Interactive Seismic Interpretation with Piecewise Global Energy Minimization

Thomas Höllt*
King Abdullah University of Science and Technology

Johanna Beyer*
King Abdullah University of Science and Technology

Fritz Gschwantner†
VRVis Research Center

Philipp Muigg‡
SimVis GmbH

Helmut Doleisch‡
SimVis GmbH

Gabor Heinemann§
Heinemann OiL GmbH

Markus Hadwiger*
King Abdullah University of Science and Technology

ABSTRACT

Increasing demands in world-wide energy consumption and oil depletion of large reservoirs have resulted in the need for exploring smaller and more complex oil reservoirs. Planning of the reservoir valorization usually starts with creating a model of the subsurface structures, including seismic faults and horizons. However, seismic interpretation and horizon tracing is a difficult and error-prone task, often resulting in hours of work needing to be manually repeated. In this paper, we propose a novel, interactive workflow for horizon interpretation based on well positions, which include additional geological and geophysical data captured by actual drillings. Instead of interpreting the volume slice-by-slice in 2D, we propose 3D seismic interpretation based on well positions. We introduce a combination of 2D and 3D minimal cost path and minimal cost surface tracing for extracting horizons with very little user input. By processing the volume based on well positions rather than slice-based, we are able to create a piecewise optimal horizon surface at interactive rates. We have integrated our system into a visual analysis platform which supports multiple linked views for fast verification, exploration and analysis of the extracted horizons. The system is currently being evaluated by our collaborating domain experts.

Index Terms: I.3.8 [Computing Methodologies]: Computer Graphics—Applications; I.4.6 [Image Processing and Computer Vision]: Segmentation—Edge and feature detection

1 INTRODUCTION

Fossil fuels are today's most important source of energy. According to the International Energy Outlook 2010 [36] by the U.S. Energy Information Administration, roughly 58% of the marketed energy in 2007 were oil and gas products. It is expected that in 2035 oil and gas will still account for more than 50% of the marketed energy. The combination of dwindling oil in place and increasing energy demand in general results in rising prices and justifies the exploration of smaller or more complex reservoirs.

Planning of reservoir valorization usually starts with creating a model of the subsurface structures, the *seismic interpretation*. A seismic reflection volume, or *seismic cube*, is acquired by sending sound waves, for example triggered by explosions, into the earth and recording the emerging echoes. At the boundary of two subsurface layers, a *seismic horizon*, a part of the induced wave proceeds, while another part is reflected. The reflected waves are recorded over time by a grid of geophones arranged on the ground, resulting in a set of 1D traces containing a reflection value for each time step. The values for a single time step of all geophones arranged in the grid form a *time-slice*. The grid itself defines the spatial resolution. In the volume, a set of traces with equal position in the grid for either the x- or y- coordinates form the *crossline* and *inline* slices, respectively. The traces and time-slices are inscribed along the z-axis (see Figure 3).

The extraction of seismic horizons is one of the main tasks when interpreting the seismic cube to build a model of the subsurface structures. Horizons are represented in the volume by bands of locally extremal values, whereas most other structures like *faults* are primarily defined by their interaction with horizons.

The seismic interpretation of the volume can be used for the *time-depth conversion*, a process where the time axis is converted to actual spatial depth values. Estimating the sonic speed for each layer, the layers can then be stretched or compressed using the associated sonic speed resulting in a depth-converted volume.

In most cases, the seismic interpretation is not based solely on the seismic cube. Usually, physical drillings provide additional data, resulting in *well logs*. A well log is a detailed record of a certain property of the subsurface structures at the drill hole. Logs can be either geological or geophysical. The former are acquired by visually or chemically inspecting samples brought to the surface, but also contain live drilling parameters such as the speed at which the drill bit deepens the borehole (i.e., the rate of penetration), the latter by lowering a measurement device into the borehole. A comprehensive survey of a drilling usually contains several well logs, geological as well as geophysical. Typical properties besides the rate of penetration are for example weight, porosity or resistivity. In the following, we use the term well log synonymously for a single log as well as for a complete survey of one drill hole, containing several logs. The well log data is especially useful when performing time-depth conversion, as it contains actual information about the depth location of subsurface structures.

Interpreting the seismic cube is a cumbersome, time-consuming process. The data is dense, hard to visualize in 3D, noisy, and ambiguous. It may happen that after hours of interpretation work it becomes apparent during time-depth conversion that large parts of the volume were misinterpreted due to a single wrong decision in what seems to be a branching horizon. In current practice, which is mainly based on manual inline-, and crossline-slice inspection, this is very tedious, as it requires manual correction of multiple slices. Additionally, these slices need to be adjusted according to well log information which usually is only available for very few slices, and might be located between interpreted slices where no interpretation has been performed.

In this paper, we propose a novel workflow for seismic horizon interpretation based on well positions. We triangulate the spatial positions of all available well logs, which divides the volume into a set of triangular prisms. Instead of interpreting the volume slice-by-slice in 2D, we propose performing full 3D seismic interpretation, going from prism to prism. The part of a horizon intersecting a

*e-mail:{thomas.hollt|johanna.beyer|markus.hadwiger}@kaust.edu.sa
†e-mail:fritz.gschwantner@vrvis.at
‡e-mail:{muigg|doleisch}@simvis.at
§e-mail:gheinemann@heinemannoil.com

IEEE Pacific Visualisation Symposium 2011
1 - 4 March, Hong Kong, China
978-1-61284-934-8/11/$26.00 ©2011 IEEE

given prism can be extracted by specifying as little as a single seed point, using global energy minimization techniques to compute a horizon surface of minimal "cost". We combine very fast 2D minimal cost path tracing for determining horizon intersections along the faces of the prisms resulting from well log triangulation with subsequent 3D minimal cost surface computation, which is constrained by the 2D contours on the prism faces. By combining the optimal surfaces from the individual prisms, we receive a piecewise globally optimized horizon. Extracting the horizon in a piecewise manner allows for an interactive algorithm which gives the user full control over adjusting the surface by setting an arbitrary number of constraints, forcing the surface to pass through specified locations. We provide a framework with multiple linked 2D and 3D views, to enable easy interaction with the data and verification of the resulting surfaces. We have integrated our system into the SimVis visual analysis framework [11].

2 Related Work

The importance of oil and gas for today's societies has resulted in a considerable amount of previous work in the area of seismic interpretation and visualization, as well as commercial software packages, such as HydroVR [20], Petrel [34] or Avizo Earth [37].

One important area is seismic interpretation and seismic horizon extraction. Pepper and Bejarano [28] give a good overview over these seismic interpretation techniques. A lot of work has focused on fully automatic approaches that require the definition of some parameters, but afterwards work without user interaction. Keskes et al. [17] and Lavest and Chipot [19] show an abstract outline of such algorithms for fully automated 3D seismic horizon extraction and surface mesh generation. Tu et al. [35] present an automatic approach for extracting 3D horizons, but based on grouping 2D traces. Faraklioti and Petrou [14] and later Blinov and Petrou [3] present real 3D surface reconstruction using connected component analysis parameterized by the local waveform and layer direction. However, processing the complete volume at once is a lengthy process, thus parameter tuning is inconvenient. Additionally, optimal parameters are not necessarily equal for all features in the volume.

Castanie et al. [9] propose a semi-automatic approach. Horizons are traced one by one from a user-defined seed point. Interactivity in these kind of methods is limited as the horizon extraction is costly, resulting in long waiting times between seeding.

Patel et al. [26] present a technique for quick illustrative rendering of seismic slices for interpretation. They use transfer functions based on precomputed horizon properties. Their method, however, only works on 2D slices. In a subsequent publication [27] they extended their illustration technique for rendering of 3D volumes to be used for knowledge-assisted visualization of seismic data. This method still relies on slice-based precomputation.

Borgos et al. [4], as well as Patel et al. [25], propose interactive workflows for the surface generation based on an automatic preprocessing step. Both extract surface patches from the volume in a preprocessing step by extrema classification and growing, respectively. This step lasts for several hours. In a second, interactive step the user then builds the horizon surface using the pre-computed patches. Where in the work of Borgos et al. the surface patches function as simple building blocks, Patel et al.'s approach is more sophisticated. The extracted patches are subdivided and stored hierarchically. The user starts the surface generation with a single seed patch and by climbing up the hierarchy adds connected patches. The authors themselves target their approach to quick extraction of horizon sketches which then need to be refined in a second step.

Lampe et al. [18] present a technique to deform and render volumes along curves. They show two applications, one of which is the visualization of seismic data along well logs.

We propose an interactive seismic interpretation workflow exploiting global energy minimization inspired by segmentation techniques employed in computer vision. Our method does not require any pre-computation and allows modification of computed surface patches by adding constraints. By partitioning the volume into prisms, the time waiting for global optimizations is minimized. By applying the surface extraction on smaller chunks (i.e, prisms) instead of the complete volume at once, we maintain an interactive workflow for extracting piecewise global optimal horizon surfaces.

Typical image segmentation systems based on energy minimization can be divided into two classes based on the types of constraints they provide. Approaches based on graph cuts [5, 7, 8] work by dividing the image in fore- and background elements. The boundary is then defined implicitly between neighboring fore- and background image elements. Consequentially, constraints can be set by defining specific elements as fore- or background. The second type of energy minimization algorithms for segmentation are algorithms like Live Wire/Intelligent Scissors [22, 24], which trace a line through a set of given points (the constraints) which is supposed to bound the feature which is to be extracted. Here constraints are explicit, giving the user full control over the boundary. Depending on the application, both types of constraints have their advantages. For seismic interpretation it is crucial to be able to directly interact with the boundary (representing the seismic horizon), making explicit constraints a much better choice than implicit ones.

While graph cuts adapt naturally to 3D [2, 6], or higher dimensions in general (the cut in an n-dimensional graph is always of dimension n-1), the application of Live Wire/Intelligent Scissors will always produce a line. These lines can be used as a boundary for 2D objects, but not 3D objects, which are bounded by a surface. Poon et al. [29] exploit this property to track vessels in 3D. There are a number of publications dealing with the extension of Live Wire to 3D to create minimal cost surfaces. Falcão and Udupa [13, 12] propose automatic tracing in 2D using a user-defined trace on orthogonal slices as constraints. Their approach, however, requires considerable user supervision, especially when trying to segment objects with a topology different from a sphere. Schenk et al. [33] introduce a combination of Live-Wire and shape-based interpolation. Poon et al. [30, 31] present a technique similar to Falcão et al. [12, 13] which can handle features with arbitrary topology, by defining inner and outer contours. However, all these methods are based on creating some kind of network of lines computed using the classical 2D Live Wire algorithm, which has several drawbacks; A set of minimal cost paths in a 3D image does not necessarily assemble a minimum cost surface. Another major problem is that the placement of the *key paths* is crucial for the quality of the resulting surface, especially if the topology changes between slices as might be the case for concave objects. Additionally, from a user perspective it can be much harder to edit a network of minimal cost paths. For example, a constraint set in one slice will not necessarily propagate into the next slices, etc.

The only method we know of to find minimum cost surfaces in 3D with explicit constraints like in the Live Wire approach was proposed recently by Grady [15]. This work formulates the minimal cost surface problem as an extension of the original 2D minimal cost path problem and shows that it can be solved with linear programming. In later work, Grady [16] presents an optimized solution for regular 6-connected lattices using minimum-cost circulation network flows. This approach results in the real minimal cost surface and is also independent of topology. The algorithm requires a single arbitrarily shaped closed contour as input and allows discretionary closed contours as constraints.

3 Workflow

As input to our system we assume a seismic volume, where the x- and y-dimensions correspond to actual spatial coordinates, and the z dimension corresponds to the travel time of reflected seismic waves acquired by geophysics during reflection seismology.

3.1 Current Practice

The workflow of current seismic interpretation systems is typically based on the manual or semi-automatic interpretation of every n-th inline- and crossline-slice of a seismic volume, where n can range from skipping several slices each step to processing each slice individually, one by one. Generally, in these systems, multiple horizons are extracted for each slice. After all desired slices have been interpreted (i.e., horizons have been extracted), the gaps in between are filled using automatic growing and surface interpolation. In a subsequent step the geological model is built, based on the extracted seismic horizons. For building the geological model, the z-axis of the seismic volume (i.e., the time axis) has to be transformed to the spatial dimension z (i.e., the depth of the volume), the so called time-depth conversion. Therefore, the seismic volume data is transformed such that the previously extracted horizons fit and correspond to the additional spatial data provided by well logs. This is an iterative process. Often errors done during the interpretation phase will show in this step, requiring refinement of the extracted horizon. Because this current workflow is heavily based on 2D slice processing, this often results in multiple slices that need to be updated manually. Additionally, in this workflow the selected axis-aligned slices used for extracting horizons do not necessarily correspond to the well positions, making it much harder to fit horizon surfaces to actual well log data.

3.2 Proposed Workflow

Figure 1 displays our integrated and interactive workflow for seismic interpretation with focus on the linked 2D and 3D views for semi-automatic horizon tracing. Our system has been integrated into SimVis [11], a visual analysis framework with support for linked views, advanced data selection techniques, and data import and export functions.

In contrast to other seismic interpretation systems, we propose an interactive workflow that is based on well positions rather than axis-aligned slices. Conceptually, we always work on the subvolume that is defined by three well positions that make up a triangle on the geological surface and form a triangular prism through the

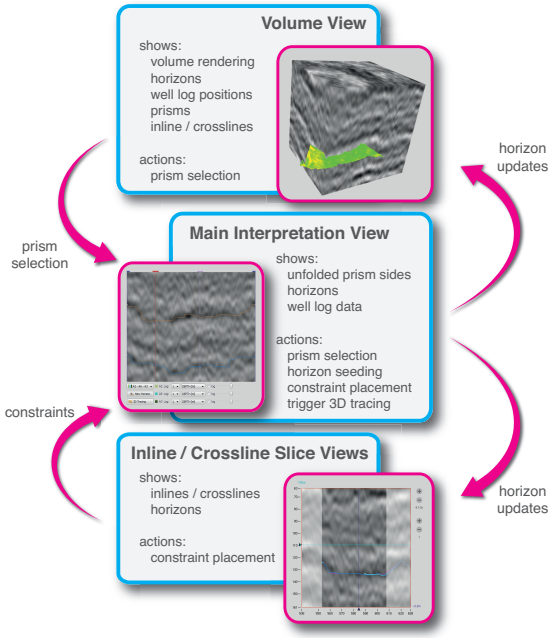

Figure 1: Linked-view interaction diagram. The different views of the system are coupled to each other, triggering actions upon interaction, and updating the other views accordingly.

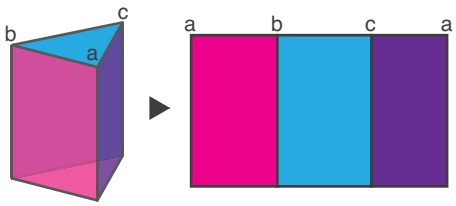

Figure 2: The original prism (left) unfolded into a single slice (right).

seismic volume along the z-direction. Therefore, the well logs that we use in our system are created by drillings that are roughly orthogonal to the surface, resulting in a unique (x,y)-coordinate for each well log. This approach was proposed by our collaborating domain experts. The main advantage of this approach is that the horizon can easily be adjusted directly at the well position, which is the point where errors in the interpretation become visible when adjusting the horizon to the ground truth data from the well log during time/depth-conversion of the seismic volume. Additionally, if horizons need to be adjusted during the depth-conversion of the volume, our semi-automatic method does not require the user to manually adjust several different slices, but can automatically update only those triangles that are adjacent to the corresponding well log. In the best case, a single constraint placed at the well position can be enough to correct all adjacent prisms. After extracting a horizon surface patch for one well log prism, the horizon can then be extended by processing the neighboring well log prisms, creating a set of horizon surface patches that constitute the single horizon surface. The result from one well log will therefore be transferred to the neighbors functioning as constraints for the two- and three-dimensional tracing, eliminating the need for seed points in all but the first prism.

In our approach, the volume is divided into a set of prisms by triangulating the well positions. The triangles are computed using Delaunay triangulation, guaranteeing non-intersecting triangles and resulting in the maximum area covered by the well logs, i.e. their convex hull. Additionally, we take advantage of the fact that Delaunay maximizes minimum angles. Small angles would result in very close edges and limit the freedom of the 3D tracing.
The interpretation itself is done triangle (prism) by triangle (prism), instead of slice by slice. To extend the horizon trace to areas not covered by wells, additional points of interest (e.g. the corners of the volume) can be added to the triangulation.

To compute the seismic horizons we have developed an algorithm that combines the advantages of semi-automatic 2D and 3D global minimal cost path algorithms. We separate the horizon extraction algorithm into a 2D minimal cost path tracing on prism faces orthogonal to the (x,y) plane, and a subsequent 3D minimal cost surface computation, which is initialized by the previously computed 2D contour. For more details on the tracing process refer to Section 4.1. Implementation details on the 2D part can be found in Section 4.1.1, 3D tracing is described in Sections 4.1.2 and 4.1.3.

Dividing the tracing into an easy to handle and fast 2D tracing, followed by the slower 3D tracing constricted by the result from the 2D trace has the advantage that the user can very quickly produce a large amount of constraints for the 3D tracing (i.e. the boundary) in the 2D step. This improves the quality of the 3D tracing, minimizing the need to set additional 3D constraints and thus avoids lengthy recomputation. In our system, the horizon tracing is started by the user selecting a triangle (i.e., three well logs) in the volume or timeslice view. Next, the sides of the corresponding prism are shown as a single unfolded slice in the main seismic interpretation view (see Figure 2). Starting with a single user-defined seed point, a horizon is automatically traced on the sides of each prism. We employ a minimum cost path algorithm similar to the Live Wire algorithm presented by Mortensen et al. [22]. Due to the cyclic nature of the prism sides, only a single user-provided input point is

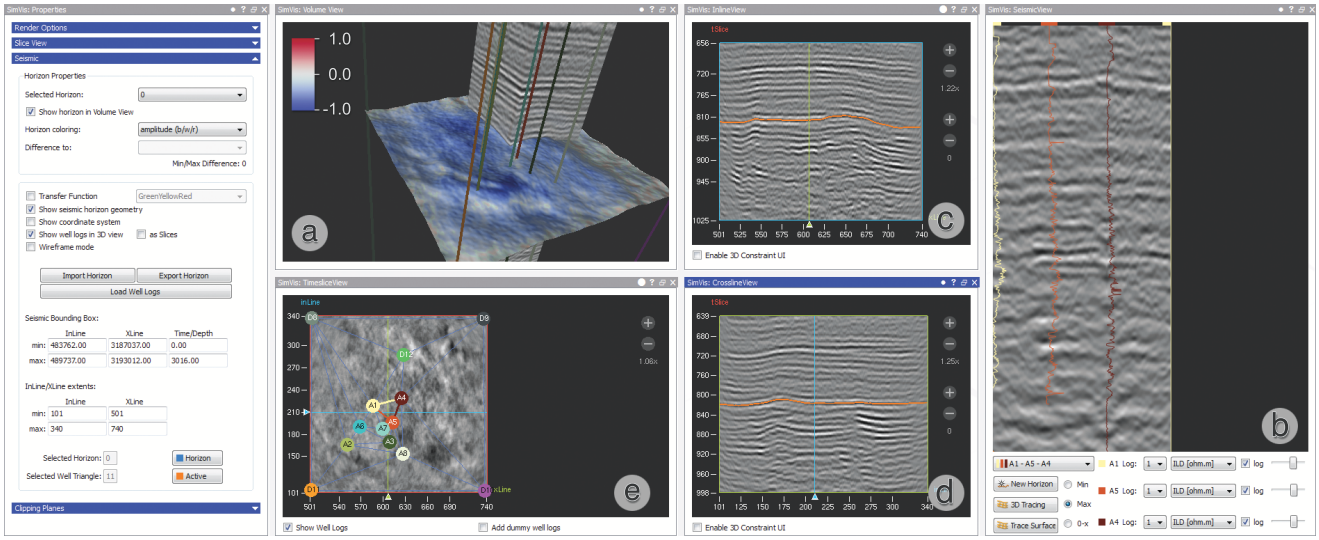

Figure 3: Our framework for interactive seismic interpretation using piecewise global energy minimization. The 3D view in (a) shows the volume alongside well positions, and a complete horizon surface. In the main interpretation view (b), the unfolded sides of a prism are shown overlaid with well logs. The slice views (c) and (d) show inline and crossline slices, respectively, (e) the timeslice view with the well positions. A horizon covering the complete volume was traced by adding prism vertices on the corners of the volume. The amplitude at the horizon vertices is plotted on the horizon using a cool to warm colormap.

required, which acts as start as well as end point for the traced 2D contour. The minimum cost path can be computed in real time for typical prisms, allowing the initial trace to be visualized instantly in the main interpretation view as well as in the volume view. In the latter, a surface inside the edited prism is interpolated right away to give a rough idea of the 3D surface.

Seismic data exhibits a high degree of uncertainty as well as a large amount of noise. Therefore, we have added the possibility for the user to add constraints to the automatic tracing algorithm by specifying additional points that have to be part of the extracted seismic horizon. New constraints instantly update the horizon and can be continuously modified by the user.

When a satisfying result for the horizon is reached on the prism faces, the user can initialize the 3D tracing. The points of the trace on the prism sides are used as constraints for the minimum cost surface algorithm to compute the horizon patch on the inside of the prism. The 3D minimum cost surface algorithm is much more computationally demanding than its 2D counterpart. Thus computation can take several seconds, but usually does not exceed more than 30 seconds for typical prism sizes. After the computation is finished, the interpolated surface in the volume view will be replaced by the correct minimum cost surface.

Adding constraints requires re-computation of the horizon patch. However, the comparably small size of the prisms in combination with the large number of constraints gained from the trace on the prism faces minimize the need for adjusting the 3D traces considerably. If, however, adjustments are needed, the fact that we use a real 3D tracing method over a network of 2D traces is a big advantage over slice-based approaches, as constraints can be set on the exact desired position and not only on key slices and every constraint has impact on the whole surface patch instead of just the current inline or crossline trace. Additionally, interpreted horizons can be imported and exported in a format readable by the industry standard software Petrel [34].

We believe that a boundary-constricted minimum cost approach, as described above is the optimal solution for horizon tracing in terms of horizon quality, as constraints define horizon elements directly rather than subsurface elements bounded by the horizon. If needed, the horizon can virtually be traced manually by placing constraints at every desired grid point, yet most of the time only very sparse constraints are needed. On the other hand, a global approach is very costly, making interactively tracing on complete volumes impossible. By dividing the volume in prisms, though, we are able to retain an interactive workflow with short waiting times between the 3D tracing of single prisms.

Figure 3 illustrates the integrated nature of our system. All our views are linked. If one view is updated, all corresponding views are updated as well. The 3D volume view (Figure 3a) shows all interpreted horizon parts in addition to the well tops in context of the seismic cube. Embedding the horizon surfaces in the volume rendering allows quick verification of the tracing results. This view can also be used to select a prism for editing. Figure 3b shows the main interpretation view. Here the unfolded sides of the selected prism as well as the corresponding well logs are displayed. For every log, multiple attributes gathered from the drillings can be visualized as 1D curves. This view is used to set the initial seed point for each horizon and allows to modify the boundary of the horizon patch in the current prism.
Additionally, we have linked the 2D inline and crossline slice views (Figure 3c/d). When the currently displayed inline or crossline slice intersects the horizon surface, the intersection contour is displayed immediately. These views can also be used to add constraints for 3D tracing. Figure 3e shows a slice view from the top, the so called timeslice view. Well positions and triangles are displayed and can be selected here.

4 Seismic Horizon Surface Tracing

Our system is based on automatic global energy minimization of seismic horizon patches, according to a cost function and user-specified constraints, leading to a piecewise optimized surface. Due to noise and the ambiguous nature of seismic data in general, the minimal cost path or minimal cost surface might not immediately match the desired horizon after initialization. Therefore, we allow the user to add an arbitrary number of constraint points to guide the tracing algorithm, forcing contours and surfaces through selected points on the horizon.

When adapting techniques like intelligent scissors for seismic interpretation, horizons are defined by the boundaries produced by the algorithms. As indicated before, precise control over the boundary is necessary in cases of false classification. This can only be guaranteed by explicit constraints. However, in contrast to graph cuts, which extend naturally to 3D, a 3D extension of intelligent scissors is not straightforward. Whereas implicit boundary approaches always operate on image elements, regardless of dimensionality,

explicit boundary approaches operate on the boundary elements which differ between dimensions. Thus, lines in 1D are bounded by points, areas in 2D are bounded by closed curves, volumes in 3D are bounded by closed surfaces, and so on. Until recently, there was no real extension of 2D minimal cost path algorithms with explicit boundary constraints to 3D minimal cost surfaces. Most proposed methods were based on simple networks of 2D minimal cost paths. A set of minimal cost paths in a 3D image, yet does not necessarily define a minimum cost surface. A simple example that illustrates this is the minimal surface connecting two circles in 3D [16]. Connecting opposing points on both circles with minimal energy results in a cylinder. However, the minimal cost surface bounded by the same two circles is a curved shape called a catenoid.

4.1 Minimal Horizon Surface Computation

Our approach for computing horizon surfaces using global energy minimization is based on previous work of Grady [16]. In that paper, Grady formulates the minimization problem for *boundary constricted* 3D minimum cost surface computation and shows that minimal-cost circulation network flow (MCNF) is an optimized solution for the dual of this particular problem. By circulating the maximum possible amount of flow in the dual graph through any initial surface z_0 fulfilling the boundary constraints, eventually a set of edges will be saturated. This forms a cut, which corresponds to the minimal cost surface in the primal. The process is similar to the maximum flow, minimum cut method used for graph cuts, with the type of constraints being the main difference. Graph cuts define fore- and background elements in the image as constraints, which then form the source and sink nodes in the graph, whereas for MCNF, with no source or sink nodes present, the constraints are added as the boundary of the initial surface z_0.
Our algorithm for computing minimal horizon surfaces consists of the following main steps:

1. A closed contour bounding the desired surface within a prism has to be computed. This is done in a flattened representation of the prism's sides (see Figure 2).
2. Next, the volume contained in the prism has to be represented as a weighted, directed graph.
3. The initial surface z_0 fitting the boundary acquired in the first step needs to be inscribed in the graph.
4. The MCNF through z_0 is computed.
5. The minimal-cost surface is constructed from the set of saturated edges of the circulation, bounded by the initial contour.
6. If the surface is not satisfactory it can be forced through desired points in the volume by placing additional constraints. These constraints are then added as additional (inner) boundary patches in step 3. Steps 4 and 5 need to be reapplied.

Below, the steps of our algorithm are explained in more detail.

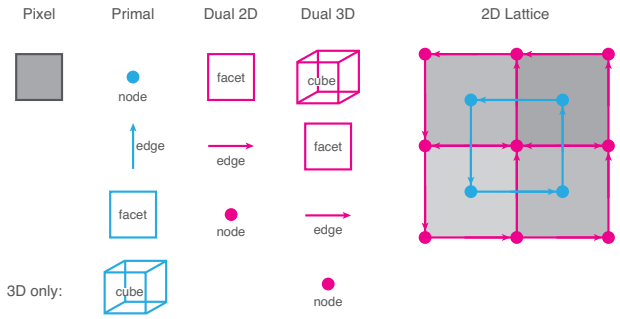

Figure 4: Correspondence between picture elements and items in the primal and dual graph.

4.1.1 2D Contour Creation

The initial closed contour for each prism is computed as a minimal-cost path on the prism sides. For any prism, the three sides are re-sampled from the volume using texture mapping with tri-linear interpolation, and stitched together into a single image. This image, superimposed with selected data of the cornering well logs, is shown in the main interpretation view and also used to construct the graph used for the minimum cost path computation. Figure 4 shows the mapping of an image to primal and dual graph elements. In contrast to standard approaches in image segmentation [16, 23], where the minimal-cost path/surface computation is done on the dual graph, resulting in a boundary in between image elements, the seismic horizons we want to extract are represented in the data by bands of extremal values. Thus, for seismic interpretation we do not compute the minimal-cost path in between the voxel grid but directly on the grid. The underlying graph then has to be the primal graph, instead of the dual graph. The resulting minimal cost path will be a set of edges connecting pixels in the image instead of dividing pixels.

We have implemented the minimal cost path computation using Dijkstra's algorithm [10]. The algorithm is applied on the primal graph constructed as a 4-connected lattice according to Figure 4, based on the stitched image of the prism sides. Edge weights are defined using the cost function defined in Section 4.1.4. The minimal path has to be bounded by at least two nodes, otherwise it will not contain any edges. To be able to compute the minimal cost path using just a single seed point, the cyclic nature of the prism sides has to be taken into account when creating the graph. In addition to the regular lattice arcs connecting each node on the right-most side of the graph with its counterpart on the left-most side and vice versa are added. To coerce the path around the prism, all arcs directly to the right of the seed, but pointing to the left are removed. That way, the single seed point functions as start- and endpoint of the path and it is ensured that the path completely encircles the prism.

Dijkstra's algorithm cannot handle constraints directly. However, constraints can be added simply by dividing the problem. Instead of trying to compute the complete path at once, the path can be coerced through any node (the constraint) in the graph by computing the minimal cost path in segments. Every constraint will then bound two connected segments, one as a start- and one as a target-node.

4.1.2 3D Graph Initialization and MCNF

MCNF describes the problem of putting as much flow as possible through the edges of a network at minimal cost, without adding or draining flow with dedicated source or sink nodes. Therefore, a capacity and a cost is assigned to every edge. As the graph does not contain a source or a sink, the incoming and outgoing flow at every node must be equal, thus flow can only circulate in the graph. The cost is always prioritized over the capacity of edges. If a cycle adds cost to the total flow network, it will not be used, regardless of its capacity. The capacity of a cycle is defined by the lowest capacity of the edges it consists of and the total flow in the graph is defined by the accumulated flow of all cycles.

To compute the maximum flow in a prism, the dual graph is created by using all voxels contained in the prism, according to the volume-graph scheme shown in Figure 4. The graph is constructed in two steps: first the basic graph structure of the dual is created (Figure 5a) and costs and capacities are assigned to each edge (Figure 5b). In this step, a cost of 1 is used for all edges (not indicated in the figure). The capacities of the dual edges correspond to the costs of the primal facets and are computed using Equation 3 in Section 4.1.4. The edges contained in the outer facets in the graph, have to be treated separately. These edges do not have all four adjacent voxels and thus the cost cannot be computed using the cost function. We initialize these edges with infinite capacity. Edge ca-

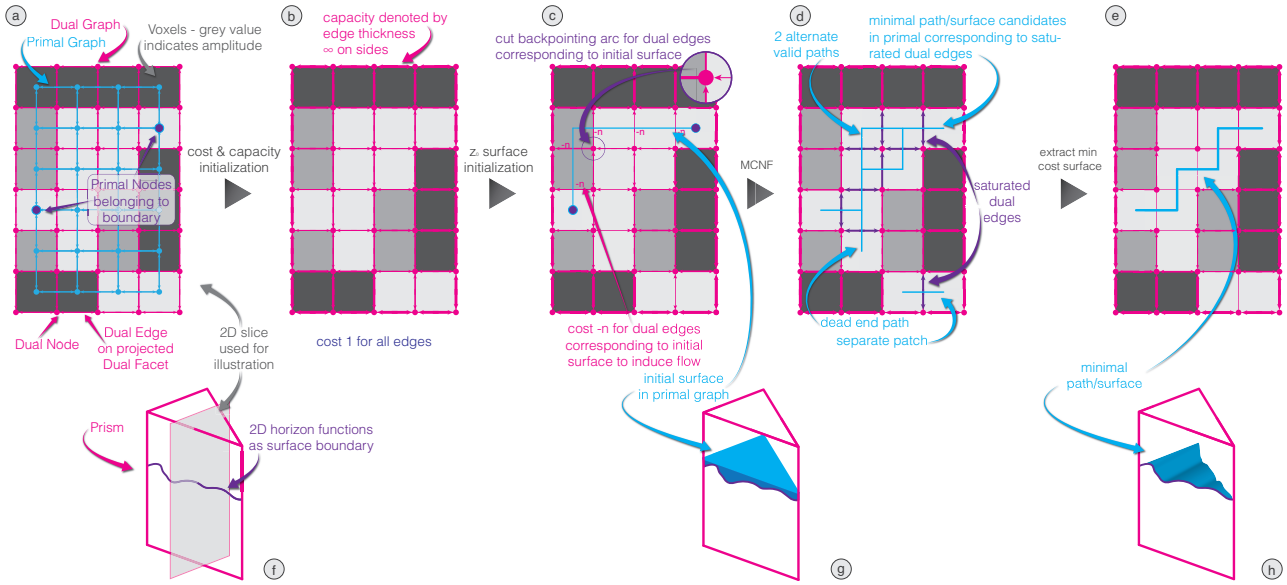

Figure 5: Tracing pipeline.

pacities are indicated in Figure 5b by the thickness of the edge. At this point there will be no flow in the graph, because all edges have positive cost. In the second step, the initial surface z_0 is added to the graph. The initial surface is a set of facets in the primal, bounded by the closed contour obtained from the prism sides.In Figure 5c, the boundary is indicated by the two purple primal nodes, the initial surface by the path between these nodes. To induce flow through the initial surface, negative costs are assigned to the dual edges corresponding to the primal facets of z_0. To allow all possible paths, the cost is the negative of the sum of all positive costs in the graph, such that every path through z_0 is negative and thus reducing total cost for the circulation. Additionally, we allow flow through the surface only in one direction by cutting all edges in the opposite direction. This guarantees that no cycles going back and forth through the surface are generated. Here it becomes obvious why we assign infinite capacity to the outer edges in the first step. All allowed cycles will go through the initial surface and the inner edges in one direction and come back on the (much fewer) edges on the faces of the prism. If the capacity of these edges were delimited, the algorithm would most likely terminate, delivering a set of saturated edges on the outside of the prism which would not be helpful for finding the minimal-cost surface.

For most image segmentation approaches, the initial 2D trace will be performed on one slice in the volume, resulting in a planar boundary. In that case, the part of the slice bounded by the obtained 2D trace is the simplest match for the initial surface z_0. The 2D trace in our approach, does not result in a single planar boundary, however, a very simple initial surface can be constructed dividing the surface into four planar parts. First we cut the prism above the highest point in the trace with an orthogonal slice, resulting in a triangle shaped boundary in that slice. This triangle functions as the cover of our surface. The other three parts are on each side of the prism, bounded by the initial trace on the bottom, the cover slice on the top and the prism edges on the sides. Figure 5g shows a sketch of an initial surface fitting a 2D horizon path on the sides of a prism, the cover is depicted in light, the sides in dark cyan, the 2D trace on the prisms sides by the purple curve.

For the actual MCNF computation we use the LEMON C++ graph library [1], which can compute the flow for all edges in the graph. This is used to identify the saturated edges in the graph.

4.1.3 Horizon Extraction

The set of saturated edges does not translate directly to the desired minimal cost surface. In real world datasets, usually a large amount of edges not contributing to the minimum cost surface are saturated. E.g., in our application, tracing a horizon in a prism of a thousand points per layer can easily result in over 10,000 saturated edges of which 2,000 contribute to the actual surface. Figure 5d shows the cases of saturated dual edges which have to be removed. Dual edges/primal facets not connected to the initial boundary could simply be avoided by growing the surface from the initial boundary. However, dead end paths/surface patches, connected to the surface would not be removed by growing.

Instead we prune facets by checking that all four of the facet's edges are either shared with another facet candidate, or are part of the initial boundary. By iterating over all facets and discarding facets which do not fit these criteria, the set of surface candidates is gradually reduced. Once a pass over all facet candidates finishes without finding any more facets that need to be discarded, the remaining facets form the minimal-cost surface and the algorithm returns (Figure 5e/h). If two alternate surfaces bounded by the initial contour exist (top left part of the path in Figure 5d), both must be of the same cost and thus are equivalent solutions. As the user can always force one of these we just use the first complete closed surface found by the algorithm.

4.1.4 The Cost Function

We define edge and facet weights using a cost function containing two components.

The first component defines the *snappyness* of the surface to ridge and valley lines/surfaces in the image/volume. It calculates the deviation of the grey values of all current voxels (i.e., the voxels corresponding to the current edge or facet whose cost is to be defined) from a specified target value. We allow tracing of minima, maxima and zero crossings. Before tracing the kind has to be chosen, resulting in a suitable target value (e.g., for a $[-1.0..1.0]$-scaled volume -1.0 for minima, 0.0 for zero crossings and 1.0 for maxima).

The second component of the cost function is important for the *smoothness* of the traced line/surface. It is defined by the similarity of the voxels belonging to the edge/facet in question. The higher the difference of voxel grey values, the higher the cost. If needed, the relative weight of the two cost function components can be adjusted.

The dimensionality of the boundary items in the primal graph varies from dimension to dimension. In the dual, however the bounding item is always an edge. Thus we define the cost using the voxels adjacent to an edge in the dual graph.

The snappyness component is defined by:

$$w_1 = \sum_{k=1}^{m} |t - f(v_k)|, \quad (1)$$

where $f(v)$ is the grey value of the image element corresponding to node v, t is the target amplitude, and m is the number of adjacent image elements (i.e. two pixels in 2D, four voxels in 3D). Accordingly,

$$w_2 = \sum_{k=1}^{m} \left| f(v_k) - \frac{1}{m} \sum_{l=1}^{m} f(v_l) \right| \quad (2)$$

describes the smoothness component of the cost function and

$$w = cw_1 + (1-c)w_2 \quad (3)$$

the complete cost function using Equations 1 and 2, including a constant $c \in [0..1]$ for linear blending between w_1 and w_2.

5 Seismic Horizon Visualization

Visualization of the extracted horizon surface is done simultaneously in all 2D and 3D views. The 2D inline and crossline views display a slice of the volume with overlaid horizon data. All extracted horizons are displayed as 2D contours, with the currently active horizon and prism being highlighted. The current horizon can be modified at any time by clicking on and dragging the contour. After the first step of the minimal cost path tracing algorithm, the inner part of the horizon is linearly interpolated, whereas after 3D surface tracing, the actual minimal surface is displayed in the 2D view.

The 3D view combines direct volume rendering of the seismic cube with the visualization of the horizon surface geometry and the well logs. Volume rendering is performed in a single-pass GLSL raycaster. For correct visibility of the horizon surfaces and well logs we simply adjust the depth values of the ray stop-positions used in raycasting, and draw the geometry prior to the volume [32]. Additionally, the 3D view enables picking of well log prisms and of already extracted horizon surfaces and is linked to the seismic interpretation-, inline-, and crossline view. The 3D view is used for visual inspection of the ongoing seismic interpretation and supports plotting of several horizon properties (i.e., amplitude, deviation from the target amplitude, cost, and the distance to a second horizon) directly on the horizon's surface geometry. This additional information is computed on-the-fly during rendering using a combination of vertex and fragment shaders. Deviations and cost are computed as scalar values, scaled to $[0..1]$ and mapped onto a blue to red colormap presented in [21] for easy interpretation. The amplitude can either be visualized as simple grey values or with the same blue to red scheme used for the other scalar values. However, here blue is mapped to minus one and red to one. Surfaces with cost and distance to another surface, respectively, can be seen in the results section (Figure 6).

6 Results

To support an interactive workflow for seismic interpretation, we have integrated our system into the SimVis visual analysis framework. The system was recently installed at our domain expert collaborators for evaluation. Figure 6 shows 3 horizons extracted from a 240x240x1509 voxels seismic dataset using the proposed technique. The horizon was extracted in the area covered by 8 wells, covering roughly one fourth of the dataset. All the horizons were traced using a single seed point plus between five and eight constraints. All constraints were set on the sides of the prism, none on the inside. For details see Table 1. For comparison, we plotted the distance to the same horizons manually traced by our domain experts onto the surfaces using the cold to warm color map presented in [21]. For the first two examples, the distance was at most 2.39 pixels. The second horizon showed a very strong reflection in large parts of the volume, resulting in the shorter editing time as well as faster tracing. For the third example, the maximum distance was over four pixels. However, it should be noted that the area of largest distance in the last example seems to actually be of higher accuracy with our approach. Figure 6f shows a cutout of the slice views showing the horizon trace (supposed to be a maximum trace) with our method in magenta and the manually traced horizon in blue. The manually traced horizon is further away from the area of maximum amplitude to allow for an overall smoother surface. Outside of the area of interest the manual interpretation was done very sparsely resulting in a rough approximation of the horizon only. With a (semi-)automatic approach like ours the horizon can be traced exactly over the complete volume at low cost. It should be noted that with placement of additional constraints, the distance to the manually traced surface can be further reduced. In theory, it would even be possible to completely mimic the manual interpretation process by placing constraints on every grid point.

	Constraints	Edit t/min	Tracing t/s	Max Dist.
a	$1+8$	< 5	61	2.39px
b	$1+5$	< 2	53	2.27px
c	$1+5$	< 5	74	4.30px

Table 1: Details for the horizons shown in Figure 6. Number of constraints (seed + additional constraints), editing time, tracing time and maximum deviation from manually traced ground truth data.

Rendering of the interpreted horizons alongside the volume in a 3D view happens in real time. Rendering a dataset of size 240x240x1509 with three interpreted surfaces and additional well geometry results in more than 30 fps on a Geforce 280 GTX at full screen resolution (1920x1080 pixels).

7 Conclusions

We have presented an interactive workflow for rapid interpretation of seismic volumes using a combination of 2D and 3D cost minimization techniques. The workflow guides the user through the interpretation process using a combination of multiple linked 2D and 3D views. The prism-based workflow, instead of working along inline and crossline directions only, was received very well by our collaborators. It bounds the size of the working volume, and allows incrementing the piecewise optimal horizon surface bit by bit, by subsequently appending additional optimal surface patches that were created by processing neighboring well log prisms. Decoupling the interaction in 2D and the eventual computation in 3D was also perceived very well by the experts, as it simplifies user interaction and time expenditure considerably. Visualization of the extracted horizons, superimposed with additional information and in context of the original 3D volume, gives instantaneous feedback on the quality of the seismic interpretation. Additionally, our approach is scalable to large seismic volumes, because the time needed for computing minimal-cost paths and surfaces depends on the size of the selected prism instead of the volume size.

Acknowledgements

This research project was funded in parts by the Austrian Research Funding Agency (FFG), in scope of the SEIVIS project (No. 818063). The datasets are courtesy of Heinemann Oil GmbH.

References

[1] LEMON library for efficient modeling and optimization in networks: `http://lemon.cs.elte.hu`.

[2] C. Armstrong, B. Price, and W. A. Barrett. Interactive segmentation of image volumes with live surface. *Computers & Graphics*, 31(2):212–22, 2007.

[3] A. Blinov and M. Petrou. Reconstruction of 3-d horizons from 3-D seismic datasets. *IEEE Transactions on Geoscience and Remote Sensing*, 43(6):1421–1431, 2005.

[4] H. G. Borgos, O. Gramstad, G. V. Dahl, P. L. Guern, L. Sonneland, and J. F. Rosalba. Extracting horizon patches and geo-bodies from 3d seismic waveform sequences. *SEG Technical Program Expanded Abstracts*, 25(1):1595–1599, 2006.

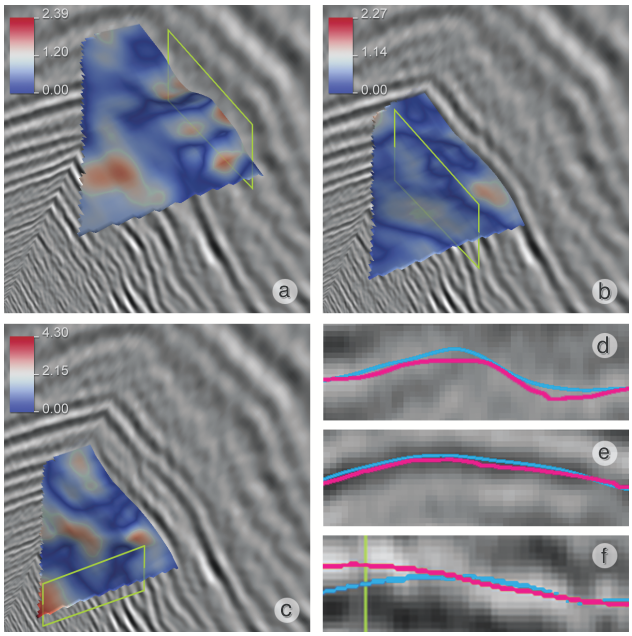

Figure 6: Three horizon patches computed using our algorithm. (a), (b) and (c) show the distance in pixels from a manual trace of the same horizon. (d), (e) and (f) show detailed slice view cutouts of the surface indicated by the green boxes in (a), (b) and (c). Manual traces are in cyan, traces computed with our method in magenta. (d) shows a region of the first horizon, with > 2 pixel difference, where the manual trace is clearly better than the automatic trace, manual and automatic traces in (e) are very similar. (f) shows a part of the horizon with > 4 pixel difference. Here, the automatic trace is closer to the actual ridgeline.

[5] Y. Boykov and M.-P. Jolly. Interactive organ segmentation using graph cuts. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 147–175, 2000.

[6] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *ICCV 2001. Proceedings of the Eighth IEEE International Conference on Computer Vision*, pages 105–112 vol.1, 2001.

[7] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *EMMCVPR '01: International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, 2001.

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

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

[10] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[11] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the 5th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2003)*, pages 239–248, 2003.

[12] A. Falcão and J. Udupa. A 3d generalization of user-steered live-wire segmentation. *Medical Image Analysis*, 4(4):389–402, 2000.

[13] A. X. Falcão and J. K. Udupa. Segmentation of 3d objects using live wire. In *Medical Imaging 1997: Image Processing*, pages 228–235, 1997.

[14] M. Faraklioti and M. Petrou. Horizon picking in 3d seismic data volumes. *Machine Vision and Applications*, 15:216–219, 2004.

[15] L. Grady. Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3d with application to segmentation. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 69–78, 2006.

[16] L. Grady. Minimal surfaces extend shortest path segmentation methods to 3d. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):321–334, 2010.

[17] N. Keskes, P. Zaccagnino, D. Rether, and P. Mermey. Automatic extraction of 3-d seismic horizons. *SEG Technical Program Expanded Abstracts*, 2(1):557–559, 1983.

[18] O. D. Lampe, C. Correa, K.-L. Ma, and H. Hauser. Curve-centric volume reformation for comparative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1235–1242, 2009.

[19] P. Lavest and Y. Chipot. Building complex horizons for 3-d seismic. *SEG Technical Program Expanded Abstracts*, 12(1):159–161, 1993.

[20] E. M. Lidal, T. Langeland, C. Giertsen, J. Grimsgaard, and R. Helland. A decade of increased oil recovery in virtual reality. *IEEE Computer Graphics and Applications*, 27:94–97, 2007.

[21] K. Moreland. Diverging color maps for scientific visualization. In *Proceedings of the 5th International Symposium on Visual Computing*, pages 92–103, 2009.

[22] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 191–198, 1995.

[23] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

[24] E. N. Mortensen, B. Morse, W. A. Barrett, and J. K. Udupa. Adaptive boundary detection using 'live-wire' two-dimensional dynamic programming. In *Computers in Cardiology 1992. Proceedings.*, pages 635–638, 1992.

[25] D. Patel, S. Bruckner, I. Viola, and M. E. Gröller. Seismic volume visualization for horizon extraction. In *Proceedings of the IEEE Pacific Visualization Symposium 2010*, pages 73–80, 2010.

[26] D. Patel, C. Giertsen, J. Thurmond, J. Gjelberg, and M. E. Gröller. The seismic analyzer: Interpreting and illustrating 2d seismic data. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1571–1578, 2008.

[27] D. Patel, Ø. Sture, H. Hauser, C. Giertsen, and M. E. Gröller. Knowledge-assisted visualization of seismic data. *Computers & Graphics*, 33(5):585–596, 2009.

[28] R. Pepper and G. Bejarano. Advances in seismic fault interpretation automation. *Search and Discovery Article 40170, Poster presentation at AAPG Annual Convention*, pages 19–22, 2005.

[29] M. Poon, G. Hamarneh, and R. Abugharbieh. Live-vessel: Extending livewire for simultaneous extraction of optimal medial and boundary paths in vascular images. In *Lecture Notes in Computer Science, Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 444–451, 2007.

[30] M. Poon, G. Hamarneh, and R. Abugharbieh. Segmentation of complex objects with non-spherical topologies from volumetric medical images using 3d livewire. In *SPIE Medical Imaging*, volume 6512-31, pages 1–10, 2007.

[31] M. Poon, G. Hamarneh, and R. Abugharbieh. Efficient interactive 3d livewire segmentation of objects with arbitrarily topologies. *Computerized Medical Imaging and Graphics*, 32(8):639–650, 2008.

[32] H. Scharsach, M. Hadwiger, A. Neubauer, and K. Bühler. Perspective isosurface and direct volume rendering for virtual endoscopy applications. In *Eurovis 2006*, pages 315–322, 2006.

[33] A. Schenk, G. Prause, and H. O. Peitgen. Efficient semiautomatic segmentation of 3d objects in medical images. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 186–195, 2000.

[34] Schlumberger Information Solutions. Petrel seismic to simulation software: `www.slb.com/services/software/geo`.

[35] P. Tu, A. Zisserman, I. Mason, and I. Cox. Identification of events from 3d volumes of seismic data. In *IEEE International Conference on Image Processing, ICIP '94*, volume 3, pages 309–313, 1994.

[36] U.S. Energy Information Administration. International energy outlook, 2010.

[37] Visualizaton Sciences Group. Avizo earth: `www.vsg3d.com/avizo/earth`.