Visualizing 3D Earthquake Simulation Data

As exemplified in a state-of-the-art bridge-foundation-ground model simulation, a suite of new visualization techniques let scientists study seismic waves and interactively investigate and explore their data. In so doing, the techniques further scientific understanding and thus facilitate the development of new methods to protect real-world infrastructures against otherwise devastating earthquakes.

Earthquakes are natural phenomena caused by the sudden release of tension from the rocks in the Earth's crust and upper mantle, resulting in potentially devastating seismic waves. For example, on 9 January 2010, a strong earthquake—6.5 moment magnitude (Mw)—shook offshore of Humboldt County closed to Eureka in northern California, causing several injuries and damaging 463 buildings. Thereafter, on 12 January 2010, a catastrophic 7.0 Mw earthquake struck Port-au-Prince, Haiti, with the epicenter located at the boundary region of the North American and Caribbean plates; an estimated 230,000 people died and 250,000 residences collapsed or were severely damaged. Three months later, a 7.1 Mw earthquake rocked the Qinghai Province in China on 14 April 2010. The epicenter was located approximately 30 kilometers from Jiegu township at a shallow depth of 10 km. In Yushu county, 15,000 residential buildings were toppled, killing 2,200 people, injuring another 12,000, and leaving more than 100,000 people homeless.

Since the turn of the 20th century, there have been more than 100 earthquakes in global urban areas with a magnitude of 6.0 Mw or greater; each resulted in the deaths of thousands of people and significant property damage. However, earthquakes can't be prevented; instead they must be prepared for. Scientists are thus using computer simulations to better understand the underlying mechanisms of an earthquake's seismic waves to build safer structures and thus minimize the potential damage that an earthquake can cause.

Recent advances in computational methods and supercomputing technology enable large-scale, realistic simulations of seismic wave propagation and the seismic response of ground structures. However, scientists have comparatively few visualization and data analysis tools adequate for studying the vast amounts of data output from their simulations. Simple direct-volume data visualization is ineffective for observing the complex combination of different motions that scientists look for in their simulations. To efficiently and properly examine their data, scientists need a solution for teasing out and isolating these complex behaviors while at the same time allowing for quick and easy comparisons.

As the "Related Work in Seismic Data Visualization" sidebar describes, our work supplements existing methods for visualizing seismic data and incorporates advanced rendering techniques

1521-9615/11/$26.00 © 2011 IEEE
Copublished by the IEEE CS and the AIP

Cheng-Kai Chen, Chris Ho, Carlos Correa, and Kwan-Liu Ma
University of California, Davis
Ahmed Elgamal
University of California, San Diego

Related Work in Seismic Data Visualization

Our work builds on several existing methods for visualizing seismic data.

Robert H. Wolfe and C.N. Liu introduced an interactive technique for examining seismic data obtained from ultrasound reflections.[1] Their approach interprets seismic data with a volumetric scheme, removes noisy portions, and shapes the pulse waveforms.

Tung-Ju Hsieh and his colleagues created three fundamental components for visualizing large-scale seismic data, which include an out-of-core approach using view-dependent mesh refinement, terrain mapping, and rendering of earthquake datasets.[2] Their work improves the rendering capacity and facilitates the exploration of large-scale seismic dataset analysis.

Daniel Patel and his colleagues introduced techniques for visualizing interpreted and uninterpreted seismic data.[3] They adopted an illustrative technique to render the interpreted data as if they were geological illustrations, while uninterpreted data was rendered in a color-coded volume. They discussed how to combine the two representations so that users could control the balance between the two visualization styles accordingly.

References

1. R.H. Wolfe and C.N. Liu, "Interactive Visualization of 3D Seismic Data: A Volumetric Method," *IEEE Computer Graphics and Applications*, vol. 8, no. 4, 1988, pp. 24–30.
2. T.-J. Hsieh, F. Kuester, and T.C. Hutchinson, "Visualization of Large-Scale Seismic Field Data," *Proc. High Performance Computing Symp.*, ACM Press, 2003, pp. 163–170.
3. D. Patel et al., "Illustrative Rendering of Seismic Data," *Proc. Vision Modeling and Visualization*, IOS Press, 2007, pp. 13–22.

to enhance the data's spatial and temporal structures. Our approach offers a new interactive visualization capability for earthquake scientists to better analyze and understand the intrinsic nature of seismic propagation. The basis of our approach is to decompose the data into different frequency bands so features of interest can be isolated or compared to each other. The data used in our work was obtained from a state-of-the-art bridge-foundation-ground model simulation, based on the real-world Humboldt Bay Middle Channel (HBMC) bridge in Eureka, California. The simulation takes into account many subtleties of the terrain around the bridge—such as soil composition—to be as accurate as possible. With our techniques, the overall seismic response behavior is enhanced for improved understanding.

Our visualization approach has several stages: frequency decomposition, resampling, and then direct-volume rendering and/or deformable implicit surfaces. By combining the deformable implicit surfaces and direct-volume rendering, we can observe two different motions separated by frequency bands while minimizing confusion between the two, allowing scientists to better comprehend the complex and previously obscured spatial and temporal relationships between the data contained in the frequency bands. Our visualization approach can be immensely helpful in validating simulation output, as well as in providing scientists with new insights to help them better model seismic phenomena. This will hopefully lead to sturdier structures and a safer future.

Earthquake Simulation: Humboldt Bay Middle Channel Bridge

The seismic data used in this work is from the earthquake simulation of the HBMC bridge performed by one of our coauthors (Ahmed Elgamal) and his colleagues.[1]

As Figure 1a shows, the HBMC bridge was built in 1971 near Eureka in northern California over waterways crossing Woodley Island and Indian Island. The bridge site is located in an area of complex tectonic interaction among the Gorda, North American, and Pacific Plates. The Little Salmon fault, which is classified as one of the principal active faults in California, is the nearest seismic source; it's located about 5 km from the site and is capable of generating a maximum credible earthquake of 7.5 Mw.

The HBMC bridge is 330 meters long, 10 m wide, and 12 m high (the average height over the mean water level). The bridge has a nine-span superstructure consisting of four precast, prestressed concrete I-girders and cast-in-place concrete slabs. As Figure 1b shows, a 3D finite element (FE) model of the bridge and nonlinear solution strategy are built with the Pacific Earthquake Engineering Research Center's open-source software platform, Open System for Earthquake Engineering Simulation (OpenSees).[2] The spatial extent of the bridge-foundation-ground simulation domain is 650 m long, 151 m wide, and 74.5 m deep, and the bridge model contains the superstructure, piers, and supporting piles. The whole FE model includes 28,431 nodes and 141,336 tetrahedra. To analyze the system's time-dependent nonlinear seismic response, we executed 1,130 time steps

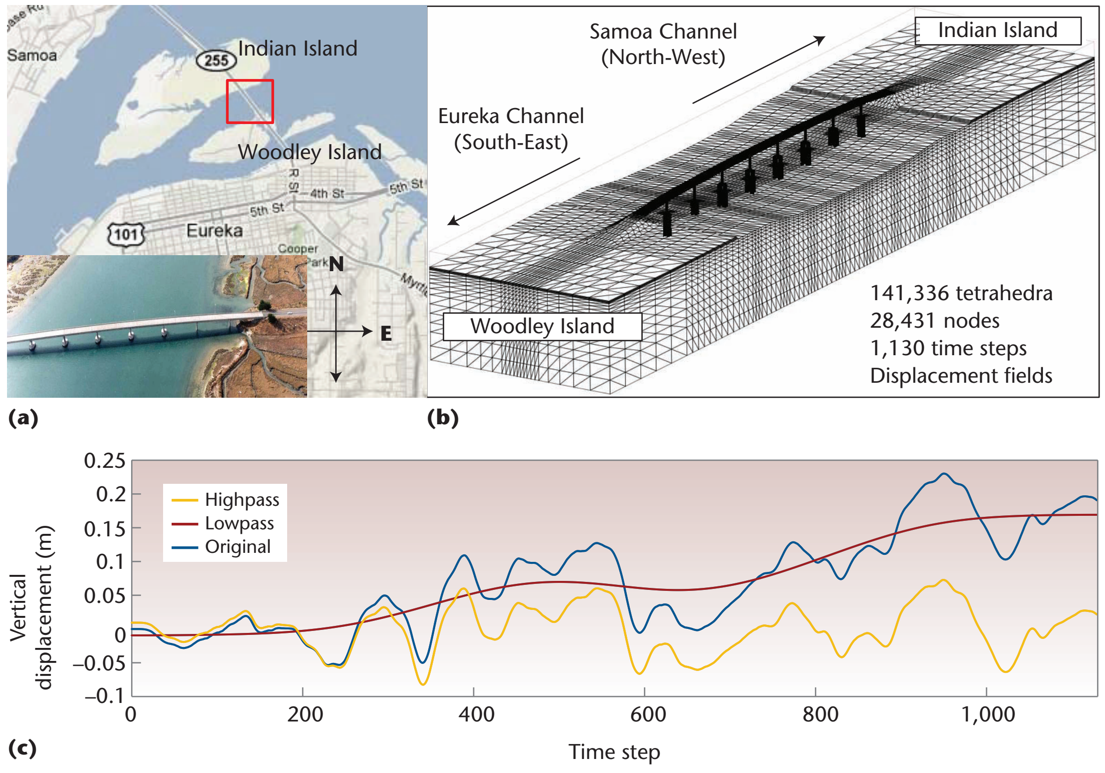


Figure 1. Earthquake simulation of Humboldt Bay Middle Channel (HBMC) bridge. (a) Satellite and aerial views of the bridge (from a Google map and the California Department of Transportation). (b) A 3D finite element (FE) model of the HBMC bridge-foundation-ground system. The simulation domain is 650 meters long, 151 m wide, and 74.5 m deep. The bridge model contains the superstructure, piers, and supporting piles. The entire FE model includes 28,431 nodes and 141,336 tetrahedra. We executed 1,130 time steps ($\Delta t = 0.02$ s) of transient dynamic nonlinear analysis to evaluate the system response during 24.75 s of earthquake excitation. (c) Vertical displacement time histories of the original data, low-pass, and high-pass parts at one of the supporting piles in southeast side. Cutoff frequencies for low-pass and high-pass filters are 0.1 and 0.3 Hz, respectively. After applying frequency analysis, drifting and cyclic components are extracted from the original data, and corresponding slumping and shaking motions are revealed explicitly.

($\Delta t = 0.02$ s) of transient dynamic nonlinear analysis to evaluate the system response during 24.75 s of earthquake excitation. The simulator took about 40 hours to generate the results for all time steps, and the total size of the data is 1.3 Gbytes.

The simulation data contains 3D displacements with coherent and incoherent noise signals. The displacement on each node is a 3D vector representing movements caused by complex incident wave motions such as drifting and shaking motions under the surface. Drifting is a permanent deformation that's usually contained within the low-frequency band, and is the product of system slumping, settlement, and/or heaving. Shaking is the cyclic motion of seismic displacements contained in the high-frequency band, seen as a back-and-forth movement. The dynamic shaking motions are superimposed with accumulated drift values, and it's therefore difficult to distinguish their different contributions. To provide an intuitive visualization of the simulation, we must first perform seismic analysis on the displacement fields to extract intertwined motions.

Seismic Data Analysis

Seismic analysis has been widely applied to a range of problems, including reservoir characterization, enhanced oil recovery strategies, and earthquake investigation. Frequency-time (F-T) analysis is a technique for manipulating signals with frequency components that vary over time. It's commonly used to separate data with different frequency components, and becomes an important technique in interpreting seismic data. All the F-T analysis methods include delineation of sequence interface and determination of seismic sequence cycles.

We use a fourth-order Butterworth filter to separate different band-pass data.[3] The Butterworth filter can be used as a digital filter in motion analysis and is designed to flatten frequency responses in pass bands. Compared to other filters, the

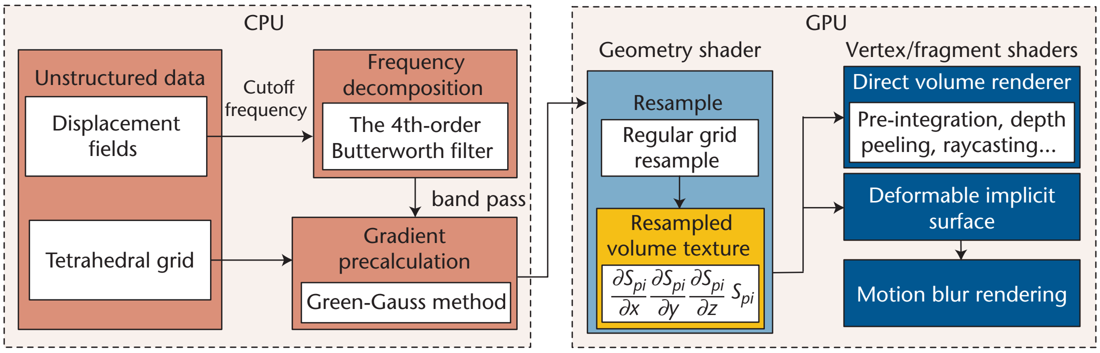


Figure 2. The visualization process. This process consists of frequency decomposition, gradient precalculation, resampling, direct-volume rendering, deformable implicit surfaces, and motion-blur rendering. Both frequency decomposition and gradient precalculation are executed in the CPU side once the cutoff frequency is determined. After gradient precalculation, the data is sent to the GPU for resampling and rendering. The geometry shader resamples the unstructured tetrahedral grid to a regular grid, and encodes all the relevant information of each node as a 3D texture. Then, the texture is sent to the vertex and fragment shaders for visualization.

Butterworth filter rolls off more slowly around the cutoff frequency while showing no ripples, which is a desirable combination for our analysis. A typical Butterworth filter is a low-pass filter. However, by removing the low-pass part from the original signals, the filter can be used as a high-pass filter. A low-pass filter passes relatively low-frequency components in the signal and rejects the high-frequency components. In other words, a low-pass filter drops frequency components higher than a cutoff frequency. A filter's behavior can be summarized by the *frequency response function*; for the Butterworth filter, that function is

$$|H_c(j\omega)|^2 = \frac{1}{1+(j\omega/j\omega_c)^{2N}},$$

where $j = \sqrt{-1}$, ω = frequency (radians/s), ω_c = cutoff frequency (rad/s), and N = the filter's order.

We first decompose 3D displacements to x, y, and z components, representing longitudinal (parallel to bridge traffic), transverse, and vertical directions. We then apply the Butterworth filter along the temporal dimension for each individual component on all 1,130 time steps. The suggested cutoff frequencies from scientists are 0.3 Hz and 0.1 Hz for extracting high-pass and low-pass data, respectively.

Figure 1c shows vertical displacement time histories of the original data, low-pass, and high-pass parts at one of the southeast side's supporting piles. With proper cutoff frequencies, drifting and shaking components are extracted, and corresponding slumping and cyclic motions are explicitly revealed. This figure also indicates that the high-pass part becomes increasingly small compared to the low-pass part as the shaking dissipates. Therefore, we are faced with a 3D time-varying multivariable data visualization problem, where different band-pass data are in different scales.

Methodology

We developed and integrated a set of techniques for earthquake scientists to explore and use in examining their seismic datasets from simulations.

Figure 2 shows an overview of the resulting visualization process. Besides frequency analysis, the process also consists of gradient precalculation, resampling, pre-integrated direct-volume rendering, implicit surface deformation, and motion-blur rendering stages. Both frequency decomposition and gradient precalculation are executed in the CPU side once the cutoff frequency is determined. After gradient precalculation, the data is sent to the GPU for resampling and rendering. The geometry shader resamples the unstructured tetrahedral grid to a regular grid, and encodes all the relevant information as a 3D resampled texture. Then, the texture is sent to the vertex and fragment shaders for visualization. We now describe each of these steps in detail.

Gradient Precalculation and Resampling

To achieve high-quality rendering at interactive frame rates, we resample the unstructured tetrahedral meshes into a regular grid. Figure 3 shows an overview of our resampling procedure.

Figure 3a shows a high-level picture of the transition from tetrahedral meshes to the regular meshes, and Figure 3b shows the details of each procedure in the resampling process. First, the

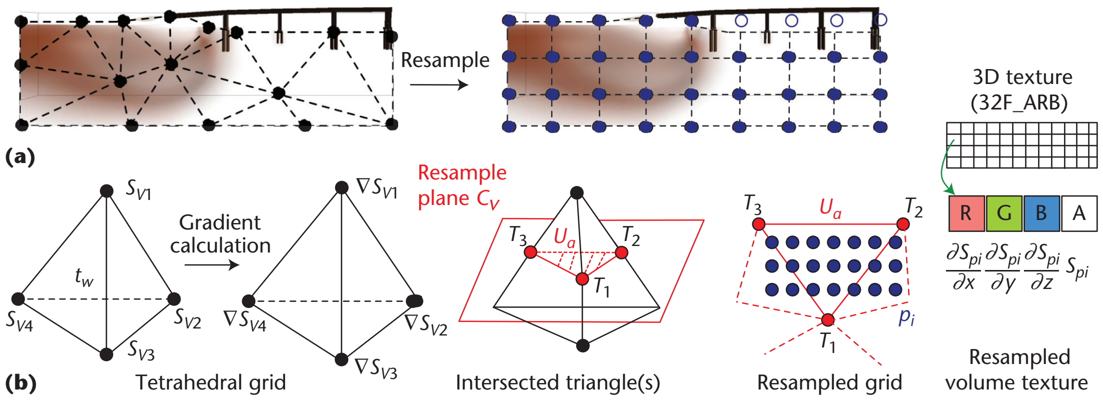


Figure 3. An overview of the proposed resampling procedure. (a) A high-level picture of the transition from tetrahedral meshes to regular meshes. (b) Details of each step in the resampling process. When users select a time step or band pass to visualize, we calculate gradients of the component on every grid node using the Green-Gauss theorem. Then, each tetrahedron is passed to a geometry shader as a single point with eight attributes, including four node locations and their corresponding packed gradient and signal values. This tetrahedron is then tested against the current cutting plane, and—if an intersection is found—a triangle is emitted, using linearly interpolated vertex values at the intersection points.

```
$B$:= Bounding volume
$T$:= Tetrahedra mesh
$M$:= Resampled 3D volume texture
$S_x$:= Signal value at node $x$
for all cutting plane $c_v \in B$ do
    for all tetrahedron $t_w \in T$ do
        $u_a \leftarrow Intersection(c_v, t_w)$
        if $u_a$ is a triangle then
            $\{p_i\} \leftarrow ResampleTriangle(u_a)$
            for all $p_i$ do
                $e_i \leftarrow CorrespondingMemoryLocation(M, p_i)$
                $e_i[0] \leftarrow \partial S_{pi}/\partial x$
                $e_i[1] \leftarrow \partial S_{pi}/\partial y$
                $e_i[2] \leftarrow \partial S_{pi}/\partial z$
                $e_i[3] \leftarrow S_{pi}$
            end for
        end if
    end for
end for
```

Figure 4. The pseudocode for the resampling scheme. Resampling occurs for each time step and each time users choose to visualize a different component or band pass.

gradients are reconstructed at each node via the Green-Gauss method.[4] To rapidly resample our tetrahedral meshes, we use the GPU to define a volume-bounding box for the entire tetrahedral mesh, and slice the entire mesh with cutting planes into structured points. We use a $512 \times 256 \times 256$ bounding volume B to resample our tetrahedral mesh T, which contains 28,431 nodes and 141,336 tetrahedra. Each tetrahedron $t_w \in T$ is passed to the geometry shader with all the relevant information: the four node locations, and their corresponding gradient and signal values. The tetrahedron is then tested against the cutting plane $c_v \in B$. If an intersected triangle u_a is found, we use linear interpolation to obtain vertex values at resampled points $\{p_i\}$ within the triangle u_a, and then emit the intersected triangle u_a.

Assume S_x represents the signal value at node x. Each resampled point p_i contains four vertex values, a signal value S_{pi}, and the corresponding derivatives, $\partial S_{pi}/\partial x$, $\partial S_{pi}/\partial y$, and $\partial S_{pi}/\partial z$. Those points are then copied into our resampled volume texture, which has the same dimensions of the volume-bounding box. Notice that the internal format of our resampled volume texture is `32F_ARB`, which has four floating-point channels. Therefore, for each resampled point, we store each vertex value to each channel in the corresponding location of the volume texture. This process is repeated for each cutting plane in our resampled texture. Resampling occurs for each time step and each time users choose to visualize a different component or band pass. Figure 4 shows Algorithm 1, which is the resampling procedure's equivalent pseudocode.

Direct-Volume Rendering

Now that we have a resampled volume stored as a regular grid, we can use direct-volume-rendering methods for multivariable visualization. To maintain interactive rates while ensuring high-quality images, we use a hardware-accelerated raycasting algorithm following the implementation of Jens Kruger and Rüdiger Westermann.[5] In our implementation, rather than using a simple bounding box as our starting fragments, we use tetrahedral meshes to tightly fit our area of interest. Because of concavities in the tetrahedral mesh, it's also necessary to perform the depth peeling method[6] for empty space skipping. The rendering's quality is proportional to our resampled grid's resolution.

Figure 5 compares images between the renderings of two resampled volumes using low (128 × 64 × 64) and high (512 × 256 × 256) resolutions, and the arrows indicate the notable visual differences. The high-resolution volume can capture subtle changes in the original grid, but the low-resolution one takes less time to render. In our system, the default resampling resolution is 512 × 256 × 256, and is user configurable.

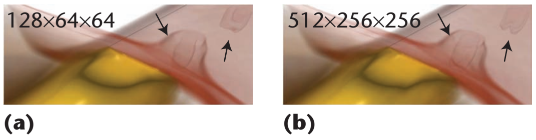


Figure 5. Comparison images between the renderings of two resampled volumes using low (128 × 64 × 64) and high (512 × 256 × 256) resolutions. The arrows indicate the notable visual differences. The high-resolution volume can capture subtle changes in the original grid, but the low-resolution one takes less time to render.

Deformable Implicit Surfaces

With direct-volume rendering, displacement fields are mapped to color and opacities. However, because of the additive nature of color in the composition process, it becomes increasingly difficult to visualize patterns of interest as the different variables quickly occlude each other. As an alternative, we use deformable implicit surfaces to modulate the opacity of one of the components.

Using implicit surfaces offers several advantages over geometry or using a tiling texture. Implicit surfaces are easily integrated into the volume-rendering loop, allowing complex transparency effects at a low constant cost. Geometry, on the other hand, would require several rendering passes to achieve complex transparency effects. Another approach for achieving a similar effect would be to use a separate 3D modulation texture. This would allow for complex shapes, but would require another texture fetch in the main volume-rendering loop. However, we found that simple surfaces provide the most intuitive visualization for displacements because it's easier to correlate a complex deformed surface with a simple base surface.

Let $D(x)$ be a function returning the displacement at point x. If we define the distance to the nearest surface function as $S(x)$, and surface thickness as s_w, we can define the modulation function M as $M(x) = H(s_w - S(D(x))$, where H is the Heaviside step function. After sampling the displacement value for a location in the volume-rendering loop, we use that to displace the sample location for the modulation function, resulting in deformation. For our particular implementation, we used sets of planes, which greatly simplifies $S(x)$.

Lighting

For volume rendering, lighting is done using a pre-integrated method to achieve high-quality volume visualization. We use methods by Eric B. Lum and his colleagues[7] to speed up lookup table generation and minimize lighting artifacts. By using our precalculated and stored gradients, six extra texture fetches can be saved in our shader program. Moreover, for our deformed implicit surfaces, we needed the Jacobian matrix to properly deform a surface normal. In most cases, filling the Jacobian matrix is trivial because the displacement's gradient is included in the step to find the displacement amount, and the displacement affects only one x, y, or z component at a time.

Motion Blur

Deformable implicit surfaces give a good overview of how much the surfaces have been displaced, but provide a poor overview of the displacement's direction over time. Therefore, we use a motion-blurring technique based on either the gradient of the displacement over time or the current displacement. The basic idea is to leave some color behind when the plane moves past the current point. We first find the signed distance to the nearest surface, $S_s(x)$. Then, we displace the point and find its new location, and check if the sign has changed. If it has, we divide that signed distance by the displacement distance, and square it to create a quadratic falloff for the blurring function. Thus, the blur modulation function B becomes

$$B(x) = \begin{cases} \left(\dfrac{S(x)}{D(x)}\right)^2 & \text{if} \quad S_s(x) \neq S_s(x - D(x)) \\ 0 & \text{if} \quad S_s(x) = S_s(x - D(x)) \end{cases}.$$

Seismic Data Visualization

In our seismic data, the overall displacement can be decomposed into two main motions: drifting and shaking. Drifting is the permanent displacement over time, which is the product of slumping and settlement. Shaking, on the other hand, is the back-and-forth movement of the

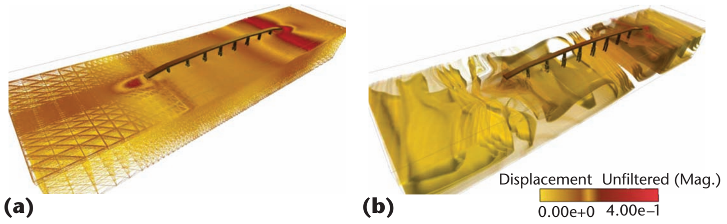


Figure 6. Visualization of unfiltered seismic data. (a) Current time step: 600 (12,620s). The conventional wireframe representation makes it difficult to see the inside and to convey underground motions because of occlusions. (b) For volume rendering, with a well-defined transfer function, we can remove visual cluttering. However, it's hard to see two intertwined motions together.

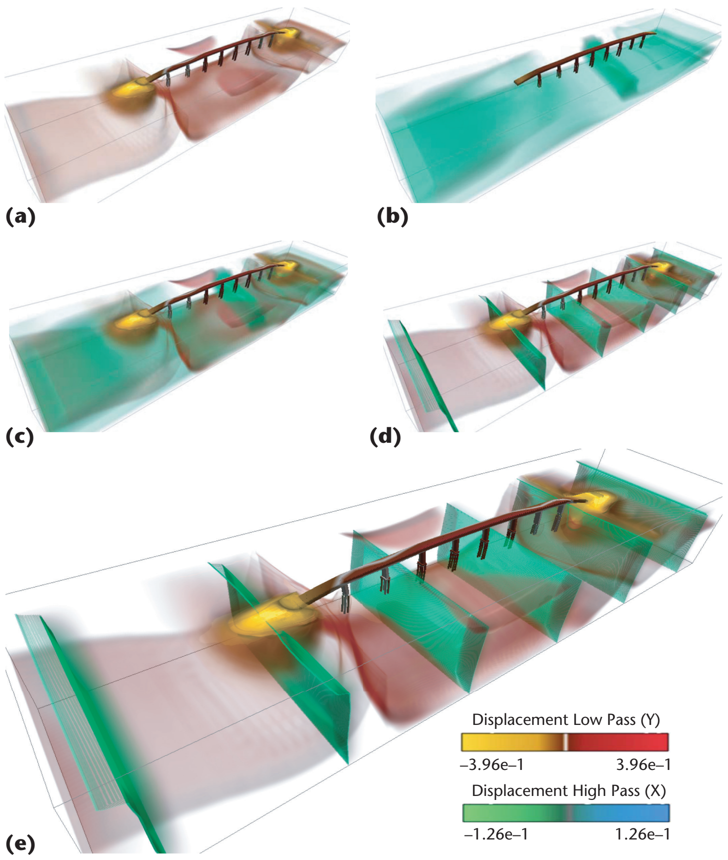


Figure 7. Volume visualization. (a) Displacement low pass (y)—the vertical drifting (permanent) motion. (b) Displacement high pass (x)—the horizontal shaking (cyclic) movement. Yellow-brown iso-surfaces near the approach ramps show downward settlement, and reddish surfaces in the middle of the bridge suggest upward motions. In contrast, high-pass back-and-forth motions are difficult to capture because of small movements. In (c) both low- and high-pass parts are composited together. (d) An example of using deformable implicit surfaces along the vertical direction. The high-frequency portion is displayed using surfaces superimposed with the low-pass volume. (e) The motion-blur effect is applied to convey the high-pass movements. At a casual glance, it would appear in (d) that the shaking waves are moving away from the viewer, while in reality—as (e) shows—the waves are actually moving toward the viewer, with the tops being pushed the fastest.

ground during the earthquake. As Figures 6–8 show, different transfer functions are used to represent vertical low-pass and horizontal high-pass displacements, which depict the drifting and shaking components.

Figure 6a and 6b show the original view of unfiltered seismic data rendered in wireframe and volume. In the conventional wireframe representation, it's difficult to see beyond the surface and into the volume's interior, rendering it nigh impossible to gain any insight from the inside motions. On the other hand, volume visualization lets us use a transfer function to pick out the areas we want to see, but it's still difficult to visually separate the two motions of interests.

The solution is *band-pass filtering*. Once the low-pass and the high-pass parts are extracted from the original seismic signals, we can render them separately. Figures 7a and 7b show the volume visualization of the vertical component of the low-pass motion, and the horizontal component of the high-pass motion. With a carefully designed transfer function, we can tease out the low-pass motions that the scientists are looking for. In Figure 7a yellow-brown iso-surfaces near the bridge's approach ramp zones show downward settlement, while the reddish iso-surfaces in the middle of the bridge show an upward motion.

Compared to low-pass motions, the high-pass back-and-forth motions shown in Figure 7b are difficult to capture because of their relatively small magnitudes. However, we can also combine these two renderings to simultaneously view both the low- and high-pass data as shown in Figure 7c. This is the same data shown in Figure 6b, except that with two transfer functions, it's much easier to pick out the two motions that we're looking for.

Figure 7d shows an example of using deformable implicit surfaces along the vertical direction. The high-frequency portion is displayed using surfaces superimposed with the low-pass volume. Compared to Figure 7c, we have a much clearer view of the low-pass volume while maintaining a sense of the motion of the high-pass volume. These surfaces provide a much more intuitive view of the motion when playing the time steps as a video. With a static image like Figure 7d, it's still difficult to discern the direction of the displacement at a glance, especially with an arbitrarily oriented image. However, with motion-blur rendering (Figure 7e), we can easily tell that the waves are moving from right to left in image space. Overall, our techniques lead to a much better presentation of the data's relevant portions. Figure 8 shows additional comparisons.

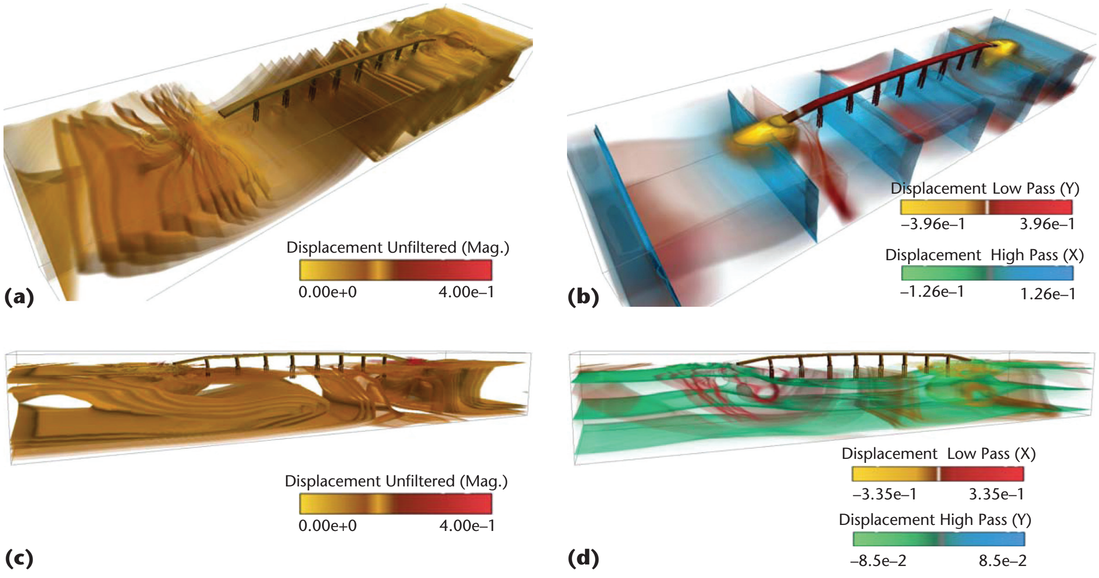

Displacement Unfiltered (Mag.)
0.00e+0
4.00e–1
(a)
Displacement Low Pass (Y)
–3.96e–1
3.96e–1
Displacement High Pass (X)
–1.26e–1
1.26e–1
(b)
Displacement Unfiltered (Mag.)
0.00e+0
4.00e–1
(c)
Displacement Low Pass (X)
–3.35e–1
3.35e–1
Displacement High Pass (Y)
–8.5e–2
8.5e–2
(d)

Figure 8. An earthquake's impacts on the bridge abutments and supporting structures. (a) Current time step: 550 (11,620s). Different motions including settlement, push, and heave can be easily observed with band-pass separation and volume-rendering techniques. (b) Current time step: 550 (11,620s). Volume visualization of vertical displacements for three time steps. Reddish and yellowish colors represent downward (positive) and upward (negative) movements, respectively. As shaking continues, the low-pass permanent deformations near the abutments accumulate, and the bridge's center has been lifted up. (c,d) Current time step: 650 (13,620s). Volume visualization of horizontal displacements (same as the bridge traffic direction). Horizontal forces from each side of the bridge push the abutments toward the center of the river.

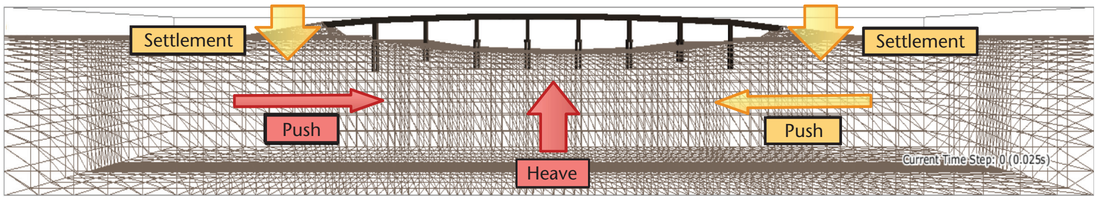
We implemented our system on an Intel Xeon X5450 CPU at 3 GHz, with 16 Gbytes of RAM and an Nvidia GeForce 285 GTX video card. Compared to simple volume rendering, our two-variable volume rendering requires an extra texture fetch per sample to accommodate our separate variable, and then two extra texture samples for Lum and his colleagues' lighting. The base number of ray casting steps is 512 across the smallest dimension. To maintain interactivity, we cut the number of steps taken during ray casting to one-fourth the steps (that is, 128 steps) when exploring the visualization, providing an average of 15 frames per second (fps) for a window size of 1,280 × 720. Rendering at the normal sampling rate averaged 3 fps. Initial resampling of the volume for the initial time step takes an average of 310 ms; subsequent time steps take an average of 91 ms because only the displacement data is re-uploaded and the mesh data is reused. The actual resampling time, not including the time it takes to upload the data, is 20 ms.

Our system can be modified to handle other datasets using different meshing schemes by inserting a mesh conversion step before resampling. With proper mesh decomposition methods, we can first convert polyhedral meshes to tetrahedral meshes, and send all the tetrahedra to the resampling and rendering pipelines for visualization. For instance, to deal with hexahedra meshing datasets, we can convert each hexahedron into six tetrahedra meshes, and then visualize the data using the same system.

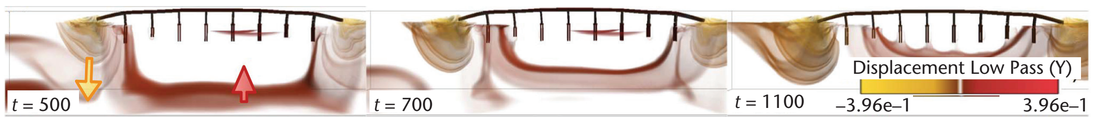
Evaluation

Elgamal, who created the earthquake simulation, evaluated the visualization results and found our techniques effective in isolating the two main features of interest in his team's data: drifting over time and the back-and-forth shaking motions that happen over the earthquake's entire duration. With hardware-accelerated high-quality rendering techniques, he can easily see how different components interact with each other.

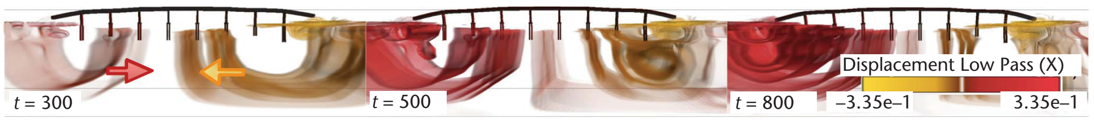
Prior to our solution, he and other earthquake scientists used *X–Y* scatterplots and geometry-based renderings to analyze the displacement time histories at points of interest. There was no simple solution here to tease out complex behaviors for quick and easy comparisons. Our system provides an integrated environment with a graphical user interface, which let them visually explore their datasets with ease. Our system also provides better visualizations of seismic datasets. Compared to geometry-based methods, which use wireframes or glyphs to indicate the displacements at each node, or methods that use color to represent the displacement's magnitude, our method significantly reduces



(a)

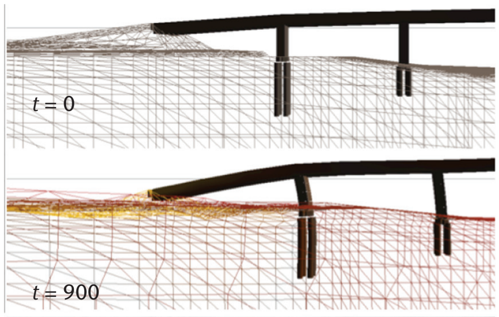


(b)

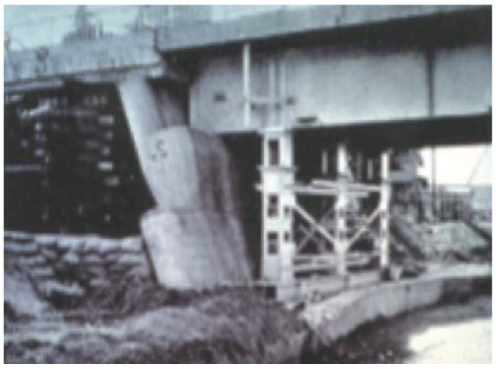


(c)

Figure 9. Comparison images between direct-volume rendering of the magnitude versus rendering different components using our method, for two different time steps. (a) It's difficult (if not impossible) to determine the shaking direction with direct-volume rendering of the magnitude. (b) Using our method, the shaking direction of this time step is clearly moving away from the viewer and the bridge ramps are sinking. (c) Another example of direct-volume rendering of the magnitude. (d) Using our method, it's easy to see that shaking motion is up and down and the two bridge ends are being pushed toward the center.



(a)



(b)

Figure 10. More comparison images. (a) Residual deformation of the bridge-foundation-ground system after earthquake shaking on an exaggerated scale by a factor of 25. (b) A similar damage scenario observed in a historical earthquake event.

visual clutter and provides enhancements for the simultaneous display of the two different motions.

In most cases, we render the high-pass data using our deformable implicit surfaces and the low-pass parts with direct-volume rendering. Elgamal found this combination extremely informative during the animation of the time steps, as it shows the back-and-forth shaking without over-occluding the low-pass filtered data, which was observed to move in a single direction before settling into their final positions. He also found the motion blur useful for looking at still images, as it gave him a sense of how strong the shaking was while maintaining a relatively clear view of the overall movement.

Figures 9 and 10 show the impact on the bridge abutments and supporting structures when an earthquake occurs. Our techniques allowed Elgamal to make two main observations: during an earthquake, the bridge abutments sank over time, and they also moved toward the center of the bridge over time.

Figure 9b shows volume visualization of the vertical displacement for three time steps. The reddish and yellowish colors represent downward (positive) and upward (negative) movements, respectively. As the shaking continues, the low-pass permanent deformations near the abutments accumulate, and the center of the bridge lifts up. Meanwhile, horizontal forces from each side of the bridge push the abutments toward the center of the river.

Figure 9c shows volume visualization of horizontal components (going along the same direction as the bridge's traffic) of the displacements.

Similarly, reddish and yellowish colors represent right (positive) and left (negative) directions of movements. Figure 10a shows the residual deformation of the bridge-foundation-ground system after earthquake shaking on an exaggerated scale by a factor of 25; Figure 10b shows similar damage scenarios observed in a historical earthquake event.[8]

We're now looking at several interesting areas for future work. First, we'd like to apply parallel rendering techniques coupled with parallel I/O strategies to visualize large-scale seismic data. In general, large-scale seismic response simulation requires high temporal and spatial resolutions, and the simulators output hundreds of gigabytes to terabytes of data. In our current implementation, all the data is processed on a single machine, so the size of the dataset to be visualized is bounded by the machine's memory storage. To fully explore massive seismic data from large-scale earthquake simulations (as well as field-collected data), we need parallel rendering techniques.

Second, we'd like to investigate efficient strategies for using glyphs and annotations to provide quantitative information along with current rendering results. Although our system now successfully generates simultaneous but distinct visualizations of two different motions using color and textures, it lacks visual analysis features to show quantitative dataset information directly on the rendering image. Using glyphs and annotations is a simple solution for displaying such information; however, without a smart placement strategy, glyphs and annotations could easily clutter the image rather than offer insight. We believe that the placement issue can be solved (or minimized) by considering both dynamic view-dependent criteria, such as view angles, and static rendering parameters, such as lighting and transfer functions in the volume renderer.

Finally, we'd like to extend our resampling kernel to support other meshing schemes that scientists use. In addition to tetrahedral meshes, scientific computing communities often use hexahedral and hybrid tetrahedral-hexahedral meshes. With efficient methods for detecting plane-and-polyhedron intersections, we can modify our resampling to support other types of grids and let scientists visualize the datasets directly without preprocessing them. CiSE

Acknowledgments

This research was supported in part by the US National Science Foundation through grants OCI-0749227, OCI-0749217, OCI-00905008, OCI-0850566, and OCI-0325934, and the US Department of Energy through the Scientific Discovery through Advanced Computing (SciDAC) program agreements DE-FC02-06ER25777 and DE-FG02-08ER54956.

References

1. A. Elgamal et al., "Three-Dimensional Seismic Response of Humboldt Bay Bridge-Foundation-Ground System," *J. Structural Engineering*, vol. 134, no. 7, 2008, pp. 1165–1176.
2. F. McKenna and G.L. Fenves, *Open System for Earthquake Engineering Simulation User Command-Language Manual*, Univ. California, Berkeley, 2001.
3. S. Butterworth, "On the Theory of Filter Amplifiers," *Experimental Wireless and the Wireless Engineer*, vol. 7, 1930, pp. 536–541.
4. C.D. Correa, R. Hero, and K.-L. Ma, "A Comparison of Gradient Estimation Methods for Volume Rendering on Unstructured Meshes," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 3, 2009, pp. 305–319.
5. J. Kruger and R. Westermann, "Acceleration Techniques for GPU-Based Volume Rendering," *Proc. 14th IEEE Visualization Conf.*, IEEE CS Press, 2003, pp. 287–292.
6. F.F. Bernardon et al., "GPU-Based Tiled Ray Casting Using Depth Peeling," *J. Graphics Tools*, vol. 11, no. 3, 2006, pp. 23–29.
7. E.B. Lum, B. Wilson, and K.-L. Ma, "High-Quality Lighting and Efficient Pre-Integration for Volume Rendering," *Proc. Joint Eurographics-IEEE Symp. Visualization*, Springer-Verlag, 2004, pp. 25–34.
8. K.V. Steinbrugge, *Karl V. Steinbrugge Collection*, Univ. California, Berkeley, 2002; http://nisee.berkeley.edu/visual_resources/steinbrugge_collection.html.

***Cheng-Kai Chen** is a doctoral candidate in computer science at the University of California, Davis. His research interests include scientific data visualization—centering on the development of efficient partition schemes for complex volume—and vector field data visualization. Chen has an MS in computer and information science from the National Chiao-Tung University in Taiwan. Contact him at ckchen@ucdavis.edu.*

***Chris Ho** is a graduate student in computer science at the University of California, Davis. His research interests include scientific visualization and computer graphics. Ho has a BS in computer science and philosophy from the University of California, Davis. Contact him at csho@ucdavis.edu.*

Carlos Correa *is a postdoctoral researcher at Lawrence Livermore National Laboratory, and a postdoctoral scholar in the Department of Computer Science, University of California, Davis. His research interests include computer graphics, visualization, and user interaction. Correa has a PhD in electrical and computer engineering from Rutgers University. Contact him at correac@cs.ucdavis.edu.*

Kwan-Liu Ma *is a professor of computer science and the chair of the Graduate Group in Computer Science at the University of California, Davis, where he leads the VIDI research group and directs the US Department of Energy's Scientific Discovery through Advanced Computing (SciDAC) Institute for Ultrascale Visualization. His research interests include visualization, high-performance computing, and user interface design. He's on the editorial boards of* IEEE Computer Graphics and Applications, IEEE Transactions on Visualization and Computer Graphics, *and the* Journal of Computational Science and Discoveries. *Ma has a PhD in computer science from the University of Utah. He's a senior member of the IEEE. Contact him at ma@cs.ucdavis.edu.*

Ahmed Elgamal *is a professor in the Department of Structural Engineering at the University of California, San Diego. His research interests include earthquake engineering and computational geomechanics. Elgamal has a PhD in civil engineering and engineering mechanics from Princeton University. He received the Shamsher Prakash Award and the NSF Presidential Young Investigator Award, and is a member of the American Society of Civil Engineers. Contact him at aelgamal@soe.ucsd.edu.*

stay connected.