VisContest: Visualization of the Terashake Dataset

Ralf Sondershaus[*]
WSI / GRIS, University of Tübingen, Sand 14, 72076 Tübingen, Germany

Wolfgang Straßer
WSI / GRIS, University of Tübingen

1 Introduction

We use direct volume rendering to explore the dataset and to find answers for the questions of the contest. Although the visualizations of the Terashake Visualization web page also shows movies that are created by direct volume rendering, we differ in the chosen classifications, in the selection of the intervals of the Terashake data values that are visualized, and in a computed new dataset that approximates acceleration (i.e. how the velocity changes over time).

Instead of using an existing visualization package (like VTK) we decided to use our own implementation of a volume renderer for structured grids. Although it might not have the visual quality of other packages, we can easily integrate own shaders and use them to render the Terashake Dataset in the way we want it to be visualized.

The volume renderer basically visualizes a volume with viewplane-aligned slices which are rendered from back to front and textured by a 3D texture. Our volume renderer can read 3D textures of different resolutions and data types. But to support the largest possible texture, we restrict the renderer to read textures of unsigned bytes only. Using such textures only, the graphics card of our computer supports a 3D texture of up to $512 \times 512 \times 128$ voxels which covers most parts of the Terashake dataset and frees us from segmenting the dataset into a series of 3D textures (as it would be done by more sophisticated renderes of course). We placed the texture into the northwestern area of the dataset and skipped the southeastern area.

We implemented a collection of programs that convert the original data into our volume representation. Mostly, the x,y, and z components of the velocity vector are converted independently. Firstly, the original floating point data format of the dataset can be transformed into an unsigned byte representation. The user can choose an interval I of the original data that is mapped onto the [0,255] unsigned bate interval. The most interesting parts of the dataset lie within the interval $[-0.1f, 0.1f]$ (as described later) which are mapped such that 0 becomes 127. All values that fall outside of I are clamped to the interval borders.

Secondly, the acceleration can be approximated by simple differences between the values of two consecutive time steps, i.e. $\mathbf{a}_t = \mathbf{v}_t - \mathbf{v}_{t-1}$. Thirdly, the values at a specific position can be written into an ASCII file and displayed like seismograms (using gnuplot for instance) which helped us a lot to find out what's going on in the dataset.

In order to get an orientation within the dataset, the volume renderer can show meta information like streets and borders of California as well as a simple map (texture) that we cut out of the VisContest web page and cropped to align with the dataset. The streets and border information is read directly from the ASCII files that are provided at the VisContest web page.

2 Basin Rendering

Instead of extracting isosurfaces for the basins, we store the material stiffness (VS value) in a 3D texture (unsigned bytes again). The minimal and maximal values are mapped to 0 and 255, respectively. This texture is called masking texture because the fragment shader uses it to mask different areas of the volume.

A fragment shader accesses the masking texture and merges its values with the values of the terashake dataset. This enables us to steer how the dataset values are rendered inside a basin and outside a basin. Normally, the values that are inside a basin are rendered with full colors while the values outside a basin are rendered a little bit darkened where the darkness factor can be freely chosen by the classification of the masking texture.

Furthermore, a meta texture is rendered on top of the visualization and shows the basins as red areas. This texture is rendered with 90% transparency and just hints the landscape and the basins.

3 Wave Types

This was the most complex part because wave types interfere with each other. We analyzed the dataset at different locations and recorded the seismograms as shown in figure 1. Using these seismograms as a hint, we rendered all X velocity values of the interval $[-0.1, \ldots, 0.1]$ only (X values that are greater or less than 0.1 or -0.1 respectively are clamped) and colored the different areas using a cartoon-like coloring classification (i.e. discrete color steps). Thus, we distinguish the wave types by their amplitudes and choose the classification carefully. The seismograms give hints about the intervals that separate the colors of different wave types in the cartoon-like rendering. We found a sepatation value of 0.01 plausible.

The selected classification shows values above 0.01 in bright red and values below -0.01 in dark blue while the values inbetween use the colors green, yellow (0-values), and light-blue, see figure 2. Note that the basin rendering further reduces the brightness of the colors of all values that are outside of a basin. The values near zero are rendered fully transparent in order to not disturb the visualization while all other values are rendered semi-transparent.

Although this might not correspond to the correct wave types, the different colors give hints about the changes of the type of a wave. Red and dark-blue colors correspond to great amplitudes and thus likely to Rayleigh- and Love-waves while green and light-blue colors correspond to small amplitudes and thus P- and S-waves.

We also tried to get the frequency out of the dataset (using Fourier or Windowed Fourier Transformations) but found this too time-consuming and less practicable (although the computed frequencies show up clearly as high frequencies at the beginning and mixed high-low frequencies in the middle and high frequencies at the end again; computed using Maple and a windowed Fourier transform of the seismogram of figure 1, but not added to this material).

4 Reflections

The volume renderer shows the reflections that basically originate at the borders of the basins. The most effective visualization that we found used the acceleration datasets although other datasets and classifications also show this effect. Especially waves with large

[*]e-mail: sondershaus@gris.uni-tuebingen.de

amplitudes and low frequencies get reflected when they hit the basin boundary. Again, the acceleration values are mapped from the interval $[-0.1, ..., 0.1]$ to unsigned bytes.

If we compute the differences of the velocities between two consecutive time steps, we get something like acceleration that swings around zero with both negative and positive values as shown in figure 1b. Because these values swing around 0, they can be easily separated by a classification that nevertheless has to separate the value 0. So, using the acceleration instead of the velocity values shows a more detailled image and highlights all changes of the values. Note that for instance the x velocity is always below zero at the first time steps when the ground starts to shake (around time=50) which is not easily separatable by a classification.

5 Wave Guides and Basins

The waves seem to highly interact with the borders and interior structures of the basins. This causes them to change their speed and to interfere with consecutive waves. Using a zoom view into the most interesting regions, our volume renderer shows how the waves break apart and change their direction. This happens at most in the Los Angeles area and especially the Whittier-Narrows region which can be located by the streets that are rendered on top.

We again render the X and Z velocity using the cartoon-like classification. The colors show all areas that still contain waves with high amplitudes as well as all areas that contain waves with low amplitudes. The basin rendering furthermore highlights the colors of the basins such that the visualization can show if such high-amplitude waves are only in the basins (this is the case if all darkened areas do not have red or dark-blue colors).

6 Demo Program

The demo program needs support for multi texturing, 3D textures, and ARB vertex and fragment shaders. Otherwise, the program will fail or render wrong pictures.

You can open a volume by clicking File - Open Volume. Similarily, a masking texture can be opened by File - Open Mask Volume. The meta files can be opened by File - Open Meta Streets (streets and border, `*.asc` files or by File - Open Meta File (texture). The meta information can be rendered transparent which is steered by the parameter window (VolumeSlicer - meta information - color - fourth component).

Every classification (volume and mask) can be adjusted by the appropriate tab in the classification window. The mask texture only reads the a-component (r, g, b are unused). Click on update to make the changes current or click the ”always” check box to immediatly see the changes.

The path to the volume files as well as their name schemes can be set in the parameter window. The masking shader can be switched off by a check box in order to use the ”normal” volume shader again. If no masking texture is loaded, the ”normal” volume shader is used.

We used the file `TS21VelocityMesh_VS_R2` as masking texture.

The volume renderer is part of a big graphics library and the source code cannot be distributed yet. Nevertheless, the source code of the data converter is accompanied with the submission.

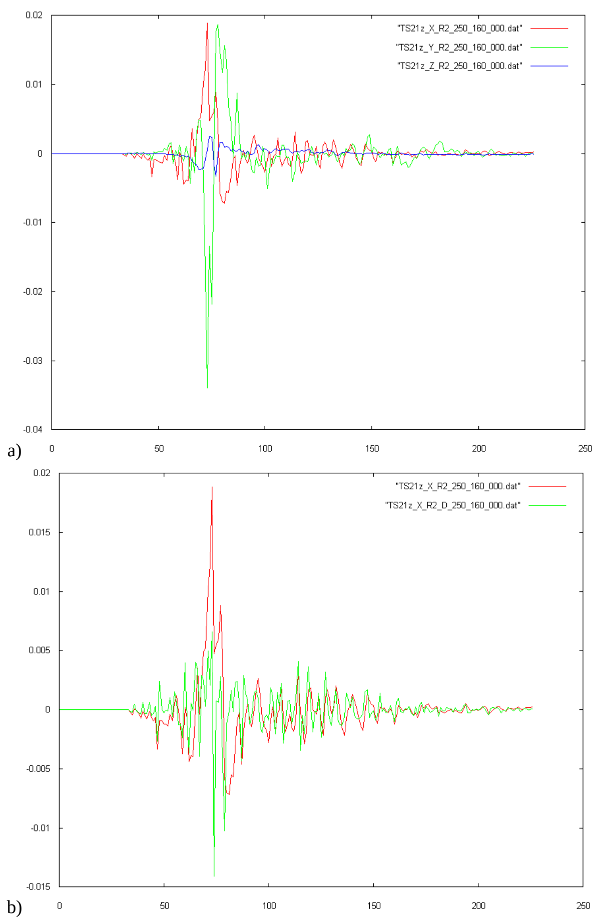

Figure 1: The seismogram at the position $x = 250, y = 160$ [voxel] that is recorded at the surface ($z = 0$). (a) shows the x,y, and z velocities while (b) shows the x velocity (red) and its approximated derivative, the acceleration (green). Note how the acceleration always swings around zero while the x velocity does not. We used such seismograms to get an idea about the dataset and adjust the intervals that are to be rendered.

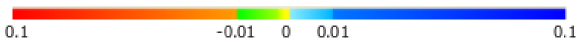

Figure 2: We used this cartoon-like classification. The original data values between $[-0.1, ..., 0.1]$ are mapped and all values that are outside of this interval are clamped to -0.1 or -0.1, respectively.