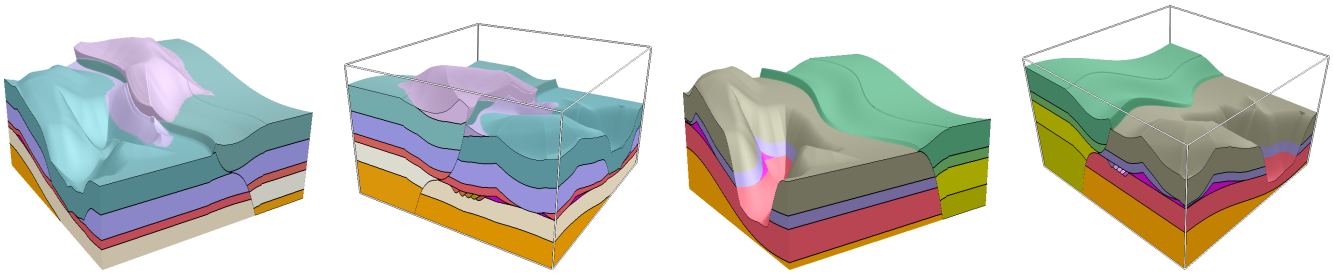
Rapid Modelling of Interactive Geological Illustrations with Faults and Compaction

Mattia Natali*
University of Bergen

Julius Parulek†
University of Bergen

Daniel Patel‡
Christian Michelsen Research



(a1) (a2) (b1) (b2)

Figure 1: Illustrative geological models showing non-planar faults and compaction of layers: (a1)-(a2) and (b1)-(b2) are different views of the same model.

Abstract

In this paper, we propose new methods for building geological illustrations and animations. We focus on allowing geologists to create their subsurface models by means of sketches, to quickly communicate concepts and ideas rather than detailed information. The result of our sketch-based modelling approach is a layer-cake volume representing geological phenomena, where each layer is rock material which has accumulated due to a user-defined depositional event. Internal geological structures can be inspected by different visualization techniques that we employ. Faulting and compaction of rock layers are important processes in geology. They can be modelled and visualized with our technique. Our representation supports non-planar faults that a user may define by means of sketches. Real-time illustrative animations are achieved by our GPU accelerated approach.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.6.5 [Simulation and Modeling]: Model Development—Modeling Methodologies

Keywords: sketch-based modelling, illustrative geology, fault visualization, interactive animations

1 Introduction

For many years the focus of modelling has been on achieving an accurate representation of the real world. This has led to complex techniques that produce precise models, but demand considerable working time of specialized personnel. Rapid modelling and illustrative visualization [Viola et al. 2006] help reduce such efforts. Rapid modelling can be carried out either in form of automated procedures or through sketches that are interpreted by specialized algorithms. Sketch-based modelling is a way of making a new model or modifying an existing one, with the help of human drawing ability. Sketching is more intuitive for a designer than setting parameters. Sketch interpretation is closely tied to a particular modelling scenario, but it is basically an automatic way of filling in parameters. A sketch-based approach has advantages compared to procedural modelling in terms of expressibility: with the former, a user can freely design shapes and this is usually more intuitive and simpler to learn. Some techniques interpret sketches as if they were made on a sheet of paper [Masry and Lipson 2007; Karpenko and Hughes 2006], hence they attempt to bring back to 3D space what were meant to be projections of the model. Other techniques allow to directly define strokes in 3D [Bernhardt et al. 2011; Igarashi et al. 1999; Gain et al. 2009; Vital Brazil et al. 2010]. It is useful to combine sketch-based modelling with illustrative visualization, because sketches are meant to be approximations and this is best conveyed through a style that relates to qualitative rendering.

In illustrations, details are less important, while the focus is on the concept a model should convey, rather than the preciseness of representing data measurements. Illustrations are mainly utilized for communication and teaching purposes. In our work, we merge sketch-based modelling and illustrative visualization to produce geological models (such as the example in Figure 1). They are represented as layer-cake objects that show the internal stratigraphy of the subsurface. We address the modelling and interactive animation of geological aspects, such as faults and compaction of layers.

Currently, illustrators in geology spend a considerable amount of time in designing their models in 2D vector graphics software. That results in two-dimensional images that convey three-dimensional structure. Such models cannot be rotated in 3D, cut into or animated without enormous efforts. Animations are particularly important in the domain of geology to communicate variations and combinations of processes that take place in the subsurface. Our approach allows to quickly define illustrative animations of some of the most relevant geological events involved in the subsurface

*e-mail: mattia.natali@uib.no
†e-mail:julius.parulek@uib.no
‡e-mail:daniel@cmr.no

(as shown in Figure 2, where we use our method to quickly reproduce key-frames of a fault animation available on the web [IRIS 2014]).

Central geological events that shape the subsurface are deposition, erosion, folding, faulting, igneous or salt intrusion and layer compaction. While there has been research progress in rapid modelling of stratigraphy formation (deposition and erosion) and folding [Natali et al. 2014], there are no rapid modelling methods for faults and compaction events.

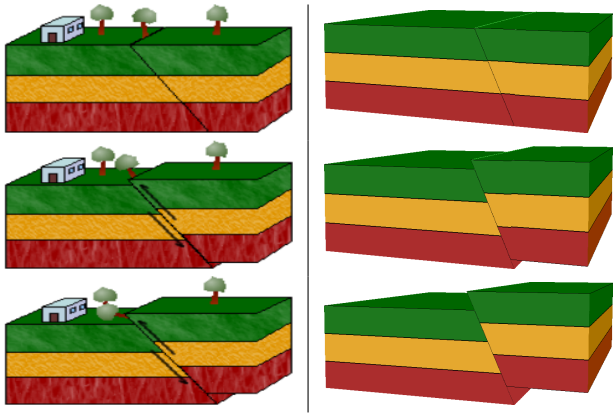


Figure 2: Key-frames of a hand-made fault animation next to key-frames of our animated technique.

Faults are important geological features. Their interpretation leads to an understanding of the behaviour of the crust of the Earth. Movements in the crust produce faults in rock layers. For instance, a standard approach to derive the direction of two lithospheric plates is to study faults generated by their displacement.

In summary, our contribution is an improvement to the work by Natali et al. [2014] with the addition of interactively animated faulting processes and compaction effect, due to deposition of upper layers. Moreover, we employ different visualization techniques that facilitate a visual navigation through the stratigraphic model, such as exploded view and adaptable staircase view. Computations that handle the processes and model representation are carried out on the GPU. In this way, we are able to interact with faulted models composed of many layers and animate them in real time.

2 Related Work

For a wide overview on illustrative visualization techniques and their application to several fields, we suggest the tutorial by Viola and colleagues [2006]. When considering medical illustrations, Sousa et al. [2005] present a volume illustration method for interactive simulation sessions. While in geological illustrative visualization, Patel et al. [2007] propose an approach to display volumetric seismic data.

Sketch-based modelling has evolved in recent years, together with interpretation of 2D curves for 3D reconstruction. For instance, the work by Zhu et al. [2011] achieves illustrations of scientific concepts, that are then enriched with parts animated by 2D flow simulation. Similarly, the paper by Rivers et al. [2010] enables users to generate complex 3D models with sketch-like input. Owada et al. [2004] present an interactive tool for modelling and inspecting volumetric illustrations with textures.

Sketch-based modelling of terrains has been addressed in the last years by a handful of works [Gain et al. 2009; Hnaidi et al. 2010; Natali et al. 2013]. Nevertheless, all of them aim at obtaining a terrain surface, they do not focus on the subsurface or geological features in it. What is really important for geological illustrations is the internal structure. That is: stratification (deposition and erosion), folds, faults, compaction of layers, fluvial systems and salt domes.

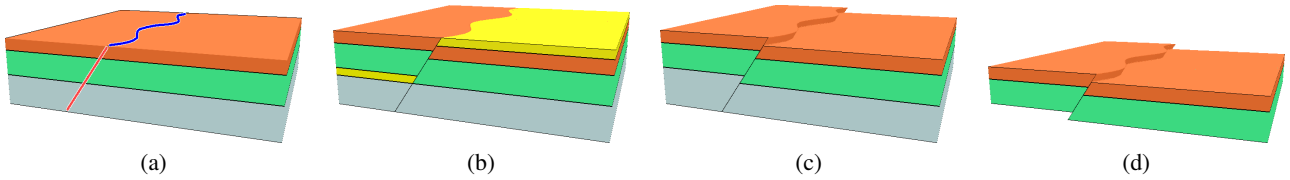
If we look for illustrative visualization in stratigraphic geology, the work by Natali et al. [2012] and an extension to produce animation, proposed by Lidal et al. [2013], can be found in literature. Both methods are able to include faults, but one limitation is given by the fact that each model is an extrusion of 2D curves to 3D space. Therefore it is not possible to define internal structures or faults which have other than linear edges when seen from aerial view. In the former work [Natali et al. 2012], two-dimensional sketches are drawn on a cross-sectional view and interpreted as boundaries of the layers or as fault surfaces. The user has to define the shape of the folded layers, a 2D fault and its displacement. Sketches correspond to surfaces that are then triangulated to build the layer-cake model. In the latter work [Lidal et al. 2013], time is taken into consideration as well. Key-frames are drawn on cross-sectional views. Interpolation between key-frames leads to an animation of the sketches, and thus of an extruded 3D model. On the other hand, with our approach, we are able to define internal structure, because the modelling process happens sequentially. Each stratigraphic layer represents a deposition at a certain period of time. Similarly to nature, where depositional and erosional processes generate historical imprint of the rocks, every following layer (partly) covers the one below. The idea of using layered data was initially proposed by Benes and Forsbach [2001], and subsequently extended in other works [Neidhold et al. 2005; Natali et al. 2014].

Looking at the state of the art of geological modelling [Natali et al. 2013], few approaches can in theory be adapted to produce animation of faulting. One possible solution is given by representing the layer-cake model by means of meshes (either two- or three-dimensional). The disadvantage in this way of proceeding is given by the continuous computations that are required to obtain all the intersections between each surface and the fault. Every small displacement in the faulting process involves a re-computation of intersection points. To compute the intersection between two meshes is already an expensive task, if a fault passes across many layers, the computational cost increases. This approach would require longer processing time compared to our rapid modelling. For simple planar faults, a mesh of the model can be divided in two by the fault plane. When this subdivision has been calculated, each part can then be slide along some user defined vector on the fault plane. No recalculation of intersections needs to be performed, and none of the parts need to be deformed due to the planarity of the fault. If the model is represented by Constructive Solid Geometry (CSG), faults can be made by clipping the model into smaller pieces. However, each piece would not deform, because only affine transformations are supported.

In literature, a further technique would be available to generate fault illustrations, also for non-planar faults. This technique is known as illustrative deformation, as introduced by Correa et al. [2007]. Models are defined by 3D volumes. The shape of a fault and its animation could be described by a predefined template. This would require that an expert has to author a set of templates in an authoring stage, one for each type of fault to be illustrated. Authoring a single template is time demanding, for communicational or teaching purposes many templates would be needed. Furthermore, every template has high memory requirements, much more than what is required by our compact layered representation.

3 Outline

In this section we give a complete overview of the process that allows us to generate a geological illustration from scratch, as shown in Figure 3. We start from an empty box in three-dimensional space,



(a) (b) (c) (d)

Figure 3: Steps needed to obtain a reverse fault are shown in this figure: from the sketches in red and blue defining the fault surface (a) to the final illustration (d). (b) and (c) are intermediate steps to displace the fault blocks, respectively with and without highlighting the supporting wedges. In (c) the top wedge is transparent and the bottom wedge is given the colour of the bottom blue layer.

that, as a container, bounds the volume of the model we will define. A geological event is described by a single sketch (either a point or an open or closed polygonal curve) and sketches are drawn on top of the box, as shown in the left image of Figure 4 where the box is viewed from above. For defining faults, the user also draws on the side surfaces of the box. Our system directly deduces what shape to create from the sketch: a single point means a layer of constant thickness (for example, Figure 3 (a) is initially defined by three layers of constant thickness); an open curve defines a river (as the upper sketch in Figure 4); a closed curve defines an area of erosion or deposition (as the lower sketch in Figure 4); a curve on a side of the box, followed by an open curve on top, defines a fault (an example of this case is shown in Figure 3 (a) by the red and the blue curve respectively). After having sketched a structure, the user can set parameters such as thickness of constant layers, degree of deposition/erosion or depth of rivers. Each map-view sketch is interpreted and converted to a heightmap, as shown in Figure 4. The superimposition of the heightmaps produces the layer-cake model, as described by Natali et al. [2014]. The order in the sequence of sketches is important, because each corresponding heightmap is interpreted as a specific geological event in time. The sketches, each representing a depositional or erosional event, are sequentially stacked. The conversion from sketch to heightmap requires a computation of a distance field on a grid, i.e. the distance of a grid point from the polygonal curve describing the sketch. This step is done only once during the model construction.

When a fault intersects a layer, the layer is split in two parts and saved as two distinct heightmaps, as described in Section 4.2. The displacement that is caused by the fault is defined by the user and assigned to the layers intersected by the fault (only one of the two layers obtained by the splitting moves according to the fault displacement). If more than one fault acts on the same layer, the displacements associated to each fault are summed together. The layers are only displaced in the rendering stage. Animation of a fault is obtained with a slider that reduces/increases the displacement associated to the fault.

We also simulate compaction. As the number of layers increases, the covered layers decrease in thickness as a function of the amount of material above.

4 Modelling Approach

We briefly describe how the model representation is defined in Section 4.1. We then describe the procedure that is necessary to perform faulting and compaction in Section 4.2. In Section 4.3, we propose three types of illustrative visualization techniques that can be employed to better show internal structures. The use of one visualization technique is not exclusive for the others, they can be combined. Finally, Section 4.4 identifies parts of the algorithm that have been parallelized and implemented on the GPU to achieve interactivity.

4.1 Internal Representation

We use the simple and effective layered representation described by Natali et al. [2014] for our 3D models. We have improved the method with respect to computational time involved in both the processing and the rendering stage, by parallelizing it on the GPU. This is necessary to create real-time animations. The approach presented by Natali et al. encodes the layer structure as a stack of heightmaps, each representing an event of deposition or erosion. Each heightmap therefore represents a geological event that happened at a specific time. In this paper, we generalize this representation to cover a larger number of cases that can be found in a geological scenario.

Each model is based on a dual representation: it uses two alternative types of heightmap, the so called *absolute* layers and the *relative* layers [Natali et al. 2014]. The two representations have different advantages. Relative layers are used in the definition of the model and computations on it. They are good for keeping track of layers independently from each other in terms of thickness; as well as for faulting by splitting relative layers, and for sliding the faults by offsetting the relative values. On the other hand, absolute layers are employed to generate and render the volume. In addition, the transformation between the two representations can be done in parallel.

The i-th *absolute layer* $h_i^{abs}(x,y)$ is defined for each point $(x,y) \in \mathcal{G}$ by the height of the top surface of the layer in the reference coordinate system of the model, $\forall i = 1,\ldots,n$. Where n is the number of layers, $\mathcal{G}$ is a uniform 2D grid and $h_0^{abs}(x,y)$ is defined to be zero in every point.

The i-th *relative layer* $h_i^{rel}(x,y)$ is defined for each point $(x,y) \in \mathcal{G}$ by the displacement

$$h_i^{rel}(x,y) := \Delta h_i^{abs}(x,y) = h_i^{abs}(x,y) - h_{i-1}^{abs}(x,y)$$

of two consecutive absolute layers, $\forall i = 1,\ldots,n$.

The relation between relative and absolute layers let us express the i-th absolute layer as

$$h_i^{abs}(x,y) = \sum_{k=1}^{i} h_k^{rel}(x,y), \forall i = 1,\ldots,n.$$

The above formalization of the relation between relative and absolute layers can be easily proven by substituting the definition of the i-th relative layer into the formula.

4.2 Geological Features

Important features that are present in subsurface geology, can be modelled with the representation we employ in our work. Every geological feature in which we are interested, can be defined in our sketching interface using curves. To support fast sketching, our system is, for most situations, able to deduce the type of operation that is sketched, based on how the sketching is performed. In this section, we describe methods to model different geological events.

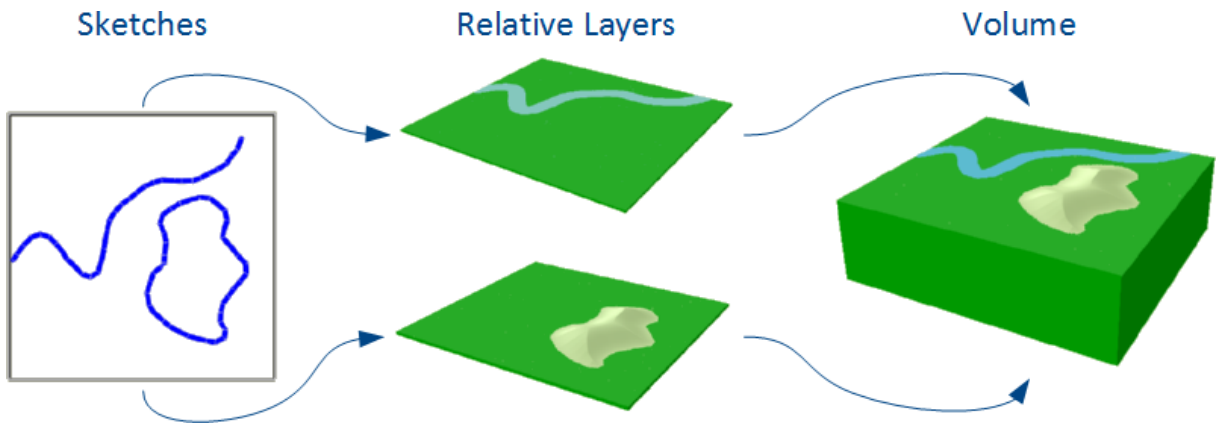


Figure 4: Overview of the conversion from sketches to relative layers first, and to volume representation afterwards.

4.2.1 Deposition and Erosion

As in nature, deposition and erosion are the basic processes that create rock layering. We express them by the sign of the heightmap values. A positive value means deposition of material; a negative value means erosion (of the material deposited by the previous layers). When deposition occurs, a new layer is introduced and its thickness depends on the heightmap values. In case of erosion, layers below are reduced in height, starting from the uppermost and proceeding to the others when their thickness is smaller than the amount of erosion. For instance, the valley visible in Figure 1 (b1) was created by an erosion which affects three layers, reaching the red one after having consumed and pierced the gray and the violet layer (because in that area the erosion is greater than their thickness).

4.2.2 Faults

A fault is created by splitting each layer that is crossed by the fault surface into two blocks. Since we use a heightmap representation, this corresponds to splitting a relative heightmap into two relative heightmaps, one heightmap for each of the two blocks separated by the fault. For this aim, absolute values of the layers are required, because the fault is defined with absolute values and intersections with layers are absolute as well. Therefore, before applying a fault, we convert the model from relative to absolute layers and we proceed with the inverse process afterwards.

In practice, the whole process of faulting a model is given by the following procedure. All the layers that are crossed by the fault surface must be detected. A fault does not need to entirely cut the model from top to bottom. It is important to be able to also handle the case of so called *blind faults*, when the fault intersects only some of the layers and not the uppermost. For every layer which is intersected by the fault, we compute the points of the heightmap that are affected by the intersection. If we also know the points of intersection of the layer below, it is possible to detect what kind of inclination the fault has. When a layer below does not exist, we have reached the bottom of the model and the intersection is computed with the plane defined by $\{z = 0\}$.

Knowing the inclination of the fault for a certain layer, permits to label each side block as a *foot wall* or a *hanging wall* (as in Figure 5). This depends on whether the block is respectively self-standing or not, in a hypothetical removal of the counterpart. This further information on the position of foot and hanging wall lets us derive from the direction of movement of the two blocks, if a *normal fault* (divergent direction, as in Figure 5 (b) and (d)) is created or a *reverse fault* (convergent direction, as in Figure 5 (b) and (c)).

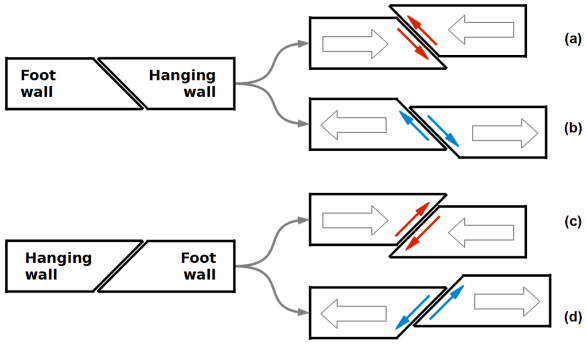


Figure 5: This scheme exhibits all fault cases our method is able to recognize. (a) and (c) represent reverse faults, whereas (b) and (d) represent normal faults.

For instance, the example in Figure 6 is a reverse fault, because the two blocks move towards each other.

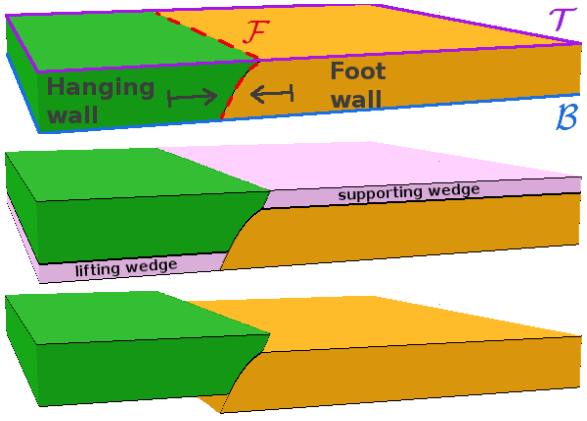


Figure 6: Procedure that: (top image) detects intersection with the fault surface and separate one layer in hanging (green) and foot (orange) wall; (middle image) inserts wedges (pink) and displaces blocks; (bottom image) sets wedges to be completely transparent.

We now give an example on how to split a layer into two with a fault. We consider a reverse fault with foot wall to the right (corresponding to case (c) in Figure 5), all the other cases (Figure 5

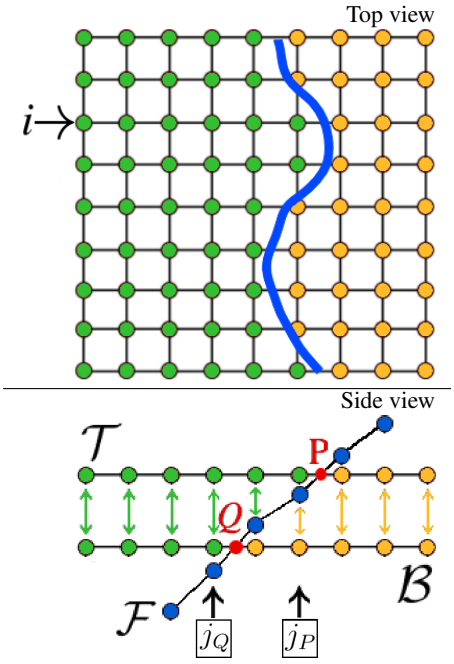


Figure 7: Top image: grid of the layer that is intersected by the fault (blue curve). Bottom image: cross-sectional view corresponding to the i-th row, where the fault $\mathcal{F}$ intersects the grids $\mathcal{T}$ and $\mathcal{B}$.

(a), (b) and (d)) are analogously defined. Notice that the process takes place on absolute layers, but output layers (hanging and foot blocks) are relative. This is just a design choice, in order to use the relative layer representation in the subsequent sliding of the fault. The order of relative layers in the stack is important, for instance a foot wall layer must be always before its corresponding hanging wall layer. Let us consider the situation shown in Figure 6 top, where we have a single layer that is cut by a fault $\mathcal{F}$. The foot wall is assumed to be on the right side and the blocks to have convergent directions. The layer is delimited by a top surface $\mathcal{T}$ and a bottom surface $\mathcal{B}$. We now consider the i-th row of the grid $\mathcal{T}$. An intersection point P between the fault $\mathcal{F}$ and the grid $\mathcal{T}$ is identified by two adjacent grid coordinates j_P and j_P+1 that define the separation of the left and the right layer block respectively (see Figure 7). Analogously, j_Q and j_Q+1 identify the intersection Q between $\mathcal{F}$ and $\mathcal{B}$. The displacement of the fault is defined by the user and it is a 2D vector V lying on the plane of the grids, i.e. defined by x and y coordinates. The vector V moves $P_i^{j_P}$ (by our convention hanging wall moves, foot wall is steady) on the surface of the fault. This is done by projecting V on the surface, which returns a 3D vector. This vector is saved together with the points of the grid and it is updated every time a fault is encountered, but it does not affect the grid until all the faults have been applied and the model is ready to be rendered. This avoids computations that would slow down our algorithm and eventually our animations.

The heightmap corresponding to the hanging wall is easily obtained as absolute values, because it has the same values as the top surface that bounds the layer, that is $h^{abs}_{hang} = h^{abs}_{\mathcal{T}}$. The heightmap corresponding to the foot wall, instead, implicitly store the information of the geometry of the fault cut. If we refer to Figure 7, the foot wall is defined by the absolute value at a grid point (i,j) as follows:

$$h^{abs}_{foot}(i,j) = \begin{cases} h^{abs}_{\mathcal{B}}(i,j) & \text{if } j \le j_Q \\ h^{abs}_{\mathcal{F}}(i,j) & \text{if } j_Q < j \le j_P \\ h^{abs}_{\mathcal{T}}(i,j) & \text{if } j > j_P \end{cases}$$

At this stage, we have our data structure ready to display the fault. In case there are other layers on top of the faulted one, they are automatically shifted and deformed due to the sequentiality of the data structure. A hanging wall block moves according to the shape of the fault surface. In case the fault surface is not defined by the user's sketch beyond a certain displacement, we perform a linear extension of the fault surface.

Two hidden layers have to be inserted in the model and we will call them *wedge* layers (depicted in pink in the middle image of Figure 6). One is needed to lift, if in the presence of a reverse fault, or to lower, in case of a normal fault, the layers of the hanging wall and convey the displacement. In the middle image of Figure 6, the green block is lifted by the left wedge and its tip is supported by the right wedge. The second wedge is only needed when a reverse fault occurs. It is always transparent and supports the material that otherwise would fall down on the foot wall after the fault displacement. This behaviour is similar to what is found in abstract fault illustrations, like the one displayed in Figure 2 left.

4.2.3 Compaction of Layers

Rock layers are exposed to a pressure (called *overburden* or *lithostatic* pressure), due to the weight of above layers. Lithostatic pressure increases with depth. In general, porosity decreases with depth, because lithostatic pressure tends to compact empty or fluid filled spaces of a material. To compute the compaction to which each layer is subject (Figure 8 shows an example before (left) and after (right) compaction), we need to know the total amount of deposition above the layer and the type of rock it is stratified on top. That is, we have to go through the layers on top and detect the height at a specific point (i,j) and its corresponding density of the material.

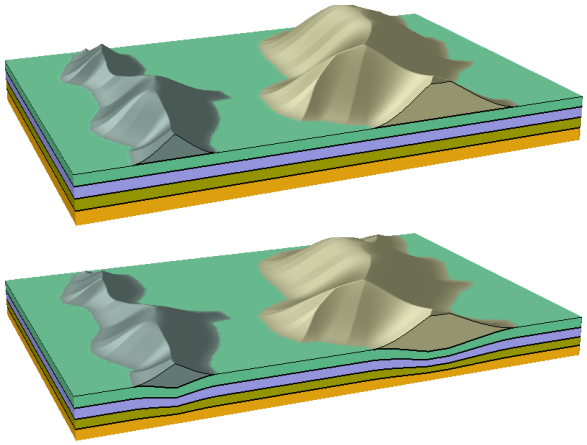


Figure 8: Comparison between a model where compaction is not applied (left) and where compaction is applied (right).

In a simplified situation, all the layers have the same mass density and it is enough to multiply relative values of the considered layer by a coefficient that depends on the height h^{abs}_{top} (the last layer on top) - h^{abs}_{l} (the considered layer), even for negative values as in Figure 9.

In this way, the more we deposit on top of a layer, the more it compress and lower its height in the final visualization. In a more realistic situation, a coefficient of compressibility of the material is taken into consideration as well.

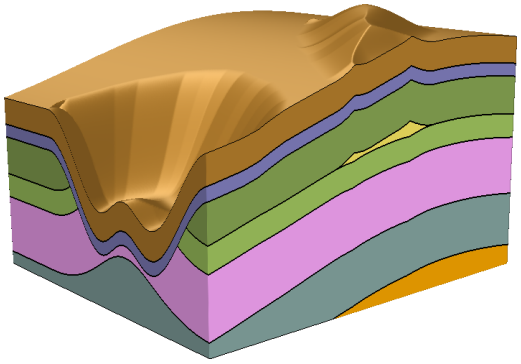


Figure 9: An exaggerated example that shows in the canyon that our approach produces a decompaction of layers when pressure has been removed from above them due to erosion of material.

4.3 Alternative Visualizations

Cutting planes, as in Figure 10 (b), are not always ideal for revealing important features in an illustration. We give the possibility to interact with the layer-cake utilizing alternative visualization techniques.

4.3.1 Temporal Slider

In contrast to a cutting plane, which creates a spatial cut, we want to consider a more domain specific temporal cut. Each absolute layer corresponds to a configuration of the terrain in a certain point in time. Therefore, if the aim is to show how the subsurface changed in time, it is enough to slide through the absolute layers by setting only some of them visible. Assuming to have reached a time step t, the illustration is given by rendering only the first t layers $h_1^{abs}, \ldots, h_t^{abs}$.

4.3.2 Staircase View

In this visualization, cutting planes interact with only one layer at a time. A default view employs as many cutting planes as the number of layers, all of them parallel to each other. Starting at the bottom layer and going through all of them, the associated cutting plane removes an increasing part of stratification that is visible (see Figure 10 (d)).

4.3.3 Exploded View

Exploded views make the internal structure visible by separating each layer. This is achieved by inserting a transparent displacement between layers. Its thickness can be set, which changes the space between layers accordingly. This type of visualization helps to show internal structures, such as channels (see Figure 10 (e)).

4.4 GPU Realization

In the realization of our framework, we build on the representation introduced by Natali et al. [2014]. The representation is simple; although powerful enough to tackle all the modelling scenarios described in our paper. Additionally, we improve their method to achieve a real-time modelling performance in order to maintain interactivity when updating the 3D model. We achieve this by migrating parts of the modelling stage to the GPU. Here, we utilize CUDA platform, which realizes a final heightmap construction from a set of relative layers. Each relative layer encodes deposition (positive value) or erosion (negative value), which represents a geological event that happened in a specific point in time. The final heightmap is then allocated, computed and resides in the GPU memory. This allows for direct display by an OpenGL based volume renderer, since the final heightmap is represented as a 3D texture. Here, we exploit CUDA-OpenGL compatibility, which provides us generation of even complex animations at reasonable frame rates (see Section 5).

The strength of the representation we employ is given by the fact that we do not handle an entire volume during the processing phase, we only consider a limited number of heightmaps, which depends on the number of sketches in the scene. With our boundary representation, we can still simulate effects otherwise only possible with a volumetric mesh representation.

A further improvement is applied to increase the processing speed. We parallelize computations on heightmaps, i.e. every point of the grid is consider independently from other points on the same grid. It just contributes to computations with corresponding points (same coordinates) on other grids.

Also the conversion from a sketch to a heightmap which requires distance calculations to the line segments on each sketch is performed in parallell on the GPU. Parallelizations through GPU is done every time there is a conversion between relative and absolute layers or vice versa. To generate interactive illustrative models and animations we exploit the efficiency of parallel computation. In case of compaction and faulting, there is only one value that makes the illustration change. In compaction, this value is the rate of compression of the layers; in faulting, it is the vector that defines the displacement of the layers on one side of the fault. In both circumstances, none of the values of the heightmaps changes, they are just either shifted in their position or scaled by a coefficient. For that purpose, it is not necessary to update every heightmap from CPU, but rather go through them in a parallel on the GPU.

5 Results

Here we present several examples demonstrating our proposed techniques. Figure 2 (left side) and Figure 1 show reverse faults (two blocks pushing against each other) defined by a planar surface and by a non-planar fault respectively. In case of a fault, we can choose whether to have conservation of layer volumes (as in Figure 6 bottom and Figure 3 (d)), which is more teaching oriented, or not (as in Figure 3 (c)), which is a more realistic situation. The former is obtained by setting the bottom layer fully transparent. Compaction is conveyed in the lower layers of Figure 8; the more deposit, the more compacted is the volume below the deposition. Figure 11 displays three time steps of a faulting process and, at the same time, reveals the user interface we use to create our models. In Figure 10, we show different kinds of visualizations of the same model: (a) the entire volume; (b) the volume clipped by a cutting plane; (c) employing a temporal cut; (d) staircase view; (e) exploded view of the layer-cake.

We have placed two illustrations of the same geological model side by side (see Figure 2) as benchmarking (i.e. comparison with a reference illustration): the one to the left is a 2D image authored by a geologist, the illustration to the right is a 3D volume that has been created using our approach described in this paper.

For our examples, we used a 512×512 grid resolution. In general, each model is independent of metric unit; rock layers and faults can have a large scale range, but, to get an idea of the dimension of the layers a model is trying to reproduce, the reader can think of layer thickness in the order of kilometers.

Table 1 shows the time that was required to generate some of the examples in the paper.

Techniques that are traditionally used to depict subsurface phenomena would require much more time, as confirmed by geologists who collaborated with us and tried our prototype system. Two-dimensional drawings obtained by hand or software are examples of commonly used methods. Procedural 3D modelling is less widely

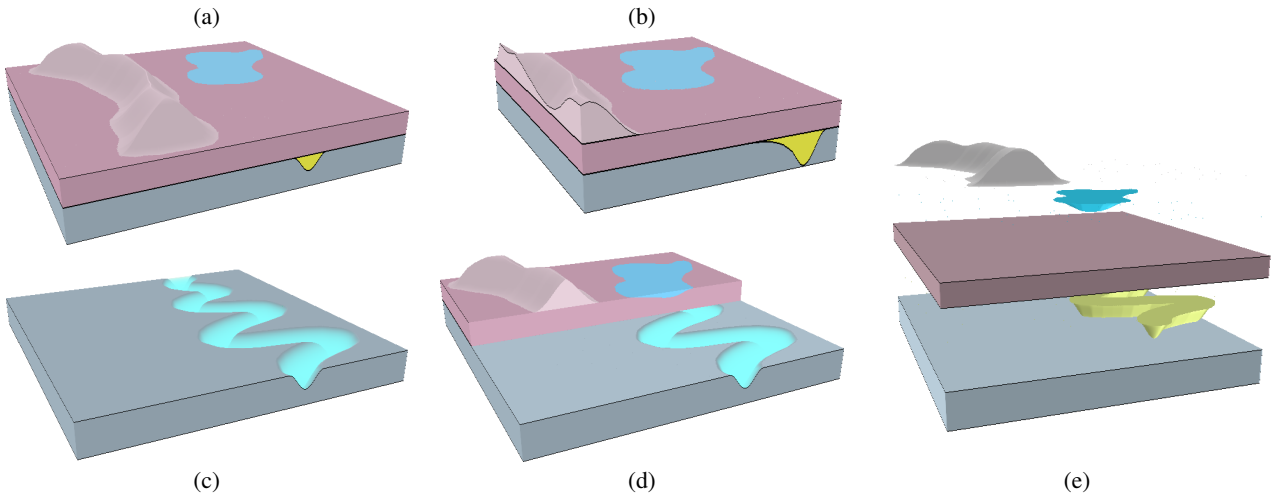


Figure 10: Different ways of visualizing the same model: (a) entire volume; (b) using two orthogonal and vertical cutting planes through the volume; (c) employing a temporal cut; (d) staircase view; (e) exploded view.

Model	Time (design and computation)
Figure 1 (a1-a2)	$<$ 2 minutes
Figure 1 (b1-b2)	$<$ 2 minutes
Figure 2 (right model)	$<$ 1 minute
Figure 3	$<$ 1 minute
Figure 8	$<$ 2 minutes
Figure 11	$<$ 1 minute

Table 1: Some of our models with their respective modelling times.

employed, requires more practice and is not developed for geological illustrations. One of the advantages of our approach is that it permits an interactive process of developing new ideas in a discussion within a group of domain experts. Moreover, a rapid approach to interactively model illustrations of geological phenomena helps to explain ideas to others, as our geologist collaborators pointed out.

6 Conclusions

We presented a new approach to rapid model subsurface geology to obtain illustrations and animations. In particular, we focus on creating illustrative visualizations of scenarios showing terrain stratigraphy, internal structures (such as channels), compaction of layers and faulting processes. We provide more than one type of visualization to be able to visually access features inside the volume. Geological events are modelled by means of sketches and real-time interaction is achieved with parallelization of our technique.

Limitations. Employment of heightmaps in the representation sets some limitations related to the shape of geologic features. For instance, a fault that is perfectly vertical could not be handled by our representation. However, most real situations can be achieved with our method. In our current implementation, we are able to achieve real-time interaction with up to about fifty layers (with grid resolution of 512×512).

Future Work. As a future improvement of our technique, we may have to define fault surfaces by meshes instead of heightmaps, so that we do not have to specify a fault surface on a whole grid. This can be useful when dealing with blind faults.

Acknowledgements

The authors would like to thank all reviewers that gave comments and suggestions for improvements, Paolo Angelelli and Ivan Viola for valuable feedback and expertise. This work is funded by the Petromaks program of the Norwegian Research Council through the Geoillustrator project (#200512). This paper has been also supported by the the PhysioIllustration research project (#218023), which is funded by the Norwegian Research Council.

References

BENES, B., AND FORSBACH, R. 2001. Layered data representation for visual simulation of terrain erosion. In *Proceedings of Spring Conference on Computer Graphics*, 80–85.

BERNHARDT, A., MAXIMO, A., VELHO, L., HNAIDI, H., AND CANI, M.-P. 2011. Real-time terrain modeling using CPU-GPU coupled computation. In *Conference on Graphics, Patterns and Images, 24th SIBGRAPI*, 64–71.

CORREA, C. D., SILVER, D., AND CHEN, M. 2007. Illustrative deformation for data exploration. *IEEE Trans. Vis. Comput. Graph. 13*, 6, 1320–1327.

GAIN, J., MARAIS, P., AND STRASSER, W. 2009. Terrain sketching. In *Proceedings of Symposium on Interactive 3D Graphics and Games*, 31–38.

HNAIDI, H., GUÉRIN, E., AKKOUCHE, S., PEYTAVIE, A., AND GALIN, E. 2010. Feature based terrain generation using diffusion equation. *Computer Graphics Forum 29*, 7, 2179–2186.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 409–416.

IRIS, 2014. `http://www.iris.edu/gifs/animations/faults.htm`.

KARPENKO, O. A., AND HUGHES, J. F. 2006. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Trans. Graph. 25*, 3 (July), 589–598.

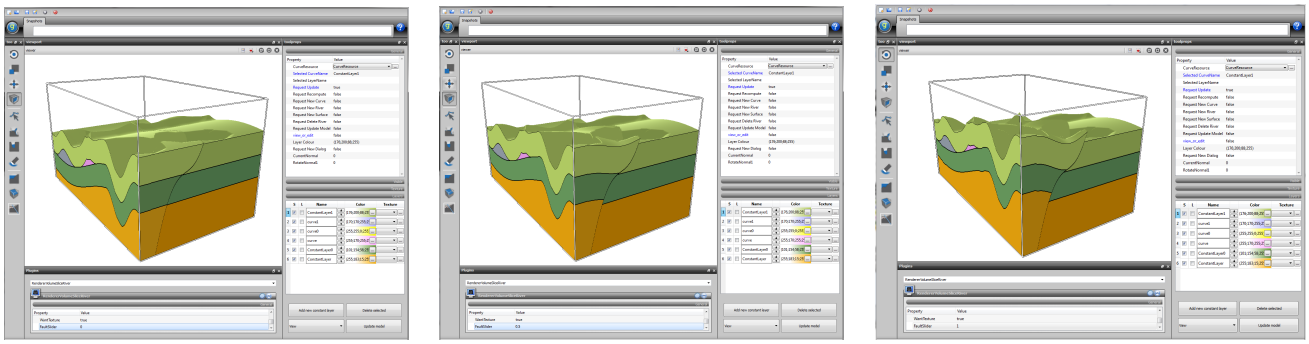


Figure 11: In this images we reveal the aspect of the user interface, within which an illustration of a faulted model has been built: in the left image the model is faulted but not displaced; in the middle image it is half-way displaced and completely displaced in the image to the right.

LIDAL, E. M., NATALI, M., PATEL, D., HAUSER, H., AND VIOLA, I. 2013. Geological storytelling. *Computers & Graphics 37*, 445–459.

MASRY, M., AND LIPSON, H. 2007. A sketch-based interface for iterative design and analysis of 3d objects. In *ACM SIGGRAPH 2007 courses*, ACM, New York, NY, USA, SIGGRAPH '07.

NATALI, M., VIOLA, I., AND PATEL, D. 2012. Rapid visualization of geological concepts. In *Proceedings 25th SIBGRAPI Conference on Graphics, Patterns and Images*, IEEE Computer Society, 150–157.

NATALI, M., LIDAL, E. M., PARULEK, J., VIOLA, I., AND PATEL, D. 2013. Modeling terrains and subsurface geology. In *Proceedings of EuroGraphics 2013 (STARs)*.

NATALI, M., KLAUSEN, T. G., AND PATEL, D. 2014. Sketch-based modelling and visualization of geological deposition. *Computers & Geosciences 67C*, 40–48.

NEIDHOLD, B., WACKER, M., AND DEUSSEN, O. 2005. Interactive physically based fluid and erosion simulation. In *Proceedings of the First Eurographics Conference on Natural Phenomena*, Eurographics Association, NPH '05, 25–33.

OWADA, S., NIELSEN, F., OKABE, M., AND IGARASHI, T. 2004. Volumetric illustration: Designing 3d models with internal textures. *ACM Trans. Graph. 23*, 3 (Aug.), 322–328.

PATEL, D., GIERTSEN, C., THURMOND, J., AND GRÖLLER, M. E. 2007. Illustrative rendering of seismic data. In *Proceedings of Vision Modeling and Visualization '07*, 13–22.

RIVERS, A., DURAND, F., AND IGARASHI, T. 2010. 3d modeling with silhouettes. *ACM Trans. Graph. 29*, 4 (July), 109:1–109:8.

SOUSA, M. C., EBERT, D. S., STREDNEY, D., AND SVAKHINE, N. A. 2005. Illustrative visualization for medical training. In *Computational Aesthetics*, 201–208.

VIOLA, I., SOUSA, M. C., EBERT, D., PREIM, B., GOOCH, B., ANDREWS, B., AND TIETJEN, C. 2006. Illustrative visualization for science and medicine. In *Eurographics Tutorial*.

VITAL BRAZIL, E., MACEDO, I., COSTA SOUSA, M., DE FIGUEIREDO, L. H., AND VELHO, L. 2010. Sketching variational Hermite-RBF implicits. In *Proceedings of SBIM '10*, 1–8.

ZHU, B., IWATA, M., HARAGUCHI, R., ASHIHARA, T., UMETANI, N., IGARASHI, T., AND NAKAZAWA, K. 2011. Sketch-based dynamic illustration of fluid systems. *ACM Transaction on Graphics 30*, 6 (Dec.), 134:1–134:8.