Real-Time Volume Rendering of Four Channel Data Sets

Jürgen P. Schulze and Alexander C. Rice
Brown University, Providence, RI
{schulze,acrice}@cs.brown.edu

Introduction

- This research has emerged from a collaboration with biologists at Brown University who work with images from confocal laser scanning microscopes (CLSM).
- CLSM generate stacks of images (*z-stacks*) from specimen that had been stained with up to four different fluorescent proteins.
- Sample sizes at typically tens of micrometers.
- Scans with different laser wavelengths result in separate intensity images, the voxels (=volumetric pixels) are perfectly co-registered between images.
- Voxels are located on a cartesian grid.
- "Channel" is data scanned at a particular wavelength.
- Biologists are used to seeing the reconstructed samples rendered with maximum intensity projection (MIP), when using the CLSM software.
- Our collaborators use a Leica TCS SP2 [2]. It scans up to four data channels and outputs them as one TIFF image per channel and per slice.
- Slice images typically have 512x512 pixels and are taken at a few dozen depths.
- Before our software had been developed, the biologists used to reconstruct the samples with standard image processing software. They used the following procedure:
  - Map three channels to red, green, blue (RGB, see Figure 2).
  - Map fourth channel (Figure 3) to a hue that does not (or not abundantly) exist in the RGB data set.
  - Alpha-blend image of 4th channel with images of other three channels (Figure 1).
- The above procedure was cumbersome and time consuming, and often had to be done several times with the same data set to try different color mappings.

Rendering

- Our algorithm reconstructs the volume with view-aligned textures.
- Real-time changes of the color mapping (transfer function) are possible with a pixel shader, written in Nvidia's Cg language [3], see Table 1.
- By using maximum intensity projection (MIP) we do not need to calculate opacity: multi-dimensional transfer functions as in Kniss et al. [1] are not necessary.
- The first three channels of volume are stored in the red, green, blue components of a 3D texture. The OpenGL extension `GL_EXT_texture3D` is required.
- Rendering with MIP requires the OpenGL extension `GL_EXT_blend_minmax`.
- The fourth channel is stored in the alpha channel.
- The first three channels are stored and rendered in fundamental colors (RGB).
- Fourth channel is rendered with user defined color: Can be selected from 24 bit color space; for optimum results a color with maximum intensity should be used, so the renderer can use the full intensity range to map the fourth channel.
- Good choices for the fourth color are: gray, yellow, purple, cyan: all colors with equal amounts of two or three fundamental colors.
- Shader parameters:
  - 3D texture of RGBA volume data set (`pix3dtex`).
  - Transfer function that maps intensity to data values (`pixLUT`).
  - Color of fourth channel at maximum intensity (`pixModifier`).
- Shader first computes color of fragment as if no shader was active (`origColor`).
- Then it computes intensity of fourth channel (`modColor`).

BROWN

Summary

Goal:
Render four channel data sets and allow real-time changes of colors and intensities.

Premise:
The majority of voxels contain data of only one channel.

Method:
- Store three channels in red, green, and blue texture components.
- Store fourth channel in alpha component of texture.
- Use pixel shader to map fourth channel data value to user defined color.
- Works best with maximum intensity projection.

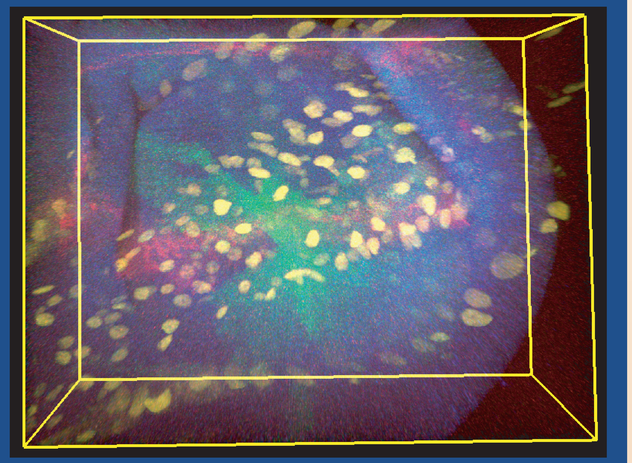

Figure 1: Sample of Drosophila larva with four data channels, obtained with a confocal laser microscope.

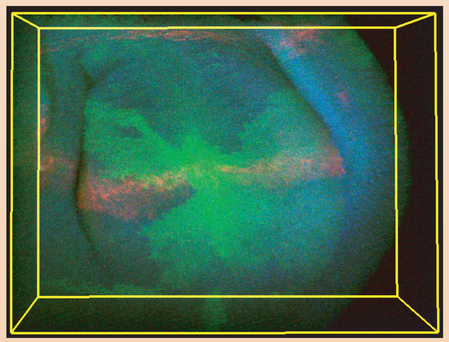

Figure 2: Three channels of Drosophila larva.

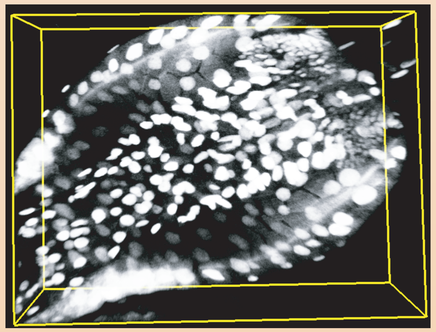

Figure 3: Fourth channel of Drosophila larva.

Results

- Integrated shader code with Brown's Cave software: 4th channel's hue and channel intensities can be controlled directly from within the virtual reality Cave.
- For the performance test we compared three versions of a 512 x 512 x 28 voxels data set: one with one channel, one with three channels, one with four channels (see Figures 1, 2, and 3).
- We observed a ~30 % performance penalty for rendering four channels vs. three (see Table 2).
- Test conditions: 1024 x 768 pixels resolution, data set reconstructed with 155 textured polygons, running on Nvidia Quadro FX 3000g.

```
struct PIN
{
  float3 coord3d : TEXCOORD0;
};

float4 main(
  const sampler3D in uniform pix3dtex : TEXTURE0,
  const sampler2D in uniform pixLUT,
  const float3 in uniform pixModifier,
  const PIN in pin) : COLOR0
{
  uniform float4 origColor = tex3D(pix3dtex, pin.coord3d);
  uniform float4 surfColor;
  uniform float modColor = tex2D(pixLUT, float2(0, origColor.w)).w;

  surfColor.x = tex2D(pixLUT, float2(0,origColor.x)).x + pixModifier.x * modColor;
  surfColor.y = tex2D(pixLUT, float2(0,origColor.y)).y + pixModifier.y * modColor;
  surfColor.z = tex2D(pixLUT, float2(0,origColor.z)).z + pixModifier.z * modColor;
  surfColor.w = max(surfColor.x, max(surfColor.y, surfColor.z))

  return surfColor;
}
```

Table 1: Pixel shader code written in Nvidia's Cg language.

Channels	Frame Rate (fps)
1	4.3
3	1.8
4	1.4

Table 2: Rendering performance. Data set: 512x512x28 voxels, rendered full screen (1024x768 pixels) with 155 textures.

Conclusion

- With our software, our collaborators can explore new and complex data sets quickly on the desktop and in virtual environments.
- No more need to merge the fourth channel into slice images with standard image processing software.
- Color mappings can be changed in real-time from within the desktop software or the virtual environment.
- Future work: Improve results achieved when rendering with alpha blending.

References

- J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun, *Gaussian Transfer Functions for Multi-Field Volume Visualization*, IEEE Visualization '03 Proceedings, pp. 65-72, 2003.
- Leica TCS SP2, *Leica Microsystems home page*, 2004. URL: http://www.leica-microsystems.com
- Nvidia Cg Toolkit, *Nvidia developers' home page*, 2004. URL: http://developer.nvidia.com/page/cg_main.html

VIS 2004 austin, texas