Volume Rendering

Full Volume Rendering
Full Volume Rendering

- Transfer function maps voxel density to color and *opacity*
- During raycasting, rays not only stop at isovalue as isosurface rendering
- Rays accumulate colors and opacities as they traverse the volume
- Once accumulated opacity reach 1.0, the ray stops
Ray Traverse

- $\alpha = 0.0$
- $\alpha = 0.5$
- $\alpha = 0.75$
- $\alpha = 0.85$
- $\alpha = 0.99$

- voxel value = 200
- voxel value = 100

stop here
Volume Integration

- Continuous volume shading effects accumulated by integration, \( \text{delta}_s \) is the step distance

\[
I_{\lambda}(x, r) = \int_{0}^{L} C_{\lambda}(s) \mu(s) e^{-\int_{0}^{s} \mu(t) dt} ds
\]

\[
I_{\lambda}(x, r) = \sum_{i = 0}^{L/\Delta s} C_{\lambda}(i\Delta s) \mu(i\Delta s) \Delta s \cdot \prod_{j = 0}^{i-1} e^{-\mu(j\Delta s)\Delta s}
\]
Simplified Volume Integration

- Define transparency and apply Taylor expansion to exponential term
  \[
  \exp(-\mu(i\Delta s)\Delta s) = t(i\Delta s) \approx 1 - \mu(i\Delta s)\Delta s
  \]

- Then, opacity alpha
  \[
  \mu(i\Delta s)\Delta s \approx 1 - t(i\Delta s) = \alpha(i\Delta s)
  \]

- Then the volume integration becomes
  \[
  I_x(x, r) = \sum_{i=0}^{L/\Delta s} C(i\Delta s)\alpha(i\Delta s) \cdot \prod_{j=0}^{i-1} (1 - \alpha(j\Delta s))
  \]
Compositing

• The integration now can be solved by compositing step by step
  – Accumulating colors weighted by opacity
  – For example, composite two images

\[
\text{rgb} = \text{RGB}_{\text{back}} \cdot \alpha_{\text{back}} (1 - \alpha_{\text{front}}) + \text{RGB}_{\text{front}} \cdot \alpha_{\text{front}}
\]

\[
\alpha = \alpha_{\text{back}} (1 - \alpha_{\text{front}}) + \alpha_{\text{front}}
\]
Volume Composition

- Two image composition extends to multiple steps recursively

\[
\text{rgb}_{\text{front}} = \text{rgb}_{\text{back}} (1 - \alpha_{\text{front}}) + \text{rgb}_{\text{front}}
\]

\[
\alpha_{\text{front}} = \alpha_{\text{back}} (1 - \alpha_{\text{front}}) + \alpha_{\text{front}}
\]

- Ray stops when
  - Move out of the volume
  - Opacity bigger than 1.0

- Return rgb values to assign pixels on image plane
Full Volume Rendering Algorithm

```
for each image pixel i, j

ray = (P(i, j) - eye) / | (P(i, j) - eye) |  /* the ray direction vector, normalized */

P(i, j) = P(0, 0) + i * v + j * u;  /* the location of image pixel (i, j) in world (volume) space */

{r, g, b} = 0, α = 0  /* initialize red, green, blue color and opacity α to 0 */

for (t = t_front; t ≤ t_back; t += stepSize) /* traverse the volume front to back */

    sampleLoc = P(i, j) + t * stepSize * ray  /* step along the ray */

    intVal = Interpolate(V, sampleLoc)

    if (AlphaTransFunc(intVal) > 0.05) /* only do work for non-transparent samples */

        gradVec = ComputeGradientVector(V, sampleLoc)

        {R,G,B} = Shade(gradVec, lightSource, eye, sampleLoc, {R,G,B} = TransFunc(intVal))

        {r, g, b} = AlphaTransFunc(intVal) * {R,G,B} * (1 - α) + {r, g, b}  /* composite color */

        α = AlphaTransFunc(intVal) * (1 - α) + α  /* composite opacity */

    if α > 0.95 /* everything further is hidden and can't be seen, so stop the ray */

    I(i, j) = {r,g,b};  break;  /* write color to image pixel and go to next pixel */
```
• Transfer functions analysis
• Automatic classification and segmentation
• Visual Analytics