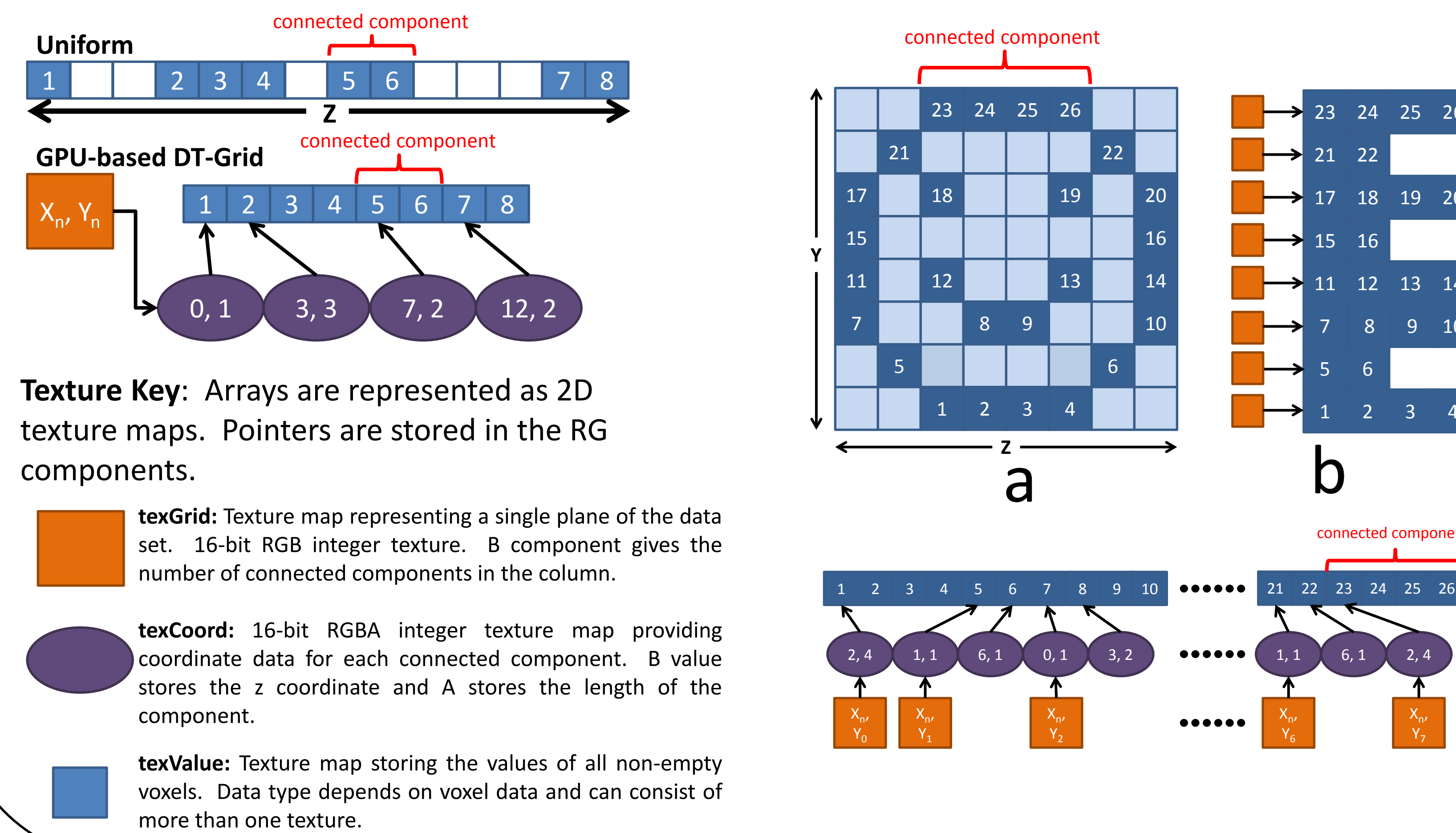


Abstract

The visualization of large-scale biomedical data is important for understanding the structure and function of biological tissue. Recent advances in hyperspectral and high-throughput imaging pose a particular challenge in visualization since the resulting data sets are very large. Fortunately, many of these data sets are sparse, containing near-zero values. While sparse data is generally handled using spatial partitioning methods, constructing efficient representations on the GPU for interactive visualization is difficult, since logarithmic-time requirements for random access make ray-tracing significantly less efficient than on uniform grids. We show that Dynamic Tubular Grids (DT-Grids) [1], a spatial partitioning method designed for narrow-band level set simulations, can be represented on the GPU and used to efficiently render sparse hyperspectral and volumetric data.

GPU-Based Dynamic Tubular Grids



Build

- Voxel values along the z-dimension are projected onto the (X, Y) plane. Empty voxels are removed.
- This results in a series of voxel stacks, or *columns*, at each point (x, y). Raw voxel values are stored in **texValue**.

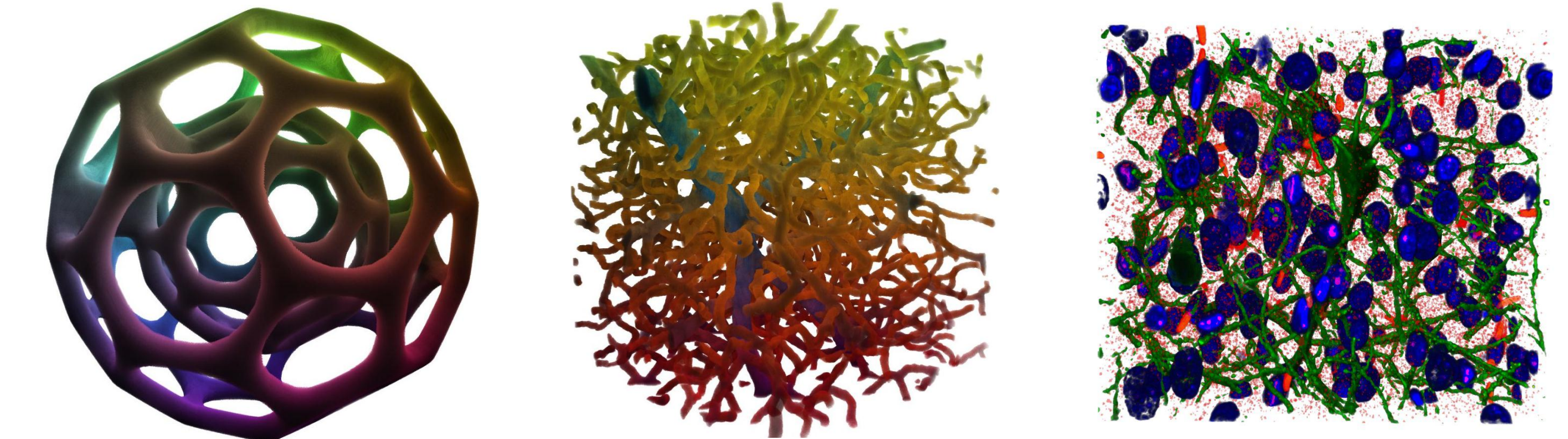
Traverse

Look up (x, y) coordinate in **texGrid**.

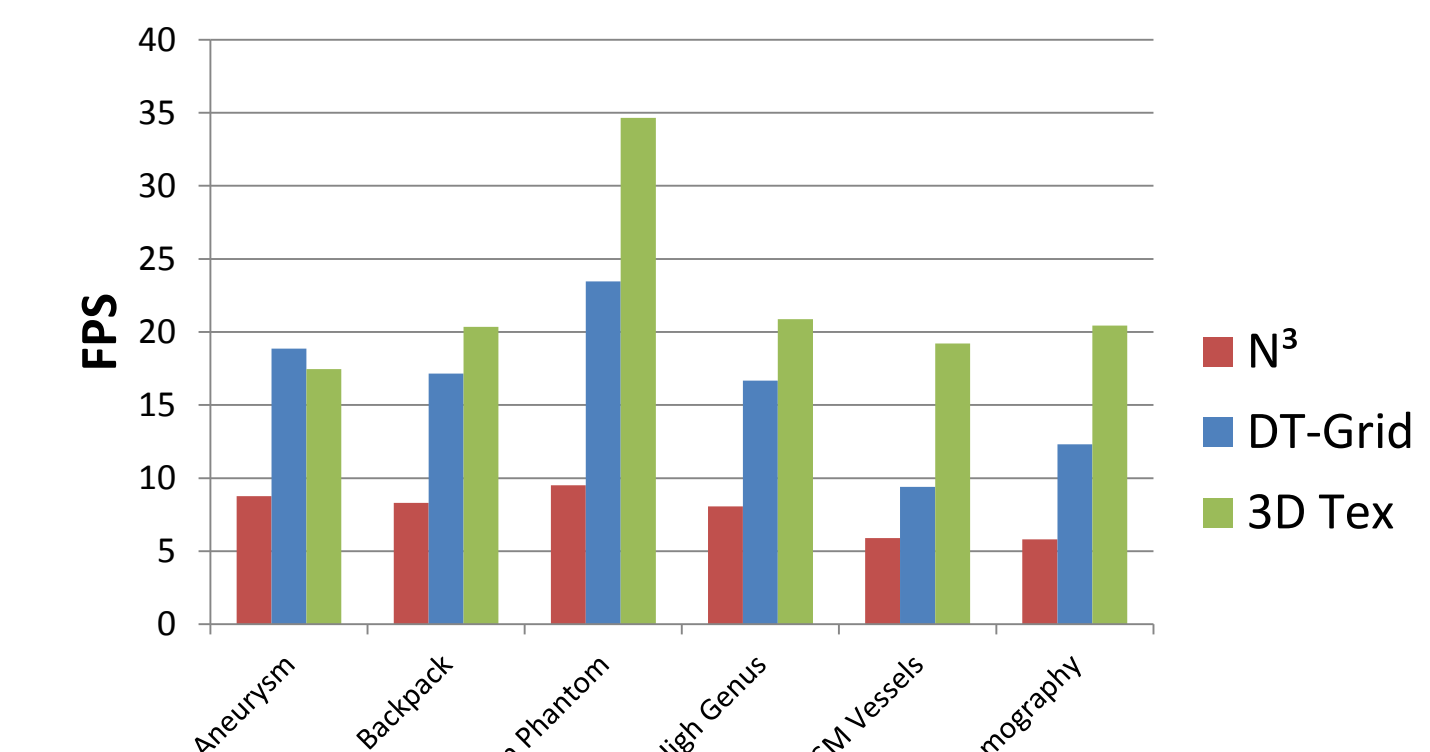
Orthographic: Step along **texValue**. Use **texCoord** to get z-axis coordinates.

Perspective: Binary search through **texCoord**. Fetch final value in **texValue**.

Volume Rendering



Proxy data sets for volume rendering: (left) High-genus shape represented as 32-bit RGBA color values (2.1GB). (center) Cortical vessels represented as 32-bit RGBA color values (2.1GB). Red = cortical depth, blue-green = vessel diameter. (right) Array tomography data represented as 32-bit RGB data (1.6GB). Each color represents protein density: Red = synapsin, Green = GFP, Blue = DAPI.



Dynamic tubular grids can be used to quickly render sparse volumetric data sets. Ray-tracing uses random texture fetches, requiring an iterative search along the **texCoord** array. This texture fetch is logarithmic. Unlike an octree, required time depends on the number of connected components rather than the size of the data set. This results in better performance for structures that have relatively few connected components. These types of structures are often found in vascular and neuronal networks.

Frame rates for several proxy data sets using volume ray-tracing. Comparison to N³-trees and uniform grids [2][4].

Summary

Advantages

- Comparable compression to GPU-based octrees
- Faster than octrees for many sparse biomedical ray-tracing applications
- Comparable to uniform grids for hyperspectral visualization, provide linear-time rendering
- Simple to integrate: implemented as a fragment shader
- RAW data can be streamed directly from out-of-core storage

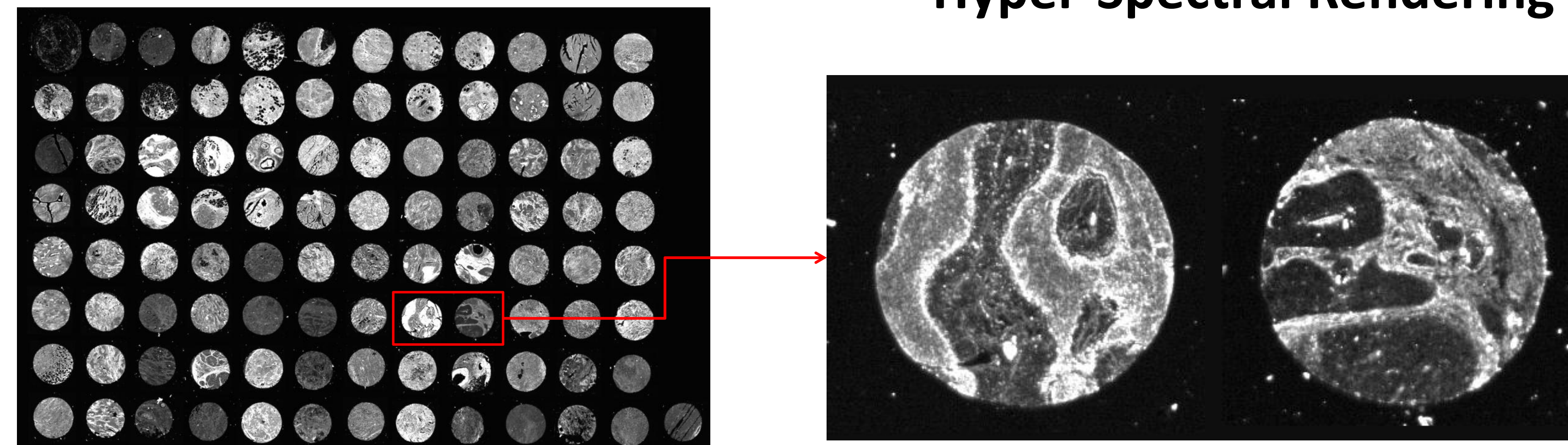
Disadvantages

- Does not support texture interpolation through graphics hardware
- No current support for level-of-detail

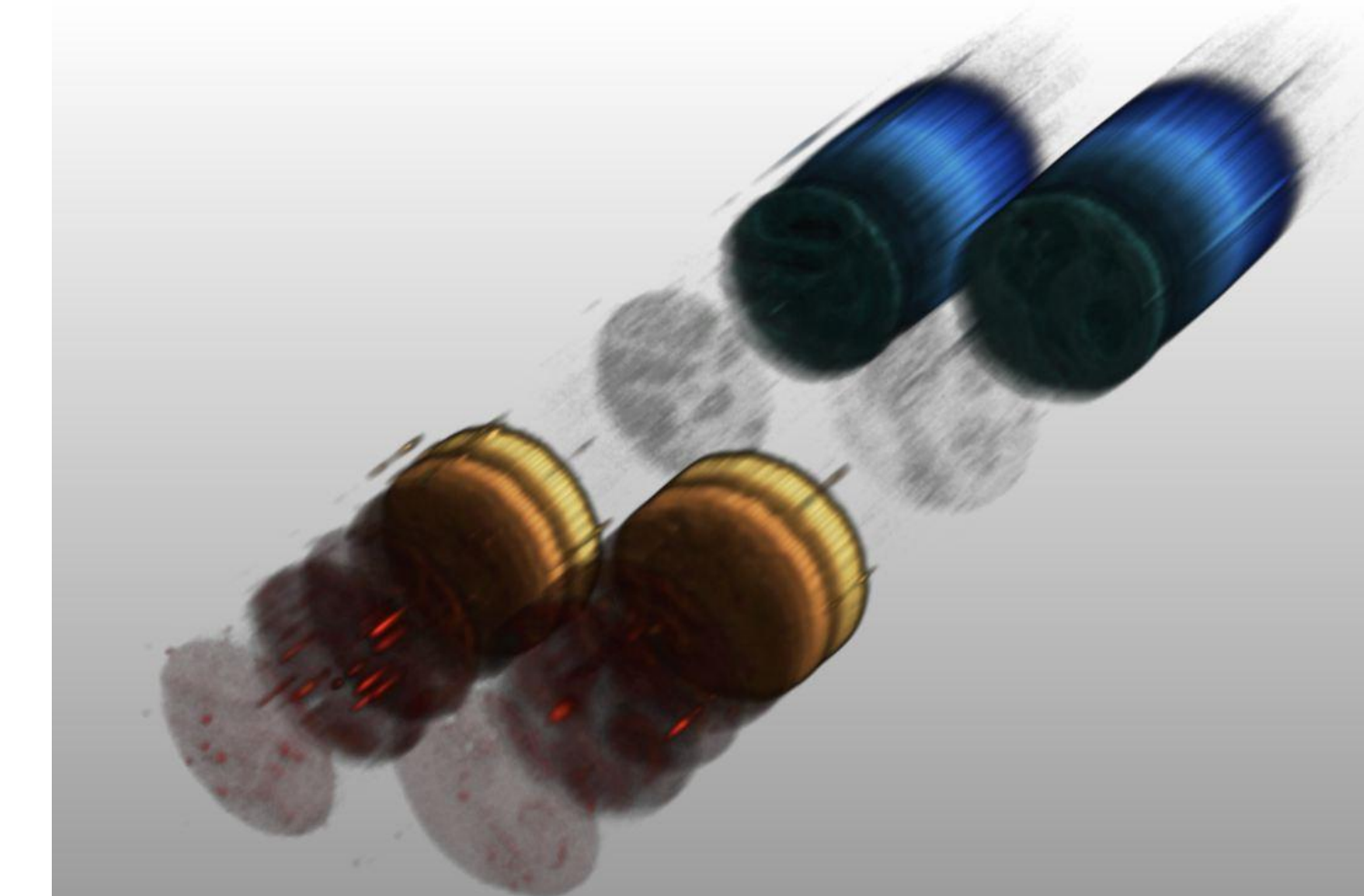
Implementation

Render times are based on performance using an nVIDIA Quadro FX 4800 with a GLSL shader. Frame rates for uniform grids were estimated using 8-bit proxies.

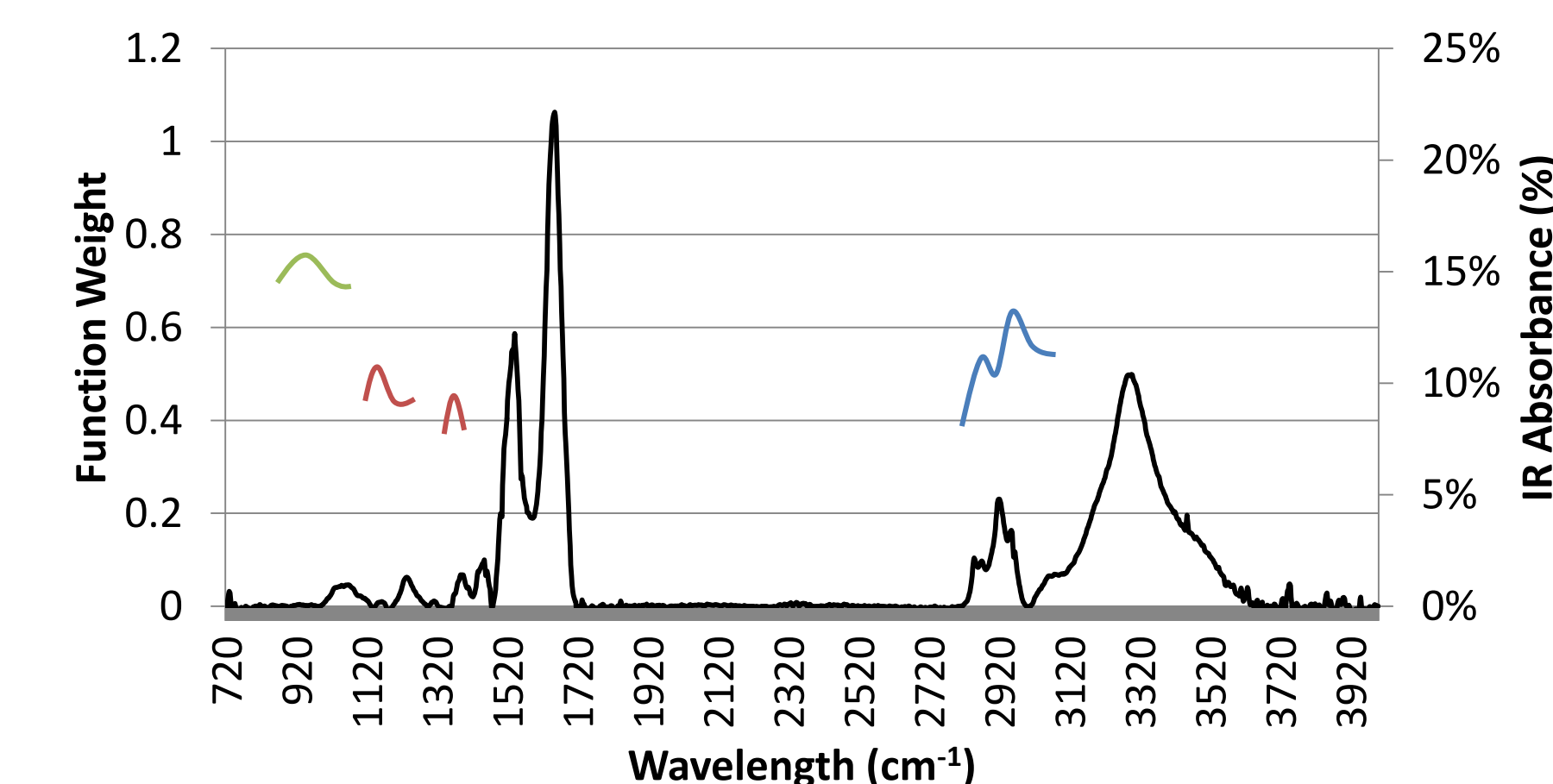
Hyper-Spectral Rendering



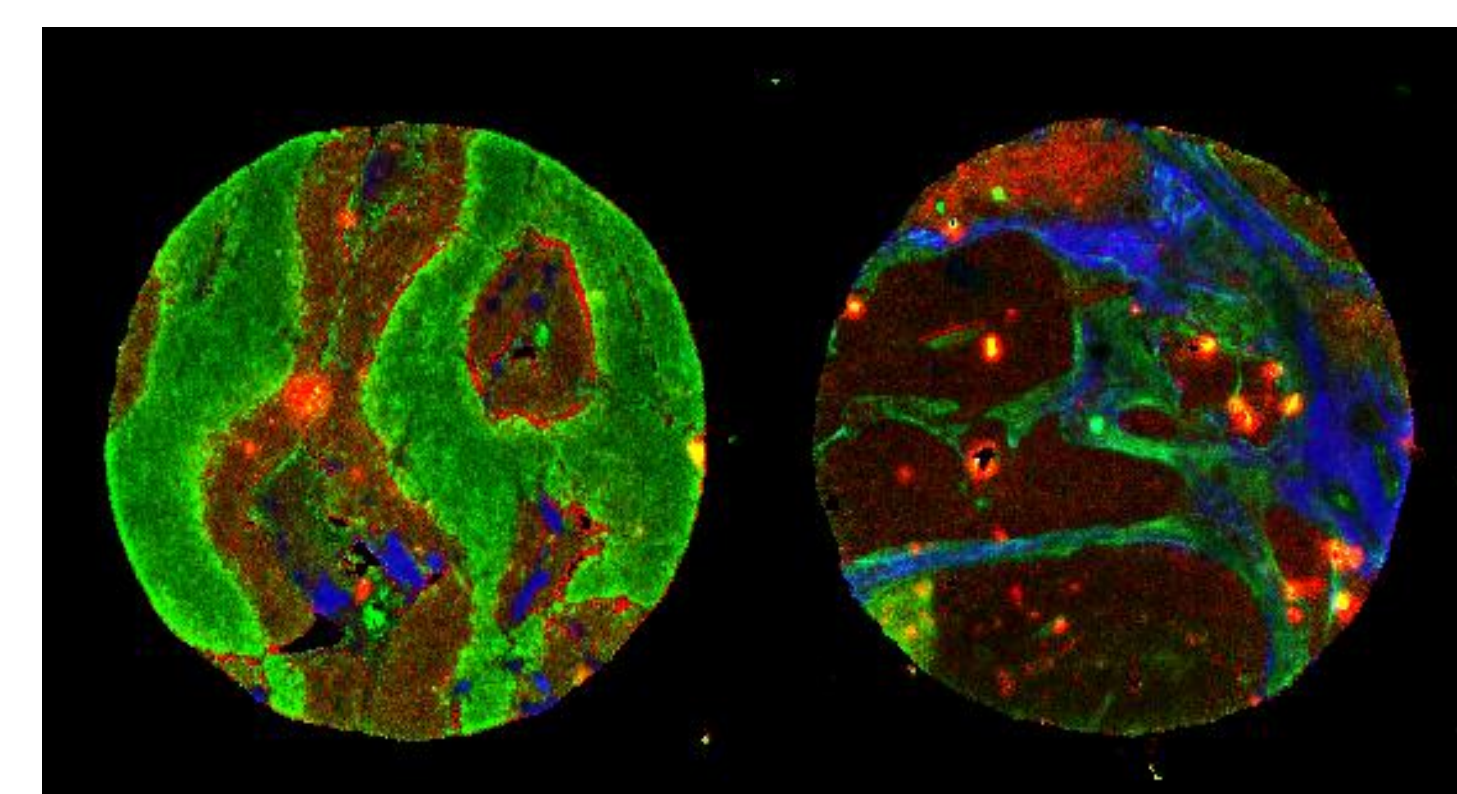
Hyperspectral imaging is commonly used in biomedicine, astrophysics, and chemistry. Methods such as mid-infrared spectroscopic imaging (above) and mass spectrometry create large three-dimensional data sets: 2D images with 1D of spectral data. The spectral dimension represents an array of tissue features at each pixel location. These data sets are generally sparse and can be greatly compressed using DT-Grids and octrees. (top, left) Tissue micro-array of breast tumor biopsies (approx. 80GB). (top, right) Close-up of two biopsy cores (approx. 2GB).



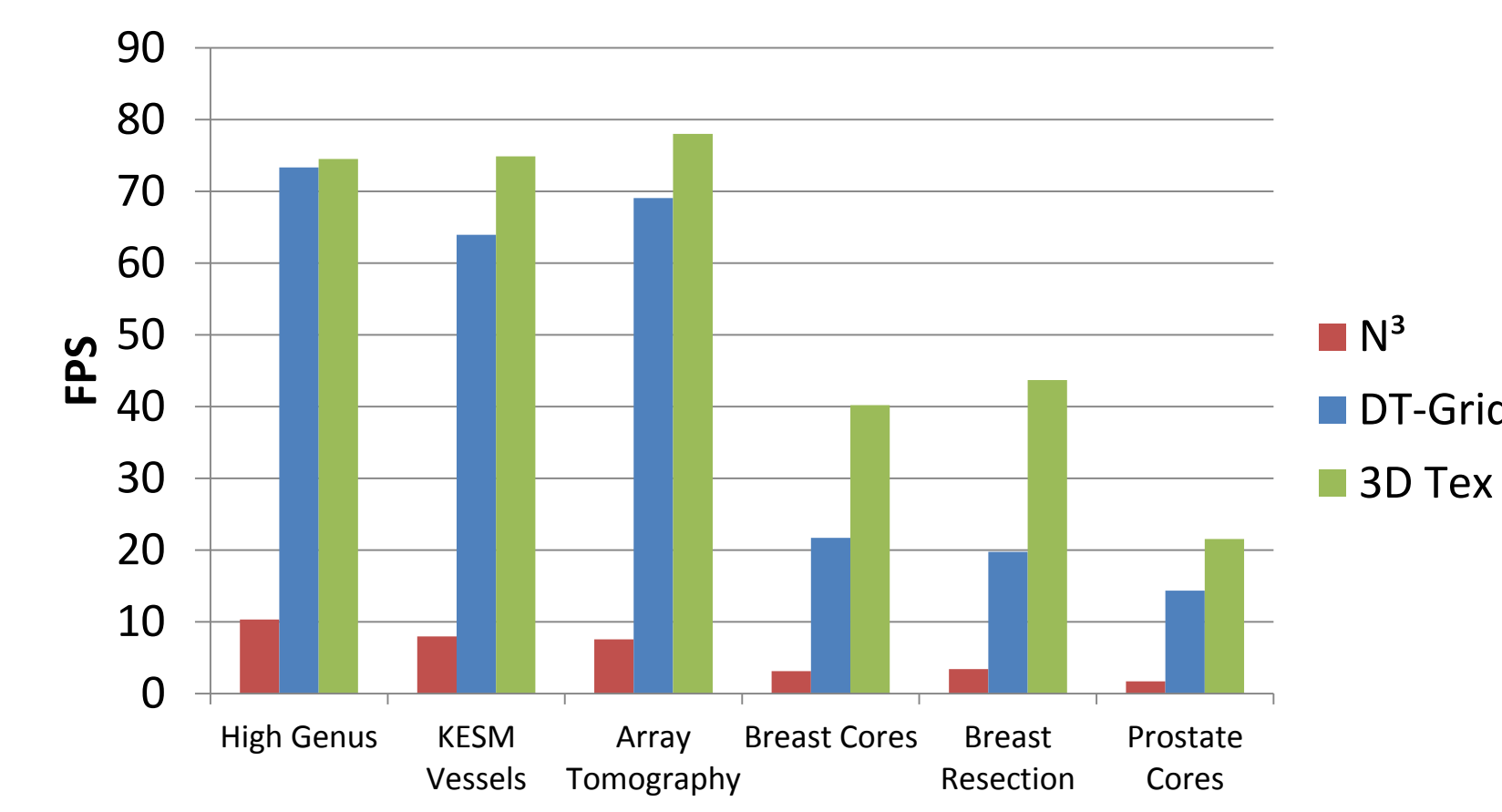
(top) Perspective view of the hyperspectral data for both biopsy cores. High frequencies are shown in blue and low frequencies are shown in red.



(top) Spectral intensities at a single pixel in the breast biopsy array. The red, green, and blue functions are endmember filters that map spectral data to RGB color values for display. Using DT-Grids allow filters to be created and adjusted interactively.



(top) Result of applying interactive filters to the hyperspectral data. Green = collagen proteins, Red = DNA, Blue = lipids. Constructing endmember functions can be done at > 20fps (right).



(top) Frame rates for orthographic rendering. Proxy data sets are shown (see "volume rendering"). Breast cores = 1.3GB, Breast Resection = 1.2GB, Prostate Cores = 2.34GB.

Acknowledgements

We would like to thank Michael Walsh and Rohit Bhargava for their mid-IR spectroscopy data sets and help evaluating our visualization methods. We would also like to thank Steven Smith and Brad Sutton for the Array Tomography data sets. 8-bit data sets were downloaded from volvis.org and contributed by Michael Meibner and Kevin Kreeger, Viatronix, Inc. This work was supported in part by NIH/NINDS grant #R01-NS54252 and the Beckman Institute for Advanced Science and Technology.

References

- Nielsen and Museth. Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *Journal of Scientific Computing*, 26:261-299, 2006.
- S. Lefebvre, S. Hornus, and F. Neyret. Octree textures on the GPU. *GPU Gems 2*, pages 595-613, 2005.
- D. Mayerich and J. Keyser. Visualization of cellular and microvascular relationships. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1611-1618, Dec. 2008.
- C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. GigaVoxels: Ray-Guided streaming for efficient and detailed voxel rendering. *In ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, Feb. 2009.