Direct Volume Rendering
Discrete Implementation

Han-Wei Shen
The Ohio State University
Discrete Implementation

- Numerical integration:

\[ \Delta x = x_i - x_{i-1} \]

\[ \int_0^D h(x) \, dx = \sum_{i=1}^{i=n} h(x_i) \Delta x \]  
(Riemann Sum)

\[ e^{-\int_0^D \tau(t) \, dt} = e^{-\sum_{i=1}^{i=n} \tau(t_i) \Delta t} \]

\[ = e^{-\sum_{i=1}^{i=n} \tau(i \Delta x) \Delta x} \]

\[ = \prod_{i=n}^{i=n} e^{-\tau(i \Delta x) \Delta x} \]

\[ = \prod_{i=1}^{i=n} (1 - \alpha_i) \]  
(Remember \( 1 - e^{-\int_0^D \tau(t) \, dt} = \alpha \))
Discrete Implementation

\[ I(D) = I_0 \times e^{-\int_0^D \tau(t)dt} + \int_0^D g(s)e^{-\int_s^D \tau(t)dt}ds \]

\[ I_0 \prod_{i=1}^{n}(1 - \alpha_i) \]

\[ \sum_{i=1}^{n} g_i \times \prod_{j=i+1}^{n} (1 - \alpha_i) \]

= \( g_n + (1-\alpha_n)(g_{n-1} + (1-\alpha_{n-1})(g_{n-2} + (1-\alpha_{n-2})(\ldots (1-\alpha_2)(g_2 + (1-\alpha_1)(g_1 + I_0))))\ldots) \)

This is called - Back to Front Compositing

\( I_0 \)

\( S = 0 \) to \( S = D \)

\( g_1 \ g_2 \ g_3 \ g_4 \ g_5 \ g_6 \ g_7 \ g_8 \ g_9 \ g_{10} \ldots \)

\( \alpha_1 \ \alpha_2 \ \alpha_3 \ \alpha_4 \ \alpha_5 \ \alpha_6 \ \alpha_7 \ \alpha_8 \ \alpha_9 \ \alpha_{10} \ldots \)
Discrete Implementation

\[ I(D) = I_0 \times e^{-\int_0^D \tau(t)\,dt} + \int_0^D g(s) e^{-\int_s^D \tau(t)\,dt} \,ds \]

\[ I_0 \prod_{i=1}^{i=n} (1 - \alpha_i) \]

\[ \sum_{i=1}^{i=n} (g_i \times \prod_{j=i+1}^{n} (1 - \alpha_i)) \]

\[ = g_n + (1 - \alpha_n)(g_{n-1} + (1 - \alpha_{n-1})(g_{n-2} + (1 - \alpha_{n-2}(.... (1 - \alpha_2)(g_2 + (1 - \alpha_1)(g_1 + I_0)))))))...)))) \]
Ray Casting Algorithm

• For each pixel
  – Cast a ray into the volume
  – Linearly interpolate data values from cell (voxel) corners
  – Convert the data values to optical properties (color and opacity)
  – Composite the optical properties
  – Return the final color
Shading and Classification

- **Shading**: computer a color for every sample in the volume
- **Classification**: computer an opacity for every sample in the volume

This is often done through a table (transfer function) lookup
Shading

• Use the Phong illumination model

illumination = ambient + diffuse + specular

\[ = C(x_i) \times I_a + C(x_i) \times I_d \times (N.L) + C(x_i) \times I_s \times (R.V)^n \]

C(x_i) : color of sample i
I_a, I_d, I_s: light’s ambient, diffuse, and specular colors (usually set as white)
N: normal at sample i
V: vector from sample point to eye
L: light vector, from sample to light source
R: reflection vector of light vector
n: shininess
Normal Estimation

• How to compute the sample normal $N$?
  – Normal: a vector that is perpendicular to the local surface, which is the gradient of the sample point

1. Compute the gradient $G$ at the cell corners using central difference
2. Linearly interpolate the gradients

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

• Classification: mapping from data values to opacities

\[
I(D) = I_0 \times e^{-\int_0^D \tau(t) dt} + \int_0^D g(s) e^{-\int_s^D \tau(t) dt} ds
\]

– Region of interest: high opacity
– Rest: translucent or transparent

• The opacity function, or called transfer function, is given by the user
Ray Sampling

• Sample the volume at discrete points along the ray
• Perform tri-linear interpolation to get the sample values
• Look up the transfer function to get the color and opacity
• Compositing the color/opacity (front-to-back or back-to-front)
Back-to-Front Compositing

The initial pixel color = Black

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

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

\[ C_{\text{out}} = C_{\text{in}} \times (1-\alpha(x)) + C(x) \times \alpha(x) \]
(this is the alpha blending formula)
Front-to-Back compositing: use ‘over’ operator

C = background ‘over’ C1
C = C ‘over’ C2
C = C ‘over’ C3
...

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