Interactive visualization of volumetric data with WebGL in real-time

Mario J. Lorenzo
Problem

Rendering of volumetric data with the following ...

- Real-time
- Realistic
- Interactive

...supported through a web-browser
Medical imaging techniques such as CT scan, MRI, and PET produce sets of parallel slices that form volumetric dataset.

Storage and Distribution of medical images are managed by a Picture Archiving and Communication Systems (PACS)
  - Client –to- Server architecture
  - Requires specialized software

Weather data is also volumetric
  - Collected by using radar that scans a volume surrounding it
  - Collects values in a circular or conical scan surface
  - Doppler radar also samples physical variables (reflectivity, radial velocity etc)
Problem Domain Motivation

- Scientific fields such as Medical Imaging and Meteorology have been pioneers in 3D visualization
- Visualization is the primary target of analysis by Radiologist and Meteorologist
  - The advent of CT and MRI scans provides slices of images that can be used to render 3D volumes
  - 3D volume rendering as a basis for visual assessment by radiologist is still new and not widely used.
  - May require re-training for how radiologists diagnose
- Improving fidelity and dimensionality of the visualization can directly improve the overall accuracy of the decision making.
Background and Prior Research

• Volume Rendering seeks to project a 3D model into 2D image
  – The ability to interact in real-time gives the illusion of a 3D image

• Prior to WebGL, 3D visualization was performed through native applications or non-standard plug-ins installed in the web browser
  – Requires access to local hardware (i.e. GPU)

• Mostly rendering polygon meshes
  – Straightforward and supported by GPU raster rendering
Background: Volume Rending

Marc Levoy - considered a pioneer in volume rendering; worked on visualizing CT imaging


Demonstrated that volume-rendering techniques produced better images from volumetric data than using conventional geometric primitives.
Describes Volume ray-casting is most basic, but most flexible volume rendering algorithm
- Ray-casting as a numerical method for evaluating volume rendering integral
  - Integral is based on integrating light interaction effects along a viewing ray on the optical model
  - Absorption – radiant energy absorbed over distance
  - Alpha Blending – combining translucent foreground color with background color

Volume is sampled at regular intervals between starting and ending positions
Proposed Method


• Approach relies on Levoy and Hadwiger et al work on volume rendering algorithms

• Adapted Volume Ray-casting algorithms for WebGL

• Volume interpolation from 2D image slices

• Avoids use of server-side for remote rendering, enabling better scalability for large-user environments (such as in medical imaging)
  • Does not require preprocessing

• Realism is accomplished through the use of blending functions, transfer functions, and Phong illumination
Method Details

Composing Texture from Image slices
- Construct a “texture” by transforming individual image slices into mosaic (matrix) of images
- Uses interpolation to fill in the gaps in between the slices

Identification of Ray coordinates
- Uses a cube as the bounding geometry for the image being rendered
- Renders the cube into scene using view point
- First renders the color using depth function change
  - 1st pass presents closest region of the cube to the camera
  - 2nd pass presents far region

Ray generation
- Ray is calculated from front region to back region
  - Color interpolation is used
- Cartesian x,y,z coordinates defined where one dimension is an image slice selector and the other two is a coordinate within that image
- Ray is divided by S steps (number of samples)
  - Compute x,y,z for each sample in the volume
  - Value of position is interpolated from the texture

Transfer Function
- Use value of texture at (x,y,z) to identify color
- Use color in composition function
- Ray finishes when composition function reaches the end in the cube
- Assigns color value to each pixel along the ray

\[
S_a = V_a * O_f * \left( \frac{1}{s} \right)
\]
\[
S_{rgb} = V_{rgb} * S_a * L_f
\]
\[
A_{rgb}^k = A_{rgb}^{k-1} + \left( 1 - A_a^{k-1} \right) * S_{rgb}
\]
\[
A_a^k = A_a^{k-1} + S_a
\]
CT Image Slices used for Volume Rendering
## Performance Results

<table>
<thead>
<tr>
<th>Browser</th>
<th>N. Steps</th>
<th>Load Time (msec)</th>
<th>frame rate (frame/sec)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirefoxDX</td>
<td>80</td>
<td>16677</td>
<td>69</td>
<td>204</td>
</tr>
<tr>
<td>FirefoxGL</td>
<td>80</td>
<td>308</td>
<td>94</td>
<td>102</td>
</tr>
<tr>
<td>Chrome</td>
<td>80</td>
<td>18062</td>
<td>77</td>
<td>153</td>
</tr>
<tr>
<td>Opera</td>
<td>80</td>
<td>82</td>
<td>108</td>
<td>74</td>
</tr>
<tr>
<td>FF-nA</td>
<td>140</td>
<td>302</td>
<td>60</td>
<td>95</td>
</tr>
<tr>
<td>Opera</td>
<td>140</td>
<td>118</td>
<td>66</td>
<td>73</td>
</tr>
<tr>
<td>FirefoxGL</td>
<td>170</td>
<td>281</td>
<td>51</td>
<td>95</td>
</tr>
<tr>
<td>Opera</td>
<td>170</td>
<td>102</td>
<td>54</td>
<td>77</td>
</tr>
<tr>
<td>FirefoxGL</td>
<td>200</td>
<td>312</td>
<td>44</td>
<td>96</td>
</tr>
<tr>
<td>Opera</td>
<td>200</td>
<td>111</td>
<td>48</td>
<td>81</td>
</tr>
</tbody>
</table>
Demo of CT Scan 3D Rendering


http://demos.vicomtech.org/volren/
Related Work

  • Combine surface rendering combined with remote volume rendering
  • WebGL based on client side and native process for volume rendering
  • Solution for 3D medical volume reconstruction
  • Claim it is “pure web environment”, but relies on remote rendering called “master-slave”

  • XTK – “The X Toolkit”
  • Claims to be first WebGL / JavaScript-based visualization for medical imaging
  • [http://socr.umich.edu/HTML5/BrainViewer/](http://socr.umich.edu/HTML5/BrainViewer/)
Review

• A general concern on the effects of interpolation and other simulations that can distort or mislead an expert

• Latency for loading the original images (which can be very large in medical imaging)
  – Consider compression optimizations, video compression schemes may help eliminate temporal redundancy

• Surface rendering for textures can be very helpful to radiologists

• Explore use of ThreeJS framework to abstract from WebGL (such as XTK and ATM)

• Reduced Resolution
  – Medical Images use large sample bit depth (vs common 8bit)

• Issues with reliability and consistency of Javascript setTimeout() function having a 10ms overhead per call, causing performance issues

• Exposing additional controls for manipulating lighting, may be helpful
Reviewed Paper:


Additional papers

