Visual Presentation of Time-Varying Vector Field Data for Earthquakes Using the Visualization Toolkit

Ben Trube, Jared Coliadis and Han-Wei Shen

1. Introduction

The purpose of this program is to create a method of answering scientific questions about earthquakes through a visual presentation of the data. For this project the particular area of interest is the Whittier-Narrows region of California, and the data is given as time-varying vector fields for a simulated earthquake event. From this initial data properties such as magnitude, normalized vectors, and particle movement through the field are calculated using the methods that are described below. This data is then color-mapped and rendered to a single frame or sequence of frames to create animations for the whole dataset.

2. Desired Information

For this project scientists were asked what sort of questions they would like the data to answer. The following are five questions this program tries to answer:

1. Do the waves in the Whittier-Narrows area follow a pronounced sediment channel defined in the crustal structure?
2. Do waves systematically focus toward the centers of basins, thus providing a physical explanation of the correlation of amplification factors with basin depth?
3. At which locations do the largest conversions between wave-types occur?
4. Which regions produce wave reflections?
5. Do strongly shaken basins act as wave sources?

The methods used to attempt to answer these questions are described in the following sections.

3. Render Methods (General)

Our program uses the Visualization Toolkit (VTK) to process and render the dataset. VTK has many tools for performing operations on the data, as well as methods for rendering the data to the screen, positioning a camera, and exporting frames to common formats such as BMP and JPEG. This allows for the development of more interesting rendering techniques with much of the code for manipulating and viewing the data already written.

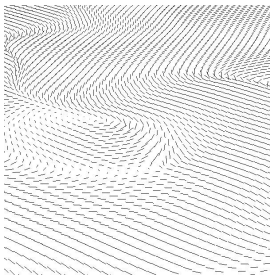
The structure of our code follows an object-oriented approach as well as trying to keep the render methods as separate as possible, while still allowing for a base of common functions that can be used in multiple methods. Libraries for math functions, conversions, and our VTK functions were created to keep to this structure.

Render methods are classified as interactive or pre-rendered. Single frames (one time step typically) tend to be interactive to allow the user to explore that particular time-step from all angles and to allow programmers to choose good camera angles for animations. Renders that involve more of the dataset are exported directly to bitmaps that are later compiled into animations. The exception to this is particle rendering which parses through all of the dataset to create a interactive single frame.

4. Render Methods (Description of Each)

Normalized Vector-Field Slices

Our first method of rendering the data, this method was primarily created as a way of ensuring the floating point data was being input correctly from the binary files. Originally this method was used to render a three-dimensional time-step of the vector field, but the amount of memory required to render this sort of image was unavailable. It was also determined a three-dimensional field was not as easy to visually interpret even with better scaling. Instead a two-dimensional slice of the surface data was taken to give an idea for how the ground was moving at any particular moment of the dataset. An instance of vtkHedgehog was used to show the normalized vectors as each coordinate in the two-dimensional slice. Particle rendering was introduced later as a better way of rendering these vector fields.



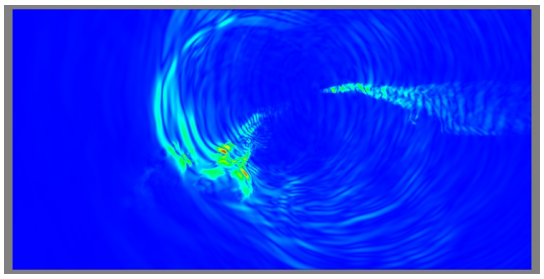
Angle-View of a Vector Field Slice using vtkHedgehog

Scalar Magnitude 2D Slices

In this method of rendering we calculate the normal or magnitude for each vector in the vector field. These values are stored in a vtkFloatArray and color mapped using a vtkLookupTable and a vtkImageData structure. Magnitude data is mapped to color, with blue being low magnitudes down to 0, and red being higher magnitudes. The range for most of these images is between 0 and 0.8 m/s, but narrower ranges provide better views in lower slices. We do this for each time step in the dataset, creating a sequence of frames we then compile into an animation.

This creates an effect much like Doppler radar, which

many people, including non-scientists, can relate to. The first part of the video shows a top-down slice of the wave movement which can be used to tell us several things about the dataset. While this image does not incorporate the sediment data at this time, it does shown the waves following two different channels or lines from the initial source point. The intensity of waves to the west (left) of the screen are of higher intensity and reflect back from about the edge of the screen.



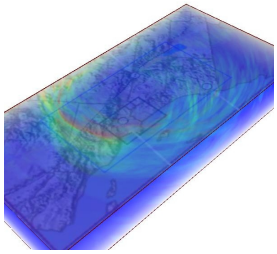
View of Magnitude Surface Slice at Time-Step 80

Magnitude Volume Rendering

The second part of our video highlights volume rendering. As with the two-dimensional version, the velocity normal is calculated for the entire three-dimensional vector field. Instead of taking slices of the data, the data is blended in a three-dimensional volume showing the wave activity throughout the whole dataset. Color ramping is the same as in 2D slicing but instead of being implemented with a vtkLookupTable it is implemented with a vtkColorTransfer function which requires that scalar to color mappings be provided by the programmer for the whole range of colors, rather then generating a table from a range of values. The ColorTransfer function allows for more varied ramping of colors and is necessary to color Volume Renders. The scaling for both of these methods can be adjusted to highlight certain intensities (activity at certain depths is more intense than others). The image is rendered using vtkVolumeTextureMapper3D, a volume rendering implementation using 3D texture mapping.

For 3D volume rendering a texture of the geographical region is applied onto the surface of the 3D volume render. In order to achieve higher resolution, sampling limiters in VTK were switched off in the VTK libraries. Using this method of rendering we can fly the camera around various areas of interest during the quake event, and even get a look inside the volume. By varying opacity features of the magnitude or the stiffness of the ground can be high-lighted, though interleaving of those two types of data has proven difficult in VTK. Both 2D and 3D projections can be used as discussed to highlight other single value features such as x, y, and z components of the vectors and stiffness data.

In addition to providing information about wave reflections and following sediment channels, volume rendering can also show how the waves center around basins, and show how intensity varies and different depths.

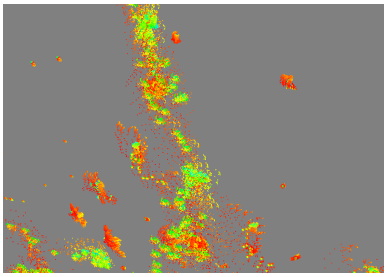


View of Time-Step 8000 using volume rendering

Particle Flow Visualization

This method produces single frames that show particle movement throughout the three dimensional volume. 100-200 particle positions are chosen at random throughout the whole volume. For each time step a bounding box is created where vector magnitude is greater than 0. If a particle lies within this bounding box a new particle is dropped and its movement caused by the vector field is tracked through each time step. Because of the small velocities of particles these velocities are increased by a factor of 25,000 in order to see the path without overlap. Some still shots of these renders are included in the third part of our video, and can show how seismic waves effect ground movement throughout the dataset.

Nearest neighbor calculations are used to determine the vector for each individual particle, but in order to achieve a more accurate result trilinear interpolation can be incorporated. Sampling particle movement can also show particle movement without needing as larger magnification factors. These methods replace the vtkStreamline functionality which did not have the ability to deal with time-varying vector fields.



Zoom of particle flow visualization after 227 time steps

5. Conclusion

The Visualization Toolkit provided many tools for rendering the data quickly but did require modification and in some cases replacement in order to render good images. The methods described above provide a preliminary basis for understanding the data from which new methods can be created.