Gigapixel Scientific Image Rendering

Sponsor Information:

Trent Hare
USGS, Flagstaff, AZ
thare@usgs.gov

Project Description

A gigapixel image is composed of one billion pixels. That's 1000 times the image data captured by a 1 megapixel digital camera. Many scientific images are 10s or 100s of gigapixels in size. A recent image data set was so huge that if it was printed at 300 dpi it would fill a football stadium!

While the amount of pixel data is one issue, another is how pixels are represented. Scientific imagery often contains more bands than the familiar Red, Green, and Blue and color data may be represented in higher bit depth than the familiar 8 bits per band used in most consumer computing applications.

The problem this project considers is how you do you quickly and accurately view images (i.e. downsampling) of such scale and complexity?

One way to approach the problem is using stochastic sampling technique, where only some of a image is sampled to create a reduced representation. In a previous project we explored this technique with extremely encouraging (but preliminary) results. In this project you will build on that work by:

1. Improving on the flexibility (and perhaps performance) of the sampling process,
2. Improving on our understanding and capabilities to tune image "quality" versus speed,
3. Developing a library extension that makes this technique available to GDAL (a popular scientific imaging library) users,
4. Developing a high performance, threaded image viewer based on this technique, which continues to improve image quality as it sits "idle."

The goal of this project is to significantly improve the speed with which scientists are able to manipulate and examine huge scientific image sets. Questions: Can the reader be generalized such that all formats are supported or do changes need to be made per format and can the quality/speed be optimized automatically by the reader for each image?

Knowledge, skills and expertise required for this project

- In-depth understanding of data-structures
- C/C++ (for GDAL, GDAL's API), potentially Python (Numpy, PyQT for the viewer TuiView).
- OpenGL (which can be picked up on the way)
- Access to unix, Windows, and/or OS X

Equipment Requirements

Deliverables:

Github code base implementing solution, Documentation and test cases. Initiate pushing the stochastic reader into GDAL's source (I can help with this as it might need funding).