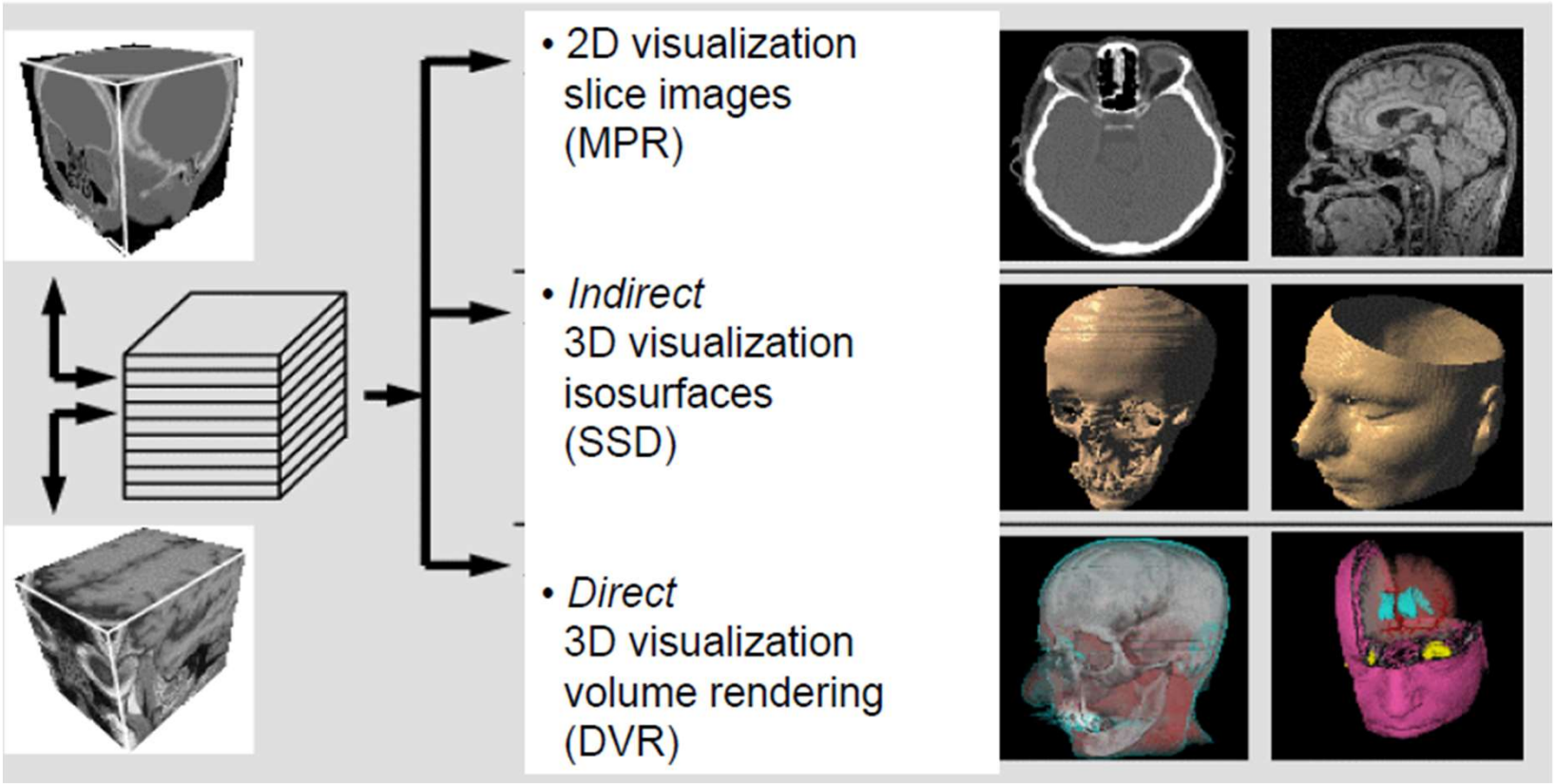
Volume Rendering

- Techniques
  - Techniques for 2D scalar fields
  - Indirect volume rendering techniques (e.g. surface fitting)
  - Direct volume rendering

- Goal
  - Integrate all different techniques in order to represent the data as “good” as possible.

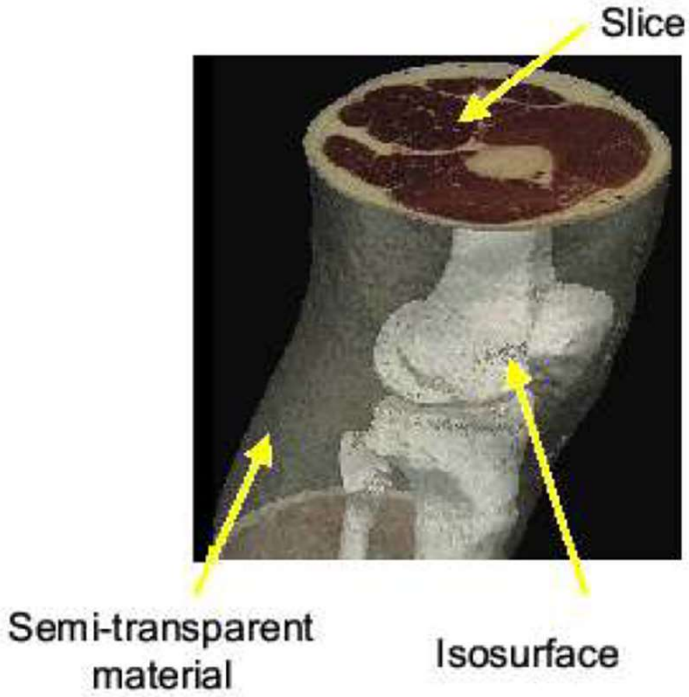
- Note
  - The most correct method in terms of physical realism may not be the most optimal one in terms of understanding the data.
  - To render and display 3D values, you always have to create 2D images, which involves a projection and a loss of data, because you throw away one dimension.

Volume Rendering Techniques - Examples



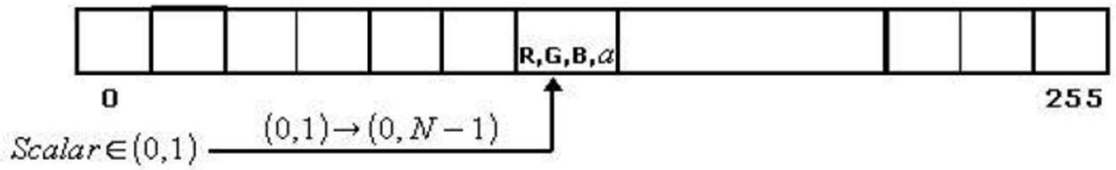
Techniques for 2D Scalar Fields

- Slicing:
  - Display the volume data, mapped to colors, on a slice plane
- Isosurfacing:
  - Isosurfaces are 2D surfaces that can be extracted from 3D (or higher dimensional)
  - Generate opaque/semi-opaque surfaces
- Transparency Effects:
  - Volume material attenuates reflected or emitted light

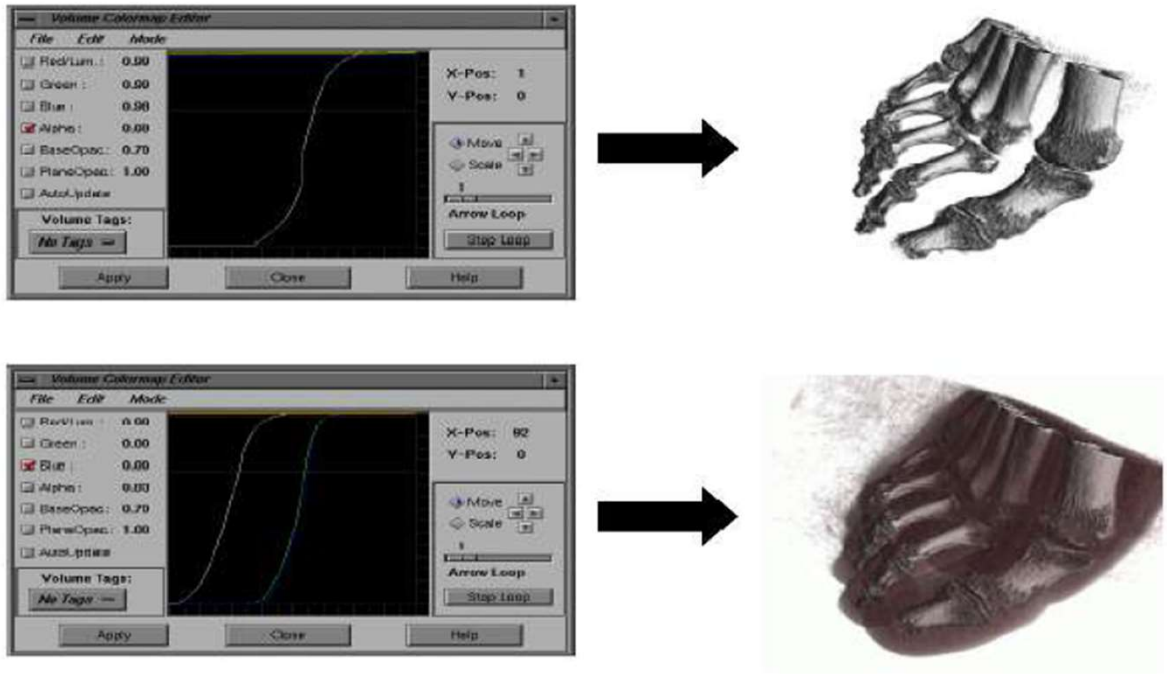


Classification

- Classification is an important process which assigns a material characteristic to each voxel, based on any of a wide variety of data characteristics, such as data value (scalar or vector), derivative measures, or local histograms.
- The so created material occupancy assignment is called a classification-, or transfer function.
- The transfer function describes the relationship between the input and the output of a system.
- Its role in volume rendering is to map the voxel information to renderable properties of opacity and color.
- It is often based on a color table that maps raw voxel value into presentable entities like color, intensity, opacity, etc.
- The most widely used approach for transfer functions is to assign each scalar value a different color value: *T : scalarvalue -> colorvalue* .
- A common choice for color representation is *R ,G , B ,* α , where α describes the opacity.
- A known problem of transfer functions is the so called **partial volume effect** which appears, when two or more substances mix in one voxel. In that case you cannot decide which material has to be represented.
- This problem can be solved with a good pre-classification, that means that each voxel has to be labeled with its associated material.



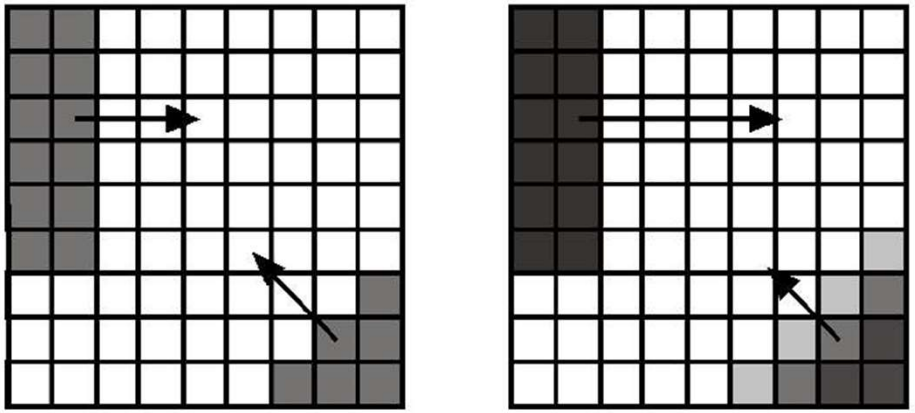
Coding scalar values into a color lookup table (LUT).



Interactive manipulation of transfer function with different results.

Gradients

- The general interest of volume visualization is not a particular isosurface but whole regions of change.
- This suggests a feature extraction with a high value of opacity in regions of change.
- Large homogeneous regions are less important than regions with
- strong structural changes.
- In order to emphasize changes it is useful to consider gradients of the scalar field, whereby the transfer function becomes two-dimensional.



The gradient of a pixel points to the direction of largest change. Transitions between same colors (e.g. same gray to white) result in same length of the gradient, no matter of its direction.

Segmentation

- Segmentation is a pre-processing method and needed for volume rendering to separate different objects from each other.
- Once the dataset is segmented, those quantities are easily measured.
- The difficult part of finding an accurate segmentation is that different
- materials can have the same scalar value
- Example: In a CT scan, different organs have similar X-ray absorption whereby a proper classification can not be distinguished.
- This is the reason why segmentation is mostly a semi-automatic or even manual process and requires expert knowledge.

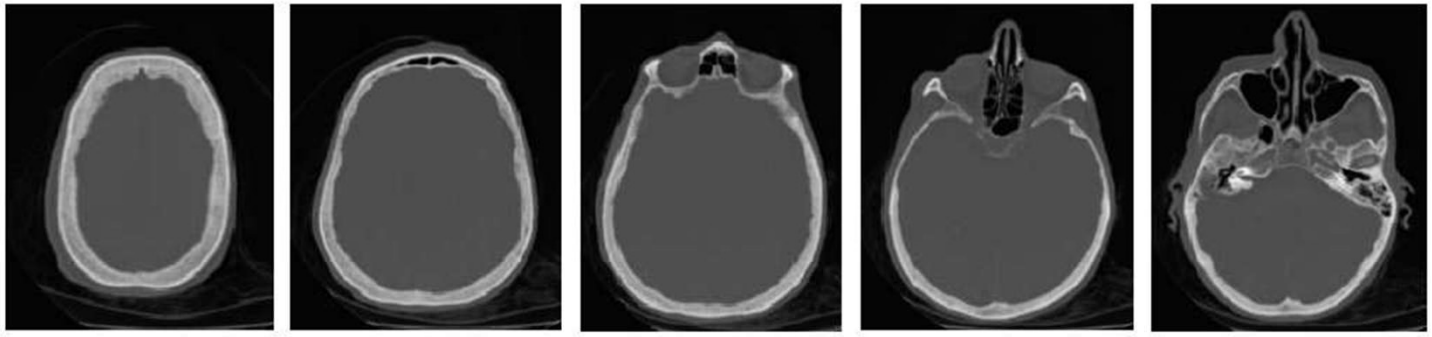
Volumetric Shading

- In general shading is used to visualize the 3D structure on a 2D plane.
- Without shading, different voxels of the same slice will have the same color after mapping by a transfer function.
- This leads to the fact that we notice the result as a plane 2D surface, although it is a 3D object.
- By shading this surface, one can pretend the human perception a 3D effect.
- With **volumetric shading** techniques, it is possible to create scenes with effects like fog or smoke by simulating the scattering and reflection of light as it passes through the atmosphere.
- This takes effect on the color of each voxel in the volume dataset.

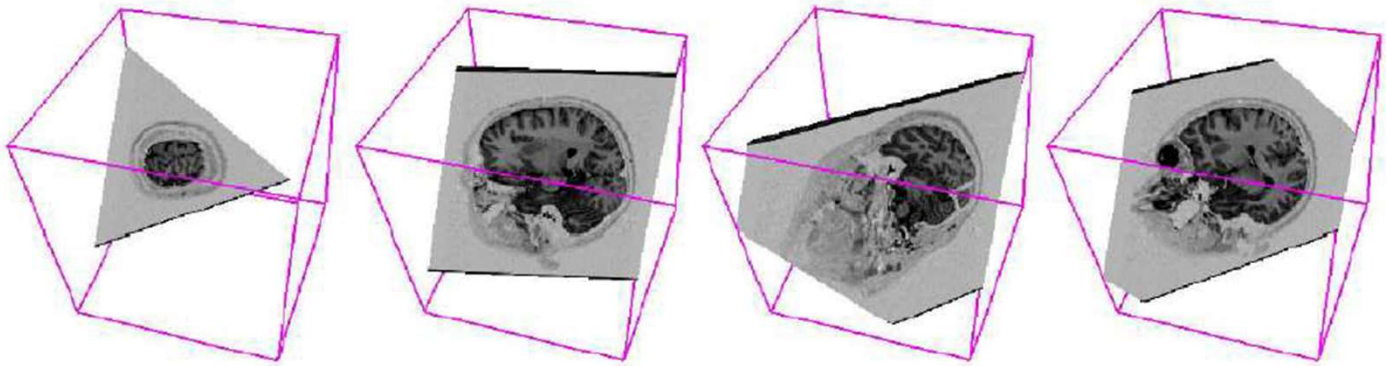
Slicing

- For indirect volume rendering there are two approaches, isosurfacing and slicing
- Slicing extract a subset of data and visualize the subset with traditional rendering techniques.
- Slicing can be divided again in two different procedures
  - Orthogonal
    - Interactively resample the data on slices perpendicular to the x-,y-,z-axis
    - Use visualization techniques for 2D scalar fields
      - Color coding
      - Isolines
      - Height fields
  - Oblique slicing:
    - Resample the data on arbitrarily oriented slices
    - Exploit 3D texture mapping functionality
      - Store volume in 3D texture
      - Compute sectional polygon (clip plane with volume bounding box)
      - Render textured polygon

Slicing



Result of orthogonal slicing.



Examples of oblique slicing.

Indirect Volume Rendering

- Convert/reduce volume data to an intermediate representation (surface model), which is efficiently manageable and representable
- It is assumed that coherent structures (e.g. skin, bone) are represented by point sets with the same sampling rate.
- The surfaces of these point sets are approximated by polygons. This often results in complex representations,
  - Pre-processing the surface representation might help
  - Use graphics hardware for interactive display
- Starts with the volume data and tries to find a triangle mesh, that represents the volume as well as possible.
- Once the 2D mesh is found, it can be rendered with traditional techniques

Indirect Volume Rendering

- Indirect volume rendering techniques first transfer the volume dataset into a new domain, before it is rendered.
- These algorithms are often chosen because of their speed advantage although they are not so precise.
- The general idea of those techniques is, if $f(x, y, z)$ is differentiable in every point, then the level-sets $f(x, y, z) = c$ are isosurfaces to the defined isovalue c.
- That means that the algorithm goes through all voxels and determines if each voxel belongs to the isosurface with value c.

Marching Cube

- Marching Cube is a common indirect volume rendering techniques to determine and reconstruct isosurfaces from volume data
- The applications of this algorithm are mainly concerned with medical visualizations such as CT and MRI scan data images.
- The algorithm works on the original volume data, it defines a voxel (cube) by the pixel values at the eight corners of the cube.
- This cube is “marching” through the whole volume dataset and subdivides space into a series of cubes.
- At each step we classify each vertex of the cube as inside or outside the isosurface.
- Edges that are adjacent to one “inside” and one “outside” classified vertex are intersected by the isosurface
- We can create a triangle patch whose vertices are found by linear interpolation along those cell edges.
- Further we use the gradients as normals of the triangle surfaces.
- By connecting the patches from every step of the cube we get an approximated isosurface represented by a triangle mesh.
- We gain efficient computation by means of lookup table that stores all
- possible constellations of triangle patches.

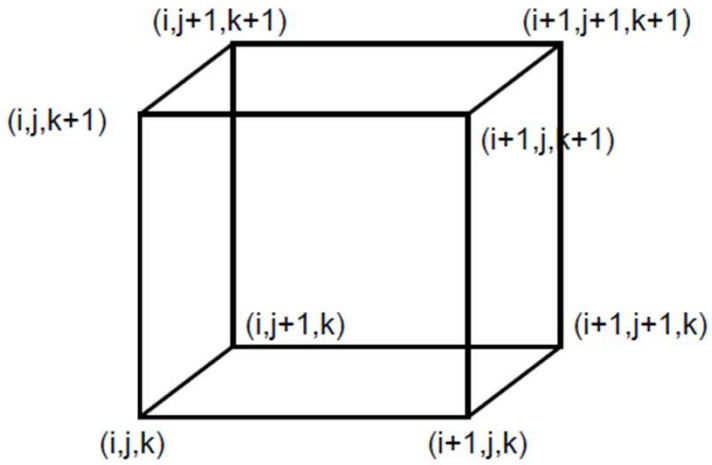
Marching Cube - Steps

1. Consider a cell
2. Classify each vertex as inside or outside
3. Build an index
4. Get edge list from table[index] for triangulation
5. Interpolate the edge location
6. Compute gradients
7. Consider ambiguous cases
8. Go to next cell

- Cell consists of 4 pixel/8 voxel values
- (i + [01] , j+[01] , k+[01])

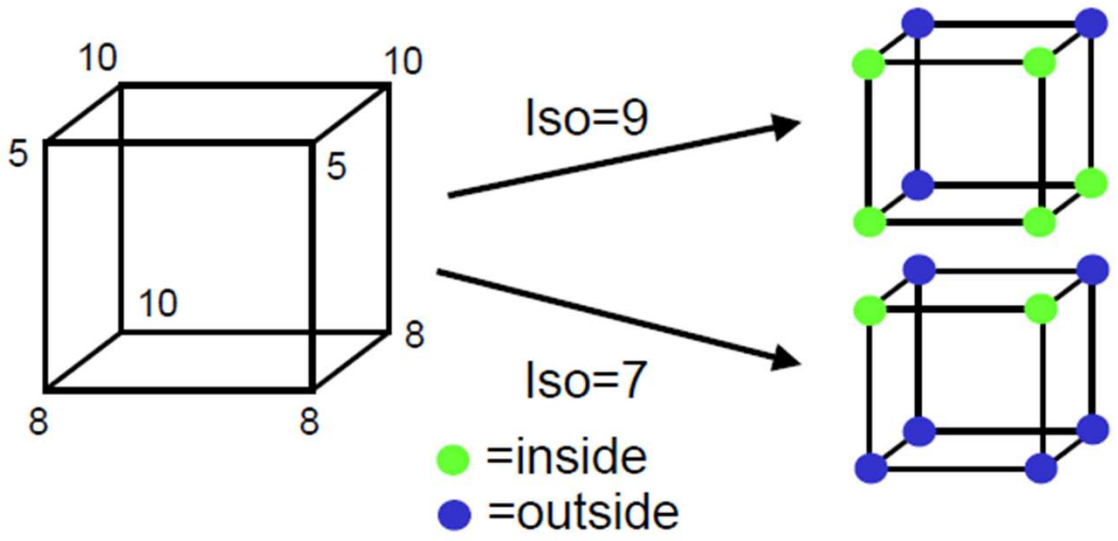
Step 1

Step 1: Consider a cell defined by eight data values



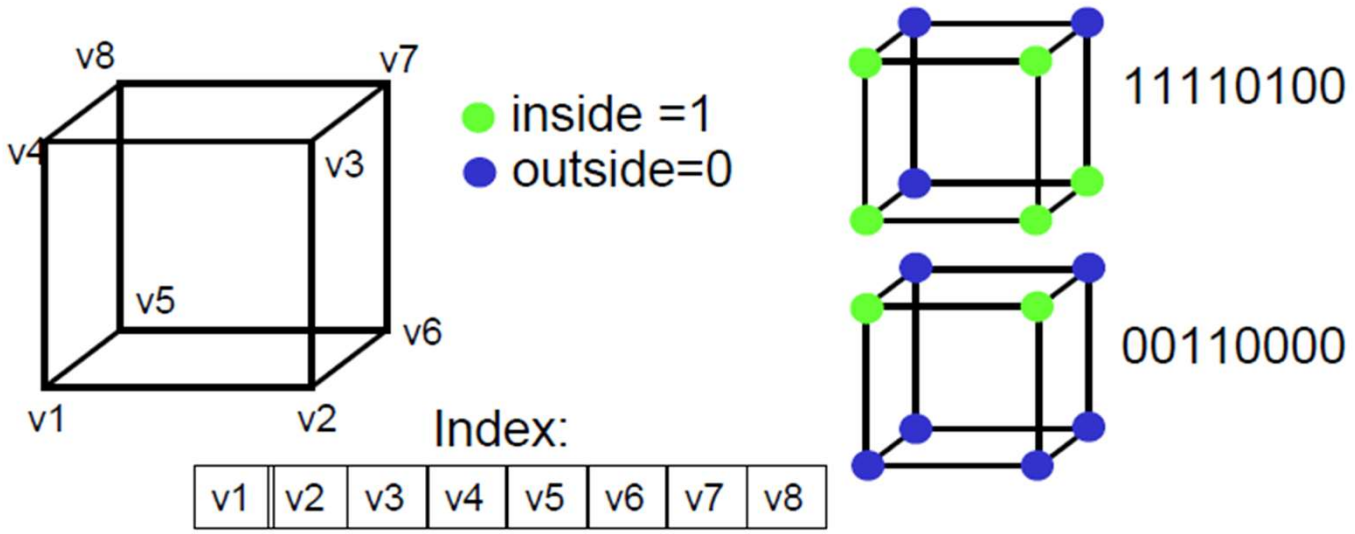
Step 2

- Step 2: Classify each voxel according to whether it lies
  - outside the surface (value > isosurface value)
  - inside the surface (value <= isosurface value



Step 3

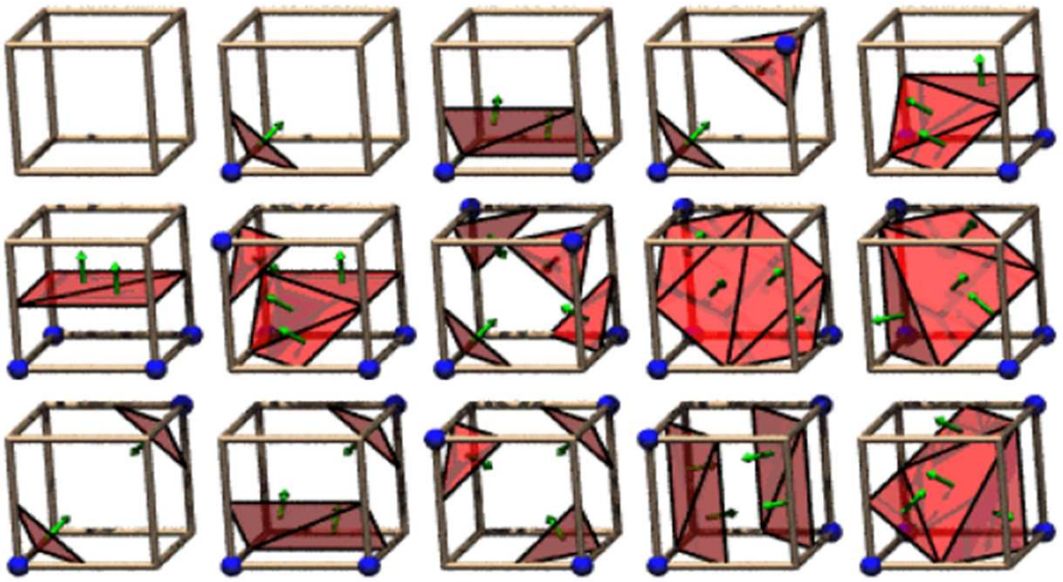
- Step 3: Use the binary labeling of each voxel to create an index



Step 4

Step 4: For a given index, access an array storing a list of edges

- All 256 cases can be derived from 15 base cases due to symmetries

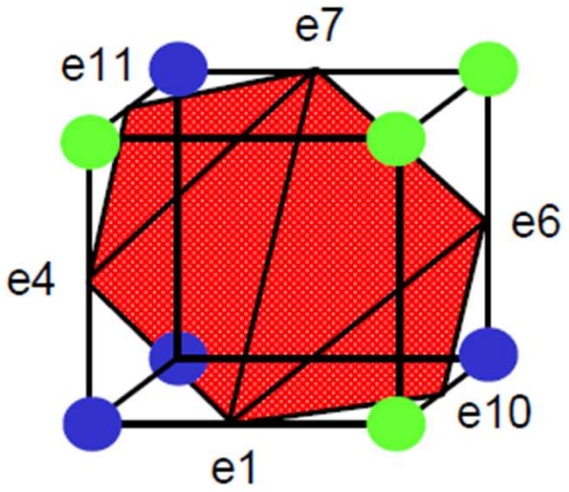


The 15 Cube Combinations

Step 4 cont

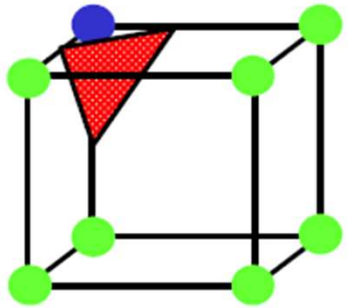
- Step 4 *cont.*: Get edge list from table
  - Example for

    Index = 10110001
    triangle 1 = e4,e7,e11
    triangle 2 = e1, e7, e4
    triangle 3 = e1, e6, e7
    triangle 4 = e1, e10, e6



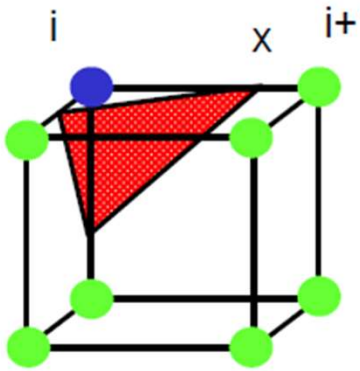
Step 5

Step 5: For each triangle edge, find the vertex location along the edge using linear interpolation of the voxel values



T=5

=10
=0

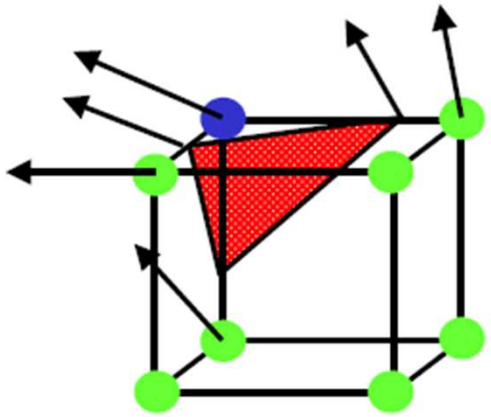


T=8

$$x = i + \left(\frac{T - v[i]}{v[i+1] - v[i]} \right)$$

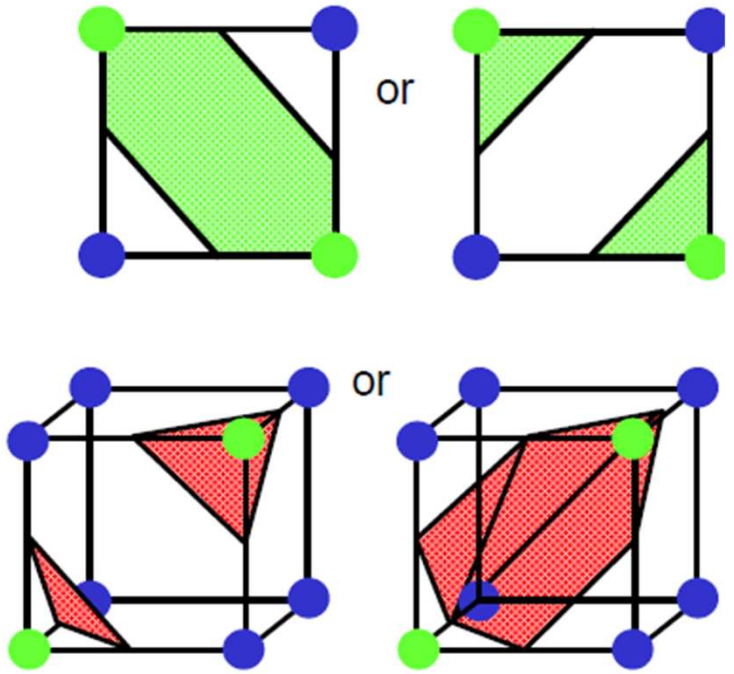
Step 6

- Step 6: Calculate the normal at each cube vertex
  - $G_x = V_{x+1,y,z} - V_{x-1,y,z}$
    $G_y = V_{x,y+1,z} - V_{x,y-1,z}$
    $G_z = V_{x,y,z+1} - V_{x,y,z-1}$
  - Normalize
  - Use linear interpolation to compute the polygon vertex normal (of the isosurface)



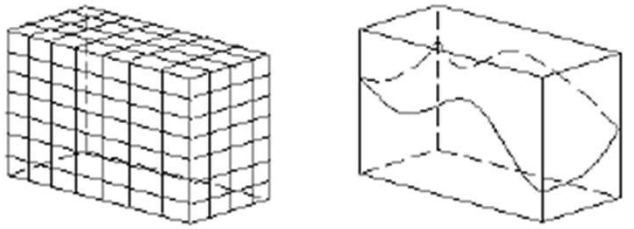
Step 7

- Step 7: Consider ambiguous cases
  - Ambiguous cases: 3, 6, 7, 10, 12, 13
  - Adjacent vertices: different states
  - Diagonal vertices: same state
  - Resolution: decide for one case

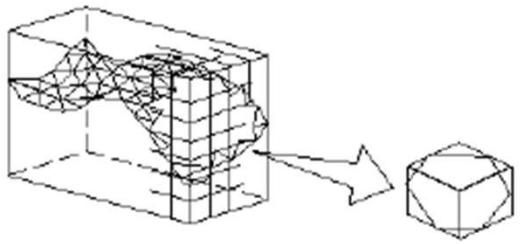


Marching Cube - Summary

- 256 Cases
- Reduce to 15 cases by symmetry
- Ambiguity resides in cases 3, 6, 7, 10, 12, 13
- Causes holes if arbitrary choices are made
- Up to 5 triangles per cube
- Dataset of 512^3 voxels can result in several million triangles (many Mbytes!!!)
- Semi-transparent representation --> sorting
- Optimization:
  - Reuse intermediate results
  - Prevent vertex replication
  - Mesh simplification



(a) Volume data (b) Isosurface S = f(x,y,z)



(c) Polygonal Approximation

Contour Propagation

- Acceleration of cell traversal
- Algorithm:
  - Trace isosurface starting at a seed cell from which one knows it contains an isosurface
  - Breadth-first traversal along adjacent faces but avoid visiting empty cells
  - Finally, cycles are removed, based on marks at already traversed cells
- Problem:
  - Isosurfaces can consist of several not connected components, and for every one a seed cell must be found or whole parts of the isosurface will be lost.
  - Issue of optimal seed set

Direct Volume Rendering

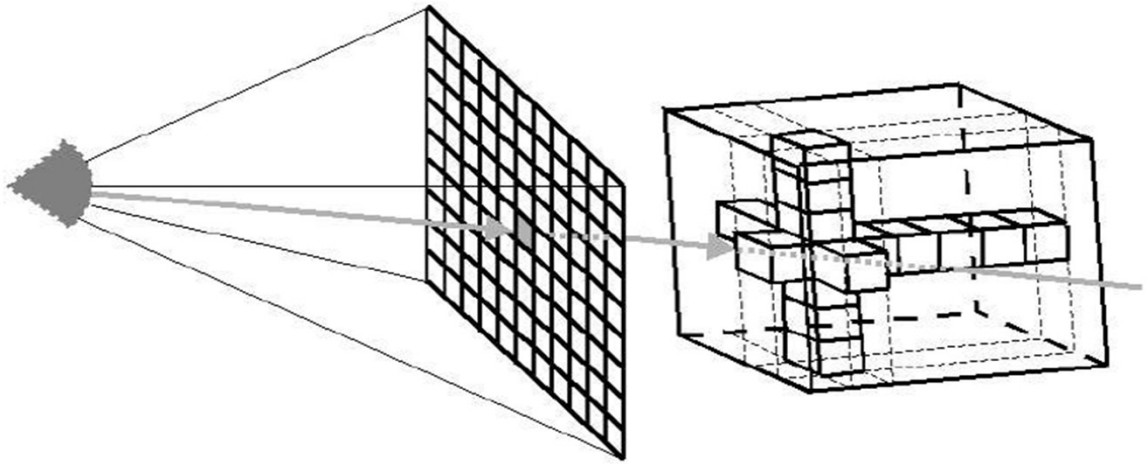
- Direct volume rendering is the direct mapping of voxels on pixels of a 2D image plane
- Direct volume rendering allows for the "global" representation integrating physical characteristics
- Modern Graphics Hardware allows interactive display due to its numerical complexity, in general
- In practice the data can be considered as a semi-transparent gel and the user decides which parts of the object should be opaque or transparent.
- The final 2D image is computed by projecting, in visibility order, the voxels onto the image plane, and incrementally compositing the voxel's color and opacity into the final pixel.

Direct Volume Rendering

- Direct Volume Rendering (DVR) is to get a 3D representation of the volume data directly.
- The data is considered to represent a semi-transparent light-emitting medium.
  - Gaseous phenomena can be simulated.
- The approaches in DVR are based on the laws of physics (emission, absorption, scattering).
- The volume data is used as a whole
  - We can look inside the volume and see all interior structures.
- In DVR either backward or forward methods can be used

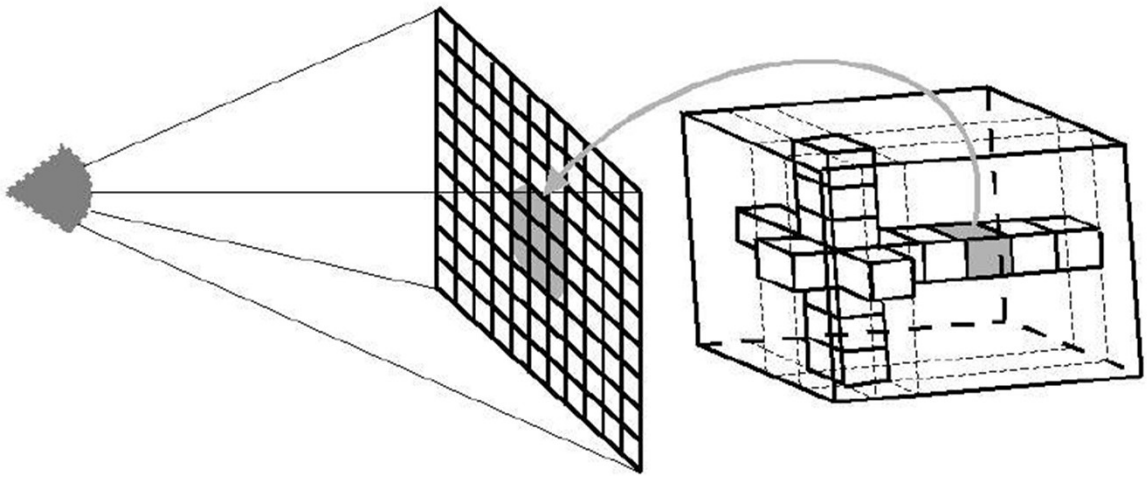
Backward Methods

- Backward methods use image space/image order algorithms.
- They are performed pixel by pixel.
- Example: *Ray Casting*



Forward Methods

- Forward methods use object space/object order algorithms.
- These algorithms are performed voxel by voxel and the cells are projected onto the image.
- Examples: Texture-based Volume Rendering



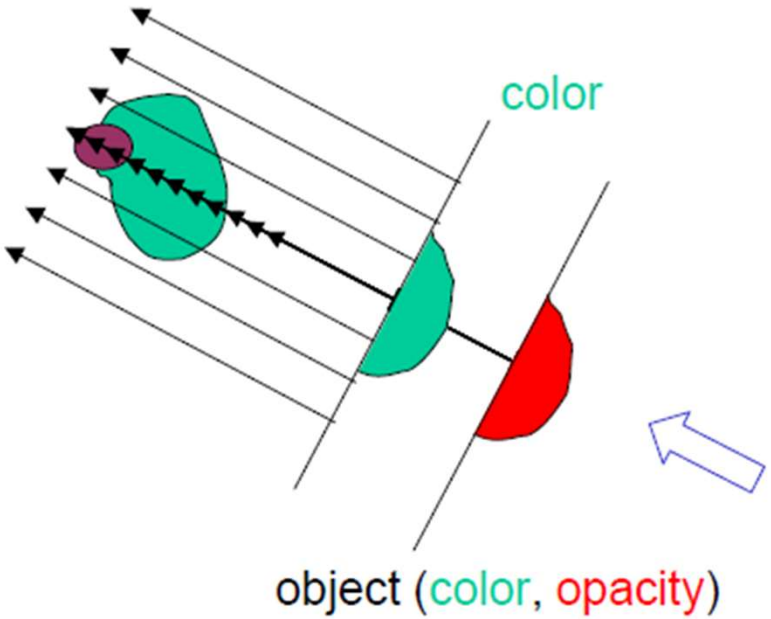
Ray Casting

- Ray Casting is a natural image order technique.
- Since we have no surfaces - we need to carefully step through the volume
- A ray is cast into the volume, sampling the volume at certain intervals.
- The sampling intervals are usually equidistant, but they don't have to be (e.g. importance sampling).
- At each sampling location, a sample is interpolated/reconstructed from the voxel grid.

Volumetric Ray Integration

- The rays are casted through the volume. Color and opacity are accumulated along each ray (compositing).
- Opacity and (emissive) color in each cell according to classification
- Additional color due to external lighting: according to volumetric shading

- Volumetric ray integration:
  - Tracing of rays
  - Accumulation of color and opacity along ray: compositing



Acceleration Techniques for Ray Casting

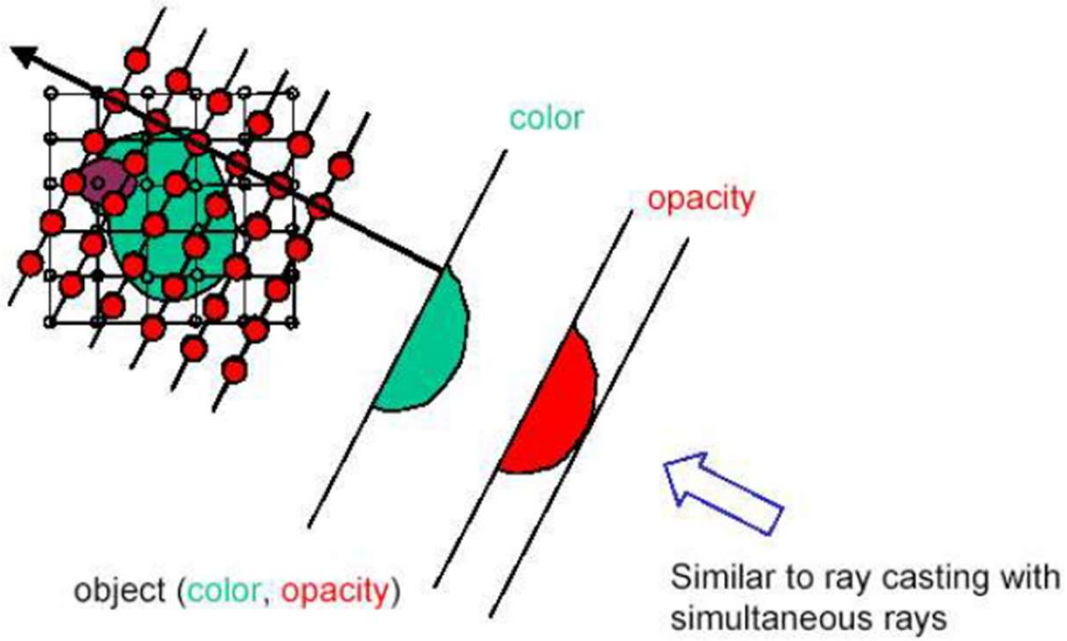
- Problem: ray casting is time consuming
- Strategies for Acceleration
  - Neglect irrelevant information to accelerate the rendering process
  - Exploit coherence

  - Early-ray Termination
    - Idea: Colors from faraway regions do not contribute if accumulated opacity is to high
    - Stop traversal if contribution of sample becomes irrelevant
    - User-set opacity level for termination
  - Space Leaping
    - Fast traversal of regions with homogeneous structure
    - Skip empty cells
    - Homogeneity-acceleration: Approximate homogeneous regions with fewer sample points

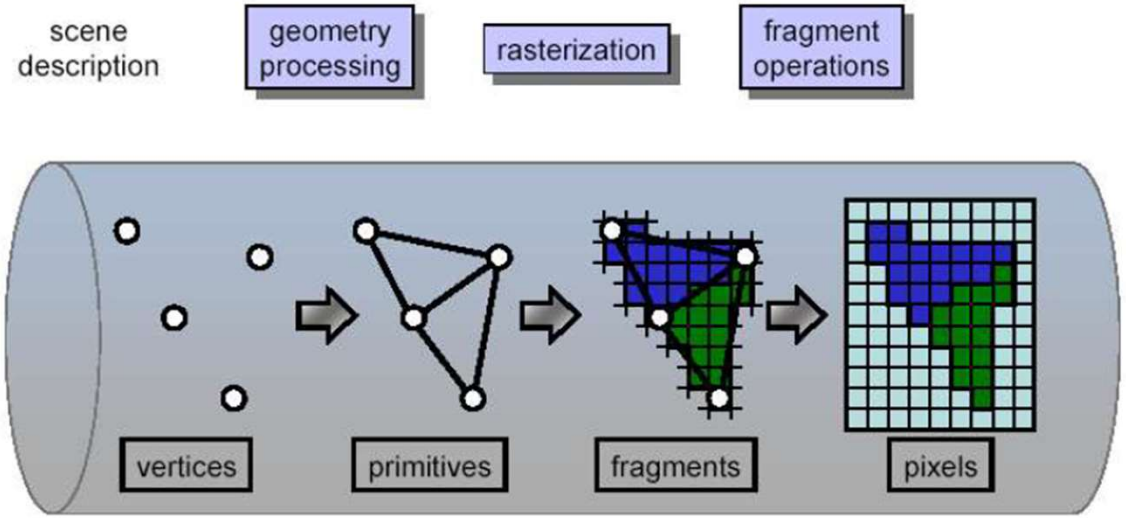
Texture-Based Volume Rendering

- **Texture-based Volume Rendering** is an approach to generate an image viewed with different camera parameters using scene information extracted from a set of reference images.
- The rendering speed of this approach depends only on image size instead of scene complexity, and complex hand-made geometric models are not required.
- Takes advantage of functionality implemented on graphics hardware like Rasterization, Texturing and Blending.
- Texture-based volume rendering is an object-space technique, since all calculations are done in object-space.
- The rendering is accomplished by projecting each element onto the viewing plane such as to approximate the visual stimulus of viewing the element with regard to the chosen optical model.

Texture-Based Volume Rendering



Rendering Pipeline

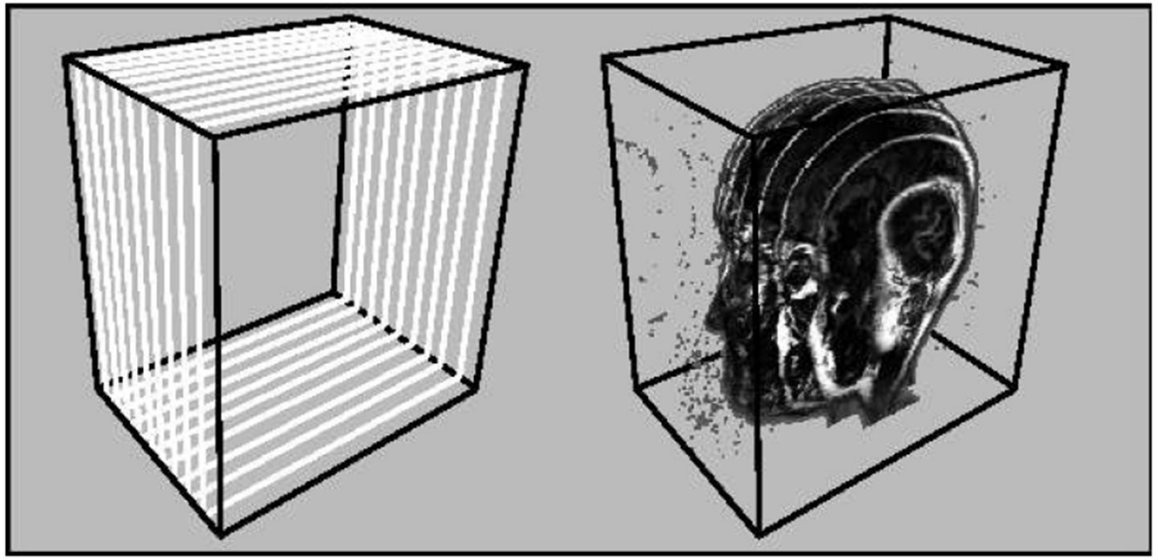


Rasterization is the task of taking an image described in a vector graphics format (shapes) and converting it into a raster image (a series of pixels, dots or lines, which, when displayed together, create the image which was represented via shapes)

2D Textures

- Each element gets projected on a slice-by-slice basis.
- Therefore, the cross sections of each element and the current slicing plane are computed.
- The computed slice has only a proxy geometry because there are no volumetric primitives in graphics hardware.
- By sending the computed vertices of each slice down the rendering pipeline, geometric primitives are generated that get fragmented in the rasterization step to finally result in a pixel based texture map.
- All texture-mapped slices are stored in a stack and are blended back-to-front onto the image plane, which results in a semitransparent view of the volume.
- The texture mapped 2D slices are object-aligned, because they get computed in object space parallel to the x-, y-, z-axis, unappreciated from which angle the viewer looks at the object.
- Due to correct visualization of the object from each possible viewing angle three stacks of 2D texture slices have to be computed. By doing this one can switch between the different stacks in case the viewing angle is close to 45 degrees to the slice normal.
- Otherwise the viewer would look on the slice edges and thus see through the volume.
- By projecting the voxels of the slice plane on pixels of the viewing plane, a pixel can lie in between voxels, so its value can be found by bilinear interpolation.

2D Textures

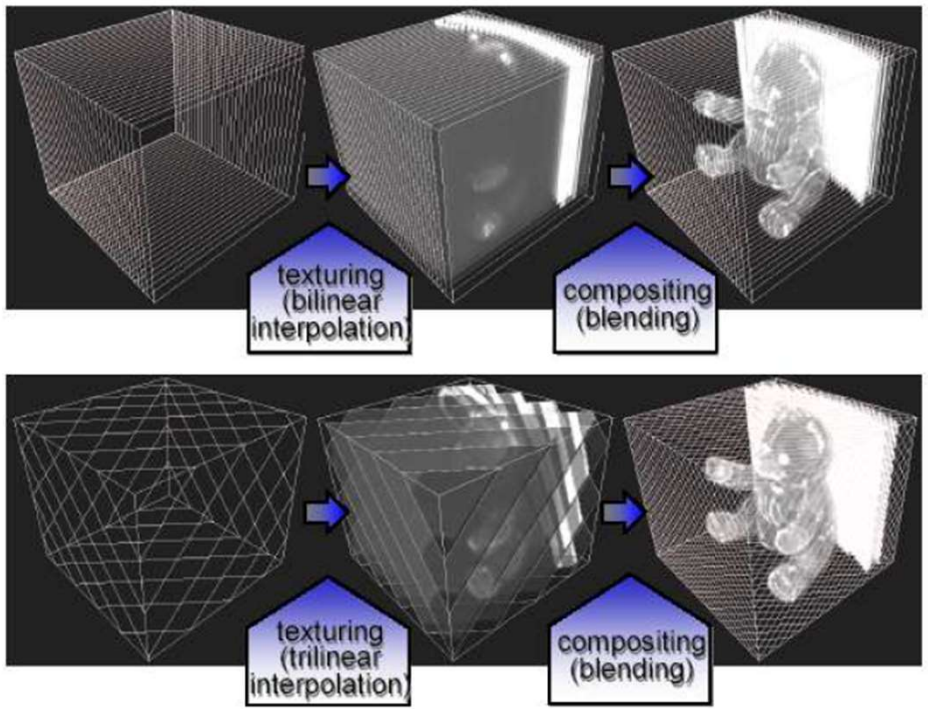


A stack of 2D textured slices and the proxy geometry that could be achieved with it.

3D Textures

- Another method is to compute view-aligned 3D textured slices.
- In contrast to the object aligned method, these slices are arranged with their surface normals in direction of the viewer.
- Here we only have to compute one stack of 3D textures, but for projection trilinear interpolation is needed to calculate values in-between.
- Further, if the object is rotated the slices have to be recomputed for each frame.

2D vs 3D Textures



2D textures
axis-aligned

3D texture
view-aligned

Overview

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

Introduction

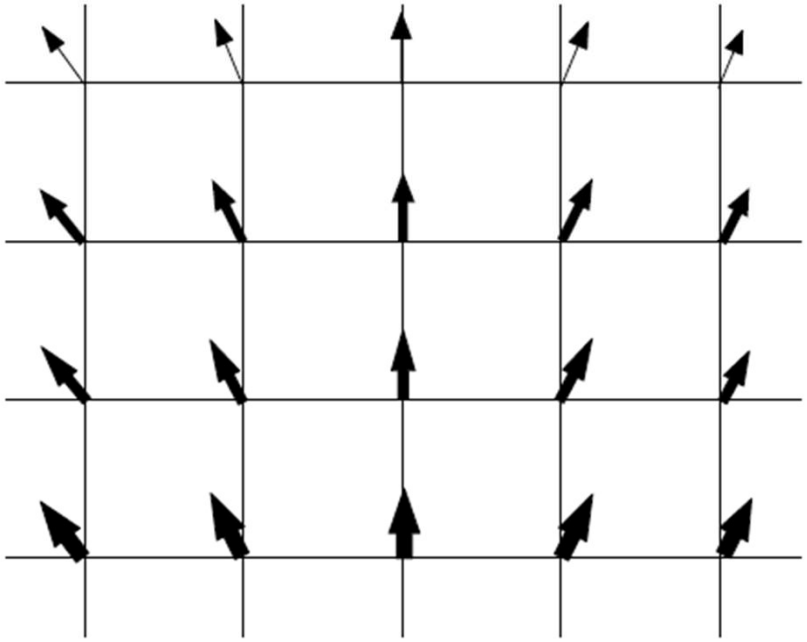
- The need to visualize vector fields in two or three dimensions is common in many scientific disciplines, as well as it is necessary in some industrial branches.
- Examples of vector fields include:
  - Velocities of wind and ocean currents
  - Results of fluid dynamics solutions, magnetic fields, blood flow
  - Components of stress and strain in materials

Vectors

- A vector is a mathematical object having the properties of magnitude and direction
  - Usually pictorially represented by a line segment with an arrowhead.
- In 3D modeling a vector is often given a geometric interpretation, for example, position or displacement.
- Vector algebra and calculus are effective tools for creating and manipulating certain kinds of models.
- Vector field: A region of space under the influence of some vector quantity, such as magnetic field strength
  - In which each point can be described by a vector.
- In a 3D vector data set each point is represented by a vector that stands for a direction and magnitude.
- The vector field is given by a n−tuple($f_1,\ldots,f_n$) with $f_k = f_k(x_1,\ldots x_n)$
  - $n<=2 \wedge 1<=k<=n$ and specific transformation properties.
  - Typically the dimension is n=k=2 or n=k=3 .

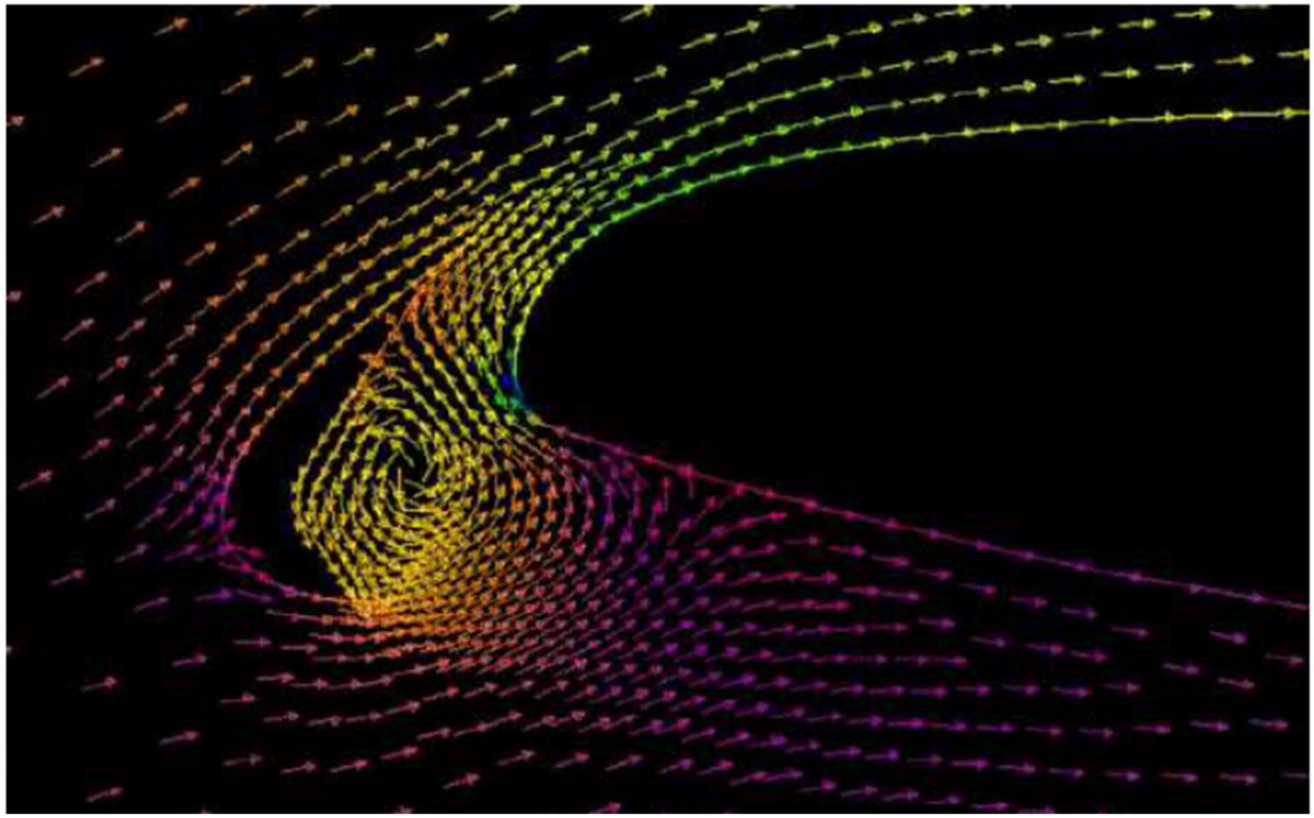
Graphical Primitives - Arrows

- Arrows visualize
  - Direction of vector field
  - Orientation
  - Magnitude:
    - Length of arrows
    - Color coding



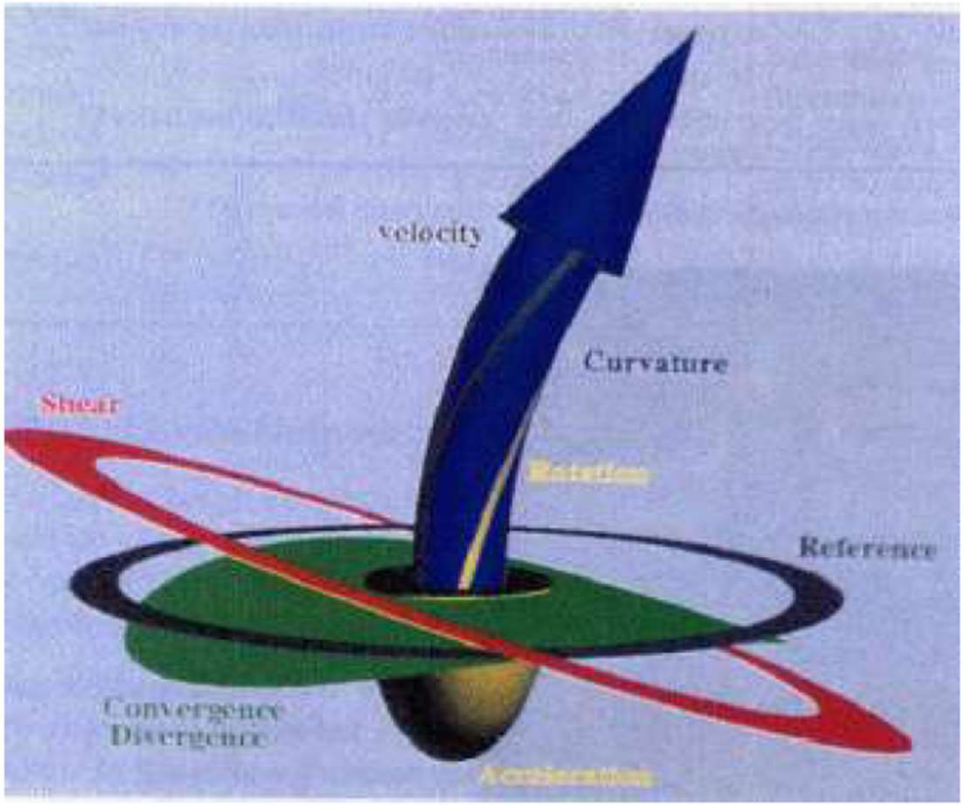
Example - Arrows

Visualization of a vector field by simple arrows.



Graphical Primitives - Glyphs

With glyphs we can visualize more features of the vector field (flow field)
However one has to learn how to read the symbol



A glyph with 7 dimensions

Flow Visualization

- One of the main application of vector field visualization is flow visualization
- **Flow visualization** is used to make the flow patterns visible, in order to get qualitative or quantitative information on them.

- Steady (time-independent) flow
  - Static over time
- Unsteady (time-dependent) flow
  - Flow itself changes over time

- Direct flow visualization:
  - Overview of current flow state
  - Visualization of vectors
- Indirect flow visualization:
  - Use intermediate representation: vector-field integration over time
  - Visualization of temporal evolution
  - Integral curves, integral surfaces

Flow Visualization - Attributes

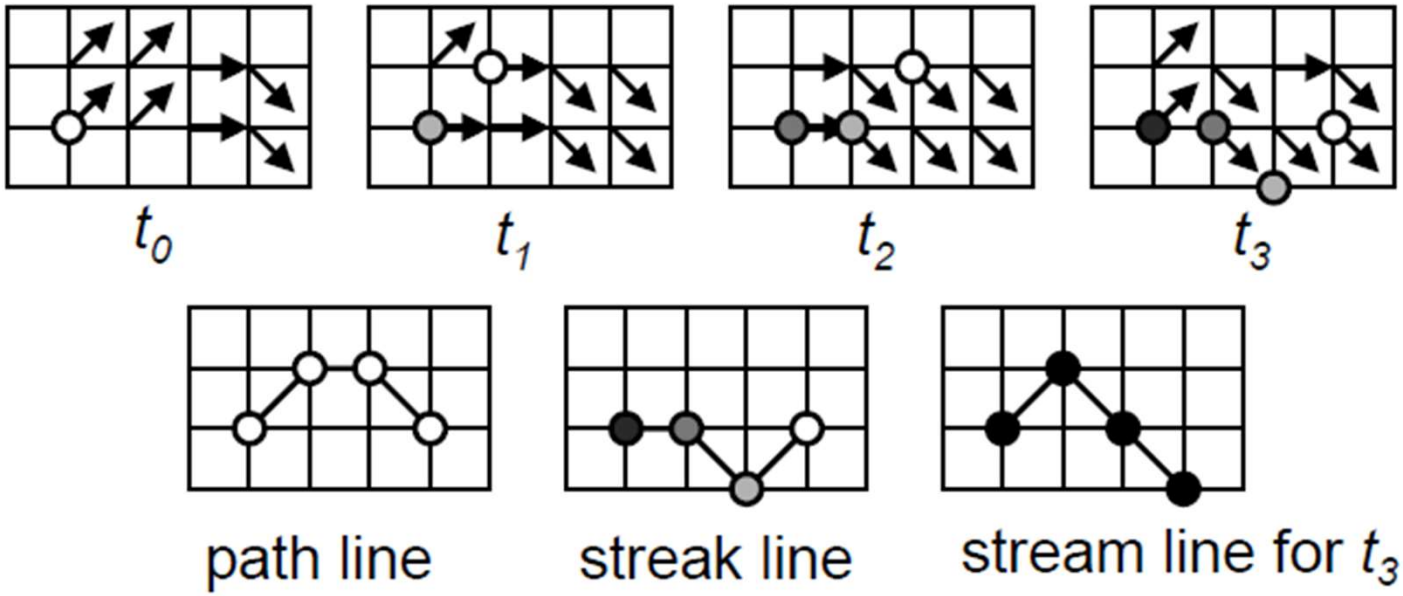
- Motion of fluids (gas, liquids)
- Geometric boundary conditions
- Velocity (flow) field $v(x, t)$
- Pressure p
- Temperature T
- Density ρ
- Vorticity: A pseudo-vector field that describes the local spinning motion of a continuum near some point (the tendency of something to rotate), as would be seen by an observer located at that point and traveling along with the flow
- Conservation of mass, energy, and momentum
- Navier-Stokes equations (P**artial differential equations which describe the motion of viscous fluid substances)**
- CFD (Computational Fluid Dynamics) *data based flow visualization with lines, icons and glyphs.*

Field Lines

- There are various field lines in a fluid flow. They differ only when the flow changes with time, that is, when the flow is not steady.
- Considering a velocity vector field in three-dimensional space in the framework of continuum mechanics, we have 4 types of field lines:
  - Stream lines: Generated by tracing the direction of flow at one single time step
  - Path lines: Generated by tracing the path of a single particle in continuous time.
  - Streak lines: Generated by continuously injecting particles from a fixed location.
  - Time lines: Generated by tracing a line of particles which are all released at the same time

Comparison of Characteristic Lines

- Comparison of path lines, streak lines, and stream lines



- Path lines, streak lines, and stream lines are identical for steady flows

Stream Lines

- Stream lines are the tangential curves of the vector field V
- Generated by tracing the direction of flow at one single time step
- For every time and every location there is one and only one stream line through it
- Curve parallel to the vector field in each point for a fixed time
- Describes motion of a massless particle in a steady flow field
- Stream line is a solution to the initial value problem of an ordinary differential equation

$$\boldsymbol{L}(0) = \boldsymbol{x}_0 \quad , \quad \frac{d\boldsymbol{L}(u)}{du} = \boldsymbol{v}(\boldsymbol{L}(u), t)$$

initial value (seed point $\mathbf{x}_0$)

ordinary differential equation

- The stream line for one point is the curve L(u) with the parameter
- If we want the stream lines for the whole domain, we have to integrate the differential equation.

- The problem of finding a function y of x when we know its derivative and its value y0 at a particular point x 0 is called an initial value problem.

Path Lines

- Path lines are obtained by setting out a particle and tracing its path in the unsteady flow.
- Generated by tracing the path of a single particle in continuous time
- Describes motion of a massless particle over the time in an unsteady time-dependent flow field
- Curve parallel to the vector field in each point over time
- Therefore, path lines are projections of the tangent curves of V into a plane t=const.
- For every location and every time there is one and only one path line through it
- Path line is a solution to the initial value problem of an ordinary differential equation:

$$L(0)=x_0 \quad , \quad \frac{dL(u)}{du}=v(L(u),u) \quad \text{where}$$

L = curve of the stream line

$L(0)$ = initial value with seed point x_0

$\frac{dL(u)}{du}$ = tension direction of the curve is the first derivative (tangential)

$v(L(u),u)$ = if we move along the path, we have to take the new velocity at that point and time.

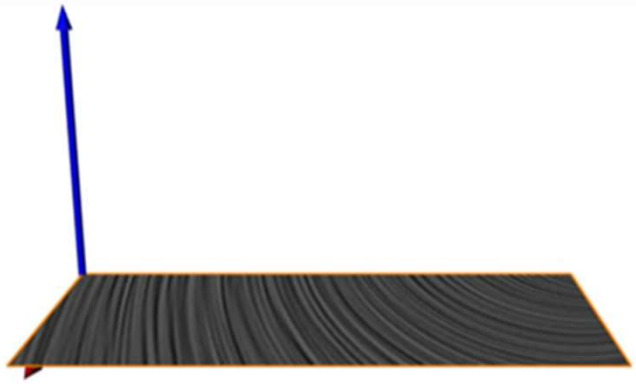
Streak Lines

- Location of all particles set out at a fixed point at different times
- A streak line is the connection of all particles set out at one seed point at different time steps.
- It can be imagined the trace of dye, that is released into the flow at a fixed point.
- Generated by continuously injecting particles from a fixed location.
- A streak line through a certain point *u* at time step *t*1 is not uniquely defined.
  - Another choice of *t*0 might lead to another streak line through *u* at *t*1

Time Lines

- Time lines are obtained by setting out particles located on a straight line at a fixed time and tracing them in the unsteady flow.
- The result is the propagation of a line (surface) of massless elements in time.
- It shows the location of all particles set out on a certain line at a fixed time
- Propagation of a line (surface) of massless elements in time
- Generated by tracing a line of particles which are all released at the same time
- Consists of many point-like particles that are traced

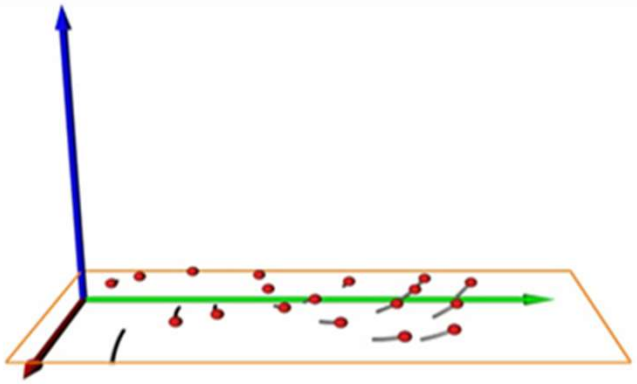
Stream Lines vs Path Lines



stream lines

curve parallel to the vector field in each point for a **fixed time**

describes motion of a massless particle in an **steady** flow field



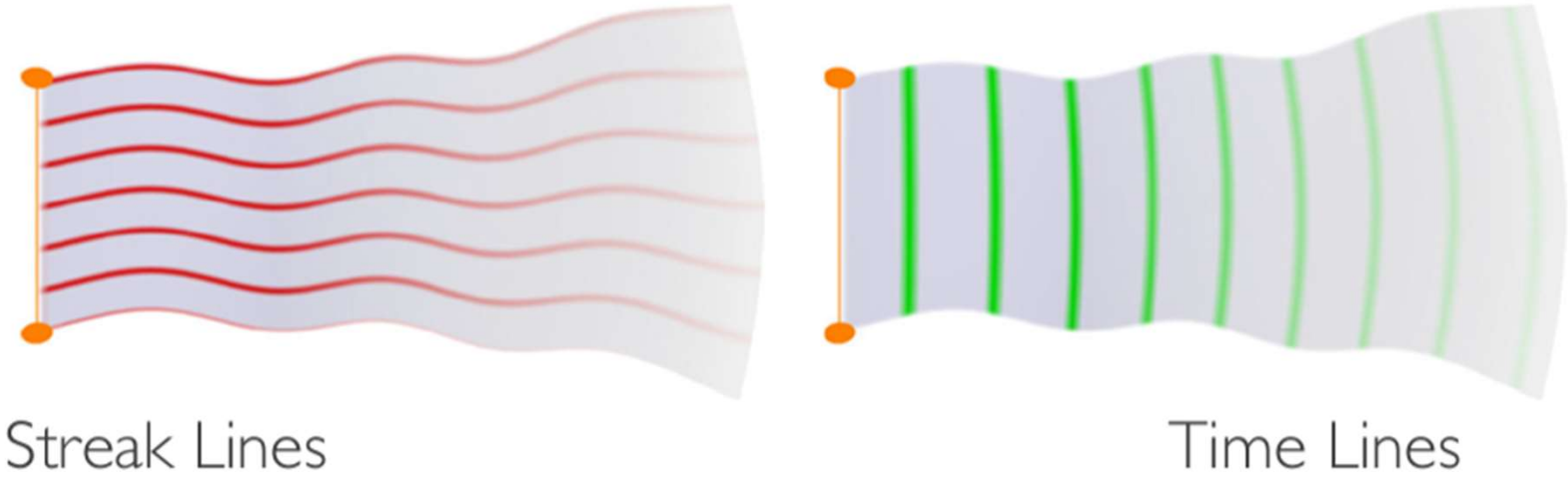
path lines

curve parallel to the vector field in each point **over time**

describes motion of a massless particle in an **unsteady** flow field

Streak Lines vs Time Lines

- (on a streak surface)



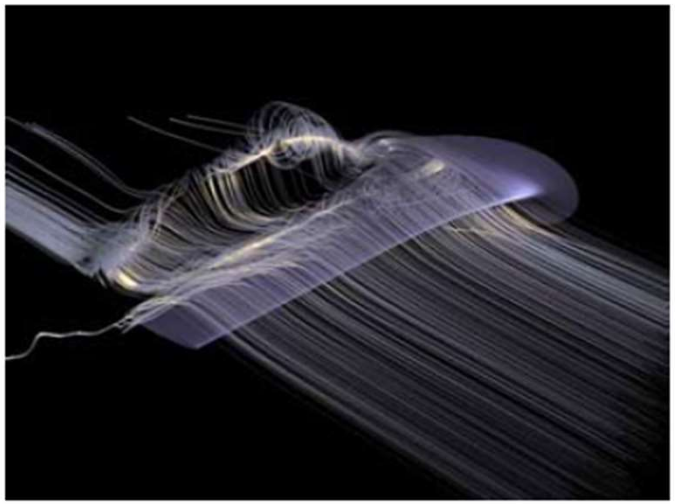
Streaklines are the loci of points of all the fluid particles that have passed continuously through a particular spatial point in the past. Dye steadily injected into the fluid at a fixed point extends along a streakline.

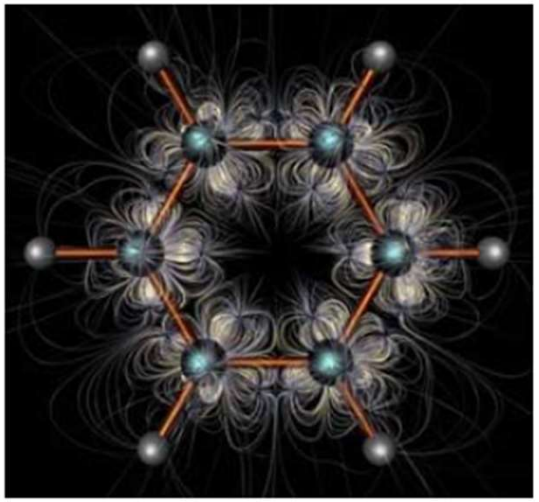
Timelines are the lines formed by a set of fluid particles that were marked at a previous instant in time, creating a line or a curve that is displaced in time as the particles move.

Particle Tracing

- Moving particles give a good indication of the direction and magnitude of the velocity.
- Particle tracing methods try to trace particles and visualize them effectively with characteristic lines.

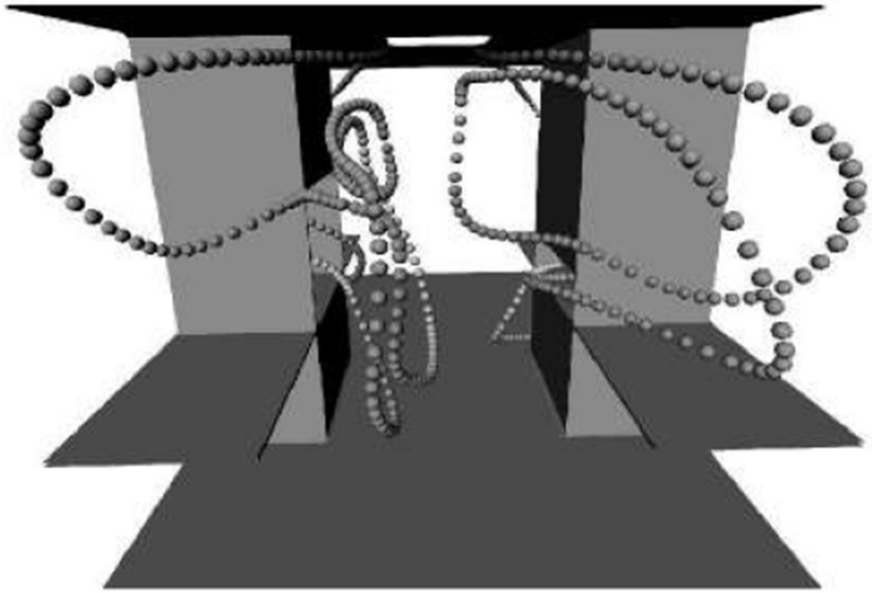
Path Lines





Trajectories of individual particles released in the flow

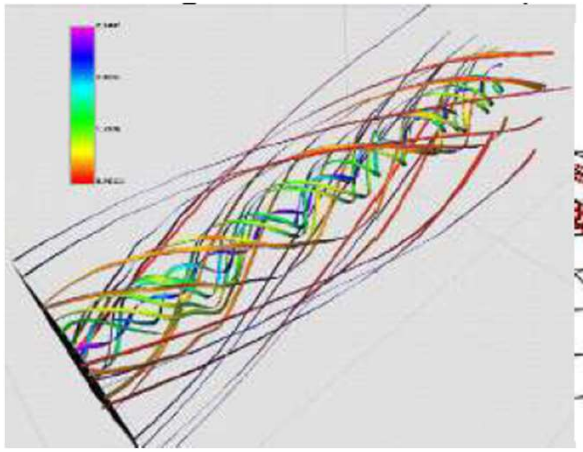
Stream Balls

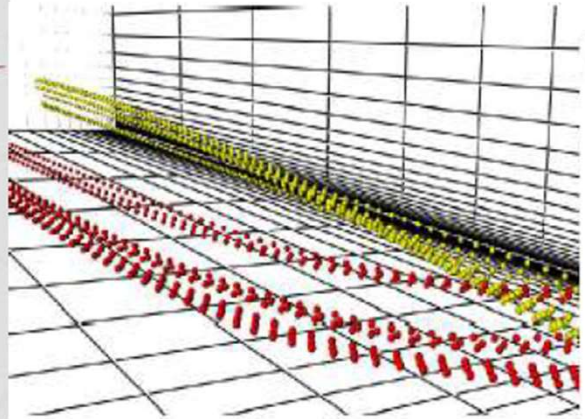


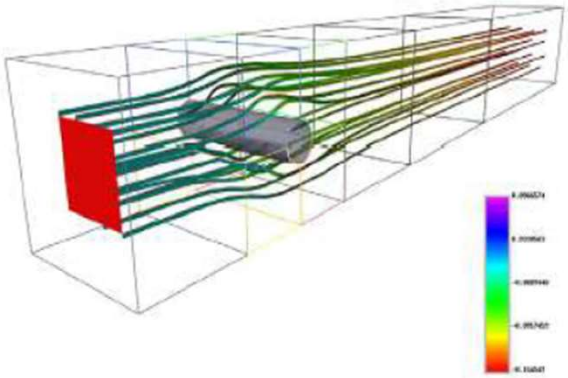
The stream balls represent the air flow in a ventilated room

Encode an additional scalar value by radius.
If the balls are far apart, velocity is high, if they are close to each other, velocity is small

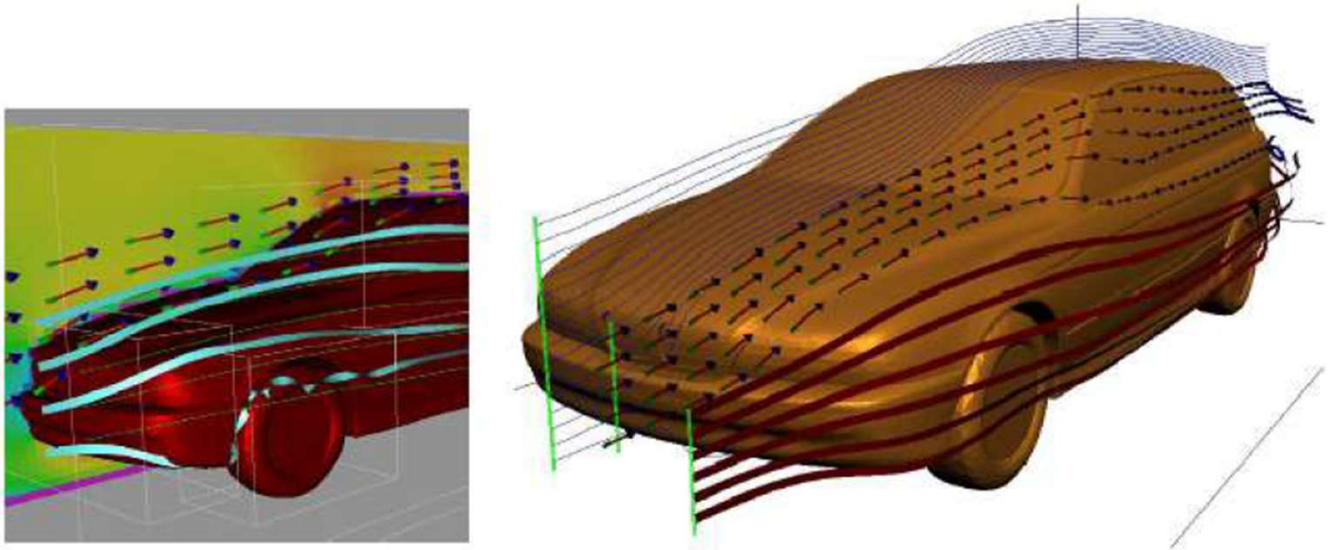
Streak Lines





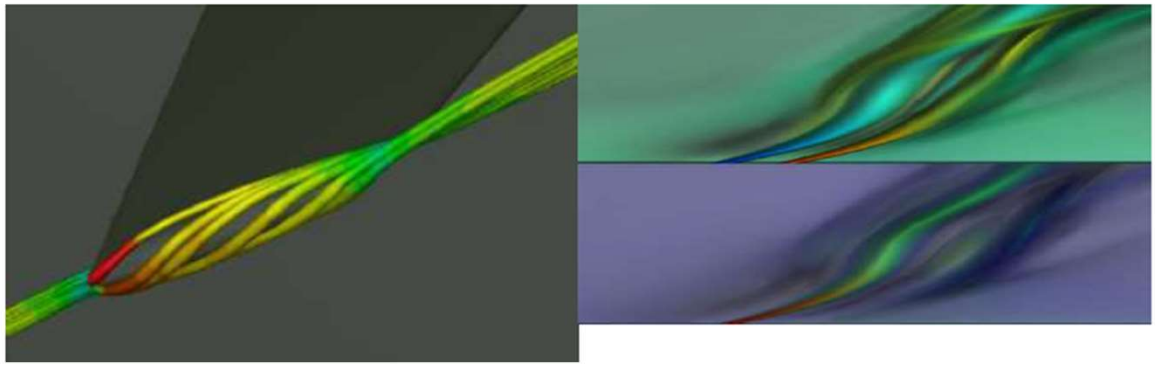


Stream Ribbons



- The concept of a streamline can be readily extended to a surface.
- If the stream lines of two close-by traced particles are connected by a mesh of polygons, a stream ribbon results
- (Note that we have to keep the distance constant).
- Alternatively, a ribbon can reflect local rotation (vorticity) of the flow by using the vorticity vector to orient the ribbon at each step.
- *With stream ribbons vorticity can be displayed effectively*

Stream Tubes



Stream tubes are the extension of stream lines to a 3D tube. We specify a contour, e.g. a triangle or circle, and trace it through the flow.

Readings

- Gregory M. Nielson, Hans Hagen, Heinrich Müller. Scientific Visualization: Overviews, Methodologies and Techniques. IEEE Computer Society Press, 1997