Chapter 5

Paper I: Illustrative Rendering of Seismic Data

Reprinted, with permission, from Prof. Hans-Peter Seidel

Illustrative Rendering of Seismic Data

Daniel Patel[1,2], Christopher Giertsen[1], John Thurmond[3], Eduard Gröller[4,2]

Christian Michelsen Research, Bergen, Norway[1]
University of Bergen, Bergen, Norway[2]
Norsk Hydro, Bergen, Norway[3]
Vienna University of Technology, Austria[4]
Email: daniel@cmr.no, chrisgie@cmr.no
john.thurmond@hydro.com, groeller@cg.tuwien.ac.at

Abstract

In our work we present techniques for illustrative rendering of interpreted seismic volume data by adopting elements from geology book illustrations. We also introduce combined visualization techniques of interpreted and uninterpreted data for validation, comparison and interdisciplinary communication reasons. We introduce the concept of smooth transitions between these two semantical levels. To achieve this we present transfer functions that map seismic volume attributes to 2D textures that flow according to a deformation volume describing the buckling and discontinuities of the layers of the seismic data.

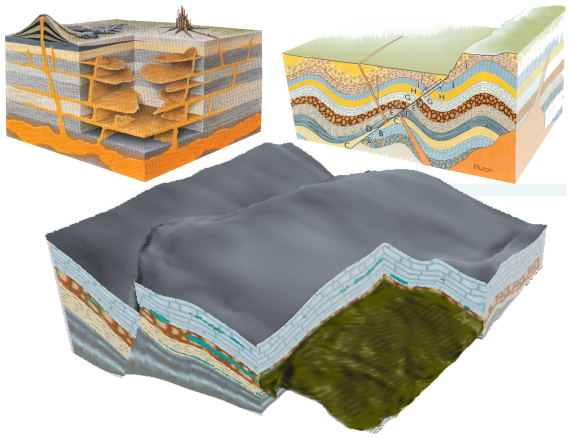

Figure 1: Geological and rendered illustrations. Top left: A cutout with extruding features. Top right: Textured layers with a fault discontinuity in the middle. Pictures are taken from Grotzinger et al. [6]. Bottom: Illustration rendered with our techniques.

1 Introduction

In geology faults and horizons are central subsurface structures. The earth has a layer-like structure and horizons are defined as the surfaces that separate one layer from another. Tension in the crust of the earth deforms the layers over time and creates cracks. These so called faults are more or less vertical discontinuities of the layers.

Geological illustrations in text books try to convey faults, horizons and other structures of the earth by using different artistic techniques as seen in the top of Figure 1. The illustrator draws a cubical subsection of the earth defining the area of interest. The horizons and faults are represented by using textures flowing inside the layers that are discontinuous across faults. The textures are drawn on the exterior side faces of the cubical subsection whose extent we hereby refer to as the roaming box. Axis-aligned cutouts with textures on the interior side faces are used to show features inside the cubical subsection. The cutouts sometimes contain extruding 3D features. Our illustrative renderings adopt all these techniques as seen in the bottom of Figure 1.

Figure 2 presents the flow from data acquisition to data visualization. The faults, horizons and other subsurface structures are discovered by geoscientists interpreting volumetric descriptions of the subsurface. These volumetric descriptions are typically obtained in geophysical surveys by processing the reflections of waves sent into the surface. The volume storing the reflection data is called the reflection volume. In a time consuming process the faults and horizons are manually found from the reflection volume and stored as surfaces. Several seismic attributes can be computed from the reflection data

such as acoustic impedance (Ai) and the ratio between the pressure and shear wave (Vp/Vs). We will refer to these volumes as seismic attributes.

Coming up with a good visualization of interpreted data can be difficult, therefore we propose to use illustrative techniques. Illustrations are being used when there are certain high level aspects of a complex image, such as interpreted information, that need to be communicated in a simple way. Rendering of interpreted seismic data as illustrations has several advantages. It simplifies the visualization and emphasizes the elements of interest in order to disseminate gained knowledge from the interpretation process. Making a good illustration for scientific purposes takes time. Being able to render geological illustrations is advantageous both for quickly creating static images to illustrate geological books and for interactive oil exploration when interpreted survey data needs to be communicated as part of decision making.

Interpreting seismic data is a time consuming manual process and it is important to verify the interpretation with the underlying data source. By combining visualizations of interpreted and uninterpreted data it is possible to perform comparisons and look for deviations. This is another goal in our work. We propose to visualize the interpreted data as geological illustrations and to visualize uninterpreted data using color coded cutting planes and regular volume rendering. We present how to combine these two representations. The user can control the balance between these two visualization styles to fit his or her needs. For interdisciplinary communication reasons visualizations can be made to have the right balance between interpreted data which contains semantical information understandable by lay men to uninterpreted data which contains the information-rich underlying data material understandable by domain experts.

To our knowledge the concept of creating automatic illustrations of seismic data has not been thoroughly explored before, neither in the geophysics nor in the visualization research community. We also believe this applies to combined rendering of interpreted and uninterpreted seismic data.

We start with related work in Chapter 2. After an overview in Chapter 3 we describe the calculation of the texture flow in chapter 4. In chapter 5 we use the calculated flow in combination with texture transfer functions to texturize the cutting planes on the side faces of the cubical subsection and on the cutout. In chapter 6 we describe volume rendering for displaying the cutout and the surroundings and we specify how this is integrated with the rendering of textures during ray casting. Finally future work and conclusions are presented in chapter 7. The bottom half of Figure 2 shows a high level overview of the paper.

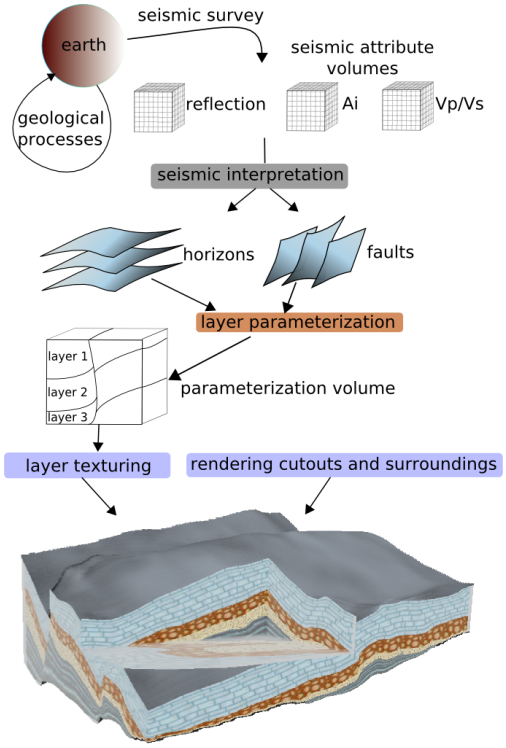

Figure 2: Overview of the process from data collection to visualization. The paper covers the lower three colored rectangles in chapter 4, 5 and 6.

2 Related work

We first review work dealing with illustrative techniques and then review work in the field of seismic visualization. Illustrative rendering is a non-photo realistic visualization technique using the advantages of conveying information through illustrations. In recent years several illustrative rendering techniques, mainly in the domain of anatomical visualization, but none in the domain of seismic visualization, have been proposed. Some of these techniques deal with applying textures from reference images.

Owada et al. [11] present an interactive system for texturing arbitrary cuts through polygonal ob-

jects. The user defines the texture flow by specifying a flow field and a distance field on the cut which is used in the texture synthesis to create a texture on the cut that follows the flow. Their method is general and therefore requires user interaction to specify the deformation and the texture. We calculate a parameterization up front so texturing can be achieved quickly and without the need for texture synthesis. In our approach many of the parameters defining the visualization are known prior to rendering, therefore less user specification is required

There are also several papers dealing with textures in medical volume rendering. Lu and Ebert [9] generate illustrative renderings of color and scalar 3D volumes by applying textures sampled from illustrations and photographs. 3D textures are created by combining color information from the illustrations with 3D volume data of a corresponding area. Finally the 3D textures are made tileable with Wang Cubes. With segmented volume data they apply the corresponding 3D textures on each segment. With unsegmented scalar data they use a transfer function to map scalar voxel values to the 3D textures in a similar way to what we propose. They do not deal with multi-attribute texture transfer functions and with deforming the textures to follow the underlying flow of the data as we do. In addition their method of calculating the textures is tailored to handle 3D textures whereas we use 2D textures.

Dong and Clapworthy [4] present a technique that achieves 3D texture synthesis following the texture orientation of 3D muscle data. Their algorithm has two steps. First they determine the texture orientation by looking at the gradient data of the volume and by using a direction limited Hough transform. Second they perform a volumetric texture synthesis based on the orientation data. In our work, instead of considering the volume for evaluating texture flow, we consider the geometric layers. In addition the texture synthesis of Dong and Clapworthy has the drawback of not working on textures with large interior variation as textures in geologic illustrations commonly have.

Wang and Mueller [14] use 3D texture synthesis to achieve sub-resolution zooming into volumetric data. With 2D images of several zoom levels of a tissue, they synthesize 3D volume textures for each level and use constrained texture synthesis during zooming to blend smoothly between the levels. They address the issue of sub-resolution details but do not consider texture flow.

In the domain of seismic processing GeoChron [10] is a formal model for parameterizing the layers defined by faults and horizons. The GeoChron model allows for several inputs which act as constraints to the parameterization. It considers the physical processes behind the deformation whereas our parameterization is fully defined by the fault and the horizon data. We believe that for illustration purposes a less physically accurate and computationally less intensive algorithm requiring a minimal amount of input and expertise such as our parameterization is preferable. However since our visualization algorithm is decoupled from the parameterization, it would also accept a GeoChron parameterization.

Cutouts on seismic data and interaction in VR was presented in the work by Ropinski et al. [13] where they use volume rendering with two transfer functions. One transfer function is used for the volume inside a user defined box and one transfer function is used for the volume outside. We incorporate and extend this concept in our work. Several papers on visualizing seismic data exist. Some deal with automatic horizon extraction [2] or fault extraction [2, 7], others deal with handling large volumes [2, 12], but none deal with illustrative rendering. Somewhat related is the dissertation of Frank [5] where the GeoChron parameterization [10] is used as a lookup to unfold and flatten a seismic data volume. In commercial systems seismic attribute data is presented with volume rendering and geometric surfaces are used to present horizons and faults.

3 Overview of the rendering process

Our methods render interactively the illustrative features found in geological images. Texturing is achieved by rendering deformed 2D textures on the exterior side faces of the roaming box and on the interior side faces of the cutout. For each layer the user assigns a texture and the texture's horizontal and vertical repeat rate. To also represent seismic attributes the user can assign textures and opacities to intervals of the seismic attribute values. These attribute textures are then blended and laid over the layer textures. We represent extruding features in the cutouts by volume rendering using a color transfer function together with a depth based opacity

transfer function. The opacity is a function of the layer depth and the transparency can be set to restrict volume rendering to certain layers or to certain depths within a layer. Also in the surrounding area outside the cutout we perform volume rendering that can be restricted to certain layers or to certain depths within a layer. In the surrounding area the voxel colors are equal to the average color of the 2D texture used in the layer the voxel is in. This gives a consistent coloring with the cutting plane textures as can be seen in the bottom of Figure 2 and the top of Figure 10. There we render opaquely the top and bottom horizon with the average color of the 2D texture in the horizons. To visualize the uninterpreted seismic data we render the cutting planes and the surrounding volume with the color transfer function used for the cutout volume rendering. The user can smoothly change between the uninterpreted data rendering and the interpreted illustrative textured rendering by changing the blending factor. An overview of the texturing process can be seen in Figure 6 while the lower part of Figure 2 shows how the texturing fits into the 3D visualization. In the next three chapters the details of the visualization process described above is presented.

4 Layer parameterization

We parameterize the volume to render 2D planar textures following the flow of the layers and to achieve depth controlled volume rendering. Using the coordinate system shown in Figure 3d we define horizons as non-intersecting surfaces stacked in the z-direction of the type $z = H(x, y)$ and faults as non intersecting surfaces stacked in the x-direction of the type $x = F(y, z)$ or in the y-direction of the type $y = F(x, z)$. The faults, horizons and the side faces of the roaming box divide the volume into subvolumes which we will refer to as slabs. Conversely, each of these slabs is horizontally confined by what we will refer to as the upper and lower horizon and are laterally confined by fault surfaces and the side faces of the roaming box (see Figure 3a).

There exists no unique solution to parameterize a volume. We have designed the parameterization so that it represents the slabs in a flattened version where horizons and the layer between are planar. Figure 3d shows the parameterization coordinate system (u, v, w) embedded in the world coordinate system (x, y, z).

The parameterization consists of several steps. First the upper and lower horizon of the slab is extended by extrapolation (see dotted lines in Figure 3a). We do this extension to get a correct volumetric parameterization close to the vertical slab borders. Then the lower horizon surface is parameterized and the depth parameter w is calculated for the volume, (see curves in Figure 3b). Finally the 2D parameterization of the lower horizon is projected into the volume along the gradient field of the w parameter (blue curves in Figure 3c), resulting in a 3D parameterized slab.

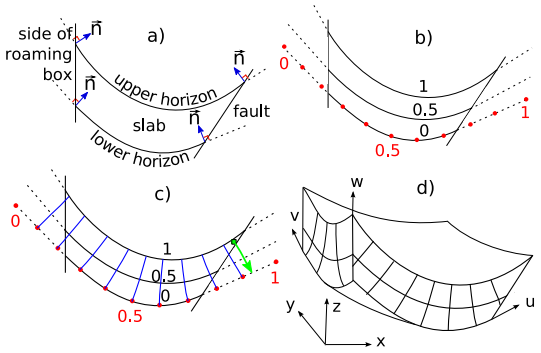

Figure 3: A 2D version of the steps needed for parameterizing a slab is shown in a-c. The world and the parameter coordinate system is shown in d.

Let $w_p = \{x, y, z\} \in \mathbf{R}^3$ be a point in W(orld space), and $p_p = \{u, v, w\} \in \mathbf{R}^3$ be the corresponding point in P(arameter space). We represent the mapping from W to P as :

$$\mathbf{P} : W \rightarrow P : p_p = \mathbf{P}(w_p) = \{P_u(w_p), P_v(w_p), P_w(w_p)\}$$

Let $min_{upper}(w_p)$ and $min_{lower}(w_p)$ express the Euclidean distance from w_p to the closest point on the upper and lower horizon respectively. The w parameter, or layer depth, is defined as:

$$P_w(w_p) = \frac{min_{lower}(w_p)}{min_{lower}(w_p) + min_{upper}(w_p)}$$

min_{upper} and min_{lower} are found by discretizing the upper/lower horizon into a point cloud. For discretizing we linearly subsample the horizon grid four times, and store the points in a kd-tree for efficient searching. Note that P_w does not express the distance to the closest surface as found by a distance transform, but the relative distance between the upper and lower horizon, it maps the lower horizon to 0, the upper horizon to 1 and is linear in between. In

effect it flattens the layer and defines a local depth measure on it. See curves in Figure 3b.

We now have a w parameterization of the slab. The (u, v) values in the slab are found by projecting the (u, v) values from the parameterized lower horizon, which is described in 4.2. Projections into the volume is done along the streamlines seen as blue curves in Figure 3c which are defined by the vector field ∇P_w. ∇P_w is calculated using central differences. For each voxel we trace along the streamline in the opposite gradient direction toward the lower horizon, seen as a green arrow in Figure 3c. We assign to the voxel the (u, v) value of the intersected point on the lower horizon.

Assigning (u, v) values for the voxels inside the slab that are close to the vertical slab borders might result in streamlines leaving the slab and entering an area where P_w has not been calculated. See green arrow in Figure 3c. We have extended the horizons with the method described in 4.1 prior to the w parameterization and prior to the (u, v) parameterization of the bottom horizon. By doing this we have gradient data outside the slab as well as a parameterized surface outside the lower horizon which makes it possible to calculate streamlines leaving the slab. The parameterization procedure ensures that the (u, v) parameterization is orthogonal to the w parameterization which in turn will result in angle preservation in the 2D textures. The parameterization works well for surfaces of low curvature and without folds as seen in this application but would require some extension for handling other types of surfaces.

The parameterization is done on each slab and is stored in an RGB volume consisting of the (u, v, w) parameters. The parameters of each slab are all in the $[0, 1]$ range. To encode segmentation information for each slab we scale and shift the w and u parameter values. Each layer's w values are scaled and shifted such that values go from 0 at the lower horizon in the bottom layer to 1 at the upper horizon in the top layer with each layer having equally sized intervals. Similarly, the u values are scaled and shifted on each side of the fault. At the left side of the fault in Figure 4 the u values are between 0 and 0.5 and on the right side they are between 0.5 and 1. The segmentation information is used for having different textures in different layers and possibly on different sides of faults. The parameterization is not meant to be geologically accurate

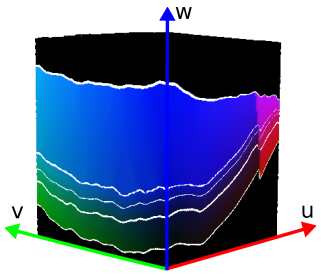

Figure 4: The parameterization RGB volume. White lines have been added on horizons. There is a color change across the fault due to shifting and scaling of the u parameter.

but to act as a tool for 2D texturing and depth dependent volume rendering. The goal is to achieve images with illustrative quality. The two following sections will describe the horizon extrapolation and the bottom horizon parameterization which was assumed to be done prior to the layer parameterization but were not explained in detail.

4.1 Horizon extrapolation

For parameterization of areas close to the vertical slab borders we need surface information beyond the horizon borders as described earlier. To achieve this we carry out a simple surface extrapolation in all directions. First we extrapolate the surface in positive and negative x direction by considering the surface as a collection of curves parallel to the x axis and extending the endpoints of the curves in tangential direction. See normals and dotted lines in Figure 3a. We then do the same procedure on the resulting surface in positive and negative y directions. Finally we crop the horizons so that their projections to the xy plane are rectangular and so that sufficient data exists beyond their original borders. On our data we ended up with a heuristic extension of 20 percent of the horizon length in each direction to correctly parameterize the areas close to the vertical slab borders.

4.2 Surface parameterization

For the (u, v) parameterization of the lower horizon surface we calculate a parameterization that locally minimizes the area distortion. Red dots on Figure 3b show the corresponding 1D version. The parameterization is created with the CGAL library [1] using the discrete authalic parameterization [3].

The parameterization defines (u, v) values for each vertex on the surface. The parameterization is constrained by giving initial values to the surface borders whose projection to the xy plane forms a rectangle due to the surface extrapolation. For the initial values we clockwise assign the border vertices with values (0,0) (0,1), (1,0) and (1,1) and interpolate the values along each edge with equidistant spacing. We now have a (u, v) parameterization of the lower horizon and a w parameterization of the slab.

4.3 Interpolation problem around horizons and faults

The parameterization volume is a discrete specification of our parameterization function. With trilinear sampling we get a smoother function which however leads to invalid interpolation in cells on slab boundaries where the eight cell corners are in different slabs. We calculate new values for the invalid corners by extrapolating from valid neighbor values. Then we perform the trilinear interpolation for the new corner values on the GPU. The extrapolation will try to assign a new value for invalid corners by considering the corner's two neighbors in positive x direction. If both are valid then their values are linearly extrapolated and assigned to the corner. If not, then the search continues in negative x direction and then similarly in y and z direction. In rare occasions the procedure fails to extrapolate all the invalid corner values and an erroneous interpolation is performed. The resulting artifacts will be noticeably only at the slab borders and will be of the same size as a voxel in the parameterization volume. The procedure improves the quality of the renderings as can be seen in Figure 5.

4.4 2D texture mapping on axis-aligned cutting planes

Our parameterization volume now makes it possible to apply an undeformed 3D texture stored in parameter space and deform it into world space for texturing voxels in our layers. However this would require to first generate 3D textures which is a research topic in its own as investigated by Lu and Ebert [9]. Since we are going to texture axis-aligned cutting planes as done in geological illustrations we can reduce the problem to a 2D texturing problem. This has several advantages. 2D tileable textures are easy to generate, take little space, and

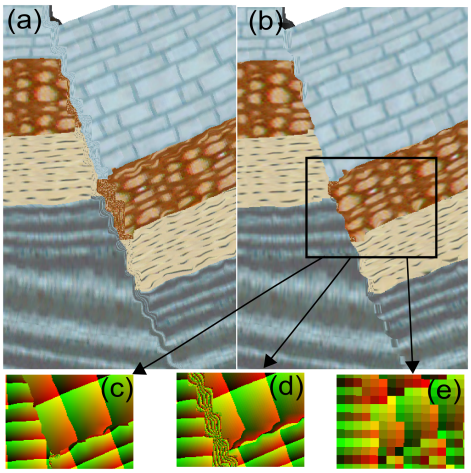

Figure 5: Fault and interpolation problems. In a) linear interpolation is used. In b) extrapolation as described in 4.3 improves the quality. In c) we see the parameterization of the zoomed-in rectangle with extrapolation as opposed to without in (d). In (e) we see the parameterization with nearest neighbor interpolation showing the resolution of the parameterization.

can be sampled from illustrations directly. With our method the 2D textures maintain coherency when moving the cutting planes and we have better control over the repetitive appearance than for 3D textures. However we need to define a transformation from 3D parameter space (u, v, w) to 2D parameter space (u', v'). The mapping is straightforward. For texturing in the xz plane we use the (u, w) values, for texturing in the yz plane we use (v, w) and for texturing in the xy plane we use (u, v) values. The mapping conserves the angle preservation property of the 3D parameterization.

5 Layer texturing

This chapter presents three transfer functions that are being used together to texture and color cutting planes. First we present the layer texture transfer function, abbreviated as layer TTF. It assigns textures to each layer. Then we present the scalar texture transfer function, abbreviated as scalar TTF. It assigns textures and opacities to regions having seismic attribute values in certain ranges. The resulting scalar TTF texture for a cutting plane is blended according to its opacities with the layer TTF texture using the over operator. The combined results are cutting planes with deformed tex-

tures similar to the ones in geology illustrations. Finally we present the concept of smoothly moving from illustratively rendered cutting planes to color coded cutting planes. Here seismic attribute values are mapped to colors using a color transfer function, abbreviated as color TF. See Figure 6 for an overview and Figure 7 for a texture example. Visualizing horizon, fault, deformation and seismic

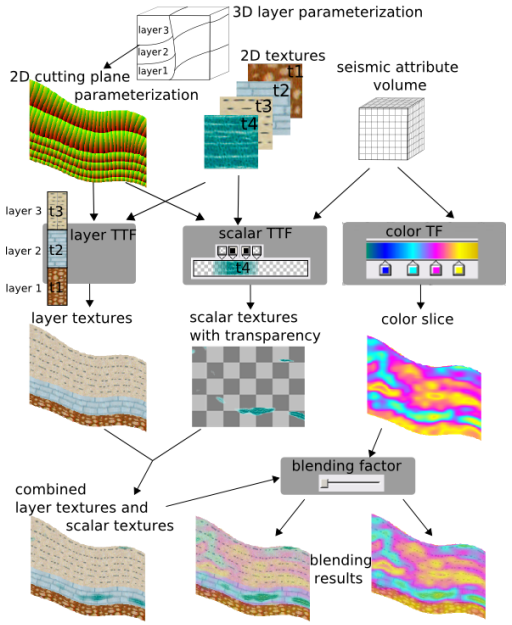

Figure 6: Overview of how textures are combined. Layer TTF, scalar TTF and color TFs are explained in 5.1, 5.2 and 5.3 respectively.

attribute information through textures has several advantages. By looking at Figure 7 one sees that textures communicate the id, orientation and compression of layers on a local scale. An example is given with the two small patches in the black circles in Figure 7. The texture of a patch reveals its layer id. The angles in the texture express the orientation of the layer in that area. Compression is presented through the vertical texture repeats in a layer. Since the vertical texture repeats are constant throughout the layer (there are always 16.5 bricks stacked in the height in layer 1 in Figure 7), compression will be high where the layer is thin and low where the layer is thick. It is possible to see that the texture patch in the left circle is slightly more compressed than the texture patch in the right circle of Figure 7. Finally, by letting both the horizontal and vertical texture repeat rate be a function of an underlying scalar value, scalar data can be presented in the texture as seen Figure 8. All this information is communicated with textures even on zoom scales where no horizons are visible, and also when zooming beyond the resolution of the seismic attribute volume. On such sub-resolution scales color transfer functions yield blocky or monotonous colored results whereas textures give aesthetically pleasing results. One can imagine zooming past the attribute volume resolution when inspecting overlaid high resolution data, such as bore well core data.

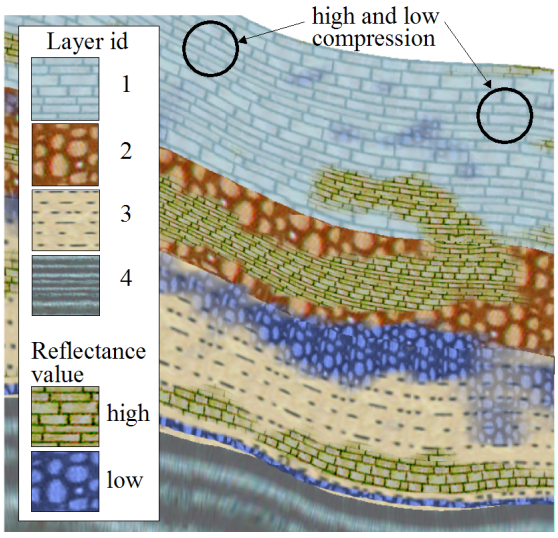

Figure 7: Combination of layer TTF and scalar TTF for the reflectance volume. The brown brick texture shows areas of high scalar values and the violet texture shows areas of low scalar values.

5.1 Layer texture transfer function (TTF)

To texture the cutting planes we use 2D RGB textures with wrap-around and bilinear filtering. They are taken from geological illustrations, and mapped on the layers. We use a layer TTF that maps from a voxel's w parameter value to a texture id, a horizontal and vertical texture repeat rate and an opacity value: $layer_{ttf}(w) = \{layerId, h_{rep}, v_{rep}, \alpha\}$. The opacity value is only used later in the cutout volume rendering. Since w varies in distinct intervals for each layer, each layer can have its own texture assigned. Texture variations within one layer such as having different textures in the top and bottom half of the same layer or having different textures on each side of a fault is also possible.

5.2 Scalar texture transfer function (TTF)

While the layer TTF represents the interpreted horizons as textures, the scalar TTF represents uninterpreted seismic attribute data as textures. The scalar TTF is equal to the layer TTF except that it is the seismic attribute values that are used as look up values. This makes it possible to control the textural appearance for regions on the cutting plane which have seismic attribute values in certain ranges. The scalar TTF texture is overlaid on the layer TTF texture using the over operator. The α value defines its transparency in the various regions. This combined view expresses how individual seismic attributes relate to layers, i.e., if intervals of an attribute are confined within certain layers or change significantly (or subtly) between layers. It also represents a unified visualization of layer data and seismic attribute data through textures.

Typically the repeat rates of a scalar texture are taken from the layer it is drawn on. However the user can set multiplicative factors in the repeat values in the scalar TTF to change this. We do this by default so textures can maintain the same repeat factors when crossing layers of different thicknesses. For a layer twice as thick as another one the vertical repeat of the thick layer's texture will be half of the thin layer's. A scalar texture crossing the layers would abruptly change its repeat rates. To have consistent repeats across layers as can be seen for the brown brick texture in Figure 7, the user can change in the layer TTF the vertical repeat for the thin layer to half of what it is for the thick layer.

The selection of repeat rates for the textures is highly dependent on the degree of zoom. When zooming out, textures will be perceived as being too high frequent and when zooming in they will be perceived as being too low frequent. For this reason we multiply all the repeat factors with a global user definable repeat factor which is manually set according to the zoom level.

5.3 Rendering uninterpreted and interpreted data

To inspect the uninterpreted data directly on cutting planes we apply the color TF on the scalar values of a seismic attribute. We also introduce the concept of a continuous transition from illustrative rendering of interpreted data to rendering of uninterpreted data. The transition is done by smoothly blending from visualizing textured cutting planes to visualizing cutting planes colored by the color TF with seismic attribute values. This not only gives a smooth transition from one mode to the other but also introduces an intermediate rendering mode where interpreted data is superimposed on the uninterpreted data. The balance between the two data sources can be adjusted to get what the user perceives as an optimal balance between the rendering techniques. See Figure 10.

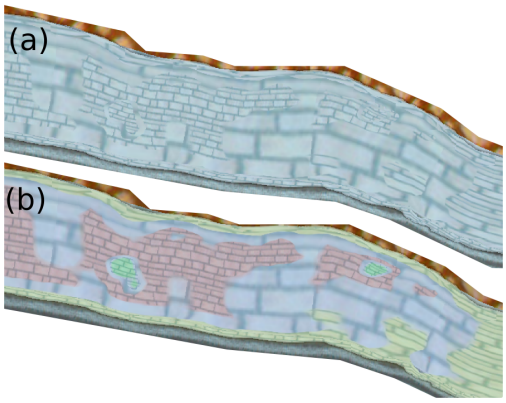

Figure 8: Layer TTF, scalar TTF and color TF combined. Instead of using different textures on intervals of the scalar values we use the same texture with four different repeat rates. It is difficult to discern the textures in a). In b) we blend in colors from the color TF to more easily discern the textures.

6 Rendering cutouts and surroundings

We implement volume rendering with one transfer function for the cutout and another one for the surroundings to support different rendering styles. By doing this we can achieve rendering of extruding features in the cutout and opaque ground rendering in the surroundings as seen in geological illustrations.

For volume rendering in the cutout we use the color TF on seismic attribute data introduced earlier. To specify transparencies in the volume rendering we extend the color TF with an α channel. By multiplying a voxel's α value from the color TF with the α value from the layer TTF we can adjust the transparencies based on the w value of the sample. Now we can do volume rendering on selected layers by manipulating the α in the layer TTF and

making layers transparent or semitransparent.

For visualizing the surroundings we do volume rendering where each voxel is given the average color of the 2D texture at the voxel position. The average color is precalculated for each 2D texture. The opacity is controlled by a separate opacity transfer function for the surroundings. It maps the w parameter to opacities enabling a layer oriented volume rendering of the surroundings. The opacity can then be set for instance to render certain horizon surfaces or layers semitransparently. When performing smooth transitions from rendering of interpreted data to rendering of uninterpreted data we go from using the average color of the 2D texture at the voxel position to using the voxel's color according to the color TF and the seismic attribute value at that position. In the images of this article we render the top and bottom horizon opaquely to get an opaque ground as seen in geological illustrations.

In the following paragraphs we describe how volume rendering is combined with texturing of the cutting planes. We perform ray casting with empty space skipping as suggested by Krüger and Westermann [8]. The entry and exit point of each ray is further clipped to the roaming box. Volume rendering with the transfer function for the surrounding is performed outside the cutout, while the transfer function for the cutout is used inside the cutout. Texturing is done at points where the parameterization volume intersects the exterior of the roaming box or the interior of the cutout box. See Figure 9 for a 2D depiction which acts as a reference to the following description. Texturing is done at the entry

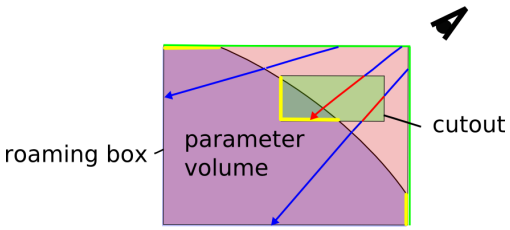

Figure 9: Combining volume rendering with texture rendering. The green line depicts the entry points. The yellow lines show where texturing is done. Red ray segments show where the cutout transfer function is used and blue ray segments show where the surrounding transfer function is used

point if the entry point is inside the parameter volume (green/yellow border). If not volume rendering with the transfer function for the surrounding is performed until the end point (upper blue ray) or until the cutout is intersected. If the cutout is intersected then volume rendering with the transfer function for the cutout is used (red segments) until the cutout exit point is reached. If the cutout exit point is inside the parameter volume texturing is performed (yellow border). If not, ray casting with the transfer function for the surrounding is performed until the exit point. A ray is always terminated if opacity reaches 1.

By doing volume rendering only on selected layers we can easily achieve the effect seen in geological illustrations of extruding layers in the cutouts. For exploration of the seismic data this is useful when the user wants to consider only one layer at a time. For instance the oil reserves are typically trapped between horizons in so called reservoirs. If the expert wants to perform volume rendering to explore such a reservoir it would be natural to confine the volume rendering to the layer the reservoir is in. See the bottom of Figure 1 for an example of volume rendering in a cutout limited to a layer. It shows a combined texture and volume rendering with an extruding layer. Volume rendering is performed only for layer 3 with brown color to mimic a geological illustration. The layer discontinuity is due to a fault. Turquoise patches on the textures show areas with high reflection values. Figure 10 shows a smooth transition from illustrative rendering to seismic attribute rendering.

The texture calculation and volume rendering is performed on the GPU in a single pass. With a Geforce 8800 GTX graphics card and an image size of 800×800 we achieve 5 frames per second. Without the extrapolation as described in 4.3 the frame rate is doubled. The three component parameterization volume is of size $128 \times 128 \times 128$ and the reflectance volume of size $240 \times 271 \times 500$. The Ai volume is of size $96 \times 96 \times 500$ and covers a smaller area than the reflectance volume. The 2D textures are each of size 64×64.

7 Conclusions and future work

We have presented a technique for illustrative rendering of interpreted geological data. We have also shown how to create combined visualizations of interpreted and uninterpreted seismic data for validation and comparison reasons and for creating visu-

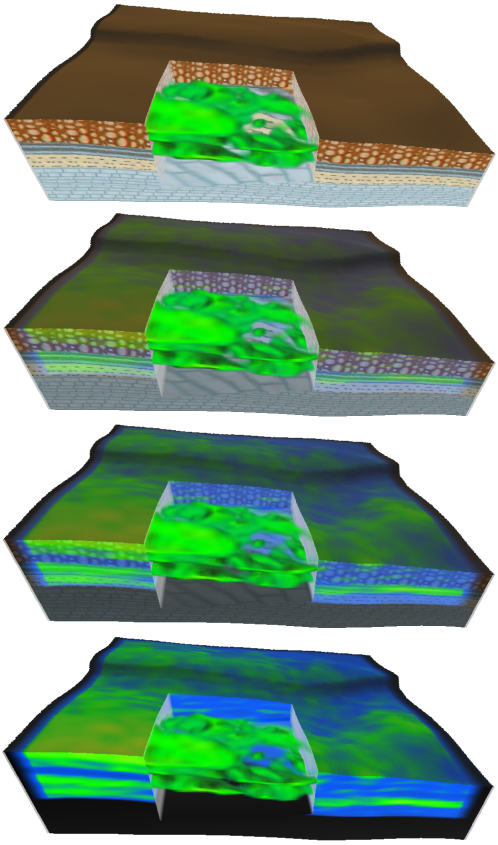

Figure 10: Blending from illustrative rendering to uninterpreted data rendering of the Ai attribute. Volume rendering is performed in areas with high Ai values. On the right of the cutout one can see how the green area having high Ai values corresponds to a layer. The black areas contain no Ai data.

alizations that can be targeted to anyone from laymen to domain experts. On the technical side we have presented the concept of 2D texture transfer functions with deformed textures.

Illustrative techniques can make it faster to evaluate large oil prospects. It can also improve communication between different stakeholders and towards media, public sector and politicians. In the future we will look into methods making it possible to do illustrative rendering of uninterpreted data.

References

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] L. Castanie, B. Levy, and F. Bosquet. Volumeexplorer: Roaming large volumes to couple visualization and data processing for oil and gas exploration. *Proceedings of IEEE Visualization '05*, pages 247–254, 2005.

[3] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21:209–218, 2002. Eurographics conference proceedings.

[4] F. Dong and G. Clapworthy. Volumetric texture synthesis for non-photorealistic volume rendering of medical data. *The Visual Computer*, 21(7):463–473, 2005.

[5] T. Frank. *Advanced Visualisation and Modeling of Tetrahedral Meshes*. PhD in geosciences, Institut National Polytechnique de Lorraine, 2006.

[6] J. Grotzinger, T. H. Jordan, F. Press, and R. Siever. *Understanding Earth*. W. H. Freeman and Company, 1994.

[7] W.-K. Jeong, R. Whitaker, and M. Dobin. Interactive 3d seismic fault detection on the graphics hardware. *Volume Graphics*, pages 111–118, 2006.

[8] J. Krüger and R. Westermann. Acceleration techniques for gpu-based volume rendering. *Proceedings of IEEE Visualization '03*, pages 38–47, 2003.

[9] A. Lu and D. S. Ebert. Example-based volume illustrations. *Proceedings of IEEE Visualization '05*, pages 83–92, 2005.

[10] R. Moyen. *Parametrisation 3d de lespace en geologie sedimentaire: le modele geochron*. PhD in geosciences, Institut National Polytechnique de Lorraine, 2005.

[11] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi. Volumetric illustration: designing 3d models with internal textures. *SIGGRAPH '04*, pages 322–328, 2004.

[12] J. Plate, M. Tirtasana, R. Carmona, and B. Fröhlich. Octreemizer: a hierarchical approach for interactive roaming through very large volumes. *Proceedings of VISSYM '02*, pages 53–64, 2002.

[13] T. Ropinski, F. Steinicke, and K. H. Hinrichs. Visual exploration of seismic volume datasets. *Journal Proceedings of WSCG '06*, 14:73–80, 2006.

[14] L. Wang and K. Mueller. Generating subresolution detail in images and volumes using constrained texture synthesis. *Proceedings of IEEE Visualization '04*, pages 75–82, 2004.