Volume Rendering (2)

Remember ...
- The paper we discussed last time
  - Discrete Sampling (jagged edges)
  - Binary Classification (no fuzzy objects)
  - Shading is based on binary classification (quality is bad)

Levoy's 1988 paper
- Tried to improve the above problems
  - Node-Center Voxel
  - Floating point sampling
  - No explicit surface detection
  - Shading and classification are done separately

Volume Rendering Pipeline

Basic Idea
- Data are defined at the corners of each cell (voxel)
- The data value inside the voxel is determined using tri-linear interpolation
- No ray position round-off is needed when sampling
- Accumulate colors and opacities along the ray path

Shading and Classification
- Shading: compute a color for each data point in the volume
- Classification: Compute an opacity for each data point in the volume
- Done by table lookup (transfer function)
- Levoy preshaded the entire volume
Shading

- Use a Phong illumination model

Light (color) = ambient + diffuse + specular

\[ C(x) = C_p \cdot K_a + \frac{C_p}{K_1 + K_2 \cdot d(x)} \cdot (K_d \cdot (N(x) \cdot L + K_s \cdot (N(x) \cdot H)^n) \]

- \( C_p \): Color of light
- \( K_a, K_d, K_s \): Ambient, diffusive, specular coefficients
- \( K_1, K_2 \): Constants (used for depth attenuation)
- \( N(x) \): Normal at \( x \)

Normal estimation

How to compute \( N(x) \)?

1. Compute the gradient at each corner
2. Interpolate the normal using central difference

\[ N(x,y,z) = \left[ \frac{(f(x+1)-f(x-1))}{2}, \frac{(f(y+1)-f(y-1))}{2}, \frac{(f(z+1)-f(z-1))}{2} \right] \]

Classification

Classification: Mapping from data to opacities

- Region of interest: high opacity (more opaque)
- Rest: translucent or transparent

The opacity function is typically specified by the user

Levoy came up with two formulas to compute opacity

1. Isosurface
2. Region boundary (e.g., between bone and fresh)

Opacity function (1)

Goal: Visualize voxels that have a selected threshold value \( f_v \)

- No intermediate geometry is extracted
- The idea is to assign voxels that have value \( f_v \) the maximum opacity (say \( \alpha \))
- And then create a smooth transition for the surrounding area from 1 to 0
- Levoy wants to maintain a constant thickness for the transition area.

Opacity function (2)

Maintain a constant isosurface thickness

Can we assign opacity based on function value instead of distance? (Local operation: we don’t know where the isosurface is)

Yes – we can based on the value distance \( f - f_v \) but we need to take into account the local gradient

Opacity function (3)

Assign opacity based on value difference \( f - f_v \) and local gradient

Gradient: the value fall-off rate \( \text{grad} = \frac{\Delta f}{\Delta s} \)

Assuming a region has a constant gradient and the isosurface transition has a thickness \( R \)

- \( f(x) \)
- \( f_v \)
- \( \text{grad} \)
- \( R \)

Then we interpolate the opacity

\[ \text{opacity} = \alpha - \alpha \cdot \left( \frac{f_v - f(x)}{(\text{grad} \cdot R)} \right) \]

\[ \text{thickness} = R \]
Continuous Sampling

- We sample the volume at discrete points along the ray (Levoy sampled color and opacity, but you can sample the value and then assign color and opacity)
- No integer round-off
- Use trilinear interpolation
- Compositing (front-to-back or back-to-front)

Tri-linear Interpolation

- Use 7 linear interpolations
- Interpolate both value and gradient
- Levoy interpolate color and opacity

Compositing

The initial pixel color = Black

Back-to-Front compositing: use ‘under’ operator

\[ C = C_1 \text{ 'under' background} \]
\[ C = C_2 \text{ 'under' } C \]
\[ C = C_3 \text{ 'under' } C \]

\[ ... \]

\[ C_{out} = C_{in} \times (1 - \alpha(x)) + C(x) \times \alpha(x) \]

Compositing (2)

Or you can use ‘Front-to-Back’ Compositing formula

Front-to-Back compositing: use ‘over’ operator

\[ C = \text{background ‘over’ } C_1 \]
\[ C = C \text{ ‘over’ } C_2 \]
\[ C = C \text{ ‘over’ } C_3 \]

\[ ... \]

\[ C_{out} = C_{in} + C(x) \times (1 - \alpha_{in}) \]
\[ \alpha_{out} = \alpha_{in} + \alpha(x) \times (1 - \alpha_{in}) \]