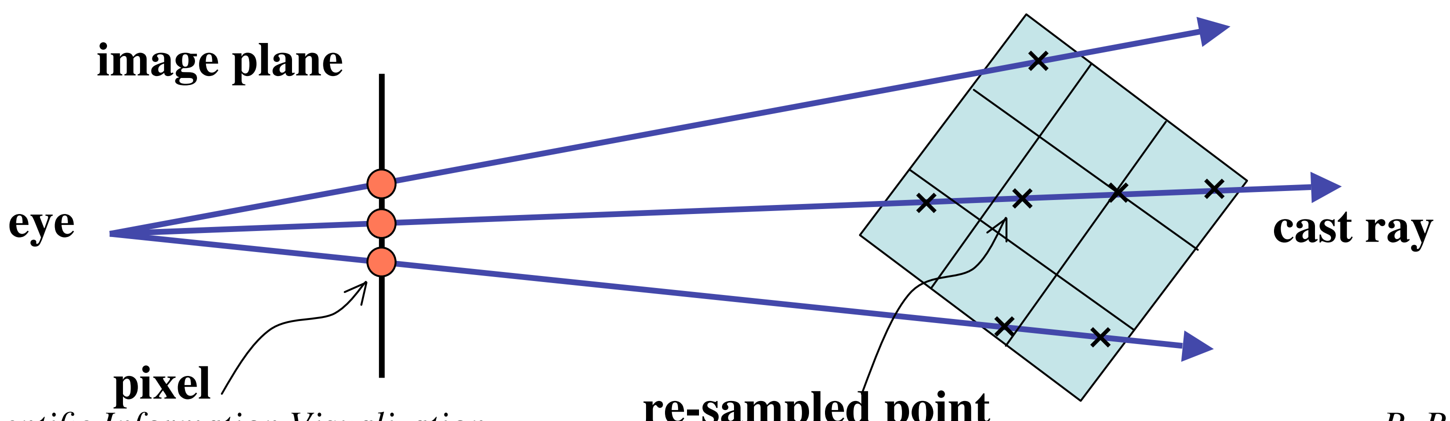
# Ray Casting

# Ray Casting

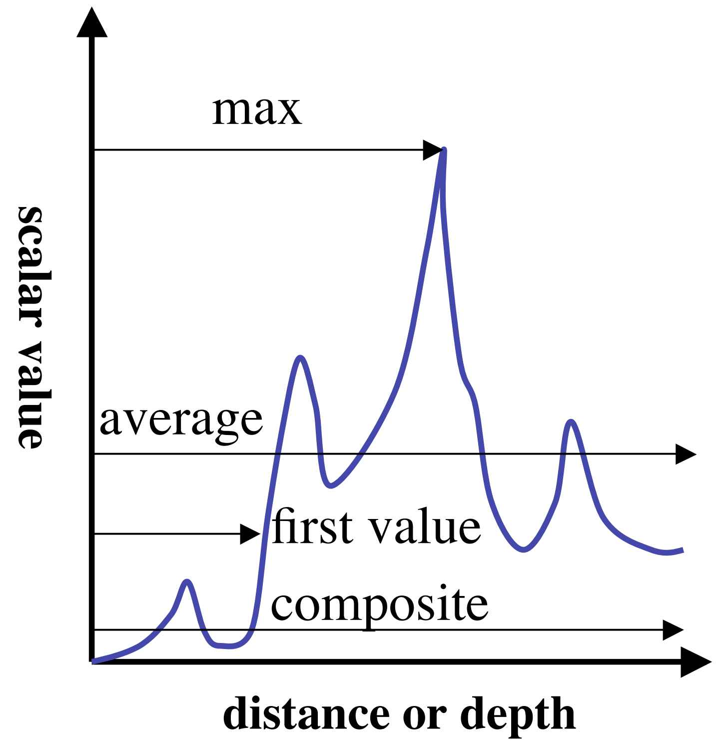
- Image-order rendering
  - We determine for each pixel on the image plane which data samples contribute to it

- Ray casting is a rendering method that
  - Creates rays from the eye toward world objects
  - Determines the color of the ray using light transport information
  - Pixel color of the image is the color of the ray (passing through the pixel)
  - Rays typically do not spawned (only primary rays)



# Projection Schemes

- Use some specified scheme (also called a ray function) to process data encountered along the ray
- Different schemes
  - ➢ Maximum value projection
  - ➢ Average value for the ray path
  - ➢ Distance to first voxel
    (of value at or above the given value)
  - ➢ Composition technique
    (mapping scalar values on the ray path to various colors and opacity using transfer functions)
- Composite technique is most effective for volume rendering



# Classification

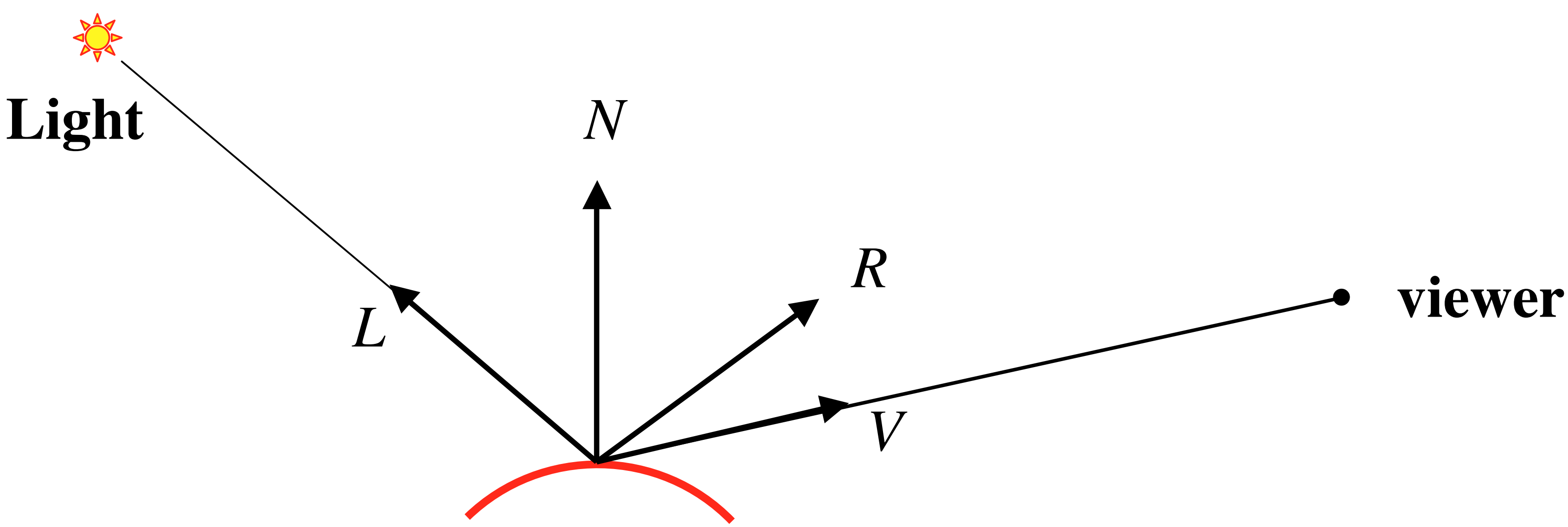
- Classify the relevant objects (or structures) of interest within a dataset
- Map information (e.g., density) at a voxel location into different values such as material, color or opacity
- This mapping function is called transfer function
- Examples:
  - Binary classification
    - 0 means background (air)
    - 1 means part of the object (bone)
  - Voxel density → optical properties (RGBA)
  - Classification based on scalar value and gradient magnitude

# Volumetric Illumination

*Refer OpenGL lighting notes for details*

$$\text{Color} = M_e + G_a M_a + \sum_{i=1}^{n-1} f_i s_i [I_a M_a + I_d M_d (\max\{\vec{N} \bullet \vec{L}, 0\}) + I_s M_s (\max\{\vec{R} \bullet \vec{V}, 0\})^n]_i$$

$$f_i = \frac{1}{k_c + k_l d + k_q d^2}$$ is attentuation factor



# Interpolation

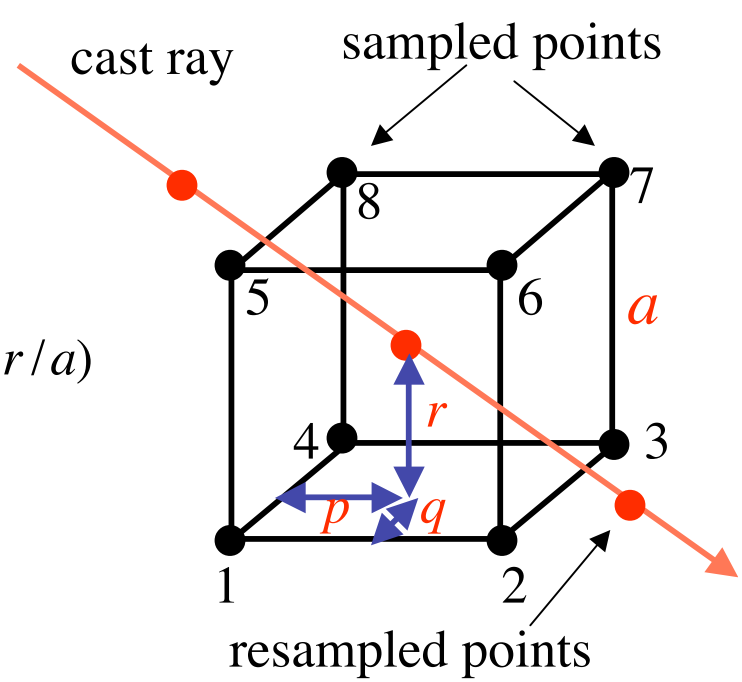
- Traverse the data along a ray (parametric form)

$$(x, y, z) = (x_0, y_0, z_0) + (a, b, c)t$$

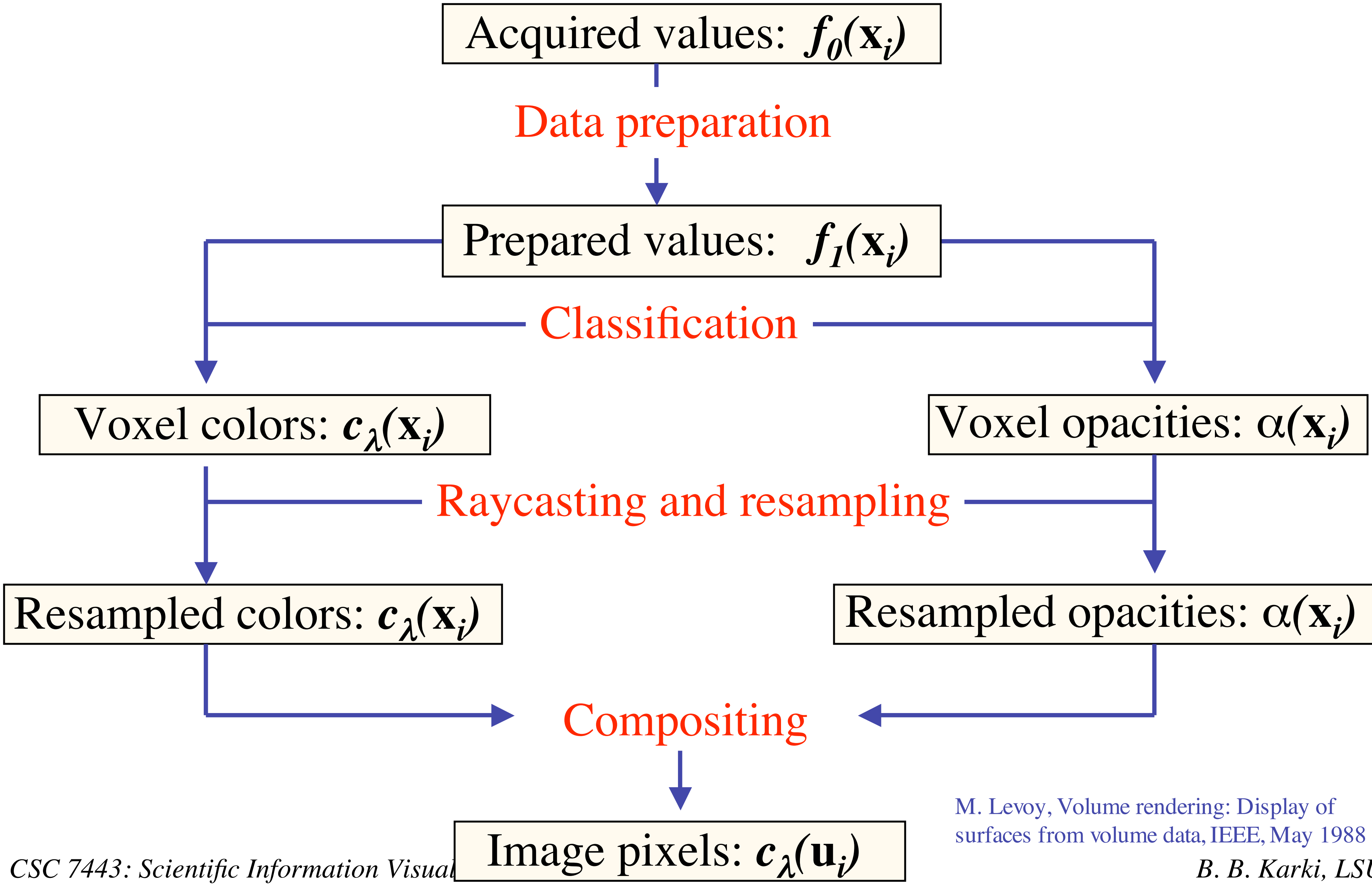
- Need to sample the volume at uniform intervals

- Trilinear interploation
  - Value at some location is defined by using linear interpolation based on distance along each of 3 axes

$$\boldsymbol{f_v} = f_1(1-p/a)(1-q/a)(1-r/a) + f_2(p/a)(1-q/a)(1-r/a)$$
$$+ f_3(p/a)(q/a)(1-r/a) + f_4(1-p/a)(q/a)(1-r/a)$$
$$+ f_5(1-p/a)(1-q/a)(r/a) + f_6(p/a)(1-q/a)(r/a)$$
$$+ f_7(p/a)(q/a)(r/a) + f_8(1-p/a)(q/a)(r/a)$$



# Levoy’s Method: Rendering Pipeline



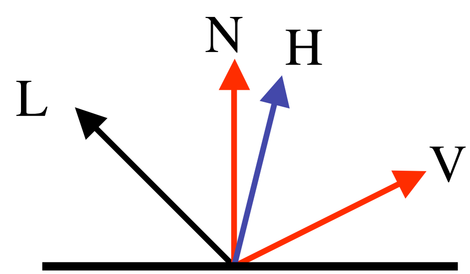

M. Levoy, Volume rendering: Display of surfaces from volume data, IEEE, May 1988

# Data Preparation

- Begin with an array of acquired values $f_0(\mathbf{x}_i)$ at voxel locations $\mathbf{x}_i = (x_i, y_i, z_i)$
- Make sure all grid points have data
  - Fill the missing data by interpolation
  - Corrections for nonorthogonal sampling grids
- Remove noises (filtering, patient's motion)
- Enhance the contrast
- Output is an array of prepared values, $f_1(\mathbf{x}_i)$

# Illumination or Shading

- A shading model provides illusion of smooth surfaces




- Phong model

$$c(\vec{x}) = c_p k_a + \frac{c_p}{k_c + k_l d(\vec{x})}\left(k_d(\vec{N}(\vec{x}).\vec{L}) + k_s(\vec{N}(\vec{x}).\vec{H})^n\right)$$

**H** is half-vector between **L** and **V**

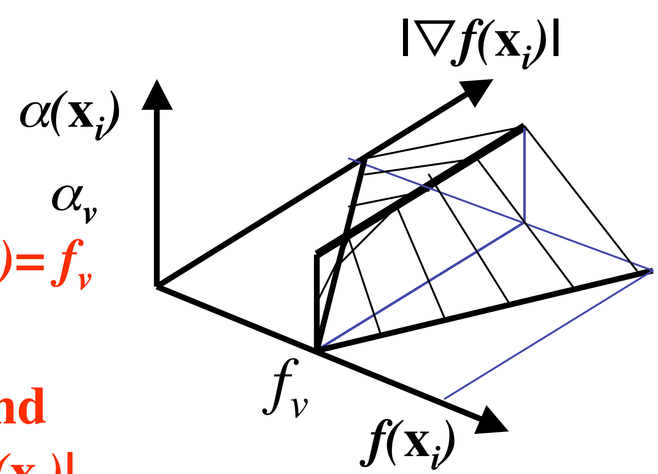
- Compute normal **N(x)** at voxel using central difference

$$\nabla f(x_i, y_j, z_k) = \begin{pmatrix} f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k) \\ f(x_i, y_{j+1}, z_k) - f(x_i, y_{j-1}, z_k) \\ f(x_i, y_j, z_{k+1}) - f(x_i, y_j, z_{k-1}) \end{pmatrix}$$

# Classification Based on Isovalues

- Assign an opacity $\alpha_v$ to voxels having selected value $f_v$, and zero opacity to all other voxels

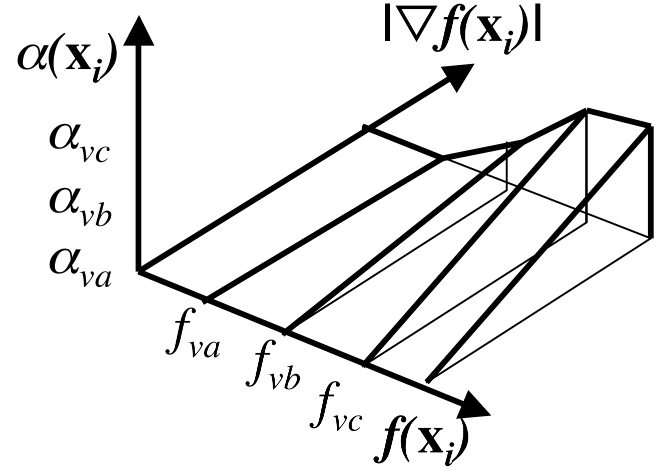
- Make the opacity fall off as you move away from the selected value at a rate inversely proportional to the magnitude of the local gradient vector




$$\alpha(\vec{x}_i) = \alpha_v \begin{cases} 1 & \text{if } |\nabla f(\mathbf{x}_i)| = 0 \text{ and } f(\mathbf{x}_i) = f_v \\ 1 - \dfrac{1}{t}\left|\dfrac{f_v - f(\vec{x}_i)}{|\nabla f(\vec{x}_i)|}\right| & \text{if } |\nabla f(\mathbf{x}_i)| \neq 0 \text{ and } |f(\mathbf{x}_i) - f_v| \leq t|\nabla f(\mathbf{x}_i)| \\ 0 & \text{otherwise} \end{cases}$$

# Classification Based on Boundaries

- First map the voxel values of distinct regions to different opacities
- Linearly interpolate the boundary regions between different regions
- Enhance the boundaries using the gradient value




If $f_{v_n} \le f \le f_{v_{n+1}}$

$$\alpha(\vec{x}_i) = |\nabla f(\vec{x}_i)| \left\{ \alpha_{v_{n+1}} \left( \frac{f(\vec{x}_i) - f_{v_n}}{f_{v_{n+1}} - f_{v_n}} \right) + \alpha_{v_n} \left( \frac{f_{v_{n+1}} - f(\vec{x}_i)}{f_{v_{n+1}} - f_{v_n}} \right) \right.$$

Otherwise

$$\alpha(\vec{x}_i) = 0$$

# Resampling & Compositing

- Resampling is done at evenly spaced locations along the ray using trilinear interpolation

- Compositing:

$$c_{out} = c_{in} \cdot (1-\alpha) + c \cdot \alpha$$
$$\alpha_{out} = \alpha_{in} \cdot (1-\alpha) + \alpha$$

where $c$ is the color (gray in Levoy's method), and $\alpha$ is the opacity at a resampled point

- Resampled colors and opacities are merged with each other and with the background by compositing in back-to-front order to yield a single color for the ray.

# Ray Casting V-Buffer

- Ray casting method (Upson and Keeler, 1988)
  - Rays are cast from each pixel on the image plane into the volume
  - Each ray is stepped inside the cell, with calculations for scalar values, shading, opacity, etc., performed at each point
  - The process is repeated for each cell along the ray, accumulating color and opacity.
- Computed colors can be stored in a buffer, called V-buffer
  - Z-buffer is 2D pixel array with depth value
  - V-buffer is an extension of 2D to 3D
- V-buffer improves the performance because no need to re-compute the colors as the viewpoint moves.

# Nodal Shading Function

- Goal is not to provide a realistic image, but rather to provide an useful representation

- A simplified shading model is used:
  - Omit the phong term keeping only the ambient and diffuse term so lighting is independent of viewing direction

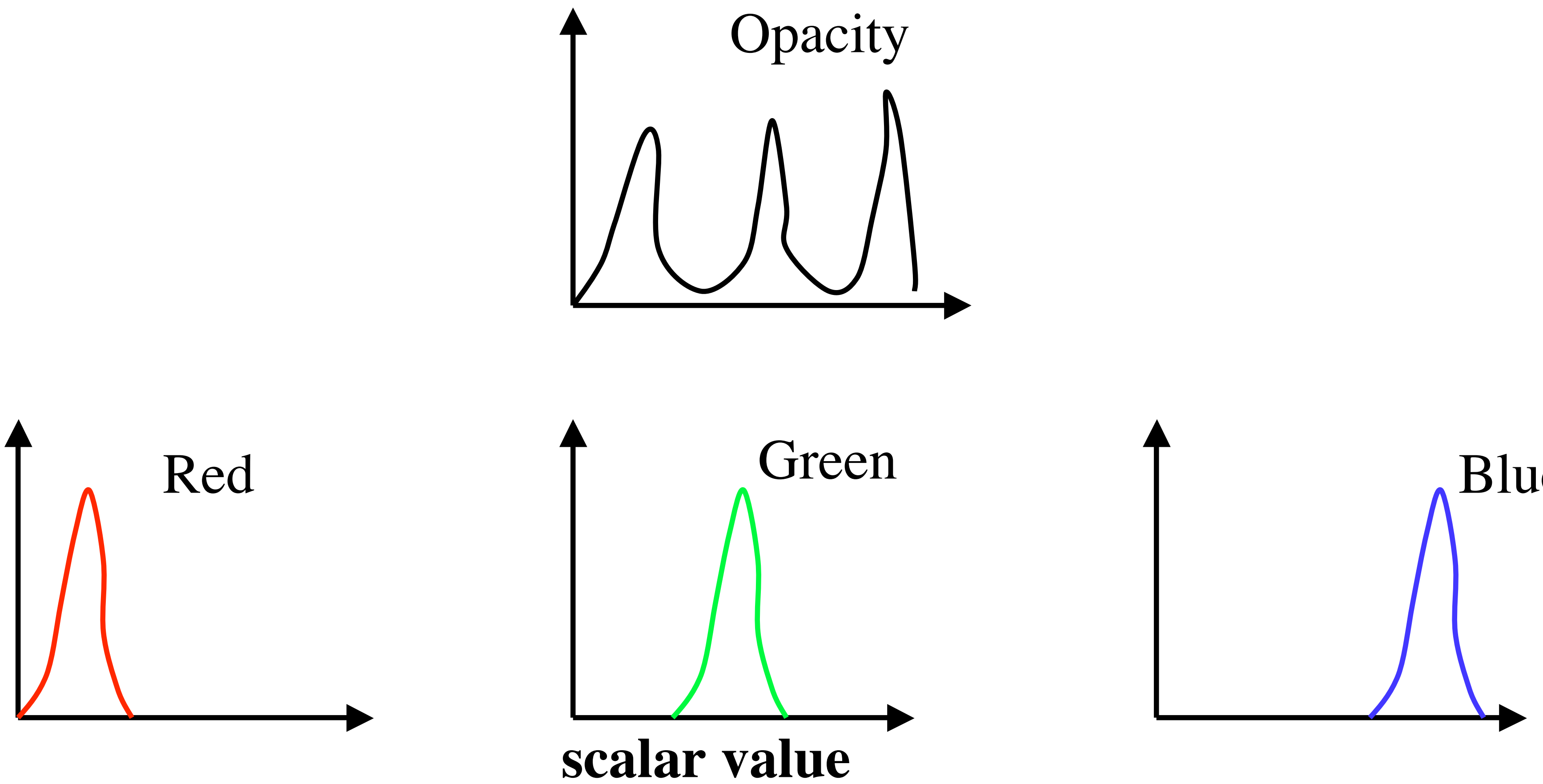
$$c(\vec{x}) = c_p k_a + \frac{c_p c_L}{k_c + k_l d(\vec{x})} k_d (\vec{N}(\vec{x}).\vec{L})$$

  - The diffusion coefficient is also varied with scalar value to highlight certain features in the final image

- Accumulating along the ray gives the intensity or the pixel's color

# Transfer Functions

- Highlight three surfaces in the volume, one in red, one in green and one in blue



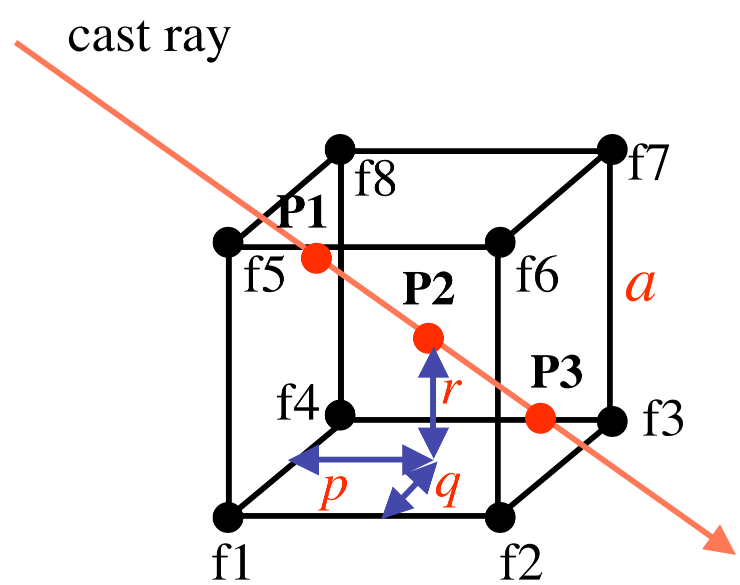
# Improving Ray Casting

- Improvement approaches use one or more of the following principles:

- Image-space coherency
  - M. Levoy., Volume rendering by adaptive refinement, *The Visual Computer*, 6(1):2-7, 1990.
- Object-space coherency
  - T. van Walsum, et al., Efficient hybrid rendering of volume data and polygons. *Advances in Scientific Visualization*, 83-96. Springer-Verlag Berlin-Heidelberg, 1992.
- Inter-ray coherency
  - R. Yagel and A. Kaufman. Template-based volume viewing. *Computer Graphics Forum*, 11(3):153-167, 1992.
- Inter-frame coherency
  - R. Yagel and Z. Shi. Accelerating volume animation by space-leaping. In *Proceedings of Visualization 1993*, 62-69, 1993.
- Empty space skipping
  - M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245-261, 1990.
- Efficient memory access
  - S. Parker et al., Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):238-250, 1999.

# Example: Raycasting

- The scalar values at eight vertices of the voxel of unit length (**a=1**) are
  **f1=5.0, f2=5.1, f3=5.2, f4=4.6**
  **f5=4.9, f6=5.1, f7=5.3, f8=4.7**
  The ray is sampled at three points
  **P1(0.2,0.3,0.8), P2(0.5,0.5,0.5), P3(0.8,0.7,0.2)**
  Color and opacity of the cast ray before it hits **P1** is (0,0,0,0)




- Find the color of the ray after it leaves **P3** by compositing from back-to-front order
  -Use trilinear interpolation to find the scalar values at P1, P2 and P3
  -Use transfer functions (mapping) for the colors (RGB) and opacity (O)