Graphical Models 84 (2016) 1–14

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Interactive cutaways of oil reservoirs

Felipe Moura de Carvalho[a,*], Emilio Vital Brazil[b], Ricardo Guerra Marroquim[a], Mario Costa Sousa[b], Antonio Oliveira[a]

[a] Systems Engineering and Computer Science Program, Federal University of Rio de Janeiro, Cidade Universitária Ilha do Fundão, Rio de Janeiro, Brasil

[b] Department of Computer Science, University of Calgary, 2500 University Dr. NW, Calgary, Alberta, Canada

ARTICLE INFO

Article history:
Received 13 June 2015
Revised 16 January 2016
Accepted 1 February 2016
Available online 27 February 2016

Keywords:
Cutaway
Corner-points
Visibility
Clipping
GPU

ABSTRACT

In the Oil and Gas industry, processing and visualizing 3D models is of paramount importance for making exploratory and production decisions. Hydrocarbons reservoirs are entities buried deep in the earth's crust, and a simplified 3D geological model that mimics this environment is generated to run simulations and help understand geological and physical concepts. For the task of visually inspecting these models, we advocate the use of Cutaways: an illustrative technique to emphasize important structures or parts of the model by selectively discarding occluding parts, while keeping the contextual information. However, the complexity of reservoir models imposes severe restrictions and limitations when using generic illustrative techniques previously proposed by the computer graphics community. To overcome this challenge, we propose an interactive Cutaway method, strongly relying on screen-space GPU techniques, specially designed for inspecting 3D reservoir models represented as corner-point grids, the industry's standard.

1. Introduction

The Oil and Gas industry is considered to be one of the largest industries of all times, involving many large-scale companies worldwide. The increasing demand for energy, along with the gradual depletion of the easy accessible hydrocarbons reservoirs, has motivated the industry to maximize the oil recovery of existing fields. This shift stimulated intensive research to better understand complex fluid flow mechanisms that occur inside the oil fields, as well as the development of advanced reservoir visualization and simulation techniques [1].

In order to understand the flow behavior of a reservoir, a 3*D* computational geological model is created by domain experts. This model represents the reservoir geometry, its intrinsic properties and its fluid content. Once the geometry is constructed, the other properties of the model should be filled, and a tuning process, known as *history matching*, is triggered. Based on sparse data the model is populated with static properties, then the flow is computed through a simulation, and the results are tested against observed data. If the computed result diverges from the observed data, the model parameters are manipulated and the simulation is restarted. This interactive process is repeated until an acceptable degree of accuracy is reached.

Visual inspection is an important asset for analysis in all steps of this interactive process, as well as when using the final model as a predictive tool. The standard representation for oil reservoirs is corner-point grids, and there are in the order of millions corner-point models employed by the industry nowadays. Traditionally, geologists use parallel cuts (a plane oriented with one of the axes) to see the interior of the grid, or manually set a range of cells to be removed from the visualization. Nonetheless, because these

* Corresponding author.
E-mail addresses: felipe.celer@gmail.com (F. Moura de Carvalho), evbrazil@ucalgary.ca (E. Vital Brazil), marroquim@cos.ufrj.br (R. Guerra Marroquim), smcosta@ucalgary.ca (M. Costa Sousa), oliveira@cos.ufrj.br (A. Oliveira).

http://dx.doi.org/10.1016/j.gmod.2016.02.001
1524-0703/

models have volumetric characteristics, and some important phenomena may occur in the interior cells that are usually occluded by several layers of outer cells, they are hard to visualize using naive methods. These simple methods do not preserve context, so it is difficult to have a clear insight of what is happening around the cells of interest. Furthermore, removing cells manually offer very little intuitive visual control. Wireframe also does not provide a good visualization choice, since it produces a lot of cluttering for large models.

Illustrative techniques are a natural way to solve this issue. A commonly used method is *Cutaways*, where the artist creates the illusion of the object being cut to expose its interior without losing the general context. Researchers have investigated solutions to generate these illustrations automatically using computational methods. Given that an object in focus (the part or object of interest) is selected or identified, the goal is to compute the appropriate cut surface that eliminates occluding parts.

In this paper, we present a method for generating cutaway visualizations of reservoir models. We follow suggestions on how to generate meaningful cutaways from previous studies (e.g., Lidal et al. [2] and Sigg et al. [3]), and transfer these good practices to corner-point models. However, due to the volumetric nature of reservoirs, a straightforward application of known methods performs poorly with these complex models. Our method works by selecting a subset of the volumetric cells from a range of attribute values, such as pressure or porosity. These are the *primary* cells, since they are the focus of the visualization. All other cells are *secondary* cells, and are either removed when occluding primary ones, or used to increase context perception.

Our cutaway visualization method is based on a flexible screen-space representation for clipping volumetric cells. The main contributions are:

- the cutaway surface is dynamically defined as the union of frustums that better preserves context by tightly adapting to the corner grid structure;
- the volumetric intersection is generated on-the-fly and on a per fragment basis, allowing for a precise cut of the grid structure;
- a local ray-casting procedure to render important edges;
- some feature emphasizing techniques to enhance the cuts and provide more visual clues during the inspection process, such as contour lines and shading techniques.

In Fig. 1 we illustrate a corner-point model and the proposed technique in action, while Fig. 2 shows an overview of our method. The method achieves interactive framerates even for complex topology and geometry.

Apart from the technical points above, we have carried an informal user study with domain experts that suggests that our visualization method applied to corner-point models can have a great impact in some important tasks in the petroleum exploration processes. It may help petroleum engineers answer one of the most critical questions by providing insightful visualizations: where should the wells be placed to maximize oil recovery.

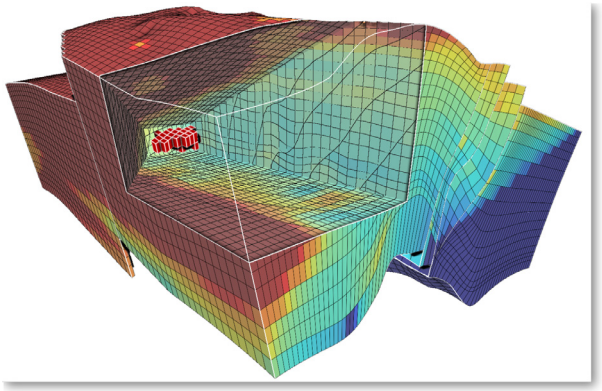


Fig. 1. Our cutaway method in action with a real reservoir model. The colors correspond to the Oil Saturation dynamic property. The reservoir volume is cut to visualize some selected cells of interest in its interior, while maximizing the visual comprehension of the context around them.

The remainder of this paper is as follows: in Section 2 we present a review of the related works that inspired our technique. A brief introduction to reservoir engineering is given in Section 3 to present our study domain and some terminologies. Our technique is detailed in Section 4, and in Section 5 we present some results. In Section 6 we describe a design critique carried out with two domain experts. In Section 8 we provide our final remarks and discuss some directions for future works.

2. Related works

Complex models arise in various domains, such as architecture, manufacturing industry, and medical imaging. The conception of even larger and more detailed models has brought challenges for visual comprehension and making inferences on the data. One important issue is occlusion, i.e., when key parts of the models are nested inside or hidden behind others. In this case, artists often make use of the cutaway technique, that removes less important parts of the object to reveal the most significant ones. Researchers in computer graphics have been inspired by these technical illustrations and have developed algorithms to automate the process of creating cutaway illustrations. Here we review some of these works that are most related to our approach.

2.1. Cutaway in volume rendering

Volume rendering methods create visualizations by employing opacity functions to reveal the interior structures, or by extracting iso-surfaces. However designing a suitable transfer function can be a challenging task. Some approaches have aimed at increasing the visual understanding of a selected region by using cutaways or similar techniques.

Bruckner et al. [4] describe a focus+context method to highlight regions of interest in volumetric models. They use the idea of lighting-driven feature classification, creating images that resemble artistic illustrations. In a previous work, Bruckner and Gröller [5] proposed a discrimination of interior and exterior parts by preserving

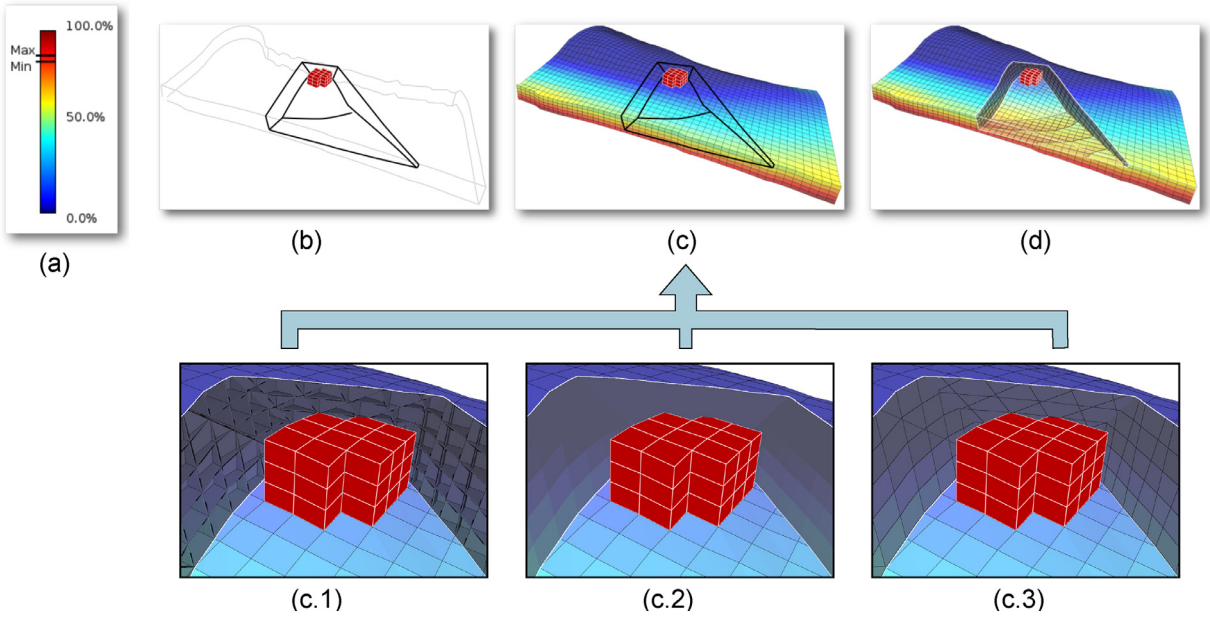


Fig. 2. An overview of our method. **(a)** The user selects a value range to specify the primaries. **(b)** The selected primary set is rendered and a frustum is created around them. **(c)** The frustum is used to clip secondary cells: **(c.1)** fragments inside the frustum are clipped, leaving a hollow appearance (Section 4.1); **(c.2)** the normals of the clipped fragment are modified to fill the holes (Section 4.2); **(c.3)** intersection lines are computed using a ray-casting approach (Section 4.3). **(d)** The resulting cutaway illustration.

clear shape cues, in order to maximize context. By separating the volume into three categories (background, ghost, and selection), they are able to achieve a simple cutaway effect by modulating the ghosting effect. By the same token, Wang et al. [6] propose a framework based on the Magic Lens metaphor [7]. They advocate the use of various types of lenses to magnify regions of interest. The basic concept is the same for cutaways, to enhance the visual perception of a region or object, without suppressing the rest. Kruger et al. [8] implement a similar system by combining several layers of the volumetric data in the hotspot region. They propose different shading strategies to enhance visual understanding based on values such as normals and curvatures, as well as a user defined focus region. Viola et al. [9] propose importance-driven volume rendering, that generates automatic cutaway visualizations based on the object and view direction. Their method is based on a depth image representation modified by the Chamfer distance transform [10] to clip parts of the models in front of the object of interest.

In spite of all the advances in volume rendering methods, they are not specially suitable for corner-point models. In this representation there is no underlying surface that should be extracted or highlighted using transfer functions, since the corner-points are already constructed from the set of surfaces. Furthermore, naively changing the transparency of the secondary cells results in poor contextual information around the cells in focus. In fact, for corner-point grids, transparency should be avoided in order to not lose the internal lines resulting from a hard cut, i.e., the interface between the cutaway and the non clipped secondary cells (see Fig. 1). These lines convey significant information about the model's internal structure, and rendering these features in a clear way is important.

2.2. Cutaway in polygonal rendering

Following the concept of Intent-Based Illustration Systems [11], Feiner and Seligmann [12] developed a system that makes use of image processing techniques to expose hidden objects. The system generates two masks: a *cutaway-mask* based on the z-buffer for clipping the models; and another called *edge-mask* to highlight the cutaway boundaries. Another early work that resembles the cutaway illustrations was proposed by Viega et al. [13], that extends to a 3*D* environment the metaphor of a see-through interface of the Magic Lenses proposed by Bier et al. [7].

Diepstraten et al. [14] classify cutaways in two subclasses: cutout illustrations and breakaway illustrations. For both cases, they present several methods to create cutaways in polygonal models as well as a set of rules to apply the cuts. In order to enhance the experience in virtual environments, Coffin and Hollerer [15] propose an interface to make arbitrary cuts on the occluding geometry. This method enables the ability to see through solid walls with a virtual X-ray vision, the analog for cutaways for virtual environments.

Li et al. [16] present a formal survey of illustration conventions to reduce occlusion, as well as techniques to algorithmically apply them. In the case of cutaways, they developed an authoring system to help users establish parameters, place the cut object, and set good viewpoints that mimic artistic illustrations. While their technique reveals objects of interest, it is intended to produce a static view of the scene, thus providing a limited solution for exploring complex 3*D* dynamic and interactive scenarios.

Kubisch et al. [17] describe a system that applies a set of "Smart Visibility" techniques to help in planning a

surgery. Their method uses explicit cut volumes to apply the cutaway technique on a specific organ. Based on the bounding sphere of the object of interest, a cone is generated in clip space to serve as the cut volume. The cone is rendered and the depth and illumination information is stored in an RGBA buffer. This buffer is then used as the cut surface to clip the model appropriately.

Focusing on interactive environments, Burns and Finkelstein [18] created an object-driven and view-dependent cutaway rendering for architectural models. They extended the approach by Viola et al. [9] to achieve interactive frame rates. They replace the CPU based distance transform [10] with a GPU based technique [19]. We use a similar approach in our work, but instead of using the depth footprint of the rear hull of the object of interest, we create one proxy geometry for each cell. This enables more control over the cutaway shape and size.

Sigg et al. [3] propose a fully automatic method for placing the cut volume and interactively specifying important features in a model. To achieve comprehensibility of the cutaway they suggest the use of simple cut geometries such as cuboids, spheres, or cylinders. We have also followed this direction by using pyramidal shapes for our cut volumes, i.e., frustums.

Even though we were able to gather a few guidelines from these previous methods, none alone is suitable for corner-point models. Most methods work with surface models, where there exists a clear distinction between different objects. In our case, our grid has no obvious separator since our set of primary cells is dynamic, i.e., depends on the value range of selected attributes. Furthermore, even though each cell has to be handled individually, we need to, at the same time, present the global context.

2.3. Cutaway for geological models

In the literature, there are very limited examples of illustrative techniques applied to geological models, and even fewer that use cutaways.

In Ropinski et al. [20], a method is presented to provide interactive exploration of seismic datasets using volume rendering based on two specialized transfers functions. One is used to render the volume of the region of interest defined by the *lens volume* (box or sphere), and the other to render the part external to the cut geometry. They also extend a set of immersive techniques to apply their approach to virtual reality environments.

Patel et al. [21] extended the previous method by using a similar approach in combination with a 2*D* transfer function texture to produce illustrative renderings of interpreted seismic volume data. They took inspiration from geology illustration books. Both methods focus on visualizing important geological features for seismic datasets (e.g., faults and horizons).

The work proposed by Lidal et al. [2] and Martins et al. [22] are the most related to our domain of study. In a similar investigation as Li et al. [16], but in a specific context, Lidal et al. [2], comprised a study with domain experts and geological illustrators to come up with a series of design principles to effectively apply cutaways to geological models. Their approach aims at solving the problem of depth and shape perception to emphasize features inside the geological models. Unlike Kubisch et al. [17] and Burns et al. [23], their cut geometry is decoupled from the camera, and is defined in model space using a two-pass rendering strategy. The first pass extracts the bounding rectangle from the depth footprint of the objects in focus. This rectangle serves as the back plane of the proxy geometry generated in a second pass. We take a similar path, but we are able to accomplish the whole process in one render pass. Another notable difference from our method, is that their input is much simpler since they work with a set of surfaces representing geological layers. Corner-points, on the other hand, impose further challenges to visually separate the primary and secondary objects.

The method proposed by Martins et al. [22] combines cutaway and transparency to resolve occlusion problems. Their cutaway approach is a simple procedure that casts rays from the center of the camera to the centroid of the primary cells. Secondary cells that are intersected by these rays are eliminated. Although they resolve the occlusion problem, the discrete elimination criterion binarily eliminates secondary cells. In our case, we clip secondary cells in a continuous and tighter manner, saving valuable context information in the neighborhood of the cells in focus.

3. Corner-point grids

Oil reservoirs consist of layers of porous and permeable rocks, normally found deep below the earth's surface. The rock must have a special formation to prevent the hydrocarbons from migrating upwards. This formation, together with specific organic compounds, create the geological structure capable of storing potential hydrocarbons [24].

The main problem that reservoir engineers try to solve is to predict where the fluids are in the reservoir and how they flow through the pores during the productive life of the field. This problem is handled using the knowledge resulting from analysis of indirect measurements, such as well logs, seismic and outcrops. However, gathering these data is expensive and time consuming. Furthermore, they provide only a sparse representation of the entire subsurface environment.

In order to have a general approximation of the reservoir, all information coming from these indirect tools, in combination with previous knowledge, are used by domain experts to create a 3*D* geological model. Due to the highly heterogeneous nature of porous rock formations, geomodels tend to have highly irregular geometries and very complex hydraulic connectivities. The corner-point grid is the representation chosen by the industry that both handles the complexity of geological structures, and are adapted for fluid flow simulations.

Corner-point models consist of irregular hexahedral cells arranged in a 3D Cartesian grid (ideal for fluid simulations), as illustrated in Fig. 3. Degenerated cells can be produced during the modeling phase, and some may also completely disappear, introducing connections between cells that were not initially neighbors. These

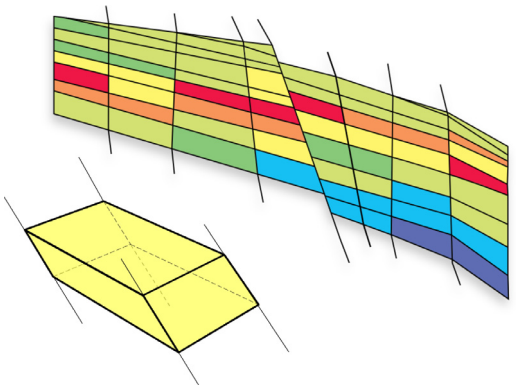


Fig. 3. A corner-point grid in an adaptation of a rectilinear grid so that it conforms to important reservoir boundaries.

particularities of corner-point grids allow for easily introducing discontinuities across faces, and therefore, the inclusion of fractures and faults [25].

Once the 3*D* model is constructed, the next step is to assign rock and fluid properties to the cells, and through visual inspection domain experts can spot inconsistencies before running the simulations. After this initial step, a process called history matching is triggered, that interactively adjusts the model by comparing the simulations to the observed data. However, this process is time consuming and highly dependent on visually inspecting the models to gain further insights. Once the model has converged, the domain experts have a more precise description of the reservoir that is highly useful in management decisions during the life time of the field [26]. As can be noted from this general description, visualization techniques are of utmost importance in all steps, since they can provide crucial clues to the reservoir engineers.

When a reservoir model is analyzed it is important to correlate cells within a property range with their spatial location. A typical scenario is when performing well analysis. In corner-point grids wells are represented as a chain of cells. The petroleum engineers do not only need to locate the cells that represent the well, but are also interested in their surrounding. Another example is when performing a procedure known as water injection, they need to find a path of cells that gives the higher mobility, or connectivity, and track this front without losing reference to the rest of the model. The front may assume different forms depending on the selected attribute: temperature, saturation, pressure, etc.

These cells can form a single cluster or be spread apart, in any case the expert must be able to visually identify them and the correlation with the neighbors. We define these cells as the primary object that has to be in focus, and the remaining cells as the secondary object that composes the context. It is important to observe an interesting issue that rises from the fact that both objects have the same volumetric nature and are located in the same space: the difference between them is usually very subtle. Consequently, this imposes a strong limitation to directly applying previous computer graphics techniques to corner-point models.

4. Cutaway implementation

As explained in the previous section, our target is corner-point grids, where each element is defined by a cuboid with possible irregular shape. The objects of interest are cells and the primary/secondary discrimination is dependent on the range of chosen values for a specific attribute. First an attribute is selected, such as pressure or porosity. Note that different models may have different attributes. The user then specifies a minimum and maximum value for the selected attribute, all cells inside this range will compose our primary set.

To help render the cutaways, we classify cells' faces into two groups: faces that are internal to the model (interior faces); and the faces on the surface's boundary (shell faces). The main reason is that shell faces do not need to be reconstructed, we simply either render the fragments that are behind the cutaway surface, or clip them otherwise. On the other hand, the non clipped fragments from interior faces, have their normals and depth value modified. Moreover, with this discrimination, we can also apply important feature enhancements, such as contour lines.

To reduce ambiguities during visualization, we shade the wireframe lines of the primary and secondary cells with different colors. On the other hand, since the attributes color code is an important inspection feature, we try to respect it as much as possible when shading the faces.

As our focus object is a set of small irregular cells, we construct our cut volumes in model space, similar to Li et al. [16], Lidal et al. [2], and Sigg et al. [3]. However, we build a cut geometry for each primary cell, and unify them based on the depth footprint of their rasterization. The main reason is that we have no prior knowledge of the location of primary cells, e.g., they might be tightly packed or highly dispersed inside the model.

In the following subsections we describe in details our method to generate cutaway renderings of 3*D* reservoir models. Our approach is performed in three main stages:

1. First, we generate the cut surface and represent it as a depth image: Section 4.1.
2. Second, we render the secondary cells and continuously clip them against the cut surface, rendering the remaining ones: Sections 4.2, 4.3 and 4.4.
3. Finally, we render the primary cells in a very straightforward manner.

4.1. Cutaway surface

To achieve interactive frame rates, we employ a modification on the rendering pipeline to create the cut volumes and the unified cut surface in a single render pass. A 2*D* simplification of the complete process is illustrated in Fig. 4.

We start with the bounding box of the cells in focus, and reshape them into frustums facing the camera. This is achieved by taking the minimum and maximum *x*,*y* and *z* values of the cells vertices after transforming them to camera space. We also move the back plane away from the cell to allow for a small space margin between the cutaway

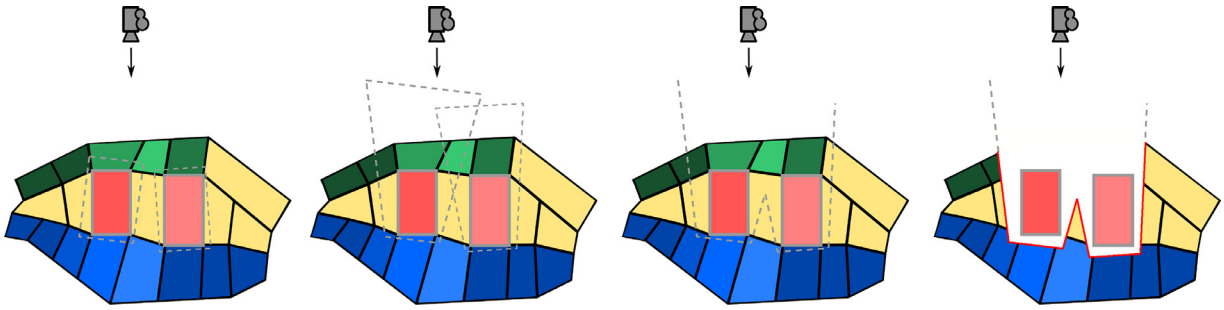


Fig. 4. Proxy Geometry Generation: 4a first the bounding box of objects of interest are computed; 4b the bounding boxes are then transformed into frustums; 4c the frustums are rendered with an inverted depth test to register their union; 4d the rest of the model (everything that is not in focus) is clipped against the depth image of the cut surface.

surface and the primary cells. Each frustum defines a cut volume for a single primary cell (see Fig. 4a and b).

For a single primary cell the depth image generation is simple. The bounding box is transformed into a frustum with a given aperture angle and rendered into a depth buffer. This buffer is our screen-space representation of the cutaway (cut surface), and is used to remove occluding geometry. This is trivially achieved using a depth test, i.e., during the rasterization of the secondary cells each generated fragment's depth is tested against the value in the depth image.

When there is a set of primary cells, however, it is possible that their cut volumes overlap, as shown in Fig. 4b. We treat the overlap by inverting the default depth test when rendering the frustum in order to keep only the rear hull of the proxy geometry. This simple modification allows for the union of any number of cut volumes, as shown in Fig. 4c. The accumulated depth buffer of all rasterized frustums completely defines the cut surface for our rendering purposes.

Since we render one frustum for each primary cell, we can produce the cutaway surface representation in one render pass without any necessary posterior image space computation. This avoids extracting the bounding rectangle information from the buffer, such as done by Lidal et al. [2], or performing image space methods (such as the Chamfer algorithm) to compute the screen-space surface, like Viola et al. [9] and Burns and Finkelstein [18].

Furthermore, the horizontal and vertical aperture angles of the frustums can be modified interactively. Since the frustums are created on-the-fly inside the shaders, this implies in no extra cost.

Following, we briefly describe this sequence of steps within a shader pipeline perspective:

Vertex Shader Extract the bounding box and reshape it into a frustum facing the camera.

Geometry Shader Receive the 8 points defining the frustum and generate its faces for rasterization.

Fragment Shader Save the rasterized frustum into an *RGBA* texture, where the *RGB* channels store the normal of the cut surface, and the alpha channel stores the depth information. We also store in a separate buffer the model space coordinates of the surface, necessary for the SSAO algorithm.

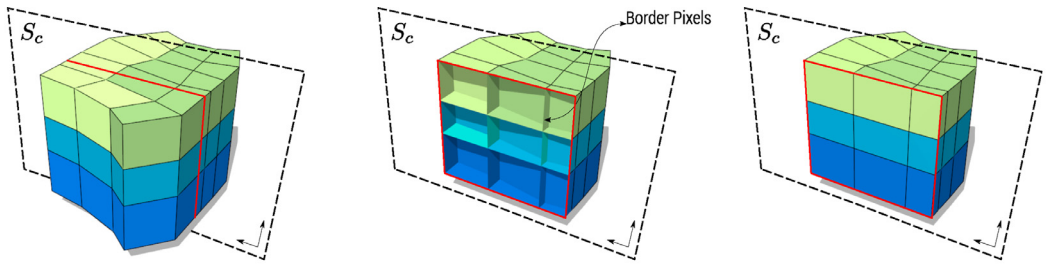
4.2. Clipping

As aforementioned, given a depth image representation of the cut surface, clipping of the secondary cells is realized in a second render pass with a straightforward depth test. By simply eliminating fragments with depth in front of the cut surface, a hollow geometry is rendered, as illustrated in Fig. 5b. However, secondary cells should be clipped in a continuous manner, i.e., a partially clipped cell should appear as a sliced solid object. To achieve this visual impression, the normals of the fragments of the interior faces are set as the normals of the cut surface for the corresponding pixel. Note that the cut surface's normals were stored during the first render pass (Section 4.1). The whole process is illustrated in Fig. 5.

4.3. Rendering lines

To render a cell's wireframe lines (its contours), we follow the approach proposed by Bærentzen et al. [27]. This technique renders the model and its wireframe lines in a single pass inside the rasterization pipeline. However, since we do not render the cut surface explicitly, the lines representing the boundaries between the cut surface and the clipped cells are not known a priori, and have to be determined during rendering. Thus, we have to detect the border pixels representing where a cell face is clipped by the cut surface.

A simple approach to detect the border pixels is to use a threshold between the depth image of the cutaway and the depth of the fragment. The idea is to identify fragments that were not clipped, but are very close to the cut surface. Unfortunately, although some lines are rendered correctly, others are rendered with erroneous thickness (Fig. 7a), mainly due to discretization issues and to the fact that the depth test has limited precision [28]. To avoid this undesirable effect a more precise test is carried out by checking the neighbors of the fragments. If one or more adjacent fragments were clipped, it means that this is a border pixel. Since in the fragment shader we do not have access to other fragments being processed, we employ a simple



(a) Secondary cells intersected by the cut surface S_c.

(b) Clipping fragments in front of S_c

(c) Reconstructing the cells and its respective lines.

Fig. 5. Clipping and filling secondary cells.

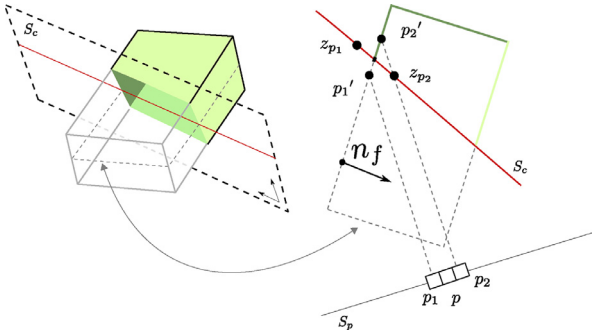
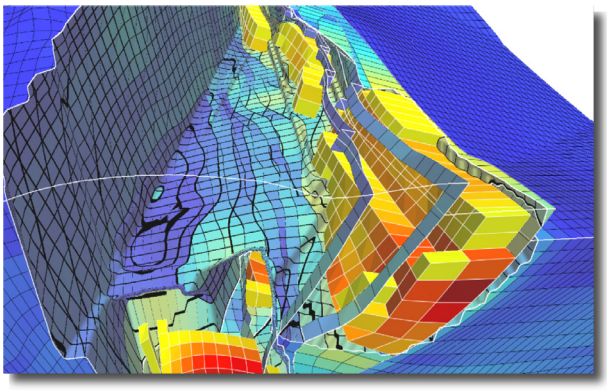
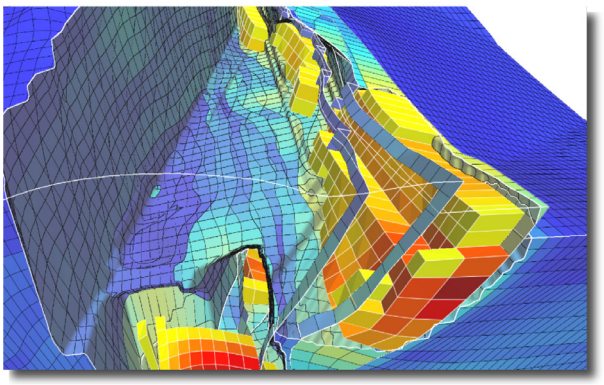


Fig. 6. Ray Casting border detection: the pixel p tests its neighbor's by reprojecting them onto its generating face's plane (indicated by normal n_f). We define it as a border pixel if at least one neighbor's reprojected depth p_i' falls in front of the cut surface S_c (line in red), that is, it is near to the projection plane S_p. In this case the reprojected point p_1' on the plane is in front of the stored depth value z_{p_1} of the cutaway surface, thus p is a border pixel. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).



(a) Threshold based line rendering. Note how some lines are rendered with different thickness.



(b) Raycasting based line rendering. Line thickness is kept constant along all cuts.

Fig. 7. A corner-point model with some cells selected as primaries, and using a narrow cutaway angle. Fig. 7a illustrates the internal lines rendered using a threshold between the cut surface and the depth of the fragment. In Fig. 7b internal lines are rendered using our ray casting procedure. Note how the thickness of the black lines varies arbitrarily using a threshold approach, and is stable using the ray-casting test.

ray casting algorithm to check if the geometry projected on the neighboring fragment would be clipped against the cut surface. Our only assumption is that the neighbors belong to the same cell face, i.e., are coplanar in model space. A four pixel neighborhood is used: top, down, right and left. To test a neighbor, we start by reading the depth value z_{p_i} from the cutaway buffer (Fig. 6). To predict the neighbor's depth, we reproject the neighboring pixel p_i onto the plane defined by the current pixel position in world coordinates and the normal n_f of the face that generated it. The distance $d(p_i, p_i')$ is tested against the value z_{p_i} to determine if the neighbor pixel would be clipped or not. If at least one neighbor is clipped, the current pixel is marked as a border pixel and is shaded in a different manner to emphasize the cut. The size of the neighborhood determines the thickness of the lines, for example, if we test neighbors farther away from the pixel, thicker lines are produced.

4.4. Extra lines and features

Lines are an important feature style in many illustrative techniques [29]. They offer a minimal visual representation of a scene with little visual clutter [30]. Contours, for example, are feature lines that convey the structure of the object. We extract these contours from the shell faces in a preprocessing stage, and render them in a last render pass. Another important feature line is the one that defines the intersection of the shell faces with the cut surface, emphasized in red in Fig. 5b. These are also rendered in this final pass. The overhead of these feature lines is minimal and

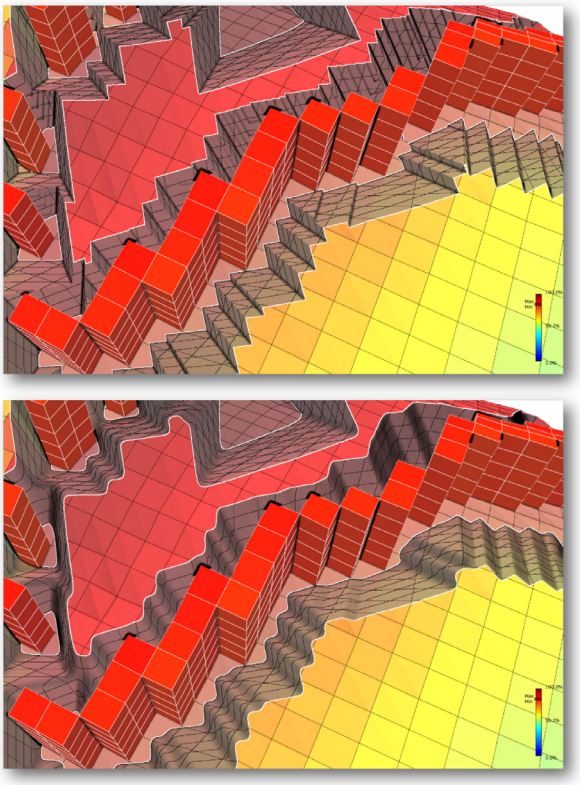


Fig. 8. Application of a mean filter to smooth the union of the cut surfaces in image space. Top: no filter. Bottom: with mean filter.

they provide useful clues about the global structure of the model.

Another visual improvement is the smoothing of the cutaway depth image. When many frustums are generated, a staircase effect is produced after unifying them in screen-space. To smooth the transition between different frustums a mean filter is applied directly on the depth image. The filter radius is dynamic, that is, it varies with the distance to the camera to avoid changing the effect while zooming. We sample the kernel to avoid a significant overhead when the kernel increases, that is, we do not use all pixels inside the kernel, but just a subset to avoid too many texel fetches. Fig. 8 illustrates the effect of applying the filter. Note that we are smoothing the cutaway surface and not the actual cells, so the rest of the pipeline is exactly the same. However, it is important to comment that over smoothing may cause the cutaway surface to slightly cover the primary cells at some corners. This is not a great issue since the main context and idea of the primary set is still preserved, and it is possible to change the smoothing parameter dynamically while inspecting the model. (Fig. 9).

We also employ the concept of a freeze-view mode suggested by Lidal et al. [2], which takes advantage of motion parallax to improve depth cue [31], where one can rotate the model without changing the frustums. In other words, the transformation matrices for the frustums are kept still, while the model continues to rotate and translate. This allows for an improved depth perception of the cut surface. This effect is illustrated in Fig. 10.

To improve spatial perception, a Screen-Space Ambient Occlusion (SSAO) shader is applied on the final image based on the method from Shanmugam and Arikan [32]. The only required modification is that we have to store the cutaway surface's coordinates as well as the normals in a buffer.

Finally, to enhance the contrast between primaries and secondaries, we slightly increase the saturation channel of the first group while decreasing it for the second group.

5. Results

We tested our method on a system with an Intel *i*5 Quad Core processor, 8GB of RAM and an nVidia GeForce 660 GTX graphics card, 2GB of VRAM. The application is written in C++ using OpenGL 4. The datasets employed in this work are real reservoir models provided by an industry partner. The tests were performed with a screen resolution of 1920×1080 for all the stages of the algorithm. All results were rendered with a mean filter kernel with size 9×9 and an internal line thickness of 1 pixel.

Table 1 shows rendering times for reservoir models of various sizes, and Figs. 14 and 15 depict renders of these models. The tests were realized by selecting some primaries cells and rotating the model in different orientations, while, at the same time, changing the cut surface aperture. The reported times are averaged frame rates. We noted that the frame rate slightly drops when using a wide angle aperture for the cut surface. Although in this scenario we are eliminating more secondary cells, and consequently sending fewer cells down the rendering

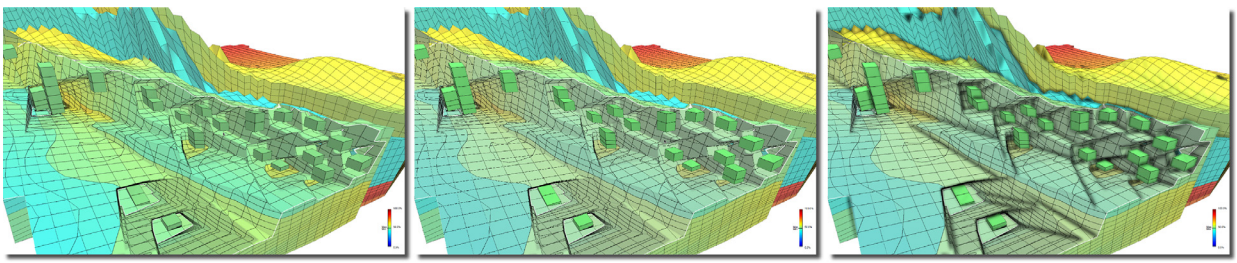
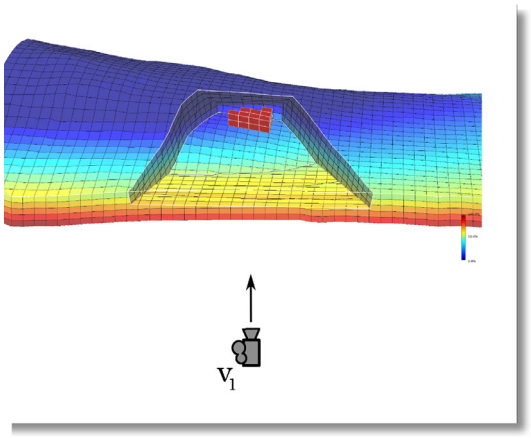
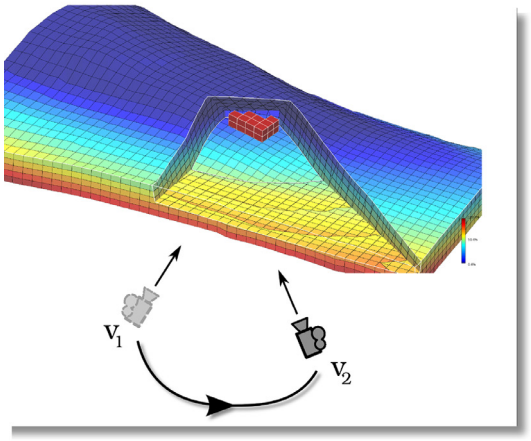


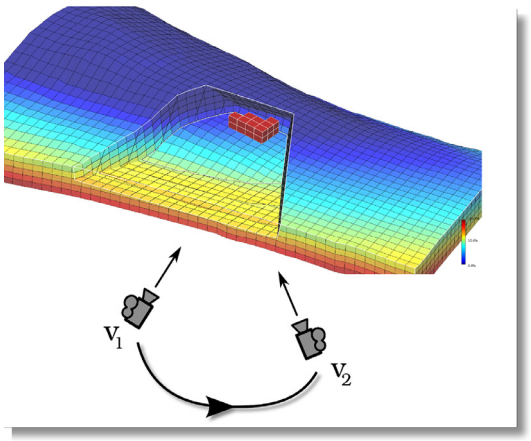
Fig. 9. From left to right: the result of the cutaway; increasing/decreasing the saturation channel of the primaries/secondaries; and applying SSAO. Note how these two techniques help in distinguishing primaries from secondaries. Since the selected value range is very tight, the minimum and maximum values on the color bar are practically overlapping for this example.



(a) Original View



(b) View Dependent



(c) Freeze View

Fig. 10. Illustration of the freeze view feature for improved depth perception, as proposed by Lidal et al. [2]. In Fig. 10a the start view position v_1 from the camera. In Fig. 10b we have the scene rendered with view dependent cutaway generation. Fig. 10c shows the freeze view in action, where the original camera view v_1 is used to generate the cutaway rendering and a second camera view v_2 to visualize the scene. With freeze view it is possible to gain more insight about the spatial location of the features in focus.

Table 1
Rendering time for four reservoir models using our cutaway method. All times are expressed in milliseconds. The models were rendered using a 1920 × 1080 screen resolution and a mean filter with kernel size of 9 × 9. In the experiment, we have chosen around 1% of the cells as primaries.

	Zaphirus	Petra	Opharine	Zapphirus
Cells	7500	32760	76000	208320
First stage	1.13	1.9	2.0	2.35
Second stage	6.5	11.1	26	39.7
Third stage	0.15	0.38	0.7	1.78
SSAO	4.8	5.2	4.6	5.33
Mean filter	3.0	4.6	4.7	4.6
Total	15.8	23.2	38	53.9

pipeline, we are also defining larger cutaway walls and thus exposing more context. In other words, more time is dedicated to the ray casting algorithm to reconstruct the internal lines and create the internal faces for the sliced cells.

As described in Section 4, our method is performed in three stages. One to select the primary cells and generate the depth image from their bounding boxes. The second clips the secondary cells and reconstructs the internal walls. Finally, the last one emphasizes some features of the model and renders the primary cells.

The first stage computes and orients a bounding box for each primary cell in the vertex shader using only transformation matrices and the geometry of a cell. With this information the shader calculates the points of the bounding box and normals of each face. Moreover, the bounding boxes are assembled inside the geometry shader, and require a considerable amount of primitives to be produced. In our experiment, we use around 1% of the total number of cells as primaries. As we are concerned with cell inspection, the method targets small amounts of primaries, since a selected set too large would practically eliminate the secondaries, i.e., the context.

In the second stage we clip the secondary cells against the cut surface, fill the cells, and reconstruct the lines. Filling the cells is trivial, we just need to change the information of each fragment generated from an internal face. The expensive part is to reconstruct the lines. For a border line of minimum width (i.e., one), we have to access the depth buffer four times per fragment. When we make the surface angle wider, more fragments are discarded, and consequently more of the cutaway wall is exposed, implying in more texture accesses (ray casting tests).

The third stage imposes a very small overhead. This stage renders some contour lines and the primary cells. Even though, it would be possible to render the primaries during the second pass, we decided to render them in a separate stage, to allow for more freedom when applying some effects to emphasize the features in focus. For example, it is possible to change the color of the cells and lines since it might be an important contrast enhancement.

6. Design critique

Gathering feedback from experts who have experience analyzing corner-point models is essential to fine tune the

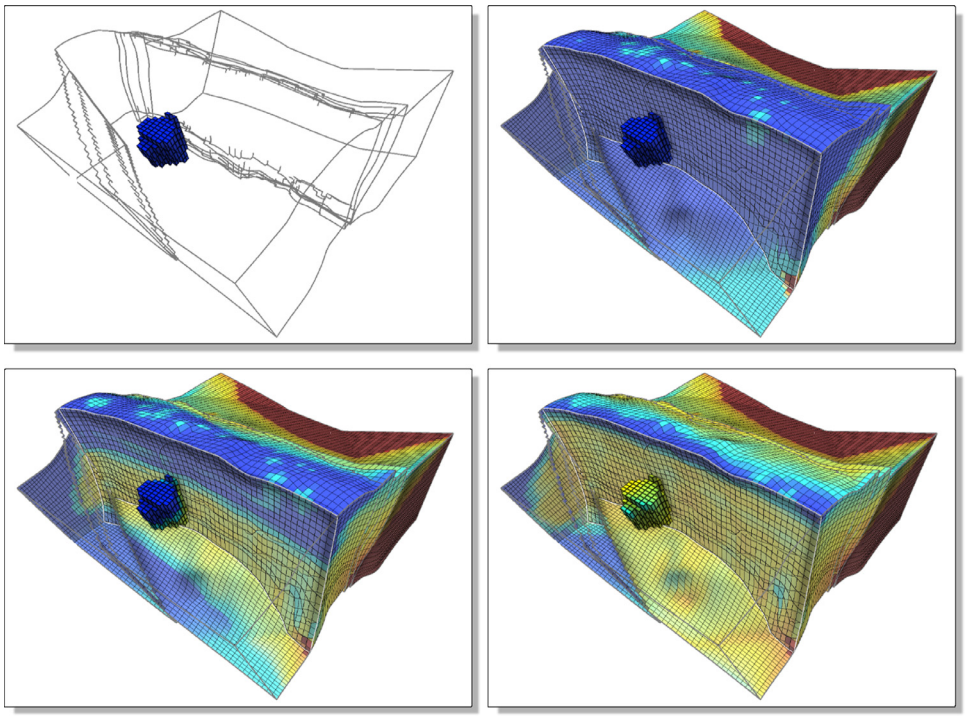


Fig. 11. A wide angle cut in three different moments of a time varying dataset (property values change over time). The cutaway is the union of individual planar surfaces, and removes everything behind the primary set given a specific direction.

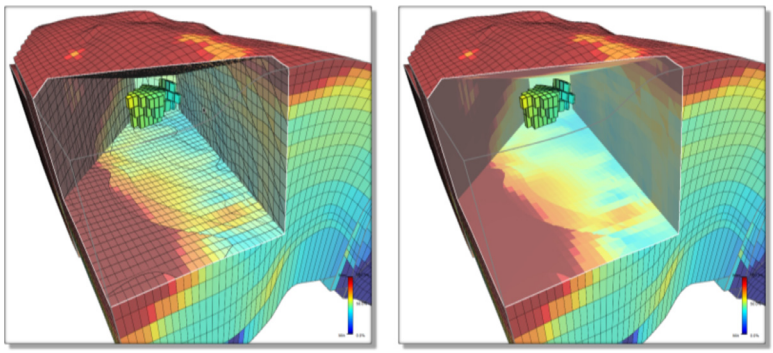


Fig. 12. Internal lines of the cutaway region can be hidden when convenient.

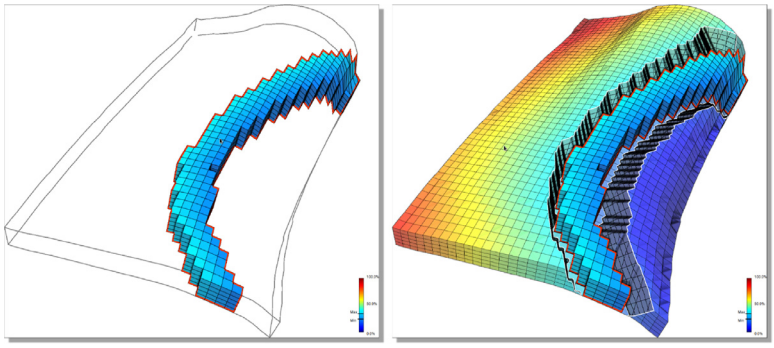


Fig. 13. A silhouette around the primary cluster can be drawn with a simple render pass, where every pixel that is not primary, but has a primary neighbor, is drawn in red. The silhouette width can also be adjusted by consulting a wider neighborhood. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

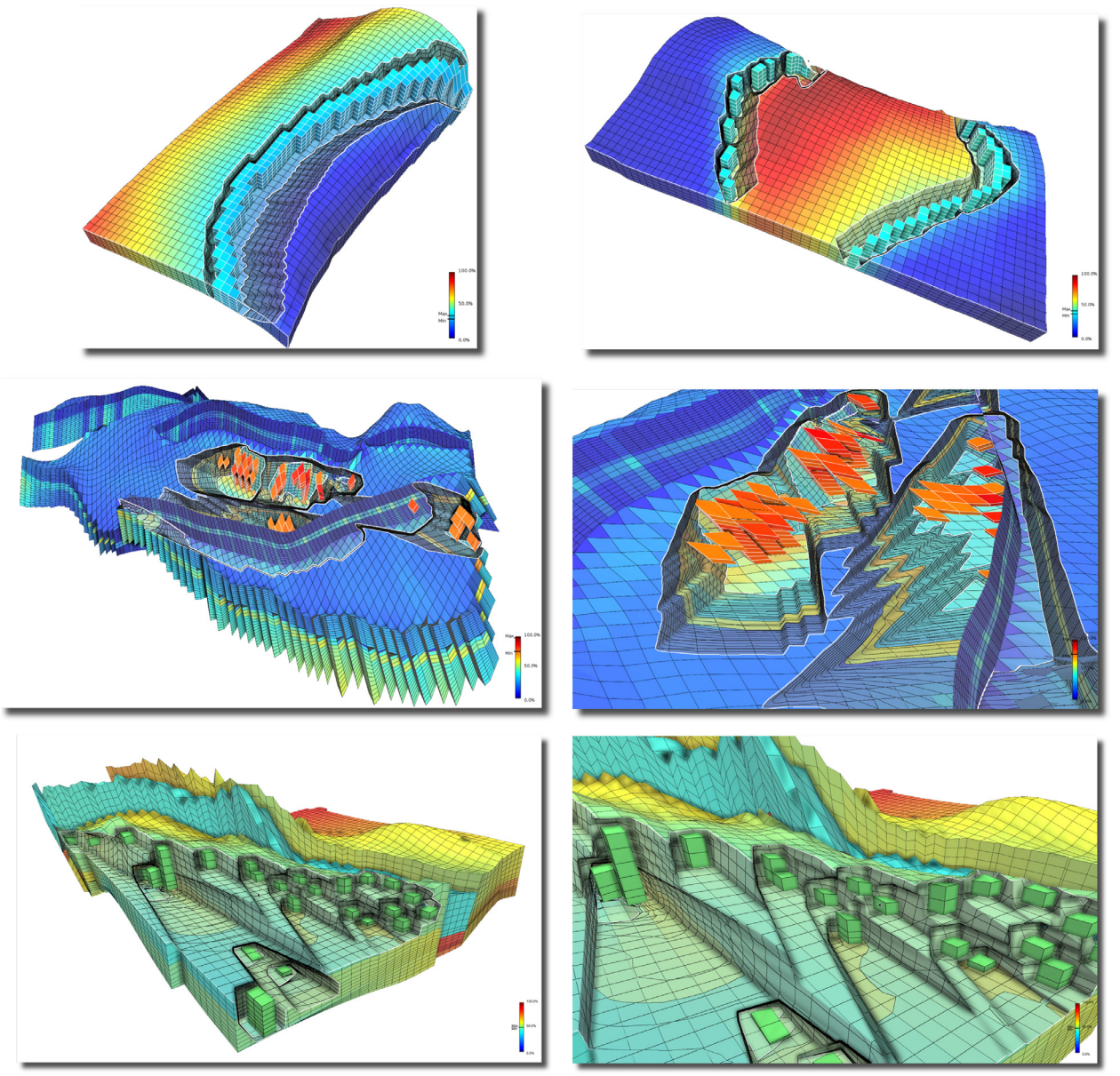


Fig. 14. Reservoir models rendered using our cutaway technique. Top: ZMap model with two different properties, Modified Block Volume and Porosity. Middle: Two views of the Zaphirus model with the static property porosity. Bottom: Two views of the Sapphire model with the static property modified block volume.

visualization technique to achieve the expressiveness required by real industrial tasks.

We hosted design critique sessions, with two expert practitioners involved in upstream research in the oil and gas industry. One domain expert has more than 30 years of industry experience including research and development of reservoir simulation gridding and visualization techniques and software systems; the other domain expert has 10 years of experience in fluid flow simulation in various gridding methods and models, including corner-point. Both domain experts have PhD degrees in petroleum engineering.

We have led one independent session with each expert, letting them test the prototype for about 30 min. The two experts stated that the visualization technique allows easily locating the primary cells in the reservoir, while keeping the context of the surrounding cells. They also commented that to the best of their knowledge, no current system for the oil-and-gas industry is able to achieve similar results. Both highlighted that the freeze view, silhouettes curves, and smoothing edges features improve the reservoir's spatial perception. They suggested this technique could have a great impact for front track visualization of fluid simulation and inspection of wells inside the reservoirs.

For front track visualization, the method allows following a set of cells during a sequence of time steps with-

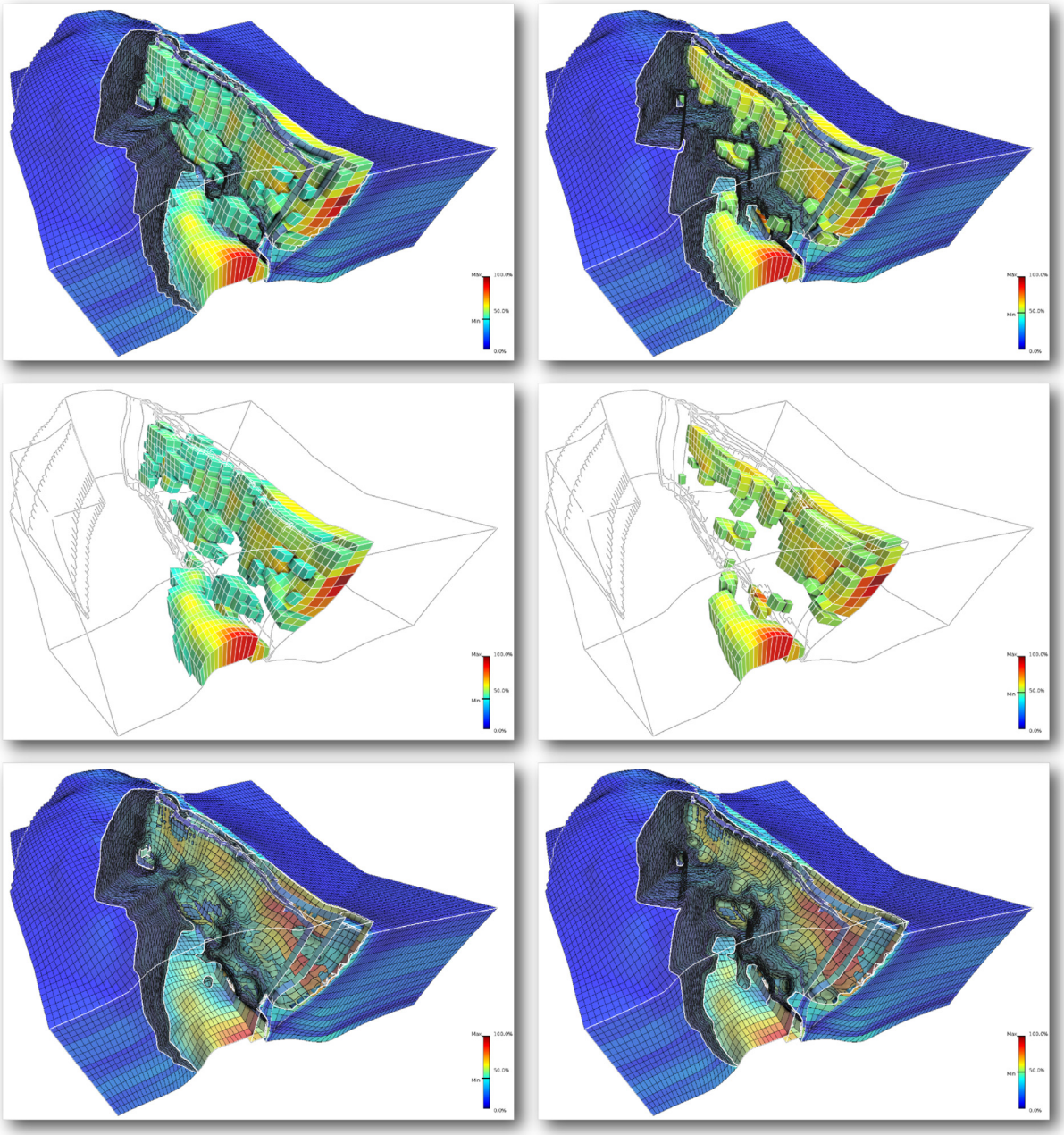


Fig. 15. Opharine model rendered with the static property modified block volume, and two different selection ranges. The middle row shows only the primary cells, while the bottom row shows only the secondaries.

out losing reference to the surrounding cells. This is particularly difficult to achieve with more direct methods (such as planar cuts). Since the front is dynamic, it can change shape quite drastically in a short period of (geological) time. The inspection of the fluid flow behavior inside the model is a typical and important task of petroleum engineers.

In addition, they pointed out as a limitation the difficulty to differentiate between primary and secondary cells when there is a large number of primaries. To tackle this issue they suggested the following as options that could be enabled/disabled within the system: (1) remove all cells that are between the primary cells and the camera, (2) hide/draw the edges of the secondary blocks, and (3) draw the silhouette of the primary cluster in different color and line style.

All these requests are easily achievable using our method without introducing any technical challenge. We briefly describe the necessary modifications with illustrative examples. Suggestions (1) only requires setting the frustum aperture angle to 180° (Fig. 11). Suggestion (2) is achieved by not drawing the internal lines, as illustrated in

Fig. 12. Finally, we have implemented a last screen-space pass to illustrate how Suggestion (3) could be achieved (Fig. 13).

Although this was still an informal study, it shed light on the potential of the proposed technique to visualize a set of corner-point cells while providing a good understanding of the surrounding reservoir geometry. The study also indicates good application and improvement opportunities.

7. Limitations

A main limitation of our method is that our input model has to be pre-processed in order to provide a discrimination of interior and exterior cells. Note that our model in study does not contain geometrical information of geological features, such as faults and horizons. For some models we successfully retrieved this information during pre-processing, but for other cases with artifacts and lots of degenerated cells they were not fully identified.

Another important point is that due to the complex nature of reservoir models they are many times hard to visualize even by domain experts. In some cases the fragmented primary groups or highly distorted cells can cause the static images to be confusing even if we are respecting all the cutaway premises. This issue is indeed alleviated by the smoothing and shading filters, but depth perception is truly enhanced when the model and the frustum parameters are interactively manipulated.

8. Conclusions and future works

In this work we have presented an interactive cutaway visualization method to aid in the inspection of Oil&Gas reservoir models represented by corner-point grids. The method runs in interactive frame rates and does not depend on the complexity or topology of the model. Our solution is based on a depth representation of the cut surface to perform the clipping process. All cuts and lines are realized in image space, making the method strongly bounded by the screen resolution and not the model's size. In fact, most of the work bounded by the model's resolution regards the primitives' projection only.

The proposed method opens many opportunities for possible applications. Our cutaway technique can be extended to dynamic scenarios associated with these reservoirs. However, tracking dynamic properties over time may result in visual confusion as cells may arbitrary pop in and out of the primary set. Thus, further studies on how to provide a smooth time-coherent cutaway visualization are necessary.

Following the works of Li et al. [16], we would like to design an authoring tool that allows domain experts to create their own cutaway volumes. Since our method does not require any special geometry for these volumes, only a depth image representation, new cut geometries can be trivially added to the pipeline.

Finally, our method applies only a few of the possible illustrative effects. Most works in the literature are tailored for models with well defined topology and geometry, i.e., clear structure separation such as organs in anatomical models. Since reservoir models have volumetric characteristic and are prone to degenerated and complex topologies, a more detailed study has to be carried out to better adapt and introduce new features.

Acknowledgments

We would like to thank our colleagues for their useful discussions and advice. We also thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported in part by the Brazilian funding agency Carlos Chagas Filho Research Support Foundation (grant: E-26/100.393/2012)(FAPERJ). We also acknowledge the NSERC/AITF/Foundation CMG Industry Research Chair program in Scalable Reservoir Visualization at the Department of Computer Science, University of Calgary.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.gmod.2016.02.001.

References

[1] F. Evans, W. Volz, G. Dorn, B. Fröhlich, D.M. Roberts, Future trends in oil and gas visualization, in: Proceedings of the Conference on Visualization '02, in: VIS'02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 567–570. http://dl.acm.org/citation.cfm?id=602099.602200.

[2] E.M. Lidal, H. Hauser, I. Viola, Design principles for cutaway visualization of geological models, in: Proceedings of the 28th Spring Conference on Computer Graphics, in: SCCG '12, ACM, New York, NY, USA, 2013, pp. 47–54, doi:10.1145/2448531.2448537.

[3] S. Sigg, R. Fuchs, R. Carnecky, R. Peikert, Intelligent cutaway illustrations, 2012 IEEE Pacific Visualization Symposium, 2012, pp. 185–192, doi:10.1109/PacificVis.2012.6183590.

[4] S. Bruckner, S. Grimm, A. Kanitsar, M.E. Gröller, Illustrative context-preserving volume rendering, in: Proceedings of EuroVis 2005, 2005, pp. 69–76.

[5] S. Bruckner, M.E. Gröller, Volumeshop: An interactive system for direct volume illustration, in: H.R. C. T. Silva (Ed.), Proceedings of IEEE Visualization 2005, 2005, pp. 671–678.

[6] L. Wang, Y. Zhao, K. Mueller, A. Kaufman, The magic volume lens: an interactive focus+context technique for volume rendering, in: Visualization, 2005. VIS 05. IEEE, 2005, pp. 367–374, doi:10.1109/VISUAL.2005.1532818.

[7] E.A. Bier, M.C. Stone, K. Pier, W. Buxton, T.D. DeRose, Toolglass and magic lenses: The see-through interface, in: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, in: SIGGRAPH '93, ACM, New York, NY, USA, 1993, pp. 73–80, doi:10.1145/166117.166126.

[8] J. Kruger, J. Schneider, R. Westermann, Clearview: an interactive context preserving hotspot visualization technique, Vis. Comput. Graph. IEEE Trans. 12 (5) (2006) 941–948, doi:10.1109/TVCG.2006.124.

[9] I. Viola, A. Kanitsar, M. Groller, Importance-driven volume rendering, in: IEEE Visualization 2004, IEEE Computer Society, 2004, pp. 139–145, doi:10.1109/VISUAL.2004.48. http://dl.acm.org/citation.cfm?id=1032664.1034441.

[10] G. Borgefors, Distance transformations in digital images, Comput. Vis. Graph. Image Process. 34 (3) (1986) 344–371, doi:10.1016/S0734-189X(86)80047-0.

[11] D.D. Seligmann, S. Feiner, Automated generation of intent-based 3d illustrations, SIGGRAPH Comput. Graph. 25 (4) (1991) 123–132, doi:10.1145/127719.122732.

[12] S.K. Feiner, D.D. Seligmann, Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations, Vis. Comput. 8 (5-6) (1992) 292–302, doi:10.1007/BF01897116.

[13] J. Viega, M.J. Conway, G. Williams, R. Pausch, 3D magic lenses, in: Proceedings of the 9th annual ACM Symposium on User Interface Software and Technology - UIST '96, ACM Press, New York, New York, USA, 1996, pp. 51–58, doi:10.1145/237091.237098.

[14] J. Diepstraten, D. Weiskopf, T. Ertl, Interactive cutaway illustrations, Comput. Graph. Forum 22 (3) (2003) 523–553, doi:10.1111/1467-8659.t01-3-00700.
[15] C. Coffin, T. Hollerer, Interactive perspective cut-away views for general 3d scenes, in: Proceedings of the IEEE Conference on Virtual Reality, in: VR '06, IEEE Computer Society, Washington, DC, USA, 2006, p. 118, doi:10.1109/VR.2006.88.
[16] W. Li, L. Ritter, M. Agrawala, B. Curless, D. Salesin, Interactive cutaway illustrations of complex 3d models, ACM Trans. Graph. 26 (3) (2007) 31. http://doi.acm.org/10.1145/1276377.1276416.
[17] C. Kubisch, C. Tietjen, B. Preim, Gpu-based smart visibility techniques for tumor surgery planning, Int. J. Comput. Assist. Radiol. Surg. 5 (6) (2010) 667–678, doi:10.1007/s11548-010-0420-0.
[18] M. Burns, A. Finkelstein, Adaptive cutaways for comprehensible rendering of polygonal scenes, in: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers, ACM, New York, NY, USA, 2008, pp. 1–7. http://doi.acm.org/10.1145/1457515.1409107.
[19] G. Rong, Jump flooding in GPU with applications to Voronoi diagram and distance transform, Proceedings of the 2006 Symposium on Interactive, 2006, p. 109, doi:10.1145/1111411.1111431.
[20] T. Ropinski, F. Steinicke, K.H. Hinrichs, Visual exploration of seismic volume datasets, J. Proc. 14th Int. Conf. Cent. Eur. Comput. Graph. Vis. Comput. Vis. (WSCG06) 14 (2006) 73–80. http://viscg.uni-muenster.de/publications/2006/RSH06a.
[21] D. Patel, C. Giertsen, J. Thurmond, M.E. Gröller, Illustrative rendering of seismic data, in: H.S. Hendrik. Lensch (Ed.), Proceeding of Vision Modeling and Visualization 2007, 2007, pp. 13–22.
[22] Z. Martins, F. de Carvalho, E. Vital Brazil, R. Marroquim, M. Sousa, Cutaway applied to corner point models, in: SIBGRAPI 2012 - WGARI (Workshop on Industry Applications), 2012.
[23] M. Burns, M. Haidacher, W. Wein, Feature emphasis and contextual cutaways for multimodal medical visualization, ...on Visualization (2007) 275–282, doi:10.2312/VisSym/EuroVis07/275-282. http://dl.acm.org/citation.cfm?id=2384179.2384223 http://dl.acm.org/citation.cfm?id=2384223.
[24] J.S. Gomes, F.B. Alves, The Universe of the Oil and Gas Industry: From Exploration to Refining, 1st, Partex Oil and Gas, 2013. ISBN 9892037782.
[25] J. Aarnes, S. Krogstad, K.-A. Lie, Multiscale mixed/mimetic methods on corner-point grids, Comput. Geosci. 12 (3) (2008) 297–315, doi:10.1007/s10596-007-9072-8.
[26] G. Adamson, M. Crick, B. Gane, O. Gurpinar, J. Hardiman, D. Ponting, Simulation Throughout the Life of a Reservoir (Oilfield Review), 1996.
[27] A. Bærentzen, S.L. Nielsen, M. Gjøl, B.D. Larsen, N.J. Christensen, Single-pass wireframe rendering, in: ACM SIGGRAPH 2006 Sketches, in: SIGGRAPH '06, ACM, New York, NY, USA, 2006, doi:10.1145/1179849.1180035.
[28] D. Shreiner, G. Sellers, J.M. Kessenich, B.M. Licea-Kane, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3, 8th, Addison-Wesley Professional, 2013. ISBN 0321773039. 9780321773036.
[29] M.C. Sousa, P. Prusinkiewicz, A few good lines: suggestive drawing of 3d models, Comput. Graph. Forum (Proc. of EuroGraphics) 22 (2003) 2003.
[30] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, M. Singh, How well do line drawings depict shape? in: ACM SIGGRAPH 2009 Papers, in: SIGGRAPH '09, ACM, New York, NY, USA, 2009, pp. 1–9, doi:10.1145/1576246.1531334.
[31] S. Ohtsuka, S. Saida, Depth perception from motion parallax in the peripheral vision, in: Robot and Human Communication, 1994. RO-MAN '94 Nagoya, Proceedings., 3rd IEEE International Workshop on, 1994, pp. 72–77, doi:10.1109/ROMAN.1994.365953.
[32] P. Shanmugam, O. Arikan, Hardware accelerated ambient occlusion techniques on gpus, in: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, in: I3D '07, ACM, New York, NY, USA, 2007, pp. 73–80, doi:10.1145/1230100.1230113.