Daniel Patel *Editor*

Interactive Data Processing and 3D Visualization of the Solid Earth

Springer

Interactive Data Processing and 3D Visualization of the Solid Earth

Daniel Patel
Editor

Interactive Data Processing and 3D Visualization of the Solid Earth

Springer

Editor
Daniel Patel
Rapid Geology AS and Western Norway
University of Applied Sciences (HVL)
Bergen, Norway

ISBN 978-3-030-90715-0 ISBN 978-3-030-90716-7 (eBook)
https://doi.org/10.1007/978-3-030-90716-7

Mathematics Subject Classification: 68U35, 68U07, 65D17, 68U05, 65D18, 65S05, 65D05, 65D10, 68U10

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The solid earth contains valuable resources that can be extracted such as oil, gas, ground water, minerals and geothermal energy. The subsurface can also be used to store harmful human made byproducts such as CO_2. There is an ongoing shift away from oil and gas towards environmentally friendly energy. Although several of the book chapters focus on oil and gas, the techniques presented are generic and can be applied in other fields that address the mapping and interpretation of the subsurface, such as for identifying suitable reservoirs for CO_2 storage.

This book presents works detailing the application of processing and visualization techniques for analyzing the Earth's subsurface. The topic of the book is interactive data processing and interactive 3D visualization techniques used on subsurface data. Interactive processing of data combined with interactive visualization is a powerful combination which have in the recent years become possible due to hardware and algorithm developments. The combination enables the user to perform interactive exploration and filtering of datasets while simultaneously visualizing the results so that insights can be made immediately. This makes it possible to quickly form hypotheses and draw conclusions.

Case studies from the geosciences are not as often presented in the scientific visualization and computer graphics community as e.g., studies on medical, biological or chemical data. This book will give researchers in the field of visualization and computer graphics valuable insight into the open visualization challenges in the geosciences, and how certain problems are currently solved using domain specific processing and visualization techniques. Conversely, readers from the geosciences will gain valuable insight into relevant visualization and interactive processing techniques.

Subsurface data has interesting characteristics such as its solid nature, large range of scales and high degree of uncertainty, which makes it challenging to visualize with standard methods. It is also noteworthy that parallel fields of research have taken place in geosciences and in computer graphics, with different terminology when it comes to representing geometry, describing terrains, interpolating data and (example-based) synthesis of data.

The domains covered in the book are geology, digital terrains, seismic data, reservoir visualization and CO_2 storage. The technologies covered within these topics are 3D visualization, visualization of large datasets. 3D modelling, machine learning, virtual reality, seismic interpretation and multidisciplinary collaboration. People within any of these domains and technologies are potential readers of the book.

Chapters "Modeling Terrains and Subsurface Geology"–"Overview of Seismic Attributes and Seismic Object Extraction" present key concepts and review relevant literature within geometric modelling of the solid earth; visualization of seismic and reservoir data; and processing of seismic data for subsurface interpretation respectively. Together they give a good foundation for the chapters "Using Interactive Visualization and Machine Learning for Seismic Interpretation"–"Subsurface Evaluation Through Multi-scenario Reasoning".

Chapters "Using Interactive Visualization and Machine Learning for Seismic Interpretation"–"Subsurface Evaluation Through Multi-scenario Reasoning" present novel solutions to subsurface related problems using visualization and processing techniques. More specifically the chapter "Using Interactive Visualization and Machine Learning for Seismic Interpretation"–"Visualization of Large Scale Reservoir Models" present novel techniques for semiautomatic seismic interpretation (using convolutional neural networks); filtering and rendering of volumetric data; and large reservoir rendering respectively.

Chapters "When Visualization and Virtual Reality Made a Paradigm Shift in Oil and Gas" and "Evolution of VR Software and Hardware for Explosion and Fire Safety Assesment and Training" present VR solutions for the oil and gas domain. Chapter "When Visualization and Virtual Reality Made a Paradigm Shift in Oil and Gas" presents the successful application of VR and visualization for well planning and exploration. Chapter "Evolution of VR Software and Hardware for Explosion and Fire Safety Assesment and Training" presents a VR solution for safety training and visualizing of fire and gas simulations. Although the chapter "Evolution of VR Software and Hardware for Explosion and Fire Safety Assesment and Training" does not address the subsurface, it describes the technological continuation of the solution in the chapter "When Visualization and Virtual Reality Made a Paradigm Shift in Oil and Gas".

Chapter "Groupware for Research on Subsurface CO_2 Storage" introduces the multidisciplinary research field of carbon capture and storage (CCS). It presents the various data types in CCS, requirements for CCS collaboration software and an evaluation of commercial and research-based software.

Chapter "Subsurface Evaluation Through Multi-scenario Reasoning" presents logic-based techniques for subsurface modeling. The chapter demonstrates how spatial and temporal aspects can be formally captured and expressed with mathematical logic. By formally expressing the state of a subsurface layering, a logic system infers several scenarios that result in hydrocarbon accumulations in any of the layers. This approach is an interesting contrast to the learning based artificial neural network methods described in the chapter "Using Interactive Visualization and Machine Learning for Seismic Interpretation".

What is the target group?
Graduate students, researchers and professionals working on visualization and/or subsurface domains.

The fields of subsurface data processing and visualization is rapidly developing. It is therefore not uncommon that professionals are only aware of the data exploration tools and techniques being used at work. This book will present alternative tools and techniques in their domain and in related domains.

This book will also be an interdisciplinary platform for computer scientists and professionals in the industry to develop new ideas, as well as an interdisciplinary platform bridging computer graphics algorithms and geoscience algorithms.

Bergen, Norway Daniel Patel
August 2021

Acknowledgements Thanks to Stian Anfinsen, Jeremy Cook, Ingunn Westvik Jolma and Kristin Fanebust Hetland for supporting this work.

Thanks to the internal reviewers Emilio Brazil, Ivan Viola and Manfred Bogen, to the external reviewers Ilona Heldal, Rodrigo da Silva Ferreira and Simon Buckley, and to the anonymous reviewers. And not at least, thanks to all the authors who have contributed to this book and in particular authors from industry who do not have strong incentives to spend time disseminating their knowledge.

Contents

Modeling Terrains and Subsurface Geology

Daniel Patel, Mattia Natali, Endre M. Lidal, Julius Parulek, Emilio Vital Brazil, and Ivan Viola

Abstract The process of creating terrain and landscape models is important in a variety of computer graphics and visualization applications, from films and computer games, via flight simulators and landscape planning, to scientific visualization and subsurface modelling. Interestingly, the modelling techniques used in this large range of application areas have started to merge in the last years. This chapter is a report where we present two taxonomies of different modelling methods. Firstly we present a data oriented taxonomy, where we divide modelling into three different scenarios: the *data-free*, the *sparse-data* and the *dense-data* scenario. Then we present a workflow oriented taxonomy, where we divide modelling into the separate stages necessary for creating a geological model. We start the report by showing that the new trends in geological modelling are approaching the modelling methods that have been developed in computer graphics. We then introduce the process of geological modelling followed by our two taxonomies with descriptions and comparisons of selected methods. Finally, we discuss the challenges and trends in geological modelling.

D. Patel (✉)
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

M. Natali
Norwegian Institute of Bioeconomy Research (NIBIO), Aas, Norway
e-mail: mattia.natali@nibio.no

E. M. Lidal
Ulriken Consulting, Bergen, Norway

J. Parulek
Equinor, Bergen, Norway
e-mail: jparu@equinor.com

E. V. Brazil
IBM Research, Rio de Janeiro, Brazil
e-mail: evital@br.ibm.com

I. Viola
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
e-mail: ivan.viola@kaust.edu.sa

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_1

1 Introduction

Realistic appearance of natural sceneries has been a key topic in computer graphics for many years. The outcome of this research primarily targets the film and gaming industries. The modelling is often procedural and can be constructed with little user intervention, at interactive framerates. In most cases, only the top surface is the final product of the modelling process, even if subsurface features have been taken into account during the modelling.

Parallel to this development, the modelling of geological structures has been developed from the geological domain. This modelling process usually requires heavy user involvement and substantial domain knowledge. The model creation can often take up to one year of intensive work. The modelling process also includes data acquisition from the site which is to be modelled. The resulting model is usually a very complex 3D structure, consisting of a number of different subsurface structures.

The needs of the entertainment industry and the geoscientific domain are substantially different, although they represent similar natural phenomena. While the former one puts emphasis on interactive realistic visual appearance, the latter one focuses on the correctness from the geoscientific point of view. An analogy can be observed in digital capture of the flow phenomenon: computer graphics proposes visually pleasing methods of flow, while computational fluid dynamics proposes physically plausible methods.

In recent years, research in geosciences has identified the importance of rapid generation of geologic models at early stages in exploration for the purpose of expressing and communicating scenarios where a hydrocarbon trap is present. For rapid generation of models, the extensive development period of a typical geological model becomes a severe limitation. This raised the need for rapid modelling approaches that are common practice in computer graphics terrain modelling. This is one of the reasons why geoscientific modelling techniques are approaching traditional terrain modelling approaches.

One essential difference still remains: While the terrain synthesis for entertaining industries is carried out by artists for the purpose of content creation, the geoscientific models are created based on actual measurements and are done by geologists and other geoscientists based on a substantial level of background knowledge and expertise. Moreover, when modelling based on measurements, the input data are either densely covering a certain spatial area, for instance by means of large-scale acoustic surveys, or consist of sparse samples that are completed by extrapolating known values over the areas where no measurements are taken. There is a multitude of approaches to collect geological information, for instance from seismic surveys (2D, 3D, 4D), boreholes (1D), virtual outcrops from LIDAR scans (3D), or vertical outcrop analysis (2D) [1]. It is outside the scope of this report to provide in-depth information on the acquisition processes. However, the nature of these measurements strongly influences the modelling approach from the geoscientific perspective. The aim of this report is to assess different approaches for modelling geological structures, to compare methods from the computer graphics and geoscience domains and to suggest promising topics for future research agendas.

It is important to note that only a part of the research in geomodelling is of academic origin and publicly available. There is a substantial amount of geological modelling research available only in the form of ready-to-use tools in commercial software packages. This research is, unfortunately for this report, protected by commercial vendors and their algorithmic details are often not known to academia. Therefore we focus on describing those geomodelling methods that have been made publicly available. The aim is not to describe the functionality of commercial modelling packages or to attempt to reverse-engineer the methodology behind them.

We firstly provide, in Sect. 2, a light-weight background on essential properties of geo-bodies as compared to man-manufactured objects typically modelled by means of Computer-Aided Design (CAD). After becoming familiar with essential background knowledge, we discuss our proposed taxonomies and highlight principal differences in Sects. 3 and 4. High-level distinctions between a computer graphics approach and a domain science workflow are reviewed. The aim is to give the interested reader a notion of possible future integrative tendencies between these two fields by means of mutual adaptation of methodologies typical for one or the other domain. Section 5 consists of a comparison of selected techniques for modelling surfaces and for modelling solids. Finally, in Sect. 6 we suggest possible developments of computer graphics in geological applications.

2 Geological Elements

The study of structural geology divides the subsurface into geo-bodies [2] of different categories. Central objects are layers, horizons, faults, folds, channels, deltas, salt domes and igneous intrusions.

Much of the reviewed work shows how to represent geological feature such as horizons, folds, faults and deposition. Deposition occurs when eroded particles in nature are brought by wind, water or gravity to a different place, where they accumulate to form a new rock layer. The subsurface is composed of a set of layers with distinct material composition. The surface which delimits two adjacent layers is known as a *horizon*. Two fundamental geological phenomena involve modification of the original structure of horizons: the process of folding and the process of faulting. A fold is obtained when elastic layers of rock are compressed. It is defined as a permanent deformation of an originally flat layer that has been bent by forces acting in the crust of the Earth. Faults originate when forces acting on layers are so strong that they overcome the rock's elasticity and yield a fracture. Horizons are thereby displaced and become discontinuous across a fault. Channels are tubular subsurface structures, they are either fluvial paths that directly originate under the ground or are created when what was once a river has been filled with sediments and then covered with other layers together with the surroundings. Geological models can be divided into two different categories, *layer-based models* and *complex terrain models* [3]. The layer-based models, built of multiple horizontal oriented surfaces, are typically created to model sedimentary geological environments during ground-water mapping, or oil and gas exploration. In regions with complex geological structures or where the layering is not dominant, for instance when modelling igneous and metamorphic terrains, a more complex terrain model is needed which can be of a probabilistic nature. These complex terrains are modelled when exploring for metal and mineral resources. In this report, we focus on the layer-based models as they are spatially well-defined and share similarities with terrain modelling in computer graphics.

In two viewpoint articles, Turner [3, 4] provides a thorough introduction to the challenges of creating computer tools for modelling and visualization of geological models. The article formulates the essential domain needs and the capability to interactively model and visualize: geometry of rock and time-stratigraphic units; spatial and temporal relationships between geo-objects; variation in internal composition of geo-objects; displacement and distortions by tectonic forces; and the fluid flow through rock units.

Furthermore the following characteristics of the geo-bodies are highlighted: complex geometry and topology, scale dependency and hierarchical relationships, indistinct boundaries defined by complex spatial variations, and the intrinsic property heterogeneity and anisotropy of most subsurface features. These characteristics are, according to Turner, not possible to satisfy with traditional CAD-based modelling tools. Thus, dedicated geological modelling and visualization tools are necessary.

In geological modelling there are often scenarios that lack sufficient data, so in order to build a meaningful model, the creator must interpolate between the sparse sampled or derived data available. Traditional interpolation schemes for discrete signal reconstruction are not sufficient as the process needs to be guided by geological knowledge, often through many iterations, to produce a successful result. A plausible geological scenario has to follow certain geo-physical constraints. Caumon et al. [5] describe specific structural modelling rules for geological surfaces defining boundaries between different lithological layers. Geo-bodies exhibit spatial continuity, therefore abrupt geometric variations such as sudden change of normal orientation on the surface, and abrupt changes within a fault are not common. This implies that a structural model may be validated via reconstructing its depositional state. Caumon et al. also describe the typical process of creating a structural model. The modelling usually starts with fault modelling. The mesh can either be produced directly as triangle strips, based on the dip information or indirectly using a specific interpolation scheme. The second and most important step is to define the connectivity among fault surfaces. The last step is the horizon modelling.

Wellmann and Caumon [6] present a large work from the geoscientists point of view regarding geological modelling. They review geometric representations of subsurface structures with focus on geological realism with respect to observations, information, and knowledge. They also present methods to analyze, quantify, and communicate uncertainties in the models.

In Sect. 3 we will briefly categorize previous works in terms of the type of data that is being addressed (see Fig. 1) followed by an in-depth categorization of papers according to where they fit in the workflow of geomodelling (see Fig. 2) in Sect. 4.

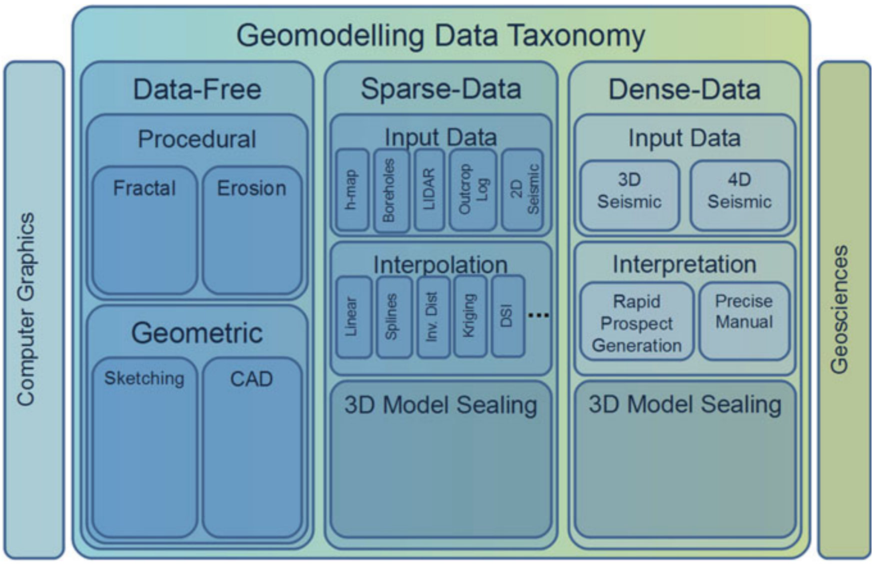


Fig. 1 Several methods are classified according to which domain they originate from (computer graphics or the geosciences) and what underlying data they use (data-free, sparse or dense data)

3 Geomodelling Data Taxonomy

By comparing the outcome of computer graphics terrain modelling with geoscientific modelling, we can roughly divide modelling into three distinct categories: the *data-free*, *sparse-data*, and *dense-data* scenario. The first category represents current and future trends in rapid modelling, where current methodology originates from the computer graphics research, while the latter two categories are developed from explicit needs in the geoscientific domain. This categorization forms one of the high-level taxonomies of the discussed modelling approaches. Figure 1 shows the taxonomy together with different modelling scenarios.

The *data-free* scenario has no ground truth information and therefore the geometric synthesis relies entirely on *procedural* [7, 8] and *geometric* modelling. The typical computer graphics research agenda proposes methodologies that alleviate the user from labor-intensive tasks by automating parts of the modelling. Procedural modelling offers the modeller specific, easy to handle input parameters which control the process of geometry generation. The geometry typically represents terrain surfaces. Procedural techniques in modelling have been facilitated mainly through *fractal* modelling [9, 10]. In the dynamic case, simple *erosion* models [11, 12] are utilized to create dynamic and realistic landscapes [13, 14].

The shortcoming of procedural modelling is usually the lack of direct control over the landscape development. The modeller has a rough idea of the landscape, but implicit parameter settings do not guarantee a match with the modeller's idea of an intended shape. Therefore a combination of explicit geometric modelling, to represent the modeller's expectations, with procedural modelling, to add realism, is a preferred strategy. On the other hand, geometric modelling can be a labor-intensive task. For rapid modelling scenarios, various forms of *sketching* metaphors [15–17] or modelling by example [18] provide fast ways to express the rough structure of a terrain or of a stratigraphic model [19, 20].

The *sparse-data* modelling scenario is the most frequently seen in the geoscience domain [21]. Very often it contains networks of boreholes [22–24], where the data needs to be interpolated between. Besides boreholes, there are often other acquisition types available, such as surface elevation models [25, 26], obtained through the process of remote sensing, typically acquired using satellite or aircraft-based sensor technologies. This heterogeneous pool of geoscientific data raises the challenge of data integration and data interpolation.

The main interpolating methods are the Kriging method, the Discrete Smooth Interpolation (DSI) method, the Natural Neighbor Interpolation method, Radial Basis functions, the Inverse Distance method, and spline methods. They will be discussed in Sect. 4.2.3.

Turner [3] demonstrates how to build a typical geological model from a *sparse-data* scenario by firstly interpreting bore-hole logs to construct triangulated surfaces of horizons [27] and then create the geo-bodies [28, 29] in sealed, boundary-representations [30] of the volumes between these surfaces. The modelling of faults [31] is also very important and Turner describes the challenge of modelling the interface between the boundary representation and the fault to avoid an unwanted crossing or empty spaces between the fault and geo-bodies. Using a structured mesh representation of the boundary surface can result in discretization errors, while using an unstructured grid representation [32] adds computational complexity and results in slow model construction.

The *dense-data* scenario is typically based on a single- or multi-attribute volumetric seismic dataset. The first challenge is purely of computational character, i.e., how to interactively display huge amounts of volumetric data, addressed, e.g., in the work of Plate et al. [33]. Utilizing volume rendering concepts, these datasets can be displayed without prior extraction of geo-bodies. Extracting geological structures from this data is necessary for consecutive steps along the workflow, such as reservoir modelling. This process is known as geoscientific interpretation and is a very time-demanding task. Typically, the original seismic dataset, consisting of the amplitudes of reflected sound waves, can be used to extract a number of derived attributes. These attributes are not geo-bodies, but their distribution over the 3D domain indicates the presence of certain geological structures. The *SHIVR* interpretation system [34] can extract geo-bodies based on scatter plots, as shown by Andersen and van Wijngaarden [35]. Rapid prospect generation can certainly benefit from faster interpretation. Patel et al. proposed methods for rapid horizon extraction in two [36] and three dimensions [37]. Afterwards, once the interpretation is available, 3D visualization can assist in validating the correctness of the extracted horizons with respect to original or derived attribute data [38].

A natural next step after 3D structural modelling is the development of a time-varying structural model. Inverse methods are often utilized in geomodelling to restore hypothetical geological scenario by going backwards in time [39–41]. Such an approach aims at restoration of deposited sedimentary layers, for example through unfolding. Restored information about palaeogeography often gives good indication where to search for hydrocarbon reservoirs.

4 Geomodelling Workflow Taxonomy

In Fig. 1, methods are categorized according to which domain they arise from, which is also tightly correlated with the amount of measured geologic data they handle. However, for the rest of the paper we have found it more appropriate to describe methods in the order they would be applied in a workflow for creating a geological model. Such a taxonomy is shown in Fig. 2. We have defined two separate workflows,

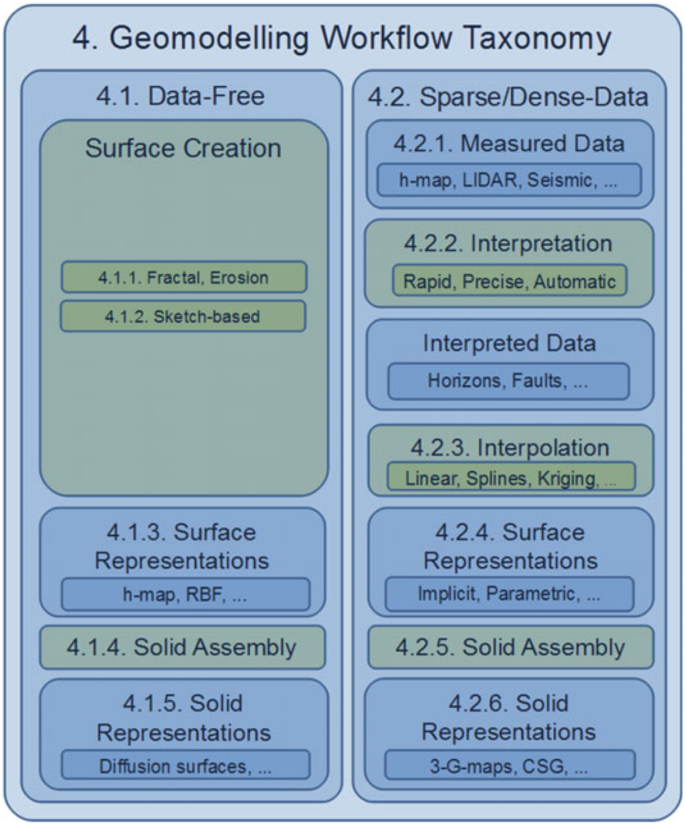


Fig. 2 Workflow taxonomy. Blue boxes represent data and green boxes represent action on the data to create richer data. The smaller boxes inside show examples of methods for that subtopic

one for creating models from no data, and one for creating models when data exists. The steps in each workflow are aligned with each other and we have specified in front of each category which subchapter they are described in. In the workflow where no measured data is required as input (data-free), some papers focus on surface creation, other general papers describe different mathematical surface representations. Several sketch-based papers describe different ways of fast sketching and assembling of solid objects and some focus on compact representations and fast rendering of complex solid objects. For the case where data is being used (the sparse/dense-data column), the workflow begins with measuring data, interpreting relevant structures, interpolating these into higher-order objects and representing these in some appropriate mathematical way. The structures are then assembled into solid geometry describing the subsurface. We have devoted a subsection to each of the topics described in the geomodelling workflow taxonomy.

4.1 Data-Free

This section describes works that do not rely on any measured or sampled input data, and where the models are created from scratch, driven by imagination or concept ideas and domain knowledge.

4.1.1 Fractal and Erosion Surface Creation

There are usually three approaches to generate synthetic terrains: fractal landscape modelling, physical erosion simulation and terrain synthesis from images or sample terrain patches. Before the work by Olsen [7], it was mostly possible to use simple fractal noise to obtain terrain surfaces, because computers were not fast enough to simulate erosion processes in real-time. Olsen proposes a synthesized fractal terrain and applies an erosion algorithm to this. His representation of terrains is a two-dimensional heightmap. To simulate erosion, he considers the terrain slope as one of the main parameters: a high slope results in more erosion, a low value produces less erosion. Starting from a noisy surface, called the *base surface*, erosion occurs to simulate weathering on a terrain. He applies two types of erosion algorithms to the base terrain: thermal and hydraulic. After testing the two erosion methods, he decides to combine the advantages of each, namely, the speed of the thermal erosion and the realism of the hydraulic erosion.

The ability to model and render piles of rocks without repetitive patterns is one of the achievements of the work by Peytavie et al. [8]. They focus on rocks and stones, which are found everywhere in landscapes. They provide realism to the scene, reveal characteristics of the environment and hint on its age. Before this paper, the canonical way of generating rocks was to produce a few models by artists, which were then instantiated in the scene. To create piles of rocks, collision detection techniques were applied with a high computational cost and low control. The authors propose aperiodic tiling of stones to avoid repetitive patterns. Two steps define the method proposed in the paper: the first is a preprocessing step that generates a set of aperiodic tiles constrained to maintain contact between neighbours; subsequently, the rock piles are created by exploiting aperiodic tiles from the previous step. Voronoi cells are employed to control the shape of rocks. In the construction of the Voronoi cells, an anisotropic distance to avoid round-shaped stones is used. Finally, erosion is applied to produce the final model. Meshes of the rocks are represented by using standard implicit surface meshing techniques.

Musgrave et al. [42] describe the creation of fractal terrain models, avoiding global smoothness and symmetry; these two drawbacks arise from the employment of the first definition of fractional Brownian motion (fBm) as introduced by Mandelbrot [43]. Moreover, there is a second stage in which the surface undergoes an approximation of a physical erosion process. Terrain patches are represented as heightmaps and the erosion process is subdivided into a thermal and a hydraulic part.

Concerning modelling terrains with rivers, Sapozhnikov et al. state that at the time when their paper [44] was written it was impossible to simulate the process of natural river network formation without making a substantial approximation; i.e. a simpler model that does not make use of the physical laws, but nevertheless reproduces the main geometrical features of a real river network. They use a random walk method to generate a set of river networks of various sizes.

Stachniak et al. [45] point out that fractal methods have been used to create terrain models, but these techniques do not give much control to the user. They try to overcome this by imposing constraints to the original randomly created model, according to the user's wishes. The method requires two inputs: the initial fractal approximation of the terrain and a function that incorporates the constraints to be satisfied in order to achieve the final shape. As an example, they show how to adapt a fractal terrain to accommodate an S-shaped flat region, representing a road. The constraint function defines a measure that indicates how close a terrain is to the desired shape. The final solution is provided by a minimization of the difference from the current terrain to the desired one.

Another way of combining procedural modelling with user constraints is described by Doran and Parberry [46]. In their work, they procedurally generate terrain elevation heightmaps, taking into consideration input properties defined by the user. The model lets the user choose amongst five terrain tools: coastline, smoothing, beach, mountain and a river tool. Together, these tools can be used to generate various types of landscapes.

A terrain surface is created by fractal noise synthesis in Schneider et al.'s work [9]. They aim to solve the problem that was one of the biggest disadvantages in fractal terrain generation at the time, namely the setting of parameters. They reduce such an unintuitive process of setting parameters by presenting an interactive fractal landscape synthesizer.

Roudier et al. [47] propose a method for terrain evolution in landscape synthesis. Starting from an initial topographic surface, given by a heightmap, they subsequently apply an erosion process to obtain the final 3D model. The erosion consists of mechanical erosion, chemical dissolution and alluvial deposition.

Chiba et al. [48] propose a method that overcomes the limitation of previous techniques for generating realistic terrains through fractal-based algorithms, but lacks ease of handling, i.e. it is not possible to modify the surface on the basis of constraints drawn by the user. The topology of the landscape is created by a quasi-physically based method, that produces erosion by taking velocity fields of water flow into consideration. The whole process of erosion, transportation and deposition is derived on the basis of the velocity field. Dorsey et al. [49] focus on erosion applied to one stone or rock, represented by its volume, taking into consideration weathering effects on it.

In the work by Benes et al. [50], a method for eroding terrains is described. A concise version of a voxel representation is utilized, and thermal weathering is simulated to erode the initial model. This new way to represent terrains has the advantage of being able to represent caves and holes. When applying erosion, all the layers and ceilings of the caves are involved in the process.

In a subsequent paper [51], a technique for procedural modelling of terrains through hydraulic erosion is introduced. The purpose is to use a physically-based approach together with a high level of control. Their algorithm takes in consideration that water dissolves material and transports it to other areas where, after evaporation, it is deposited. Contrary to previous techniques which tend to oscillate during water transportation, they provide a tool for hydraulic erosion that is fast and stable. They overcome oscillation by relying more on physical constraints than was previously the case. The erosion process consists of four independent steps, where each step can run repeatedly and in any order. These four steps are: introduction of new water (simulation of rain); material capture by water (erosion); transportation of material; and deposition at a different location.

Another work by Benes et al. [52] applies a hydraulic erosion fully based on fluid mechanics and thus on the Navier-Stokes equations that describe the dynamics of their studied models. They use a 3D representation provided by a voxel grid and the erosion process leads to a model that can show a static scene or part of an animation illustrating the terrain morphology. At each iteration of the process of erosion, a solution to the Navier-Stokes equations is computed to determine a pressure and velocity field in the voxels.

Interactive physically-based erosion is employed by Stava et al. [53] (Fig. 3). This work is based on physics by making use of hydraulic erosion, and on interactivity, which allows the user to take an active part during the generation of the terrain. The technique is implemented on the GPU and, because of the limited GPU memory, the terrain is subdivided into tiles, which allow them to apply erosion to local areas. Each terrain-tile is represented as a heightmap.

Kristof et al. [11] adopt 3D terrain modelling through hydraulic erosion obtained by fluid simulation using a Lagrangian approach. Smoothed Particle Hydrodynamics (SPH) [54, 55] is employed in this paper to solve dynamics that generate erosion. SPH requires low memory consumption, it acts locally, works for 3D features and is fast enough to work on large terrains.

For Hnaidi et al. [12], the terrain is generated from some initial parametrized curves which express features of the wished terrain (see Fig. 4). Each curve is enriched

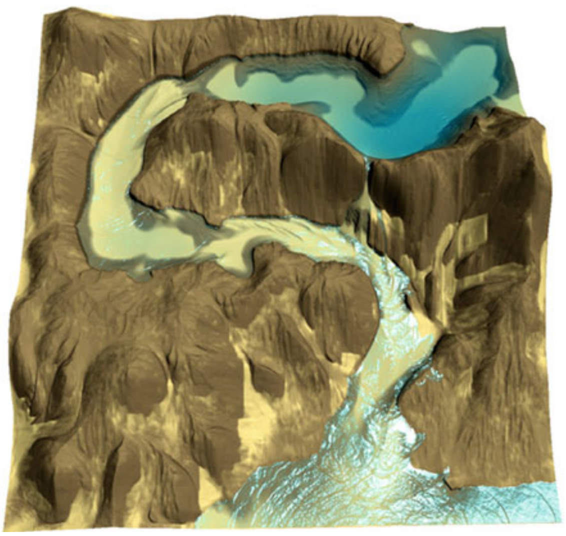


Fig. 3 The eroded terrain is obtained by simulating the movement of the water flow and transportation of rock particles [53]

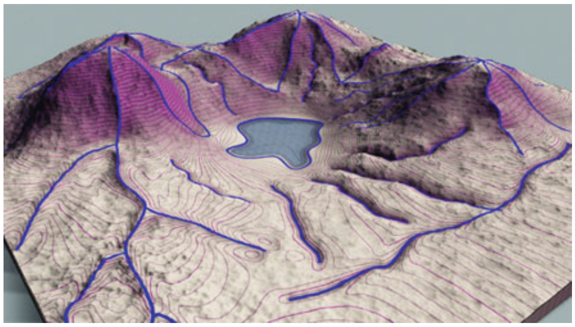


Fig. 4 Sketches, that are visible in the figure as blue strokes work as constraints during the method proposed by Hnaidi et al. [12]

with different types of properties (such as elevation and slope angle) that become constraints during the modelling process.

Prusinkiewicz and Hammel [13] address the problem of generating fractal mountain landscapes, which also includes rivers. They do it by combining a midpoint-displacement method for the generation of mountains with a method to define river paths.

Hudak and Durikovic [14] tackle the problem of simulating terrain erosion over a long time period. They use a particle system and take into consideration that terrain particles can contain water. The Discrete Element Method (DEM) is used for the simulation of the soil material and Smoothed Particle Hydrodynamics (SPH) simulates water particles.

Instead of procedural and erosion synthesis, Brosz et al.'s paper [18] introduces a way to create realistic terrains from reference examples. This process is faster than starting the terrain generation from scratch. Two types of terrains are necessary to obtain the final one: a base terrain, used as a rough estimate of the result, and the

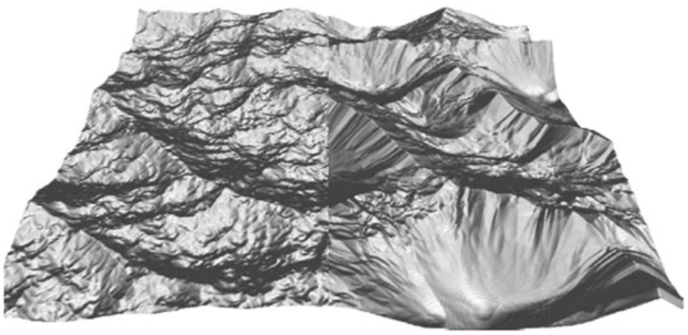


Fig. 5 Two types of noise shown on the left and right side of the image applied by de Carpentier and Bidarra [56] to achieve a realistic terrain

target terrain that contains small-scale characteristics that the user wants to include in the reconstruction. Brosz et al. have two common ways to generate landscapes represented as a heightmap: using brush operations to bring some predefined information or action on the surface; alternatively, simulation and procedural synthesis can be applied to obtain a realistic terrain. One drawback of using simulation is that it can be slow, while in the case of procedural synthesis, expressability is reduced by a limited set of parameters. De Carpentier and Bidarra try to combine brushing and procedural synthesis in their work [56] (an example is shown in Fig. 5).

Cordonnier et al. [57] use a volumetric representation to create models of terrains and near-surface geology. The method uses layers and sublayers with different physical properties to simulate erosion (improving their former work [58]) and uplift movements of the earth's crust. The authors embody these techniques in an interface that makes use of hand gestures to create large scale terrains (100×100 km) with a good level of realism. In another work, Cordonnier et al. [59], use a similar method to simulate erosion on multi-layer data but connecting it to combine various ecosystems.

A good source of further reading is Galin et al. [60] who perform an extensive review of the most recent works on terrain modeling from a computer graphics perspective. The authors classify the representation of the terrain into two broad groups, one using heightmaps, and one using volumetric representation. They also categorize over fifty papers according to factors such as:

- The variety of landforms the method can create.
- The realism of the landforms, where the authors used different ways to to measure realism.
- The range of scales of the landforms, which takes into account the precision and extensions—from meters to hundreds of kilometers.
- The authoring of the landforms, which evaluates how easy it is to create the terrain the designer has in mind.
- The efficiency of the algorithm, which takes into account time and space costs.

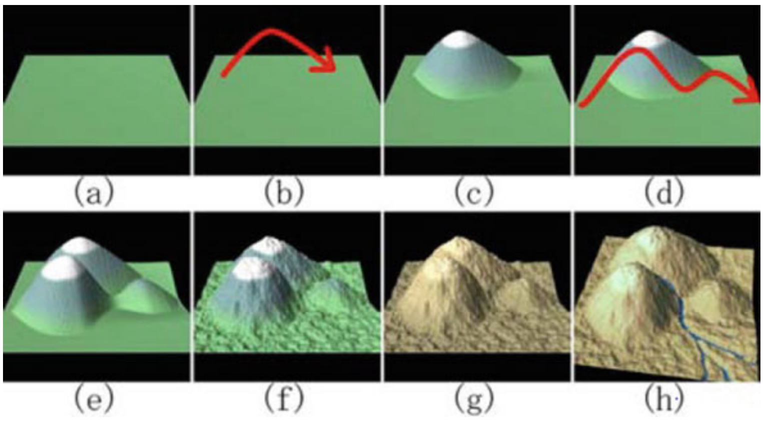


Fig. 6 Watanabe and Igarashi's process for obtaining a landscape surface by sketching [62]

4.1.2 Sketch-Based Surface Creation

Modelling a terrain surface with sketch-based tools rather than procedural techniques is a more controlled and intuitive process. Gain et al. present a paper [15] that describes procedural terrain generation with a sketching interface. Their approach aims to gather benefits and overcome some limitations of previous methods of sketch-based terrain modelling [61–63]. Watanabe and Igarashi [62] employ straight lines and, even though they yield a boundary for landforms using local minima and maxima of the user's sketch, they do not give the user the possibility to change the proposed shape (see Fig. 6). Furthermore, they apply noise onto the terrain after surface deformation, hence the obtained surface does not interpolate the user strokes exactly. Whereas landforms rarely follow straight lines, Zhou et al. [63] allow landforms to have more free-shape paths using a heightmap sketching technique as guidance for an example-based texture synthesis of terrain. In contrast to the method suggested by Gain et al. [15], they provide low and indirect control over the height and boundary of the resulting landform.

Extending the use of parametric curves to modeling terrains and their properties in a volumetric representation, Becher et al. [64] were able to create terrains with arbitrary vertical layouts, enabling terrain features such as arches, caves and overhanging cliffs. The application makes use of feature curves that define the local terrain properties and extrapolate these properties to the rest of the model area. By performing computations in compute shaders on the GPU, interactivity is achieved.

Guérin et al. [65] use a Conditional Generative Adversarial Network (cGAN) [66] to create terrains from sketches. A Generative Adversarial Network is a deep-learning method that maps images to images and has successfully been used in many applications [67]. In general, the cGAN method receives a pair of images as a training unit—the input image and the expected output image. To generate the terrains, the authors trained four cGANs, each one specialized in one task. The first one takes

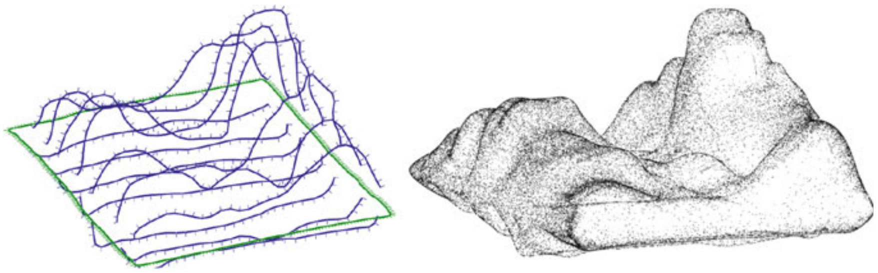


Fig. 7 Landscape example generated with a sketch-based approach by Vital Brazil et al. [17]. The model is represented with Hermite Radial Basis Functions. Sketch input is shown to the left, and result is shown with a stipple rendering style to the right

sketches representings rivers, valleys and ridges along with points with elevation information and creates the first terrain. Then the authors train a second network where the user can specify areas of constant elevation. To be able to complete missing information on the models, the authors train a third network with images of terrains. Finally, they create a network to simulate erosion processes. The first three cGANS were trained using real terrains images, while the last one uses a synthetic data set. After a large scale terrain model has been created, fine procedural details can be added by using the technique of *terrain amplification* [68].

Vital Brazil et al. [17] introduce a sketch-based technique to generate general 3D closed objects using implicit functions. They also show how to exploit their tool to obtain simple geological landscapes from few user strokes as shown in Fig. 7. Further sketch-based techniques that also define subsurface features in addition to the top terrain surface are presented in Sect. 4.1.4 Solid Assembly.

4.1.3 Surface Representations

Several of the fractal and erosional surface creation methods represent the surfaces as heightmaps. This is an easy-to-maintain datastructure which fits well with erosional calculations. The method by Vital Brazil et al. [17] can represent complex surfaces with overhangs or closed objects, using implicit functions defined as a sum of radial basis functions. Based on points with normals as input, a smooth implicit function, interpolating the points while being orthogonal to the normals, is created. Further details on this method is found in Sect. 4.2.3 Interpolation which also presents other methods that are closely related to surface representations.

Peytavie et al. [16] represent complex terrains with overhangs, arches and caves. They achieve this by combining a discrete volumetric representation, which stores different kinds of material, with an implicit representation for the modelling and reconstruction of the model.

Bernhardt et al. [69] present a sketch-based modelling tool to build complex and high-resolution terrains, as shown in Fig. 8. They achieve real-time terrain modelling

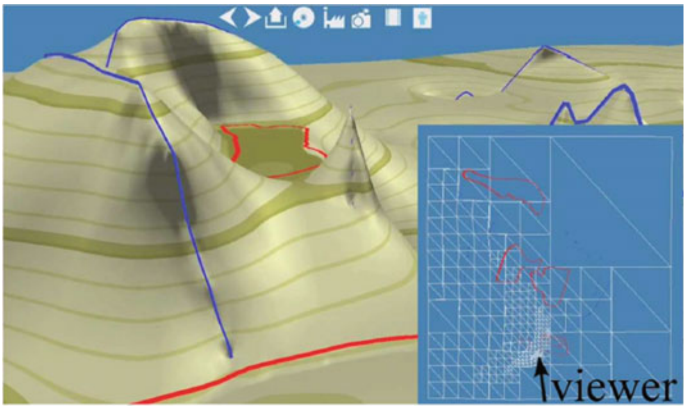


Fig. 8 Surface modelling with the sketch-based tool proposed by Bernhardt et al. [69]

by exploiting both CPU and GPU calculations. To represent large terrains, they use an adaptive quadtree data structure which is tessellated on the GPU.

4.1.4 Solid Assembly

By solid assembly, we refer to the process of assembling boundary surfaces or basic solid building blocks into a complete solid object. This work process is supported by CAD based tools.

As opposed to solid assembly for the sparse/dense data scenario (Sect. 4.2.5), in the data-free scenario described in this section, sketch-based methods are predominantly used both for defining the building blocks and for assembling them into a final model. With sketch-based techniques, the process to shape geological structures becomes intuitive and fast.

Natali et al. [19] present an example of sketch-based solid assembly of geological layer-cake models. Their approach allows a user to sketch geological layers and faults on a 2D cross section of the model which is then extruded to a 3D model. Texturing is applied on each layer and the the texture can be deformed using conformal texture mapping to e.g. follow the shape of the layer. The texturing helps convey information that is not captured in the geometric model such as the material type and material orientation within each layer.

Some of the stratigraphic layers found on earth are formed by deposition and erosion caused by fluvial systems. In 2014, Natali et al. [70] explored a way to provide fluvial systems illustrations to geologists, by means of an interactive 3D modelling approach. The focus is on evolving depositional and erosional landscapes by modelling rivers, deltas, basins, lakes and mountains. For allowing fast sketching, a minimalistic user interface is applied where a user draws on the top surface of the model, either a point, an open curve or a closed curve. A point is interpreted as a

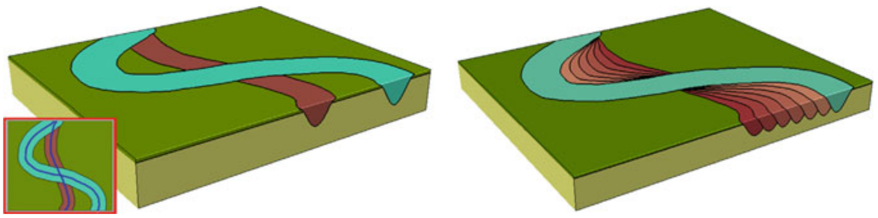


Fig. 9 Left: first and last configurations of the river are sketched and imprinted. Right:imprint of additional five intermediate stages of the depositional history by interpolating the two sketched rivers [71]

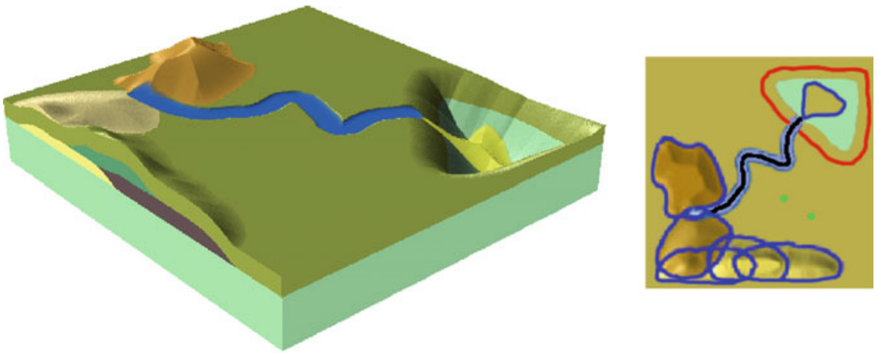


Fig. 10 Left: a sketch-based approach is used to create 3D surface and subsurface geological features. Right: map view of the sketched strokes used to define the model to the left. Black stroke defines a river, blue strokes define deposition, red stroke defines erosion, and green dots define constant deposition [70]

deposition of constant thickness on the whole model. Open curves are interpreted as rivers; the terrain is therefore eroded using a simplified mathematical shape that mimics a river cross section. The eroded river can be filled up again with deposited material. The depositional history of a meandering river can be represented on the 3D model by interpolating between two river curves (see Fig. 9). Closed curves can be associated to erosion for defining basins or lakes, or to deposition for creating deltas or mountains. The amount of deposition or erosion is defined as a function of the distance to the closed curve (see Fig. 10).

The internal representation of the 3D model associates each curve to a heightmap. The sequence of curves results in a stack of heightmaps which defines a solid model. Deposition translate to a heightmap with positive values. Erosion translates to a heightmap with negative values which affects the heightmap below in the stack. The final visualisation is achieved with a ray-casting technique (for each pixel, a ray is sent into the scene and intersections are found) on the heightmap stack. The paper includes a user study indicating that the approach is user friendly and that it covers most of the structures relevant for fluvial systems and depositions but is too limited for more general geology such as structural geology.

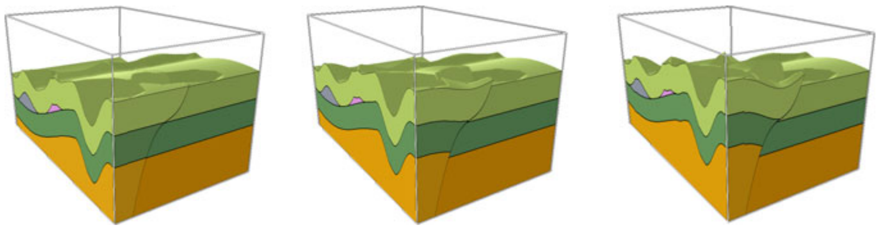


Fig. 11 Geological models showing a sequence of sliding along a nonplanar fault [71]

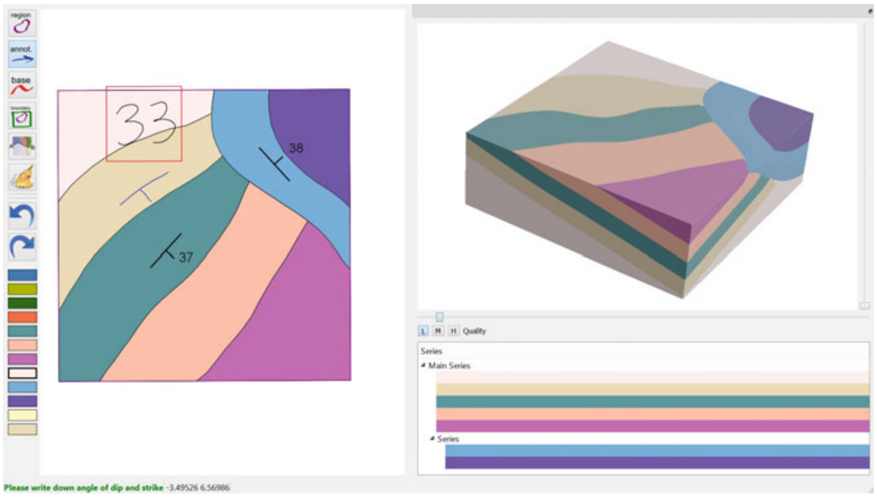
Further improvements and features are provided by Natali et al. [71] in a following article, where faulting and compaction are introduced in their models (see Fig. 11). The internal representation is founded on their previous work [70]. Faults can be directly drawn on the 3D model (one sketch on the side to define inclination and one on the top to shape the fault surface). From this information, each involved layer is detached in two by splitting the heightmap for each layer into two heightmaps. One of the advantages of this representation is that it can be easily parallelised on the GPU, which enables interactive sliding of the layers on either side of a nonplanar fault.

Amorim et al. [72] describe a sketch-based tool that produces a 3D conceptual model when the user draws and annotates a 2D geologic map. Their idea is to have the tool interpret the sketching symbols that geologists are trained in when drawing geological maps rather than requiring users to use the modelling metafors provided by the programmer. Geologists register their observations on a 2D map which is directly translated to 3D. Amorim et al. proceed essentially in three main steps, as shown in Fig. 12: firstly they provide a 2D canvas where to draw geologic contacts (basically a separation of different rock layers) as seen from map view, and related annotations for depth and inclination. Then a graph-based representation is generated from the drawn geologic contacts which is used to detect unconformities and the rock layers sequence. Lastly, a Constructed Solid Geometry (CSG) conceptual model is built by combining surfaces that represents rock separation. The surfaces are implicit surface interpolations of the initial sketches with further gradient notion coming from the user's annotations.

Lopes et al. [73] present a hardware and software solution where the user wears a VR headset and is able to shape a layered geological model using gestures in free air. The user defines with hand movements vertical-like surfaces which constitute the boundaries of the layer-cake model.

4.1.5 Solid Representations

In this subchapter we consider solid representations. They differ from boundary representations in that they are not hollow, but have spatially varying properties inside. Takayama et al. [74] present diffusion surfaces as an extension of diffusion



Please write down angle of dip and strike -3.49526 6.56986

Fig. 12 Left: the user draws a geologic map which includes geologic contacts (black curves) and dipping angles (T shaped symbols). Bottom right: the system finds a sequence of the rock layers from bottom to top that fits the sketch. Top right: a 3D geological model is generated from the user's sketch [72]

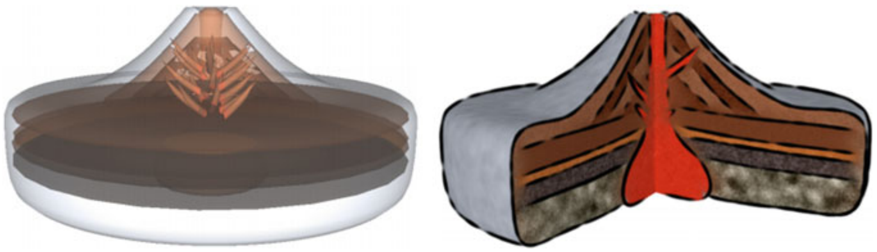


Fig. 13 A volumetric representation of a geological scenario using diffusion surfaces [74]

curves [75] for defining solids. The representation consists of a set of coloured surfaces in 3D, describing the model's volumetric colour distribution. A smooth volumetric colour distribution that fills the model is obtained by diffusing colours from these surfaces. Colours are interpolated only locally at the user-defined cross-sections using a modified version of the positive mean value coordinates algorithm. A result of the work by Takayama et al. [74] is shown in Fig. 13.

In the work by Wang et al. [76], objects are represented as implicit functions using signed distance functions. Composite objects are created by combining implicit functions in a tree structure. This makes it possible to produce volumes made of many smaller inner components. This multi-structure framework lets them produce models irrespective of resolution (see Fig. 14 for their geological application example).

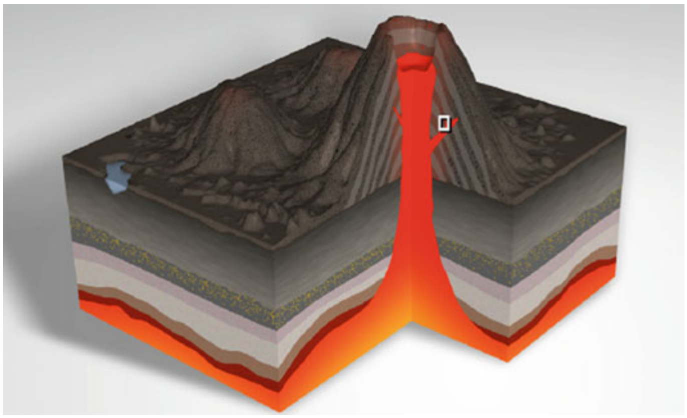


Fig. 14 A volumetric representation of a geological scenario using an implicit representation [76]

4.2 Sparse and Dense Data

This section describes methods that use sparsely scattered geologically measured data such as wells, or dense data such as 3D seismic reflection volumes, for creating a subsurface model. In contrast to data-free modelling where the user produces a model based on a mental image or hypothesis, the modelling is now constrained by values in the data.

4.2.1 Measured Data

Subsurface data can be collected in several ways, at various effort and expense. Seismic 2D or 3D reflection data is collected by sending sound waves into the ground and analysing the echoes. When the sound waves enter a new material with a different impedance, a fraction of the energy is reflected. Therefore, various layer boundaries of different strength are visible in the seismic data as linear trends. Well logs are obtained by drilling into the ground and performing measurements and collecting material samples from the well. Outcrops are recorded by laser scans together with photography (LIDAR) to create a 3D point cloud of the side surface of geology [1]. This surface can be investigated and visible layer boundaries can be identified and outlined as curves along the surface. Heightmap data can be collected from state-run databases.

4.2.2 Interpretation

Many interpretation methods operate on dense data which we gave a description of in Sect. 3. Several commercial tools exist for interpreting 3D seismic data. One example is the extensive software Petrel [77] where the user can set seed points and the system grows out a surface. The user can change the growing criteria or the seed points until a satisfactory surface is extracted. This can be time consuming. Kadlec et al. [78] present a system where the user interactively steers the growing parameters to guide the segmentation instead of waiting until the growing is finished before being able to investigate it. Fast extraction of horizon surfaces is the focus of Patel et al. [37]. Their paper introduces the concept of brute-force and therefore time-consuming preprocessing for extracting possible structure candidates in 3D seismic reflection volume. After preprocessing, however, the user can quickly construct horizon surfaces by selecting appropriate candidates from the preprocessed data. Compact storage of all surface candidates is achieved by using a single volumetric distance field representation that builds on the assumption that surfaces do not intersect each other. This representation also opens up for fast intersection testing for picking horizons and for high quality visualization of the surfaces. The system allows the user to choose among precomputed candidates, but editing existing surfaces is not possible. Editing is addressed by Parks [79]. He presents a method that allows to quickly modify a segmented geologic horizon and to cut it for modelling faults. Free-form modelling is achieved using boundary constraint modelling [80] (a method originating in the Computer Graphics community); this is simpler and more direct than spline modelling, which requires manipulation of many control points. Discontinuities arising from faults are created by cutting the mesh. Amorim et al. [81] allow for more advanced surface manipulation in their system. Surfaces with adaptive resolution can be altered and cut with several sketch-based metaphors. In addition, the sketching takes into account the underlying 3D seismic so that it can automatically detect strong reflection signals which may indicate horizons and automatically snap the sketched surface into position. Motta et al. [82] present a sketch-based approach to segment salt bodies in seismic data. This is useful as salt bodies are hard to extract automatically or even semi-automatically. Their work take an initial mesh and deform it using Laplacian methods for surface editing [83]. Liu et al. [84] present a sketch-based interpretation approach for 3D seismic data where the user can, in a 2D slice view, perform a fast partitioning of the slice for segmenting it and for illustrating it with textures of geological symbols.

The terrain and the subsurface on Earth is created through a series of geological events such as erosion, faulting and folding. Interpretation of the ground results in a model of the current subsurface state. The act of geological restoration is important in geology and tries to stepwise undo in reverse chronology the geological events that have taken place. In Lidal et al.'s paper [85], a tool for 2D creation, modelling and interpretation of geological scenarios in the subsurface is described. The tool allows for sketching up several alternative restoration timelines and organize them in a story tree. Garcia et al. [86] develop this concept further by incorporating a deformation simulation using a mass spring system which supports simple backward simulation directly on the 2D sketch. The backward simulation can undo the geological events of deposition, erosion, compaction, folding and faulting. Laurent et al. [87] apply the deformation method presented by Botsch and Kobbelt [80] to interactively deform, forward model and restore (backward simulate) 3D geological structures. The deformations presented in their work are however more coarse and global than the ones presented in Garcia et al. [86].

4.2.3 Interpolation

Key interpolating methods for surfaces in geosciences are the Kriging method, the Discrete Smooth Interpolation (DSI) method [88–90], the Natural Neighbor Interpolation method [91], Radial Basis functions, the Inverse Distance method, and spline methods.

Kriging is a statistical approach to interpolation that incorporates domain knowledge and is uncertainty-explicit [92, 93]. Kriging, like exemplar-based synthesis, creates a surface that has similar properties to an example dataset. Kriging, originating from geostatistics, calculates, based on available samples, how the variation of heights between samples change as a function of the distance between the samples. In terrains, neighbouring points have more similar height values than points further away. By calculating a variogram, which has variance on the y-axis and distance on the x-axis, the variability is captured. This information is then used to interpolate values by finding height values, so that the interpolated point fits the characteristics of the variogram.

The Discrete Smooth Interpolation allows for integration of geo-physical constraints into the interpolation process. The interpolator takes as input a set of (x, y) positions, some with height values and others lacking. After interpolation, the lacking height values have been calculated. Discontinuities between positions can be defined so that certain points do not contribute during interpolation. Typically, for a horizon surface, discontinuities would be added over fault barriers. In addition, constraints such as having points being attracted towards other points, having points being limited to movement along predefined lines or on surfaces can also be defined. These constraints are useful for interpolating geologic surface data. However, the method might not be well suited for cases with very little or no observation data (as indicated by De Kemp and Sprague [28]), such as in the data-free scenario.

Natural Neighbor Interpolation is also based on a weighted average, but only of the immediate neighbours around the position to be interpolated. A Voronoi partition is created around all known points and the weight is related to the area of these partitions around the unknown point.

Inverse Distance Weighted Interpolation works on points with planar position (x, y) and height z. The height for an unknown (x, y) position is a weighted average of the known points. The weight assigned to each known point diminishes with the distance to the unknown point according to a power function with a user defined factor p. When $p = 2$, Euclidean distance is used.

Carr et al. [94] present a method to build 3D surfaces from point clouds using radial basis functions (RBF). The proposed technique creates an implicit surface $S = \{(x, y, z) \mid F(x, y, z) = 0\}$, where F is a RBF, and interpolates the given points; i.e., $F(p) = 0$ for p in $\mathcal{P}$, where $\mathcal{P}$ is the given point cloud sampled from the surface. By the time of publication, the computational power restricted the number of points

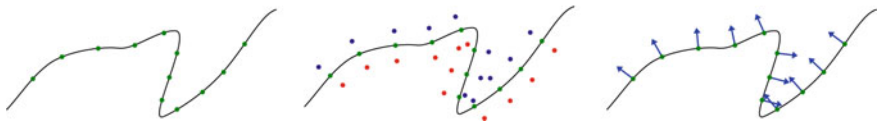


Fig. 15 Example of samples used by methods based on RBFs. From left to right: only (green) points on the surface; the auxiliary points proposed in [94] red inside and blue outside; and the points and normals used by the HRBF formulation in [96]

that were possible to fit using RBF's techniques. The authors showed that it was possible to adapt RBFs to fit a large number (up to 3×10^5) of points using methods to accelerate the function evaluation. This work also introduced an approach to improve the surface description enabling the representation of complex topologies. The authors add auxiliary points with values representing the signed distance of each point to the surface, then fitting an RBF that interpolates these values, i.e., $F(x_i, y_i, z_i) = v_i$, with $v_i = 0$ for points on the surface the, and $v_i \neq 0$ for auxiliary points. The authors proposed to place the auxiliary points on the surface normals (see Fig. 15). This method has successfully been adopted in mining industry to create geobodies with respect to minerals [95].

Later, Macêdo et al. [96] improved the RBF formulation to interpolate the surface's points and normals – Hermite-RBF (HRBF). This approach eliminates the need to create auxiliary points and is more apt to represent complex geometries. This new formulation enables the creation of closed surfaces using sparse samples, which was used by Vital Brazil et al. [17] to create terrain models from sketches. In Fig. 15, we illustrate the auxiliary points proposed in [94] and the points and normals used in [96]. It is easy to see the possible issue when one uses auxiliary points to represent surfaces that have narrow parts; the distance assigned in the normal direction could not represent the real distance to the surface. On the other hand, the use of normals makes the surface representation much more robust to this problem.

While the HRBF interpolator by Macêdo et al. was developed in the computer graphics and modelling domain, the implicit function interpolator methods [97, 98] had been developed a few years earlier in the geosciences domain for interpolating geological interfaces. Both methods use implicit functions for creating a surface that interpolates data based on both surface points and surface normals.

Another method interpolating both points and normals is made by Wu et al. [99]. They introduced a numerical technique based on an augmented-Lagrangian method and give an example of creating terrains. The method allows for defining a terrain based on two types of data, either the height value at a specific (x, y) position, or the normal vector of the terrain surface at a specific (x, y) position. The input is similar to the method used in the Hermite-Birkhoff Radial Basis Functions [72], but the mathematics behind is different, and the final result is a heightmap instead a set of implicit functions (RBFs). This approach may be suitable for sketch-based techniques, although it does not support overhangs which methods based on Radial Basis Functions do.

For all methods using radial basis functions for interpolating surfaces, it can be noted that there exists three degrees of specialization. Methods only interpolating points [94] (with no extra information such as normal orientation) use regular radial basis functions. When each point also has a normal, the Hermite-RBF can be used [96]. Finally, when each point either has a normal, or a position, or both, the Hermite-Birkhoff RBF formulation can be used [72].

An interpolation and surface representation system for geology is discussed in the work by Floater et al. [100]. Scattered point measurements can come in many forms, uniformly scattered, scattered in clusters, along measurement lines or along iso-curves. Fitting a surface through the points requires interpolation. Different interpolation methods vary in quality depending on the distribution of the scatter data. Floater et al. offer interpolation in form of piecewise polynomials (splines) on triangulations, radial basis functions or least squares approximations.

Although more of a connectivity algorithm than an interpolation algorithm, Ming and Pan [24] present a method for constructing horizons from borehole data. Each borehole dataset consists of a sequence of regions. Each region has its start and end depth specified as well as its rock type. Figure 19a exemplifies this. One rock type might appear in several layers and also the rock type sequence might vary between boreholes. This results in several possible connectivity solutions. The challenge is to make a suitable matching of layers to create a solid layer for each rock type.

Faults define the discontinuity of horizons, however, when interpreting seismic data, fault and horizon surfaces will not be perfectly aligned and will either have gaps or overlaps between each other. Closing gaps by extending horizon surfaces slightly and then cutting them at fault intersections can yield topological errors. Euler et al. [101] propose to use a constrained interpolator (DSI) to control which horizons will be extrapolated and to which faults they will be extrapolated to. This creates a correctly sealed model (see Fig. 16).

The paper uses a test dataset called Overthrust model SEG-EAGE 1994 [102] consisting of four horizons and two faults that merge into one fault. The four horizons divide a cube into five layers. Each layer is divided into three parts by the faults resulting in fifteen distinct blocks.

Belhadj [10] models a terrain through a fractal-based algorithm. The aim of the author is to reconstruct Digital Elevation Map (DEM) models. The surface is reconstructed with constraints consisting of scattered points of elevation provided by a satellite or other sources of geological data acquisition. Furthermore, it is possible for the user to change the final shape of the terrain by intervening with sketches on the model. As the goal was to have an interactive model, the choice of the algorithm has been a fractal based approach instead of a physically based one. Specifically, part of the work is based on the so-called Midpoint Displacement Inverse process (MDI), shown in Fig. 17. MDI does not allow reconstruction constraints, therefore a new adapted version of this technique has been proposed, named Morphologically Constrained Midpoint Displacement (MCMD). To be able to include constraints in the interploation computation, MCMD introduces changes in the order of computation of the midpoint displacement.

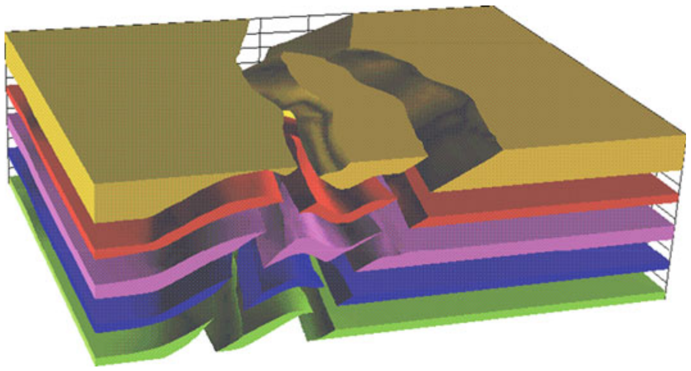


Fig. 16 The work by Euler et al. [101] presents a solid model made from surfaces in the Standard overthrust SEG-EAGE 1994 model [102]

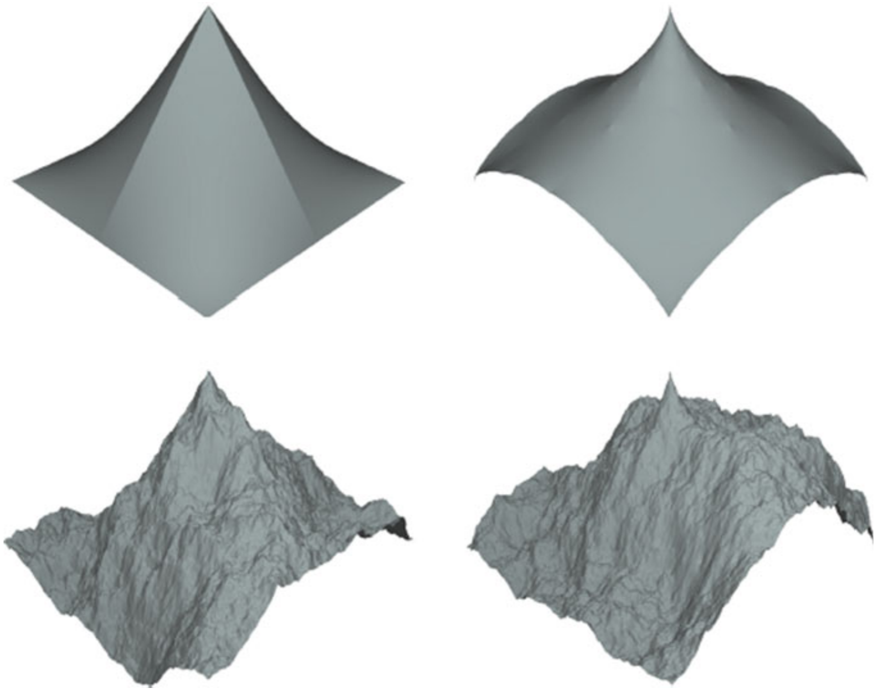


Fig. 17 Example by Belhadj et al. [10] of constrained mid-point displacement. Five points define the constraints for the generated surface

4.2.4 Surface Representations

In the work by Floater et al. [100], surfaces can be created from scattered data and are represented either as an explicit surface ($f(x, y) = z$), parametric surface ($f(i, j) = (x, y, z)$) or a triangulation. In addition they support offset surfaces defined by one parametric surface and a function that offsets the main surface along the surface normal. Parametric surfaces can be created from triangulated surfaces.

In the works by De Kemp and Sprague [28, 29], surface modelling using traditional Bezier curves and B-splines [103] is discussed. Bezier curves are used as approximative curves, while B-splines are employed when interpolative curves are better suited. All points can be interpolated or approximated. For non-interpolated points, attraction weights can be specified. However, controlling a large number of control points individually can be tedious and lead to meaningless localized distortions. To ameliorate this issue, the authors present the technique of having hierarchical control points of decreasing resolution so that the user can move control points in the hierarchy he/she wishes to displace the surface, to avoid manipulating an excessive number of control points at the lowest resolution level. They use the concept of structural ribbons for describing a curve with normals. It can be considered as a thin strip of the surface that can be fitted on available geological information such as outcrops or map traces.

In areas of importance in an interpretation with sparse interpolated data, the paper expresses the need for expert users to be able to override and alter the coarse approximation and easily update the model when new data arrives. An expert typically attempts to get an understanding of the processes that were operative in shaping the final geometry of a given structure while at the same time respecting the local observational data.

4.2.5 Solid Assembly

Solid geometric representations of subsurface structure are important for analysis. A sealed model enables consistent inside/outside tests, providing well-defined regions and good visualizations. It is also the first step for producing physical simulations of liquid or gas flow inside the model at later stages.

Baojun et al. [27] suggest a workflow for creating a 3D geological model from borehole data using commercial tools and standards. They use ArcGIS [104] for creating interpolated surfaces from the sparse data. They use geological relevant interpolation such as Inverse Distance Weighted, Natural Neighbor, or Kriging interpolation. This approach results in a collection of heightmaps which are imported into 3D Studio Max and stacked into a layer cake model (see Fig. 18). Then Constructive Solid Geometry (CSG) [105] operators are used to create holes (by boolean subtraction) at places where data is missing in the well logs. The model is then saved as VRML [106] enabling widespread dissemination since it can be viewed in web browsers.

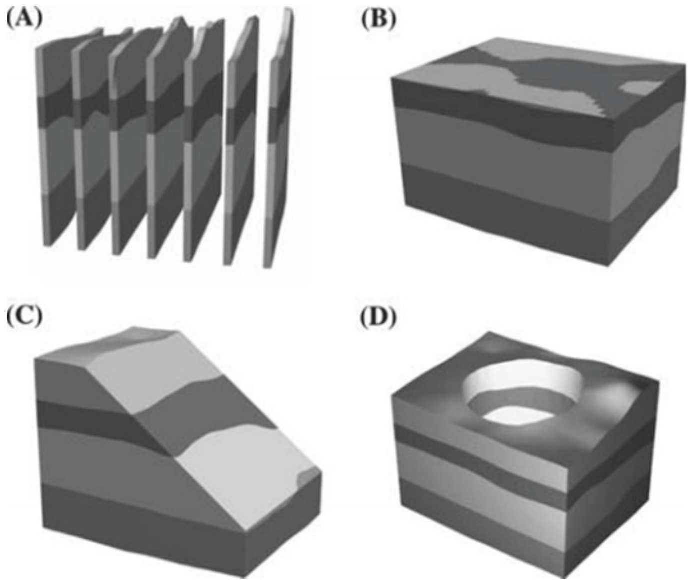


Fig. 18 Geological model made with CSG operations in 3D Studio Max shown with different cut styles available in the program [27]

When representing subsurface volumes using geometric surfaces for each stratigraphic layer, Caumon et al. [30] describe two constraints that must be followed. The constraints are that only faults can have free borders, i.e. horizon borders must terminate into other surfaces, and that horizons can not cross each other. Following the rules results in a correct and sealed model. This requires that each volume is described by a boundary triangulation with no holes and with shared vertices on seams of intersecting surfaces. Maintaining these constraints when editing horizons and faults is discussed. Mass conservation and deformation constraints during editing is also discussed. In their later work [5], additional geometric rules are introduced. The surface orientation rule states that geological surfaces are always orientable (i.e. having no twists, no Möbius ribbon topology and no self-intersections). Due to the physical process of deposition, they suggest an optional constraint requiring that horizons must be unfoldable without deformation, i.e. that they are developable surfaces with zero Gaussian curvature everywhere. They state that using implicit surfaces instead of triangulated surfaces directly enforce several validity conditions as well as making model updates easier, however at the cost of larger memory consumption. They also discuss the importance of being aware of the varying degree of uncertainty in the different measured data modalities and, for instance, using triangulations of different coarseness according to the sparseness and uncertainty of the underlying observations. The paper presents general procedures and guidelines to effectively build a structural model made of faults and horizons from sparse data such as field observations. When creating a model, they start with fault modelling and

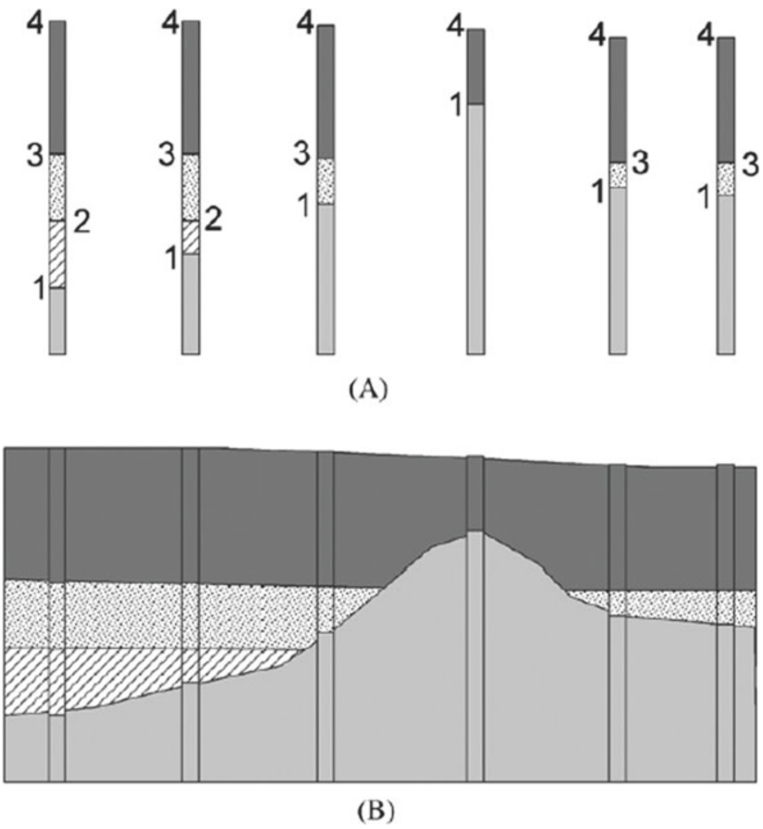


Fig. 19 Borehole data in **a** and resulting interpolation in **b** from the method by Lemon and Jones [22]

then define the connectivity among fault surfaces. Finally, horizons are introduced into the model. However, if the fault structure is very complex, they state that it is wiser to define the horizons first as if there were no faults and introduce the faults and their consequence on horizon geometry afterwards.

Lemon and Jones [22] present an approach for generating solid models from borehole data (see Fig. 19). The borehole data is interpolated into surfaces. For creating a closed model, they state and exemplify that CSG together with set operations can be problematic as the set operation trees grow quickly with increased model complexity. They simplify the model construction by representing horizons as triangulated surfaces while letting all horizon vertices have the same set of (x, y) positions and only varying the z positions (see Fig. 20). This simplifies intersection testing between horizons and makes it trivial to pairwise close horizons by triangulating around their outer borders.

Complexity increases when models must incorporate discontinuities in the layers due to the faults. Wu and Xu [31] describe the spatial interrelations between faults

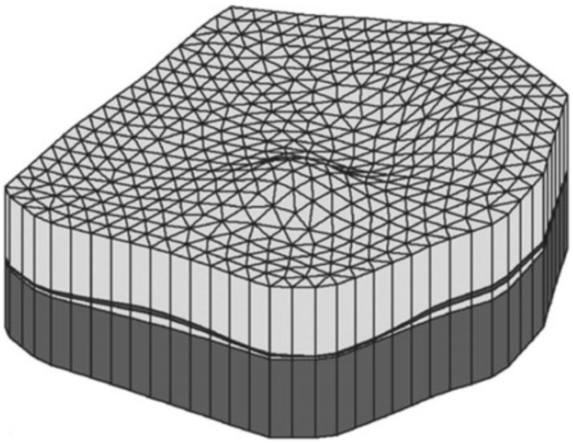


Fig. 20 Example of model created with method by Lemon and Jones [22]. The shared (x, y) vertex positions can be seen on the side surfaces

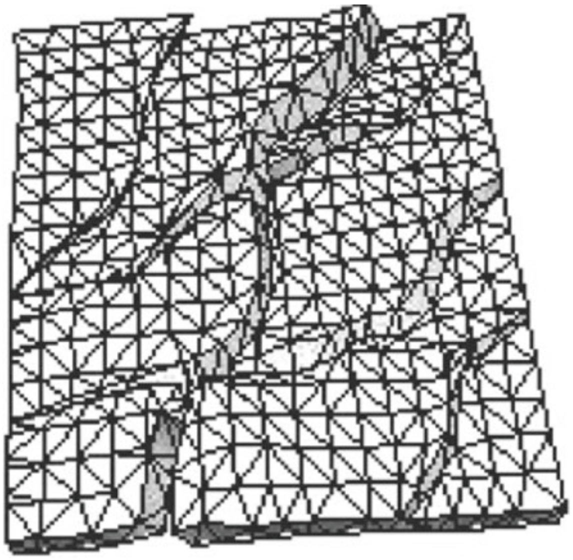
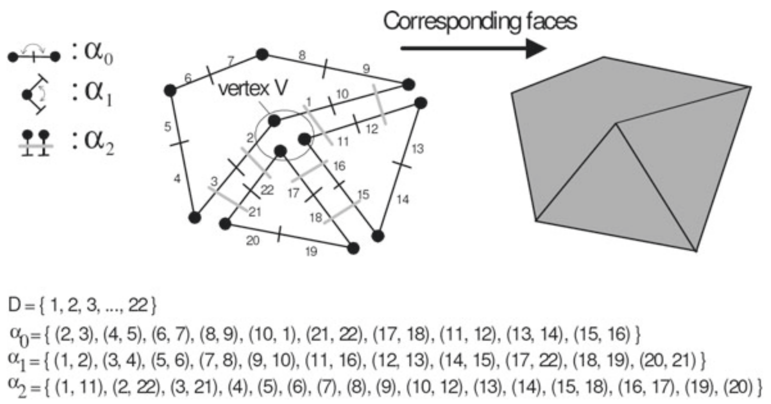


Fig. 21 Mixed-mesh model employed by Wu [31]. A regular mesh is used in non-boundary continuous areas, whilst an irregular triangulated mesh is adopted elsewhere

and horizons using a graph with horizons and faults as nodes. The graph is used to find relevant intersections and bounding surfaces which are Delaunay triangulated to form closed bodies (as shown in Fig. 21). In a follow-up paper [107], two types of fault modelling techniques are compared (based on what they call *stratum recovery* and *interpolations in subareas*) and a unified modelling technique for layers and faults is presented to solve the problems of reverse faults (i.e. convergent sedimentation blocks), syn-sedimentary faults (when slumping of sedimentary material happens before it is lithified) and faults terminated inside the model (blind faults).



$D = \{1, 2, 3, \ldots, 22\}$

$\alpha_0 = \{(2, 3), (4, 5), (6, 7), (8, 9), (10, 1), (21, 22), (17, 18), (11, 12), (13, 14), (15, 16)\}$

$\alpha_1 = \{(1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 16), (12, 13), (14, 15), (17, 22), (18, 19), (20, 21)\}$

$\alpha_2 = \{(1, 11), (2, 22), (3, 21), (4), (5), (6), (7), (8), (9), (10, 12), (13), (14), (15, 18), (16, 17), (19), (20)\}$

Fig. 22 Example of data structure explaining the 3-G-map definition [108]

4.2.6 Solid Representations

Solid modelling tools in CAD do not easily support subsurface features such as hanging edges and surface patches. Many papers describe data structures for representing the solid blocks that horizons and faults subdivide the subsurface into. Boundary representations are frequently used. Generalized maps, used for describing closed geological models, are introduced by Halbwachs and Hjelle [108].

A 3-Generalized map (3-G-map) [109] is a boundary representation appropriate for defining the topology of subsurface structures. A 3-G-map is defined as a set of darts D and three functions on them: α_0, α_1, α_2 and α_3 (see Fig. 22 for a 2D example). If one considers an edge as the line between two vertices, then a dart is a half-edge starting at a vertex and ending at the centre of the edge. The three functions map from darts to darts and sew half-edges into edges (by α_0), edges into polygons (by α_1), free polygons into connected polygons (by α_2), and defines neighbouring connected closed polygon volumes (by α_3). The 3-G-map is a simple yet powerful structure for defining the topology, in such a way that it is easy to traverse the space between connected or neighbouring vertices, surfaces and solids. Apel [110] presents a comparison of 3-G-maps with other boundary structures.

To create a 3-G-map, relations are defined on the faults and horizons, describing how a surface is terminated onto/cut by another surface. The construction process is divided in two, first geometries are created, then they are glued together through defining topology. The 3-G-map encapsulates the topology of the final model. For 3-G-maps, topology must be described very detailed. To relieve the user from this task, several abstractions have been suggested. By letting the user instead define the relation and cuts between horizons and faults in a graph or tree datastructure, the system can then generate a detailed topology description from this. In the work by Brandel et al. [111], the user specifies a graph of chronological order for when the surfaces have been physically created. In addition a graph describing the fault network using the relation "fault A stops on fault B", is specified. This work is

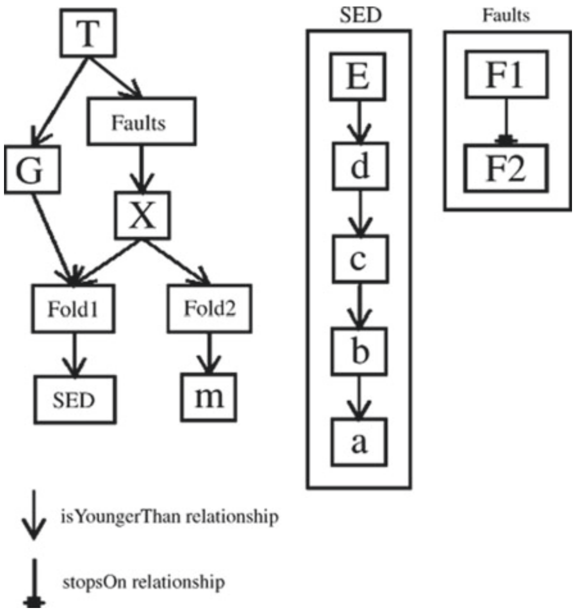


Fig. 23 Example of a graph that describes the relations between geological surfaces [112]

extended [112] to include meta-information in the nodes of the graphs for explicitly expressing the geological knowledge attached to each of the geological surfaces. Examples of such information is that a surface is onlap, erosional, older than or stops on another surface. An example of such a graph can be seen in Fig. 23.

Implicit surfaces (implicits) provide a suitable way to represent geological solids [98]. Essentially, such solids are described by implicit functions that can be expressed in different forms, e.g., distance based models, analytical functions, interpolation schemes like for instance RBFs, etc. Pasko et al. [113] generalized the above representations, which lead to an inequality, $f \geq 0$, also called *functional representation* of solids. Kartasheva et al. [114] introduced a robust framework to model complex heterogeneous solids, which was based on functional representation. The implicit solid definition is quite broad, and for instance, the terrain modelling using a heightmap can easily be represented by implicits [115].

Although not exactly a solid representation, we describe here methods for generating solid models having a discrete volumetric representation (i.e. consisting of voxels) based on descriptors such as parameters or 3D example datasets.

When a detailed model of subsurface geology is required, stochastic generation methods can be applied. A common usage is the filling of unknown volumetric content inside a closed boundary, such as the ore grade (grams per tonne) of a precious metal. Ensembles of plausible models are created stochastically for getting a better intuition of the probability distribution. This information can be used to decide where to extract the resource or where to take more samples to reduce uncertainty. The stochastic variability can be described by input parameters that match the (sparsely) measured sample data taken in the field.

Variograms (Kriging) as described in Sect. 4.2.3 for interpolating height data can also be used to create volumetric data. However, it produces smooth results and therefore cannot reproduce the heterogenous distributions found in e.g. ore grade. Multiple-Point Statistics (MPS) is more powerful and is able to reproduce hetereogeneities. As opposed to the variogram which uses two-point statistics as it is a function of two variables (the field values at two locations), MPS can closely reproduce spatial patterns from a training image (TI). The training image represents an explicit example of the heterogeneity that one wants to reproduce and can be physical field measurements from similar areas, or manually artificially made data that have the wanted variability. A recent overview of MPS has been made by Tahmasebi [116].

Well-defined geometric objects can also be generated stochastically. See Abdollahifard and Ahmadi [117] as an example. This is called object-based methods and is achieved by producing binary volumetric data instead of scalar-valued data. The binary volumetric data must follow certain topological constraints, such as having connected long strands in the case of creating channels. Stochastic methods have also been used for perturbing the geometry of an existing model. The motivation for perturbing the model is that the actual position of boundaries is only approximate due to e.g. limited resolution of measurements. An ensemble of models can be created to better represent the space of possible variations. Realistic simulations of, e.g., fluid flow (oil migration) can be performed on each model, and the results can be analyzed for identifying the sensitivities. The analysis can answer if, e.g., all variations create the same result, or if there are two main results (attractors) that the simulations evolve towards, and which parts of the model are most sensitive. When perturbing the model, it is important that the result also abides by known constraints. This is explored by Wellmann et al. [118] where they perturb horizons and faults of a model.

An interesting parallel development of MPS and texture synthesis in computer graphics has taken place and is described by Mariethoz and Lefebvre [119]. Both fields build on the same methods. In computer graphics, there is a need to generate realistic textures for applications such as video games and animated movies. For this, training images called exemplars are used as input.

Generative adversarial networks (GANs) are a relatively new technique that has been shown to create more realistic models than existing geostatistical modeling methods [120]. GANs have also been used to generate terrains, as discussed in Sect. 4.1.2. Using a GAN, a wide range of conceptual geological models honouring constraints such as well data can be generated. The GANs are trained using a library of models. GANs couple two competing deep convolutional neural networks: a generator, which creates new example models, and a discriminator, which differentiates between real and synthetic models. The networks are trained in an adversarial manner until the generator can create synthetic images that the discriminator cannot distinguish from "real" images. Two examples generated with the method by Zhang et al. [120] are shown in Figs. 24 and 25.

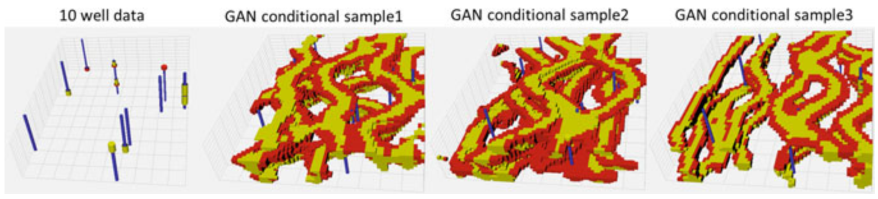


Fig. 24 Conditional fluvial samples generated by GANs in 3D. Ten well data with the interpreted facies in the left-most display. Three categories are used (shale: blue; channel sand: yellow; levee: red). All three generated models (sample 1–3) generated by GANs honor the well data [120]

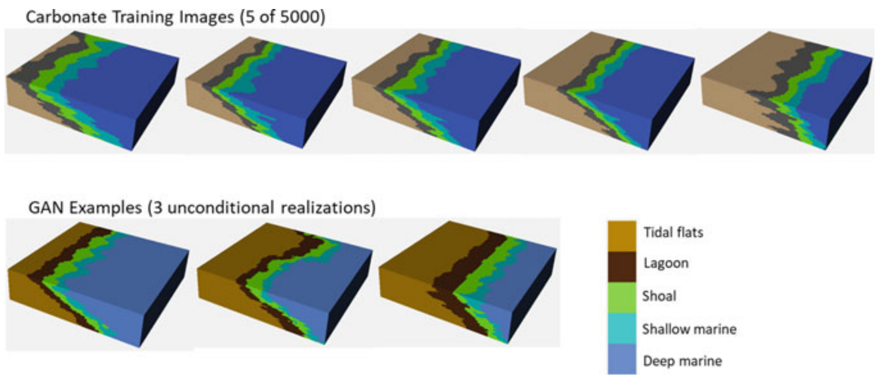
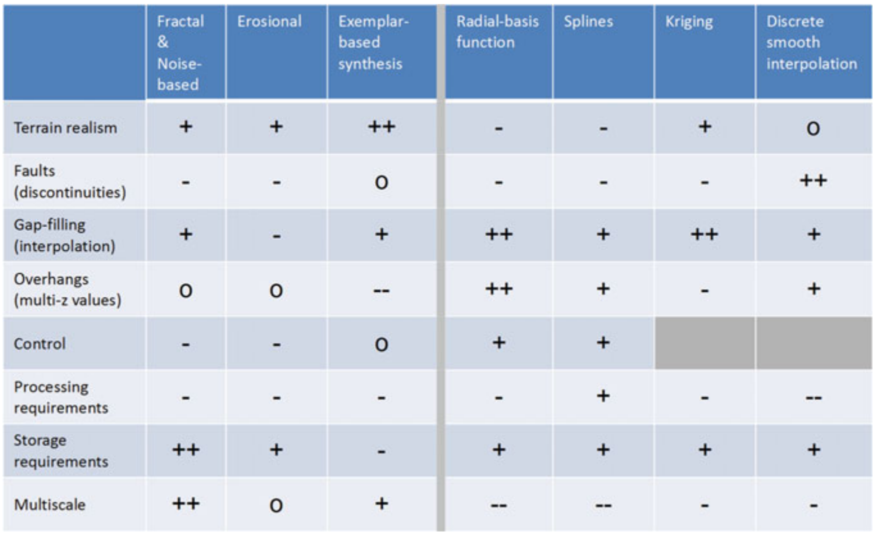

Carbonate Training Images (5 of 5000)
GAN Examples (3 unconditional realizations)
Tidal flats
Lagoon
Shoal
Shallow marine
Deep marine

Fig. 25 Training image examples (top) in 3D with 5 voxel categories (see legend bottom right) and unconditional realizations generated by GANs (bottom) [120]

5 Comparing Surface and Solid Representations for Geomodels

In this section, we compare and discuss the surface representation methods in Table 1 and solid representations in Table 2, in the context of how well suited they are for modelling geologic structures. The interesting features of such representations are: how close to a natural terrain the top surface is (*Terrain realism*, i.e., the ability to portray the properties of terrain such as randomness and the occurrence of all frequencies); modelling of faults (discontinuities); interpolation of input points (*gap-filling*); support for multi-*z* values (*overhangs*). Ease of modelling (*control*), processing requirements, storage space requirements and the ability of simultaneous representation of high- and low-level details (*multiscale*) will also be discussed for each category. In Table 2, we compare the techniques with respect to their ability to model layers; support for tubular structures such as channels, caves or holes; ease of modelling; processing and storage requirements and multiscale support. These features are graded with plus for good support, minus for bad support, or 0 if neutral.

Table 1 This table compares the abilities of surface representation methods in terms of modelling geological features



	Fractal & Noise-based	Erosional	Exemplar-based synthesis	Radial-basis function	Splines	Kriging	Discrete smooth interpolation
Terrain realism	+	+	++	-	-	+	o
Faults (discontinuities)	-	-	o	-	-	-	++
Gap-filling (interpolation)	+	-	+	++	+	++	+
Overhangs (multi-z values)	o	o	--	++	+	-	+
Control	-	-	o	+	+		
Processing requirements	-	-	-	-	+	-	--
Storage requirements	++	+	-	+	+	+	+
Multiscale	++	o	+	--	--	-	-

Methods to the left of the gray separator procedurally create surfaces, while methods to the right are interpolative

5.1 *Surfaces*

This subsection discusses the surface methods described in Table 1. The methods are split in two by a gray separating line. The three first methods produce surfaces procedurally, while the four last methods (four last columns) interpolate points into surfaces. Fractal techniques create a surface from input parameters; erosional methods create a surface from an input surface and simulation parameters; while exemplar-based methods create surfaces based on a collection of surface examples. Radial-basis functions and splines are defined by control points possibly set by a user. Kriging and DSI methods are completely automatic; therefore user control does not apply to them, and they are grayed out in Table 1. For comparing the capability to model faults, although any method can support this by splitting the surface into two, we strictly evaluate the methods in their mathematical formulation without allowing such a heuristic.

Fractal and *noise-based* methods (Sect. 4.1.1) are well suited for achieving a realistic appearance of the surfaces. In particular, fractals are ideal for expressing the self-similarity found in nature. In addition, noise can increase the random behaviour of real geological surfaces. Faults are difficult to represent with fractal or noise approaches, as they usually are represented by heightmaps that do not allow discontinuities. For the same reason, multi-z values can usually not be expressed with these techniques. Fractal and noise-based methods do not allow intuitive or local control of the surface, but it is easy to vary few parameters to obtain a different result. There is no need to store data, fractal and noise behaviour is represented by compact analytical formulas. On the other hand, processing requirements can be high, depending on the complexity of the formula describing the surface shape. Multiscale behaviour is present in fractals by their definition.

Erosion (Sect. 4.1.1) is a process that affects terrain by simulating weathering. Therefore it is very well suited for modelling a natural appearance of the top layer. Erosion is modelled as a flow process and, therefore, does not handle discontinuities well. When used with Smoothed Particle Hydrodynamics (SPH), erosion needs to incorporate some data interpolation method to fill the gaps. Erosion processes can result in carvings, and thereby multiple z values. The erosion process is hard to control. Essentially a simulation with given parameters is initiated and the user can either accept the results or modify input parameters for a more satisfactory result. Storage requirements are low, while the resulting model can be of arbitrary size. On the other hand, erosion is a dynamic process that requires processing resources for the simulation.

Exemplar-Based (Sect. 4.1.1) techniques can, to a certain degree represent faults, but not real discontinuities since the methods (mostly) use heightmaps. It can, in theory, synthesize terrains with abrupt changes if the exemplars contain steep cliffs. The method was not initially designed for data interpolation. However, exemplar synthesis often works with having a filter expanding the border of the so-far-made-texture by filling in with parts of exemplars that have similar neighbourhoods. Therefore interpolation can be made by starting with a texture having the interpolation values set and letting the rest be synthesized. Multi-z values are not supported for methods using a 2D heightmap. One could perform 3D texture synthesis, but this has not been explored for terrain generation. Classical exemplar-based methods offer no control at all, whereas more recent methods allow for a coarse input mesh [18] and can be guided by a user-sketched feature [63]. Brosz et al. [18] show how to use a base terrain and add details by texture synthesis. Storage requirements are quite high since many exemplars must be stored. Furthermore, creating the terrain is computationally expensive.

Radial-Basis Functions (RBFs) (Sect. 4.1.3) represent a variational interpolation technique that enables to fit/approximate an iso-surface to a given set of points and normals associated with these points. Here, the points can be given in arbitrary order, unlike splines which require a grid structure. An important feature, that might be seen as a drawback for geological models, is the C^n continuity of the resulting surface, which results in surfaces that are too smooth for geological structures. To produce highly realistic terrains one would need to specify a substantial number of points with varying normals to interpolate. On the other hand, the RBF method can easily fill the gaps in the surface model and model overhangs, which comes from the nature of the technique [121]. In practice, the specification of control points and normals can guide the appearance of the final surface. Moreover, modelling multiscale features is not directly supported due to the linear model composition. In order to visualize the final iso-surface, one needs to evaluate the function at the given point, which puts the computational burden on the surface generation step. Nevertheless, to store the implicit function, one only needs to specify the function evaluation process based on the given points and normals.

Splines are defined as parametric surfaces that, similarly to other interpolation techniques, produce the surface from a set of control points and the tangent or normal vectors associated with the points. Note that splines require an ordered list of points, which makes the modelling procedure somewhat tedious. Similarly to other interpolation-based surfaces, spline surfaces are continuous by their nature, which makes it hard to create discontinuous faults or realistic terrains. In comparison to RBFs, splines require a greater effort to change a surface model to fill the gaps or to produce overhangs. On the other hand, the parametric form facilitates the computation and visualization of the resulting surface. Multiscale representations are natively, similarly to RBFs, not supported by the spline model definition.

Kriging (Sect. 4.2.3) produces good terrain realism because the interpolated values are correct in a statistical sense. In addition, Kriging is ideal for filling gaps in the input dataset since the method is tailored for interpolating terrains using statistics.

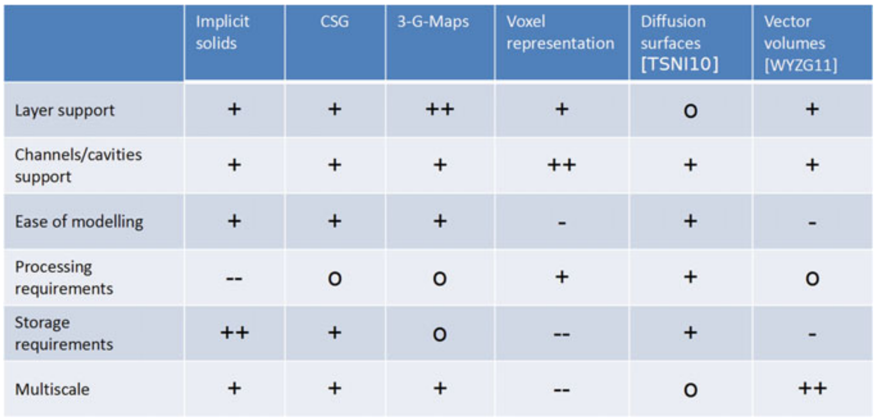
Discrete Smooth Interpolation (DSI) (Sect. 4.2.3) belongs to the family of interpolation techniques that compute the missing information (function values) on a given graph. As such, it provides a powerful framework for modelling specific features in geology. For instance, the information about discontinuities on a set of vertices can be specified by cutting out connected nodes or by adjusting their contributing weights [88]. Since the entire evaluation procedure that computes the unknown values at a graph node requires a minimization (iterative) algorithm, the processing complexity is very high compared to other interpolation techniques. However, DSI is efficient in iterative modelling when one needs to adjust an existing model. Essentially, to update the node values, only a few steps of the iterative minimization procedure are required. Since DSI evaluates values at nodes and not anywhere else, one stores only the graph nodes with their attributes and connectivity information. Due to this property, they do not support multiscale surface representations.

5.2 Solids

The output from the surface methods in Table 1 is input to the solid-creation methods in Table 2. Solids can then be faulted or carved after creation if the method supports this.

Implicit Solids (Sect. 4.2.6) do not offer any special classes of implicits aimed at geological models. Nevertheless, they offer a variety of techniques to represent such models. For instance, layers can be represented by a combination of implicit primitives or by the utilization of RBFs. Additionally, cavities can be realized by a subtraction operator applied to two or multiple compound objects [114]. The representation of implicits in multiscale models has also been successfully introduced [121]. Moreover, the interactive modelling capabilities become more and more prominent with the introduction of sketch-based interfaces [17, 122]. One of the major advantages of implicits, when representing even very complex objects, is their storage require-

Table 2 This table shows and compares the abilities of solid representation methods in terms of modelling geological features



	Implicit solids	CSG	3-G-Maps	Voxel representation	Diffusion surfaces [TSNI10]	Vector volumes [WYZG11]
Layer support	+	+	++	+	o	+
Channels/cavities support	+	+	+	++	+	+
Ease of modelling	+	+	+	-	+	-
Processing requirements	--	o	o	+	+	o
Storage requirements	++	+	o	--	+	-
Multiscale	+	+	+	--	o	++

ments, which is simply represented by the function evaluation process. On the other hand, to visualize the final solids, one needs to convert the implicit models into a set of triangles or adopt a direct ray-casting method.

Constructive Solid Geometry (CSG) can be used to compose a layer-cake model with simple layers in terms of shape definition. It is also adaptable to multiscale solutions and channels/cavities representation (e.g., employing the logical set operator minus). CSG is defined by simple primitives and set operators, but the global shape is difficult to intuitively control when the model starts to become complex ([22]). If the primitives are basic geometrical objects, CSG does not require much memory, and their logical interactions are relatively quick.

3-G-maps are the representation of choice for several geological solid modelling approaches [108, 111]. This is a boundary representation where the boundaries are typically triangulations. Details at different scales are supported and depend on the detail level of the underlying geometry. During modelling, the triangulations and their topology must be synchronously updated. There are no particular challenges with respect to the processing or storage of 3-G-maps.

A voxel representation is essentially a regular 3D discretized volume representation with given values in each sample. It can store layer information by simply tagging each voxel with a bit pattern defining a certain segmentation mask that defines the layer. Due to its expressiveness, it can also easily handle faults. In both horizons and faults, the final representation might need to be further processed in order to avoid visual artefacts arising from the space discretization. It can also support complex shapes, like cavities and channels. Voxel representations have been used for expressing channels [16]. This data structure does not offer a natural modelling approach that would be simple to use for a modeller. Therefore, it is often combined with other modelling representations. A voxel representation is very space demanding, but it

does not require a computational stage for evaluation, as it explicitly stores values in memory.
Diffusion Surfaces (Sect. 4.1.5) apply to layered models, but the approximation introduced by Takayama et al. [74] in their tool restricts each layer to have a rotational symmetry. Diffusion surfaces lack ease of modelling when dealing with multiscale models because a user has to define every boundary surface that delimits a piece of the solid model. Cavities are representable with diffusion surfaces, in particular if they have symmetries in shape. In producing the volumetric colour distributions, it is not necessary to perform precomputations, as opposed to Poisson approaches. Storage requirements are low because colours are interpolated locally at cross-sectional locations.
Vector Volumes (Sect. 4.1.5) is a volumetric representation of objects represented as a tree of signed distance functions (SDF trees). Thus, vector volumes combine the benefits of voxel and implicit models. Since each SDF tree contains information about the interior and the exterior of an object in a hierarchical fashion, vector volumes provide a powerful way to represent solids at different levels of detail, although the storage requirements can become very high. Moreover, such a representation requires a tedious way to interactively update the model. Although a volumetric object markup language is described [76], we do not consider it a straightforward solution. A volumetric object is usually achieved instead by developing and updating the model via different representations, e.g., boundary representation, and then performing the conversion into vector volumes. Due to the unique voxel and SDF identification when performing a volume ray-casting, this representation becomes efficient in direct visualization. Nevertheless, it is still required to evaluate the implicit function at each node in addition.

6 Challenges and Trends in Geological Modelling

Geoscience technology on closed model representations and model updating has not progressed at the same speed as in computer graphics. Better knowledge transfer between these groups could be advantageous. Caumon et al. [5] state that beginners with 3D modelling too often lose their critical sense about their work, mostly due to a combined effect of well-defined graphics and non-optimal human-machine communication. It is also important that a structural model can be updated when new data becomes available, or perturbed to account for structural uncertainties. In other words, with current modelling technology, uncertainty is difficult to express, and models are hard to update.

Researched literature from this domain emphasizes a strong need for modelling technology for communication and further analysis of the Earth's subsurface. While several matured methods are now in use by the domains of geology and geosciences, all tools require considerable effort to build structural models. Current tools focus on precise modelling in favour of rapid modelling. We believe that rapid modelling is the key for the ability of expertise exchange, especially in the early phases of the interpretation process.

As interesting research directions that would benefit from future attention of the graphics community, we point out two distinct ones: one research direction can be procedural geological modelling that takes advantage of sparsely defined acquired information about the subsurface. Ideally, an automated procedural method could be refined by the user through a series of sketches. Another research direction can be the consideration of temporal aspects in geology. Erosion has been investigated in this context, but geological processes are driven by many more phenomena than only surface erosion. Here, also the intended process can benefit from user input in the form of sketched information.

References

1. J. Pringle, J. Howell, D. Hodgetts, A. Westerman, D. Hodgson, Virtual outcrop models of petroleum reservoir analogues: a review of the current state-of-the-art. First Break **24**, 33 (2006)
2. S. Houlding, *3D Geoscience Modeling: Computer Techniques for Geological Characterization*. Springer (1994)
3. A.K. Turner, Challenges and trends for geological modelling and visualisation. Bull. Eng. Geol. Environ. **65**(2), 109–127 (2006)
4. A. Turner, C. Gable, A review of geological modeling, in *Three-Dimensional Geologic Mapping for Groundwater Applications. Minnesota Geological Survey Open-file Report*, 2007, pp. 07–4
5. G. Caumon, P. Collon-Drouaillet, C. Le Carlier, S. de Veslud, Viseur, J. Sausse, Surface-based 3D modeling of geological structures. Math. Geosci. **41**(8), 927–945 (2009)
6. F. Wellmann, G. Caumon, Chapter one—3-d structural geological models: concepts, methods, and uncertainties, in *Advances in Geophysics*, vol. 59 (Elsevier, 2018), pp. 1–121
7. J. Olsen, Realtime procedural terrain generation, tech. rep., Department of Mathematics and Computer Science (IMADA) University of Southern Denmark, 2004
8. A. Peytavie, E. Galin, J. Grosjean, S. Merillou, Procedural generation of rock piles using aperiodic tiling. Comput. Graph. Forum **28**(7), 1801–1809 (2009)
9. J. Schneider, T. Boldte, R. Westermann, Real-time editing, synthesis, and rendering of infinite landscapes on GPUs, in *Vision, Modeling, and Visualization 2006: Proceedings*, 22–24 Nov 2006, Aachen, Germany, 2006, p. 145
10. F. Belhadj, Terrain modeling: a constrained fractal model, in *Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, vol. 1, 2007, pp. 197–204
11. P. Krištof, B. Beneš, J. Křivánek, O. Št'ava, Hydraulic erosion using smoothed particle hydrodynamics. Comput. Graph. Forum **28**(2), 219–228 (2009)
12. H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, E. Galin, Feature based terrain generation using diffusion equation. Comput. Graph. Forum **29**(7), 2179–2186 (2010)
13. P. Prusinkiewicz, M. Hammel, C. Tn, A fractal model of mountains with rivers, *Panorama*, 1993, pp. 174–180
14. M. Hudak, R. Durikovic, Terrain models for mass movement erosion, in *Proceedings of EG-UK TPCG Conference*, 2011
15. J. Gain, P. Marais, W. Straß er, Terrain sketching, in *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, 2009, pp. 31–38

16. A. Peytavie, E. Galin, J. Grosjean, S. Merillou, Arches: a framework for modeling complex terrains. Comput. Graph. Forum **28**(2), 457–467 (2009)
17. E. Vital Brazil, I. Macêdo, M. costa Sousa, L. H. de Figueiredo, L. Velho, Sketching Variational Hermite-RBF Implicits, in *Proceedings of Sketch Based Interfaces and Modeling*, 2010, pp. 1–8
18. J. Brosz, F. Samavati, M. Sousa, Terrain synthesis by-example, *Advances in Computer Graphics and Computer Vision*, 2007, pp. 58–77
19. M. Natali, I. Viola, D. Patel, Rapid visualization of geological concepts, in *SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images)*, Aug 2012
20. E. M. Lidal, M. Natali, D. Patel, H. Hauser, I. Viola, Geological storytelling, *Computers & Graphics*, 2013
21. R. Groshong, *3-D Structural Geology: A Practical Guide to Surface and Subsurface Map Interpretation* (Springer, 1999)
22. A. Lemon, N. Jones, Building solid models from boreholes and user-defined cross-sections. Comput. Geosci. **29**(5), 547–555 (2003)
23. O. Kaufmann, T. Martin, 3D geological modelling from boreholes, cross-sections and geological maps, application over former natural gas storages in coal mines. Comput. Geosci. **34**(3), 278–290 (2008)
24. J. Ming, M. Pan, An improved horizons method for 3D geological modeling from boreholes, *2009 International Conference on Environmental Science and Information Application Technology*, 2009, pp. 369–374
25. D. Dhont, P. Luxey, J. Chorowicz, 3-D modeling of geologic maps from surface data. AAPG Bull. **89**(11), 1465–1474 (2005)
26. Z. Zhang, 3D Terrain reconstruction based on contours, in *Computer*, 2005, pp. 3–8
27. W. Baojun, S. Bin, S. Zhen, A simple approach to 3D geological modelling and visualization. Bull. Eng. Geol. Environ. **68**(4), 559–565 (2009)
28. E. De Kemp, K. Sprague, Interpretive tools for 3-D structural geological modeling part I: Bezier-based curves, ribbons and grip frames. GeoInformatica **7**(1), 55–71 (2003)
29. K. B. Sprague, E.A. de Kemp, Interpretive tools for 3-D structural geological modelling part II: surface design from sparse spatial data. GeoInformatica **9**(1), 5–32 (2005)
30. G. Caumon, F. Lepage, C.H. Sword, J.-L. Mallet, Building and editing a sealed geological model. Math. Geol. **36**(4), 405–424 (2004)
31. Q. Wu, H. Xu, An approach to computer modeling and visualization of geological faults in 3D. Comput. Geosci. **29**(4), 503–509 (2003)
32. T. Frank, Geological information retrieval using tetrahedral meshes, *gocad.org*, 2006, pp. 12–15
33. J. Plate, M. Tirtasana, R. Carmona, B. Fröhlich, Octreemizer: a hierarchical approach for interactive roaming through very large volumes, in *Proceedings of the Symposium on Data Visualisation*, VISSYM ’02, Eurographics Association, 2002, pp. 53–ff
34. E.M. Lidal, T. Langeland, C. Giertsen, J. Grimsgaard, R. Helland, A decade of increased oil recovery in virtual reality. IEEE Comput. Graphics Appl. **27**(6), 94–97 (2007)
35. C. Andersen, A.-J. van Wijngaarden, Interpretation of 4d avo inversion results using rock physics templates and virtual reality visualization, North Sea examples, *SEG Annual Meeting*, September 2007
36. D. Patel, C. Giertsen, J. Thurmond, J. Gjelberg, M.E. Gröller, The seismic analyzer: interpreting and illustrating 2D seismic data. IEEE Trans. Vis. Comput. Graph. **14**(6), 1571–8 (2008)
37. D. Patel, S. Bruckner, I. Viola, M.E. Gröller, Seismic volume visualization for horizon extraction. IEEE Pacific Visual. **2010**, 73–80 (2010)
38. D. Patel, C. Giertsen, J. Thurmond, M. Gröller, Illustrative rendering of seismic data, in *Proceedings of Vision Modeling and Visualization*, 2007, pp. 13–22
39. C. Wijns, Inverse modelling in geology by interactive evolutionary computation. J. Struct. Geol. **25**(10), 1615–1621 (2003)

40. A. Guillen, P. Calcagno, G. Courrioux, A. Joly, P. Ledru, Geological modelling from field data and geological knowledge Part II. Modelling validation using gravity and magnetic data inversion. Phys. Earth Planetary Interiors **171**(1–4), 158–169 (2008)
41. G. Caumon, Towards stochastic time-varying geological modeling. Math. Geosci. **42**(5), 1–25 (2010)
42. F. Musgrave, C. Kolb, R. Mace, The synthesis and rendering of eroded fractal terrains. ACM SIGGRAPH Comput. Graph. **23**(3), 41–50 (1989)
43. B.B. Mandelbrot, *The Fractal Geometry of Nature* (W. H. Freedman and Co., New York, 1982)
44. V. Sapozhnikov, V. Nikora, Simple computer model of a fractal river network with fractal individual watercourses. J. Phys. A: Math. Gen. **26**, L623 (1993)
45. S. Stachniak, W. Stuerzlinger, An algorithm for automated fractal terrain deformation, in *Computer Graphics and Artificial Intelligence*, 2005
46. J. Doran, I. Parberry, Controlled procedural terrain generation using software agents. IEEE Trans. Comput. Intell. AI Games **2**(2), 111–119 (2010)
47. P. Roudier, B. Peroche, M. Perrin, Landscapes synthesis achieved through erosion and deposition process simulation. Comput. Graph. Forum **12**(3), 375–383 (1993)
48. N. Chiba, K. Muraoka, An erosion model based on velocity fields for the visual simulation of mountain scenery. J. Visual. **194** (1998)
49. J. Dorsey, A. Edelman, H. Jensen, J. Legakis, H. Pedersen, Modeling and rendering of weathered stone, in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 225–234
50. B. Benes, R. Forsbach, Layered data representation for visual simulation of terrain erosion, in *Spring Conference on Computer Graphics (SCCG)*, 2001
51. B. Benes, R. Forsbach, Visual simulation of hydraulic erosion. J. WSCG (2002)
52. B. Benes, V. Tesinsky, J. Hornys, S.K. Bhatia, Hydraulic erosion. Comput. Animat. Virtual Worlds **17**(2), 99–108 (2006)
53. O. Stava, B. Benes, M. Brisbin, J. Krivanek, Interactive terrain modeling using hydraulic erosion, in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008, pp. 201–210
54. R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars. Mon. Not. R. Astron. Soc. **181**, 375 (1977)
55. L. Lucy, A numerical approach to the testing of the fission hypothesis. Astron. J. **82**, 1013 (1977)
56. G.J.P. de Carpentier, R. Bidarra, Interactive gpu-based procedural heightfield brushes, in *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG ’09, (New York, NY, USA), ACM, 2009, pp. 55–62
57. G. Cordonnier, M.-P. Cani, B. Benes, J. Braun, E. Galin, Sculpting mountains: interactive terrain modeling based on subsurface geology. IEEE Trans. Visual Comput. Graphics **24**, 1756–1769 (2018)
58. G. Cordonnier, J. Braun, M.-P. Cani, B. Benes, E. Galin, A. Peytavie, E. Guérin, Large scale terrain generation from tectonic uplift and fluvial erosion. Comput. Graph. Forum **35**, 165–175 (2016)
59. G. Cordonnier, E. Galin, J. Gain, B. Benes, E. Guérin, A. Peytavie, M.-P. Cani, Authoring landscapes by combining ecosystem and terrain erosion simulation. ACM Trans. Graph. **36**, 134 (2017). The paper was presented at Siggraph 2017
60. E. Galin, E. Guérin, A. Peytavie, G. Cordonnier, M.-P. Cani, B. Benes, J. Gain, A review of digital terrain modeling. Comput. Graph. Forum **38**(2) (2019)
61. J.M. Cohen, J.F. Hughes, R.C. Zeleznik, Harold: a world made of drawings, in *Proceedings of NPAR ’00*, 2000, pp. 83–90
62. N. Watanabe, T. Igarashi, A sketching interface for terrain modeling, in *ACM SIGGRAPH 2004 Posters*, 2004, p. 73
63. H. Zhou, J. Sun, G. Turk, J.M. Rehg, Terrain synthesis from digital elevation models. IEEE Trans. Visual Comput. Graphics **13**(4), 834–48 (2007)

64. M. Becher, M. Krone, G. Reina, T. Ertl, Feature-based volumetric terrain generation and decoration. IEEE Trans. Visual Comput. Graphics **25**, 1283–1296 (2019)
65. E. Guérin, J. Digne, E. Galin, A. Peytavie, C. Wolf, B. Benes, B. Martinez, Interactive example-based terrain authoring with conditional generative adversarial networks. ACM Trans. Graph. **36** (2017)
66. P. Isola, J. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks. CoRR abs/1611.07004 (2016)
67. T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, B. Catanzaro, High-resolution image synthesis and semantic manipulation with conditional gans, 2017
68. E. Guérin, J. Digne, E. Galin, A. Peytavie, Sparse representation of terrains for procedural modeling. Comput. Graph. Forum **35**, 177–187 (2016)
69. A. Bernhardt, A. Maximo, L. Velho, H. Hnaidi, M.-P. Cani, Real-time terrain modeling using CPU-GPU coupled computation, in *XXIV SIBGRAPI*, (Maceio, Brazil), Aug 2011
70. M. Natali, T.G. Klausen, D. Patel, Sketch-based modelling and visualization of geological deposition. Comput. Geosci. **67**, 40–48 (2014)
71. M. Natali, J. Parulek, D. Patel, Rapid modelling of interactive geological illustrations with faults and compaction, in *Proceedings of the 30th Spring Conference on Computer Graphics*, SCCG '14, ACM, 2014
72. R. Amorim, E.V. Brazil, F. Samavati, M.C. Sousa, 3d geological modeling using sketches and annotations from geologic maps, in SBIM '14, ACM, 2014, pp. 17–25
73. D. Lopes, D. Mendes, M. Sousa, J. Jorge, Expeditious illustration of layer-cake models on and above a tactile surface. Comput. Geosci. **90**, 02 (2016)
74. K. Takayama, O. Sorkine, A. Nealen, T. Igarashi, Volumetric modeling with diffusion surfaces, in *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, vol. 29, no. 6, 2010
75. A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, D. Salesin, Diffusion curves: a vector representation for smooth-shaded images. ACM Trans. Graph. **27** (2008)
76. L. Wang, Y. Yu, K. Zhou, B. Guo, Multiscale vector volumes. ACM Trans. Graph. **30**, 167:1–167:8 (2011)
77. SIS, Petrel seismic interpretation software, in *Schlumberger Information Solutions*. https://www.software.slb.com/products/petrel/petrel-geology-and-modeling, Accessed Mar 2021
78. B.J. Kadlec, H.M. Tufo, G.A. Dorn, Knowledge-assisted visualization and segmentation of geologic features. IEEE Comput. Graphics Appl. **30**, 30–39 (2010)
79. D. Parks, Freeform modeling of faulted surfaces in seismic images. SEG Tech. Prog. Exp. Abst. **28**(1), 2702–2706 (2009)
80. M. Botsch, L. Kobbelt, An intuitive framework for real-time freeform modeling, in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, 2004, pp. 630–634
81. R. Amorim, E.V. Brazil, D. Patel, M.C. Sousa, Sketch modeling of seismic horizons from uncertainty, in *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, 2012, pp. 1–10
82. S. Motta, A. Montenegro, M. Gattass, D. Roehl, A 3d sketch-based formulation to model salt bodies from seismic data. Comput. Geosci. **142** (2020)
83. A. Nealen, O. Sorkine, M. Alexa, D. Cohen-Or, A sketch-based interface for detail-preserving mesh editing, SIGGRAPH, (NY, USA), Association for Computing Machinery, New York, 2005
84. R. Liu, L. Shen, X. Chen, G. Ji, B. Zhao, C. Tan, M. Su, Sketch-based slice interpretative visualization for stratigraphic data. J. Imag. Sci. Technol. (2019)
85. E.M. Lidal, H. Hauser, I. Viola, Geological storytelling—graphically exploring and communicating geological sketches, in *Proceedings of Sketch-Based Interfaces and Modeling (SBIM 2012)*, 2012, pp. 11–20
86. M. Garcia, M.-P. Cani, R. Ronfard, C. Gout, C. Perrenoud, Automatic generation of geological stories from a single sketch, in *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering. Expressive '18*, 2018

87. G. Laurent, G. Caumon, M. Jessell, Interactive editing of 3d geological structures and tectonic history sketching via a rigid element method. Comput. Geosci. **74** (2015)
88. J.-L. Mallet, Discrete smooth interpolation. ACM Trans. Graph. **8**, 121–144 (1989)
89. J.-L. Mallet, Discrete smooth interpolation in geometric modelling. Comput. Aided Des. **24**(4), 178–191 (1992)
90. J. Mallet, Discrete modeling for natural objects. Math. Geol. **29**(2), 199–219 (1997)
91. R. Sibson, A brief description of natural neighbor interpolation, in *Interpreting multivariate Data*, 1981, pp. 21–26
92. J.-P. Chilès, P. Delfiner, *Geostatistics: Modeling Spatial Uncertainty* (Wiley, 1999)
93. T. Viard, G. Caumon, B. Lévy, Adjacent versus coincident representations of geospatial uncertainty: which promote better decisions? Comput. Geosci. (2010)
94. J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, T.R. Evans, Reconstruction and representation of 3d objects with radial basis functions, in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (New York, NY, USA), ACM, 2001, pp. 67–76
95. Leapfrog, Leapfrog radial basis function. https://www.seequent.com/comparing-leapfrog-radial-basis-function-and-kriging/, 2020. Accessed Aug 2021
96. I. Macêdo, J.A.P. Gois, L. Velho, Hermite radial basis functions implicits. Comput. Graph. Forum **30**(1), 27–42 (2011)
97. P. Calcagno, J. Chilès, G. Courrioux, A. Guillen, Geological modelling from field data and geological knowledge Part I. Modelling method coupling 3D potential-field interpolation and geological rules. Phys. Earth Planet. Int. **171**(1–4), 147–157 (2008)
98. P. McInerney, A. Goldberg, P. Calcagno, G. Courrioux, A. Guillen, R. Seikel, Improved 3D geology modelling using an implicit function interpolator and forward modelling of potential field data. Proc. Explor. **7**, 919–922 (2007)
99. B. Wu, T. Rahman, X.-C. Tai, Sparse-data based 3d surface reconstruction for cartoon and map, in *Imaging, Vision and Learning Based on Optimization and PDEs*, ed. by X.-C. Tai, E. Bae, M. Lysaker (Springer International Publishing, 2018)
100. M. Floater, Y. Halbwachs, O. Hjelle, M. Reimers, Omega: a cad-based approach to geological modelling.," GOCAD ENSG, Conference Proceedings, June 1998
101. N. Euler, C.H. Sword, J.-C. Dulac, A new tool to seal a 3d earth model: a cut with constraints, in *Proceedings of 68th Annual Meeting*, Society of Exploration Geophysicists, 1998, pp. 710–713
102. F. Aminzadeh, N. Burkhard, N. Nicoletis, F. Rocca, W. K, SEG-EAGE 3D modeling project, in *The Leading Edge*, 1994
103. C. De Boor, *A Practical Guide to Splines*. No. v. 27 in Applied Mathematical Sciences (Springer, 1978)
104. ArcGIS, Arcgis mapping and analytics platform, *ArcGIS Software*. https://www.esri.com/en-us/arcgis/about-arcgis/overview, Accessed Mar 2021
105. F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction* (Springer, New York, 1985)
106. Web3D, Vrml97 functional specification, *VRML Standard*. https://www.web3d.org/documents/specifications/14772/V2.0/index.html, Accessed Mar 2021
107. L. feng Zhu, Z. He, X. Pan, X. cai Wu, An approach to computer modeling of geological faults in 3d and an application. J. China Univ. Min. Technol. **16**(4), 461–465 (2006)
108. Y. Halbwachs, O. Hjelle, Generalized maps in geological modeling: object-oriented design of topological kernels, in *Advances in Software Tools for Scientific Computing* (Springer, Berlin, 2002)
109. P. Lienhardt, Topological models for boundary representation: a comparison with n-dimensional generalized maps. Comput. Aided Des. **23**, 59–82 (1991)
110. M. Apel, *A 3d geoscience information system framework.* PhD thesis, TU Freiberg, pp 16–31, 2004
111. S. Brandel, S. Schneider, M. Perrin, N. Guiard, J.-F. Rainaud, P. Lienhardt, Y. Bertrand, Automatic building of structured geological models, in *Proceedings of the ninth ACM Symposium on Solid Modeling and Applications*, SM '04, 2004, pp. 59–69

112. M. Perrin, B. Zhu, J.-F. Rainaud, S. Schneider, Knowledge-driven applications for geological modeling. J. Petrol. Sci. Eng. **47**, 89–104 (2005)
113. A.A. Pasko, V. Adzhiev, A. Sourin, V.V. Savchenko, Function representation in geometric modeling: concepts, implementation and applications. Vis. Comput. **11**(8), 429–446 (1995)
114. E. Kartasheva, V. Adzhiev, P. Comninos, O. Fryazinov, A. Pasko, Heterogeneous objects modelling and applications, in *An Implicit Complexes Framework for Heterogeneous Objects Modelling* (Springer, Berlin, 2008), pp. 1–41
115. M.N. Gamito, S.C. Maddock, Topological correction of hypertextured implicit surfaces for ray casting. Vis. Comput. **24**, 397–409 (2008)
116. P. Tahmasebi, *Multiple Point Statistics: A Review*. 06 2018
117. M.J. Abdollahifard, S. Ahmadi, Reconstruction of binary geological images using analytical edge and object models. Comput. Geoscie. **89**, 239–251 (2016)
118. F. Wellmann, M. Lindsay, J. Pohb, M. Jessell, Validating 3-d structural models with geological knowledge for improved uncertainty evaluations. Energy Procedia **59**, 374–381 (2014)
119. G. Mariethoz, S. Lefebvre, Bridges between multiple-point geostatistics and texture synthesis: review and guidelines for future research. Comput. Geosci. **66**, 66–80 (2014)
120. T.-F. Zhang, P. Tilke, E. Dupont, L.-C. Zhu, L. Liang, W. Bailey, Generating geologically realistic 3d reservoir facies models using deep learning of sedimentary architecture with generative adversarial networks. Pet. Sci. **16**(3) (2019)
121. R. Schmidt, B. Wyvill, M.C. Sousa, J.A. Jorge, Shapeshop: sketch-based solid modeling with blobtrees, in *ACM SIGGRAPH 2006 Courses, SIGGRAPH '06* (NY, USA), ACM, New York, 2006
122. O. Karpenko, J.F. Hughes, R. Raskar, Free-form sketching with variational implicit surfaces. Comput. Graph. Forum **21**(3), 585–594 (2002)

Real-Time Algorithms for Visualizing and Processing Seismic and Reservoir Data

Daniel Patel, Thomas Höllt, and Markus Hadwiger

Abstract In this chapter, we present research that deals with the rendering and processing of seismic volumes and unstructured reservoir grids. We focus on approaches that facilitate interactive workflows, i.e., real-time rendering of seismic data, as well as real-time extraction of seismic structures. This is enabled by the parallel processing power of graphics processing units (GPUs). The chapter consists of three parts. The first part introduces basic and established techniques for seismic volume and reservoir visualization. The second part discusses more experimental and advanced methods that are not (yet) common among geoscientists. The third part covers real-time methods for extracting objects and structures from seismic data.

1 Visualization of Seismic and Reservoir Data

The earth subsurface consists of material layers with distinct material density and porosity characteristics. Subsurface volumetric data is obtained by sending sound waves into the ground and processing the resulting echoes. The interfaces between materials with different acoustic impedances create reflections which are present in the seismic volume. These interfaces are called seismic *horizons*. In seismic reflection data, the horizons are indicated by bands typically representing high and low reflection values as shown on the crosslines and inlines in Fig. 1, and are central to interpretation. Other important structures are *fault surfaces* which represent breaks in the subsurface layering, where horizons on one side of the fault are moved laterally

D. Patel (✉)
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

T. Höllt
Delft University of Technology, Delft, The Netherlands
e-mail: t.hollt-1@tudelft.nl

M. Hadwiger
King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
e-mail: markus.hadwiger@kaust.edu.sa

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_2

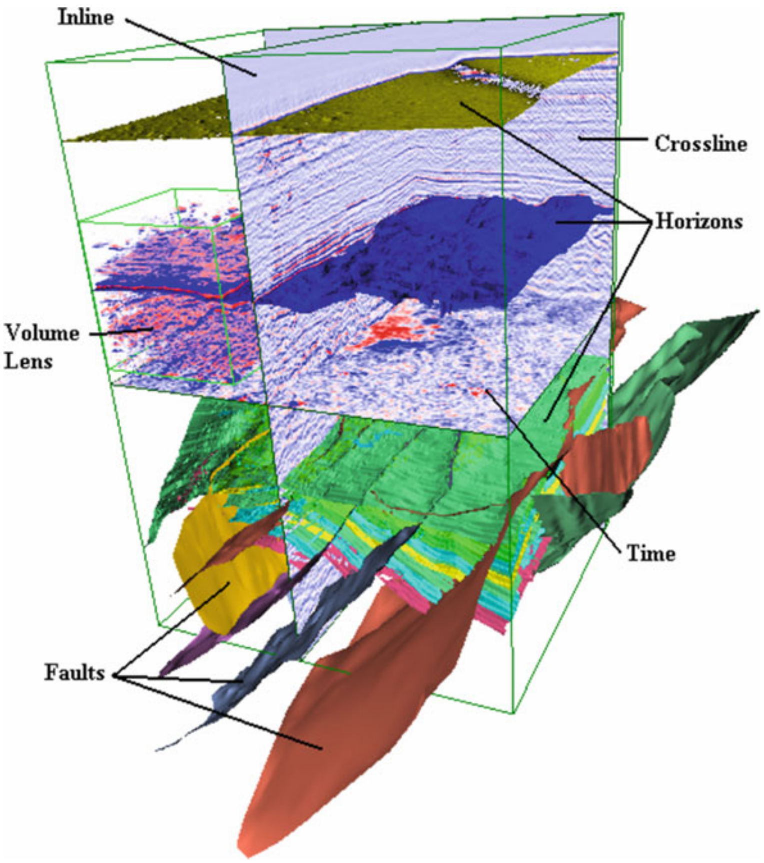


Fig. 1 Central objects in a seismic dataset. For browsing the data, inline, crossline and time slices are used in addition to a volume lens. The horizon and fault surfaces are interpreted structures. Plate et al. [1]

relative to the horizons on the other side. Horizons and faults must be interpreted from the seismic data either manually or semi-automatically. Several objects that are important in oil exploration are shown in Fig. 1. Slices through the seismic data are called *inlines*, *crosslines* and *time(lines),* according to their orientation with respect to the side surfaces of the volume bounding box. Here, they show the reflection values using a red to white to blue color scheme. Translucent volume rendering confined to a small area is shown to the left (volume lens) as well as interpreted horizon and fault surfaces.

The notions of a *trace* and of a *seismic attribute* are central to seismic data. A trace comprises an array of consecutive samples along the depth axis, and when plotted on a graph it will resemble a waveform according to the reflected sound as a function of time (and not absolute physical depth). A seismic attribute is a new volume derived from the original reflection volume using some mathematical calculations. Different

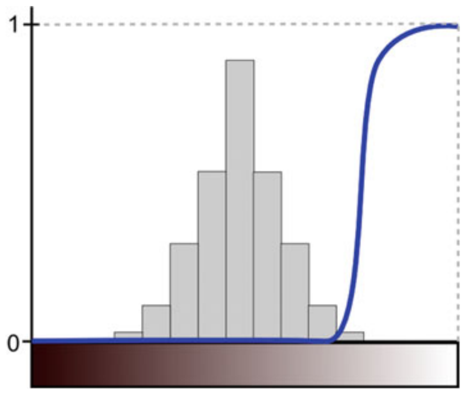


Fig. 2 Example of a 1D transfer function. The opacity curve is shown in blue, defined to highlight only areas of high values. The color assignment is shown at the bottom along the *x*-axis and a histogram of all volume values is shown in gray

seismic attributes are used during interpretation for highlighting certain aspects of the data such as the probability of a voxel being on a fault.

Before a seismic volume can be rendered in 3D using a volume rendering approach, each sample in the volume must be mapped to optical properties such as color and the degree of opacity/transparency. Instead of performing this for each spatial position individually, it is most commonly achieved by using a transfer function which maps reflection strength in the data to color and opacity. The most common transfer function is mathematically defined as a mapping from a 1D scalar value to color and opacity: *transferfunction(value)* = *(red, green, blue, opacity)*. An example of a transfer function can be seen in Fig. 2. The *x*-axis represents the reflection strength, from minimum at the left to maximum at the right, and the blue curve defines the opacity for each value, where $y = 0$ defines full transparency and $y = 1$ defines full opacity. A user-defined color map along the *x*-axis defines the color. In addition, a histogram can be displayed behind the transfer function to communicate how many samples reside in the specific intervals. Editing the transfer function curve in intervals where the histogram count is zero, will have no effect on the volume rendering. Seismic data typically has a symmetric bell-shaped histogram which peaks in the center.

1.1 Fundamentals of Seismic Volume Visualization and Interpretation

Kidd [2] presents the basics of seismic volume visualization. He argues that volume visualization can be used to directly provide an overview of the data without first performing time-consuming interpretation and triangulation of surface structures. He defines a zone system as an aid to systematically manipulate the color and opacity settings of the transfer function for exploring the seismic volume. The zone system divides the data range into six intervals. Each interval is assigned a pre-defined color designed to improve the color contrast between important structures. Instructions are given for how to manipulate the transparency differently in each of the intervals. Kidd recommends using a black background to more easily identify colors in the volume. To focus the volume rendering on important areas, rendering can be limited by a user-defined axis-aligned bounding box, by a pre-defined vertical distance around an interpreted horizon surface, or by the layer between two interpreted horizon surfaces. Chopra et al. in two works [3, 4] describe and compare the use of the RGB, HSV and HSL color spaces for displaying timelines and inlines. The perceptual effect of using different color tables is also demonstrated.

Gao [5] presents the workflow for visualizing and interpreting seismic volume data as consisting of four phases. First, data is collected and conditioned. Then seismic structure is explored by slicing the data along inlines, crosslines and timelines. This results in a conceptual geologic model. In the third phase, the geologic model is analyzed with respect to hydrocarbon sources, fluid migration pathways, reservoirs, trapping geometry and sealing capacity. This phase consists of volume rendering techniques as described in the previous paragraph. In addition, seismic attribute volumes can be calculated from the original volume for highlighting specific structures. Growing can be performed in which structures in the data are computationally identified by classifying each volume sample as either being part of or not being part of a specific structure. When areas of potential hydrocarbon contents have been identified, one enters the final phase where the well-bore path is designed. In this phase, factors such as well length, well curvature and hardness of penetrated material are taken into account.

The Master thesis by Li [6] and the book chapter by Ma and Rokne [7] also give good introductory overviews of seismic volume rendering and interpretation.

A common way to perform volume rendering is to send a ray from each screen pixel through the volume and summing up the light contribution along the way as the ray passes through volume samples until the ray exits the volume [8]. For each sample, the color and opacity of the sample is found by looking up in the transfer function based on the sample intensity. The pixel is then assigned the accumulated color contributions from each sample, see Fig. 3.

1.2 Visualizing Large Volumes

The oil and gas industry acquires large amounts of volumetric seismic data for subsurface exploration. Real-time visualization and exploration of volumetric data has been made possible by using graphics processing units (GPUs) which are able to render data residing in local GPU memory very fast. However, GPU memory is of limited size and at the same time seismic survey technology produces volumes of increasing surface coverage, depth and resolution. The result is that the data of a full seismic survey often does not fit into GPU memory, or even into main memory, and can therefore not be rendered in real-time. Possible solutions for this are to show only parts of the volume at full resolution, to show the full volume at a down-sampled resolution,

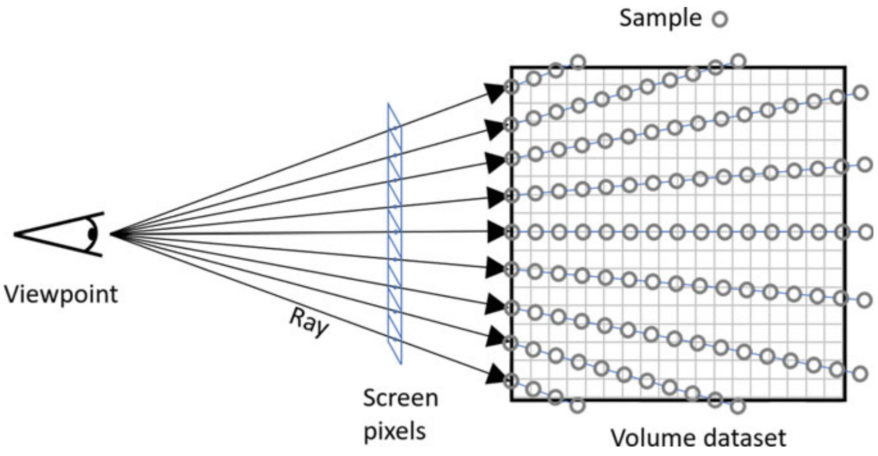


Fig. 3 Ray casting: A common volume-rendering approach. Rays are sent from the center of each pixel relative to the viewpoint and sample the volume at regular intervals until the volume is exited. The square represents the volume data, and the grid represents the individual samples in the volume

or a balance between these extremes. Plate et al. [9] present a balanced approach, allowing real-time rendering of freely oriented slices intersecting large volumes in combination with volume rendering of all or selected portions of the data. This is made possible by combining subsampling with fast and smart management of data transfers from the hard disk into main memory and further into GPU texture memory according to which parts of the data are to be visualized. Visualization methods that work for data sizes larger than CPU or GPU memory are called out-of-core methods.

The building block of this data structure is called a brick which is of a fixed memory size, for instance $64 \times 64 \times 64$ samples, and represents a cube-shaped subset of the volume data. The optimal brick size is hardware dependent and is typically identified by trying out different sizes and measuring the speed of rendering. Bricks are organized in a tree structure called an octree, where each tree node points to one brick and to eight child nodes. The root node points to a brick representing the coarsest possible sampling of the full volume. The bricks in the eight child nodes each cover one of the eight octants of the parent (see Fig. 4). Each of these bricks represents a finer sampling of the volume data for its subregion. The subdivision continues down to the leaf nodes. Each leaf node represents the full resolution of a small part of the volume. In practice, to increase speed by reusing calculations, the tree structure is created bottom-up starting with the leaf nodes. Eight neighboring bricks are filtered and downsampled into a single brick of the next coarser level until only a single brick remains at the top of the octree. The octree pointer structure itself fits in memory and has, for each node, a pointer to the hard disk or memory where the brick data resides. The bricks are stored in a breadth-first order on disk to increase the number of cache hits.

Since only a limited amount of the large seismic dataset on disk fits into main memory, and only a subset of this data again fits into GPU memory, a smart scheme

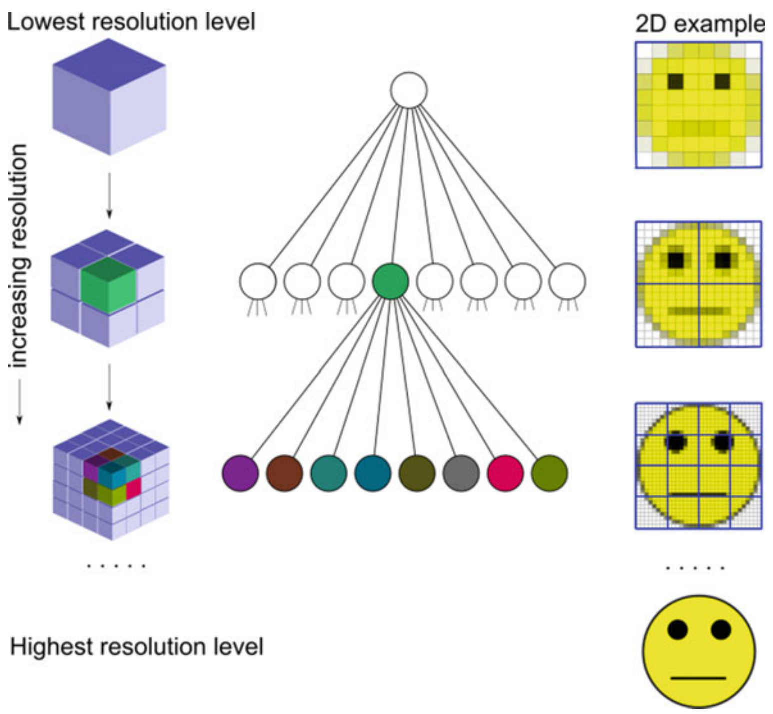


Fig. 4 Octree bricking scheme used for out-of-core rendering of seismic data. Left: The spatial coverage of each brick over the volume, for different levels. Middle: The octree pointer structure. Colored nodes point to the corresponding colored brick to the left. Right: A 2D example with 2D bricks outlined in blue, each having a resolution of 16 × 16 samples

must be applied for making sure that the correct data is in GPU memory at any time during rendering. The data to upload from disk to main memory is based on which part of the data the user is looking at. The user can define a bounding box in which volume rendering should take place or a seismic cross-section that should be textured. We will call this the region of interest (ROI). To identify the data that must be uploaded for the visualization, first the octree's leaf nodes covering the ROI are found and all their parent nodes are identified. These nodes are then traversed and loaded from disk into memory level by level, starting at the lowest resolution level (top of Fig. 4) until the memory is full. This guarantees that all data is loaded for the ROI by first retrieving a subsampled version covering the ROI and then gradually refining the data until memory is used up. During data refinement of one level, blocks closest to the viewpoint are refined first. If there is more available memory after all nodes in all levels for the ROI have been loaded, additional bricks in the immediate neighborhood around the ROI are loaded. Then the neighborhood is expanded until the available memory is filled up. This scheme increases the possibility that data already reside in memory when the user moves the ROI by a small distance. The same data loading approach takes place between main memory and GPU memory.

Whenever the ROI changes, it is first checked if nodes for the new region are already loaded. When old bricks in memory need to be replaced, the bricks that have not been accessed for the longest amount of time are overwritten first. With this approach, the GPU memory will quickly contain enough data for covering the area of interest, at increasing resolution, with a prioritized refinement close to the viewpoint.

On the other hand, an example of a simple storage scheme of volume data would serialize the regular 3D volume array as a sequence of xy-slices along the z-axis where each xy-slice is subdivided into strips parallel to the x-axis, from bottom to top. This scheme has two disadvantages that bricking solves. Firstly, since the data is not subsampled, not all data can be visualized at the same time. Secondly, loading only parts of the data corresponding to a ROI takes a lot of time since it requires nonlinear hard disk access as neighboring samples on disk are not necessarily neighbors in 3D space. In contrast, all samples in a brick reside in a local 3D neighborhood. In later work [10], the octree and bricking approach is extended to handle multiple arbitrarily overlapping volumes, possibly of different resolutions. Here, each volume is represented by an individual octree structure. The authors propose slice-based volume rendering for these data. However, since not all data fits into GPU memory, it is not possible to simply render slices from back to front that fully cover the area of interest at maximum resolution. Instead, in several passes, different parts of the volumes must be rendered back to front, which are then assembled in the right order depending on the area of interest, the viewpoint, the octree and the geometry of the volume-overlaps. Zhou and Hansen [11] present an approach which tackles both large and multivariate seismic data. They use parallel coordinate plots for defining a multiattribute transfer function.

For further information on rendering of large volumes in general see the state-of-the-art report by Beyer et al. [12]. In this work they present other hierarchical data structures that have advantages over the octree bricking scheme described in Fig. 4 and suggest additional ways to accelerate the ray casting.

1.3 Visualizing Isosurfaces Using Pre-integration

Pre-integrated volume rendering [13] is a method that works especially well for visualizing thin structures in volumetric data. As such it is well suited for, and has been applied to, volume rendering of seismic data [14, 15]. With standard volume rendering, small value intervals mapped to high opacity in the transfer function might be missed. This is exemplified in Fig. 5. The figure shows samples along a viewing ray into the volume with step size *d*. The values of the samples are shown as heights on the vertical axis. These samples are mapped to opacity using a transfer function, shown rotated 90° counterclockwise in blue. When the transfer function has a small interval of opacity falling in between two samples, the isosurface will be missed. However, had the sample distance been smaller so that sampling would also be performed in the gray areas, the peak would be sampled, and the isosurface would be rendered. Pre-integration solves this problem without the need for using a

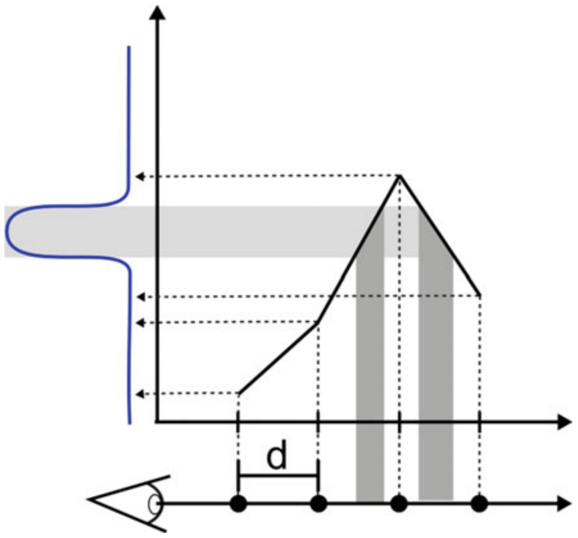


Fig. 5 The lookup from the sample values, along a viewing ray, to opacity in the transfer function (blue curve) is shown with the stippled arrows. A thin opacity spike in the transfer function (grey part of the blue curve) is missed due to coarse sampling. This problem is alleviated using pre-integration

time-consuming smaller sample distance. Pre-integration instead precalculates a 2D lookup table based on the transfer function. When two consecutive sample values are looked up in the table, the correct integrated color contribution is returned.

1.4 Visualizing Horizons

A straightforward way to display height fields (also called heightmaps) describing horizons is to render triangles. A height field is a 2D array of height values. For large meshes, triangles may end up having sizes smaller than a screen pixel. In such cases it can be faster to render the height fields using ray casting. Then the speed of rendering depends on the number of pixels (image-order algorithm) and not the number of triangles (object-order algorithm). For each pixel, a ray is sent out and intersections with the triangles must be found. For this approach to be efficient, it is important to have fast intersection testing. Lux and Frohlich [16] describe a fast approach for ray casting of height fields. Similar to the octree structure described earlier for subdividing large volumes into smaller parts, they use a quad structure for subdividing a height field. By subdividing the height field into a quadtree that stores minimum and maximum height for each node, one can check if a ray's minimum–maximum height interval is overlapping the minimum–maximum height interval of a region in the height field. If there is no overlap, then there will be no intersection with any of the triangles of the subregion that the node covers. If there is overlap, then the quadtree is traversed in depth until possible intersection is found. The quadtree data structure residing on the GPU is updated depending on view-point so that out-of-core rendering of large height fields is supported. Their method supports multiple heightfields and facilitates the integration with volume ray casting.

Graciano et al. [17] present another ray casting method suitable for geological models consisting of multiple layers. The paper also contains a good overview of ray casting methods of heightfields. Their method does not support out-of-core rendering. Instead, they present a way to compress the data in such a way that it can be rendered directly. They compare their methods with other compact representations of spatial data.

There are various methods for coloring the horizon surfaces. Each surface can be given a single individual color as shown in Fig. 1. It is also common to map the horizons varying height to a rainbow colormap. Carlson and Peloso [18] map the properties of the traces (i.e., the waveforms) around surfaces, such as horizons, to hue, saturation, value and opacity, and use transfer functions for changing the mapping ranges of each of the four measures. They also show how to perform a combined volume visualization of four different attribute volumes by mapping each to hue, saturation, value and opacity respectively.

1.5 Shadows and Shading

Shadows and shading are important visual cues for understanding shape and spatial relations. In this context we refer to shading as the coloring or darkening on an object's surface due to the angle of incoming light, and by shadow we mean darkening due to occlusion of light by other objects. As shading is essentially only dependent on the orientation of the surface and the incident light direction, it can be computed quickly and has been employed in real-time computer graphics for a long time. Shadows, however, require a global calculation which takes into account all objects between the light source and the point of interest and is thus more challenging to calculate in real-time.

Shading has been used in novel ways to represent seismic horizons, for instance by illuminating the surface with different colors from different angles (Fig. 8). Basic illumination for volume rendering is often based on surface shading techniques such as Phong shading [19]. Such techniques rely on the calculation of surface parameters, in particular surface normals for calculating the shading information. Phong shading [19] calculates shading at any point on a surface according to two angles; the angle between the light source and the surface normal, and the angle between the viewer and the surface normal. This calculation is straight-forward for surface geometry, however, for volumetric data there is no explicit surface information. While different ways of normal calculations have been proposed for seismic volumes, they generally do not provide very good quality due to the noisy and high frequency nature of seismic data. For seismic volumes, the volumetric gradient has often been used to define the surface normal, which is still the standard approach employed by commercial tools. However, using the gradient is not optimal for visualizing horizons. If one considers a horizon as the surface that intersects the voxels that are peak amplitudes in their traces (irrespective of the actual strength of the peak), then the volumetric gradient will not be a good approximation of the normal since it is based on the strength. This observation is made by Silva [20]. He shows that the gradient of the so called instantaneous phase attribute yields a better normal candidate since it is normal to the peak amplitudes. He compares the renderings of volumes shaded with standard gradients (Fig. 6 right) with volumes shaded with the improved gradients (Fig. 6 left) which are of higher quality.

Shaded relief maps have a long tradition in cartography for communicating the shape of terrains by shading the hills and valleys according to a simulated light source. Barnes [21] presents the use of shaded relief for shading horizontal slices through seismic data (Fig. 7). As the slice is completely flat, the surface normals are instead taken from the dip and azimuth seismic attribute. This attribute represents the angle of intersecting structures in a more accurate way than just using the volumetric gradients. The reflection strength is mapped to a black-to-white color scale which is modulated by the shading. The result is that the flat slice looks curved due to the shading. Also, the degree of shading is amplified by the reflection strength, thereby emphasizing strong structures. However, the original seismic value is no longer displayed.

For extracted polygonal surfaces, such as explicit horizon geometry, the Phong shading model works well. Liro et al. [22] use Phong shading with three light sources

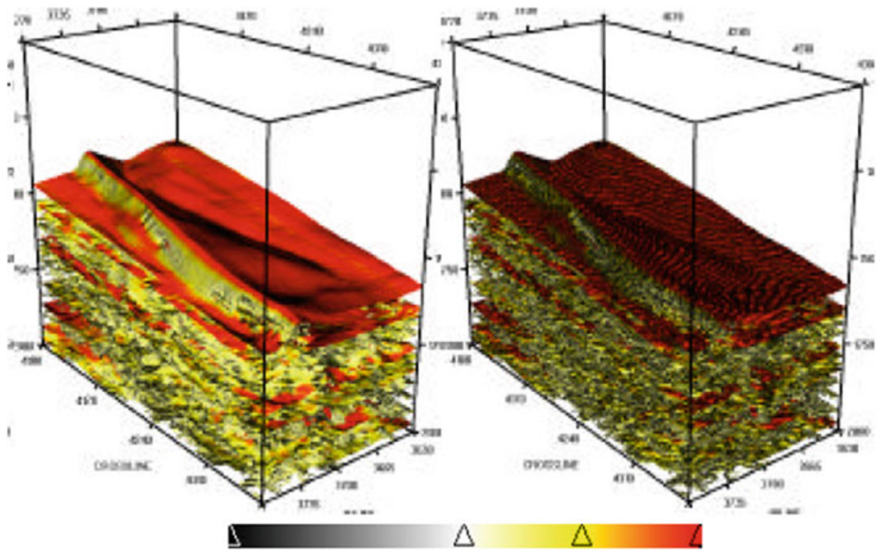


Fig. 6 Volume rendering of seismic data. Using the instantaneous phase attribute volume for calculating the gradient (left) gives better surface shading than using standard gradient calculation (right). Silva [20]

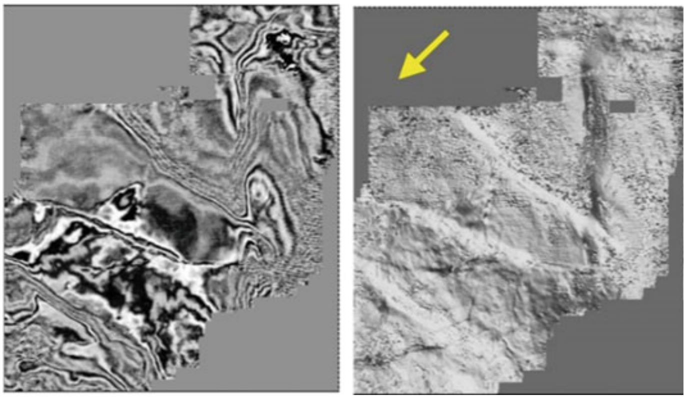


Fig. 7 The left image shows an unshaded horizontal slice through a seismic volume. The right image shows the same horizontal slice Phong shaded using normals from the corresponding dip/azimuth slice. The result is a terrain- looking rendering that conveys the dip and azimuth structure through shading. The light has a 30 degrees elevation and its direction is indicated by the yellow arrow. Barnes [21]

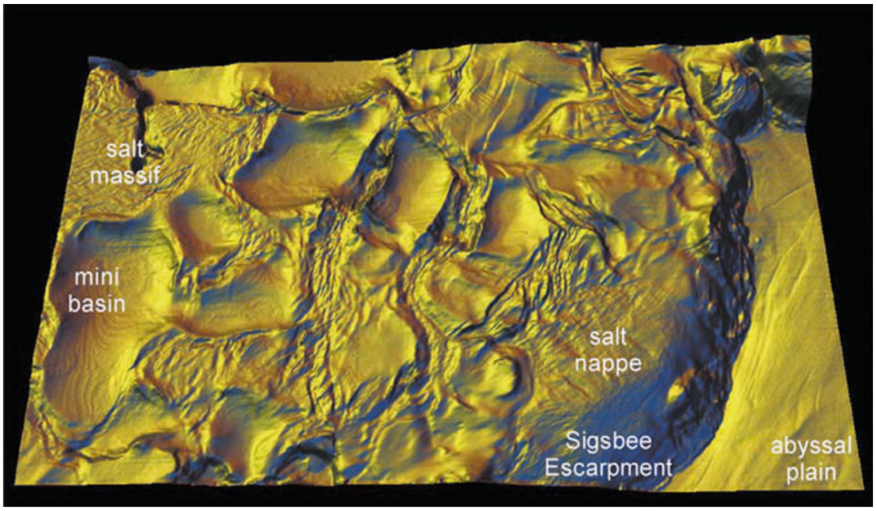


Fig. 8 Phong shading with three light sources for giving a better understanding of the surface shape. Yellow light is positioned overhead, blue light in the north, and orange light in the northwest. Liro et al. [22]

from different angles having different colors to emphasize the curvature and shape of the horizon surface (Fig. 8).

Patel et al. [23] introduced real-time shadows in rendering of volumetric seismic data and horizons. The renderings give better depth understanding and reduce noise whereas Phong shading in noisy areas often results in disturbing specular reflections

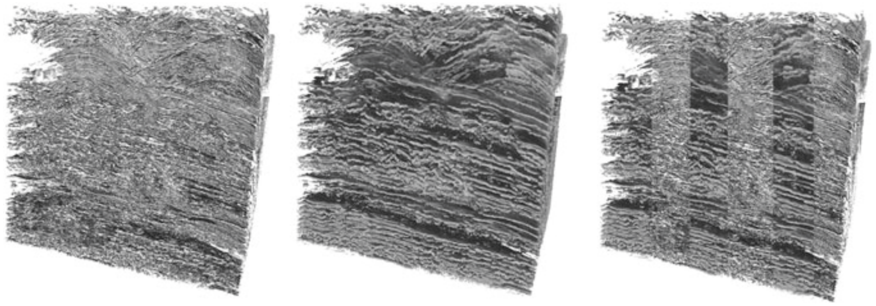
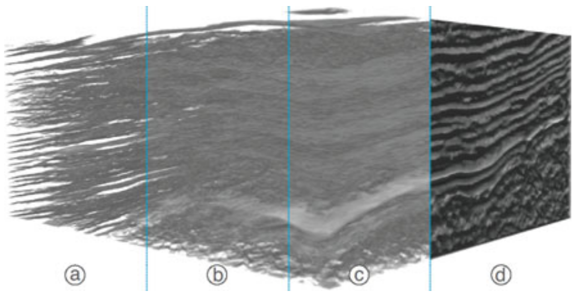


Fig. 9 The left image shows Phong shading. The middle image shows rendering with shadows cast in the view direction from an area light source. The right image shows alternating stripes from the two left renderings for comparison. Phong shading gives a flat appearance and reduces insight into the volume. Patel et al. [23]

from small spherical isosurfaces. Figure 9 shows a comparison of the two methods with shading to the left and shadows in the middle picture. A transfer function is used where low values are set to transparent.

Höllt et al. [24] introduce a shading technique specifically targeted to seismic applications. They analyze the local waveform along the trace of each sample and use the result to modify opacity and shading. Figure 10 shows a comparison of the approach with different kernel sizes for the waveform analysis (a, b, c) and basic Phong shading (d). On the one hand, it can be seen that the approach allows enhancing the visualization of horizon structures, without requiring transfer function adjustments, by modulating the opacity based on the waveform. Since the waveform is used to classify a sample, horizons for which the strength of the reflection changes throughout the volume can be identified more reliably, compared to amplitude-based classification. Secondly, it also enables surface-like shading similar to Silva et al. [20]. Since only the gradient along the trace is used for shading, false features, visible in Fig. 10d, caused by the often broken 3D gradients used for Phong shading do not occur in their approach. Note that the rendering in Fig. 10d is actually a completely flat, opaque slice, and the apparent structures are caused only by the shading based on the not necessarily correct 3D gradients.

Fig. 10 Volume rendering with waveform-based shading. **a**–**c** Show different kernel sizes for the waveform analysis (3 × 1 × 1, 5 × 1 × 1 and 7 × 1 × 1 respectively). **d** Shows standard Phong shading (with the same transfer function for comparison). Höllt et al. [24]



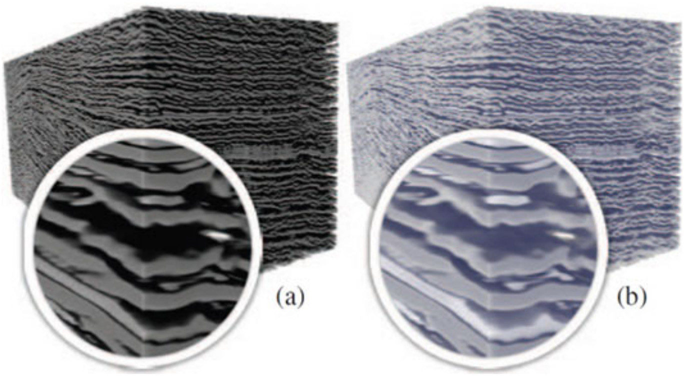


Fig. 11 In the left image, black shadows are cast. Overdarkening hides information such as the shape of the surfaces in the zoom-in. In the right image, the shadowiness is mapped towards a blue color to preserve surface shape cues. Soltészová et al. [25]

Soltészová et al. [25] conducted a user study to see how shadows and shading relate to curvature and depth perception. Shadows convey information, but they also darken and thus hide data (Fig. 11a). They propose a solution to the darkening by representing the shadow with luminance, creating blue shadows so that shading details are still preserved (Fig. 11b).

1.6 Reservoir Visualization

To predict content and movement of liquid or gas in the subsurface, a model of the subsurface is subdivided into cells. Most reservoirs use hexahedral cells which are aligned with features such as layers and faults. The cells, that constitute a 3D irregular grid are populated with initial values such as porosities and liquid/gas content. A simulator then calculates at discrete timesteps, scalar properties for each cell such as an updated liquid/gas content, temperature, pressure, as well as vector properties such as flow direction. One way to inspect the results of a simulation is to visualize the reservoir. A standard rendering approach is to draw the boundary surface of each cell in the reservoir and color the cell according to a selected attribute and a color table. The outline of each cell can be rendered to differentiate individual cells of same color. An example is shown in Fig. 12. Cells have an (i, j, k) index according to an internal parameterization of the reservoir and it is typically possible to select index intervals for showing parts of a reservoir. Also, it is useful to be able to filter cells based on attribute values to e.g., only show cells with water content less than a certain value. As with seismic data, out-of-core methods are necessary to visualize large reservoirs. The chapter titled "Visualization of Large Scale Reservoir Models" describes such an out-of-core method and gives more information about reservoir grids.

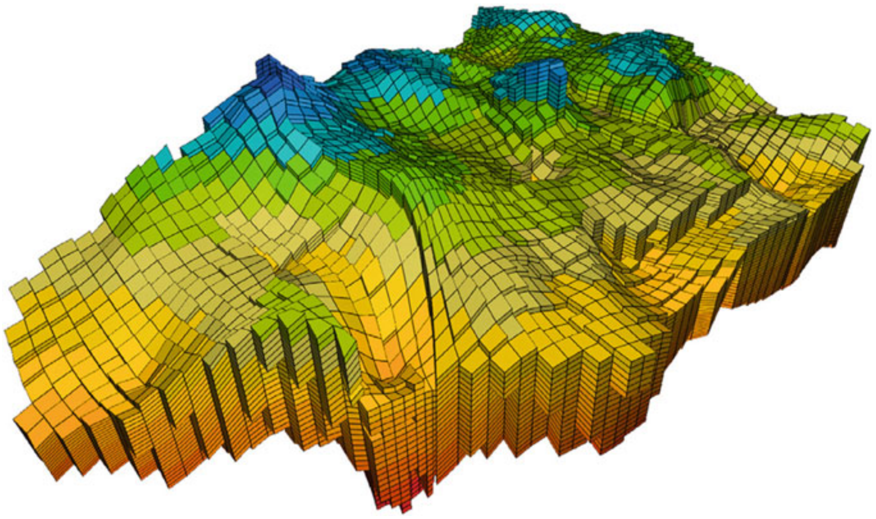


Fig. 12 Example of a reservoir visualization. Individual hexahedral cells are colored according to a color table that maps a specific attribute to a color. Cells are outlined with black lines. Image is from [26]

2 Advanced Visualization of Seismic and Reservoir Data

While the previous chapter described techniques that are more or less industry standard in geoscience, with focus on showing the data as it is without e.g. using textures and deformation, this chapter presents promising and possibly more experimental and advanced methods that are not widespread in industry.

2.1 Illustrative Rendering

Geologists and illustrators of geology use a standardized language of symbols, textures and colors for expressing their knowledge. Textures can be used to express rock and layer types (Fig. 13), whereas colors can be used to express geologic time (Fig. 14). Examples of illustrations employing these techniques can be seen in geologic textbooks [27] and is exemplified in Fig. 15.

Patel et al. [30] present a method for performing illustrative rendering of a seismic volume in which horizon and fault surfaces have been interpreted. The method renders the data using geologic textures on the side surfaces of a cutout to communicate layer types (see Fig. 15). The illustrative rendering style can be mixed with ground truth rendering of measured data (Fig. 16).

Patel et al. [31] present a method where the user assigns 2D textures along a vertical line (white line in Fig. 17) on a seismic slice. The textures are then automatically

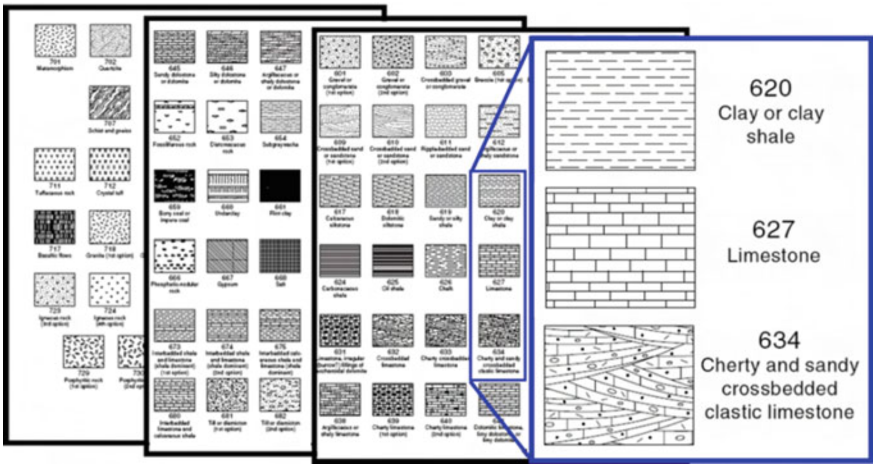


Fig. 13 Standardized textures used in geology to express layer content and structure [28]

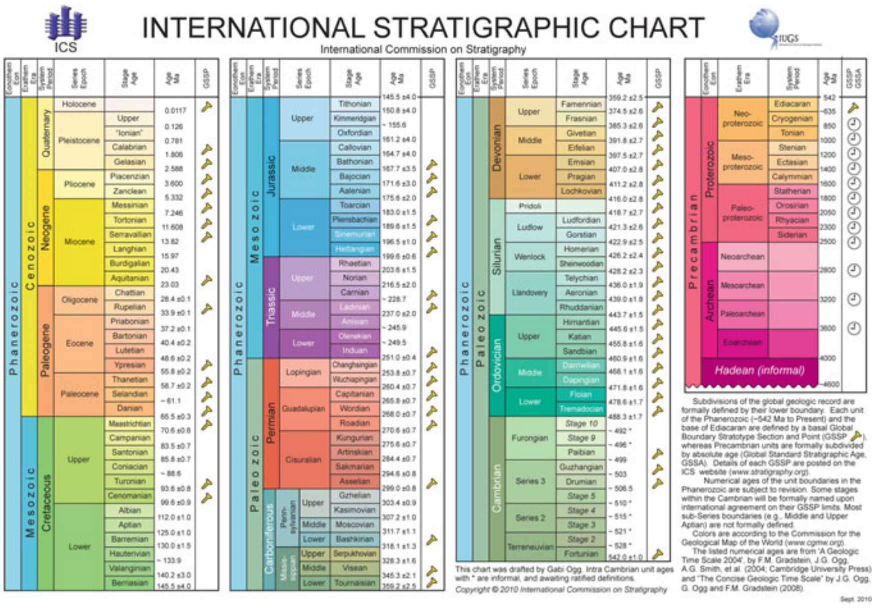


Fig. 14 Color scheme used to express geologic time [29]

extruded horizontally such that they follow the layer structures and deform in a visually pleasing way. The texturing is achieved by parameterizing the seismic line in a preprocessing step based on extracted horizon curves. This is a type of multiattribute visualization where the texture type communicates a category such as layer type and the texture deformation communicates the orientation of the underlying data. The

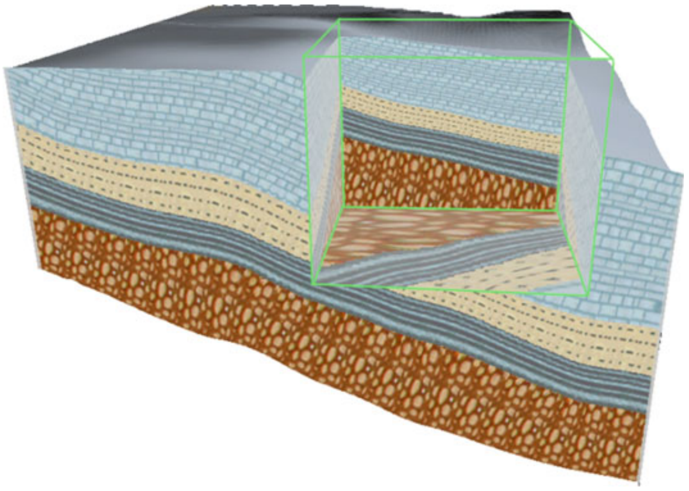


Fig. 15 Computer-generated illustration based on a seismic volume. Patel et al. [30]

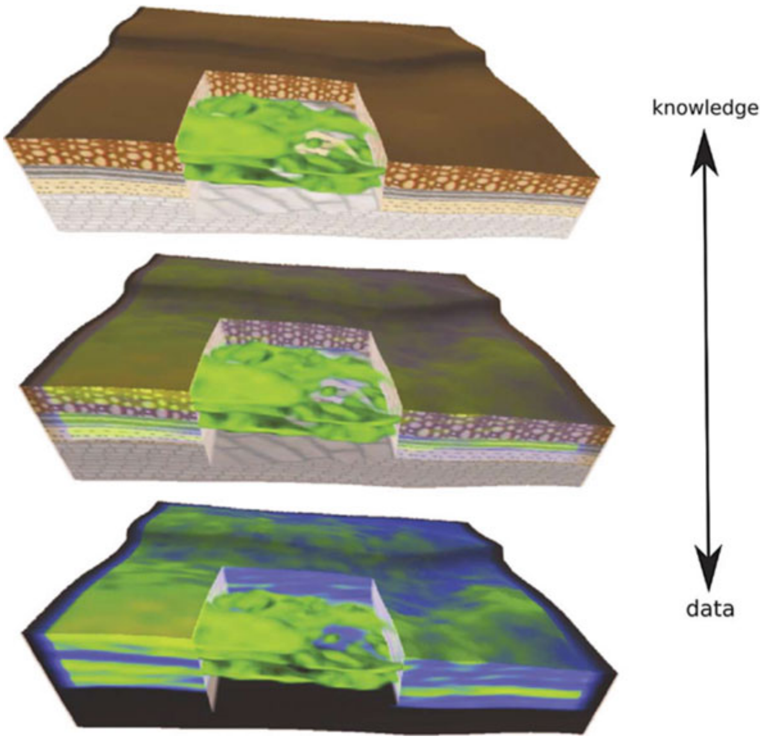


Fig. 16 Smooth transition between quantitative visualization (bottom) and illustrative visualization (top) based on an interpretation. Volume rendering of the seismic data is performed in the cutout. Patel et al. [30]

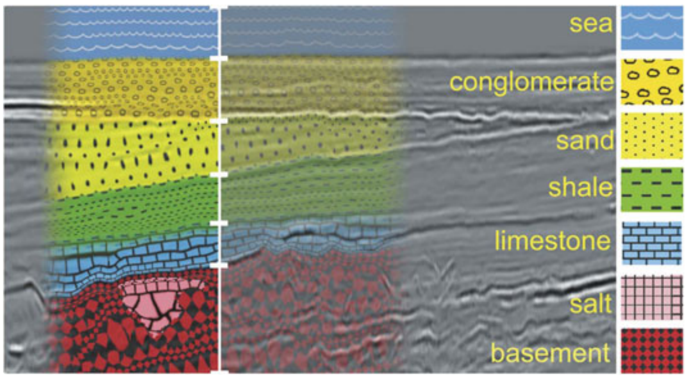


Fig. 17 Quick sketching of hypotheses by texturing 2D seismic slices with computer-assistance. Patel et al. [31]

user can also draw textures freely as demonstrated in the patch of salt texture in Fig. 17. This enables quick sketching of hypotheses. The work also shows how to simultaneously visualize several seismic attributes by assigning each attribute to an illustrative texture and overlaying the textures on a seismic slice.

An attempt to generalize 2D seismic textures into semi-transparent 3D textures for representing layer types and deformations at the same time, in 3D has also been performed [32]. See Fig. 18 for examples.

Cutting away parts of the seismic by moving a cut plane through the volume is a common way to explore seismic data. A drawback of using a planar clipping plane is that one indiscriminately cuts through features and breaks up their 3D appearance. For better results, illustrators, when making cuts through volumetric data, consider the structures in the vicinity of the cut and make sure that smaller structures near the clipping plane are conserved in the image and not cut. Inspired by this approach,

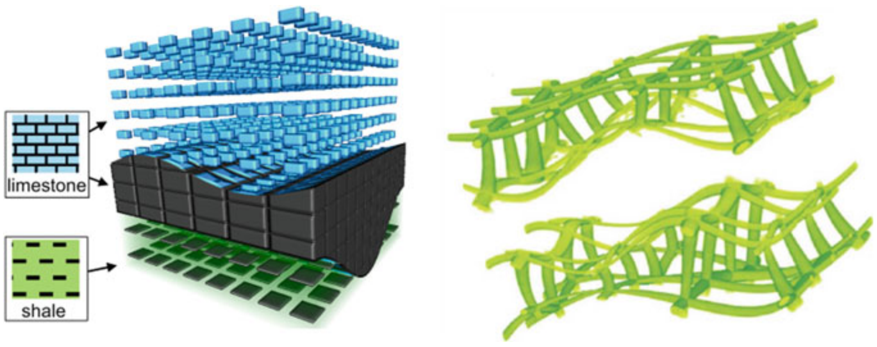


Fig. 18 3D textures. The left image shows two 2D textures and their corresponding 3D versions. The right image communicates the deformation of two layers through deformed 3D textures. Patel et al. [32]

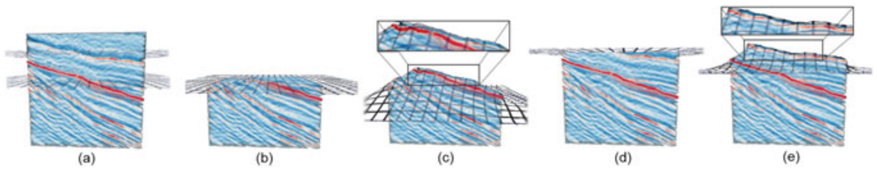


Fig. 19 Adaptive cutting planes. **a** Shows the side of a seismic volume with two cut planes indicated. **b**, **d** Show standard cutting of the volume by the respective cut planes. **c**, **e** Show adaptive cutting of the respective cut planes. One can see in the zoom-ins how the cut membranes adapt to the strong reflectors in the seismic shown in red. Birkeland et al. [33]

Birkeland et al. [33] have created a cut plane that adapts to the volumetric data by acting like a deformable membrane. This works particularly well when cutting through seismic data along the depth axis. The membrane will then adapt itself and snap onto and cling to strong reflectors as the membrane is pushed downwards. This is illustrated in Fig. 19.

2.2 *Exploded Views*

Occlusion is a common problem when visualizing three-dimensional objects. Opacity adjustments are commonly used to reveal hidden structures in volume visualization. However, for very dense data, such as seismic data, this is often not sufficient. Cutaways or clipping can solve this problem, but at the cost of losing contextual information. Illustrators often employ exploded views to reveal hidden structures while retaining context information. An exploded view is basically created by cutting the data into several parts, which are then shown at displaced locations in order to reveal the structures of interest.

Bruckner and Gröller [34] introduce exploded views (Fig. 20) for rendering volume data. They employ a flexible framework that uses a force-based model where the volume is divided into several parts, which are controlled by a number of forces and constraints. The presented framework can handle multiple arbitrarily placed

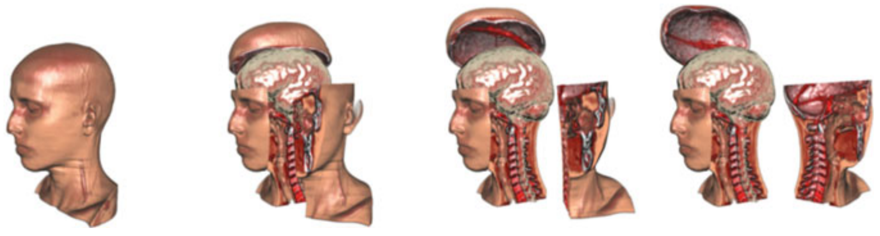


Fig. 20 Interactive exploded-view illustration of a human head with increasing degrees-of-explosion. Two hinge joints are used to constrain part movement. Bruckner and Gröller [34]

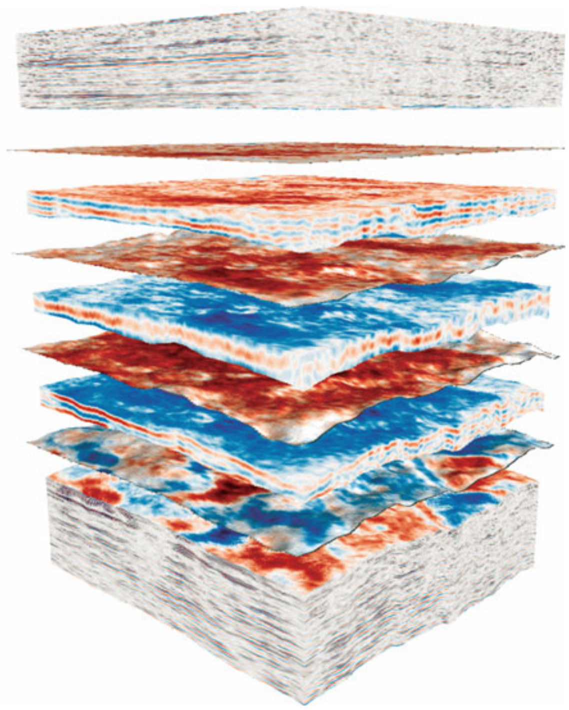


Fig. 21 Rendering of a dataset using deformation and exploded views along the depth axis. Höllt et al. [24]

cutting planes. To achieve this flexibility, each of the defined parts of the volume is rendered separately in multiple passes and the resulting subimages need to be composited.

Höllt et al. [24] present a simpler approach that specifically targets seismic data. By constraining the exploded views to translations along the depth axis of the volume (Fig. 21), they achieve a technique that, while less flexible, is very simple to implement and allows rendering the complete dataset in a single rendering pass. While their technique is limited with regard to the explosion axis, it allows arbitrary height fields as cutting geometry, which makes it possible to cut the volume open along horizon surfaces.

2.3 Annotations and Visibility of Features

Focus and context visualization is a technique for visualizing an object of primary interest with high detail while visualizing surrounding information more sparsely which will be acting as a context. Ropinski et al. [35] use the concept of focus and context by employing different transfer functions for volume rendering different regions of seismic data as shown in Fig. 22. Their system also allows the creation of bookmarks in the data for later lookups.

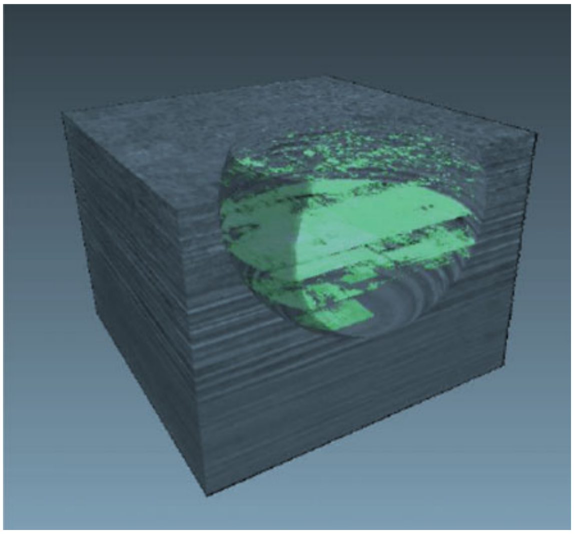


Fig. 22 Focus and context visualization. Within a spherical area, strong seismic reflectors are visualized in green by using a different transfer function from the surrounding one. Ropinski et al. [35]

Interpreted seismic data can contain several structures such as horizons, faults, wells and segmented geological bodies. Annotations could be applied to better visualize and tag these structures. Bruckner and Gröller [36] describe a methodology for tagging structures with text and lines which could be advantageous to apply to seismic data. They create annotations that automatically position themselves so that they do not overlap each other or the image. Lines do not cross each other, and annotations are positioned close to the relevant structure and reposition themselves smoothly during rotation. In addition, structures of interest can be visualized separately and unoccluded as seen at the top in Fig. 23.

Viola et al. [37] present a technique that guarantees visibility of selected structures regardless of the viewpoint. This can be used for seeing certain structures in relation to their neighborhood or in relation to other structures (Fig. 24). Such guaranteed visibility could be useful in seismic data since it is of such a dense nature.

Some of the ideas by Viola et al. have been explored in geology by Lidal et al. [38] through cutaway visualizations of 3D geological models as shown in Fig. 25 left. In [39] cutaways were used for rendering sub-terrain tubular networks, see Fig. 25 right. The work focuses on urban infrastructure just below the surface such as sewer systems, underground/metro structures and tunnels. However, such an approach can be useful for revealing well pipes, including their interior content, that are just below occluding horizons or other surfaces such as reservoir boundaries.

Takahashi et al. [40] present an approach for automatically identifying optimal viewpoints in volume visualization based on important features. Such a methodology

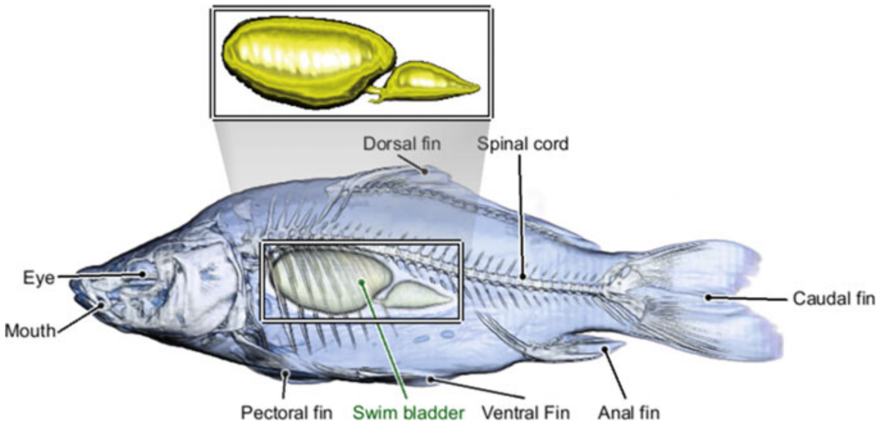


Fig. 23 Automatic placement of annotations. Bruckner and Gröller [36]

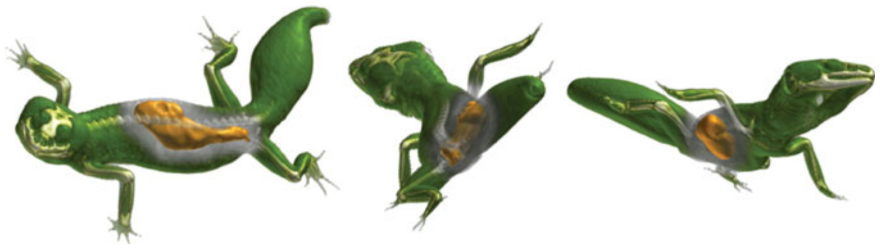


Fig. 24 Guaranteed visibility of important structures. The brown structure inside the Gecko is visible from any of the three viewpoints. Viola et al. [37]

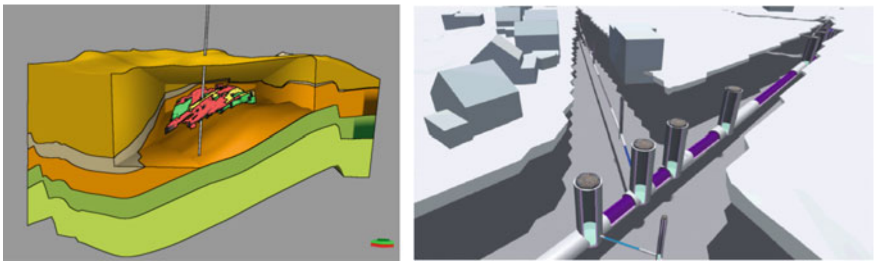


Fig. 25 Left: A cutaway in a 3D geologic model of horizons and a well, revealing segmented structures inside [38]. Right Cutaway view of an underground sewer network [39]

could be interesting to investigate with respect to seismic data, perhaps in combination with cutaway views. With such an approach, the system could automatically suggest a view direction on the seismic volume, and cutaways could be applied for revealing structures of importance.

2.4 Advanced Reservoir Visualization

While rendering reservoir cells directly as described earlier is useful for getting quantitative information on the values of each cell, volume rendering can be used to show more accurately the position of values by using interpolation. This is useful for continuous attributes such as water content or temperature. With volume rendering it is possible to e.g. show multiple semitransparent isosurfaces representing specific water densities. Such volume rendering of reservoir grids is explored by Campagnolo and Celes in [26].

Early work on multiattribute visualization of reservoir data was performed by Giertsen [41]. He presents multi-attribute volume visualization of oil pressure, oil saturation, reservoir porosity and reservoir permeability in an oil reservoir by mapping selected intervals from each attribute to distinct colors. The goal was to make important features in the data visually discernible for the observer. Rocha et al. [42] also present various illustrative techniques for visualizing multiattribute reservoir data. Each cell in their reservoir grid contains the three attributes: rock type, porosity and permeability. By using both 2D textures and 3D textures (glyphs) they attempt to display all attributes simultaneously such that they are perceptually discernable.

Toledo and Celes [43] present a GPU implementation for visualizing the direction of flow on user-defined surfaces. The flow is based on the movement of fluid or gas such as hydrocarbons or water and is generated by a reservoir simulation, see Fig. 26, left. Flow directions are displayed on a surface using a technique called line integral convolution (LIC) [44]. To display the flow vectors which are 3D, on a 2D surface, the tangential component of the vector is used for generating a LIC texture, while the magnitude of the normal component is shown using a colormap.

Franceschin et al. [45] present visualization of colored 3D streamlines based on a reservoir simulation. The streamlines follow the direction of flow from points of water injection in injector wells to points of oil extraction in production wells. The color represents the oil saturation, see Fig. 26, right.

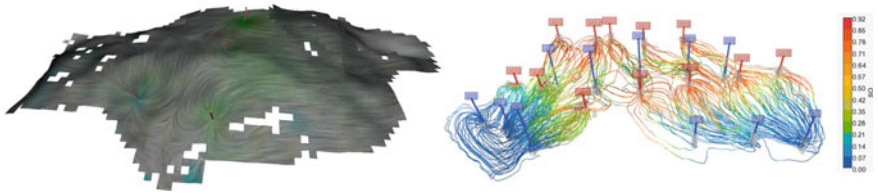


Fig. 26 Visualizations of flow fields in reservoirs. Left: Line integral convolution texture and color for displaying flow, shown on a reservoir surface. Position of wells is also displayed [43]. Right: Streamlines between wells colored based on degree of oil saturation [45]

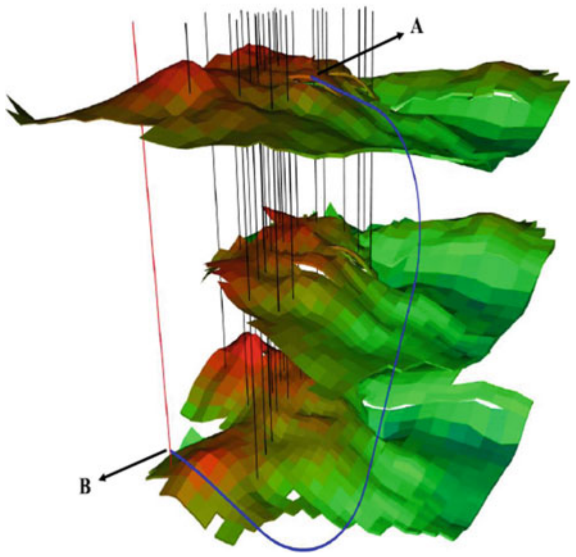


Fig. 27 For efficient browsing of the model, a collision-free navigation route is automatically created (shown in blue), from well A at the top surface to well B shown in red at the bottom surface. Calomeni et al. [46]

Calomeni et al. [46] present a solution for effective navigation in a subsurface model consisting of layers constructed by hexahedral cells as well as curves representing drilled wells (see Fig. 27). Their approach allows the user to navigate smoothly without collisions along the layers and between layers for inspecting the cells, with particular emphasis around wells. Navigation is semi-automated as manual navigation can result in spatial disorientation and object collision. Particularly in immersive environments, collisions make the user's experience uncomfortable. To address this problem, a global roadmap of the environment is pre-computed, using a randomized motion planning algorithm. This results in a spatial graph describing collision-free routes. At runtime, graph searching is performed using the A* algorithm to compute shortest routes between points. The system offers a fully automatic navigation mode, in which the user selects a target and the system computes a smooth, collision-free path throughout the environment.

In [47] Höllt et al. present a way to visualize the result of ensemble simulations. By running a reservoir simulation multiple times with varying simulation parameters, it is possible to achieve a better overview of the outcome space. The multiple simulation results produce a probability distribution for each cell, instead of a single final result. This enables one to understand the sensitivity to initial conditions and the uncertainty of the end result. In Fig. 28 the calculated water saturation for a single layer of a reservoir is shown. Pins indicate injector/extractor wells. To show the distribution of values produced from the multiple simulation runs, the mean value of the simulation runs is mapped to a white (low mean saturation) to purple (high mean saturation) color gradient. Variance is shown as a texture transitioning from no noise through low frequency noise to high frequency noise to indicate increasing variance.

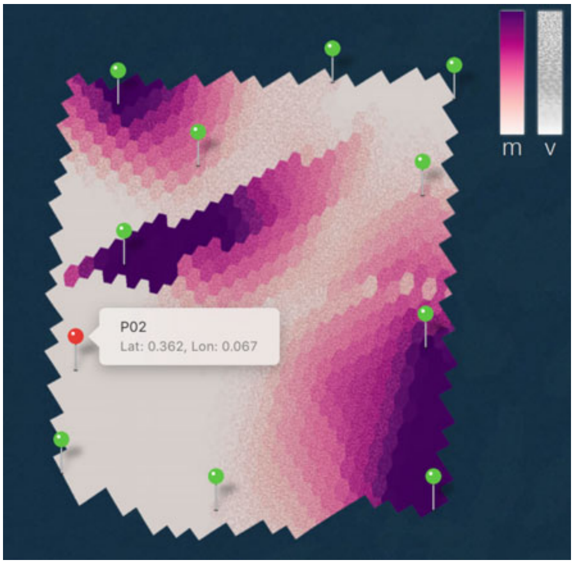


Fig. 28 3D view of a single layer of the reservoir, showing the reservoir geometry in combination with water saturation mean and variance. Placement of wells is also shown

2.5 Privacy Preserving Volume Rendering

Visualization of large datasets such as seismic volumes requires powerful computers which must be regularly maintained and upgraded. For many organizations, it could be beneficial to outsource the rendering to cloud services. However geoscientific volumetric data for hydrocarbon and mineral exploration can be considered highly confidential due to their potential value. Privacy can currently only be achieved by storing and processing the datasets locally.

The work by Mazza et al. [48] can be considered a first step towards secure volume rendering on insecure servers. The method encrypts the volume and performs volume rendering directly on the encrypted data. The method is limited to X-ray style volume rendering where the average intensity value along each ray is found (Fig. 29). The method supports assigning colors to different intensities but does not support varying opacities and shading. An overview of the state of the art on Privacy-Preserving Data Visualization has been made by Bhattacharjee et al. [49].

3 Processing of Seismic Data

In this section we present real-time methods for extracting objects and structures from seismic data.

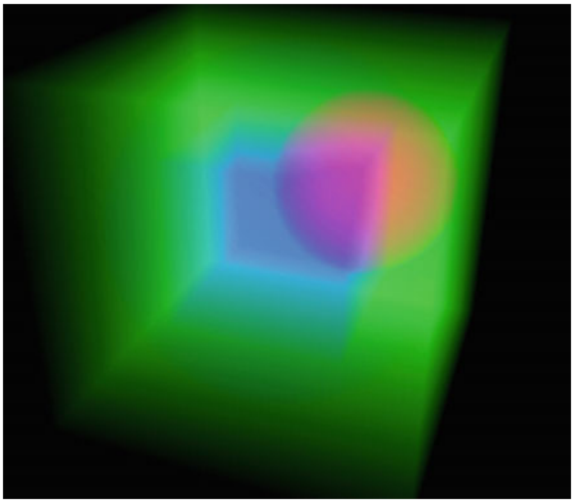


Fig. 29 Volume rendering on encrypted data using the method by Mazza et al. [48]. The image shows a rendering of a volume consisting of a box (green) with a smaller box (blue) and a sphere (red) inside, each having different intensity values. A transfer function is used to assign colors to the intensities. The rendering is limited to showing the average value along each ray (X-Ray rendering)

3.1 Frequency Decomposition of Seismic Data

When considering the traces of seismic data as signals, they are typically in the range of 10–90 Hz. Filtering out and analyzing frequencies from seismic data is useful for identifying structures. This can be performed in a preprocessing step, for instance in 1 Hz increments. Then individual single-frequency volumes, with scalar values describing the strength of that particular frequency in that position, can be assessed separately. Alternatively, three user-selected frequencies can be combined into an RGB cube which is then analyzed. The combination is done by mapping each frequency onto a separate color channel as shown in Fig. 30.

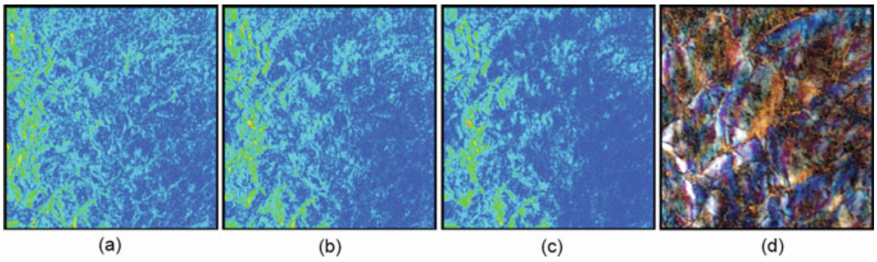


Fig. 30 Combining the response for 24 Hz (**a**), 32 Hz (**b**), and 40 Hz (**c**), of a horizontal slice to the red, green and blue color channel respectively (**d**). Henderson et al. [50]

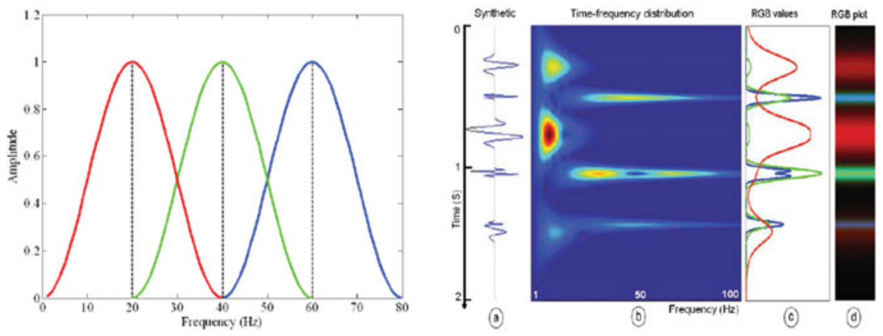


Fig. 31 The left image shows the mapping of frequencies to colors. Frequencies around 20 Hz, 40 Hz, and 60 Hz are respectively mapped to red, green, and blue, with a fall-off according to the distance from the respective frequency. The right image shows the signal along a trace, it's frequency distribution, it's mapping to the RGB curves, and the resulting color. Liu and Marfurt [51]

Methods to combine all frequencies into a single volume have also been attempted. One method [51] maps low, medium and high frequencies to the red, green and blue color space respectively, creating an RGB volume (see Fig. 31).

Another method identifies the strongest frequency response for each voxel as well as the strength of the response. This reduces the frequency curve to two scalar values. Liu and Marfurt [52] perform a similar dimensional reduction, but instead of using the strength of the strongest frequency, they calculate the difference between the peak frequency strength and the average strength of all frequencies for that position. This expresses how much the peak differs from the rest of the frequency curve. Thus, it communicates whether the frequency response is rather flat, has a small peak, or if the peak is narrow and strong (see Fig. 32). Guo et al. [53] perform a principal component analysis on all frequencies to reduce the number of dimensions. They find the 3 major component axes and map the corresponding values to the RGB color space. However, the three components no longer represent actual frequencies.

Vucini et al. [54] present a GPU-accelerated frequency transfer function that is able to interactively assign color and opacity to voxels in the seismic volume data based on the frequencies of each voxel's trace. This can be used to detect structures based on thickness. Thanks to NVDIA's CUDA framework for GPU calculations, including methods for calculating the Fast Fourier Transform (FFT) [55], real time spectral decomposition is not hard to implement. Chen et al. [56] visualize wave signals from earthquake simulations (see Fig. 33). They present several novel and expressive methods which might be inspirational for visualizing the waveforms in the traces found in seismic data. In addition, frequency decomposition, rendering of unstructured grids, the use of pre-integration, motion blur, and real-time animation is presented.

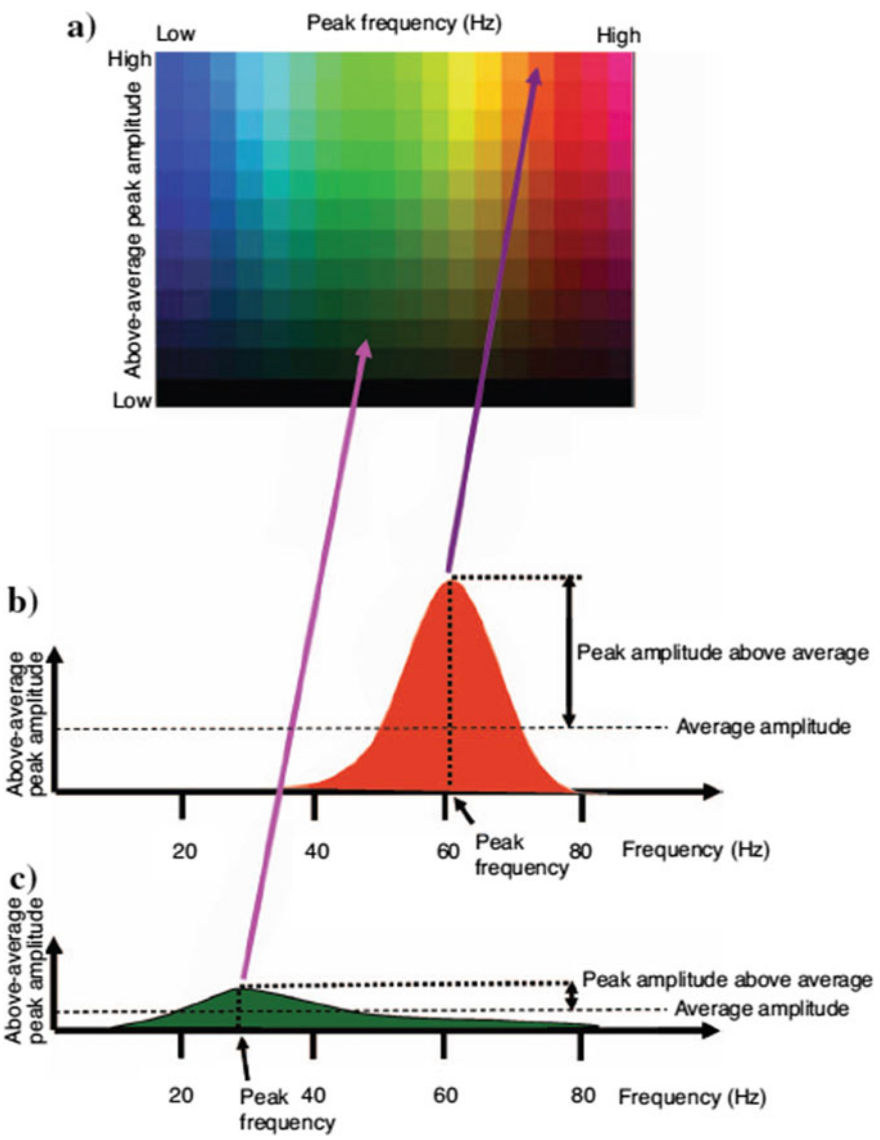


Fig. 32 Decomposing a frequency curve into two expressive values that are mapped to a color using hue and value. Liu and Marfurt [52]

3.2 *Computer-Aided Object Extraction*

The most straightforward method to browse the seismic volume for structures is to interactively alter the transfer function, making certain reflection intensity intervals transparent and others opaque. Setting only the highest or lowest values to opaque

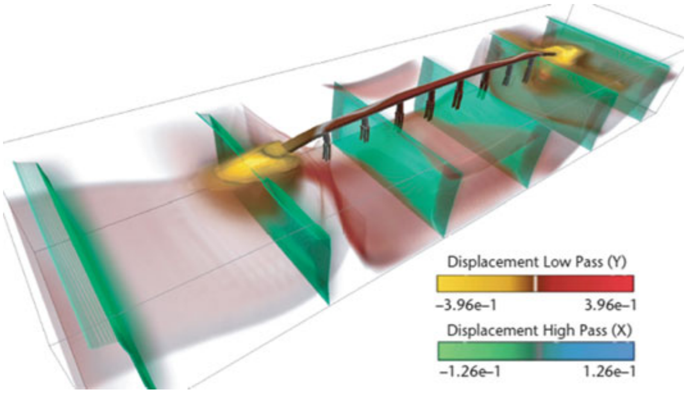


Fig. 33 Volume rendering of a simulated earthquake around a bridge structure. The top color legend represents vertical drifting. The bottom color legend represents horizontal shaking. Chen et al. [56]

will result in all strong reflectors being visible. Setting all values as transparent except for some small interval is one way of showing approximate horizon surfaces. Gerhardt and Machado [57] use the gradient magnitude as input to the transfer function similarly to the earlier work of Kindlmann [58]. The method makes it possible to discern voxels of identical intensity that have different gradient magnitude (rate of intensity change in neighborhood).

Castanie et al. [14] present an interactive system for visualizing and interpreting out–of-core seismic volumes (see Fig. 34). During interpretation, horizon and fault surfaces can be extracted. Horizons are extracted by picking a seed point and growing out a surface from the seed point based on local trace similarity around the seed point.

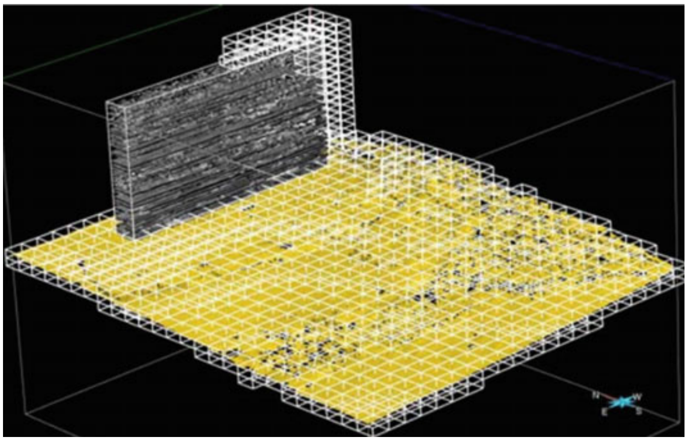


Fig. 34 A grown horizon is shown in yellow. A ROI with volume rendering is shown in gray. The white cubes show the bricks loaded into memory. The full volume size is 5.2 GB. Castanie et al. [14]

Faults are extracted by identifying positions with high vertical discontinuity in the volume and locally triangulating the positions into fault surfaces.

Patel et al. [23] present a method for fast horizon interpretation by having the seeding and growing of horizons performed in a preprocess, which results in a large amount of horizon candidates. These are then subdivided into smaller patches which are interactively selected and assembled into correct horizons without needing to wait for sequential growing algorithms to execute. The subdivision is performed since automatic horizon growing methods often wrongly connect horizons which are unrelated. The horizon preprocessing enables high-level filtering of horizon patches based on properties that are not known until the horizon is grown, such as horizon shape, average strength or variance of strength, instead of filtering on the low-level properties of individual voxels. Midtvåge and Finstad [59] perform a similar preprocessing of horizons. Gallon et al. [60] present another horizon extraction algorithm based on seeding and growing. The horizon is grown by evaluating all voxels on the horizon boundary and iteratively adding the neighbor voxel with the highest correspondence to the seed point. With this method it is possible that the growing bifurcates into thin branches (see Fig. 35). In their paper, they discuss how to handle growing in volumes that do not fit in memory. They present a method that uses less memory than ordinary bricking. In their method, the bricks are subdivided into traces. As soon as a voxel in a brick has been grown, the corresponding trace can be released from memory, making room for other traces. This is different from ordinary bricking, where only complete bricks can be released. They also discuss a hole filling method for increasing the continuous area that is grown, allowing the release of complete bricks.

Kadlec et al. [61, 62] present a method for real-time calculation of level sets on the GPU (see red surface in Fig. 36). Their method is used for extracting structures by setting a seed point and growing a closed surface. The user observes the real-time expansion of the level set and stops the growing when he/she is satisfied. Depending on the seismic attribute the level set is working on, different structures can be extracted. In previous work [63], they apply a similar method targeted at segmenting channels in seismic data.

Höllt et al. [64] present a minimum cost approach for tracing out a horizon patch confined in the triangular area between three wells. In contrast to the growing approaches presented before, the minimum cost calculation produces a global optimum instead of relying only on local voxel information. However, this also makes it computationally more expensive and infeasible for running on a complete seismic volume. Instead, the authors propose to subdivide the volume into smaller parts, for example by triangulating well positions, but arbitrary subdivision is also possible. Along each well, extra information has been measured which enables better informed seeding and verification of the calculation. The user selects three wells/positions to set up the triangle and picks a horizon seed point on the sides of the resulting prism. A horizon line with minimum cost, on a measure such as variation of intensities along the path, is traced out along the vertical side walls of the triangulation of the well (see Fig. 37 center). These boundary constraints are then used to start a minimum cost

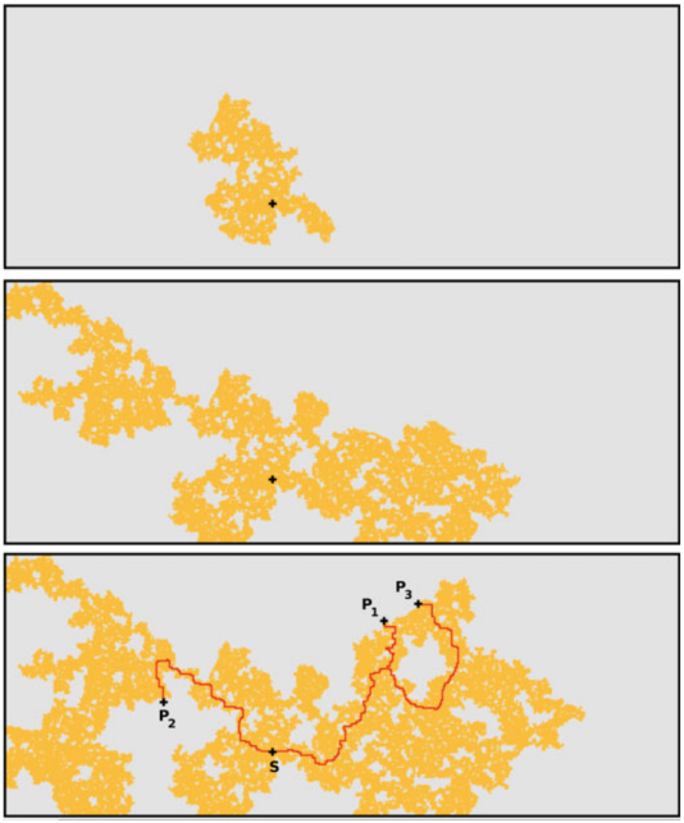


Fig. 35 From top to bottom: Three timesteps of the horizon growing method. The black cross denotes the initial seed point. Orange points show already grown parts. P1, P2 and P3 are the three most similar points to the seed point in decreasing order. The red lines show the growing route to these points. Gallon et al. [60]

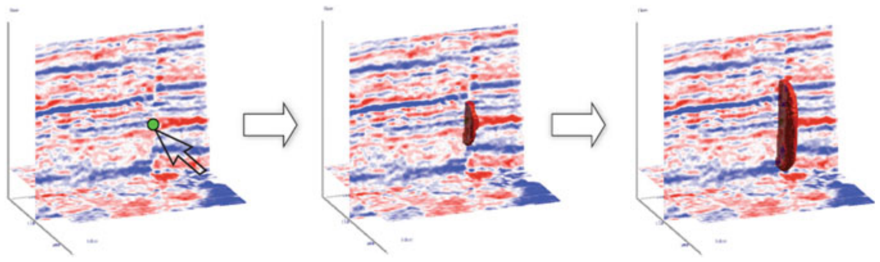


Fig. 36 Using level sets for growing a fault. The original seismic reflection data is shown whereas the fault is grown based on a seismic fault attribute volume. To the left, the user sets a seed point and the level set starts expanding along the fault (middle and right). Kadlec et al. [61]

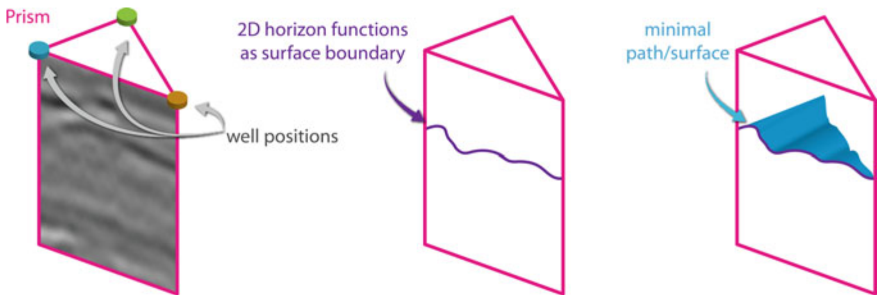


Fig. 37 The user selects three wells which form a prism, then a curve around the prism acts as a surface boundary based on which a minimal cost surface is calculated. Höllt et al. [64]

surface extraction inside the prism. The user can change the constraints by editing the boundary, or by adding points that need to be part of the final horizon patch.

In subsequent work, Höllt et al. [24] integrate horizon tracing, velocity modeling and live depth-conversion to create an integrated application for subsurface modeling. Whenever the interpreter starts the extraction of a new horizon, their application automatically creates the corresponding subsurface layers and the user can immediately input the velocity model for the newly created layers. By coupling the interpretation and velocity model in this way, it is possible to show side by side views of the original data (which has the unit of time along the depth axis) and depth-converted data (which has absolute depth in meter along the depth axis) even at very early stages of the interpretation process. Since the depth conversion happens on the GPU during rendering, no additional computation time is needed. This allows data gathered from exploration wells, which usually is available in the depth domain, to be used in the time domain-based interpretation workflow. See Fig. 38.

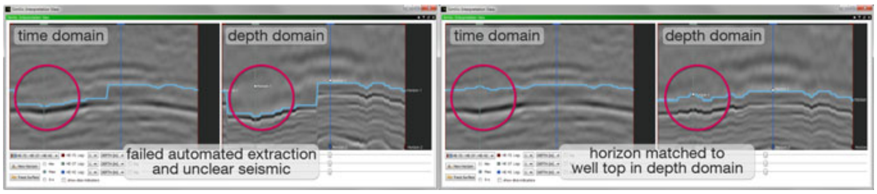


Fig. 38 Side by side time and depth domain views as presented in [24]. The visualization in the depth domain is computed on the fly according to the current intermediate interpretation and velocity model, allowing the use of data given in the depth domain. Höllt et al. [24]

4 Summary

This chapter has discussed recent advances in realtime visualization and processing for subsurface data. This topic does not have a single forum for dissemination. Research results are therefore spread throughout the several venues that exist within the distinct fields of visualization, geophysics and geology. We have attempted to collect the relevant literature with focus on visualization, structure it into categories, and review these in a meaningful order. We hope that the techniques presented in this chapter can be inspirational across the two domains of geoscience and computer graphics.

References

1. J. Plate, A. Grundhoefer, B. Schmidt, B. Fröhlich, Occlusion culling for sub-surface models in geo-scientific applications, in *VISSYM'04 Proceedings of the Sixth Joint Eurographics—IEEE TCVG Conference on Visualization*, 2004, pp. 267–272
2. G.D. Kidd, Fundamentals of 3D seismic volume visualization, in *Offshore Technology Conference*, 1999
3. S. Chopra, K.J. Marfurt, Geophysical corner: 3-D seismic volume visualization in color: Part 3. AAPG Explorer, May 2020
4. S. Chopra, R.K. Sharma, K.J. Marfurt, Seismic volume visualization in color. Search and Discovery Article #42491 (2020)
5. D. Gao, 3D seismic volume visualization and interpretation: an integrated workflow with case studies. Geophysics **74** (2009)
6. Z. Li, Volume visualization of 3D seismic data. Master Thesis. The University of Calgary. Alberta. Dept. of Computer Science, 2002
7. C. Ma, J. Rokne, 3D seismic volume visualization (Chap. 13), in *Integrated Image and Graphics Technologies* (Springer, 2004)
8. J. Kruger, W. Westermann, Acceleration techniques for GPU-based volume rendering, in *VIS '03: Proceedings of the 14th IEEE Visualization 2003* (2003)
9. J. Plate, M. Tirtasana, R. Carmona, B. Fröhlich, Octreemizer: a hierarchical approach for interactive roaming through very large volumes, in *Proceedings of VISSYM '02*, 2002, pp. 53–64
10. J. Plate, T. Holtkaemper, B. Fröhlich, A flexible multivolume shader framework for arbitrarily intersecting multi-resolution datasets. IEEE Trans. Visual Comput. Graphics **13**(6), 1584–1591 (2007)
11. L. Zhou, C. Hansen, Interactive rendering and efficient querying for large multivariate seismic volumes on consumer level PCs, in *IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, 2013
12. J. Beyer, M. Hadwiger, H. Pfister, State-of-the-art in GPU-based large-scale volume visualization. Comput. Graph. Forum (2015)
13. K. Engel, M. Kraus, T. Ertl, High-quality pre-integrated volume rendering using hardware-accelerated pixel shading, in *Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2001, pp. 9–16
14. L. Castanie, B. Levy, F. Bosquet, Volumeexplorer: roaming large volumes to couple visualization and data processing for oil and gas exploration, in *Proceedings of IEEE Visualization*, 2005, pp. 247–254
15. L. Castanie, F. Bosquet, B. Levy, Advances in seismic interpretation using new volume visualization techniques. First Break **23**, 69–75 (2005)

16. C. Lux, B. Frohlich, GPU-based ray casting of stacked out-of-core height fields. Adv. Visual Comput. ISVC (2011)
17. A. Graciano, A.J. Rueda, A. Pospisil, J. Bittner, B. Benes, An efficient representation and direct rendering of layered datasets. IEEE Trans. Visual. Comput. Graph. (2020)
18. S. Carlson, A. Pelosi, Multi-attribute visual classification of continuous and fragmented seismic data, 2007
19. B.T. Phong, Illumination for computer generated pictures. Commun. ACM **18**(6), 311–317 (1975)
20. P.M. Silva, M. Machado, M. Gattass, 3d seismic volume rendering, in *Eighth International Congress of The Brazilian Geophysical Society*, p22(3), 2003, pp. 181–193
21. A. Barnes, Shaded relief seismic attribute. Geophysics **68**(4) (2003)
22. L. Liro, K. Cline, A new paradigm for data integration and collaboration using 3-D visualization technology. The Leading Edge (2001)
23. D. Patel, S. Bruckner, I. Viola, E. Gröller, Seismic volume visualization for horizon extraction. Pacific Visual. (2010)
24. T. Höllt, W. Freiler, F. Gschwantner, H. Doleisch, G. Heinemann, M. Hadwiger, SeiVis: an interactive visual subsurface modeling application. IEEE Trans. Visual. Comput. Graph. **18**(12) (2012)
25. V. Soltészová, D. Patel, I. Viola, Chromatic shadows for improved perception. Non-Photorealistic Animation and Rendering (NPAR), 2011
26. L.Q. Campagnolo, W. Celes, An experimental study on volumetric visualization of black oil reservoir. Comput. Graph. (2020)
27. J. Grotzinger, T.H. Jordan, F. Press, R. Siever, *Understanding Earth* (W. H. Freeman and Company, 1994)
28. Federal Geographic Data Committee, FGDC Digital cartographic standard for geologic map symbolization: Reston, Va, 2006, pp. 195–197 “37—LITHOLOGIC PATTERNS”. http://www.fgdc.gov/standards/projects/FGDC-standards-projects/geo-symbol/FGDC-GeolSymFinalDraft.pdf
29. International Stratigraphic Chart. Commission for the geological map of the world. http://ccgm.free.fr/charte-souris_gb.html
30. D. Patel, C. Giertsen, J. Thurmond, E. Gröller. Illustrative rendering of seismic data, in *Proceedings of Vision Modeling and Visualization*, 2007
31. D. Patel, C. Giertsen, J. Thurmond, J. Gjelberg, E. Gröller, The seismic analyzer: interpreting and illustrating 2d seismic data. IEEE Trans. Visual Comput. Graphics **14**(6), 1571–1578 (2008)
32. D. Patel, O. Sture, H. Hauser, C. Giertsen, E. Gröller, Knowledge-assisted visualization of seismic data. Comput. Graph. **33** (2009)
33. Å. Birkeland, S. Bruckner, A. Brambilla, I. Viola, Illustrative membrane clipping. Comput. Graph. Forum **31** (2012)
34. S. Bruckner, E. Gröller, Exploded views for volume data. IEEE Trans. Visual. Comput. Graph. **12**(5) (2006)
35. T. Ropinski, F. Steinicke, K.H. Hinrichs, Visual exploration of seismic volume datasets. J. Proc. WSCG **14**, 73–80 (2006)
36. S. Bruckner, E. Gröller, VolumeShop: an interactive system for direct volume illustration, in *Proceedings of IEEE Visualization*, 2005, pp. 671–678
37. I. Viola, A. Kanitsar, E. Gröller, Importance-driven volume rendering, in *Proceedings of IEEE Visualization*, 2004, pp. 139–146
38. E. Lidal, H. Hauser, I. Viola, Design principles for cutaway visualization of geological models, in *28th Spring Conference on Computer Graphics (SCCG)*, 2012
39. A. Konev, M. Matusich, I. Viola, H. Schulze, D. Cornel, J. Waser, Fast cutaway visualization of sub-terrain tubular networks. Comput. Graph. (2018)
40. S. Takahashi, I. Fujishiro, Y. Takeshima, T. Nishita, A feature-driven approach to locating optimal viewpoints for volume visualization, in *Proceedings of IEEE Visualization*, 2005
41. C. Giertsen, Direct volume rendering of multiple scalar fields. J. Visual. Comput. Animat. **5** (1994)

42. A. Rocha, R.C.R. Mota, H. Hamdi, U.R. Alim, M. Costa Sousa, Illustrative multivariate visualization for geological modelling, in *Eurographics Conference on Visualization (EuroVis)*, 2018
43. T.M. Toledo, W. Celes, Visualizing 3D flow of black-oil reservoir models on arbitrary surfaces using projected 2D line integral convolution, in *24th SIBGRAPI Conference on Graphics, Patterns and Images* (2011)
44. B. Cabral, L.C. Leedom, Imaging vector fields using line integral convolution. SIGGRAPH '93
45. B. Franceschin, F. Abraham, L.F. Netto, W. Celes, GPU-based rendering of arbitrarily complex cutting surfaces for black oil reservoir models, in *SIBGRAPI Conference on Graphics, Patterns and Images* (2019)
46. A. Calomeni, W. Celes. Assisted and automatic navigation in black oil reservoir models based on probabilistic roadmaps, in *Symposium on Interactive 3D graphics and games (I3D)*, 2006
47. T. Hollt, F.M. Ravanelli, M. Hadwiger, I. Hoteit, Visual analysis of reservoir simulation ensembles, in *Workshop on Visualisation in Environmental Sciences (EnvirVis)* (2016)
48. S. Mazza, D. Patel, I. Viola, Homomorphic-encrypted volume rendering. IEEE Trans. Visual. Comput. Graph. (2021)
49. K. Bhattacharjee, M. Chen, A. Dasgupta, Privacy-preserving data visualization: reflections on the state of the art and research opportunities. Comput. Graph. Forum (2020)
50. J. Henderson, S. Purves, G. Fisher, C. Leppard, Delineation of geological elements from RGB color blending of seismic attribute volumes. The Leading Edge (2008)
51. J. Liu, K. Marfurt, Multi-color display of spectral attributes. SEG, 2006
52. J. Liu, K. Marfurt, Instantaneous spectral attributes to detect channels. Geophysics **72**(2) (2007)
53. H. Guo, K. Marfurt, J. Liu, Principal component spectral analysis. Geophysics **74**(4) (2009)
54. E. Vuçini, D. Patel, E. Gröller, Enhancing visualization with real-time frequency-based transfer functions, in *Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis*, 2011, pp. 78680L-1–78680L-12
55. cuFFT, the NVIDIA® CUDA™ Fast Fourier Transform (FFT). CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/cufft/index.html. Accessed Mar 2021
56. C. Chen, C. Ho, C. Correa, K. Lu Ma, A. Elgamal, Visualizing three-dimensional earthquake simulation data. Comput. Sci. Eng. (2010)
57. A. Gerhardt, M. Machado, Enhanced Visualization of 3-D Seismic Data. SEG (2002)
58. G. Kindlmann, Semi-automatic generation of transfer functions for direct volume rendering. M.Sc. Thesis, Cornell University, 1999
59. S. Midtvåge, A. Finstad. GIM—A Method for Computer Assisted Seismic Interpretation. SEG Houston (2009)
60. J. Gallon, S. Guillon, B. Jobard, H. Barucq, N. Keskes, Slimming brick cache strategies for seismic horizon propagation algorithms, in *l Symposium on Volume Graphics*, 2010
61. B. Kadlec, H. Tufo, G. Dorn, Knowledge-assisted visualization and segmentation of geologic features. Comput. Graph. Appl. (2010)
62. B. Kadlec, G. Dorn, Interactive Visualization and Interpretation of Geologic Surfaces in 3-D Seismic data. SEG Houston, 2009
63. B. Kadlec, Confidence and curvature-guided level sets for channel segmentation, in *78th Annual International Meeting*. SEG 2008
64. T. Höllt, J. Beyer, F. Gschwantner, P. Muigg, H. Doleisch, G. Heinemann, M. Hadwiger. Interactive seismic interpretation with piecewise global energy minimization. Pacific Visual. (2011)

Overview of Seismic Attributes and Seismic Object Extraction

Tore Grane Klausen, Behzad Alaei, and Daniel Patel

Abstract This chapter aims to provide an accessible summary of key steps in the practice of seismic mapping, along with a brief explanation of relevant geoscience terminology. We present how attributes derived, and objects automatically extracted, from seismic data are used to evaluate and map the subsurface.

1 Introduction

This chapter aims to provide an accessible summary of key steps in the practice of seismic mapping, along with a brief explanation of relevant geoscience terminology. We show how attributes derived from the original seismic reflection data are typically used to evaluate and map the subsurface. First, we briefly summarise the backdrop for what is being mapped in the subsurface and the cause for the layer-cake structure of the earth visible in the seismic data. These basic prerequisites are necessary in order to understand what an interpreter has in mind when making sense of the patterns recorded in the subsurface. Seismic data is noisy, has low spatial resolution and only depict certain structures depending on depth. Interpreting seismic data can be considered looking at the underground through a keyhole, and the more tools that can help make efficient and objective assessments the better.

Several promising machine learning algorithms for seismic attribute and object extraction have recently been developed. At the end of the chapter, we discuss the use of machine learning algorithms as an aid to interpreting seismic data, but in general we do not give such algorithms much focus in this chapter as describing e.g. deep neural network algorithms does not necessarily give the reader a better

T. G. Klausen
Petrolia NOCO AS, Espehaugen 32, 5285 Bergen, Norway

B. Alaei
Earth Science Analytics AS, Oslo, Norway
e-mail: behzad.alaei@earthanalytics.no

D. Patel (✉)
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_3

understanding of the geological structures that the algorithm extracts. For a more extensive overview of machine learning algorithms, see the chapter titled "Using Interactive Visualization and Machine Learning for Seismic Interpretation", and the "Machine Learning in Geosciences" book edited by Moseley and Krischer [68].

This chapter covers steps 3, 4, 5 and 6 in the geological processing pipeline:

1. Seismic survey data collection
2. Data inversion
3. Data filtering
4. Attribute calculation
5. Visualization
6. Low level object extraction (for objects such as faults, horizons and salt diapirs)
7. High level object extraction (for objects such as reservoirs)
8. Geomodel creation
9. Flow simulation.

There are several more exhaustive literature sources on the subject, and especially on each of the different methods and attributes presented here.

1.1 Geologic Processes Forming the Subsurface

Sedimentary rocks in the subsurface have a layer-cake nature due to the processes involved in the deposition and compaction of sediments. Sediments can be derived from eroding exposed landmasses or can be created by organisms living in the sea. They are typically deposited within a sedimentary basin, of which the deep-marine realms of today's oceans are one example. The physical environment and spatial configuration at the earth surface changes slowly and in cycles. Accordingly, the material types that are deposited vary with these cycles. The layering or succession of rock in the subsurface is termed by geologists as the lithostratigraphic succession. Lithostratigraphy tries to explain the subsurface (earth) as connected or disconnected rock volumes and their relationships: depositional hiatuses (periods with no sediment accumulation) and erosional boundaries (the removal of previously deposited sediments or rocks) across which the sedimentary rock succession is interrupted and might be of a totally different source, or a continuation of the preceding.

Understanding the subsurface and different basins found therein can be used to identify organic-rich accumulations that represent hydrocarbon source rock. Further, reservoirs for hydrocarbons will be emplaced by sedimentary processes accommodating porous rock volumes, such as a sandy beach. The sedimentary record along with the tectonic history will determine whether hydrocarbons will be sealed in reservoirs.

Hydrocarbons are created when organic material has been deposited, buried and transformed to kerogen. Maturation of kerogen into source rock for hydrocarbon occurs under correct temperatures referred to as the "oil-window" (60–140 °C for oil, and 120–225 °C for gas). Time is important in the respect that it gives the kerogen opportunity to mature. How much time is needed to reach the necessary temperatures, and how much time is needed to expel hydrocarbons depends on the basin, but is typically millions of years. These hydrocarbons then migrate upwards through the subsurface via porous and permeable rock or fractures before it can potentially be collected in reservoirs, given that the reservoir is overlain by an impermeable rock layer. Such impermeable rock layers are referred to as cap rock and it is crucial for the accumulation of hydrocarbons. In short, one needs a hydrocarbon source from an organic-rich deposit, a permeable migration path to a reservoir made of porous and permeable rock as storage space, and an impermeable top layer that caps the reservoir and traps the hydrocarbons.

A reservoir might consist of three fluid zones: gas at the top (gas cap), an oil rim in the middle and a water zone (brine) at the bottom of the reservoir rocks. These fluid layers are naturally horizontal and permeable rocks saturated with these different fluids have different acoustic properties (velocity and density). These characteristics (horizontal delineation and acoustic properties) have been used to directly identify and distinguish hydrocarbon saturated reservoir rocks from brine saturated rocks in seismic data. Such clear indicators are called Direct Hydrocarbon Indicators (DHI) and can sometimes be seen directly in the seismic data (shown later in Fig. 12b) or by using rock physics in combination with Amplitude vs Offset/Angle data [89]. There are also other DHIs such as the ones derived from the interpretation of Controlled Source Electro-Magnetic (CSEM) data [31, 90].

1.2 Seismic Reflection Data

Reflection seismologists make images of the earth's interior [24]. The earth is excited using a source of energy, for example air gun in marine surveys. The pressure wave (e.g. originating from an air gun) propagates into the earth. The acoustic properties of rocks of contiguous layers vary (sharply or gradually), giving rise to acoustic impedance (the product of the rocks pressure wave velocity and average material density) contrasts at the layer boundaries. The pressure wave is reflected and/or refracted when the acoustic impedance of the material changes. The reflected energy from geological layers is recorded on a series of receivers laid down along a line (creating a 2D seismic image), or several lines (creating a 3D seismic volume) on the surface of the earth. The subsurface image is created using data recorded at the receivers and through a series of mathematical applications that is referred to as seismic processing.

Three dimensional seismic (reflection) volumes provide the opportunity to map the spatial distribution of sedimentary rocks in three dimensions using certain techniques that will be presented in this chapter. Features that can be distinguished and extracted from a three-dimensional dataset are called geobodies. In general, geobody is a term that aims to capture the bulk of volumetrically significant and geomorphological distinct rock bodies, i.e. those that can be differentiated (typically within seismic data) and described in three dimensions. Examples of such could be salt domes, ancient beach ridges and river deposits, and a variety of other hydrocarbon reservoirs. River deposits are referred to as channels, due to their shape, in the following text. As an aid to extracting geobodies from seismic data, horizons and fault surfaces can be extracted first. Horizons are the layer boundaries imaged in the seismic reflection volumes. Faults represent discontinuities in the subsurface layering, where horizons on one side of the fault have been shifted relative to the horizons on the other side due to subsurface forces.

1.3 Introduction to Seismic Attributes

A seismic attribute is any quantitative measure derived from the seismic data that helps us visually enhance or quantify geological features of interpretation interest [20]. A historical perspective on seismic attributes is given in [17]. A wide range of seismic attributes are available today. Some go by different names and computation methods but provide the same information. Similarities and relevance of different seismic attributes are discussed by Barnes [7]. Seismic interpreters can build one or more geological hypotheses or models after screening the seismic data. The choice of seismic attributes to be used depends to a great degree on those hypotheses. The application of various seismic attributes often boils down to case-specific tasks and, ultimately, personal preference.

We will mention the array of geological disciplines and the difference between geoscientists and petroleum engineers at this stage. Individuals who consider one set of attributes as unusable may fail to appreciate its application in other disciplines. A sedimentologist will pay attention to different features than a structural geologist when investigating a seismic volume. For example, geomorphological features defined by their lateral *continuity*, such as channelized deposits that can be observed in map-view and extracted as discrete geobodies, will often be in focus. A structural geologist will be concerned with the lateral *discontinuities* (e.g. faults). The petroleum engineer, on the other hand, will be concerned with the reservoir properties of certain intervals of the rock volume, and the fluid flow within these units. They are thus much more focused on rock volume parameters, and upscaled versions of the geological model that will best capture the reservoir properties of a specific interval of the subsurface succession. Computation time and accuracy are the main priorities for the petroleum engineers. Therefore, although a multidisciplinary effort is crucial for a successful exploration and production campaign, different geoscience disciplines will pay different aspects of the seismic dataset more attention. Interesting pitfalls in interpreting seismic data are pointed out in [9, 41, 63].

Some expressions used in geoscience and throughout this chapter will be defined in this paragraph. When visualizing 3D subsurface seismic data, often 2D sections of the 3D data are shown. These can have different orientations and shapes. A **time-slice** view is a horizontal slice through the seismic volume along a specific depth or time value. Here, "time" does not refer to geologic time, but time, in the order of milliseconds, from a sound signal was sent to the reflection was received by a geophone. A **cross-section** view is a vertical slice through the seismic volume. Each vertical column of seismic values in a cross-section view is called a (seismic) **trace**. This is an array of consecutive samples along the depth axis, and when plotted on a graph it will resemble a waveform according to the reflected sound. A cross-section can be referred to as an **in-line** or a **cross-line.** They are orthogonal to each other. Seismic data draped onto a gridded surface describing a geological surface is referred to as **map-view** or horizon-view (an example is shown in Fig. 3b).

1.4 Filtering of Seismic Data for Noise Removal and Feature Enhancement

Almost all seismic data is contaminated by different types of noise. Noise is part of the seismic data that does not have a corresponding reflective source in the geology. Noise reduction is best addressed during data acquisition or processing. A good seismic survey design can suppress noise. During seismic processing there are several stages of signal enhancement applications. However, a seismic interpreter does not have direct access to any of these stages. The interpreter is left with a product of seismic processing and thus with limited choices to perform noise reduction. There are two types of seismic noise: random and coherent. Human observers can recognize coherent noise if their scale and orientation are different from the signal. Removing random noise is easier since it does not influence the signal, whereas removing coherent noise is challenging since it may influence the signal content of the data. Noise content of offshore seismic data is usually lower than land seismic data. A common operation to reduce noise is to employ noise-filtering techniques. Such techniques are used to enhance the spatial continuity of seismic data, whilst maintaining subtle details such as small-scale fault displacements. Noise reduction usually results in data that is easier to interpret and provides a cleaner input for the subsequent stages of the attribute analysis workflows, such as horizon autotracking (automatic segmentation of stratigraphic boundaries, discussed in more detail in a subsequent section).

The type of noise removal application and specific parameters depends on the noise content of the seismic data. A careful investigation of seismic data prior to noise removal stage is required. Small features in the data, such as subtle fault displacements, should be checked before and after the noise reduction process to make sure that important features are not removed from the seismic volume. One must also consider the potential pitfall of over-smoothing, which may obscure relevant information within the dataset. There are different noise filtering approaches which can be chosen according to the noise type and content of the input seismic data.

Noise can be suppressed by assuming that the seismic signal has some degree of lateral continuity along the horizon structures (i.e. that the signal varies less along these structures than it does along the depth). This is exploited by smoothing the data along the direction of horizon structures. This is called structurally oriented smoothing. The first step in this approach is to differentiate the spatial orientation (Dip and Azimuth) of seismic reflectors from noise. Then a noise reduction filter is applied in a small window (in 3D) rotated so the filter's coordinate system aligns with the estimated orientation of the data. Two common filters are the mean filter which takes the average of the samples within the analysis window, and the median filter which replaces each sample within the analysis window by the median of the samples that fall within the analysis window. The analysis window size has an impact on the results. Figure 1 shows an example of the structurally oriented median noise removal filter. The analysis window is 11 × 11 × 11 which means 11 inlines, 11 cross lines, and 11 samples in time.

Structurally oriented edge-preserving filters (e.g. [57]) do a good job of removing coherent noise while keeping subtle detail such as small-scale faults. Hoecker and Fehmers [42] and Fehmers and Hoecker [30] present in a series of articles the use of anisotropic diffusion for performing smoothing along the directions of horizons. The smoothing operation is sensitive to discontinuities (edges) resulting in faults being preserved and not smeared away. The number of diffusion steps can be controlled to choose how smooth the result should be (See Fig. 2 for an example). A different structure-oriented filtering method is presented by Labrunye and Mallet [48]. For each sample in the seismic data, they consider the signal above and below relative to the sample in a local 1D neighborhood which is referred to as a trace. They then take the corresponding trace of a horizontal neighbor and shift the two traces until they have maximum correspondence. This shift corresponds to the angle of the

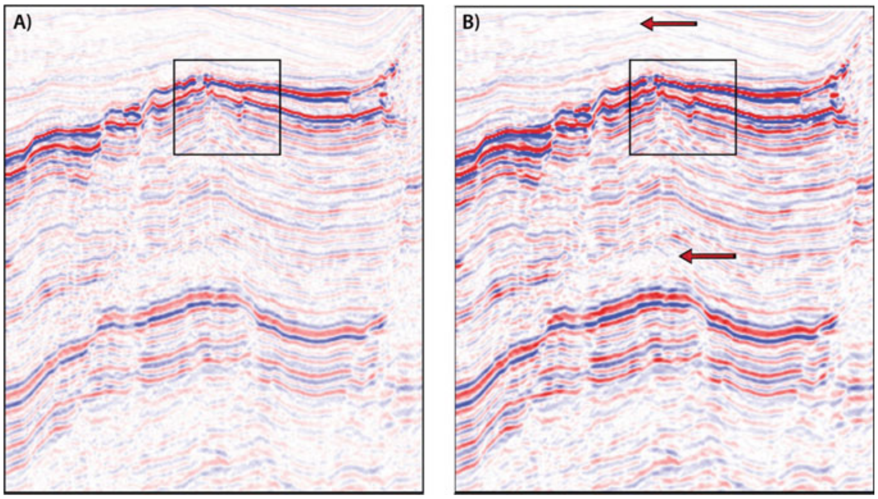


Fig. 1 Seismic cross-sections. Noise removal using a structurally oriented median filter. **a** Unfiltered seismic image. **b** Seismic image after noise reduction has clearer and sharper edges than the unfiltered image (**a**). The black box show how faults are accentuated, while the red arrow indicates areas with better continuity of reflectors and less noise

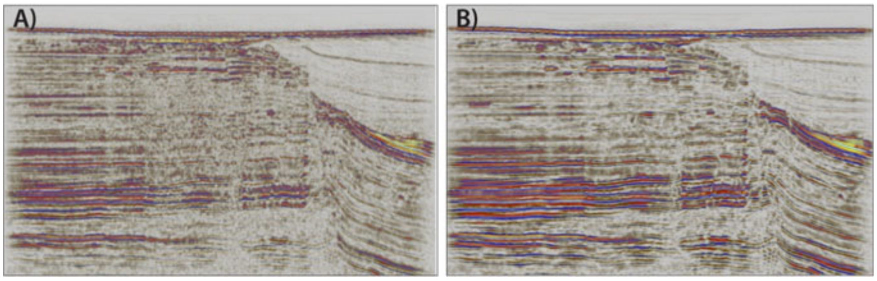


Fig. 2 Seismic cross-sections. **a** Original seismic section. **b** Structurally smoothed seismic section. Structural smoothing can be used to enhance continuity of seismic reflectors

structure. Noise reduction is then calculated by averaging the center sample of the original trace with the corresponding sample value of the other. This has the effect of removing random noise and highlighting coherent signals. To avoid averaging the traces that come from different structures, averaging is not performed when the maximum correspondence is too low. One approach is to model the discrete traces using trigonometric polynomials. The covariance between the traces can also be expressed as a trigonometric function and its maximum value and thereby the shift can be found analytically and fast. Helmore [39], and Chopra and Marfurt [23] split the seismic volume into different frequency domains and perform the structure-oriented smoothing on each volume before combining them back to one volume. This approach will avoid smearing out high frequency details in low frequency structure and vice versa since frequencies are smoothed out separately. Al-Dossary and Wang [2] present a conceptually simple and computational fast method for smoothing of the seismic data. For each voxel they try to fit a plane so that the neighboring sample values intersecting the plane optimize some measure, such as 1) minimum deviation or 2) maximum, minimum, or absolute summation. This plane will then represent the local orientation of the horizon. The center voxel is then smoothed according to the neighboring values on the plane. Davogustto and Marfurt [25] employ principal component analysis on a neighborhood around each voxel to determine the angle of the horizon and use the coherent part of the signal for smoothing steps. To avoid smoothing the true discontinuities, data is not smoothed where strong discontinuities are detected.

Figure 2 shows two vertical seismic profiles through a 3D seismic volume, before and after structural smoothing with a Gaussian filter. Even when the original seismic is of relatively high quality, with a high signal to noise ratio, (semi) automated mapping algorithms, for instance for horizon extraction, are likely to be more accurate when using smoothed data.

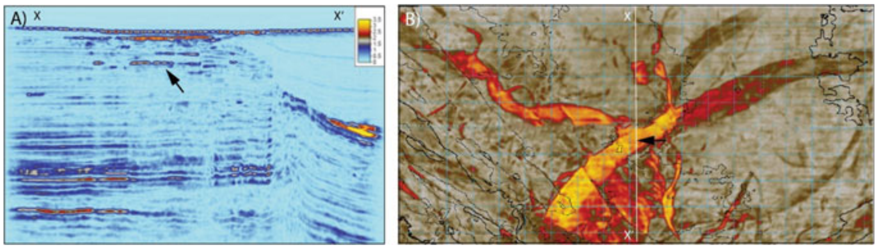


Fig. 3 **a** A cross-section with a colour map showing the max signal strength in yellow and min signal strength in blue. The original seismic on which this attribute is based, can be seen in Fig. 2. **b** Map-view of signal strength attribute extracted on an interpreted horizon. The colour map used is grey (min) to yellow (max). Location of the cross-section in **a** is indicated by the white line X-X′. The seismic geomorphology observed here is a fluvial channel complex partly filled with gas, as indicated by arrows in **a** and **b**. The anomaly and geobody is discussed further in following sections.

1.5 Basic Seismic Attributes

As noted above, a seismic attribute is a measurement derived from seismic data. Attributes are usually based on the measurements of time, amplitude, frequency, and/or attenuation. Generally, time-based measurements relate to structure, amplitude-based ones to stratigraphy and reservoir characterization, and frequency-based ones (while often not clearly understood) to stratigraphy and reservoir characterization [87]. Since the publication of complex trace analysis by Taner et al. [91], seismic attributes are frequently used in the 3D seismic interpretation process. Attributes derived from individual seismic traces are called trace-based seismic attributes (e.g. signal strength, and instantaneous phase) and attributes derived from unit data volumes are called volume-based attributes (e.g. coherency attributes).

Different commercial actors may provide software for calculating the same attribute. Since they are using different algorithms, the quality will differ. Also, some attributes are calculated based on other attributes as input.

The following section will present some of the most important attributes and how they are calculated.

1.5.1 Signal Strength

Signal strength is one of the most basic attributes, and it is known by different names, such as envelope amplitude, magnitude, or reflection strength. If we consider the seismic trace f(t) as the real part of a complex trace and calculate the imaginary part q(t) using the Hilbert transform [91] then signal strength E(t) will be

$$E(t) = \sqrt{f^2(t) + q^2(t)}$$

The signal strength is often used to help identify major lithological changes, i.e., differences within the subsurface succession, for which there is a strong energy reflection. This attribute has a poor vertical resolution but good lateral or horizontal resolution. This basic attribute may also help identify more subtle lithological changes, particularly in time-slice (having constant time or depth) or cross-sectional (increasing time or depth downwards along y axis) views, not otherwise resolvable in the seismic data.

Figure 3 illustrates the signal strength attribute of the seismic section shown in Fig. 2. Areas of high amplitude signals are mapped to bright yellow. In this example, the peak amplitudes are found along the seabed; a regional erosive surface that merges with the seabed signal; regional flooding surfaces of stratigraphic significance; and the regional basement reflector at the very bottom of the profile.

1.5.2 Instantaneous Phase

Instantaneous phase is the phase of the seismic trace. Instantaneous Phase (Ip) is calculated using:

$$Ip(t) = arctan(\frac{f(t)}{q(t)})$$

This is independent of the amplitude, and thus useful for the recognition of the continuation of reflectors that show great variety in their amplitude. It can be used to resolve the spatial continuation of weak horizons and to distinguish more subtle discontinuities such as faults or pinch-outs (a layer thinning out until it disappears). Even bed interfaces, sequence boundaries and smaller dipping events can be resolved using this phase attribute. Figure 4a illustrates the instantaneous phase attribute of the seismic section shown in Fig. 4b (and in Fig. 2a).

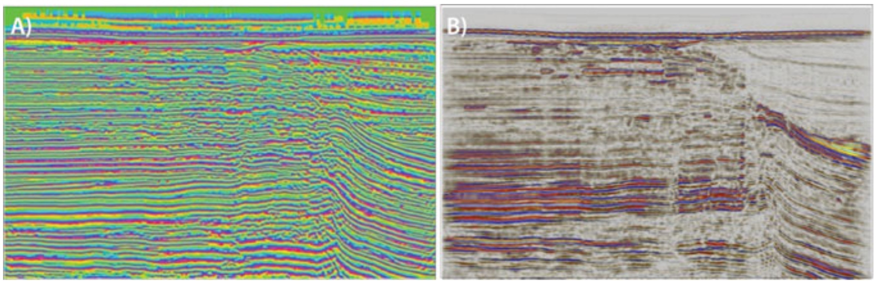


Fig. 4 **a** Instantaneous phase for the cross-section used in the previous figures. The attribute highlights continuity of seismic reflectors, as opposed to for example relative amplitude. **b** The original, amplitude data from which the phase attribute is extracted

Instantaneous frequency is the derivative of the instantaneous phase with respect to time. It can be used to identify depositional features at small scale or high frequencies. Most reflection events are the composite of individual reflections from several closely spaced reflectors. The superposition of individual reflections may produce a frequency pattern which characterizes the composite reflection. The character of a composite reflection will change gradually as the sequence of layers gradually changes in thickness or lithology. Variations, for instance at pinch-outs and at edges of hydrocarbon-water interfaces, tend to change the instantaneous frequency more rapidly [91].

1.5.3 Cosine of Phase

Cosine of instantaneous phase is continuously smooth and does not contain the discontinuities observed in the instantaneous phase attribute volumes. The cosine of phase attribute emphasizes the continuity of the events and is a function of time and not frequency. Because phase is independent of signal strength, it often makes weak coherent events clearer. Prograding sedimentary layer patterns and regions with onlapping and offlapping layers [16] often appear with better clarity on the cosine of phase attribute than in conventional seismic (amplitude) data. Figure 5 shows a seismic cross-section with corresponding cosine of phase attribute. The reflector terminations are better imaged on the attribute and can easily be interpreted.

1.5.4 Dip and Azimuth

Dip and azimuth, shown in Fig. 6, are the angles that define the orientation of a plane in 3D space. Quantitative estimates of dip and azimuth throughout seismic volume is key to mapping seismic reflectors. Identification of subtle changes in reflector dip and azimuth allows us to infer features such as faults, angular unconformities, coherent progradational packages, as well as fans, chaotic slumps, and braided-stream complexes. The dip attribute refers to the inclination (dip) of the layers in an in-line cross-section, and the azimuth is the orthogonal inclination (the dip in the cross-line). A layer within the dataset might have varying dip due to either depositional processes or post-depositional overprint (regional tilting of the whole lithological succession, or local/regional tectonic influence that in one way or another alters the dip and trend the layer/horizon had at its time of deposition) at one or more stages. Such trends can be resolved by the dip-azimuth attribute. Figure 6 shows in (a) an azimuth volume, and in (b) a dip volume.

A naive approach for calculating dip and azimuth is to find the normal vector of the structure in the seismic data based on the central difference method. This method will be sensitive to intensity variations along the horizons and for that reason more advanced methods have been proposed [58, 62, 81]. These methods include the use of Gaussian smoothing followed by principal component analysis [81].

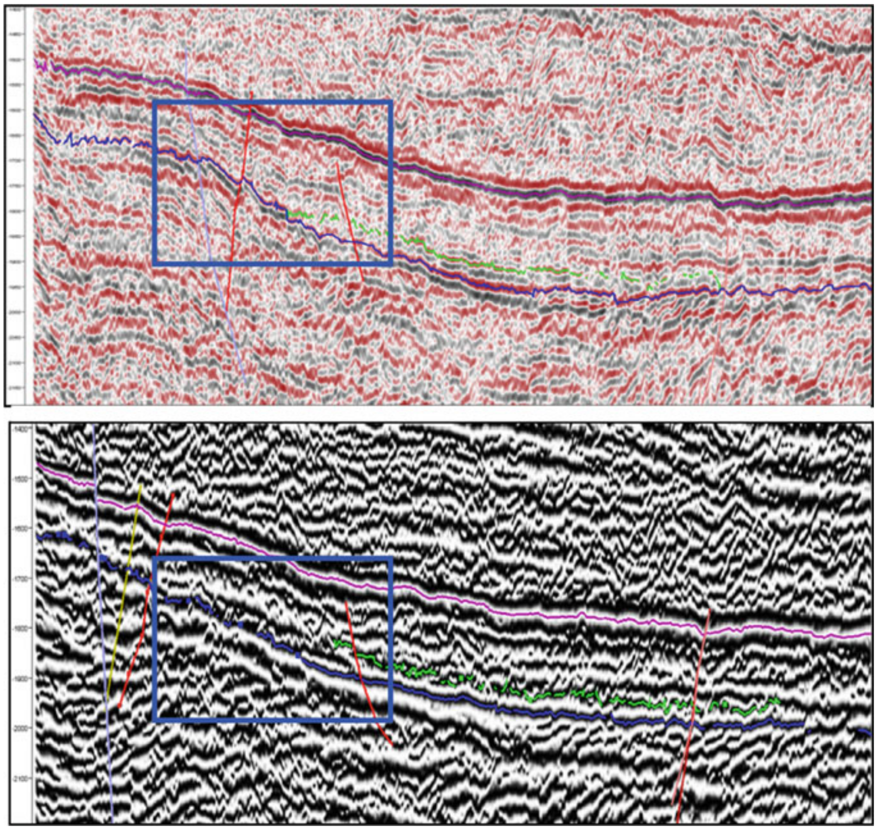


Fig. 5 2D seismic cross-section (top) and corresponding cosine of phase attribute (bottom). Overlaid vertical red lines represent faults, the blue and violet line is the top and base of a geological formation, whereas the green line represents a layer that pinches out within the formation. The pinch outs and weak amplitude events within the blue square are better imaged on the cosine of phase attribute (bottom)

There are three common methods of estimating volumetric dip and azimuth attributes using 3D seismic volumes as input [20]: (1) Calculating temporal and spatial derivatives of the phase estimated using complex trace analysis (e.g. Luo et al. [56]), (2) Explicit dip scan to find the most coherent reflector [60], and (3) Calculation of vector dip using the gradient structure tensor by cross-correlating the gradient of the data and forming a gradient structure tensor e.g. [42].

Dip and azimuth volumes are used directly to interpret faults, coherent progradational and transgressive deposits, fans, braided-stream complexes, and unconformities. The dip and azimuth volumes are also used to guide the other attribute calculations such as coherence attribute in a structurally oriented flow. Noise removal by structural smoothing, as described earlier section, often needs dip and azimuths for identifying the rotation of the smoothing neighborhood.

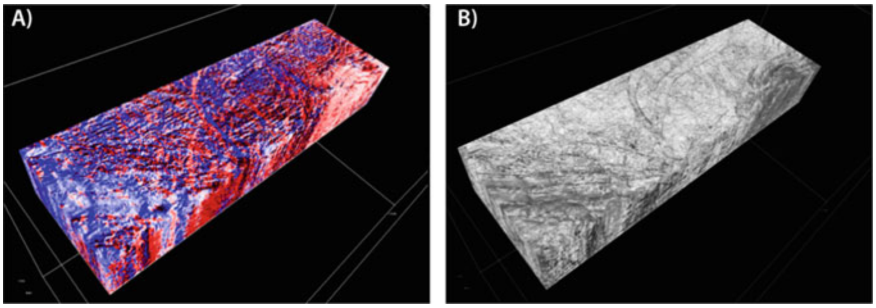


Fig. 6 **a** Azimuth attribute, a red to blue colour map is used. The colours show that the layering within this seismic volume has two marked dip trends perpendicular to one another. **b** Dip attribute of the same volume, white to black colour map is used. A combined visualization of these volumes using an appropriate colour mapping would create a dip-azimuth attribute co-blend

1.5.5 Coherence

Coherence is a measure of similarity between adjacent seismic traces. The coherence attribute, shown in Fig. 7, basically aims to transform a volume with emphasis on the continuous reflectors, into a volume where the discontinuities instead are highlighted. Such discontinuities are often indicative of faults, but can also indicate other steep lithology variations, for example the valley-incisions caused by the rivers. Valleys are either incised by glaciers or rivers and both tend to create steep valley walls depending on the substrate into which the river/glacier eroded. These steep walls are picked up as discontinuities in the rock record and will stand out as such in coherence attribute data. Where faults are readily traced for many meters to kilometers, valley incisions are typically restricted to tens (hundreds at most) of meters.

The coherence attribute was first introduced by Bahorich and Farmer [3] where they calculated the similarity of neighboring traces using a cross correlation function. Samples on continuous horizons will have similar traces in their neighborhood

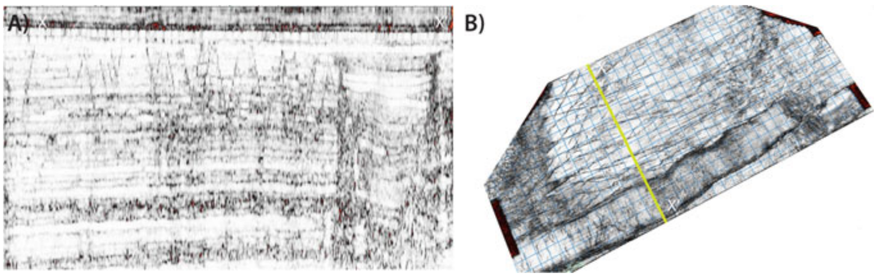


Fig. 7 Coherence (variance) attribute of a 3D seismic dataset. Dark colour signifies lack of coherence and possible presence of faults. In **a** is shown a seismic cross-section. Upper left part shows V shaped fault patterns. In **b** is shown map-view where faults are recognized as cuspate lines. The location of the seismic cross-section in **a** is given by the yellow line

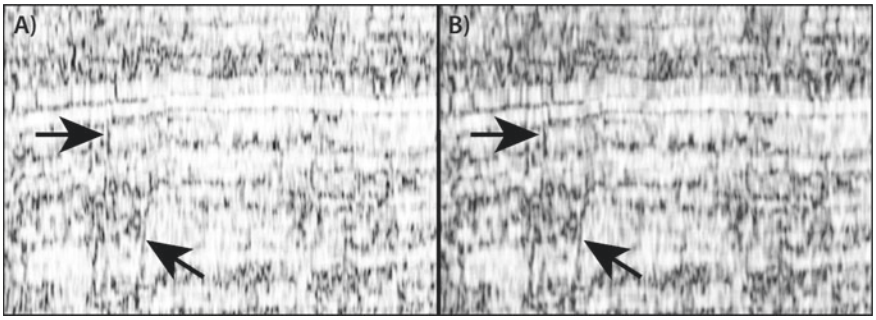


Fig. 8 The effect of analysis window size on coherence attribute results. **a** 5 by 9 window. **b** 9 by 9 window. A wider spatial window shows better continuity of the faults on the section, as indicated by the arrows

and therefore high coherence, whereas samples on faults, which have cracked and displaced horizons on each side, have low coherence. Gibson et al. [34] extend the method by thresholding the coherence data to identify fault areas followed by clustering and triangulating the data so that vertical surface objects are created through potential faults. All coherence attribute measurements take place in a spatial window of adjacent seismic traces. The number of traces involved in the coherence attributes starts with 3 traces in simple cross-correlation methods and increases to five, nine, and even more in advanced algorithms such as eigen-structure coherence algorithms. The coherence attribute algorithms include cross correlation (by Bahorich and Farmer [3]), semblance [61], the variance-based method, eigenstructure based coherence, and the gradient structure tensor-based coherence method (e.g. [4]). All these methods use different mathematical or statistical applications to derive the seismic trace to trace similarity. Some of the methods of coherence calculation require dip and azimuth attributes as input (steering) volumes. Methods that use high quality dip/azimuth steering volumes usually show less structural artifacts. The analysis time window in three dimensions influences the size of the discontinuities that will be imaged in the coherence attribute. If the analysis window is laterally short and vertically long, steep faults will be better imaged while if a wider analysis window is selected, small scale discontinuities will be missed (e.g., [51]). Figure 8 illustrates the difference in the scale and extent of imaged faults using different analysis window sizes.

1.5.6 Fault Enhancement

Coherence and other attributes such as dip and azimuth volumes provide measures in seismic waveform variations and 3D orientation (dip and azimuth). With this technique, geoscientists may interpret features such as fault planes and channel boundaries. There have been several attempts to improve the coherence attribute images in order to make some of this mapping easier. Fault enhancement has received a lot of attention. Automatic fault interpretation using enhanced faults has been one of the goals of seismic mapping. The first method is to filter coherence attributes to enhance fault detection [6]. Filter parameters are dip, planarity, and resolution of discontinuities.

Pedersen et al. [75] present ant tracking for enhancing faults from coherency data by strengthening the response on samples that form lines with steep angles. This is done by simulating the movement of many virtual ants spread out in the volume where the ants are mostly allowed to move along steeply angled samples having high discontinuity. When they move in areas with low discontinuities, they carve out channels and thereby connect samples with high discontinuities that are close to each other. This will in effect enhance fault-like discontinuities. The work is extended [76] by adding a triangulation step for creating geometric fault surfaces with the addition of a filter for selecting fault groups of specific orientations and sizes. The algorithm by Donias et al. [28] can accurately detect fault discontinuities while ignoring other discontinuities. These typically arise from structures such as channels or from noise. The algorithm checks whether the points are of high discontinuity in a 3D neighborhood around a sample that coincides with an arbitrarily oriented planar geometry. Luo et al. [59] also perform plane fitting. For each sample in the data, oriented planes are inserted that splits the local neighborhood in two. If the coherence of the neighborhood before the split is lower than the coherence of each split neighborhood, then this is an indication that the sample is situated at a fault having the orientation of the split plane.

Figure 9 shows a time-slice from a coherence attribute before (a) and after (b) fault enhancement generated using a Gaussian filter applied to the coherence attribute. The faults are clearer in the fault enhanced attribute. This feature usually needs an edge sensitive attribute (e.g. chaos attribute as described in next section) as input. Several machine learning algorithms are being developed for identifying faults. One example is given in the subchapter "Seismic attributes and machine learning".

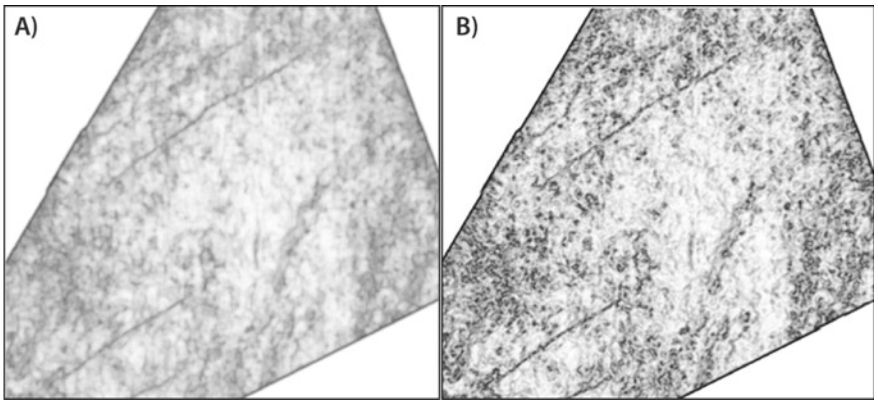


Fig. 9 Fault enhancement: **b** shows a fault enhancement of the coherency attribute shown in **a**

1.5.7 Chaos

Chaos is an attribute that quantifies any chaotic trends (lack of order) within the dataset. This can identify gas migration pathways, salt domes, reef buildups and/or chaotic channel infill (of seismically resolvable scale). These phenomena can be explained by features in the rock record, but they differ in single seismic trace by being depositional or syn-depositional features (they were initiated during or after sediments were deposited on top of them) or by being post depositional (i.e. some of these features were emplaced after the rock succession had been formed). Reef buildups and chaotic channel infill are both examples of depositional processes that might not follow the archetypical 'layer-cake' depositional trend of the sedimentary rock record. The channel infill must be of a rather significant size to be resolvable in seismic data. Salt movement, and the migration of highly ductile material, is typically triggered by the increased weight of the overlying sediments and tend to occur along some preexisting fault planes. Therefore, the salt structures are syn- to post-depositional. Gas migration pathways are post-depositional features that arise from the leakage, and percolation, of gas through the subsurface. These gas-chimneys as they are often referred to have an internal chaotic structure which is not caused by the rock type but created by acoustic impedance change due to gas saturation of the sediments. They are important as they can confirm a working petroleum system in the area. Common to all the features listed above is that they are close to vertical and have a highly chaotic internal ordering, and thus contrast the ordered layered sediments around them. For this reason, the chaos attribute is a very convenient tool for recognizing such local trends within the dataset. Figure 10 illustrates an example of a chaos attribute volume displayed on a profile through two salt domes.

Several approaches (e.g. [81]) calculate chaos by measuring the incoherence of the dip and azimuth in the data. Yang et al. [98] have a different approach where they look at the trace around a sample and calculate the entropy measures on the curve to find the degree of chaos. Also, use of the fractal dimension measure can be used as a chaos indicator, for instance for identifying reef carbonate reservoirs [99].

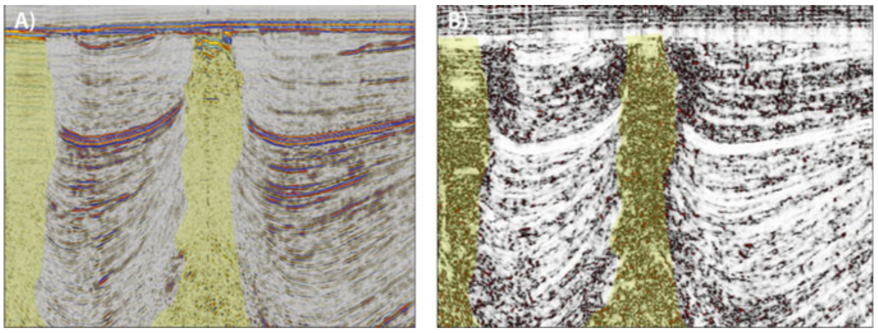


Fig. 10 **a** Normal reflection seismic. **b** Chaos attribute for the same cross-sectional view. Two vertical salt domes are present in the image and are indicated with yellow shade in both **a** and **b**

1.5.8 Curvature

Curvature measures lateral changes in dip and azimuth. It describes how bent a surface is at any given point. A structure with upwards concavity has negative curvature, and upwards convexity results in positive curvature. The direction independent curvature of a surface is defined in three dimensions as the inverse of the radius of a sphere which is tangent to that surface at the point [83]. Curvature of the trend in seismic data can be calculated from the dip/azimuth attributes by considering their derivatives. The right choice of parameters, such as window size, for seismic attributes has not been analyzed much in literature. Hunt et al. [43] perform such an analysis for seismic curvature by comparing the attribute results using different parameters with ground truth data from well logs.

A curvature volume can be used to visualize areas of maximum and minimum curvature, which signals underlying stress-regimes (anticlines and synclines) as well as structural lineaments such as faults. Measures of reflector curvature have been used to map subtle features and predict fractures [21, 36, 64]. Fractures are small scale cracks that develop due to varying intensity and angle of subsurface stress over time and can enhance permeability in reservoirs. Chopra et al. [21] use Rose diagrams to show the distribution of crack orientations and use this to group related fracture networks. Curvature attribute can also be used to better identify sedimentary features like channels and reveal more details from the internal morphology of channels. In many cases structural lineaments are better resolved using a curvature volume rather than a coherence attribute. Coherence and curvature attributes measure two different things and curvature attributes provide more detail in areas where coherence attribute may show no features [20]. Several papers discuss geological cases where curvature has been used [10, 19].

1.5.9 Frequency Filtered Data (Spectral Decomposition)

Spectral decomposition is the process of creating a continuous time-frequency analysis of seismic data (e.g. [14, 22, 72]). Spectral decomposition is converting broad band seismic data into discrete frequencies. This method creates not only one derived volume, but one for each frequency. It is not uncommon to extract frequencies at 1 Hz steps. For a seismic volume typically containing data in the range from 5 to 90 Hz, 85 new volumes are created. Each voxel then has 85 values, which cannot be represented by simply mapping to a colour. Simple approaches have been presented that take three of the frequencies and map them to RGB or HSV colour maps. Handling the full data is a challenge both in terms of effective visualization and storing the data. For each voxel, the frequency response can be considered a curve as a function of the frequency. This curve will have a maximum at the frequency that has highest strength. Liu and Marfurt [54] identify the strongest frequency response for each voxel and map this frequency to a rainbow colour map. The colour is shifted towards black the weaker the response is, for communicating the strength of the frequency. Blumentritt [11] performs a similar dimensional reduction, but instead of using the strength of the strongest frequency, he calculates the difference between the strongest frequency strength and the average strength of all frequencies. This expresses how much the peak is different from the rest of the frequency curve, thus communicating whether the frequency response is rather flat, with a small peak, or if the peak is clear and high. Guo et al. [35] performs a principal component analysis on all frequencies. They then find three major component axes and map them to the RGB colour space. However, the three components no longer represent actual frequencies.

Various time-frequency analyses can be produced from a single seismic trace, which means that the spectral decomposition process is not unique. The outcome of performing frequency decomposition for a frequency band is a scalar magnitude value and a scalar phase response value. The spectral decomposition methods include Discrete Fourier Transform (DFT), Maximum Entropy Method (MEM), Continuous Wavelet Transform (CWT), and Matching Pursuit Decomposition (MPD). DFT and MEM methods use a time window for the analysis and the nature of windowing has significant effect on the resolution of the results. MPD is an advanced and computationally intensive method that has superior temporal and spectral resolution. Sierra et al. [88] give an overview of the different mathematical formulations for performing spectral decomposition and compare them. Spectral decomposition has several applications including layer thickness determination [73] e.g. for identifying river deposits, stratigraphic imaging, and Direct Hydrocarbon Indicators (DHI).

While traditional seismic attributes are recorded over the full bandwidth of the seismic signal, the spectral decomposition technique makes use of the sub-band interferences to capture internal heterogeneities in much more detail. This can include the internal stacking of layers, layer boundaries and thickness variability. After performing spectral decomposition of the seismic signal, the magnitude and phase volumes of individual frequency bands can be investigated to identify where discrete sedimentological features either tune in or out. Here, tuning is used in the same sense as the tuner on the amplifier in your home-stereo or radio: You tune in, or out a signal. Tuning effect, on the other hand refers to the effect the seismic reflection undergoes when the geological layers seismic events (boundaries) are spaced closer than a quarter of the wavelength, at which point the seismic reflection might become subject to either constructive or destructive interference. Resolution is a function of the initial frequency of the seismic signal and energy loss with depth. In other words, the deeper in the subsurface the seismic signal has travelled, the harder it is to accurately image thin geological features [93]. For best vertical resolution, the spectral decomposition must be combined with a matching-pursuit technique to avoid vertical smearing of the data [54].

Spectral decomposition can be applied to identify sedimentary features not observable on broadband seismic data. Figure 11 illustrates an example where original seismic data showed large sandstone bodies with apparent circular trends, indicating a large meander bend of an ancient river. An RGB-blend based on the spectral decomposition into three discrete frequency bands was tuned to capture more subtle channel trends hidden in sub-bands. The RGB blend not only resolves more of the channel-body's geomorphological features, but also reveals a plethora of internal trends within the major channel body. Such heterogeneities may have implications for

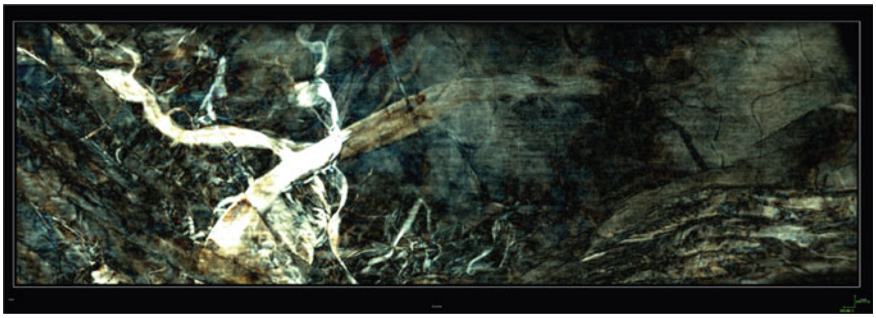


Fig. 11 RGB blend of three spectral decomposition volumes showing a time-slice through a fluvial channel complex. Subtle details become apparent compared to Fig. 3b which shows the same channel system using signal strength attribute

potential hydrocarbon production from the sandstone reservoir rocks. Phase volumes of individual frequencies can also be attributed to some sedimentary facies variations. Thickness determination and facies analysis using spectral decomposition is based on the tuning effect at different frequencies. Identifying the highest (dominant) amplitude events at specific frequencies can help determine the seismic response to certain thicknesses. In thickness mapping using spectral decomposition, the dominant amplitude and dominant frequency maps correspond to the time separation and amplitude maps of conventional thickness mapping. For thicknesses less than tuning thickness, dominant amplitude maps can be used, whereas dominant frequency can be used to quantify thicknesses greater than tuning thickness.

Hydrocarbon saturation in general, and gas saturation in particular, can cause increased amplitude in certain frequencies (mainly lower) which is not visible on the broad band seismic data. Fahmy et al. [29] provided a very good example of a discovery case from west of Africa.

1.6 Facies Terminations

There are considerable differences between seismic facies classifications and sedimentary facies (that might be extended to include sedimentary facies associations). Seismic facies are often associated with large-scale (10–100 m) structures (e.g. clinoforms, Catuneanu et al. [16]), whereas geologically defined facies are often describing centimeter to meter-scale structures that tell of sedimentary processes. To bridge the gap between seismic and geological facies, core and wireline logs are vital in order to specify what has created the seismic signal, but the nature of seismic data gives that the full details of the underground cannot be represented with such data.

Barnes [5] defines seismic facies as a distinct set of seismic reflections characterized by amplitude, spacing, continuity and configuration according to the classification of Mitchum et al. [66]. Seismic facies may also include information that extends

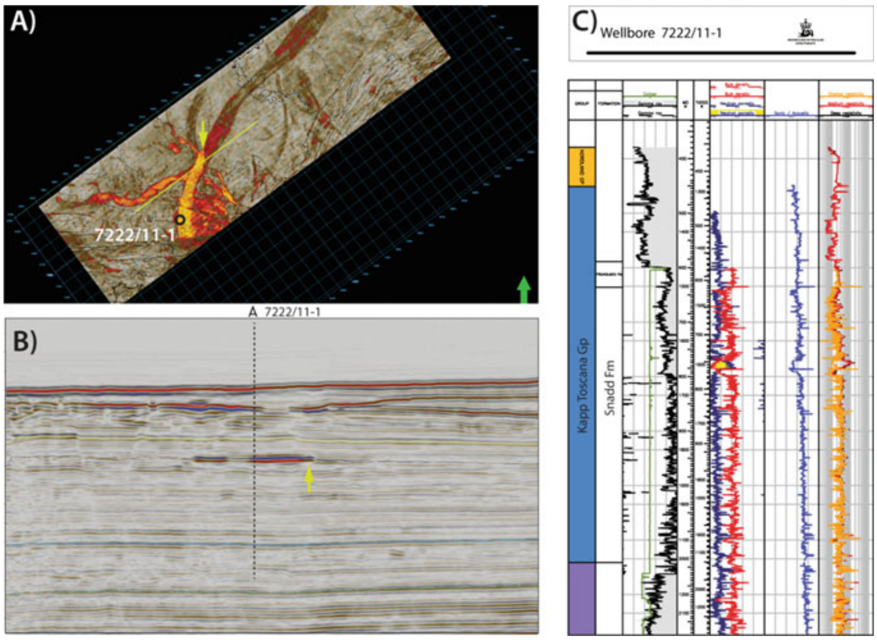


Fig. 12 An example of seismic (**a**, **b**) and petrophysical (**c**) facies combined to prove a hydrocarbon reservoir. **a** Map-view signal strength attribute showing channel geomorphologies (geobodies) with an amplitude anomaly running across the major channel body (indicated by yellow arrow). **b**) Cross-sectional view along the yellow line in A) of the channel body and its amplitude anomaly as indicated by the yellow arrow. **c** Well log (dashed line in B indicates approximate position) through the seismic interval, providing petrophysical information on the amplitude anomaly. Highlighted in yellow, where the density (red) and neutron porosity (blue) traces cross each other, is the hydrocarbon reservoir that causes the amplitude anomaly in **a** and **b**. In this example the amplitude anomaly and petrophysical response represents a gas reservoir

beyond the typical geological facies whenever derived attributes record information on the structural trends (faults, salt domes, etc.) and pore fluids (gas, oil, brine). John et al. [46] offer a similar definition. Geological facies on the other hand, is used to classify sedimentary facies that belong to the same overall depositional process and can be classified as one facies readily distinguished from other depositional processes (or geological facies). Sedimentary facies are the basic building blocks of the sedimentary succession, and at a sub-seismic scale these facies are usually investigated using core or outcrop data. A set of facies can typically be combined in a facies association. An example of such could be channel, beach or aeolian dune deposits, or it can be a carbonate reef. Attribute mapping might in some cases allow one to classify facies associations in 3D seismic data.

Latimer et al. [49] illustrates how seismic attributes can be used to more accurately resolve layer terminations such as pinch outs. An example of different facies boundaries, together constituting a hydrocarbon reservoir, is shown in Fig. 12. The hydrocarbon reservoir is located within a fluvial channel complex shown in red and orange in (a) (which can be classified as one discrete facies). Here, the hydrocarbon saturated part of the reservoir (seen as the yellow amplitude anomaly) is clearly differentiable from the unsaturated part and could therefore justify its own facies classification.

Randen et al. [81] categorize some of the patterns that can be found in facies as chaotic, parallel, wavy and divergent. They then describe mathematically how to calculate attributes that can help identify some of these facies patterns. Lucet and Fournier [55] describe how to extract information such as porosity and permeability from 4D seismic data, i.e. when 3D data is gathered of the same area over time. They perform several seismic surveys over time on a hydrocarbon producing reservoir. This allows identification of reflection differences in areas where the subsurface content has changed. They then categorize these signal differences into groups corresponding to separate facies types.

Much work has also been done in facies identification through texture determination in terms of finding voxels with particular neighborhood configurations [13, 18, 27, 32, 67, 78, 92]. These approaches compare neighboring voxels and correlates between those of similar characteristics, thereby mapping out discrete patterns within the three-dimensional volume. In [33] an overview of these techniques is presented.

2 Object Extraction from Seismic Volumes

Object extraction is closely related to facies classifications and terminations, as the object extracted often might be classified as a discrete facies or facies association. Here we highlight a few typical objects that can be extracted in seismic data. This section briefly discusses algorithms that take seismic volumetric attributes as input, traverses the data and extracts objects such as horizons, faults or geobodies in the form of surfaces or solids. Figure 13 shows in different colours, an example of extracted geobodies (in this example, channelised deposits). The user selected a "seed point" on each geobody and an algorithm identified voxels of similar signal strength connected to the seed point.

2.1 Geologic Background

Which attributes to use during the seismic mapping depends on scale (e.g. looking at kilometer-scale fault systems, or meter-scale depositional elements). Acquisition of 3D seismic surveys for exploration purpose has significantly increased over the last decade. 2D seismic surveys are only used in the very early stages of screening. A very useful feature of 3D seismic is the dense sampling of data over the area of interest, giving more accurate spatial representation of subsurface features, and multi-azimuth 3D seismic further improves imaging in all directions. There are three factors that control our ability to recognize features or objects in an image: noise,

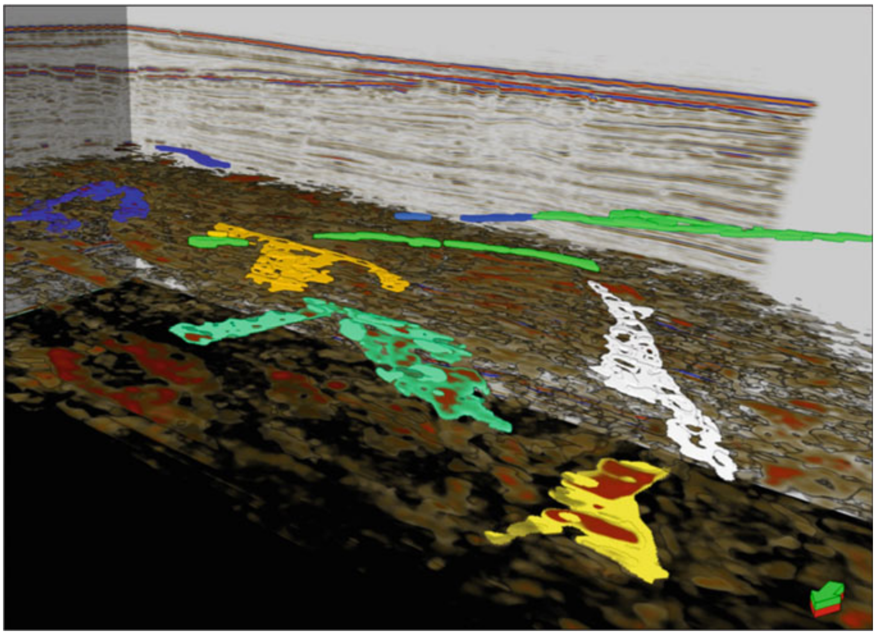


Fig. 13 Result of geobody extraction of channelised sandstone geometries in a 3D seismic survey. Here, geobody extraction is based on high-amplitude seismic reflections, seen in the backdrop as red to brown blobs

contrast, and scale. We have addressed noise earlier in this chapter. Noise will affect the clarity of the features which can be perceived on the seismic data.

Contrast is the difference in the reflectivity of features of interest with the background reflectivity. Faults are for example identified based on changes in pattern or geometry of reflectors, while amplitude anomalies are interpreted through detection of changes in reflection signal strength. In extracting geological objects from 3D seismic data, enhancing the contrast is important for better highlighting the objects and for segmenting the object from the background. A clear difference in properties of the object or geologic feature from the background properties will increase the reliability of the isolation or segmentation process. When the presence of an object is defined by a change in signal character it is necessary to go beyond traditional trace based seismic attributes and measure attributes that correlate with changes in the patterns in the image.

Scale refers to the relative size of the geological objects of interest. The filter sizes during data processing for generating attributes influence the scale of the detectable objects. The filter size controls the smoothness or sharpness of the results. Figure 8 showed an example with two different filter sizes used to image faults of different scales. In cases where there is not a priori information about the possible size of geological objects within the studied area, a range of filter sizes should be applied to increase the probability of capturing most of the geological objects.

Three-dimensional visualization has played an important role in seismic interpretation in recent years and has become a core component of the mapping workflow. Recent advances in workstation hardware capabilities and graphic card designs allow 3D seismic visualization applications to operate in high and true colour modes for realistic quantities of data. This facilitates real time visualization of 16-bit (or higher) seismic volumes on interpretation workstations and has led to the introduction of more sophisticated and higher-fidelity multivolume display techniques [40].

Seismic mapping techniques and workflows are still limited by the seismic resolution. Geological features must be of a certain size (considering both lateral and vertical seismic resolution) relative to its depth and the frequency of the seismic signal in order for them to be resolved in any given dataset. Seismically resolvable geological features might include salt domes; gas chimneys; fault surfaces and their associated damage zone; clinoforms at various scales; carbonate reefs and karst structures; beaches and desert sand dunes; and channels both deep marine and fluvial. Stray meteorite impacts and large glacial erosion surface, and their associated glacial-till deposits, that occur in certain areas and time periods can also be detected. To complicate the picture, erosion plays a major role in what part of and how much is preserved of an original geobody. As elaborated on earlier, such features are often divided into discrete facies and facies associations. Reading [85] provides a thorough overview of sedimentary facies and their depositional environments.

The subsurface has a complex structure, and it is the job of geoscientists to decipher it. Attempts towards automating parts of the process, and especially new techniques that help visualize and illustrate geological features are continuously added and improved in interpretation software. The ultimate goal for the geoscientist is to be able to communicate ideas and interpretations as accurately and fast as possible; both to his or her peers, for rapid feedback and new iterations, but also to decision-makers and engineers. Object extraction is especially useful in cases where geobodies can be extracted from the dataset and be shown in relation to other geobodies or data of interest.

Jacquemyn et al. [45] present methods where a user can easily and directly sketch, or draw the 3D structures he/she wants to express without relying on extraction algorithms. Such so called sketch based techniques are a relatively new and potentially very useful way for interpreters to quickly segment seismic volumes, create illustrations or to discuss alternative scenarios.

2.2 *Extraction Algorithms for Horizon Surfaces*

Horizons represent boundaries in the stratigraphy. They are a set of points in space that are associated with the onset of a discrete geological layer. These layers (or formation boundaries, as explained in the introductory section) can be traced out as discrete horizons when it is found necessary. The computer-based workflow of 3D seismic mapping to generate horizons is different from geobody extraction using 3D seismic attributes. Horizons are planar-like surfaces and have spatial and temporal extent within a seismic volume. Horizons are surfaces with no *body* per se, as opposed to volumetric geobodies. An introduction to classical horizon and fault extraction methods is given by Pepper and Bejarano [77].

Extraction of horizons is usually done using the original seismic reflection volumes. However, with the recent advances in seismic attribute analysis, the use of additional seismic attribute volumes has improved horizon extraction. Several algorithms for extracting horizons work by letting the user define a point on the horizon that he/she wishes to interpret, called the seed point, followed by an automatic segmentation of the horizon, called growing. During growing, more points will be added to the horizon segmentation and it will grow outwards from the seed point(s). For each iteration, any neighboring point to the so far segmented horizon will be checked if they satisfy the autotracking criterion. If so, they will be added to the horizon segmentation. The growing criterion can be simple, such as that its voxel value must be within a certain interval around the voxel value of the seed point. Alternatively, it can be more advanced, such as that the trace in a specified vertical radius must have a minimum similarity to the trace of the seed point [15]. Another common criterion is that the voxel is on a maximum positive (peak), maximum negative (trough) or zero amplitude (zero crossings) of the trace. This fundamental step of horizon mapping is referred to as the autotracking process and is standard in any modern interpretation software. It starts with defining a certain number of seed points that has high confidence. The program uses the seed points and the amplitude threshold range given to the event together with the type of seismic event (i.e. maximum, minimum or zero crossing) and searches the whole seismic volume for similar points (e.g. Sheffield et al. [86]).

Certain attribute volumes can help to guide the interpreter in uncertain areas. First, noise filtering is often used as a starting point. Structurally oriented noise removal applications explained in this chapter improves the clarity and coherency of the seismic volume, hence the autotracking works better. The phase attribute can help highlight the lateral continuation of more subtle seismic signals, and conversely the signal strength attribute can be used if the horizons in question are only the strongest ones. The coherence and curvature attributes may be effective in pointing out discontinuities thereby giving the interpreter a warning on where the reflections/horizons are cut by faults and need to be treated with greater care. The choice of seismic character to be picked for each horizon can be determined using seismic-to-well-tie analysis. In this analysis, the link between the horizons or stratigraphy and seismic signals at a well location are defined through 1D forward seismic modeling. Maximum, minimum and zero crossings are three main choices. In higher resolution interpretation studies, the exact location of horizons on the seismic signal can be used to guide the autotracking process.

A typical workflow (see Fig. 14) would start with identifying an important, regionally traceable and preferably well-tied horizon (a horizon that intersect a well log). This horizon is first interpreted on 2D seismic cross-sections with appreciable spacing. Then a coarse surface (based on relatively few data points or 2D line mapping) can be gridded from that horizon, onto which a variance attribute can be extracted as to highlight the faults and lateral discontinuity. This is done to avoid

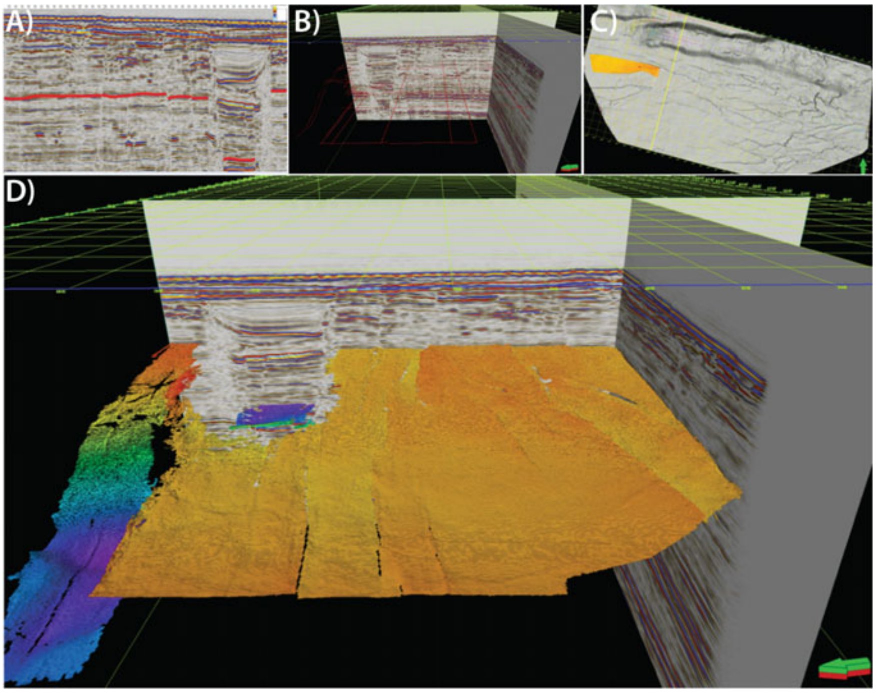


Fig. 14 Horizon mapping workflow: **a** picking a horizon (red) on several, regularly spaced 2D transects in the 3D volumes (location of transect example in **a** is shown with yellow line in **c**; **b** the sparse grid (red lines) is manually interpreted horizon lines used to make a coarse gridded surface shown in **c**; **c** Map-view autotracking based on the sparse grid, guided by a coherence attribute map. **d** autotracked horizon interpretation. The colour on the autotracked horizon reflects its relative Z-values of the horizon. A rainbow colour table is used from red for high Z values to purple for low Z values, here the interpreted horizon shows a horst and graben structure (high and low areas)

having to calculate the variance attribute for the whole volume. With this coarse attribute surface as backdrop, one can autotrack the horizon based on the sparse grid used as surface-input. Doing so will allow the interpreter to guide the autotracking in tricky areas and stay clear of faults for as long as possible. More detailed work can later be done on the horizon very close to the fault and fault zones. It is also easy to identify problematic and wrongful autotracking by toggling local z-value colouring of the horizon (leftmost horizon in Fig. 14). This will tell if the autotrack has jumped to another reflection.

In the work by Borgos et al. [12], for each trace, all peak samples are identified. Such samples will have higher values than their immediate neighbors above and below and can be considered as horizon candidates. The waveform shape of the trace around each such sample is studied. This waveform is then described mathematically by a limited number of coefficients. Finally, the coefficients are clustered into a user defined number of classes. As a result, horizons of similar trace neighborhood are assigned to the same class. Since different samples on a specific horizon typically have a very similar trace neighborhood, the method can segment out separate horizons. However, this method is sensitive to noise and fails for horizons with varying trace (seismic character). An advantage of this method compared to growing methods is that new horizons are traced out across faults for which growing methods would fail due to the lack of connectivity. Also, the user does not need to perform an explicit seeding for each disconnected part of a horizon. A more advanced and robust horizon extraction method is presented by Wu and Fomel [97]. They also compare waveform shapes so they can trace out horizons that are not connected due to e.g. faults. They use the Dynamic Time Warping method for measuring waveform similarity. Intermediate results are improved using an iterative least-squares optimization.

Other approaches [65, 74] may in some cases relieve the user for the need to perform horizon point seeding, and subsequently waiting for the horizon growing to finish. In these approaches, a time-consuming preprocessing is performed where all voxels in the volume are seeded and grown. All the grown surfaces are then stored, and the user can later interactively choose among them. This method has the advantage of allowing the user to filter on measures that are not known until the horizon is grown, such as average horizon dip/azimuth or strength.

2.2.1 Extraction of Relative Time

Algorithms similar to the ones above, that are able to track horizons across discontinuities such as faults, can be used to generate a stack of horizons from which one can derive a so called Relative Geologic Time (RGT) volume. A horizon is a chronostratigraphic surface. This means that all locations on a horizon represent material deposited at the same time. According to Steno's law of superposition, a horizon that is below another one is older. Therefore, a stack of high quality extracted horizons can be used to create a time parameterization of the seismic volume. Since the actual age of each horizons is typically not known, the parameterization is only relative. Such an RGT volume is a powerful representation. It can be used to smoothly move a virtual horizon in the space between successive or nonsuccessive horizons. It can be used to create volumetric bodies representing intervals of time. It can also be used to flatten the whole seismic volume so that all horizons become straight. This makes it easier to identify subtle irregularities on the horizons, possibly produced by channels, and is essential for reconstructing and accurately assessing ancient clinoform geometries [47]. For an overview of the different approaches to time-parameterizing a seismic volume, see the review paper by Qayyum et al. [79].

2.3 Extraction Algorithms for Fault Surfaces

A fault surfaces is a discontinuity plane. It is the surface across which distinct reflections are offset or translated relative to the direction and magnitude of the fault (Fig. 15). The notion of a fault surface is a simplification. Faults are 3D features

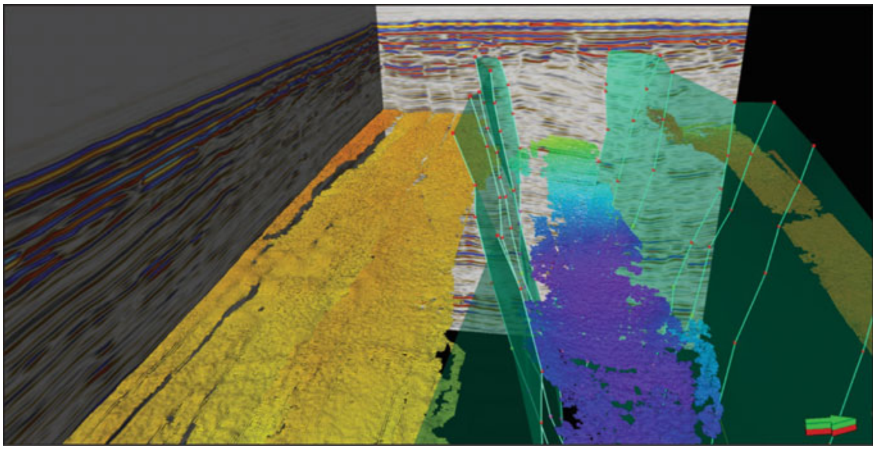


Fig. 15 Fault surfaces seen as semitransparent turquoise, delineating a graben system (downfaulted block of sedimentary strata). Note how the autotracked horizon terminates at or near the fault surface. More accurate mapping is needed to complete the horizons close to the fault. Vertical exaggeration is 5× and the colour on the autotracked horizon reflects its relative Z-values (red is shallow, purple is deep). Breaks or gaps with distinct steps in the autotracked horizon indicate smaller faults that can also be seen in the backdrop 2D seismic data

that appear and disappear and link up or die out relative to one another, and they are very often associated with a damage zone [84]. This zone is defined by the relative distance to the main slip face over which the frictional forces of the displacement is felt, taken up as destruction or even offset. The damage zone can also include numerous smaller faults that grew into one major fault which has taken up most of the offset, given by the relative magnitude of the forces responsible for creating the fault and the characteristics of the lithology in which the fault is propagating. Such damage zones often complicate the mapping of horizons close to the fault surface. With the notable exception of shallow thrust faults and low angle detachment zones, most faults are steep to almost vertical features. For this reason, they are easily resolved in the seismic data. As noted earlier in this section, faults can be highlighted by various attributes such as coherence, 3D curvature volumes, and possibly the chaos attribute. Discontinuity attributes such as coherence and other geometric attributes are local measures of waveform shape variation, change in amplitude and dip and azimuth. Human interpreters organize such features into fault planes or any other geologic feature that fit their geological model. Image processors are at work to help accelerate this mapping process (e.g. [20]). Mapping of faults on 3D seismic data is still a time-consuming task.

Randen et al. [82] present a workflow for extracting fault surfaces. Fault sensitive attributes are used to create a fault probability volume so that faults can be extracted as connected components. They separate faults with different dips into separate attribute volumes so that the connected component growing does not merge faults that intersect each other. Pedersen et al. [76] describe a fault extraction method that uses ant tracking to create a fault probability volume from which fault segments are extracted. The focus of their paper is an interactive and visually steered method for removing and combining the pre-computed fault segments by filtering them based on some measures such as their accumulated fault strength, average angles and size. Jacquemin and Mallet [44] detect planar faults using the Hough line-transform on each XZ-plane (where Z is depth). The method results in a new plane consisting of points, where each point represents a line/fault. This is carried out for all XZ-planes that result in a new volume, consisting of lines in 3D. Then a new Hough transform is performed to map these lines into points in 3D. Now each 3D point represents a 3D plane (dip, azimuth, and minimum distance to the origin). Then all voxels in the coherence cube that map to such a point (i.e. has support in its neighborhood with regards to the degree of faulting) are marked. Finally, marked voxels for each point are triangulated into fault surfaces. Hale [37] presented a robust and widely used fault detection algorithm that in essence extracts fault surface meshes from a coherence attribute. The coherence attribute describes similarity between adjacent seismic traces and the fault surfaces are extracted along the ridges of minimum coherence. More recent fault extraction algorithms include Wu and Fomel [96] using optimal surface voting and Noorim et al. [70] using Gaussian process regression. Di and Gao [26] give a review of computer-aided automatic and semi-automatic fault extraction.

2.4 Extraction Algorithms for Salt Bodies

Salt, where present, often play an important role in influencing the structural and depositional history of different sedimentary basins. Salt structures have also received a lot of attention because they tend to control hydrocarbon traps in the strata affected by salt movement (e.g. the Ekofisk Field of the Norwegian North Sea). Salt are either found in situ in layered rocks or moved due to combined gravity/tectonic movements forming salt domes with irregular shapes. Seismic images of salt domes (Fig. 16a) are usually complex and mapping the salt outline and the salt-sediment boundary can be a challenge for interpreters.

Due to the chaotic nature of the salt domes, the chaos attribute becomes a natural candidate for interpreting such structures. For the purpose of automating the process of extracting such features, it is imperative to resolve their boundaries. This is not an easy task, especially if the salt body has some horizontal component (or overhang) that creates a shadow of the seismic signal below. As shown earlier (under the chaos attribute section), there are several tools available for delineating the salt dome boundaries. A simple and quick way of doing so in the map-view is also to extract reliable horizons from reflectors around these features, as illustrated in Fig. 16b. A challenge when mapping the actual salt body is the overhanging geometry of the salt, which inherently cause multiple Z values for each XY coordinate. Another important aspect of salt domes for the interpreter is the velocity pull-up caused by the higher velocity salt compared to the lower velocity surrounding siliciclastic rocks.

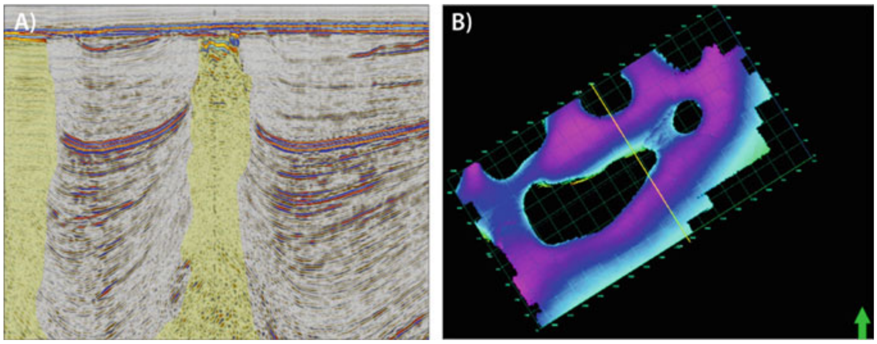


Fig. 16 **a** Salt domes coloured yellow seen in a cross-section. **b** Autotracked horizon around salt domes seen in map-view. Holes are shown where salt domes are, due to the inability to autotrack the horizon there. Yellow line shows the location of the left image. Purple colour indicate low areas, whereas high areas are green

The difference in the velocity of the pressure wave will cause the energy to arrive at the base of a thick salt dome relatively earlier than the energy traveling through a 'slow' medium. This implies that the base of this thick salt dome is placed disproportionally higher than it should be and results in reflectors below the salt to be "pulled up" and depicted closer to the surface than reflectors that are not below the salt.

Lomask et al. [52] present an image segmentation solution for delineating salt boundaries directly on the seismic (amplitude) data, using normalized cuts. Normalized cut methods take an image and assign each pixel as a node in a graph. Then each node is attached with an edge to its immediate neighbor nodes. Each edge is associated with a weight corresponding to some similarity measure between the nodes. The image is then segmented by finding minimum cuts, which are closed curves that create image segments, where the sum of the weights of the crossed edges is minimized. The result is well constrained segments. Halpert et al. [38] discuss how to combine several attributes, such as signal strength and dip to create a new attribute that better identifies salt. Wu et al. [95] presented a semiautomatic method for extracting salt boundaries. The signal strength attribute is used as it is typically high at the boundary of salt bodies since speed of sound is high in their interior. The algorithm operates on a single seismic slice at a time. Based on a seedpoint, a curve is calculated that tries to intersect as many high valued samples as possible using mathematical optimization.

3 Seismic Attributes and Machine Learning

Given the latest developments in seismic attribute analysis using Machine Learning (ML) approaches, we will finish this chapter by very briefly touching upon ML-based seismic attributes. ML algorithms have been increasingly used to generate different types of seismic attributes, and has the potential to significantly ease the interpretation process. ML approaches can be classified in two groups: unsupervised and supervised. In unsupervised ML, input seismic data is clustered into different classes and further geological constrain can be used to assign different classes into certain geological objects. In supervised approach, input seismic data (which is called ML features) is trained with labels (target training data or ground truth) to predict target features. Training data or labels can be human made interpretations. Examples of target objects are faults, horizons, sedimentary features such as channels, canyons, reefs, prograding systems, and injectites.

The quality of ML based seismic attributes in supervised ML depends on: (i) the quality of features or seismic data, (ii) quality and quantity of training labels, and (iii) choice of algorithms. Like traditional seismic attribute analysis, noise reduction can be applied on seismic data prior to ML attribute analysis in order to enhance the signal to noise ratio. In ML based attribute studies, we are not limited to use one seismic volume in training, several volumes can be included. For example, different band-pass seismic volumes (frequency filtered attributes) or any set of derived attributes can be used in training. In cases where one of the volumes contain higher noise, using multiple volumes in training will reduce the effect of noise in the attribute analysis. The quality and quantity of labels can influence the final prediction results. For example in a fault prediction study using ML on a survey with 500 in-lines and cross lines with grid spacing of 12.5 by 12.5 m, very few in-lines or cross-lines with fault labels (less than 10) will probably not be sufficient to build robust ML models to predict faults. It is not easy to give a threshold for the required number of in-lines or cross-lines with labels and it depends on the complexity of target and input seismic data quality. The last element that influences the quality of ML based attributes are the algorithms and associated hyperparameters. The possibility to share new algorithms through open source libraries enables a much wider group of users to test different algorithms. Neural network algorithms and Deep Learning (e.g., [50]) has been one of the main reasons that ML approaches received huge attention among many disciplines over the past few years. Neural networks are adaptive systems that project a vector of an input space (e.g., seismic volumes) to the corresponding vector in target space (for example images of faults, channels, reefs etc.). Deep Neural Network (DNN) algorithms can be used for classification or segmentation of seismic data. The classes correspond to different geological targets such as faults, injectites, or salt bodies. In the following we give few examples of seismic attributes generated using ML algorithms.

The first ML attribute example is about fault interpretation. Automatic interpretation of faults has been one of the challenging tasks of seismic interpretation. Two examples are shown with data from the Norwegian Continental Shelf. Figure 17 shows results of fault attributes from Arenaria area in the Norwegian Barents Sea using DNN. We have used fault labels (manual interpretation of faults) from 15 in-lines for training. Model performance has been verified by holding certain in-lines with manual interpretation.

Another example is given in Fig. 18 from the Norwegian Sea. In this example we used pretrained ML approach (e.g. [69, 94]). In this approach, a series of synthetic

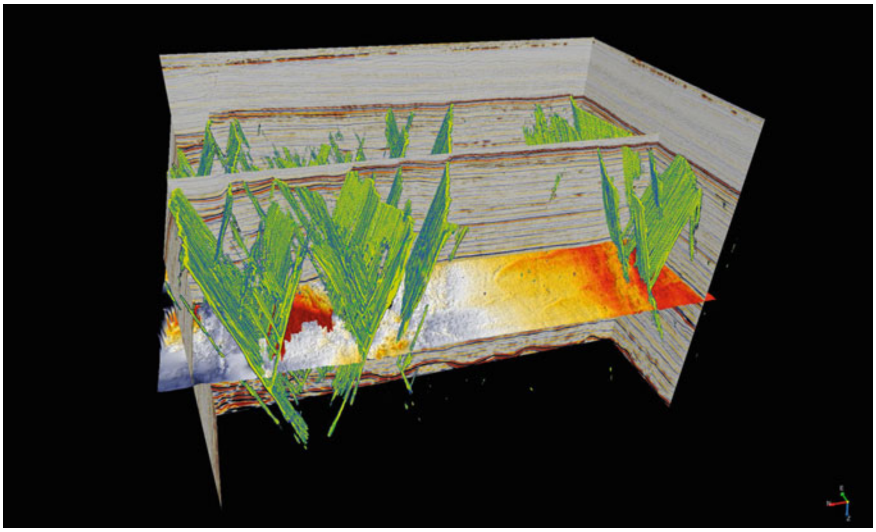


Fig. 17 Fault attributes from the Norwegian Barents Sea using Deep Learning Model trained on manual fault picks on 15 in-lines

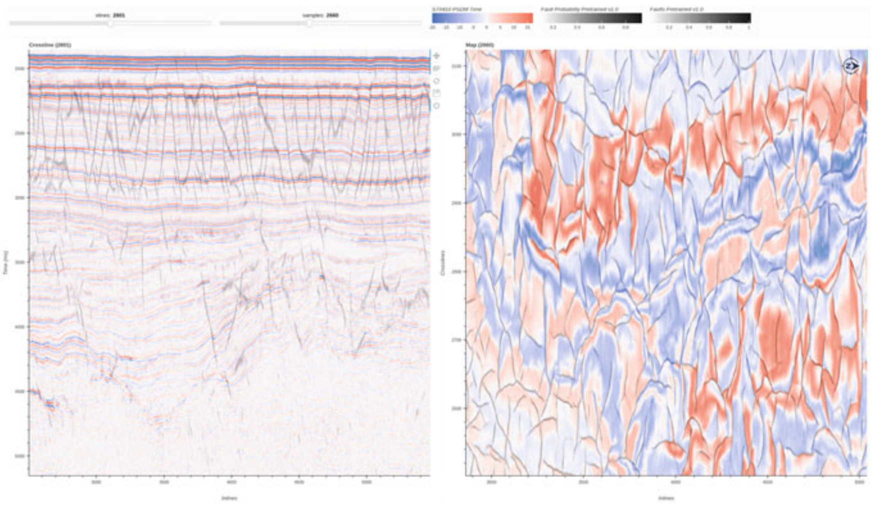


Fig. 18 Fault attribute calculated on real seismic data (Norwegian Sea) using 3D deep learning model trained on synthetic seismic data. Seismic data shown with red-white-blue colour map. Faults are shown with dark curves, where darker colour signifies higher probability of fault. No manual labelling was carried out to generate this fault attribute volume [1]. Left: time-slice view. Right: a cross-line

seismic data is generated that represents the geological settings of the target 3D survey area. The learning from faults on synthetic data is used to identify faults in the real data. This approach is of type supervised ML but does not need human generated fault labels since the labels were generated from synthetic data.

ML approaches can be used to classify sedimentary features. Figure 19 shows an example of predicting geobodies in the Parihaka seismic data [71]. The Parihaka dataset is a public seismic volume provided by New Zealand Crown Minerals. Manual labeling has been done on a certain number of cross-lines (right) and predictions is shown on the in-lines (Left). Following the extraction of several geobodies, one shallow gully (pink feature shown in cross-line and in-line) is singled out for further imaging. Figure 19 bottom shows the selected gully in time-slice (left) and cross-line (right) overlaid on the original seismic data.

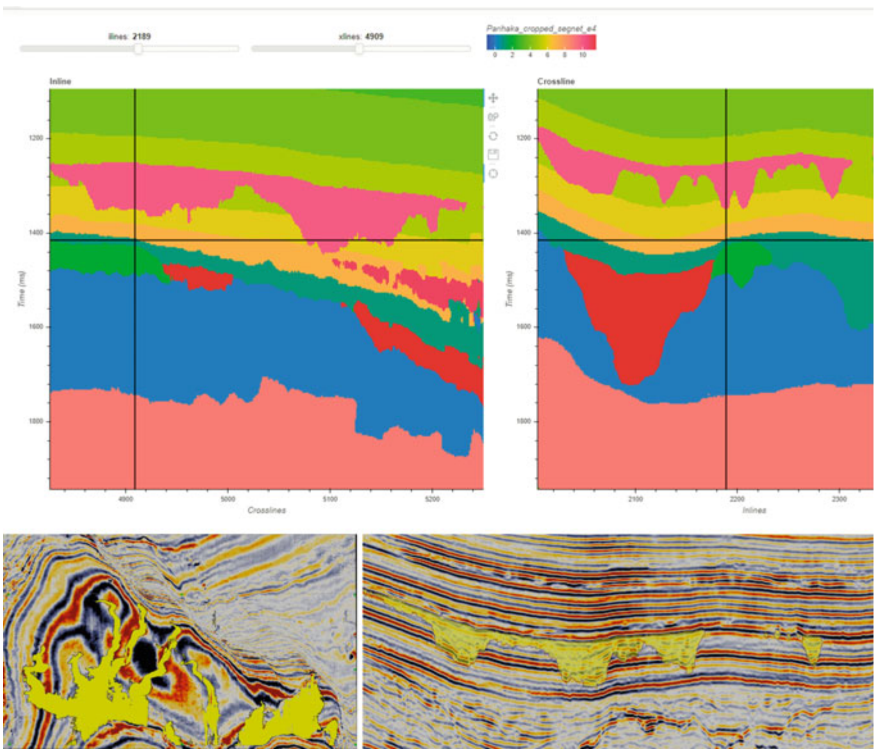


Fig. 19 Geobody interpretation using deep learning. Top left and right: Geobody segmentation of seismic data Bottom: The pink shallow gully from top segmentation is shown in yellow on the seismic data on a time-slice (left) and cross-section (right) [80]

4 Summary

This chapter provides an overview of some key topics related to the practice of seismic mapping. It is not exhaustive, and importantly there are several more in-depth sources on this topic (e.g. [8, 20, 53, 89]). Also, different practices are exercised by different geoscientist and workflows vary from software to software. The chapter attempts to reason for the use of seismic data and explain the various techniques commonly used for mapping the subsurface, backed by case-specific illustrations. We have only brushed upon the field of seismic attributes and mapping.

There is an overall trend towards ever more automated extraction techniques, and as computer power steadily increases it allows for more thorough computations. Identifying seismic signals and intervals of interest, tracing these and understanding their evolution both in terms of how they were originally emplaced and what affected them thereafter is of critical importance for exploration geoscientists. Reservoir engineers that are generally concerned with how the reservoir is producing, prioritize more accurate volumetric and petrophysical parameters in context with reservoir performance over time. For this purpose, 4D seismic and reservoir modelling is more important than large scale geological models. In all practices and disciplines, conceptual models help elucidate on areas, intervals or details not covered by present dataset, or areas that cannot be accurately resolved. In such cases, thorough conceptual models can be useful to help understand the reservoir qualitatively, and constant refinement of these models help constrain the uncertainty before any decisions are made. Therefore, a variety of automated processes that reduce mapping time will allow for faster workflows and more iterations over the same model. With a growing number of datasets and faster turnaround time, these tools are a significant help to the explorer or producer. Machine Learning is the necessary next step in this development, and what we consider the emerging trend in geoscience.

References

1. B. Alaei, S. Purves, E. Larsen, D. Oikonomou, Machine learning assisted seismic interpretation: an integrated workflow for structural/stratigraphic interpretation, combined with reservoir characterisation, in *SEG Advances in Seismic Interpretation Workshop*, Muscat (2019)
2. S. Al-Dossary, Y.E. Wang, *Structure Preserving Smoothing for 3D Seismic Attributes* (SEG San Antonio, 2011)
3. M.S. Bahorich, S.L. Farmer, 3-D seismic coherency for faults and stratigraphic features. Lead. Edge **14**, 1053–1058 (1995)
4. P. Baker, Image structural analysis for seismic interpretation, Ph.D. thesis, Delft University of Technology (2003)
5. A.E. Barnes, Attributes for automated seismic facies analysis, *70th Annual International Meeting*. SEG Expanded Abstracts, 2000, pp. 553–556
6. A.E. Barnes, Fractal analysis of fault attributes derived from seismic discontinuity data, in *67th Annual Conference and Exhibition*. EAGE Extended Abstracts, 2005, p. 318
7. A. Barnes, Redundant and useless seismic attributes. Geophysics **72**(3), P33–P38 (2007)

8. A.E. Barnes (ed.), *Handbook of Poststack Seismic Attributes* (Society of Exploration Geophysicists, 2016)
9. C.E. Bond, A.D. Gibbs, Z.K. Shipton, S. Jones, What do you think this is? Conceptual uncertainty in geoscience interpretation. GSA Today **17**(11), 4 (2007)
10. C. Blumentritt, K. Marfurt, E. Sullivan, Volume-based curvature computations illuminate fracture orientations—early to mid-Paleozoic, Central Basin Platform, West Texas. Geophysics **71** (2006)
11. C.H. Blumentritt, Highlight volumes: Reducing the burden in interpreting spectral decomposition data. Lead. Edge 27 (2008)
12. H. Borgos, T. Skov, T. Randen, L. Sønneland, Automated geometry extraction from 3D seismic data. SEG Expanded Abstracts **22** (2006)
13. A. Carrillat, T. Randen, L. Sønneland, *Automated Mapping of Carbonate Mounds Using 3D seismic Texture Attributes* (SEG, 2002)
14. J.P. Castagna, Comparison of spectral decomposition methods. First Break **24**, 75–79 (2006)
15. L. Castanie, B. Levy, F. Bosquet, Volumeexplorer: roaming large volumes to couple visualization and data processing for oil and gas exploration, in *Proceedings of IEEE Visualization* (2005)
16. O. Catuneanu, V. Abreu, J.P. Bhattacharya, M.D. Blum, R.W. Dalrymple, P.G. Eriksson, C.R. Fielding, W.L. Fisher, W.E. Galloway, M.R. Gibling, K.A. Giles, J.M. Holbrook, R. Jordan, C.G.S.C. Kendall, B. Macurda, O.J. Martinsen, A.D. Miall, J.E. Neal, D. Nummedal, L. Pomar, H.W. Posamentier, B.R. Pratt, J.F. Sarg, K.W. Shanley, R.J. Steel, A. Strasser, M.E. Tucker, C. Winker, Towards the standardization of sequence stratigraphy. Earth Sci. Rev. **92**, 1–33 (2009)
17. S. Chopra, K.J. Marfurt, Seismic attributes—a historical perspective. Geophysics **70** (2005)
18. S. Chopra, V. Alexeev, Application of texture attribute analysis to 3D seismic data. Lead. Edge **25** (2006)
19. S. Chopra, K.J. Marfurt, *Volumetric Curvature Attributes Adding Value to 3D Seismic Data Interpretation* (SEG San Antonio, 2007)
20. S. Chopra, K.J. Marfurt, Seismic attributes for prospect identification and reservoir characterization, SEG Geophysical Developments Series No. 11 (2008)
21. S. Chopra, K.J. Marfurt, H.T. Mai, *Using 3D Rose Diagrams for Correlation of Seismic Fracture Lineaments with Similar Lineaments from Attributes and Well Log Data* (SEG Houston, 2009)
22. S. Chopra, K.J. Marfurt, Spectral decomposition and spectral balancing of seismic data. Leading Edge (2016)
23. S. Chopra, K.J. Marfurt, Coherence attribute applications on seismic data in various guises—Part 2. Interpretation (2018)
24. J.F. Claerbout, *Imaging the Earth's Interior* (Blackwell Scientific Publications, 1985)
25. O. Davogustto, K.J. Marfurt, Removing acqusition footprint from legacy data volumes. SEG San Antonio (2011)
26. H. Di, D. Gao, Seismic attribute-aided fault detection in petroleum industry: a review (Chapter 3), in *Fault Detection: Methods, Applications and Technology*, ed. by D. Martins (2016)
27. H. Di, D. Gao, Nonlinear gray-level co-occurrence matrix texture analysis for improved seismic facies interpretation. Interpretation (2017)
28. M. Donias, C. David, Y. Berthoumieu, O. Lavialle, S. Guillon, N. Keskes, New fault attribute based on robust directional scheme. Geophysics **72** (2007)
29. W.A. Fahmy, G. Matteucci, J. Parks, M. Matheney, J. Zhang, Extending the limits of technology to explore below the DHI floor; successful application of spectral decomposition to delineate DHI's previously unseen on seismic data, in *78th International Annual Meeting*, SEG, Expanded Abstracts (2008), pp. 408–412
30. G. Fehmers, C. Hoecker, Fast structural Interpretation with structure-oriented filtering. Geophysics **68** (2003)
31. P.T. Gabrielsen, I. Brevik, R. Mittet, L.O. Løseth, Investigating the exploration potential for 3D CSEM using a calibration survey over the Troll Field. First Break **27**(6) (2009)
32. D. Gao, Volume extraction for 3D seismic visualization. Geophysics **68** (2003)

33. D. Gao, Latest developments in seismic texture analysis for subsurface structure, facies, and reservoir characterization: a review. Geophysics **76**(2) (2011)
34. Gibson, D., Spann, M., Turner, J., 2003. Automatic Fault Detection for 3D Seismic Data, in *Proceedings of 7th Digital Image Computing: Techniques and Applications*, Sydney
35. H. Guo, K.J. Marfurt, J. Liu, Principal component analysis. Geophysics **74**(4) (2009)
36. H. Guo, K.J. Marfurt, S. Nissen, C.E. Sullivan, Visualization and characterization of structural deformation fabric and velocity anisotropy. Lead. Edge (2010)
37. D. Hale, Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3D seismic images. Geophysics (2013)
38. A. Halpert, R. Clapp, B. Biondi, *Seismic Image Segmentation with Multiple Attributes* (SEG Houston, 2009)
39. S. Helmore, Dealing with the noise – improving seismic whitening and seismic inversion workflows using frequency split structurally oriented filters. SEG Expanded Abstracts **79** (2009)
40. J. Henderson, S.J. Purves, G. Fisher, C. Leppard, Delineation of geological elements from RGB colour blending of seismic attribute volumes. Lead. Edge **27**(3), 342–350 (2008)
41. J. Hesthammer, M. Landrø, H. Fossen, Use and abuse of seismic data in reservoir characterisation. Mar. Pet. Geol. **18**(5), 635–655 (2001)
42. C. Hoecker, G. Fehmers, Fast structural interpretation with structure-oriented filtering. Lead. Edge **21**, 2002 (2002)
43. L. Hunt, B. Beshry, S. Chopra, C. Webster, Determination of target-oriented parameters for computation of curvature attributes. Interpretation (2018)
44. P. Jacquemin, J.-L. Mallet, *Automatic Faults Extraction using double Hough Transform* (SEG Houston, 2005)
45. C. Jacquemyn, M. Pataki, G. Hampson, M. Jackson, D. Petrovskyy, S. Geiger, C. Marques, J. Machado Silva, S. Judice, F. Rahman, M.C. Sousa, Sketch-based interface and modelling of stratigraphy and structure in three dimensions. J. Geol. Soc. (2021)
46. A.K. John, L.W. Lake, C. Torres-Verdin, S. Srinivasan, Seismic facies identification and classification using simple statistics. SPE Reserv. Eval. Eng. **11** (2008)
47. T.G. Klausen, W. Helland-Hansen, Methods for restoring and describing ancient clinoform surfaces. J. Sedimentary Res. **88** (2018)
48. E. Labrunye, J.-L. Mallet, Improving seismic structural information using trigonometric interpolation. SEG Expanded Abstracts **23** (2004)
49. R.B. Latimer, R. Davidson, P. van Riel, An interpreter's guide to understanding and working with seismic-derived acoustic impedance data. Lead. Edge **19** (2000)
50. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)
51. A. Libak, B. Alaei, A. Torabi, Fault visualization and identification in fault seismic attribute volumes: implications for fault geometric characterization. Interpretation **5**(2) (2017)
52. J. Lomask, R. Clapp, B. Biondi, Application of image segmentation to tracking 3D salt boundaries. Geophysics **72** (2007)
53. L.R. Lines, R.T. Newrick, *Fundamentals of Geophysical Interpretation* (No. 13) (Society of Exploration Geophysicists, 2004)
54. J. Liu, K.J. Marfurt, Instantaneous spectral attributes to detect channels. Geophysics **72** (2007)
55. N. Lucet, F. Fournier, *4D Data Interpretation Through Seismic Facies Analysis* (SEG Expanded Abstracts, 2001)
56. Y. Luo, W.G. Higgs, W.S. Kowalik, Edge detection and stratigraphic analysis using 3D seismic data, in *66th Annual International Meeting* (SEG Expanded Abstracts, 1996)
57. Y. Luo, S. Al-Dossary, M. Alfaraj, Edge-preserving smoothing and applications. Lead. Edge **21** (2002)
58. Y. Luo, Y. Wang, N. al Bin Hassan, M. Alfaraj, Computation of dips and azimuths with weighted structural tensor approach. Geophysics **71** (2006)
59. Y. Lou, B. Zhang, R. Wang, T. Lin, D. Cao, Seismic fault attribute estimation using a local fault model. Geophysics **84** (2019)
60. K.J. Marfurt, R.L. Kirlin, S.H. Farmer, M.S. Bahorich, 3-D seismic attributes using a running window semblance-based algorithm. Geophysics **63**, 1150–1165 (1998)

61. K.J. Marfurt, V. Sudhakar, A. Gersztenkorn, K.D. Crawford, S.E. Nissen, Coherency calculations in the presence of structural dip. Geophysics **64**, 104–111 (1999)
62. K.J. Marfurt, Robust estimates of 3D reflector dip and azimuth. Geophysics **71** (2006)
63. K.J. Marfurt, T.M. Alves, Pitfalls and limitations in seismic attribute interpretation of tectonic features. Interpretation (2015)
64. J.L. Masaferro, R. Bourne, J.-C. Jauffred, 3-D seismic imaging of carbonate reservoirs and structures. Lead. Edge **22**, 18–25 (2003)
65. S. Midtvåge, A.G. Finstad, *GIM—A Method for Computer Assisted Seismic Interpretation* (SEG Houston, 2009)
66. R.M. Mitchum, P.R. Vail, J.B. Sangree, Stratigraphic interpretation of seismic reflection patterns in depositional sequences, in *Seismic Stratigraphy—Applications to Hydrocarbon Exploration*, ed. by C.E. Payton, vol. 16 (AAPG Mem, 1977), pp. 117–123
67. E. Monsen, J. Odegaard, Seismic texture analysis by curve modelling with applications to facies analysis. SEG Expanded Abstracts **21** (2002)
68. B. Moseley, L. Krischer (vol. eds.), *Machine Learning in Geosciences. Advances in Geophysics*, vol. 61 (Elsevier, 2020)
69. L. Mosser, S. Purves, E.Z. Naeini, Deep Bayesian neural networks for fault identification and uncertainty quantification, in *First EAGE Digitalization Conference and Exhibition*, vol. 2020, no. 1 (European Association of Geoscientists & Engineers, 2020)
70. M. Noorim, H. Hassani, A. Javaherian, H. Amindavar, 3D seismic fault detection using the Gaussian process regression, a study on synthetic and real 3D seismic data. J. Pet. Sci. Eng. (2020)
71. Parihaka Dataset, New Zealand Crown Minerals. https://wiki.seg.org/wiki/Parihaka-3D. Accessed Mar 2021
72. G.A. Partyka, J.M. Gridley, J. Lopez, Interpretational applications of spectral decomposition in reservoir characterization. Lead. Edge **18**(3), 353–360 (1999)
73. G.A. Partyka, Seismic thickness estimation: three approaches, pros and cons, *71st Annual International Meeting* (SEG Expanded Abstracts, 2001), pp. 503–506
74. D. Patel, S. Bruckner, I. Viola, E. Grøller, Seismic volume visualization for horizon extraction. Pacific Visual. (2010)
75. S. Pedersen, T. Randen, L. Sønneland, O. Steen, *Automatic Fault Extraction Using Artificial Ants* (SEG, 2002)
76. S. Pedersen, T. Skov, A. Hetlelid, P. Fayemendy, T. Randen, L. Sønneland, *New Paradigm of Fault Interpretation* (SEG Expanded Abstracts, 2003)
77. R. Pepper, G. Bejarano, Advances in seismic fault interpretation automation. Search Discov. (2005)
78. I. Pitas, C. Kotropoulos, Texture analysis and segmentation of seismic images. Acoust. Speech Signal Process. **1989** (1989)
79. F. Qayyum, C. Betzler, C. Octavian, Space-time continuum in seismic stratigraphy: principles and norms. Interpretation (2017)
80. S. Purves, *Incorporating Uncertainty in Automated Seismic Interpretation* (DigEx, Oslo, 2019)
81. T. Randen, E. Monsen, C. Signer, A. Abrahamsen, J. Hansen, T. Sæter, J. Schlaf, L. Sønneland, *Three-Dimensional Texture Attributes for Seismic Data Analysis* (SEG Extended Abstracts, 2000)
82. T. Randen, S. Pedersen, L. Sønneland, *Automatic Extraction of Fault Surfaces from Three-Dimensional Seismic Data* (SEG, 2001)
83. A. Roberts, Curvature attributes and their application to 3D interpreted horizons. First Break **19**, 85–99 (2001)
84. A. Rotevatn, H. Fossen, Simulating the effect of subseismic fault tails and process zones in a siliciclastic reservoir analogue: Implications for aquifer support and trap definition. Mar. Pet. Geol. **28**, 1648–1662 (2011)
85. H.G. Reading (ed.), *Sedimentary Environments: Processes, Facies and Stratigraphy*, vol. 688 (Blackwell Science, Oxford, 1996)

86. T.M. Sheffield, T.E. Bulloch, D.M. Meyer, J.M. Sutton, Geovolume visualization and interpretation: speed and accuracy with auto-tracking, in *2003 SEG Annual Meeting* (2003)
87. R. Sheriff, *Encyclopedic Dictionary of Applied Geophysics*, 4th edn. (Society of Exploration Geophysicists, 2004)
88. J.S. Sierra, W. Marín, M. Bonilla, H. Campos, Structural and stratigraphic interpretation using spectral decomposition: applications in deepwater settings, in *2009 SEG Annual Meeting* (2009)
89. R. Simm, M. Bacon, *Seismic Amplitude: An Interpreter's Handbook* (Cambridge University Press, 2014)
90. A. Stefatos, M. Boulaenko, J. Hesthammer, Marine CSEM technology performance in hydrocarbon exploration-limitations or opportunities? First Break **27** (2009)
91. M.T. Taner, F. Koehler, R.E. Sheriff, Complex seismic trace analysis. Geophysics **44**, 1041–1063 (1979)
92. B. West, S. May, J. Eastwood, C. Rossen, Interactive seismic facies classification using textural attributes and neural networks. Lead. Edge **21** (2002)
93. M.B. Widess, How thin is a thin bed? Geophysics **38**, 1176–1180 (1973)
94. X. Wu, Y. Shi, S. Fomel, L. Liang, *Convolutional Neural Networks for Fault Interpretation in Seismic Image* (SEG Technical Program Expanded Abstracts, 2018)
95. X. Wu, S. Fomel, M. Hudec, Fast salt boundary interpretation with optimal path picking. Geophysics (2018)
96. X. Wu, S. Fomel, Automatic fault interpretation with optimal surface voting. Geophysics (2018)
97. X. Wu, S. Fomel, Least-squares horizons with local slopes and multigrid correlations. Geophysics (2018)
98. H. Yang, Z. Xiaodong, X. Liu, L. Sun, X. Yu, J. Li, *Seismic Attribute Analysis Based on Multi-attributes Fractal Dimension* (SEG Houston, 2009)
99. H. Yang, Z. Xiaodong, *Seismic Attribute Analysis based on Information Entropy for Carbonate Reservoir Characterization* (SEG Denver, 2010)

Using Interactive Visualization and Machine Learning for Seismic Interpretation

Manfred Bogen, Christian Ewert, André von Landenberg, Stefan Rilling, and Benjamin Wulff

Abstract This book chapter describes the use of interactive visualization and artificial intelligence (AI) for seismic interpretation purposes. After an introduction with some basics about finding oil and gas through seismic interpretation in Sect. 1, we describe two interactive visualization methods called user-driven seismic volume classification in Sect. 2 and semi-automatic detection of anomalies in seismic data based on local histogram analysis in Sect. 3. In Sects. 4 and 5, we describe our approach to use convolutional neural networks (CNNs), a class of deep neural networks, for the detection of geobodies such as fault, channels, and salt domes. In seismic interpretation, confidence in the results and risk minimization is always very important. Which decisions can be made on the results of an artificial intelligence? To address this common concern, we describe in Sect. 6 a method how to understand the operation of the CNNs better and how to thereby increase trust in the findings based on artificial intelligence. As Fraunhofer is about applied research, we had to implement a prototype or solution for our AI-based methods. We called it DeepGeo. We describe DeepGeo in Sect. 7 of this book in detail, before we give a short overview on the status quo on the next things to come from us in the seismic interpretation field with artificial intelligence and deep neural networks.

The concept of neural networks has been around for decades and was successfully applied already in the 1990s to applications in different areas. However, network models back then were limited in complexity, due to their demand in memory and

[1] http://image-net.org/challenges/LSVRC/.

M. Bogen: Work was done when affiliated with.

M. Bogen (✉) · C. Ewert · A. von Landenberg · S. Rilling
Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Sankt Augustin, Germany

B. Wulff
Fraunhofer Cluster of Excellence Cognitive Internet Technologies (CCIT), Garching b. München, Germany

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_4

computational resources for training, which directly translated into a limitation in performance and thus practical use.

With the rise of programmable graphics processing pipelines in consumer-grade graphics processors (GPUs), it became possible to utilize the capability of massively parallel processing found in those GPUs for arbitrary calculations. Training and productive use of neural networks are by their nature largely parallel computation tasks and hence GPUs are an ideal platform for their implementation. Using the processing abilities of modern GPUs and the massive amount of labeled data available these days in most of the application scenarios, it is possible today to build and train much larger network models. These so-called Deep Neural Networks were found to be surprisingly powerful compared to their much smaller siblings.[1]

Identifying the geological structures in seismic volumes is of great importance for oil and gas exploration. However, seismic data interpretation is a time-consuming manual task even for experienced experts. This is an ideal scenario for applying artificial intelligence as big data analytics and human expertise have to and can be combined here to bring success.

Fraunhofer IAIS[2] is running the VRGeo Consortium,[3] a joint venture of the international oil and gas industry and higher education, since 1998 to identify and apply innovative methods for oil and gas exploration. Nowadays, the application of artificial intelligence in this field is innovative. The newest outcome of this joint effort is DeepGeo, an integrated Deep Learning Solution for the oil and gas industry. Based on 2-D and 3-D convolutional neural networks (CNNs), DeepGeo eventually is about automatically detecting geophysical structures such as channels, faults, and salt domes being potential traps for oil and gas reservoirs in 3-D seismic volumes and bring them to the attention of the seismic interpretation (SI) experts.

With the help of regularization techniques, the models used generalize well to new data, despite the fact that only a small training set generated from weakly labeled data is openly available in the oil and gas domain. The experiments show that convolutional neural networks trained on raw pixel intensities are capable of achieving high-quality segmentation results in the seismic interpretation field that requires specific domain knowledge.

This book chapter describes the use of interactive visualization and artificial intelligence (AI) for seismic interpretation purposes. After an introduction with some basics about finding oil and gas through seismic interpretation in Sect. 1, we describe two interactive visualization methods called user-driven seismic volume classification in Sect. 2 and semi-automatic detection of anomalies in seismic data based on local histogram analysis in Sect. 3. In Sects. 4 and 5, we describe our approach to use convolutional neural networks (CNNs), a class of deep neural networks, for the detection of geobodies such as fault, channels, and salt domes. In seismic interpretation, confidence in the results and risk minimization is always very important. Which decisions can be made on the results of an artificial intelligence? To address

[2] https://www.iais.fraunhofer.de/en.html.

[3] www.vrgeo.com.

this common concern, we describe in Sect. 6 a method how to understand the operation of the CNNs better and how to thereby increase trust in the findings based on artificial intelligence. As Fraunhofer is about applied research, we had to implement a prototype or solution for our AI-based methods. We called it DeepGeo. We describe DeepGeo in Sect. 7 of this book in detail, before we give a short overview on the status quo on the next things to come from us in the seismic interpretation field with artificial intelligence and deep neural networks.

1 Some Basics About Seismic Interpretation (SI)

Reflection seismology uses artificially generated seismic waves to explore the inner depths of the earth. Since these seismic waves are ordinary sound waves, they spread out in all directions when traveling through a medium. The speed of propagation depends on the density of a material, and seismic waves are reflected at the boundary layers of changing density. The amplitudes of these reflections can be measured using seismographs. By taking a number of these measurements at regular distances to each other (thus placing multiple traces in a row), a two-dimensional image can be constructed that represents a cross-section through the earth. This image is called a seismic line. By stacking multiple lines, a three-dimensional image (3-D survey) of the subsurface can be built. This image represents a discrete scalar field with the scalar values relating to seismic amplitudes and grid points relating to the spatial coordinate of the respective seismic events [1].

The so-called *seismic attributes* are all the information obtained from seismic data, either by direct measurements or by logical or experience-based-reasoning [2]. Similar to the field of image processing, seismic attributes are specific filters applied to the seismic amplitude data. Looking at seismic attributes may reveal certain features in the data that may not be apparent otherwise. It is common practice to calculate a multitude of different seismic attributes for a given data set, which turns the original 3-D scalar field (i.e., the recorded amplitudes in the seismic volume) into a 3-D vector field, having n-D vectors for each voxel.

The adaption of the reflection seismology method used within the oil and gas industry to find promising locations of oil and gas reservoirs leads to volumetric datasets with a size of up to several hundred gigabytes. Geologists and geophysicists analyze these datasets in a process called *seismic interpretation.* Since possible hydrocarbon deposits are in the vast majority of cases not explicitly visible in the data, domain experts have to interpret the structures in the subsurface of the earth in a difficult and time-consuming process. The goal is to find the so-called *hydrocarbon traps*: geological formations where the accumulation of hydrocarbons is very likely [3].

2 User-Driven Seismic Volume Classification with Interactive Visualization

Advances in computer and visualization technology lead within the last years to a multitude of different techniques, software- and hardware solutions to provide the necessary tools to the seismic interpreters, helping to increase both the speed and precision of the interpretation process. Although the automation of this interpretation process through emerging technologies from the field of pattern recognition and artificial intelligence has today become an active field in research, the human pattern recognition abilities are still an important factor needed to, in the end, turn the knowledge gained from seismic data into barrels of oil. Interactive visualization techniques have been an important technology within the last years to exploit the human's pattern-recognition-abilities and support the seismic interpreters to classify seismic data.

In the following sections, we overview the techniques for user-driven seismic volume classification developed within the VRGeo Consortium. We use the term "user-driven" to describe the human-centered approach of this work: a human interpreter is in the center of contemplation, and by means of visualization and interaction techniques, the human work is supported in an optimal way. We show how user-driven seismic volume classification relates to volume rendering and transfer functions, and how the problem of transfer function design relates to histograms.

2.1 Multi-attribute Visualization

The adaption of volume rendering techniques is a common approach to visualize seismic data in three dimensions and it produces images of high visual quality. Our rendering system bases on a volume ray casting technique and is capable of Phong[4]-shading, geometry intersection and stereo visualization. The latter two aspects have especially proven useful in the seismic interpretation process [4]. The system furthermore supports the visualization of multiple attributes for both direct volume rendering and slice visualization. One requirement from seismic interpreters is the capability to rapidly switch between the seismic attributes to get a comprehensive understanding of the data and the measured physical properties.

We implemented, in the VRGeo Consortium, a volume renderer, which is capable of visualizing multi-attribute data by using multiple volumetric RGBA-textures. Each attribute is loaded into one channel of the RGBA-texture, leading to 4 attributes per texture sampler. During the volume ray casting step in the fragment shader, 4 attributes can be retrieved simultaneously using one texture fetch operation. Figure 1 shows an overview of this approach.

[4] https://www.scratchapixel.com/lessons/3d-basic-rendering/phong-shader-BRDF.

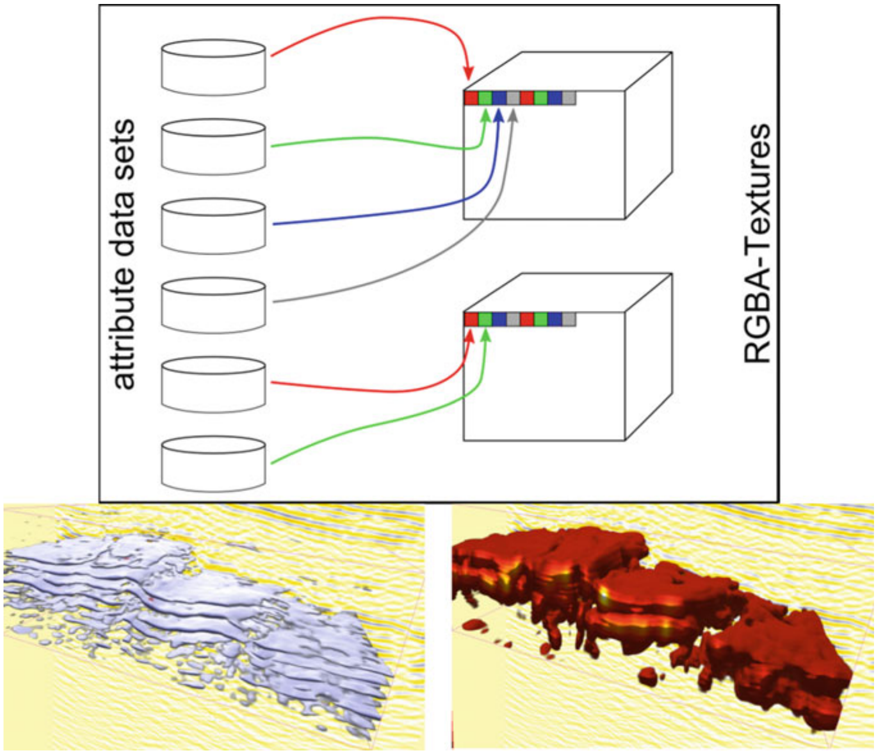


Fig. 1 Multi-attribute visualization technique. Different attributes display for the same rectangular sampling probe. The attributes can be switched through the user interface; the visualization changes instantly

2.2 *Seismic Volume Rendering and Transfer Functions*

Transfer functions map for each voxel from the data its data value(s) from a given input domain to an optical property, for example, color or opacity. Through this technique, each voxel of the dataset is assigned to a user-defined class, i.e., transfer functions are a tool for user-driven seismic volume classification. Two main challenges arise when working with transfer functions:

1. Finding an appropriate transfer function for a given classification task
2. Finding an appropriate user interaction technique to perform the actual classification task.

Figure 2 shows the application of a transfer function to a 3-D seismic volume. Color and opacity values are defined for specific data values of the input domain using a dedicated graphical user interface.

One important aspect visualizing data for seismic interpretation is to choose the appropriate color scheme for a specific attribute. Color is an important visual cue

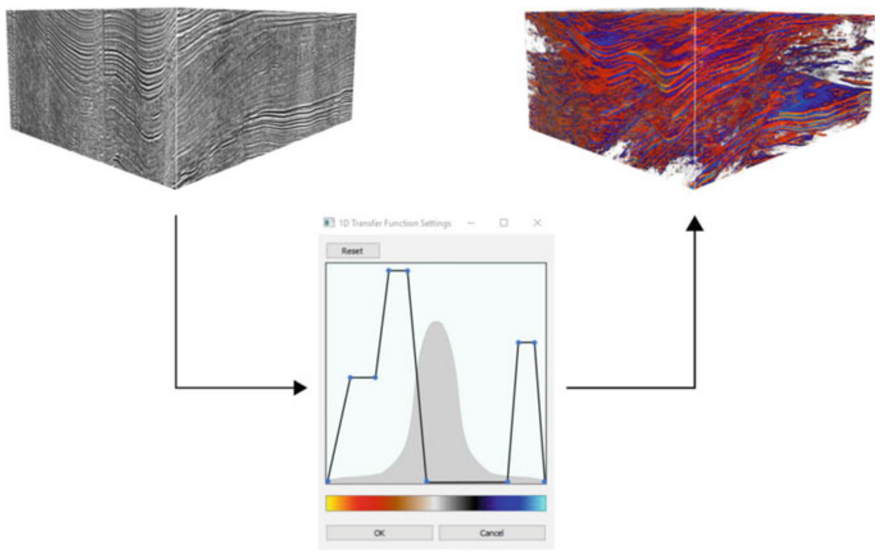


Fig. 2 Transfer function editor with color table and corresponding volume data visualization. The opacity and color for a given data value defined by defining a curve

in seismic interpretation and can impact the outcome of a seismic interpretation [5]. The colors used in the data visualizations shown in this chapter are, therefore, not chosen randomly. They rely on established coloring schemes used in well-established seismic interpretation packages like OpenDTect.[5]

When having a seismic data set with multiple seismic attributes computed, multi-dimensional transfer functions are a suitable tool to interactively classify relevant seismic features by applying a transfer function to this multivariate dataset. The visualization of multi-attribute data was quite an active field in research, and multitudes of different approaches have been described [6]. As these methods are usually used to display specific properties of various kinds of data and get insights into this data through this visualization, multi-attribute visualization techniques can also be used to define transfer functions, which then control a volume rendering system. An example of this approach based on parallel coordinate plots is shown in [7], and we build several visualization techniques based on cross plots, glyphs and parallel coordinates tailored to seismic data, which are described in Sect. 2.3.

2.3 Histograms and Transfer Functions

Multidimensional transfer functions are a valuable tool to control the volumetric rendering of seismic data sets. Using several dimensions in the input domain can

[5] https://www.dgbes.com/.

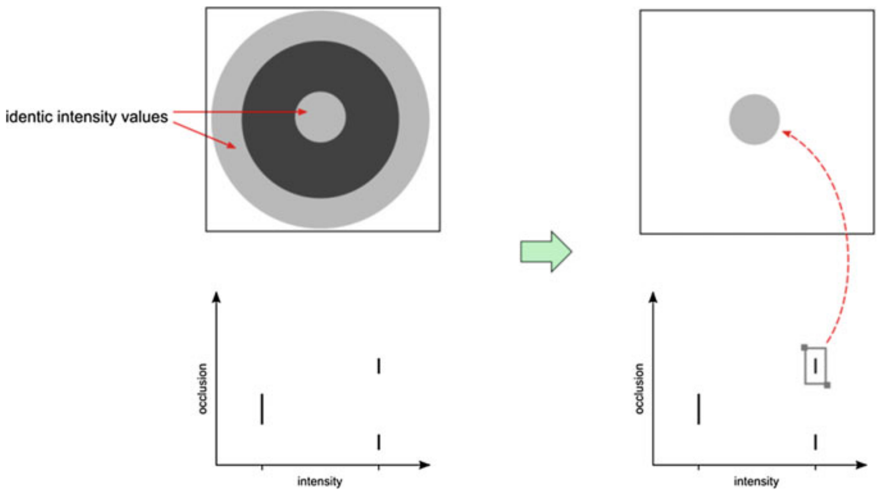


Fig. 3 2-D transfer function for the classification of features with identical intensity values. The center area of the data set corresponds to a specific area in the 2-D histogram

help to reveal structures in the data occluded otherwise when only looking at one specific dimension of the data, and the relevant structures have the same data values for this specific dimension.

Figure 3 shows this situation and how it can be resolved using a 2-D transfer function.

The transfer functions we discuss in this chapter are based on multidimensional histograms of the seismic data sets. These histograms are created from data sets containing multiple attributes (see Sect. 1), each of the dimensions in the histogram corresponds hereby to a specific seismic attribute.

In a histogram, the frequency of occurrence is shown for each possible data value from the input domain. In 8-bit datasets, we have 256 different possible data values for each data point. From an 8-bit two-attribute dataset, a 2-D histogram with 256 × 256 possible data values can be created. These 2-D histograms are frequently referred to as *cross plots* or *scatterplots*. To store these histograms in memory, special attention needs to be payed to the histogram data structures. As each entry in the histogram stores the number of data points sharing the data value related to this entry, the number of all voxels from the seismic data set needs to be stored for a complete uniform volume data set. As seismic datasets can be quite large, a minimum of 32 Bit (resulting in the possibility to count ca. 4 Billion voxels) is required from our experience to work with the smaller datasets. On modern graphic cards, RGBA-textures with 32-bit unsigned values for each channel are available, enabling the storage of histograms within textures and thus allow the fast access to histogram values within a shader.

2.3.1 Cross Plot Interaction

In Fig. 2, we introduced the “classical” transfer function editor found within many volume rendering packages. The transfer function is basically defined by drawing multiple curve segments into a coordinate-frame, where the values on the X-axis describe the data value and the values on the Y-axis describe the opacity for a specific data value. For 2-D histograms, defining transfer functions through curves is not a very suitable tool (although it would be possible). In fact, from multiple expert-interviews we concluded that a painting technique is most suitable for this interaction task. Figure 4 shows a minimal user interface for cross plot-painting. A pen and an eraser tool are provided, the size of the tooltips can be defined through the user. This interaction technique can be used with classical mouse and keyboard input devices. We have also made good experiences with the usage of pen and touch devices.

The painting interaction results actually in a bitmap image with the information which of the attribute combinations are assigned to a high opacity value. We store this bitmap in a texture continuously updated during the painting interaction, allowing for

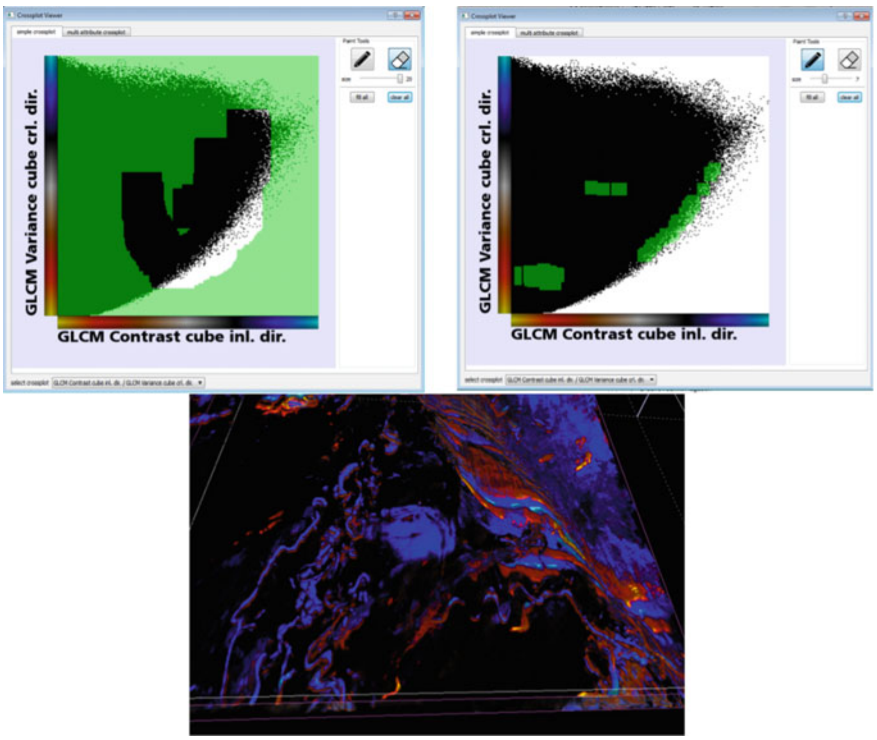

GLCM Variance cube crl. dir.
GLCM Contrast cube inl. dir.
GLCM Variance cube crl. dir.
GLCM Contrast cube inl. dir.

Fig. 4 Painting-Interface for cross plot-based transfer function. Value combinations covered by green color will be rendered. Transfer functions can be defined through erasing (top left) or painting (top right). The volume-rendering visualization updates interactively (bottom)

an interactive update of the volume rendering. For a multi-attribute seismic dataset, 2-D histograms have to be created for each possible attribute combination to enable the design of a transfer function covering the whole attribute range. On the painting user interface, the attribute combinations can be chosen through the user, and the texture storing the painting information is updated during the interaction. The texture storing the painting information (and thus the transfer function) is a 3-D texture, where each slice in the texture holds the painting information for one 2-D histogram. During the volume ray casting process in the fragment shader, the opacity information for each possible attribute combination of a raycasting sample is looked up from the corresponding slice from the 3-D painting information texture. The final decision if a ray casting sample is visible can be subject to different strategies, we found that a logical AND operation on the outcome from all visibility lookup a reasonable approach. A sample is thereby only visible if all attribute combinations are visible according to the defined 2-D transfer functions. Figure 5 shows an overview of the rendering pipeline.

The histograms do not necessarily have to be calculated over the whole dataset. They can also be created for a local region in the volume. Comparing these local histograms for different spatial locations in the dataset can lead to valuable insights into the data set. The concept behind this approach is the fact that change in the distribution of data values within a seismic data set is linked to a change in the physical or structural properties of the rocks.

We developed an interaction technique, which exploits the human pattern recognition abilities using local 2-D histograms. Local 2-D histograms are calculated in

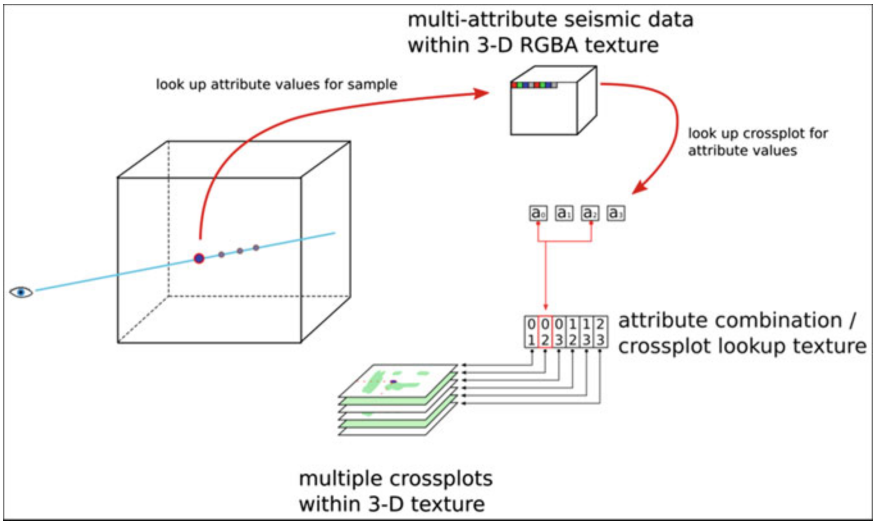


Fig. 5 Mculti-attribute cross plot rendering pipeline. The transfer function information is stored in a 3-D texture, which is accessed within the ray casting process for each sample's possible attribute combination

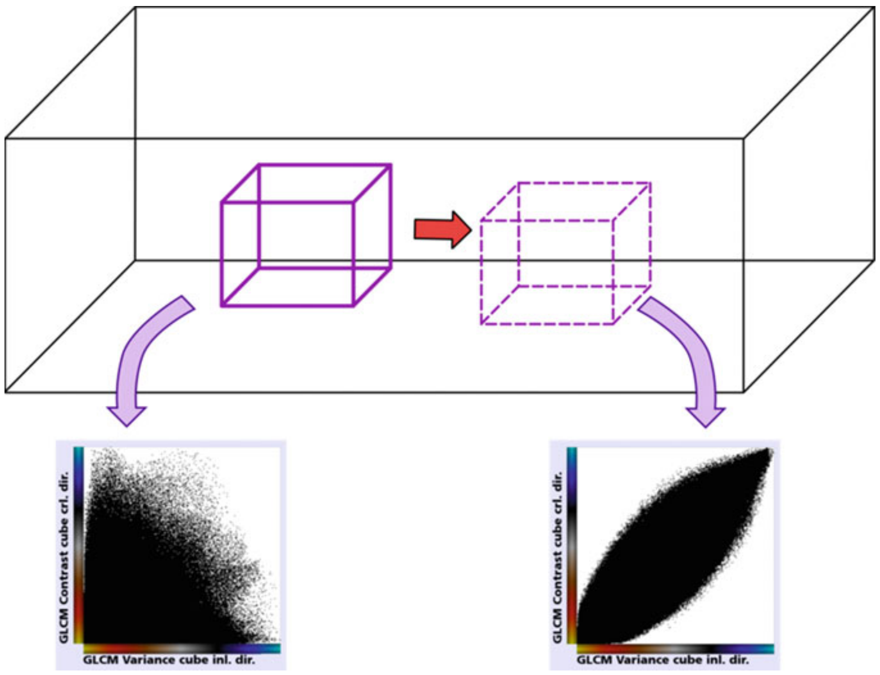

GLCM Contrast cube crl. dir.
GLCM Variance cube inl. dir.
GLCM Contrast cube crl. dir.
GLCM Variance cube inl. dir.

Fig. 6 Example of interactive local 2-D histograms. As the volume lens moves through the seismic data set, the cross plots are updating in real-time

real-time for only a small region (the so called "probe") within the dataset and are displayed to the user (see Fig. 6). The user can then move this probe through the data and observe the change of the cross plots. Within several user tests with seismic interpreters, we found that users quickly identify even subtle changes in the cross plot visualization and quickly understand the change of physical properties in the dataset. This approach can support the interpreter in identifying interesting geological structures and is a useful addition to looking at the rendering of the seismic.

For this interaction technique, it is crucial that calculations of the histogram and the update of the visualization are done in interactive framerates. The challenge here is the fast generation of the cross plot histograms. We use a parallel algorithm for the histogram calculation on the GPU using CUDA and achieve usable framerates for volume probe sizes of around $100 \times 100 \times 100$ voxels.

2.3.2 Glyph Cross Plots

Glyphs are graphical objects, which encode a certain piece of information. For example, arrows are common glyphs to visualize the direction of a flow (i.e. on a weather map). Hereby a set of arrows is used to visualize the underlying vector-field.

Glyph-based visualizations are a common means to visualize multivariate datasets [8], and a multitude of variations exists.

We extended the cross plot approach presented in the previous sections with the placement of glyphs inside the cross plots. This method combines the ability to visualize multivariate data with the advantages of cross plot-based histograms. This visualization requires the selection of two main attributes by the user; these main attributes form the X- and Y-axis of the cross plot visualization. The frequency of occurrence of a specific value combination within the two main dimensions is visualized by adapting the radius of a circle. The distribution of the other attributes is visualized through bars, which are placed inside the circle. We bin the 2-D histograms to avoid visual cluttering. Thus, the glyphs show an aggregation over a certain range. To further avoid visual cluttering and, on the other hand, allow the user to explore details of the cross plot, a level-of-detail mechanism depending on the zoom level of the cross plot view is used. This mechanism adjusts the bin size of the histogram depending on the selected zoom level. Figure 7 shows an overview of the approach.

The glyph-based cross plots can also be used in combination with the local histogram technique (see Sect. 2.3.1), as the update of the visual representation can be computed fast. Within multiple user tests, we found that this combination is considered as most valuable, as changes in the glyphs can be recognized quickly. However, to recognize subtle changes in the glyphs, a certain size of the glyphs is necessary, making the binning approach we presented mandatory.

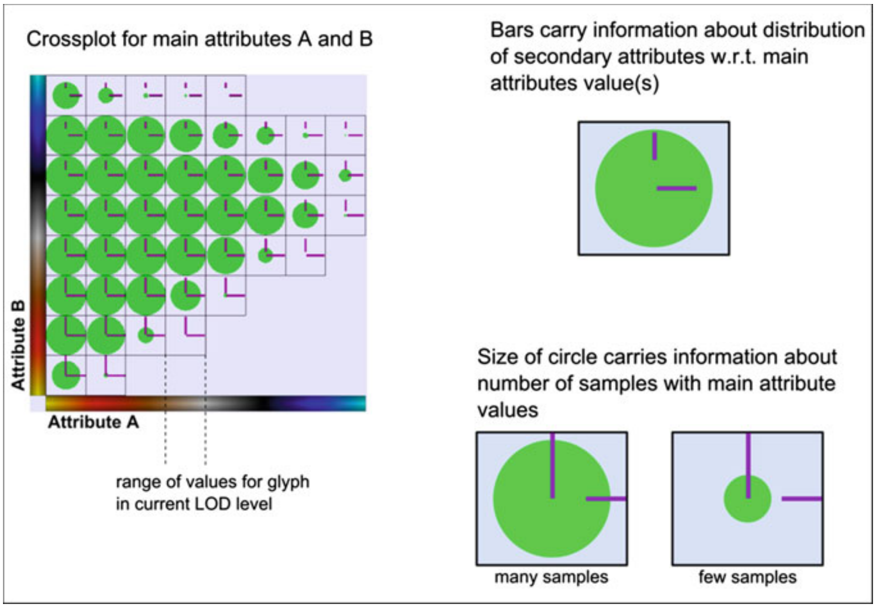

Crossplot for main attributes A and B
Attribute B
Attribute A
range of values for glyph in current LOD level
Bars carry information about distribution of secondary attributes w.r.t. main attributes value(s)
Size of circle carries information about number of samples with main attribute values
many samples
few samples

Fig. 7 Our implementation of glyph-based cross plots to show the data distribution of multivariate datasets

2.3.3 Parallel Coordinates Interaction

The adaption of parallel coordinate plots to visualize multivariate data has been proposed for many years, where the publications of Inselberg [9] can be considered pioneer work. They provide a useful visualization of multivariate data in different areas of application, for example to visualize the outcome of simulations or to explore product catalogs [10].

Whereas in typical Cartesian-coordinate plots coordinate system's axes are visualized perpendicular, axes in parallel coordinate plots are visualized parallel next to each other. This technique allows the visualization of an arbitrary number of data dimensions. A point in an n-dimensional Cartesian coordinate system corresponds to a path with n-1 path segments in a parallel coordinate system (see Fig. 8).

Parallel coordinate visualizations enable the user to recognize clusters and correlations quickly. This capability makes parallel coordinate visualizations a suitable tool for defining multi-dimensional transfer functions to control the rendering of multivariate volumetric data like weather measurements [11] or seismic data sets.

We developed a transfer function editor for multi-attribute seismic datasets based on parallel coordinate visualizations (see Fig. 9). The transfer function editor shows the correlation of the amplitude data with the other attributes and visualizes clusters in the n-dimensional space.

One key aspect of providing a parallel coordinates visualization for seismic data is the large number of data points in such data sets. Large seismic datasets can

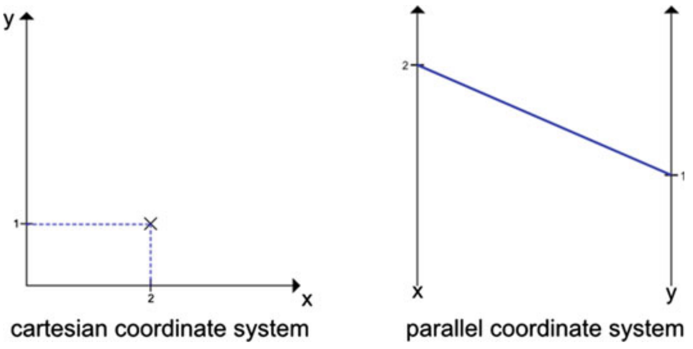


Fig. 8 A 2-D point in cartesian and parallel coordinate systems

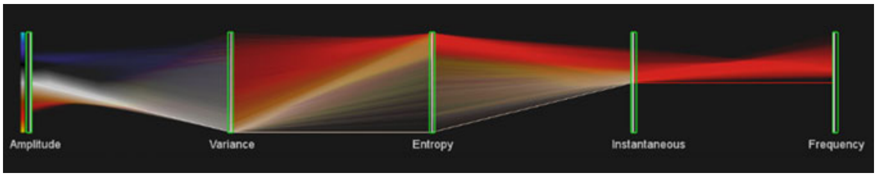


Fig. 9 Parallel coordinate visualization of a seismic dataset with 5 attributes

comprise billions of points requiring in a naïve implementation drawing billions of line segments for the whole plot. To achieve interactive framerates and to enable the combination of parallel coordinate plots with the interactive local histogram technique, we need techniques to accelerate the computation of the visualization. We developed a GPU-based parallel coordinate transfer function editor, which is capable of operating at interactive framerates. It has been evaluated and constantly improved in several user tests with the experts in the oil and gas domain.

The key aspect in accelerating the parallel coordinates rendering is the reduction of the number of lines drawn. We solve this issue by using so called-joint 2-D histograms for the line count reduction. This method builds on the observation that two subsequent parallel coordinate axes can be viewed as a 2-D histogram. Having an 8-Bit dataset will result in a 2-D histogram with 65.536 entries for each possible data-value combination. In the worst case, a line has to be drawn for each entry in the 2-D histogram. Therefore, for an n-dimensional dataset, $(n - 1)2^{16}$ line segments have to be drawn, which is a significant reduction in line numbers for a reasonable number of dimensions compared to draw several billions of lines. Figure 10 shows an overview of this approach. As the frequency of occurrence of a specific value combination is already stored in each joint histogram, this information can be used to derive visual properties like line thickness or opacity.

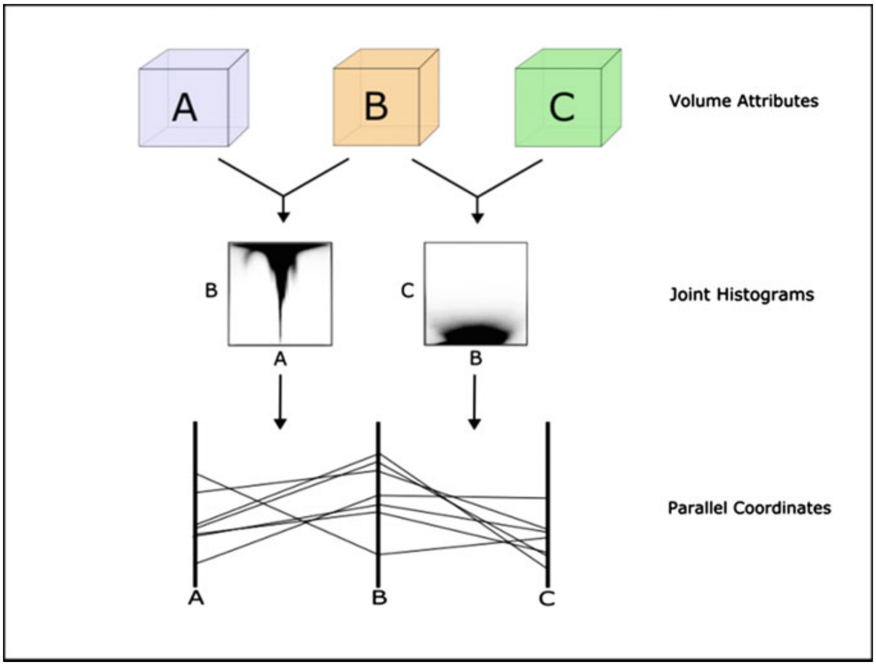


Fig. 10 Reduction of the number of lines through the usage of joint histograms

The line segments of the parallel coordinate visualization are drawn from the information stored in the joint histograms in a recursive approach. Starting with the first joint histogram for the axes A and B, for each of the 256 data values on axis A, (assuming an 8-bit dataset), the corresponding column c in the 2-D histogram is examined. For each entry r in the column, a line is drawn between the two axes A and B and this procedure is repeated recursively for the "next" joint histogram covering the axes B and C. Special care is taken that no line segment is drawn multiple times. For example, this would be the case for a dataset with three attributes and two points (a_0, b_0, c_0) and (a_1, b_0, c_0), where the line between b_0 and c_0 needs to be drawn only once. Our algorithm hereby stores the information on line-segments already drawn and avoids starting a new recursion in this case.

The frequency of occurrence for a specific value is visualized through the opacity of the line originating from this value. The mapping of the frequency of occurrence n to the opacity α can be either linear or non-linear (see Fig. 11). A non-linear mapping with a steep gradient for low frequencies of occurrence can help to emphasize these rarely occurring values, as the perceived contrast for different rarely occurring values is increased using this technique.

To enable the design of transfer functions using the parallel coordinates plot, we equipped all axes of the plot with a selection widget. The widget size and its position on the axis can be changed through point and click interaction with a classical mouse. Using this technique, the user can restrict the data-range for each attribute used for the volume rendering of the data set. Figure 12 shows an example of the selection widgets for three attributes.

Lines outside the selection ranges are not drawn in our implementation, there are many examples in the literature where lines not selected are rendered in a neutral

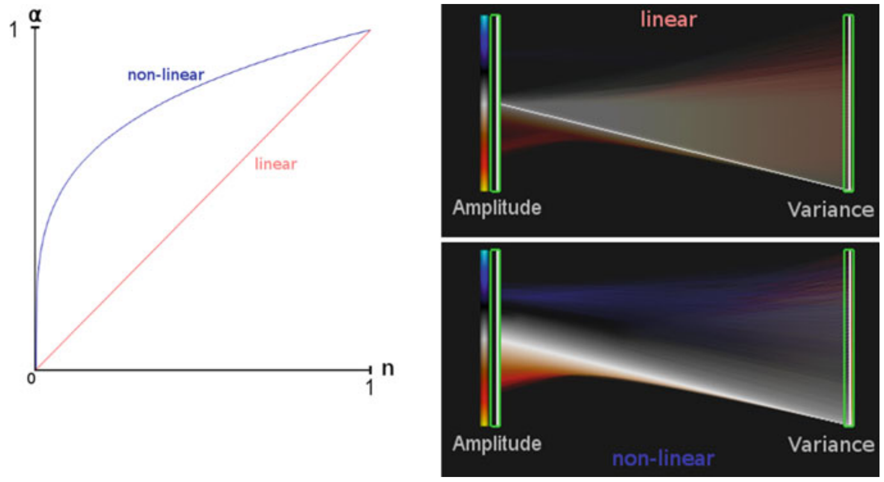


Fig. 11 Opacity of the lines in the parallel coordinates plot is determined through the frequency of occurrence of the according data value (left) and affects the visualization (right)

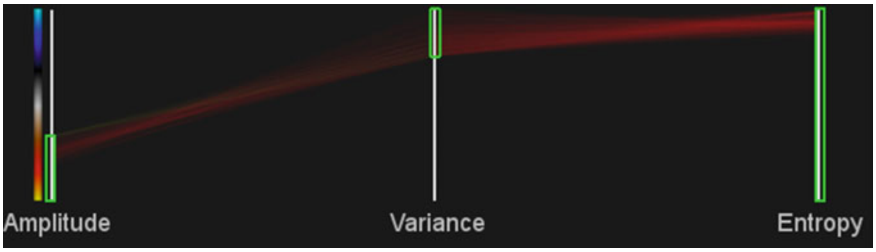


Fig. 12 Selection widgets (green) on parallel coordinate axes

color (for example a light grey) to provide an additional context. In our implementation, the color of a path through the plot is determined by the first axis in the plot. In Fig. 12, a color scheme is shown left of the first "Amplitude" axis. Using this coloring technique, the seismic interpreter can quickly identify which amplitude values correlate to other attribute values. The selection widgets design, the coloring scheme and the opacity mapping were developed iteratively through several user tests with domain experts.

To control the volume rendering using the parallel coordinates transfer function, we made available the information on the selected data ranges for each attribute in the volume-ray-casting-fragment-shader using a 1-D RGB lookup texture (see Fig. 13). The i-th texel corresponds hereby to i-th parallel coordinate axis in the plot. The upper and lower bounds of the selection are stored in the R- and G-channels

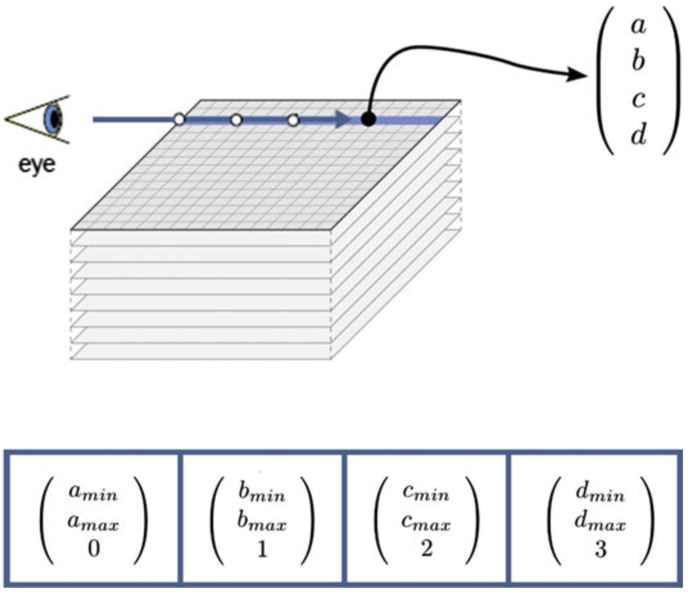

eye
$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$
$\begin{pmatrix} a_{min} \\ a_{max} \\ 0 \end{pmatrix}$
$\begin{pmatrix} b_{min} \\ b_{max} \\ 1 \end{pmatrix}$
$\begin{pmatrix} c_{min} \\ c_{max} \\ 2 \end{pmatrix}$
$\begin{pmatrix} d_{min} \\ d_{max} \\ 3 \end{pmatrix}$

Fig. 13 Direct volume rendering with restriction widget information stored in an 1-D texture

of the texture. As the order of the parallel coordinate axes can be chosen freely by the user, the i-th axis does not necessarily correspond to the i-th attribute stored in the RGBA-texture storing the seismic data. Therefore, the B-channel of the lookup texture stores the index of the axis-attribute within the RGBA-volume texture storing the seismic dataset.

Our implementation of a parallel coordinates transfer function editor runs at interactive framerates for interaction and volume rendering update on a standard desktop machine with a recent GPU. We tested the isolation of several interesting geological features like salt domes or channels using multiple attributes and our parallel coordinate transfer function editor during several user studies. The system was perceived as very useful and intuitive.

Figure 14 shows the result of an isolation process using the transfer function editor.

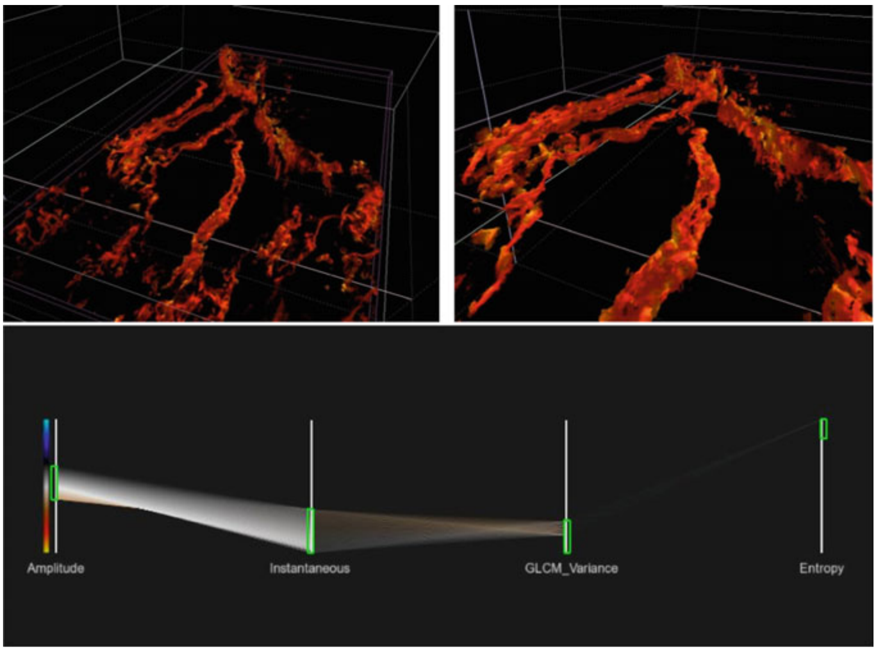


Fig. 14 Final result: volume rendering with parallel coordinate transfer function. Channel structures (top) are isolated by selecting data-ranges on four different attributes (bottom)

3 Semi-automatic Detection of Anomalies in Seismic Data Based on Local Histogram Analysis

In the previous sections, we presented a set of tools that support a user in seismic interpretation. All approaches shown have the manual interaction of the user with the tools and the data in common. They are not semi- nor fully automatic. These approaches are time-consuming due to a large number of parameters to be tuned in order to get a satisfying visualization result.

Through the given challenges of a manual approach, the automation of the analysis stands to reason. One possibility is to rely on fully automatic approaches with machine learning algorithms to detect relevant features. However, these approaches have multiple challenges. At first, they usually require an elaborate and time-intensive training step that adapts the algorithm to the specific targeted features. For machine learning models with a large number of parameters, ground-truth data with millions of interpreted samples has to be available. This is a challenging task in the domain of seismic volume data, as such data sets are usually not published by the owning companies. Moreover, domain knowledge and expertise are required to interpret it seismic data for the usage as annotated training data. This process takes time and is expensive. Finally, for every desired new feature to be learned, the training has to be repeated, always requiring new interpreted data containing the target feature.

So, we made an intermediate step. A combination of user-driven and automatic approaches could, on one-hand, lead to faster interpretation results, and on the other hand remain generic enough that relevant regions can be detected independently from the specific seismic features contained.

In this section, we present a semi-automatic approach capable of detecting interesting regions, which then could be interpreted manually. This way the time of screening the dataset and searching for interesting regions is reduced and multiple starting points for the user-driven interpretation could be offered directly to an expert.

Our approach is to detect anomalies in seismic data based on local histogram analysis. The concept behind this approach is that on a global scale the amplitude values in a seismic dataset mostly follow a Gaussian distribution. However, the data distribution may vary locally in certain parts of the dataset. These areas are of primary interest for an interpreter, as they can indicate the presence of valuable hydrocarbons. A local variation of this type is known as a geophysical anomaly [1]. The principle of the analysis is to identify these as a contrast to the general background data, taken from a given region. Thus, the geophysicist is looking for an anomaly in relation to the surroundings [3]. These are usually only present in small regions of the dataset and might not be visible in the global histogram, as the variation of the data distribution is too subtle in a global sense. However, these anomalies have a better chance of appearing in the respective region's local histogram's value distribution. Hence, these anomalies could be detectable, when comparing the local histogram of that region to the local histograms of the neighborhood, in which the anomaly is not present. The detection of these anomalies is critical since their interpretation could be a straightforward solution to finding hydrocarbons and defining the lithology [12].

Thus, the detection of the anomalous regions might be a suitable approach for a semi-automatic algorithm.

Given two histograms, it can be useful to compare their similarity for different reasons. Cha [13] describes a detailed and comprehensive survey of histogram dissimilarity measures. The dissimilarity measures for the comparison of two histograms divides into two groups: *bin-by-bin* and *cross-bin*. The bin-by-bin dissimilarity group derives its name from only comparing histogram bins with the same index. The dissimilarity between two histograms results from the comparison of all pairwise differences. From the group of bin-by-bin dissimilarity measures, the *Minkowski-form distance*, the *Chi-Square Distance* or the *Jeffrey divergence* have to be mentioned. In contrary to the first group, the cross-bin dissimilarity measures also contain terms that compare non-corresponding bins. The results of these dis similarity measures are perceptually more meaningful [14]. Examples from this group are the *Match distance* or the *Kolmogorov-Smirnov distance*.

Histogram dissimilarity measures have been adapted within image processing for many years, for example, in content-based image retrieval tasks [14], natural-image boundary detection [15] or the automatic quality control based on images in fabric production [16], as well as in quality control tasks based on the segmentation of volume data sets [17].

We based our approach on local histograms compared to other local histograms from a defined neighborhood. The definition of this neighborhood and the size of the local region from which the histogram is calculated are crucial for the outcome of the approach. For instance, the vertical neighborhood might better detect seismic horizons as they spread in a horizontal direction, and the values vary in the vertical direction. It can be expected that anomalies are detected likely, when they occupy most of the current region and are absent in the neighborhood. This way the dissimilarity of the local histogram to all histograms in the neighborhood is very large. Hence, the regions should be adapted dynamically to the present data values. Therefor the dataset has to be divided into bricks of varying size.

With the definition of a distance measure and a neighborhood, the local histogram of every brick in the dataset can be compared to local histograms of the neighborhood using dissimilarity measures. The bricks with the highest dissimilarity value to their neighborhood are most likely to contain anomalies. This makes the approach dividable in three sequential steps:

1. Recursive subdivision of the dataset.
2. Comparison of each brick to the selected neighborhood.
3. Creation of a ranked list of bricks, sorted by their dissimilarity.

Figure 15 shows an overview of the algorithm, including the required user input.

The size of the bricks used for the local histogram calculation is adapted dynamically to the present data. This has advantages compared to the alternative of a user-defined, static brick size. At first, equally sized bricks not fitted to the data might be either too big and miss small anomalies, or be too small so that the anomaly would also be present in bricks of the neighborhood. Furthermore, it is beneficial, that no

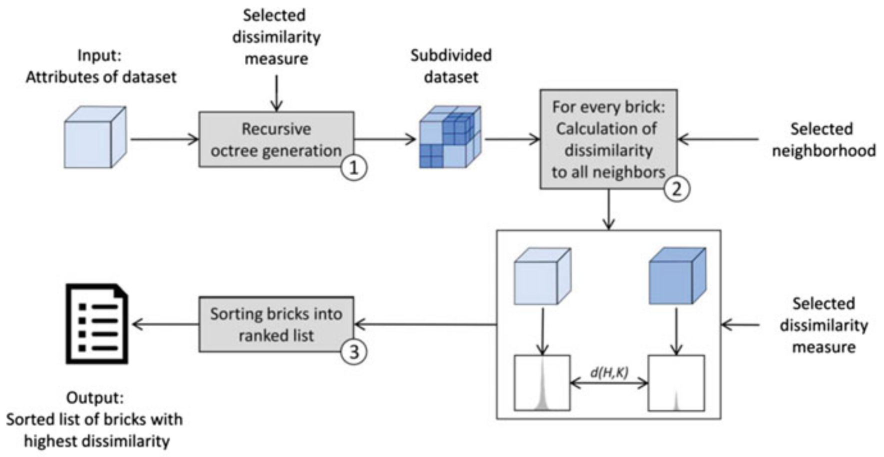

Input:
Attributes of dataset
Selected dissimilarity measure
Recursive octree generation
1
Subdivided dataset
For every brick: Calculation of dissimilarity to all neighbors
2
Selected neighborhood
d(H,K)
Selected dissimilarity measure
Sorting bricks into ranked list
3
Output:
Sorted list of bricks with highest dissimilarity

Fig. 15 Overview of the anomaly detection through local histograms

user input is required for defining the brick size. The size might be hard to determine manually, through the huge amount of possible value combinations.

Thus, a better approach is to fit the brick size to the data in an automatic manner. We invoke a recursive subdivision of the whole dataset, until the grid fits the local data values optimally. The optimal case would be a brick consisting mainly of the anomaly, which is not present in the adjacent bricks. The aim is to automatically set the borders of the bricks in such a way, that the anomalies of unknown location are isolated and fill up a huge portion of one brick. A further subdivision would lead to the division of the anomaly into multiple bricks and should be avoided. That way, the result is a high dissimilarity value to all surrounding bricks.

For further clarification of this idea, an example of the subdivision is provided in Fig. 16. The original dataset visible on the left of the top row consists of Gaussian noise and contains an anomaly in the upper right part. The first three levels of the recursive subdivision can be seen towards the right of the figure. With every additional level, the anomaly occupies more of one brick. The local histograms in the lower row always belong exactly to the brick containing the anomaly. One can easily see that with each subdivision the local histogram diverges further from the Gaussian distribution. In level three, the anomaly fills the whole brick. The comparison of the anomalous brick's local histogram shown in the top right-hand corner of the illustration, to the still Gaussian distributed local histograms of the surrounding bricks, results in a high dissimilarity. This way the anomaly can be detected. Additional subdivisions beyond level three would be pointless, as the anomaly already fills the whole brick and another subdivision would not change the local histograms any further. In fact, it could decrease the chances of detecting the anomaly, as it might be split up into four bricks, which reduces the irregularity on a local scale.

The recursive subdivision of the volume data set results in an octree structure with bricks of various sizes. It is performed using a split-and-merge strategy. First,

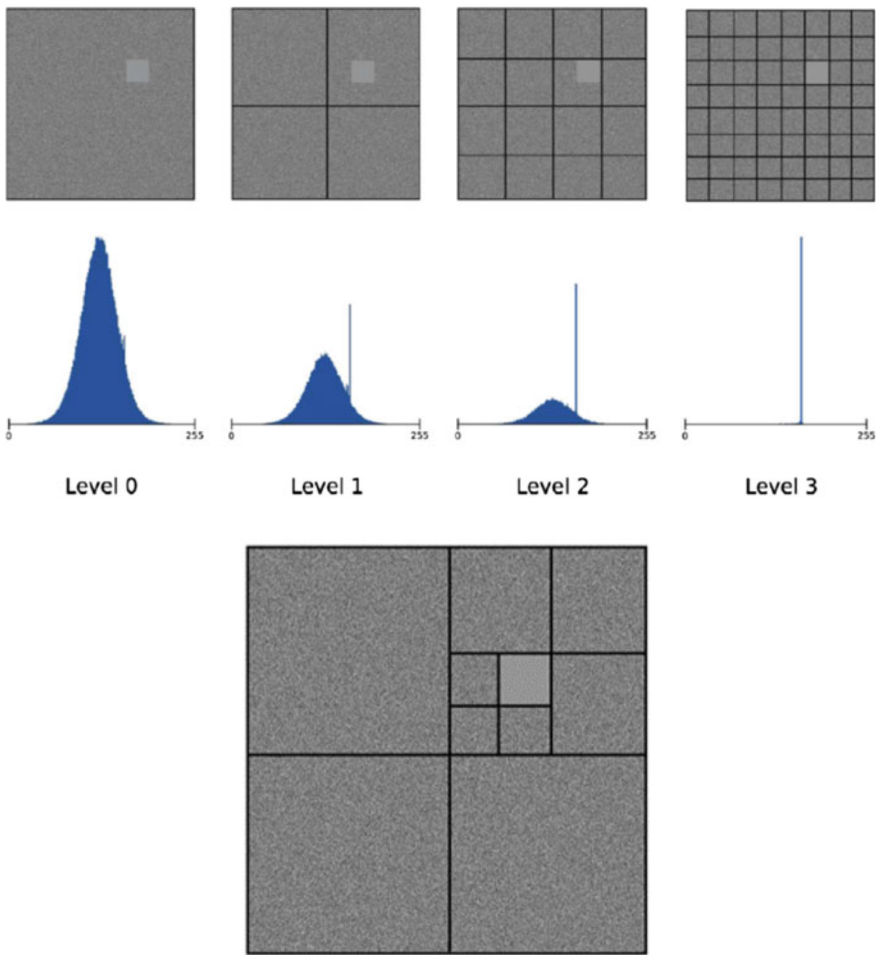


Fig. 16 Simple example for the isolation of an anomaly through recursive subdivision (top), resulting octree structure (bottom). The dataset is shown from the top

the dataset is recursively subdivided into an octree up to a user-defined maximum depth. During each subdivision step, the histogram dissimilarity is calculated for each new brick and the "root" brick, which represents the whole dataset. The calculated dissimilarity to the global histogram is used for the merge procedure. Starting with the layer above the leaf nodes, the eight children of every node are only kept if at least one of them contains the maximum dissimilarity along the path from it up to the root node. Otherwise, the eight children are merged and the corresponding nodes are deleted. Then the same validation is performed on the parent of the current node. The subdividing and subsequent merging is necessary to ensure, that the octree level with the global maximum is chosen. Otherwise, the recursive subdivision could just be stopped when the dissimilarity to the global histogram does not increase further. However, there is a chance of having detected a local maximum, while further subdivisions would reveal a level with an even bigger dissimilarity.

After the octree calculation, for each of the differently sized bricks in the octree, a dissimilarity value is calculated. This value describes the amount, by which the brick differs from its surroundings. For that process, the selected neighborhood type is used as input to the algorithm. As the octree is dynamically generated and the bricks do not have the same size, the number of adjacent bricks in one direction can vary, depending on the level of subdivision. The neighborhood calculation for a given brick has to be adapted to fit the dynamic grid size of the recursive octree. Every directly adjacent brick is taken into consideration. If the level of octree subdivision in a given direction is equal or lower, only one directly adjacent neighbor is present. The dimensions of this brick might be bigger than the current one due to fewer subdivisions in that region. If the octree in the target direction has a higher level of subdivision, the number of neighbors in that direction is greater than one. However, this does not affect the procedure of the comparison in any way. The total dissimilarity of a given brick is calculated as the average of the dissimilarities to each neighbor, using a user-defined dissimilarity measure. Besides, to total dissimilarity value, the dissimilarity of each bin of the local histogram to the bins of the neighboring histograms is calculated and stored for each brick. This bin dissimilarity describes for every bin, by which portion it contributes to the total dissimilarity of the brick. The final bin dissimilarity is also calculated as an average over the neighborhood.

After calculating the dissimilarities for each brick, the bricks are ranked w.r.t. their dissimilarity. All leaf nodes of the octree correspond to bricks compared to their neighborhood and sorted by their dissimilarity value. The highest ranked brick of the list has the largest value of dissimilarity, while the last entry of the list is not very different from its surrounding neighborhood.

The list of sorted bricks is used for several dissimilarity visualizations. Figure 17 shows the global visualization providing a first overview of different important regions in the dataset by visualizing clusters of dissimilarity. Only the borders of the bricks are visible to allow examining the data within. Depending on the size of the dataset and the level of subdivision, the ideal amount of displayed bricks may vary and the number can be adapted in real-time. The amount has to be big enough in order to identify clusters, but not too high to cause visual cluttering. The brick color is determined through a discrete heat map, allowing a clearer differentiation of the respective classes [18]. To increase the informational content of the bricks, the direction towards the region with the biggest dissimilarity is represented by an arrow added to the center of each brick. The arrow receives the same color as the corresponding brick to prevent visual cluttering and to simplify the visualization.

The part of the volume dataset inside of a selected brick can also be visualized using 3-D volume ray casting. A transfer function for this volume rendering can be calculated using the calculated bin dissimilarities, showing those parts of the active brick, which differ from their surroundings. Figure 18 shows expanded volume renderings generated by this technique.

In this section, we presented an algorithm for detecting anomalous regions in a seismic volume dataset based on the comparison of local histograms in a defined

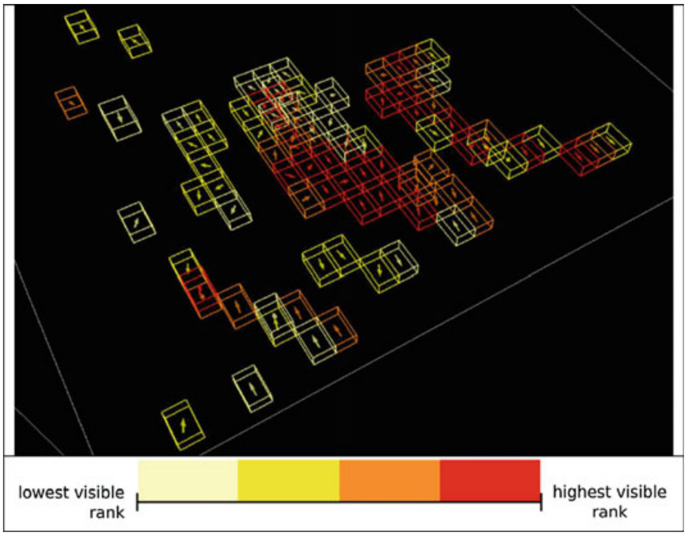


Fig. 17 Global visualization with glyphs and heat map

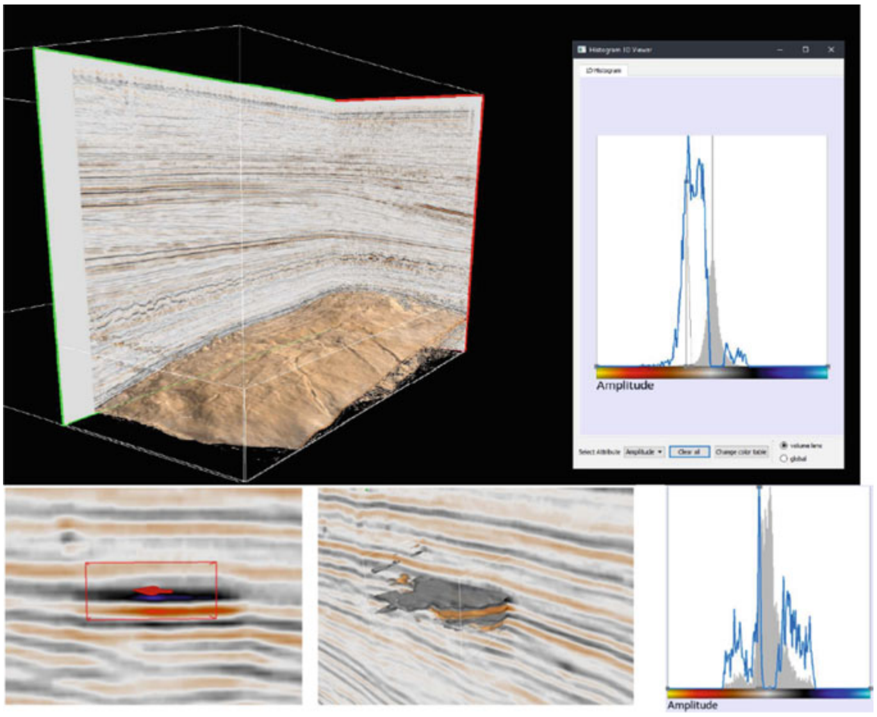


Fig. 18 Isolation of horizons by automatic transfer function generation (top), isolation of a bright spot (bottom). The automatically generated transfer function is shown through a blue line in the histogram view

neighborhood. The system we implemented offers multiple visualization tools to support the interpretation of the data. This algorithm was evaluated in multiple expert-interviews, in which the software was applied to real-world seismic datasets. This evaluation has shown that the detection of interesting regions is possible, independently from the contained feature type. The fast execution time of the algorithm makes it an appropriate candidate for an initial analysis.

4 Detecting Faults and Channels with 3-D CNNs

In the last sections, we presented a method for user-driven and semi-automatic detection of geobodies in seismic data sets. In this section, we describe our experiments with Convolutional Neural Networks (CNNs) to interpret seismic data to detect faults and channels.

4.1 Some Basics About Deep Learning

Deep Learning has had a big impact on Computer Vision [19, 20], and it has proven to be a valuable tool in medical imaging [21]. Convolutional Neural Networks [22] take a central role in this new development. CNNs have shown promising results in fields like radiology [23], dermatology [24] and general diagnostics [25]. They have not only been applied for diagnostics on 2-D images, e.g., in dermatology, but were also successfully employed in the analysis of CT and MRI images [26, 27], which are dense volumetric data. Similar dense volumetric data are acquired in modern seismic surveys. Hence, the idea to employ Deep Learning, especially CNNs, for Seismic Interpretation was obvious.

The VRGeo Consortium's task is to assess novel technologies to be employed in the VRGeo Consortium members' productive operations in beneficial ways. Deep Learning has become a huge hype in the midst of this decade. Thus, we started an R&D project of the VRGeo Consortium to evaluate this hype on its suitability and applicability for the oil and gas domain. The oil and gas industry is always keen to learn not only from higher education but also from other industries. Therefore, the first approach was to transfer methods already employed in medical imaging to the domain of seismic interpretation. We implemented state-of-the-art architectures and evaluated them in a first step. Then we conducted experimentation with customized architectures in the second part of the project.

In the following, we will give a brief overview of the related work in the fields of Deep Learning and application of CNNs in medical imaging; we will describe our experiments and the results that the different models achieved, we will detail how the acquisition of training datasets was conducted, we will give an overview about the ongoing effort to create a big standard dataset for the domain of seismic interpretation, we will conclude with a summarization of the results with respect to the value in productive seismic interpretation in the industry, and we will give an outlook on further work.

4.2 Related Work in Deep Learning

The term Deep Learning is mostly referring to Convolutional Neural Networks trained on massive amounts of training data. CNNs have been around for a while, first practically applied by LeCun'98 [19]. More than a decade later, the AlexNet architecture [28] set a new record in the ImageNet Classification Challenge 2012 (16.4%). This event set off the Deep Learning revolution. Quickly CNNs have then been applied to all kinds of computer vision problems and achieved remarkable performance, for example, in face recognition [29] but also in fields like natural language processing [30, 31].

As mentioned, CNNs have been applied in several clinical fields for diagnostics. Most successful was the application in image-based diagnostics. Examples are the classification of melanoma [24], diagnosis of breast lesions in mammograms [32] or diagnosis of diabetic retinopathy [33]. CNNs have also been successfully employed in the analysis of 3D MRI and CT data [26, 27].

4.3 Deep Learning Experiments

As a straightforward first step, we implemented some well-known state-of-the-art CNN architectures for 2-D image classification and tested them on the two-class problem of discriminating a channel from a non-channel time slice patches. First tests with LeNet [19], one of the earliest proposed CNN architectures, gave promising results, although a significant number of many false positives were present. We were able to minimize the number of false positives using AlexNet [28] and VGG [34], also taking into account faults as a third class (besides Sediment and Channel).

The initial results were promising, though not optimal. Another relevant issue for the practical application of machine learning is (apart from the training time) the amount of training data needed to train an architecture sufficiently. The number of parameters in the current architectures is quite high (compare Table 1) and as follows from the well-known Vapnik-Chervonenkis theorem [35], the required number of training samples increases exponentially with the complexity of the model [36]. As annotations on seismic data are not as plentiful available as labeled images of, e.g., ImageNet object classes, this poses a central problem for applying CNNs in seismic interpretation.

Based on the simple observation that CNNs for the classification of seismic patches will have to learn far fewer classes than networks for object classification, we attempted in the next step to create custom CNN architectures, trying to minimize the number of parameters to keep the training datasets as small as possible. We ended

Table 1 Comparison of CNN architectures. The object classification CNNs have a much higher number of parameters than the custom VRGeo architectures

Model	Layers	Kernel size	Parameters (millions)
LeNet	6	5×5	0.43
AlexNet	11	11×11	60.00
VGG	21	3×3	133.00
VRGeo CNN7	7	3×3	2.10
VRGeo CNN10	10	3×3	1.05
VRGeo 3D CNN	12	3×3	5.17

Table 2 Training and validation accuracy of VRGeo CNN architectures

Architecture	Training accuracy (%)	Validation accuracy (%)
VRGep CNN7	99.20	98.83
VRGep CNN10	97.26	97.06

Table 3 Training accuracy of VRGeo CNN7 using different optimization algorithms

Optimizer	Training accuracy (%)
Adam	98.52
Momentum	85.58
RMSProp	98.21
SGD	64.52

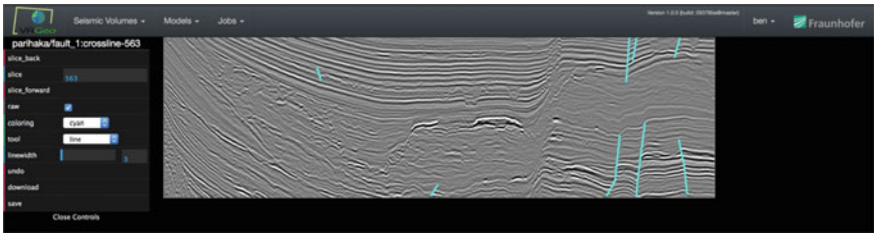
up working with two architectures that gave promising results in initial tests: VRGeo CNN7 and CNN10 (compare the number of parameters in Table 1). Table 2 gives the results using the VRGeo Seismic Dataset for training and validation.

As part of the experiments, we also tested different current optimization algorithms and found that, in accordance with current literature, the Adam optimizer [37] performed superior to older methods (compare Table 3).

The training dataset comprised samples from the Parihaka and the Penobscot volume, where we did the annotations. We also annotated channels and faults on some slices of the F3 Block.[6] We used the F3 data to test the networks' performance on unseen volumes, as it would be favorable to have a general model that analyse, any volume without having to be calibrated at first. However, the experiments showed that the 2-D CNN architectures trained on data from Parihaka and Penobscot performed at an accuracy of ~65% on the F3 test slices.

We then moved on to build a fully 3-D CNN that uses 3-D convolution and pooling layers. The VRGeo 3-D CNN is an eleven layer CNN with five convolutional layers and five pooling layers. Although 3-D convolution layers mean that the number of parameters increases in contrast to 2-D CNNs, we still achieved an architecture with a relatively small number of around 5.2 M parameters. Using the Adam optimizer,

[6] https://terranubis.com/datainfo/Netherlands-Offshore-F3-Block-Complete.



parihaka/fault_1:crossline-563

slice_back
slice 563
slice_forward
raw ✓
coloring cyan
tool line
linewidth 3
undo
download
save

Fig. 19 The DeepGeo annotation editor with some annotated faults (light blue lines) on a crossline slice of the Parihaka volume

we achieved an accuracy of 74% with the 3D CNN on the F3 test slices. That was a ~10% increase in contrast to the 2-D CNNs.

To further increase the accuracy of the model for unseen data, we then turned to the ensemble method, a well-known technique that often improves the results of neural network system. The idea is to take the weighted average across predictions from several networks [38, 39], weighting the predictions according to the score of the network instances. This way, errors of an individual network vanish in the averaging and an improved result becomes more likely.

We trained individual 2-D CNNs, one for crossline, one for inline and one for the time direction. The majority vote is then taken as the prediction of the ensemble. The trained ensemble achieves an accuracy on the F3 test data of 81%, which surpasses the 3D CNN that achieved only 74%.

4.4 Training Data Acquisition

We conducted our experiments using the DeepGeo platform for machine learning experiments on seismic data that was, at the time, developed in parallel within the same VRGeo R&D project. The system provides an intuitive interface for browsing seismic data volumes, and an annotation editor provides convenient means for labelling geobodies on selected slices. These tools help create annotations on seismic data from which training samples can later be derived (Fig. 19).

4.4.1 Weak Labelling

The annotation tools were used to create weak labels on channels and faults on the public available Parihaka[7] volume and the Penobscot[8] volume. Annotations were also

[7] https://wiki.seg.org/wiki/Parihaka-3D.

[8] https://terranubis.com/datainfo/Penobscot.

done on the F3 Block.[9] The latter annotations were used for testing and comparing the different CNN architectures on data from an unseen seismic volume.

Members of the VRGeo R&D team and later seismic interpretation experts annotated on inline, cross-line, and time slices channels and faults mostly by drawing lines. To increase the number of, we set the line width to 3 pixels most of the time. A third class, "Sediment", as background or negative class, was added later on. For the background class, areas were annotated where no interesting geobody was present. Model implementations used those annotations to create training datasets suiting the particular implementation.

4.4.2 Background Sampling

All approaches evaluated in the project were local classification methods, that look at a small window of data and result in a classification of the central pixel of that patch of data. Input for the training is differently shaped patches/sub-volumes from the original seismic volume and a label indicating the class membership of the central pixel/voxel. Accordingly, all dataset generator implementations sampled a number of patches or a sub-volumes from each annotated voxel in a volume.

In the first experiments, the dataset generators then sampled a number of background examples ("Sediment" class) from regions of annotated slices that were not containing annotated pixels. However, it quickly appeared the fact that especially non-professionals on seismic interpretation might miss some instances of target geobodies in their annotations. The automatic method for background sampling described above can lead to samples from target geobodies that have not been identified by the human interpreter, polluting the sample set for the background class. This leads to a poor classification performance of the trained model. Consequently the "Background" class of annotations was introduced and human interpreters have been asked to annotate areas not containing any interesting geologic structures using this class label.

4.4.3 Data Augmentation

We used Data Augmentation to multiply the number of samples in the training datasets. We did this by applying a set of (plausible) transformations to the set of training samples [40]. We applied the following transformations to image patches throughout all dataset generator implementations:

Flipping
Horizontal and vertical flipping was applied to patches from time slices. Since depth is a crucial parameter in vertical (inline- and crossline slices), it would not have been plausible to apply vertical flipping to them and so only horizontal flipping was used.

[9] https://terranubis.com/datainfo/Netherlands-Offshore-F3-Block-Complete.

Rotation
Random rotation between 0 and 360 degrees was applied to the sample patches. This was only applied to samples from time slices for the same reason that only vertical flipping was only applied to time slice samples. The rotation was performed by first sampling a patch twice as big as the target size, then by rotating the patch and then by extracting a patch of target size from the centre of the patch.

Scaling
Scaling was also applied to increase the variance in the datasets. The scaling was achieved by sampling a patch bigger than the target size and then scaling the patch down to target size.

Mean Subtraction
Mean-Subtraction was used on all datasets, as it is a common data normalization method that helps to stabilize the behaviour of the back-propagation during learning [41].

4.5 The VRGeo Seismic Dataset

Annotations were created on the Parihaka,[10] the Penobscot[11] and the F3 Block[12] volumes during the project form the basis for the VRGeo Seismic Dataset. The idea is to create a standard dataset for seismic interpretation, such as the ImageNet dataset for object recognition [42]. This dataset is constantly extended with additional annotations on Parihaka, Penobscot, F3 and other publicly available seismic volumes.

The VRGeo dataset serves as a base-line dataset for training and comparatively evaluating different combinations of CNN architectures, volume lenses, and optimization algorithms. Any further R&D projects in the field of Deep Learning in VRGeo are evaluated against this dataset. At the same time, the VRGeo Seismic Dataset is also available to VRGeo Consortium members. The members can use the dataset as a starting point for training their own models. Besides speeding up the adoption of ML in the member organizations, this opens the opportunity to compare results within the VRGeo Consortium without the need to share any details about the own approach or implementation.

The VRGeo Seismic Dataset, at the time of writing, contains the following numbers of samples from the three seismic volumes (Table 4).

[10] https://wiki.seg.org/wiki/Parihaka-3D.

[11] https://terranubis.com/datainfo/Penobscot.

[12] https://terranubis.com/datainfo/Netherlands-Offshore-F3-Block-Complete.

Table 4 Key figures of the VRGeo Dataset

Volume	Slices annotated	Channel samples	Fault samples
Parihaka	363	164.000	63.000
Penobscot	333	12.000	38.000
F3	357	23.000	41.000

4.6 The Detecting Faults and Channels Results

We have described our experiments with CNNs on volumetric seismic data starting from evaluating well-known current architectures. The results were promising, but a large number of parameters and the resulting need for huge training datasets pose a problem for the application of CNNs in seismic interpretation. We have shown several ways to tackle the problem: Developing custom CNN architectures that have a smaller number of parameters, creating custom tools that allow for the efficient annotation of seismic data, and using data augmentation in training dataset generation.

Our custom 2D CNN architectures perform magnificent on (unseen regions) of volumes from which training data was sampled. However, the performance was underwhelming when inferences were made on unseen data. Our 3D CNN architecture performed better than the 2D CNNs on the unseen data. Even more improvement in results was achieved using the ensemble technique, combining the predictions of several 2D CNNs. We conducted all experiments using the VRGeo Seismic Dataset. This dataset should not only serve as a baseline for the comparative evaluation of machine learning models, but it also serves as a starting point for training new architectures. Our experiments show that CNNs can successfully find geobodies like faults and channels. Further work will focus on minimizing the size of models and thus minimizing the necessary size of training datasets. In additional further work will also aim at further increasing the accuracy of predictions.

5 Detecting Salt-Bodies with 3D CNNs

The segmentation of salt-bodies in 3D seismic images is an important step in the seismic interpretation workflow, because salt-bodies can give clues to possible hydrocarbon migration pathways. However, the process of outlining those bodies can be time-consuming when done manually, which motivates the usage of (semi-) automated methods. Waldeland et al. [43] proposed to use a convolutional neural network (CNN) to classify each voxel in a seismic volume as salt or not salt. However, this approach includes a lot of redundant re-evaluation due to a large amount of overlap of the receptive fields surrounding adjacent voxels, which makes the inference at volume-scale computationally expensive. Furthermore, there is a trade-off between localization accuracy and the chosen context size.

We used a convolutional encoder-decoder with skip-connections instead, which takes in an image-patch and directly outputs a segmentation of the same size. By means of voxel-weighting, the network is encouraged to pay particular attention to the border between salt and adjacent sediment (not salt). Two encoder-decoders are trained independently on inline-/crossline-patches and time-slice-patches, respectively. At test time, the predictions of the two networks (inline-, crossline- and time-views) are aggregated to obtain a segmentation of the whole volume. To cope with the limited data available, we chose two data augmentation techniques, which have not been used in this context (the use of oblique slices and the generation of artificial salt-profiles). Volume-scale predictions of voxel-based classification approaches can take many hours, whereas the proposed method segments an example volume in just under ten minutes, aggregating three different views.

5.1 Seismic Interpretation Recap

As hydrocarbons migrate upward towards the surface over time, they can get trapped by particular formations of impermeable materials. Components of such traps cause the hydrocarbons to pool in the reservoir rock preventing them from migrating further. These components can be anticlines of impermeable rock, faults, and salt-bodies [44]. While delineating salt-bodies is of particular interest for hydrocarbon exploration and reservoir modeling, this process can be time-consuming when done manually [43], which motivates the usage of (semi-) automated methods to alleviate the workload.

Various methods have been proposed ranging from the direct extraction of salt-boundaries based on engineered features to the classification of engineered features with machine learning (ML) techniques. In recent years, convolutional neural networks (CNNs) have led to breakthroughs in various computer vision tasks, such as classification and segmentation in various domains. The core improvement of this kind of approaches is the introduction of trainable filters as parts of a neural network, which can learn to extract those features from the data that are most useful for solving the trained task. An advantage of this approach is that the parameters of the convolutional filters can be adjusted along with the weights of the neurons in the networks, using the same backpropagation method.

In the following, we will provide a brief overview of signal processing procedures that use engineered features, continue by presenting very recently developed CNN architectures to classify and segment salt-bodies and propose a modified procedure aiming to support geophysicists in this task. Although methods have been proposed for both pre-stack and post-stack data, we will concentrate on analyzing of post-stack seismic datasets in this article.

5.2 Signal Processing Procedures

5.2.1 Features

One way to extract salt-body boundaries is to use handcrafted filters to extract attributes sensitive to regions on or close to the salt-boundaries. These approaches include edge detectors [45] and gradients of texture attributes [46–48]. Zhang and Halpert [49] and Haukås et al. [50] use deformable models (active contours). These models start with an initial shape, which is gradually deformed by an iterative update rule until it converges to the salt-body outline. Other methods extract multiple attributes such as special types of contrasts, dissimilarity, homogeneity, and semblance from seismic images and feed these feature vectors to machine learning models, such as multilayer-perceptrons or support vector machines to make a decision [51]. Wu [52] defines salt-likelihood as a measure of planarity variation perpendicular to seismic reflectors to extract salt-boundary surfaces.

5.2.2 Classification with CNNs

Waldeland et al. [43] use a voxel-wise binary classification approach in which a fairly large filter of dimensions 65 by 65 by 65 voxels is used to classify the center voxel within this receptive field as being salt or not salt. Although networks seem to identify salt-bodies based on texture (chaotic, incoherent, internal reflections) and shape (deformation of adjacent layers combined with high-amplitude top-salt-reflectors), a large receptive field is crucial to provide the network with the necessary local context to make an informed decision. The network consists of four convolutional and two fully-connected layers. The convolutional layers learn filters during training that are most useful for the upcoming decision and feed the features to the subsequent fully-connected layers, which aggregate the information (also in a learnable way) and pass it to a classifier (Softmax). To obtain a classification of the whole volume, the network is applied to every voxel in the volume. The underlying goal of using a Deep Learning approach is for the network to learn how salt-bodies can look like aiming at a high-quality prediction of previously unseen data.

To artificially increase the amount of training data, the 65 by 65 by 65 voxel training samples undergo random and appropriately chosen transformations, like rotations and stretching, before being fed into the network to reduce the generalization error. Waldeland et al. [43] use the Netherlands offshore F3 block (dGB Earth Sciences B.V. [53]) for training, validate their model on similar-looking slices of the same dataset, and show that their model generalizes well even after training on only one slice of annotated training data. Ildstad and Bormann [54] apply the network of Waldeland et al. [43] to the classification of seismic facies in the Netherlands offshore F3 block. Chu and Yang [55] expand on this network by proposing modifications to increase computational speed (by means of sparse sampling of the receptive field) and apply architectural changes, like the introduction of max-pooling layers and an increasingly growing number of feature maps with each convolutional layer of the network. These changes lead to faster learning in the early stage of training. Di et al. [56] also apply a classification network (the output is a probability for one voxel of interest) to seismic facies classification.

While these networks can generalize to at least similar looking slices of the same volume, voxel-based classification approaches have drawbacks. There is a lot of redundancy and re-evaluation due to a large overlap of the neighborhoods surrounding adjacent voxels, which makes the inference at volume-scale computationally expensive. Furthermore, there is a trade-off between localization accuracy and the chosen context (receptive field size), as has been pointed out by Ronneberger et al. [57]. While a fairly large receptive field has to be chosen to provide the necessary context, this leads to a certain degree of imprecision in the prediction, which can manifest in a false classification of nearby sediment as salt. Lastly, each voxel is classified independently, given its respective neighborhood; that is, predictions of adjacent voxels are not taken into account when a new voxel is classified.

5.2.3 Segmentation with CNNs

In the aforementioned classification networks, each voxel is classified by passing a local neighborhood of that voxel through the network, which then provides a single output that can be interpreted as the probability of that voxel being salt. An alternative to this approach is semantic segmentation where the model takes as input a relatively large image (compared to the patches used in classification approaches) and translates it into a corresponding output image, of the same size as the input, where the values of the pixels represent their class membership. A successful architecture for semantic segmentation in biomedical image processing is U-Net, developed by Ronneberger et al. [57]. Instead of processing each pixel/voxel individually, segmentation networks like U-Net jointly process all pixels in the input to obtain a segmentation at the output. This method thus overcomes the aforementioned drawbacks and can be very beneficial when particular preferences over segmentation results in the loss function should be encode. Ronneberger et al. [57] apply their network to segment images displaying multiple cells. Wherever cells meet, distinguishing the boundary between them becomes challenging due to the lack of contrast. Ronneberger et al. [57] introduce a weighted penalty for making wrong decisions in these regions to the loss function to be able to promote learning of features describing the correct segmentation here. Similar strategies could be applied in seismic segmentation tasks as well (compare the following section about Methods).

In the context of seismic interpretation and concurrent with our work, several similar architectures, namely encoder-decoder CNN models resembling the U-Net, have been proposed. Zhao [58] proposes an encoder-decoder CNN without skip-layer connections (see below) and applies it to seismic facies segmentation. He validates his method in a similar manner to Waldeland et al., namely, on nearby slices. Furthermore, Zhao compares the voxel-wise classification approach to his segmentation approach and mentions a longer training time for the encoder-decoder compensated for a shorter inference time.

Alaudah et al. [59] apply a similar encoder-decoder network to salt-dome and fault segmentation and introduce a strategy that incorporates weak labels to alleviate the burden of having to annotate large amounts of training data. While their prediction does cover salt-boundaries, a substantial portion of adjacent deformed rock layers is also (falsely) identified as belonging to the boundary, which results from the usage of weak labels. Shi et al. [60] apply a similar encoder-decoder to the task of delineating salt-bodies. Their training-set consists of salt-body labels interactively generated with the method proposed by Wu et al. [61] on inline-slices of the SEAM Phase 1 dataset [62] as ground truth. They validate the model performance on a number of visually dissimilar slices not previously seen by the network.

Concurrent with our work, Alaudah et al. [63] published a paper containing a detailed comparison of encoder-decoder networks applied to image patches and whole images for the task of seismic facies segmentation. Due to the high degree of correlation between close-by slices, they point out that validations in other papers (validation on similar-looking slices) could be improved upon and propose a way to compare different models applied to the same task (e.g., seismic facies classification). In addition to their baselines for the two types of models, they also compare against standard augmentation techniques and using skip-layer connections (as was proposed by Ronneberger et al. [57]).

5.3 Methods

To obtain a segmentation for a whole 3-D volume, we trained two identical networks. The first network was trained on 256 by 256 voxel patches from inline- and crossline-slices as well as oblique slices perpendicular to the inline-/crossline-plane. These slices are perpendicular to the earth's surface and show profiles of layers (compare to Fig. 20). Since salt-bodies generally migrate upwards, the deformation of the layers above is visible on these slices. The inline- and crossline-directions are determined in the acquisition process and are somewhat arbitrary for our interpretation purposes. This is why it makes sense to train a network on inline, crossline and oblique patches perpendicular to the inline-/crossline-plane jointly. The second network was trained on 256 by 256 voxel patches from time-slices, which are oriented parallel to the earth's surface. While salt-body profiles can be distinguished on time-slices as well, segmenting them without the respective context from inline- and crossline-slices can be difficult. To avoid spurious predictions on adjacent patches, we augmented the patches of interest by one adjacent patch on each side. The input to the network was thus of dimensions 256 by 256 by 3 by batch size. For the final prediction, the inline-/crossline-network passed through the entire volume for all crosslines and again for all inlines. The time-slice-network did the same for the time axis. The resulting three volumes represent the salt-probabilities from the three views. The three volumes were then combined by means of a weighted average.

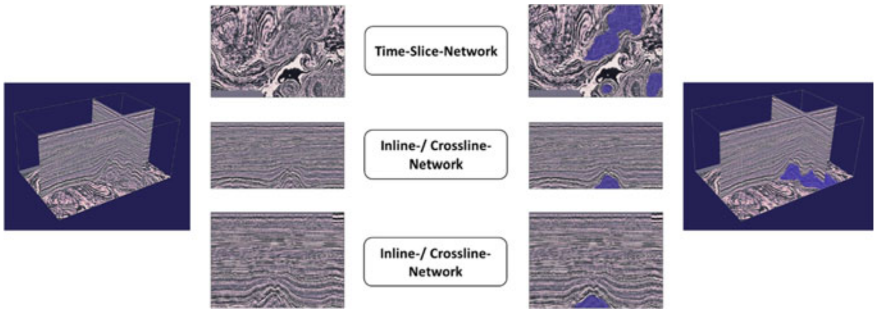


Fig. 20 Based on the input volume (left), the time-slice-network segments patches in the time-plane, which are aggregated into predictions of the entire volume from the time-view. The inline-/crossline-network does the same for the inline-and crossline-views, respectively. The three volumes are then aggregated, forming the final prediction of the network (right). Each view captures the seismic volume from a different perspective and regularizes the effect of the other views on the final prediction

5.4 Dataset and Annotations

To train and validate the model on the publicly available Netherlands offshore F3 block properly, we annotated the whole volume as salt and not salt. This extensive annotation allows us to sample (double-) oblique slices for the training-set. Furthermore, it facilitates an in-depth validation of the model going beyond the validation on single slices. Since these annotations are not straightforward in some cases, we were advised by an expert geophysicist in the VRGeo Consortium with many years of experience in the industry. Several annotation-passes were performed through inline-, crossline- and time-slices to ensure consistency of the annotation. To improve the annotation coherence from the three views, the resulting volume was smoothed and thresholded.

5.5 Data Augmentation

Patches from axis-aligned slices were augmented by patches from (double-) oblique slices. For the training of the inline-/crossline-network, patches from an inline-slice were rotated around their vertical axis at a random degree α between 0 and 360 (compare to Fig. 21), where a rotation of 90° results in the corresponding crossline and a rotation of 180° is a mirrored or horizontally flipped patch. The resulting patches remain perpendicular to the inline-/crossline-plane and capture seismic layers from different angles. These oblique patches were rotated once more, this time around their horizontal axis by angle β. This rotation was constrained to be between -15 and 15° to roughly maintain their orientation (a rotation by 90° would result in a

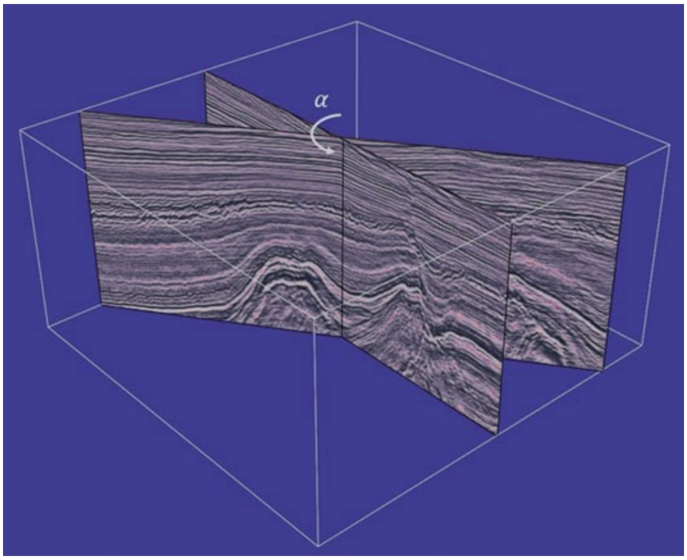


Fig. 21 An inline-slice is rotated around the vertical axis by angle α and then around the horizontal axis by angle β (not shown)

time-slice-patch). For the training of the time-slice-network, patches from a time-slice were rotated in-plane and then randomly around one of their axes between − 10 and 10°, again to roughly maintain their orientation.

To further increase the diversity of the patches containing a salt-body profile, we took the binary label for a respective patch and deformed the salt-label (see Fig. 22). To do that, we first generated a sequence of smoothly and slowly varying values. This sequence represents displacements of the normal vectors along the salt-boundary. The salt-boundary was extracted from the label patch by means of the gradient and normal vectors are perpendicular to this boundary. The values in the displacement sequence were now mapped to points on the salt-boundary and the label was deformed by the respective value of the sequence along the normal vector. Given this deformed binary label patch, we used symmetric diffeomorphic image registration [64] to find a continuous mapping f from initial patch l_{init} to the modified label patch l_{mod}. The deformation field f was then applied to transform the original seismic image pinit (corresponding to the label patch l_{init}) into a modified seismic image patch p_{mod} corresponding to the modified label patch l_{mod}. The resulting seismic image patches show altered salt-body profiles. Although the texture within the salt-body is slightly smoothed, the deformation of adjacent rock layers is consistent with our modified label.

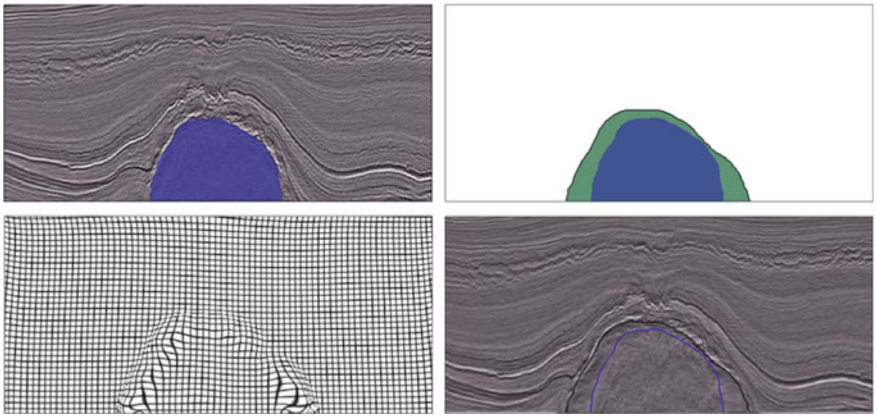


Fig. 22 The binary label mask l_{init} (first row, first column, displayed on the initial patch p_{init}) is deformed into a different binary label l_{mod} (first row, second column). The elastic deformation field f is determined as a mapping from l_{init} to l_{mod} (second row, first column). The initial seismic patch p_{init} is now deformed by f resulting in an altered salt-profile p_{mod} (second row, second column)

5.6 Training-, Validation- and Test-Set

Since we have only one volume at our disposal, we split the volume into training-, validation- and test-volumes, from which patches for the respective sets were then drawn. The volume has dimensions 950 by 650 by 462 and captures three salt-bodies of varying size (compare Fig. 23). The training-volume captured crosslines from 1 to 694, (all) inlines from 1 to 650, and (all) time-slices from 1 to 462 and contained a large portion of the major salt-body and a very small one. The validation-volume extended over crosslines from 695 to 950, inlines from 1 to 394, and (all) time-slices from 1 to 462. This volume included a part of the major salt-body in the middle. Finally, the test-volume contained the third salt-body and was intended to measure the networks capability to generalize to previously unseen data. It extended over crosslines from 695 to 950, inlines from 395 to 650, and (all) time-slices from 1 to 462. For the training of the inline-/crossline-network, we drew 30,000 patches from the training-volume with the aforementioned techniques, whereas the time-slice-network was trained on 20,000 patches.

The validation-sets for the two networks consisted of patches along the respective axes, i.e., to validate the inline-/crossline-network, the validation-volume was sliced up into inlines and crosslines (with overlap along the time axis). The network made several passes to obtain a sub-volume (or a whole volume) from predictions on 256 by 256 patches. The network, to predict the validation-volume, first predicted all inlines extending from time-slice 1 to 256 and in another pass the network predicted all inlines for time-slices from 207 to 462, where regions of overlap were averaged (the predictions are probabilities).

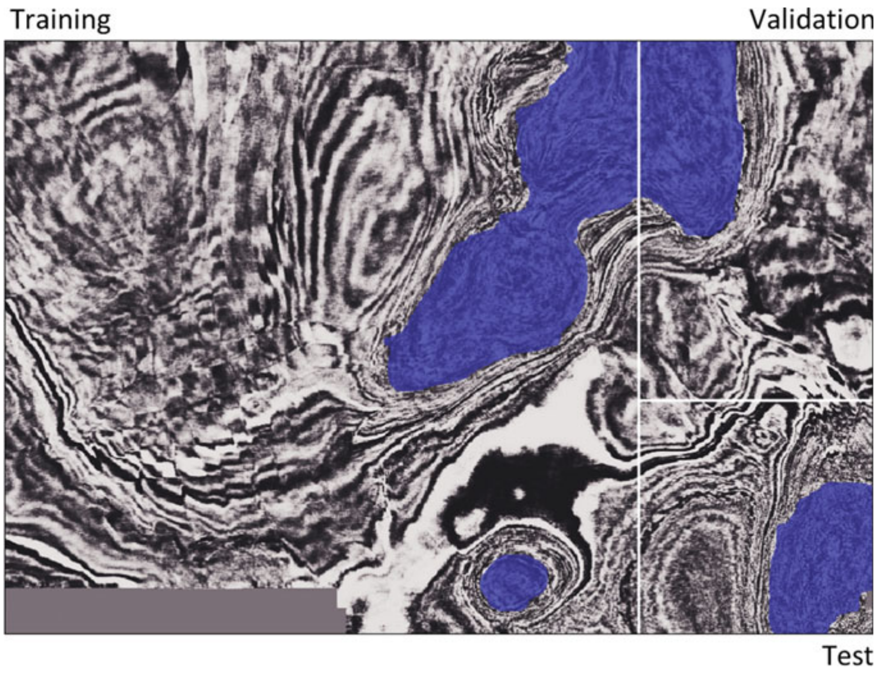


Fig. 23 Division of the F3 volume into training-, validation- and test-volumes

5.7 Architecture, Voxel-Weighting, and Loss

We adopted the recently developed QuickNAT architecture [65], which delivered superior results compared to other segmentation architectures in the context of segmenting neuroanatomy. QuickNAT extended the U-Net by making several important alterations; for instance, it included un-pooling layers [66] and dense connections [67] in each encoder/decoder block to improve gradient flow. QuickNAT consists of four symmetric encoders and decoders separated by a bottleneck layer (compare to Fig. 24). Encoders and decoders on the same resolution stage are connected with skip-connections to provide the decoders with additional context and capture resolutions at different scales. Transpose convolution is associated with certain drawbacks, such as checkerboard artifacts [68]. This is why the authors replaced them with un-pooling layers. Each dense block consists of three convolutional layers (with a kernel size of 5 by 5 and 64 output channels), where each convolutional layer is preceded by a batch-normalization layer and a Rectifier Linear Unit (ReLU) layer. The first two convolutional layers are followed by a concatenation layer that merges the output of the preceding layer with additional inputs (compare to Fig. 24). The dense blocks in the encoder part are followed by a 2 by 2 max-pooling layer to down-sample the respective feature maps.

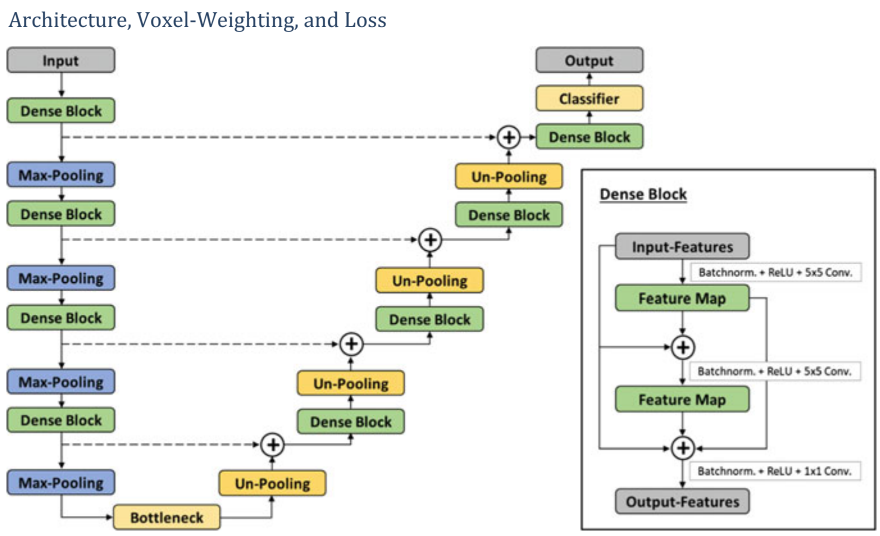


Fig. 24 The QuickNAT architecture consists of a contracting path (encoder) and an expanding path (decoder), separated by a bottleneck. Each dense block consists of three convolutional layers preceded by batch-normalization and a Rectifier Linear Unit (ReLU)

The bottleneck separating the encoder and decoder consists of one convolutional layer and a batch normalization layer. By means of this bottleneck, the network is forced to find useful and general representations of the inputs. In the decoder, each dense block is preceded by an un-pooling layer to recover information lost in the max-pooling layers of the encoder. In the classifier block, the output of the last dense block is converted into N feature maps (N equals the number of classes) followed by a Softmax.

Voxels in the image patch were not treated equally, but they were instead weighted to make up for the class imbalance [69] and emphasize the boundaries between the two classes. In a first step, voxels belonging to the less frequent class (salt) received a larger weight, i.e., not-salt-frequency divided by salt-frequency. In a second step, the network was encouraged to pay special attention to the boundary between the two classes, i.e., the separation of the salt-body and adjacent sediment. We can choose the weights on or close to the boundary based on our confidence in the accuracy of the labels. Since manual annotations are generally subjective and are often based on specific views, which takes only a portion of the 3-D data into account, we chose to place large weights on regions close to the boundary but not on the exact boundary (compare Fig. 25). We applied Gaussian smoothing to the label-patch's gradient magnitude with σ_1 and separately with σ_2, where $\sigma_2 < \sigma_1$ and subtracted the latter from the former (and replaced negative weights by zero). Both weights were then added to form the final weights for the patch.

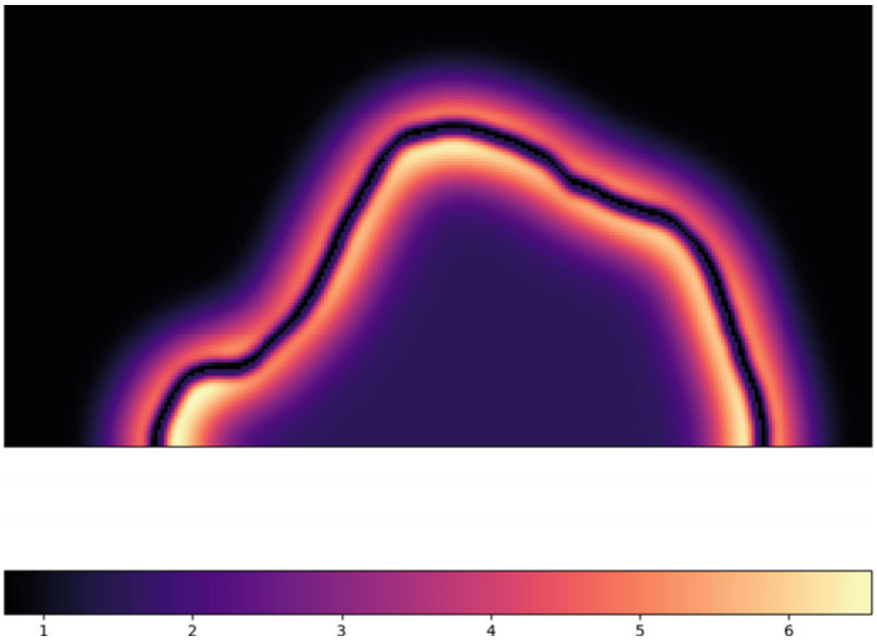


Fig. 25 Regions labeled as salt receive a larger weight than the other class (not salt) to make up for the class imbalance. Regions close to the boundary are strongly weighted, whereas the locations on the boundary do not receive additional weighting

The loss function consists of a logistic loss weighted by the aforementioned weights and a Dice loss term [70]. The logistic loss measures voxel-wise similarity between the predictions and the labels in a probabilistic sense. The Dice loss measures the similarity of prediction and the labels by means of intersection over union. For each voxel x, given weights $\omega(x)$, predicted salt-probability $s(x)$ and the ground-truth $g(x)$, the loss is calculated as follows:

$$\mathrm{L} = -\sum_{\mathrm{x}} \omega(\mathrm{x})\mathrm{g}(\mathrm{x}) \log(\mathrm{s}(\mathrm{x})) - \frac{2\sum_{\mathrm{x}} \mathrm{s}(\mathrm{x})\mathrm{g}(\mathrm{x})}{\sum_{\mathrm{x}} \mathrm{s}(\mathrm{x}) + \sum_{\mathrm{x}} \mathrm{g}(\mathrm{x})}$$

Given predictions of salt-probabilities for the inline-, crossline- and time-views, the final salt-probability is obtained by combining these views in the view-aggregation step. Hard labels are obtained by thresholding these probabilities (for example, at 0.5). For each voxel x, the final salt-probability $S_{final}(x)$ is determined by a weighted sum of the salt-probabilities from the three views S_{inline}, $S_{crossline}$, and S_{time}. We can set $\lambda_{inline} = \lambda_{crossline}$ because inline- and crossline-predictions are obtained from the jointly trained network. Because of the performance of the time-slice-network (see below), we set $\lambda_{time} = 0.2$ (and $\lambda_{inline} = \lambda_{crossline} = 0.4$).

$$\mathrm{S}_{\mathrm{final}}(\mathrm{x}) = \lambda_{\mathrm{inline}} \cdot \mathrm{S}_{\mathrm{inline}}(\mathrm{x}) + \lambda_{\mathrm{crossline}} \cdot \mathrm{S}_{\mathrm{crossline}}(\mathrm{x}) + \lambda_{\mathrm{time}} \cdot \mathrm{S}_{\mathrm{time}}(\mathrm{x})$$

The networks were trained with stochastic gradient descent with momentum and weight decay for 40 epochs on an NVIDIA TITAN X Pascal GPU (12 GB RAM). The initial learning rate (set to 0.05) was adapted by the Adam optimizer [71] during training.

5.8 Salt-Body Segmentation Results

We achieved an average Dice-score of 0.9411 (0.8849 for salt, 0.9972 for not salt) on the validation-volume. The average Dice-score was 0.9111 (0.8355 for salt, 0.9867 for not salt) on the test-volume after view-aggregation. To explore the effect of view-aggregation on the validation- and test-volumes, we first evaluated the inline-/crossline-network on the crossline-view of the respective volume. In a second step, we evaluated the inline-/crossline-network on the inline-view and aggregated the resulting volume with the crossline-view. Finally, we evaluated the time-slice-network and aggregated with the inline-/crossline-volume (Table 5).

Although the time-slice-network predictions added beneficial information, we noticed that the time-slice-network on its own performed worse than the inline-/crossline-network. While the inline-/crossline-network achieved average Dice-scores of about 0.9 in the inline- and crossline-views, the average Dice-score of the time-slice-network remained at 0.8178, which is why we chose a smaller weight for these predictions in the view-aggregation step. We also achieved an acceleration of the time needed for inference of the whole volume. It takes merely 9.6 min to evaluate and aggregate the three views forming the final prediction, compared to multiple hours for the voxel-wise approach (Fig. 26).

Segmentation networks (or encoder-decoders), like the proposed one, take in images or image-patches and segment them in one pass, whereas classification networks, like the network proposed by Waldeland et al. [43] classify an image (or volume) on a voxel-basis. Encoder-decoder networks overcome the shortcomings of voxel-wise classification approaches, like the trade-off between localization accuracy and the chosen context size and a large amount of redundant computation. In addition, they generate high-quality segmentations and do so in a fraction of the time required by classification networks (less than ten minutes in the described setting). In future work, this model could be extended to a true 3-D context, going beyond segmenting individual slices, if additional seismic datasets become available.

Table 5 The effect of view-aggregation

	Validation-volume	Test-volume
Crossline-view	0.8933	0.8658
Crossline- and inline-view	0.9351	0.8933
Crossline-, inline- and time-view	0.9411	0.9111

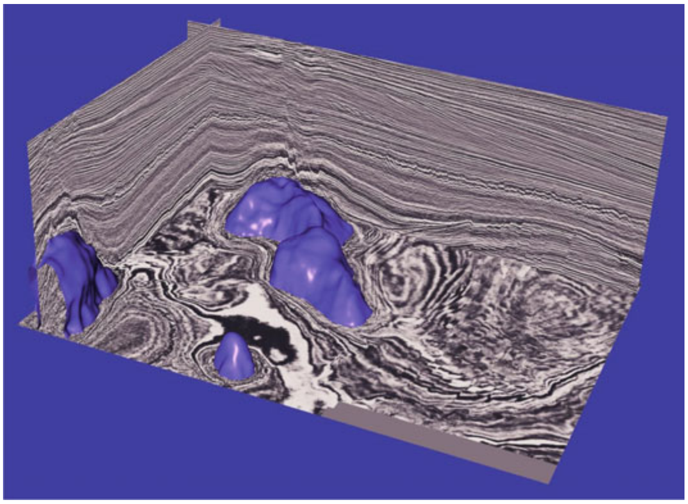


Fig. 26 Salt-boundaries are extracted as iso-surfaces of the slightly smoothed salt-probability. These surfaces closely correspond to ridge surfaces of the salt-probability gradient

6 Understanding CNNs for Seismic Feature Detection Better by Visualization

In recent years, Convolutional Neural Networks (CNNs) most commonly applied to visual image analysis have yielded astonishing results in a number of classification tasks. The areas of application include the recognition of text or speech [72, 73], various objects [74] and even faces [75, 76]. Research on automatic methods to detect geophysical structures in seismic volume data focuses on deep learning with CNNs. However, the key to automatic seismic segmentation with CNNs lies in hybrid AI, which describes a combination of artificial intelligence with human knowledge to gain the best of both worlds.

Although Convolutional Neural Networks achieved promising results in recent years, one disadvantage is their black-box property, making it difficult to understand their decisions and classification results as most of the complex processes are hidden from the user. By visualizing features of a network's layers with visualization methods like Activation Maximization or Gradient-based Class Activation Mapping, the concepts learned by a CNN can be revealed. With these gained insights into the black box, many optimization possibilities arise to further improve networks and workflow for automatic feature detection and offer domain experts the ability to contribute their domain knowledge to the automatic process.

6.1 Visualization Methods

Visualizing learned concepts and features of a CNN can help understand and strengthen trust into the classification results. Since there is no universal visualization method to be applicable to every kind of neural network and delivers useful and informative results, several visualization methods with different emphases will be explained in this section.

6.2 Activation Maximization

A common method for analyzing concepts learned by a neural network is Activation Maximization introduced by Erhan et al. [77]. The aim of this method is to generate an input image that maximizes the activation of a certain neuron in any layer. Thus, Activation Maximization can generate images that certain neurons prefer most.

The aim of Activation Maximization is to synthesize an input image x^* that maximizes the activation a_i^l of the ith neuron in the lth layer:

$$x^* = \underset{x}{\operatorname{argmax}}\big(a_i^l(x)\big)$$

The fundamental algorithm consists of three steps:

1. A random input image $x = x_0$ is generated and set as input for the algorithm.
2. The gradients $\frac{\partial \mathrm{a}_i^l}{\partial x}$ with respect to the input image x are calculated using the Backpropagation algorithm [78].
3. The direction of the gradient $\frac{\partial \mathrm{a}_i^l}{\partial x}$ is used to change each pixel of the input image iteratively to maximize each neuron's activation with a step size of η:

$$x \leftarrow x + \eta \frac{\partial a_i^l}{\partial x}$$

Steps 2 and 3 are repeated until the change in image x between two iterations is below a certain threshold or the number of iterations has reached a given maximum. Image x then represents the optimal input image for the given neuron i and thus maximizes its activation. In order to compute a correct maximization, a_i^l denotes the unnormalized activation value of the last layer in the network before applying a softmax function.

Instead of synthesizing an input image that activates a certain neuron most, Activation Maximization can also be used to create images that activate a given output class most. Thus, the optimal input image for a given class can be generated and compared to samples from the training dataset.

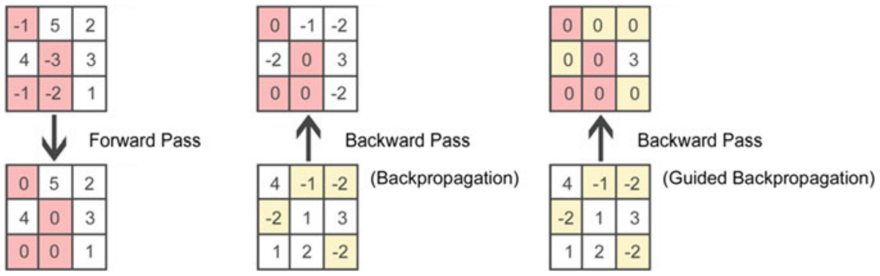


Fig. 27 Comparison of forward and backward passes. In the forward pass, the ReLU activation function sets every negative value from the bottom gradient to zero. The standard backward pass using the Backpropagation algorithm uses the top gradient as input. With Guided Backpropagation, both top and bottom gradients are combined

6.3 Guided Backpropagation

The Guided Backpropagation visualization method by Springenberg et al. [79] is an adaption of the well-known Deconvolution method by Zeiler and Fergus [80] that is able to reconstruct learned features from higher layers by performing a backward pass.

The important step in reconstructing features of higher layers is the backward pass, where an activation function like ReLU (rectified linear unit) has to be applied. In the forward pass, the ReLU function $\sigma(x) = \max(0, x)$ sets every negative value of the bottom gradient to zero. In the backward pass, the same has to be done, but with different values as depicted in Fig. 27. While Deconvolution only considers the top gradient, Guided Backpropagation applies the activation function to both top and bottom gradients resulting in fine-grained features reconstructed from a higher layer.

6.4 Grad-CAM

Originally, Class Activation Mapping (CAM) was introduced by Zhou et al. [81] to reveal important regions in the input image that contribute most to a network's decision. Nevertheless, this method was only applicable to fully convolutional networks that work without any fully connected layers. To overcome this limitation, Selvaraju et al. [82] introduced a modified version of CAM, which they called Gradient-based Class Activation Mapping (Grad-CAM).

Therefore, a neuron importance weight α_i^c for unit i and class c is defined as

$$\alpha_i^c = \frac{1}{Z} \sum_{x,y} \frac{\partial y^c}{\partial f_i^L(x, y)}$$

The first part represents a global average pooling operation and the second part the gradient of the class score y^c (before the softmax function) with respect to the feature map value f_i^L of the last convolutional layer L at position (x, y). To obtain a heat-map, a weighted combination of activation maps is performed:

$$M_c^{Grad-CAM}(x, y) = \sum_i \alpha_i^c f_i^L(x, y)$$

By passing the linear combination through a ReLU activation function, it is additionally assured that only positive influences for the class c are taken into account.

6.5 Guided Grad-CAM

With the introduction of Grad-CAM, the authors also proposed an extension of this method, which represents a combination of Grad-CAM and Guided Backpropagation. To achieve fine-grained features only in class-discriminative regions of the input image, the heat-map obtained by Grad-CAM is multiplied element by element with the features of Guided Backpropagation.

6.6 Application to Seismic Data

Applied to seismic data, CNNs can be trained to detect seismic features like faults. Therefore, a training dataset has to be created from the given labelling of faults and areas containing no faults. For demonstration purposes, the publicly available F3 dataset has been used in this chapter with self-made fault and no-fault labels (depicted in Fig. 28) to generate training data for a CNN.

6.7 Training Data

The training data can be created by gathering the neighborhood of all labeled pixels of a seismic volume dataset like F3. Using a sliding window approach, only the center pixel of an input image with a size of 32×32 pixels is taken into account for classification. If a fault in the image does not run through this center pixel, the whole image should be classified as no-fault. Visualizations with Activation Maximization and (Guided) Grad-CAM showed that faults are not recognized precisely enough and that input images are also classified as fault, if they only contain faults apart from the center.

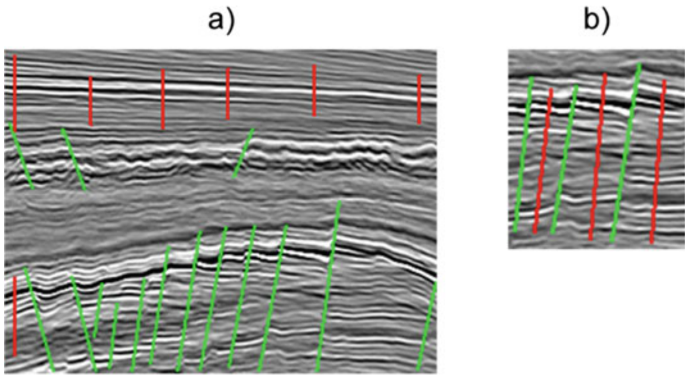


Fig. 28 Examples of labels for "faults" (green) and "no faults" (red). In the original training data, **a** there are no red lines between the green lines, while **b** shows an example of improved labeling with "no-fault" labels between "fault" labels

Better classification results could be achieved by adding additional no-fault labels between fault labels to the training data, as shown in Fig. 28. This optimization helps the network to focus the classification on the correct samples instead of a rough area.

6.8 Classification Results

First tests using a CNN to detect faults delivered coarse and rough classification results as depicted in Fig. 28. Many areas have coalesced or have been falsely classified. With the optimization of the training data, the results become much finer and show fewer false-positives (see Figs. 28 and 29).

6.9 Visualization Results

Previously shown optimizations of training data could be achieved by analyzing visualizations of learned features and concepts with Activation Maximization, Guided Backpropagation, and (Guided) Grad-CAM. In this section, visualizations before and after the optimizations will be compared to demonstrate the different behaviors of the CNN.

Activation Maximization generates optimal input images for each class that result in high activation values. After optimizing the training data, the synthetic input images show more distinctive patterns and reveal that the CNN detects faults that are positioned in the center of the input patch (see Fig. 30).

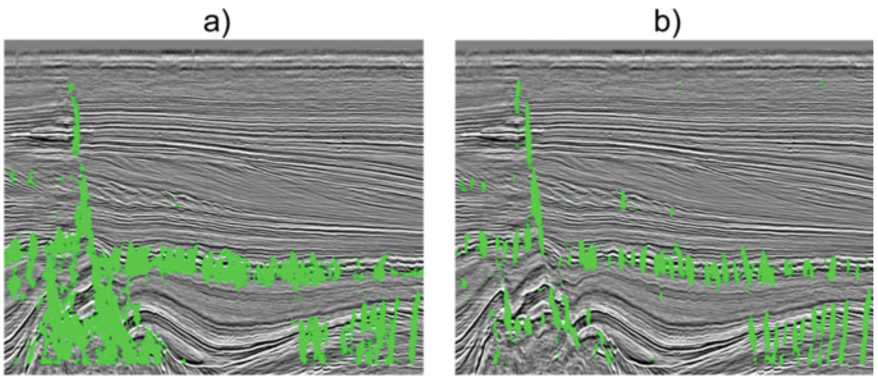


Fig. 29 Example of classification results on F3 dataset. Before optimization (**a**), many faults have coalesced and large areas of false-positive predictions are visible. With the optimization of training data (**b**), faults are separated from each other and the amount of false-positive predictions has decreased drastically

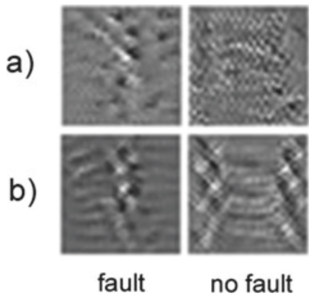


Fig. 30 Comparison of Activation Maximization results before (**a**) and after (**b**) optimization of training data. The columns show input images that maximize the activation for the classes "fault" (left) and "no-fault" (right)

With Guided Backpropagation, features in input images are reconstructed from higher layers. After the optimization process, the features got more striking and focused on the center of the image, as visible in Fig. 31.

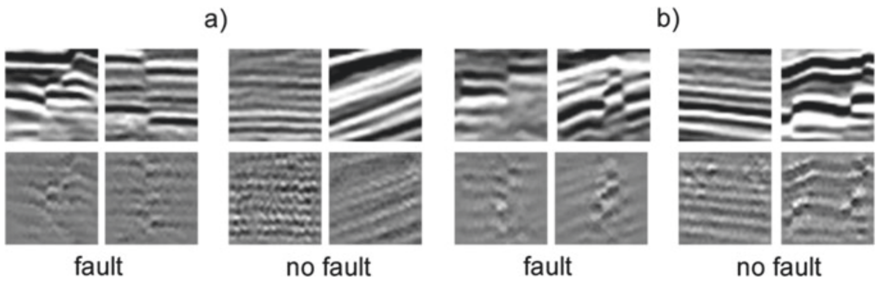


Fig. 31 Comparison of input images (top) and results from Guided Backpropagation (bottom) before (**a**) and after (**b**) optimization of training data

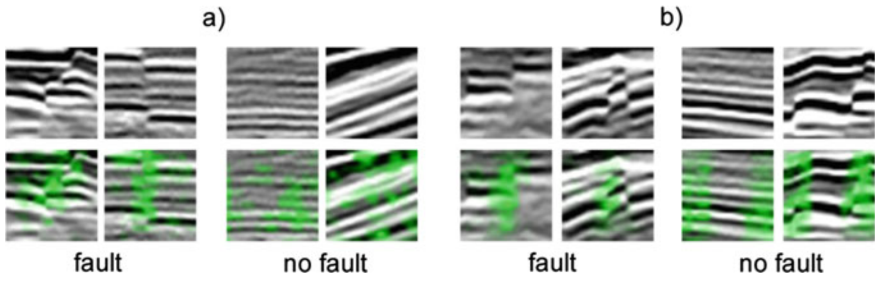


Fig. 32 Comparison of input images (top) and results from Grad-CAM (bottom) before (**a**) and after (**b**) optimization of training data

The heat-maps created with Grad-CAM are easy to understand and intuitive. Strong spots in a heat-map indicate important areas that are most discriminative for a given class. In other words, highlighted regions contribute most to the model's decision. Figure 32 depicts that after optimization the important regions for fault detection focus exclusively on the vertical center axis of the image while the "no fault" class is detected at the border.

The combination of Guided Backpropagation and Grad-CAM leads to images that show the features reconstructed by Guided Backpropagation but only in regions that are considered most important by Grad-CAM. Similar to Grad-CAM, features on the vertical center axis of the image are highlighted after the optimization (depicted in Fig. 33) for faults while the class "no-fault" shows only features apart from the center.

By visualizing learned features or important regions in an input image, a user can better understand the decision-making process and is able to visually debug a neural network. With this gained knowledge, optimizations of the training data or the network architecture as well as the whole workflow can be derived to reach better performance or classification results. Visual debugging is an intuitive way to identify

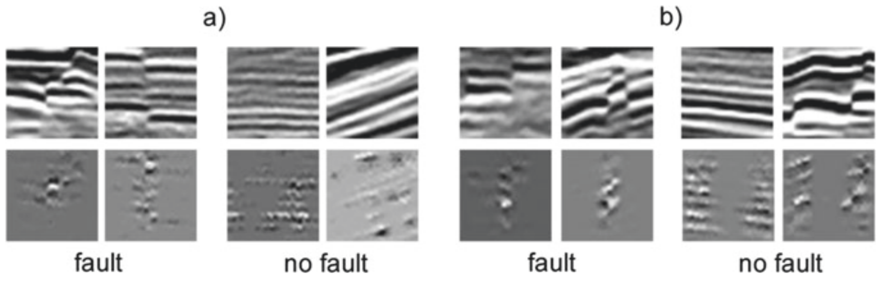


Fig. 33 Comparison of input images (top) and results from Guided Grad-CAM (bottom) before (**a**) and after (**b**) optimization of training data. Note, that the last input image in (b) contains several faults, of which none run through the center of the image. Therefore, it is correctly labeled as "no-fault"

problems and limitations of CNNs and has proven to be applicable to the automatic interpretation of faults in seismic data.

7 DeepGeo—Streamlining AI Operations for Seismic Interpretation

This section is about implementation, workflow integration, sense making, and usability.

DeepGeo is a software platform for efficient collaboration of machine learning researchers and seismic interpretation experts and for rapid validation and evaluation of machine learning methodologies on seismic data. The system addresses certain challenges that an organization will face when attempting to efficiently tie machine learning research and development (R&D) into their production operations, here for the domain of seismic interpretation. We detail these organizational and ergonomic aspects and show how the DeepGeo platform addresses them. Then we give an overview of the technical realization.

7.1 Digital Transformation in the Oil and Gas Industry

The perspective that the industry is seeking in employing Machine Learning is that it could automate time-consuming and repetitive tasks that are currently binding a valuable workforce. In addition, the industry has always tried since the times of expert systems to convert human knowledge into soft- or hardware. Nowadays, this is called Digital Transformation and the knowledge target is Artificial Intelligence (AI). This holds for the industry as such, but for the oil and gas industry in particular.

An example from practice in the oil and gas industry is the annotation of fault systems. Traditional methods can help identify areas where faults are likely existent, yet they still fail in identifying individual faults with high precision. Thus, a human interpreter needs to go through the seismic and annotate at least the prominent faults by hand. An AI solution that automatically identifies and annotates faults with high precision would relieve seismic interpreters of this time-consuming task and free their work-force to focus on more specific tasks where their expertise is needed or contribute e.g., in strategic decision making whether to drill or not.

Another reason for aiming to adopt AI/Machine Learning (ML) methods in subsurface exploration is that AI methods, once successfully adopted, often outperform humans by orders of magnitude in speed and quality of results.

An example from practice is the diagnosis of cancer in CT images. A trained oncologist takes about half an hour to form a diagnosis, including making sure nothing slipped his perception. In contrast, a Convolutional Neural Network can identify even subtle beginnings of cancer that are practically indiscernible to a human expert and produces a result in seconds [83]. Transferred to the domain of subsurface exploration, this could mean an immense speed-up of data interpretation and ultimately decision making.

In the following, we will layout the ergonomic and organizational aspects of experimentation with machine learning models, the above-described collaboration between ML and domain experts, and general challenges when integrating ML into production operations. Then we will give a describe the DeepGeo system's technical realization. Finally, we conclude with a summary and an outlook on further work planned in the DeepGeo project.

7.2 Challenges in Adopting Machine Learning in an Organization

With the potential of machine learning comes, as with every novel technology, also certain challenges that an organization, be it an academic institute or a commercial enterprise, has to face when incorporating it into its operations. For Machine Learning, these challenges mainly originate from

- the complexity of the technology itself and
- the interface to the application domain.

Machine Learning experts always have to have a certain amount of understanding to produce result data that proves to be useful in the work of the domain experts. In turn, it is often not obvious to domain experts where exactly ML can help in their domain and where not. Thus the best results are achieved when ML and domain experts work in cooperation with each other.

When ML research and development is to be tailored into a specific domain, there are also requirements reaching into the application domain itself. Sample data from the domain needs to be made available to the ML experts so that models can be developed and the output of the ML research (result data from ML models) can be evaluated in light of domain specific quality measures. Of course, that part, the interaction between the ML experts and the domain experts, should as well not be a bottleneck that would ultimately slow down the R&D process as a whole.

Ultimately, it is the aim, in the case of a commercial enterprise, to use the models developed in ML R&D in production applications. In the domain of subsurface exploration, this will most often mean applying models to newly acquired data for analysis and decision-making. In addition, as in other areas where AI is applied, it does not take an AI researcher to run a model but to develop it. Thus, it would not be efficient to consider the ML experts with applying models to data in production. Instead, ML models should be delivered to the domain specialists as functional modules that work as ready-to-use products. Additionally, with a small amount of training, domain experts could even be enabled to improve existing models with additional data or even to train models from scratch (at least in the domain of seismic interpretation where individuals are usually trained in physics and engineering).

7.3 Ergonomic Aspects of ML Research and Development

The main challenge in ML research is that machine learning is itself a data-heavy field. A researcher needs to handle a variety of different data that is incarnated as big collections of files that are unique to a particular model or instance of a model, but that can also be shared between models or model instances. The main data entities that an ML researcher is handling in every day's work are:

- Model implementations—models' source code and ML framework libraries
- Trained instances—e.g., the weights of a trained neural network and additional parameters
- Training datasets—collections of annotated data for training, most often already encoded in a format that can directly be loaded by the training procedure
- Logs of training runs—needed for comparison of learning methods and understanding of model performance
- Visualization data—for some methods, there are approaches for visualizing certain aspects of a model; some of these methods are not instantaneous, need lengthy computation and produce a great amount of data as a result.

Some of the above can combine with a certain degree of freedom. For example, it is possible to train a certain implementation with different datasets or run a trained model with different implementations of inferences code.

Another aspect is that today there is still a great amount of trial-and-error involved in developing machine learning solutions because ML researchers are only just beginning to understand the underlying mechanisms that drive learning and performance in ML models.

Usually, the work will start with one or more annotated datasets from the application domain. These datasets are often transferred into a format that can be loaded directly by the training procedure. In the next step, the ML researcher will implement a model of a certain kind and start training the model with one of the standard learning algorithms. If a combination of model and learning algorithm seems to be performant, the ML researcher will try to optimize the model in to gain as useful as possible results. Often pre-processing has to be applied to input data and post-processing has to be done on the model's output to make it useful in the application domain.

In practice, ML R&D is not as straightforward as described, and often, hyper-parameter sweeps are employed to find an initial starting point or to discover an optimal model. In the case of such a hyper-parameter sweep, there are several data items generated in each iteration.

Another challenge is that the field of ML is rapidly developing because it is entirely done digital and to keep up with competitors an organization must be able to adopt new approaches in the field within a very short time. Thus, an ML expert's business value lies almost exclusively in his/her understanding of the technology and the ability to quickly make newly developed approaches available as functional modules within an organization. As a consequence, there needs to exist an environment in which ML models can be realized and evaluated quickly and efficiently, involving as least friction as possible.

All this constitutes the need for careful data management and tracking of operations. Data management is usually done by having a naming scheme for directories and files and maintaining a certain directory system so that the data items are stored in an ordered fashion. Tracking of operations is usually done in a lab Journal. Both things, the data management and the lab journal, are substantial parts of an ML researcher's work.

Automating and data management and operations tracking and thereby relieving the researcher of this rather boring but inevitable work can significantly increase speed in R&D. Moreover, especially when there is a whole team of researchers working on a problem, automation can help to avoid errors due to an individual failing to comply with the data management or tracking system. In addition, a convenient environment for automating lab operations can optimize collaboration among the researchers in a team. This also applies to the productive use of ML methods: the modules provided by the ML experts must be easily usable by the domain experts in their daily work.

The DeepGeo system implements such an environment for automating lab operations for ML R&D. It does so in several places, like general data management, the job execution system, the operations log, and through a convenient web interface. We describe all of those system components of the system in detail in the respective sub-sections to the System Architecture section.

7.4 Ergonomic Aspects of Collaboration

As mentioned in the introduction, it is vital for successful ML R&D supporting a certain domain that the ML expert can gain insight into the application domain up to some extent. This allows the ML expert on one hand to be able to provide model results in formats useful for the domain experts but opens on the other hand ways for the ML expert to understand workflows in the application domain and gain insights into where the places are where machine learning can help in the domain.

It is thereby advisable to let the ML R&D team work closely with the domain experts. The ML researcher should always think of the final model as a (company-intern) product that should work out-of-the-box for the domain experts.

Domain experts, in turn, can help the ML researcher by providing good technical documentation about the data formats and processes used in the application domain so that interfaces can be defined for passing example data to the ML research and integrating ML model results into the processes in the application domain.

The DeepGeo system provides an integrated working space for both groups, ML researchers and seismic interpreters. Seismic volumes and machine learning models are managed in a shared interface that is intuitive to use for both groups while still making it possible to work for the expert groups in their respective professional ways.

This shared space concept enables the ML experts to work with seismic data themselves and for the seismic interpreters to apply trained models for analysis on their own or even try to train or refine models when new training data becomes available.

7.5 The DeepGeo Platform

With the considerations from the last section in mind, it becomes obvious that a technical infrastructure supporting the efficient use of AI technology in an organization needs to be tailored into the target domain to a certain extent.

DeepGeo provides a virtual space for the collaboration of machine learning and seismic interpretation experts that streamlines R&D for machine learning in seismic interpretation and minimizes friction when bringing the results of ML R&D to productive uses by domain experts.

The platform is the implementation of the rationale just laid out. It was born out of the experiences gathered in machine learning projects for seismic interpretation conducted within the VRGeo Consortium. From the beginning, there was a necessity to efficiently manage seismic data for experimentation. Also experimentation with ML models itself is data-heavy since not only the code, but also trained models and training logs have to managed in a way that results are traceable and reproducible. Finally, when more than one team is at work, there need to be defined processes in place so that R&D as a whole stays manageable. Individual tools were unified within one platform. This was the start of the DeepGeo platform. In later stages, the system was refined for practical use in collaboration with experts from the industry under the umbrella of the VRGeo consortium.

7.6 System Architecture

The DeepGeo system is a service-oriented architecture (SOA) comprised of containerized services. Each of the services is independent of the others and almost all services can easily scale just by increasing the number of running container instances. We consider this a standard for current enterprise information system architecture [84]. Since the system is comprised of independently running services, it can be deployed all-in-one on a single machine or it can be scaled out in several ways, suiting different scenarios, e.g., having several compute nodes enables an increased number of parallel experiments, speeding up hyperparameter sweeps. In another scenario, increasing the number of front-end servers supports a high number of users

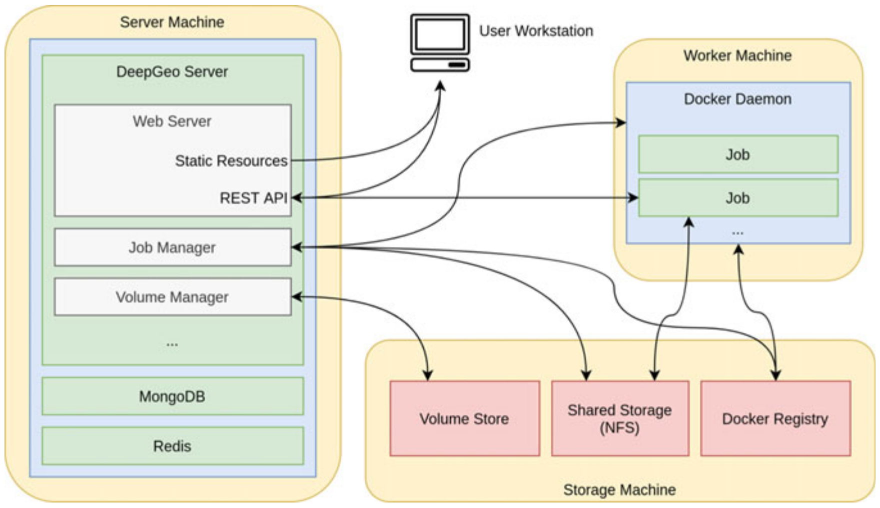


Fig. 34 The DeepGeo system architecture

working with seismic data in the web interface, e.g., for an annotation initiative with the aim to create a big amount of training annotations in one sprint (Fig. 34).

The architecture subdivides into three main layers that more or less reflect the main functionalities of the system:

- The Storage Layer
- The Execution Layer
- The Collaboration Layer

We will discuss each of the layers and the services they are comprised of in detail in the following sub-sections.

7.6.1 The Storage Layer

The storage layer is concerned with the management of the different kinds of datasets used in seismic interpretation ML workflows:

- Raw Seismic Data (single channel dense volumetric data)
- Annotations on Seismic Volumes
- Images containing the code for Machine Learning Models
- Training datasets for Machine Learning Models
- Parameters of trained Machine Learning Models.

Seismic data and annotations are stored using HDF5[13] as the underlying storage engine. HDF is a widespread storage technology that has long proven to be a solid

[13] https://www.hdfgroup.org/solutions/hdf5/.

foundation for managing huge datasets in the HPC area. With version 1.10.0, HDF introduced the ability to organize access to a data file according to the Single-Writer-Multiple-Reader pattern where one process can exclusively write to the file, but several processes can concurrently read from the same file.[14] The coordination of Read/Write operations from several independent service instances is done via Redis using the RedLock algorithm [85].

To maximize interoperability with other software packages and the overall usefulness of the system, we organize the storage of seismic volumes as follows: A raw seismic volume and all its annotations are stored together in one HDF5 file. This enables system administrators to quickly add or remove a seismic dataset just by adding/removing it from the directory where the H5 files are located. It enables easy import/export of seismics from/to other software packages commonly used in seismic interpretation such as Petrel[15] or Matlab[16] since HDF5 is a widely adopted technology in the domain. DeepGeo notices when HDF5 files are added/removed from the storage directory, and it updates its internal state accordingly.

DeepGeo provides a REST API giving access to seismic data and annotations. This API is used by the components in the Collaboration Layer, e.g., the Seismic Viewer and the jobs. The API can produce and consume web-compatible image formats such as PNG, mostly for the Collaboration components, and Pickled NumPy[17] Arrays for communication with jobs.

A client library for the Python programming language takes care of communicating with the REST API and, when run inside a DeepGeo job, takes cares of connecting to the correct server and authenticating the communication, so that a developer can read/write seismic data with a single line of code.

Seismic data can be stored as single-channel UINTx, INTx, FLOAT, or DOUBLE three-dimensional datasets. Annotations are treated as layers on the original seismic and are single-channel three-dimensional datasets that do not need to be of the same datatype as the raw seismic. Raw seismic and annotations are bundled into a HDF5 Group.

ML model data is, apart from the actual code, which is stored as Docker[18] images, the data that the implementations use to train models and store the results of the training (e.g., model weight for neural networks). What is actually stored is entirely up to the particular implementation: DeepGeo manages virtual volumes that are mounted to specified places in the jobs' container file systems when they are executed. The system cares for mounting, caching, and tracking operations on those virtual volumes.

[14] See: https://support.hdfgroup.org/HDF5/Tutor/swmr.html.

[15] https://www.software.slb.com/products/petrel.

[16] https://www.mathworks.com/products/matlab.html.

[17] https://www.numpy.org/.

[18] https://www.docker.com/.

7.6.2 The Operations Log

The Operations Log (OpLog) is a sub-system in DeepGeo that keeps track of all operations in the system. This implements an automatic lab journal, as explained in the last section. Since all operations in DeepGeo are encapsulated in containers, the OpLog saves which containers are run, which data volumes were mounted on those containers, a list of start parameters for the run, and which user started a run. Thereby the system automatically creates a comprehensive overview of all operations that have been performed, enabeling to re-run each of these operations, for example, with learning parameters changed to see the result in the trained model.

7.6.3 The Execution Layer

All ML operations performed in DeepGeo are encapsulated in Docker images with special properties. They run as Docker containers. The system manages the setup and execution of job instances. The main advantage of this pattern of total encapsulation in containers is that a job image contains a whole stack of an operating system, libraries, and tools in exactly the versions needed for the particular implementation to work. This means that any ML framework can be used for model implementations, as long as it runs in an operating system supported inside Docker containers. Communication of job parameters is done via environment variables of input files in virtual volumes and there is a very simple mechanism to change the displayed state of a job in the DeepGeo interface, using specially formatted messages to stdout/stderr. All this makes DeepGeo agnostic of the particular ML technology and enables adopters to easily migrate their existing ML model into DeepGeo.

7.6.4 The Collaboration Layer

The collaboration layer harbors several components that implement user interfaces, enabling easy access to the data and convenient interaction with the system and the data. Those interfaces were designed with the aim of supporting professional workflows for both groups of domain experts (ML and SI) while still providing intuitive access to basic functionalities for non-experts in particular. ML experts should still be able to navigate seismic volumes and SI experts should still be able to at least run inference with a trained model on seismic data.

Seismic Viewer
Seismic Viewer is one of the central tools that allows conveniently viewing seismic data and annotations in a web front-end. The implementation uses WebGL[19] to render a simplified 3-D view of seismic volumes and annotations. A crossline, an inline and a time slice from the volume are displayed. A transfer function can be selected for

[19] https://webglsamples.org/.

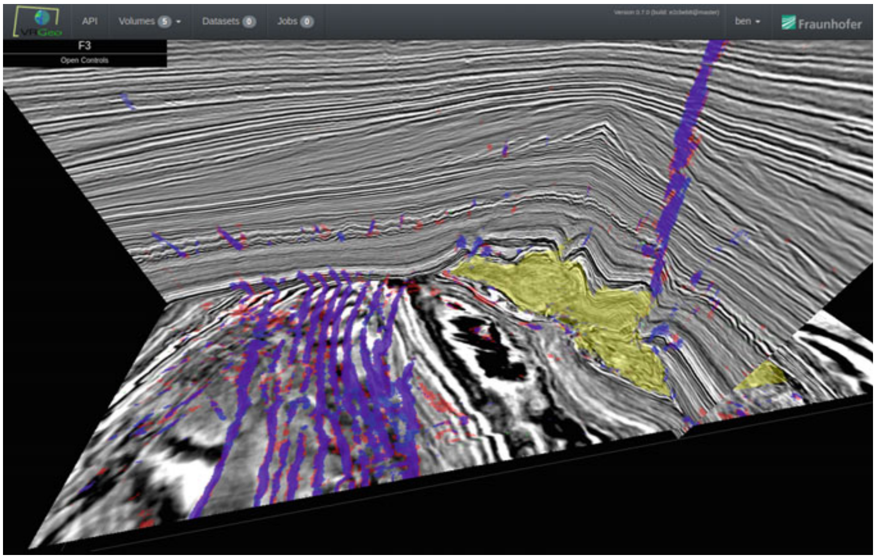


Fig. 35 The Seismic Viewer in the Web Interface showing a part of the F3 block two different fault annotations showing CNN results before and after transfer training with additional expert annotations and a CNN-generated salt annotation (yellow)

the raw seismic and annotations are treated as one-channel overlays on the seismic's texture, for which color and opacity can be freely chosen (Fig. 35).

Not only the human experts annotate the seismics, but also the results of inference runs of trained models are saved back to the seismic volumes as annotations. Thus, it is possible to e.g. superimpose inferences results and expert annotations to compare the performance of ML models with a human annotation.

Annotation Editor
In addition to the Seismic Viewer, there is also an annotation tool that allows for drawing annotations on slices in the seismic. Comments can be attached to voxels in the volume. This tool is used to create annotations on seismic volumes that indicate where to sample training data from. It can also be used to comment on result annotations or annotations from other experts. The comment function is particularly useful for marking certain places in a volume where other users should look to e.g., where an ML model performed notably (Fig. 36).

Model Visualizer
Model Visualizer is a tool for ML experts to visualize certain aspects of learning runs or trained models. An expert can see from those visualizations if the learning algorithm performed correctly or gain insights into how a trained model's decision process is working. We understand the Model Visualizer as a kind of debugging tool for neural networks. We describe the visualization tool called DeepVis in detail in Sect. 6 (Fig. 37).

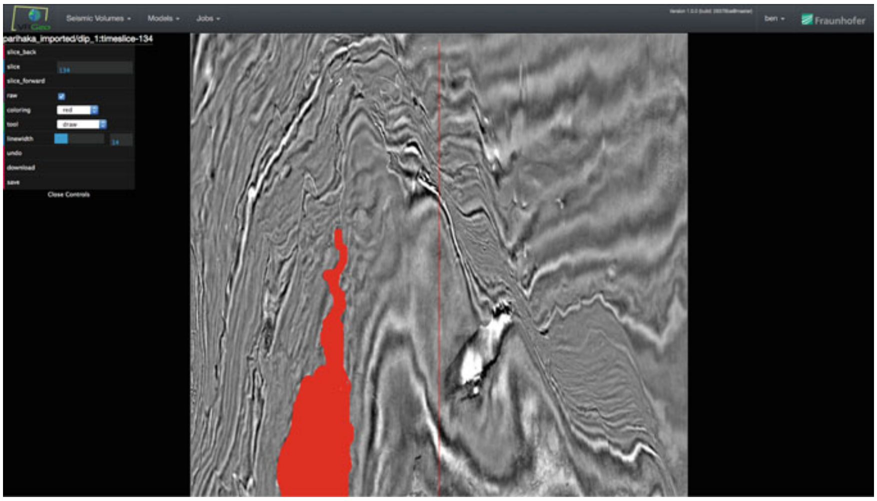


Fig. 36 The annotation editor in the Web Interface showing a time slice from the Parihaka (https://wiki.seg.org/wiki/Parihaka-3D) dataset with a hand-drawn annotation marking a channel system

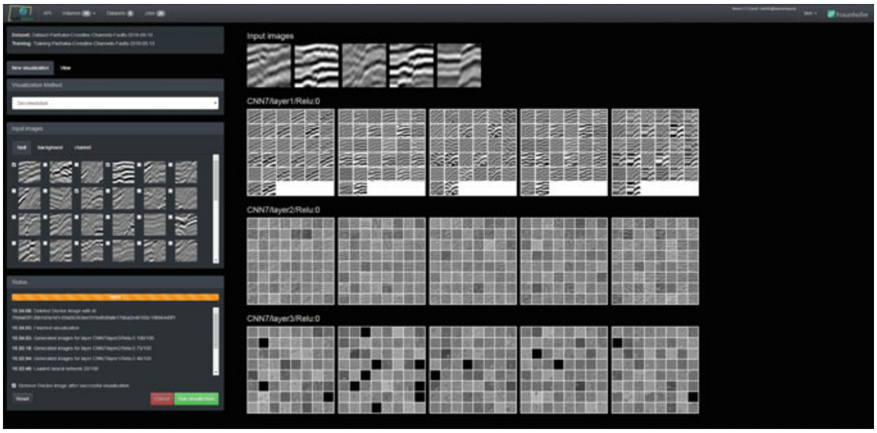
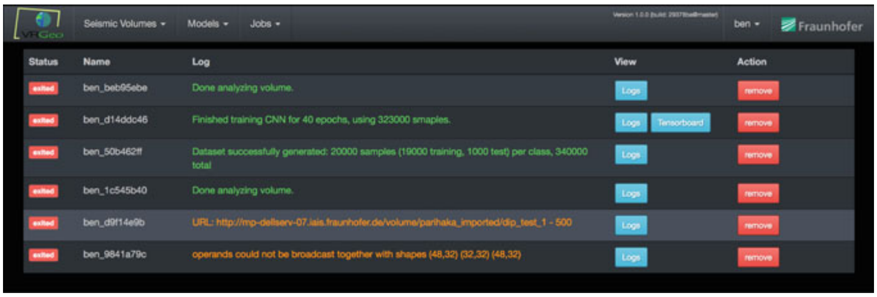


Fig. 37 DeepVis tool in the web interface showing a Deconvolution Analysis of the layers of a Convolutional Neural Network

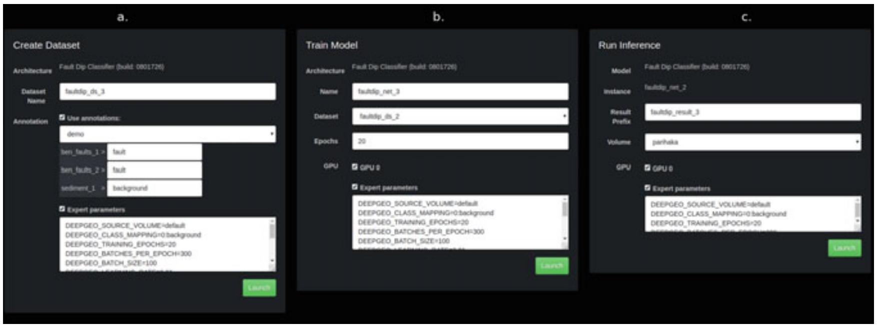
Job Management

Job Management components in DeepGeo are the tools to kick off jobs, monitor operations, and manage the execution environment. It is the place in the interface where the ML experts will do the majority of the work. The interface displays the current status of all jobs in execution, and a user can to access the complete logs of



Status	Name	Log	View	Action
exited	ben_beb95ebe	Done analyzing volume.	Logs	remove
exited	ben_d14ddc46	Finished training CNN for 40 epochs, using 323000 smaples.	Logs Tensorboard	remove
exited	ben_50b462ff	Dataset successfully generated: 20000 samples (19000 training, 1000 test) per class, 340000 total	Logs	remove
exited	ben_1c545b40	Done analyzing volume.	Logs	remove
exited	ben_d9f14e9b	URL: http://mp-dellserv-07.iais.fraunhofer.de/volume/parihaka_imported/dip_test_1 - 500	Logs	remove
exited	ben_9841a79c	operands could not be broadcast together with shapes (48,32) (32,32) (48,32)	Logs	remove

Fig. 38 The Job Management View in the web interface showing two failed jobs (orange) and four succeeded jobs (green). For the training job, there is the option to start an instance of Tensorboard to visualize the training log data



Create Dataset

Architecture	Fault Dip Classifier (build: 0801726)
Dataset Name	faultdip_ds_3
Annotation	☑ Use annotations: demo
	ben_faults_1 > fault
	ben_faults_2 > fault
	sediment_1 > background
	☑ Expert parameters

```
DEEPGEO_SOURCE_VOLUME=default
DEEPGEO_CLASS_MAPPING=0:background
DEEPGEO_TRAINING_EPOCHS=20
DEEPGEO_BATCHES_PER_EPOCH=300
DEEPGEO_BATCH_SIZE=100
```

Train Model

Architecture	Fault Dip Classifier (build: 0801726)
Name	faultdip_net_3
Dataset	faultdip_ds_2
Epochs	20
GPU	☑ GPU 0
	☑ Expert parameters

```
DEEPGEO_SOURCE_VOLUME=default
DEEPGEO_CLASS_MAPPING=0:background
DEEPGEO_TRAINING_EPOCHS=20
DEEPGEO_BATCHES_PER_EPOCH=300
DEEPGEO_BATCH_SIZE=100
```

Run Inference

Model	Fault Dip Classifier (build: 0801726)
Instance	faultdip_net_2
Result Prefix	faultdip_result_3
Volume	parihaka
GPU	☑ GPU 0
	☑ Expert parameters

```
DEEPGEO_SOURCE_VOLUME=default
DEEPGEO_CLASS_MAPPING=0:background
DEEPGEO_TRAINING_EPOCHS=20
```

Fig. 39 Dialogs for launching jobs: **a** Create a new training dataset, **b** Train a machine learning model, **c** Apply a trained machine learning model to a dataset for analysis

a job as well as an instance of Tensorboard,[20] provided the job stores Tensorflow[21] log data in a designated location in the container file system (Fig. 38).

The Job Management tools are also for the seismic experts, as users of the machine learning models. To facilitate the use of ML models by non-expert users, DeepGeo provides convenient dialogs for

- creating new training datasets
- training models
- running inference of trained models against seismic data.

The dialogs initially present only the most important parameters for performing the respective operation and use the principle of progressive disclosure to present expert options [86] (Fig. 39).

[20] https://www.tensorflow.org/guide/summaries_and_tensorboardv.

[21] https://www.tensorflow.org/.

7.7 The DeepGeo Proof-of-Concept

We have shown the organizational and ergonomic aspects and challenges of tying ML R&D into an organization's operations and have shown the current state of our DeepGeo platform for working with machine learning on seismic data. The system started as a collection of tools for ML R&D in the VRGeo Consortium. We got valuable insights into the domain-specific requirements. The virtual collaboration environment that DeepGeo implements aims at closly integrating the work of the ML experts with the work of the seismic interpreters. By automating data management and tracking of operations the system further takes tedious tasks off the researcher's shoulders. The system also facilitates the application of ML models by seismic experts by providing intuitive interfaces that enable non-experts to train and run ML models.

Future functionality will improved viewing and annotating seismic data, model visualization, and interacting with the learning processes. These are projects in the collaboration layer as the collaboration layer is the place to address ergonomic aspects most. For the storage and the execution layers, the challenge is to keep up with the ever-developing High-Performance Computing (HPC) technology and improve interoperability with systems common to the application domain. DeepGeo is an example of a topic of relevance for every field in which machine learning is required to be integrated into. The organizational and ergonomic aspects we have presented in this article can be largely generalized, and DeepGeo can serve as a blueprint of the integration on Machine Learning into other domains.

8 Conclusion, the Status Quo, and Next Steps

Since day 1, the VRGeo Consortium has been about innovation for the oil-and-gas exploration and about finding out whether hypes such as virtual reality, serious gaming, interactive visualization, and artificial intelligence can make it to the day-to-day business of finding oil and gas. In this book chapter, we have shown that interactive visualization and artificial intelligence have a very high potential and can make a difference in the future. Actually, many oil and gas companies are looking into Artificial Intelligence nowadays to find out about their respective perspective.

DeepGeo, our proof-of-concept solution, has made it in the meantime to the premises of VRGeo Consortium members, where it is tested and advanced on a day-to-day basis with proprietary seismic data not available to the public. General findings, feature requests, and requirements make it back to the VRGeo R&D team, inspiring new research and development paths and impacting the future work of the VRGeo Consortium.

References

1. P. Kearey, M. Brooks, I. Hill, *An Introduction to Geophysical Exploration*, vol. 4 (Wiley, New York, 2002)
2. M.T. Taner, Seismic attributes. CSEG Rec. **26**(7), 49–56 (2001)
3. R. Wilkinson, *Speaking Oil and Gas* (BHP Billiton Petroleum, 2005)
4. H. von Hartmann, S. Rilling, M. Bogen, R. Thomas, SEISVIZ3D: Stereoscopic system for the representation of seismic data-Interpretation and Immersion, in *EGU General Assembly Conference Abstracts*, 2015, p. 10641
5. B. Froner, S.J. Purves, J. Lowell, J. Henderson, Perception of visual information: the role of colour in seismic interpretation. First Break **31**(4) (2013)
6. W.W.Y. Chan, A survey on multivariate data visualization. Dep. Comput. Sci. Eng. Hong Kong Univ. Sci. Technol. **8**(6), 1–29 (2006)
7. L. Zhou, C. Hansen, Transfer function design based on user selected samples for intuitive multivariate volume exploration, in *2013 IEEE Pacific Visualization Symposium (PacificVis)* (IEEE, 2013), pp. 73–80
8. M.O. Ward, Multivariate data glyphs: principles and practice, in *Handbook of Data Visualization* (Springer, Berlin, 2008), pp. 179–198
9. A. Inselberg, The plane with parallel coordinates. Vis. Comput. **1**(2), 69–91 (1985)
10. P. Riehmann, J. Opolka, B. Froehlich, The product explorer: decision making with ease, in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2012, pp. 423–432
11. H. Guo, H. Xiao, X. Yuan, Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates, in *2011 IEEE Pacific Visualization Symposium* (IEEE, 2011), pp. 19–26
12. L. Vernengo, E. Trinchero, S. Chopra, *Deciphering Seismic Amplitude Language* (AAPG EXPLORER, January Issue, 2017)
13. S.H. Cha, Comprehensive survey on distance/similarity measures between probability density functions. City **1**(2), 1 (2007)
14. Y. Rubner, C. Tomasi, L.J. Guibas, The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vision **40**(2), 99–121 (2000)
15. D.R. Martin, C.C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Trans. Pattern Anal. Mach. Intell. **26**(5), 530–549 (2004)
16. V. Asha, N.U. Bhajantri, P. Nagabhushan, Automatic detection of defects on periodically patterned textures. J. Intell. Syst. **20**(3), 279–303 (2011)
17. A. Karimov, G. Mistelbauer, T. Auzinger, S. Bruckner, Guided volume editing based on histogram dissimilarity. Comput. Graph. Forum **34**(3), 91–100 (2015)
18. T. Ropinski, S. Oeltze, B. Preim, Survey of glyph-based visualization techniques for spatial multivariate medical data. Comput. Graph. **35**(2), 392–401 (2011)
19. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)
20. D. Ciresan, A. Giusti, L.M. Gambardella, J. Schmidhuber, Deep neural networks segment neuronal membranes in electron microscopy images, in *Advances in Neural Information Processing Systems* (2012), pp. 2843–2851
21. A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, J. Dean, A guide to deep learning in healthcare. Nat. Med. **25**(1), 24 (2019)
22. Y. LeCun, L. Bottou, Y. Bengio, Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324
23. L. Oakden-Rayner, G. Carneiro, T. Bessen, J.C. Nascimento, A.P. Bradley, L.J. Palmer, Precision radiology: predicting longevity using feature engineering and deep learning methods in a radiomics framework. Sci. Rep. **7**(1), 1648 (2017)
24. A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, S. Thrun, Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115 (2017)

25. P. Lakhani, B. Sundaram, Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. Radiology **284**(2), 574–582 (2017)
26. A. Jamaludin, T. Kadir, A. Zisserman, SpineNet: automatically pinpointing classification evidence in spinal MRIs, in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, Cham, 2016), pp. 166–175
27. M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, S. Mougiakakou, Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. IEEE Trans. Med. Imaging **35**(5), 1207–1216 (2016)
28. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
29. C. Zhang, Z. Zhang, Improving multiview face detection with multi-task deep convolutional neural networks, in *IEEE Winter Conference on Applications of Computer Vision* (IEEE, 2014), pp. 1036–1041
30. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
31. R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in *Proceedings of the 25th International Conference on Machine Learning* (ACM, 2008), pp. 160–167
32. T. Kooi, G. Litjens, B. Van Ginneken, A. Gubern-Mérida, C.I. Sánchez, R. Mann, N. Karssemeijer, Large scale deep learning for computer aided detection of mammographic lesions. Med. Image Anal. **35**, 303–312 (2017)
33. V. Gulshan, L. Peng, M. Coram, M.C. Stumpe, D. Wu, A. Narayanaswamy, R. Kim, Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA **316**(22), 2402–2410 (2016)
34. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014). arXiv preprint arXiv:1409.1556
35. V.N. Vapnik, A.J. Chervonenkis, Theory of pattern recognition (1974)
36. A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension. J. ACM (JACM) **36**(4), 929–965 (1989)
37. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization (2014). arXiv preprint arXiv:1412.6980
38. S. Hashem, Optimal linear combinations of neural networks. Neural Netw. **10**(4), 599–614 (1997)
39. U. Naftaly, N. Intrator, D. Horn, Optimal ensemble averaging of neural networks. Netw.: Comput. Neural Syst. **8**(3), 283–296 (1997)
40. L. Perez, J. Wang, The effectiveness of data augmentation in image classification using deep learning (2017). arXiv preprint arXiv:1712.04621
41. S. Aksoy, R.M. Haralick, Feature normalization and likelihood-based similarity measures for image retrieval. Pattern Recogn. Lett. **22**(5), 563–582 (2001)
42. J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2009), pp. 248–255
43. A.U. Waldeland, A.C. Jensen, L.J. Gelius, A.H.S. Solberg, Convolutional neural networks for automated seismic interpretation. Lead. Edge **37**(7), 529–537 (2018)
44. R. Saltzer, Structure and elastic properties of the Earth, in *Handbook of Signal Processing in Acoustics* (Springer, New York, 2008), pp. 1523–1534
45. A.A. Aqrawi, T.H. Boe, S. Barros, Detecting salt domes using a dip guided 3D Sobel seismic attribute, in *SEG Technical Program Expanded Abstracts 2011* (Society of Exploration Geophysicists, 2011), pp. 1014–1018
46. A. Amin, M. Deriche, M.A. Shafiq, Z. Wang, G. AlRegib, Automated salt-dome detection using an attribute ranking framework with a dictionary-based classifier. Interpretation **5**(3), SJ61–SJ79 (2017)

47. A. Berthelot, A.H. Solberg, L.J. Gelius, Texture attributes for detection of salt. J. Appl. Geophys. **88**, 52–69 (2013)
48. Z. Wang, T. Hegazy, Z. Long, G. AlRegib, Noise-robust detection and tracking of salt domes in postmigrated volumes using texture, tensors, and subspace learning. Geophysics **80**(6), WD101–WD116 (2015)
49. Y. Zhang, A.D. Halpert, Enhanced interpreter-aided salt boundary extraction using shape deformation, in *SEG Technical Program Expanded Abstracts 2012* (Society of Exploration Geophysicists, 2012), pp. 1–5
50. J. Haukas, O.R. Ravndal, B.H. Fotland, A. Bounaim, L. Sonneland, Automated salt body extraction from seismic using the level set method, in *74th EAGE Conference and Exhibition incorporating EUROPEC 2012* (European Association of Geoscientists & Engineers, 2012), pp. cp-293
51. H. Di, G. AlRegib, Seismic multi-attribute classification for salt boundary detection-a comparison, in *79th EAGE Conference and Exhibition 2017*, vol. 2017, no. 1 (European Association of Geoscientists & Engineers, 2017), pp. 1–5
52. X. Wu, Methods to compute salt likelihoods and extract salt boundaries from 3D seismic images. Geophysics **81**(6), IM119–IM126 (2016)
53. dGB Earth Sciences B.V. Netherlands offshore f3 block (1987). https://www.opendtect.org/osr/Main/NetherlandsOffshoreF3BlockComplete4GB
54. C.R. Ildstad, P. Bormann, Using 3D segy seismic and 3D convolutional neural networks to map and classify seismic facies (2017). https://github.com/bolgebrygg/MalenoV
55. W. Chu, I. Yang, CNN-based seismic facies classification from 3D seismic data (2018). https://cs230.stanford.edu/projects_spring_2018/reports/8291004.pdf
56. H. Di, Z. Wang, G. AlRegib, Deep convolutional neural networks for seismic salt-body delineation, in *AAPG Annual Convention and Exhibition* (2018)
57. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, Cham, 2015), pp. 234–241
58. T. Zhao, Seismic facies classification using different deep convolutional neural networks, in *SEG Technical Program Expanded Abstracts 2018* (Society of Exploration Geophysicists, 2018), pp. 2046–2050
59. Y. Alaudah, S. Gao, G. AlRegib, Learning to label seismic structures with deconvolution networks and weak labels, in *SEG Technical Program Expanded Abstracts 2018* (Society of Exploration Geophysicists, 2018), pp. 2121–2125
60. Y. Shi, X. Wu, S. Fomel, Automatic salt-body classification using a deep convolutional neural network, in *SEG Technical Program Expanded Abstracts 2018* (Society of Exploration Geophysicists, 2018), pp. 1971–1975
61. X. Wu, S. Fomel, M. Hudec, Fast salt boundary interpretation with optimal path picking. Geophysics **83**(3), O45–O53 (2018)
62. M. Fehler, P.J. Keliher, *SEAM Phase 1: Challenges of Subsalt Imaging in Tertiary Basins, with Emphasis on Deepwater Gulf of Mexico* (Society of Exploration Geophysicists, 2011)
63. Y. Alaudah, P. Michałowicz, M. Alfarraj, G. AlRegib, A machine-learning benchmark for facies classification. Interpretation **7**(3), SE175–SE187 (2019)
64. B.B. Avants, C.L. Epstein, M. Grossman, J.C. Gee, Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain. Med. Image Anal. **12**(1), 26–41 (2008)
65. A.G. Roy, S. Conjeti, N. Navab, C. Wachinger, A.D.N. Initiative, QuickNAT: A fully convolutional network for quick and accurate segmentation of neuroanatomy. Neuroimage **186**, 713–727 (2019)
66. H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1520–1528
67. G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 4700–4708

68. A. Odena, V. Dumoulin, C. Olah, Deconvolution and checkerboard artifacts. Distill **1**(10), e3 (2016)
69. V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)
70. F. Milletari, N. Navab, S.A. Ahmadi, V-net: fully convolutional neural networks for volumetric medical image segmentation, in *2016 Fourth International Conference on 3D Vision (3DV)* (IEEE, 2016), pp. 565–571
71. D. Kingma, J. Ba, Adam: a method for stochastic optimization, in *International Conference on Learning Representations*, vol. 12 (2014)
72. Y. LeCun, Y. Bengio, Convolutional networks for images, speech, and time series. Handb. Brain Theory Neural Netw. **3361**(10), 1995 (1995)
73. O. Abdel-Hamid, A.R. Mohamed, H. Jiang, G. Penn, Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition, in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2012), pp. 4277–4280
74. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks. Adv. Neural. Inf. Process. Syst. **25**, 1097–1105 (2012)
75. S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back, Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Networks **8**(1), 98–113 (1997)
76. O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition (2015)
77. D. Erhan, Y. Bengio, A. Courville, P. Vincent, Visualizing higher-layer features of a deep network. Univ. Montreal **1341**(3), 1 (2009)
78. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
79. J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net (2014). *arXiv preprint* arXiv:1412.6806
80. M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *European Conference on Computer Vision* (Springer, Cham, 2014), pp. 818–833
81. B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2921–2929) (2016)
82. R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: visual explanations from deep networks via gradient-based localization, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 618–626
83. D.C. Cireşan, A. Giusti, L.M. Gambardella, J. Schmidhuber, Mitosis detection in breast cancer histology images with deep neural networks, in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, Berlin, 2013), pp. 411–418
84. D. Krafzig, K. Banke, D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices* (Prentice Hall Professional, 2005)
85. The ReadLock Algorithm, Redis Website: https://redis.io/topics/distlock
86. J. Nielsen, *Progressive Disclosure* (Jakob Nielsen's Alertbox, 2006)

Multimodal Summed Area Tables—A Proof of Concept

Daniel Patel

Abstract This chapter presents a method that returns the bimodal distribution in any box-shaped neighborhood of scalar valued volumetric data, in constant time. This functionality allows for more accurate volume rendering, better transfer functions, edge preserving smoothing, rendering shadows, calculating blur and spectral decomposition, all in real-time.

1 Introduction

A method is presented for approximating the histogram of sample values inside any box-shaped subvolume of a scalar valued volumetric dataset in constant time. A histogram can be approximated by the sum of Gaussian distributions. In this work we use two Gaussian distributions as this gives a good balance between functionality versus time and memory consumption. The following subchapters describe how this solution gives improved results on volume rendering, feature detection, edge preserving smoothing, rendering shadows and calculating depth blur.

1.1 Visualizing Large Volumes

A common way to perform volume rendering is to send a ray from each screen pixel through the volume and summing up the light contribution along the way as the ray passes through volume samples until the ray exits the volume [1]. The pixel is then assigned the accumulated light contributions from each sample, see Fig. 1. Typically, the user can choose the ray step size (distance between the circles in Fig. 1) in the software. Having a step size less than the distance between the samples of the volume data (the squares), ensures all samples are visited along a ray. Due to improved acquisition technology, volumes are getting larger, and for large volumes, a smaller

D. Patel (✉)
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

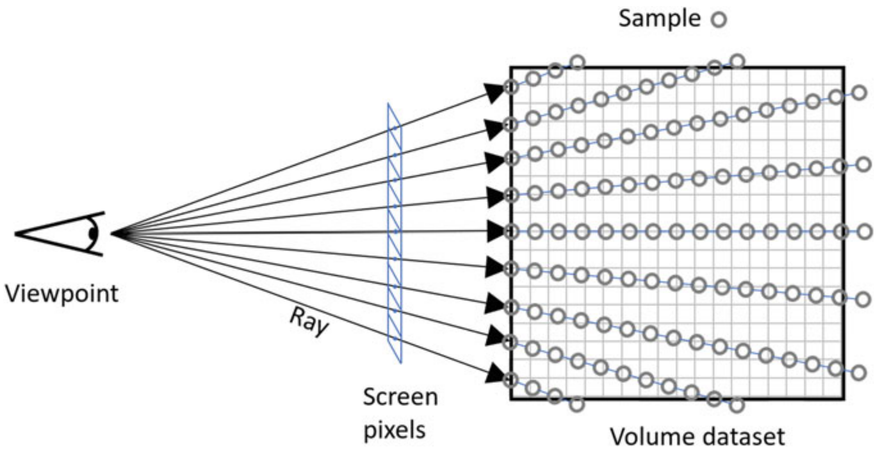


Fig. 1 Ray casting: a common volume-rendering approach. Rays are sent from the center of each pixel relative to the viewpoint and sample the volume at regular intervals until the volume is exited

step size is needed to visit all samples. Also, the relation between screen resolution and the volume resolution affects the degree of missing samples between rays. When the volume resolution is high, gaps between rays will increase. Perspective projection increases this problem further away from the viewer since rays diverge. Solving this by sending multiple rays through each pixel or decreasing step size along rays will reduce the rendering time.

We will now demonstrate how the missing of samples affects the rendering quality by demonstrating the effect on a 2D image.

1.2 Subsampling and Correct Color Representation

Consider the rendering of a single volume slice with size 2000^2 samples into a window of size 500^2 pixels. Using the volume rendering method described above, only every fourth sample in X and Y direction will be visited. The data will be highly undersampled resulting in spatial aliasing artifacts. This is demonstrated in Fig. 2. In (a) is a synthetic dataset (data) of dimension 900×600 shown in grayscale. In (b), the slice is shown after coloring with a transfer function which maps values to colors using a color table. This mapping is represented by the function *col* taking the data as input, shown above the image. The transfer function maps the lowest part of the value range to red, the middle part to green and the remaining high values to blue (a detailed description of the distributions in the data and the color mapping is found in Sect. 3.2). Then in (c) we show the result of representing the 10×10 neighborhood using the center value only (represented by the function $center_{10\times10}$), followed by transfer function coloring, i.e., we are using the color of the center pixel in each 10×10 cell of (b). Each 10×10 neighborhood of pixels is shown with a grid overlay

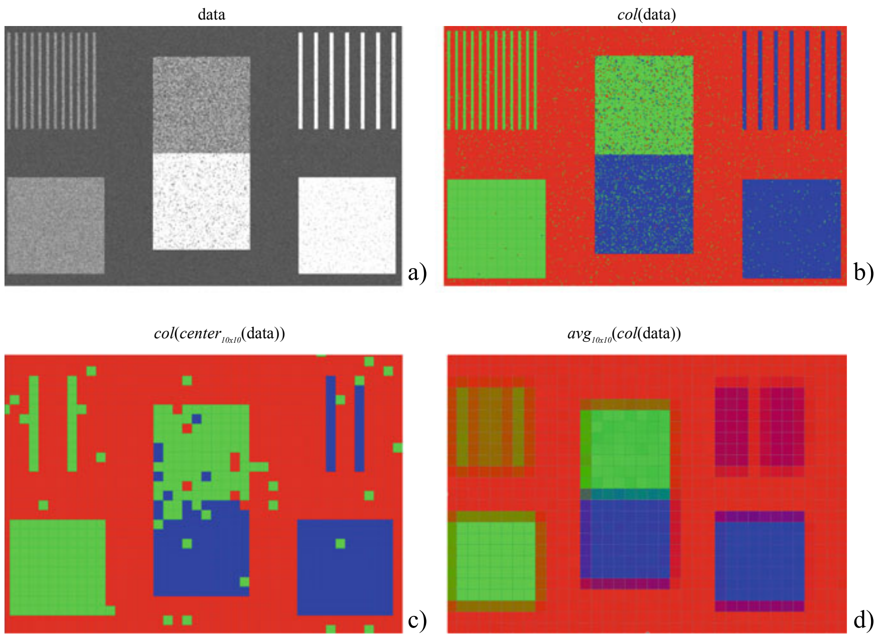
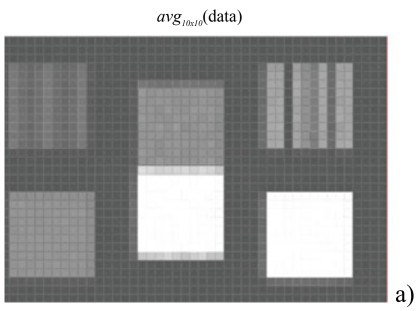


Fig. 2 **a** Original data. **b** Data after application of transfer function. **c** Data after 10× downsampling by using the center value and then applying transfer function. **d** Correct 10 × 10 downsampling by averaging the colors in **b**

in (b), (c) and (d). After a 10 × 10 downsampling, the image resolution has been reduced from 900 × 600 to 90 × 60. This represents how data is sampled in a volume renderer when there are gaps between the rays. For comparison, in Fig. 2d is shown the result of taking the average color of all pixels in the 10 × 10 cells of (b). Taking the average of a 10 × 10 neighborhood is expressed by the function avg_{10x10} shown above the image. This is the most accurate representation of the color if one must represent 10 × 10 colors as one color but is unfortunately slow as all samples must be evaluated (colored) to create the average. We will in the text use "μ", "mean" and "average" interchangeably, also "standard deviation" and "σ" is used interchangeably representing the square root of the variance. A precomputation can be performed but must be redone when the transfer function changes. The downsampled result in (c) suffers from aliasing, and this will be even more evident when interactively scaling, translating or rotating the image, as colors will change abruptly in areas of high frequency such as around the two vertically striped boxes.

Aliasing is a well-known problem in image processing. For images, this can be solved by precomputing the average values of subregions by creating a series of subsampled versions of the image (a so-called mipmap). However due to the nature of volume rendering where the colors are not static, as they are in a photo, but can be changed by the user in run-time, this approach will not work well. This will now be demonstrated. Figure 3a shows the average value of each 10 × 10-grid cell,



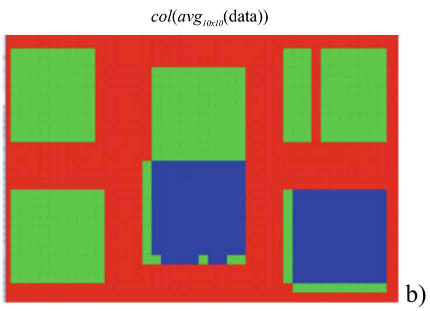


Fig. 3 **a** All samples inside each square have been averaged. **b** The result after applying the transfer function on **a**

representing a correct downsampling of the original image in Fig. 2a. Assigning colors to the downsampled values of Fig. 3a produces Fig. 3b, and as can be seen by comparing with Fig. 2d, it differs significantly in certain areas. The top right square, which originally consists of red and blue pixels, has now become green and the top left square, which is a mix between green and red, has become green. Also edges on the two bottom right squares have turned green. The top right square color has been "averaged" to green because green is exactly between red and blue in the transfer function. This results in a serious misrepresentation of the original data. Methods that render large volumes using multiresolution representations and create coarser resolutions by taking the average values of higher resolutions will also suffer from the same artifacts.

We have now shown that for volume rendering, all the presented downsampling approaches have disadvantages. Iterating over all samples to be downsampling and taking the average of the colors as shown in Fig. 2d is visually accurate but is not practical for interactive volume rendering.

The problem that arises when applying the transfer function after averaging was discussed and addressed by Younesy et al. [2]. They present an elegant solution where they in addition to the mean value, calculate and store the variance. This approximates the histogram of values inside the cell with a Gaussian distribution. We will call the function that maps a 10×10 neighborhood to a Gaussian distribution with a mean and a variance for $unimodal_{10\times 10}$. To go from the Gaussian distribution to a color, one can discretize the distribution into k bins, multiply each bin with the transfer function color for that bin, and then sum all bin colors weighted by the bin size. This requires k multiplications and k lookups in the transfer function. To speed this up Younesy et al. calculate in a preprocessing step, the resulting color for all combinations of mean and variance values and store it in a 2D lookup table. The lookup table must be recalculated when the transfer function is changed, but this is a quick operation. The result of representing the 10×10 neighborhood with a unimodal distribution is shown in Fig. 4a and gives a better result than the previous shown approaches (Figs. 2c and 3b). However, if a downsampled neighborhood contains two "materials", it will have a distribution with two peaks. And this is not

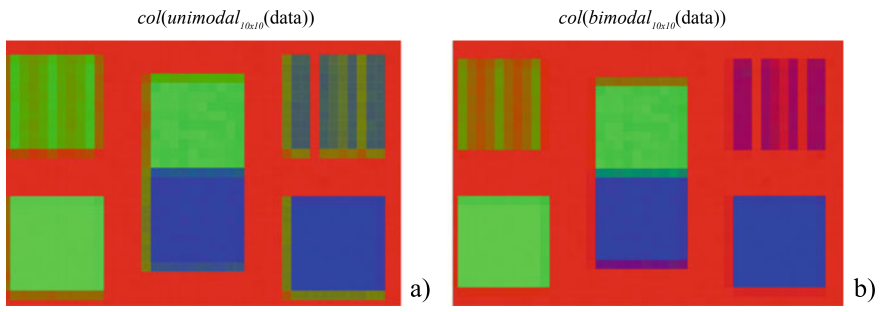

$col(unimodal_{10x10}(\text{data}))$
a)
$col(bimodal_{10x10}(\text{data}))$
b)

Fig. 4 **a** Representing the values in each cell using mean and variance. **b** More accurate colors are achieved when representing the values in each cell with a bimodal Gaussian distribution (using two means and two variances)

well represented by a Gaussian distribution with one peak as can be seen in the upper right box in Fig. 4a which is greenish although no green is present in the original data. This area has a bimodal distribution with one peak around the red and one peak around the blue color, while the fitted Gaussian distribution has its peak at the mean value on green, situated between red and blue in the transfer function.

One can extend this method in a straightforward way by storing an extra pair of (mean, variance) values for each additional peak one wants to be able to describe. Whereas fitting a single Gaussian is simple, more complex computations are required to derive a multimodal fit. One option is to find the Gaussian mixture models (GMM) using the expectation maximization algorithm [3]. This has been performed in Fig. 4b. It shows the result of using two Gaussian distributions to describe the underlying distribution ($bimodal_{10\times10}$) and is close to the ground truth shown in Fig. 2d. In the rest of the document, we will call the process of representing a set of values inside a region (cell) by fewer values that try to capture the properties of the underlying values, for decimation. We have now established that, among the represented down-sampling methods, storing a bimodal distribution when decimating data, gives the most accurate color representation without requiring costly recomputations when changing the transfer function.

1.3 Requirements for Accurate Volume Rendering

To achieve correct pixel colors in a volume rendering, one must evaluate all samples within the cone (frustum) that is projected out behind each pixel as shown in yellow behind one pixel in Fig. 5. Sampling only on the ray, shown with circles, will lead to undersampling and color problems as discussed in Sect. 1.2.

A brute force approach is to send enough rays from different positions on each pixel to cover all samples inside the frustum, then take the average color of each ray and display this as the pixel color, analogous to the approach for producing Fig. 2d.

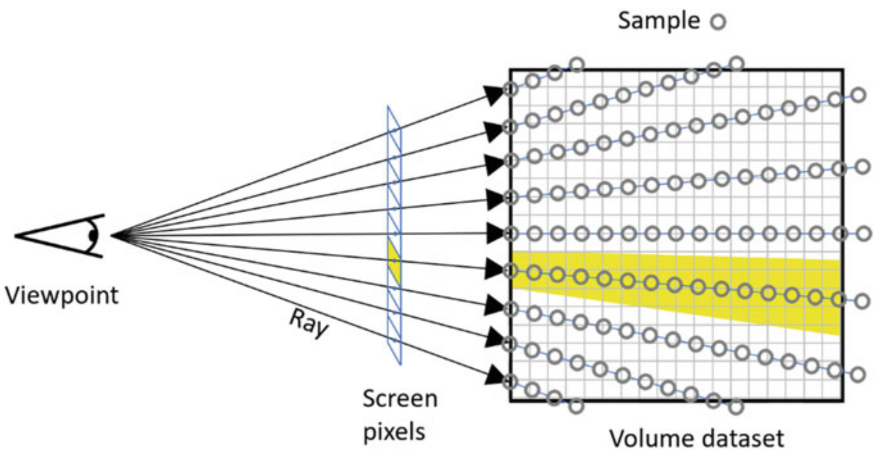


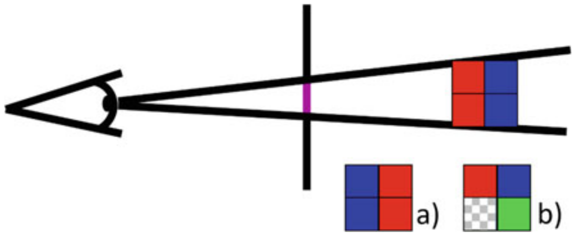
Fig. 5 The color of a pixel (yellow) depends on all samples inside the frustum shown in yellow behind the pixel, and not only along the (sparsely) sampled ray

This method will have a running time linear with the number of samples in a volume and will therefore be slow for large volumes.

A more sophisticated approach is to preprocess the data in some manner so that it is possible to evaluate larger chunks of samples in one go. Then one could divide the frustum into smaller pieces along the ray and accumulate the color from each chunk. The preprocessing stage that for each chunk stores summary data which represents the original values, must do so in an appropriate way as demonstrated in Sect. 1.2.

Even when it is possible to retrieve the correct average color from each chunk, additional considerations must be taken. Consider Fig. 6, that illustrates a frustum behind the purple line segment representing a pixel. The chunk of data contains two red and two blue opaque samples. From the viewpoint, only the red samples would be visible, and the pixel should be colored red. If the chunk was to be represented with the average color, then the color would be purple instead, which is wrong. Case (a) shown in the figure, with the front and back samples swapped, also has average color purple, whereas the correct color is blue. When also having a transparent color as expressed with checkerboard pattern in case (b), one needs to mix the green color

Fig. 6 The ordering of pixel colors within a 4 × 4 chunk affects the final color. Correct color for **a** is blue, while correct color for **b** is the mix between red and green since the color in front of green is transparent



at the back with the red color at the front. I.e., the color inside a chunk is viewpoint dependent.

Publications [2, 4] that perform fast volume rendering by evaluating chunks and attempt to find representative colors for each chunk, still don't consider the issue of the internal ordering of colors within each chunk. To deal with this issue correctly, one must for a chunk, be able to identify if the average color can be used or the chunk must be further split up. This is discussed later in the chapter after introducing how we in constant time retrieve a bimodal distribution approximating the underlying sample value distribution for any box-shaped chunk.

1.4 Related Work on Large Volume Visualization

Solutions for rendering large volumes can be divided in brute-force approaches and algorithmic approaches. Brute-force approaches send enough rays, with a small enough step size so that all data is visited. To achieve interactivity, sufficient compute and memory resources must be available. Different hardware topologies can be used such as clusters of computers, however, volumetric datasets are growing faster in size than the compute power of modern hardware [5].

Algorithmic approaches design algorithms that are able to visit a smaller set of samples while still creating a representative rendering. This can for instance be a wisely chosen subset of all samples, or a decimated dataset of smaller size which still captures the optical characteristics of the underlying samples.

1.4.1 Brute Force Volume Rendering

Examples of the brute force approach is the NVidia Index renderer [6], which samples all data along each ray for large volumes by utilizing several NVidia GPUs. However, data between the rays is not sampled, creating errors of the same sort as shown in Fig. 2c. The Intel Ospray [7] raytracer can handle large volumes. It is built on top of Intel Embree [8]. Both make use of Intels single-instruction-multiple-data (SIMD) CPUs and instruction sets, and target HPC hardware. Thread parallelism can speed up data processing since HPC hardware consists of many CPUs, each with several cores. In addition, data parallelism (SIMD) is used where each instruction can be executed on a vector of data in one cycle. Despite the advantages, data parallelism is more difficult to exploit [8], than thread parallelism as the programmer must make sure that C++ code is compiled correctly to SIMD assembly instructions. HPC hardware allows for extending with more CPUs and also supports a large shared memory compared to the memory available on GPUs.

1.4.2 Algorithmic Volume Rendering

For the algorithmic type of volume renderers, a common and effective method is to divide the volume into equally sized blocks and store a series of iteratively decimated versions of each block. A decimated value is typically created by taking the average of the values of the higher resolution that it covers. This is often called a multiresolution level of detail (LoD) or a hierarchy representation. When having a LoD representation, blocks at appropriate resolutions can be visited for creating a rendering of the volume. For instance, one can upload blocks covering the whole volume, of a coarseness so that they all fit in the memory. The coarser level, the less memory is needed, and the faster is the rendering time as fewer samples are visited, however the penalty is the quality of the rendering. The result is not only the loss of details, but as demonstrated in Fig. 3b, wrong colors can occur. The Tuvok engine [5] uses this approach.

As stated earlier there are better ways to perform decimation than to take the mean value, that capture the underlying data distribution better. Storing the variance in addition to the mean is done in the work by Younesy et al. [2], but this does not represent boundary regions well (see Fig. 4a). Also, this will not work well on seismic data since such data has a high degree of variation in small neighborhoods. An improvement of the work by Younesy et al. with respect to approximating the data distribution, is described by Sicat et al. [4] where time consuming preprocessing step is required. However, this method is not suitable for interactive change of the transfer function.

2 Summed Area Tables for Approximating Histograms

We now show how to use summed area tables [9] for calculating in constant time an approximate histogram of an arbitrarily sized block of volume data. Using summed area tables for fitting a Gaussian distribution to the values in a region has already been done by Olano and Baker [10] for filtering normals on surfaces, and by Donnelly and Lauritzen [11] for calculating soft shadows cast from surfaces occluding light, to surfaces receiving shadow.

2.1 Summed Area Tables Representation

Summed Area Tables was introduced by Crow [9] and also goes under the name Integral Images. Consider the 1D data (data) of 12 values shown below and its’ mean (mean) and standard deviation (sdev) for blocks of length 4:

In the following text we are using Python-like array notation.

data =	[0 6 0 0 1 1 1 1 6 7 6 7]	(the first and third block is marked in gray)
mean =	[1.5 1 6.5]	
sdev =	[2.6 0 0.5]	

Both the data and the mean can be stored in the same array by using a summed area table as shown below, where the value at a position is the sum of all values preceding that position it in the original data:

sum = [0 0 6 6 6 7 8 9 10 16 23 29 36]

With this array, it is possible to calculate the sum of any subrange of numbers by subtracting the last value from the first value in the range. For instance, the three blocks above have sum respectively:

$$[\text{sum}[4] - \text{sum}[0] \quad \text{sum}[8] - \text{sum}[4] \quad \text{sum}[12] - \text{sum}[8]]$$
$$= [6 - 0 \;\; 10 - 6 \;\; 36 - 10] = [6 \;\; 4 \;\; 26]$$

Dividing the sums by their length 4, yields same means as calculated above: [1.5 1 6.5], while using a subrange of length 1 recreates the original data. This method calculates the sum in a constant number of operations independent of the number of values to be summed.

One distribution

To be able to also calculate the variance in constant time, we need to store an additional array, which is the sum of the squared original values. First, the values of the data are squared (data2), then the squares are summed (sum2):

data2 = [0 36 0 0 1 1 1 1 36 49 36 49]

sum2 = [0 0 36 36 36 37 38 39 40 76 125 161 210]

The formula for variance is most commonly presented as the first expression below, but with some manipulation it can be reformulated as the expression to the right which consists of a sum of squared values (the sigma part) and a sum of original values (the μ part).

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_i)^2 = \frac{1}{n}\left(\sum_{i=1}^{n} x_i^2\right) - \mu^2$$

Using the expression to the right, we can calculate the variances of the three blocks. First, we calculate the sigma part:

$$[\text{sum2}[4]-\text{sum2}[0] \quad \text{sum2}[8]-\text{sum2}[4] \quad \text{sum2}[12]-\text{sum2}[8]]$$
$$=[36-0 \;\; 40-36 \;\; 210-40]=[36 \;\; 4 \;\; 170]$$

Then we divide with n = 4 and subtract μ^2: $[36 \;\; 4 \;\; 170]/4-[1.5^2 \;\; 1^2 \;\; 6.5^2] = [6.75 \;\; 0 \;\; 0.25]$.

This constitutes the variance. By taking the square root to get standard deviation, we get the same values as when calculated explicitly: sdev = [2.6 0 0.5].

This method allows us to calculate the mean and variance of any subsequence of numbers in constant time, irrespective of the content and length of the subsequence.

This approach extended to 2D was used to create Fig. 4a. We further improve on the method to produce Fig. 4b which is much closer to the correct solution (Fig. 2d). This is done by retrieving a *pair* of mean and variance values given any subsequence of numbers, which better approximates the distribution of the values in the subsequence. However, if the sequence is best described by a single distribution, then this is detected, and the second mean and variance are not used.

Two distributions

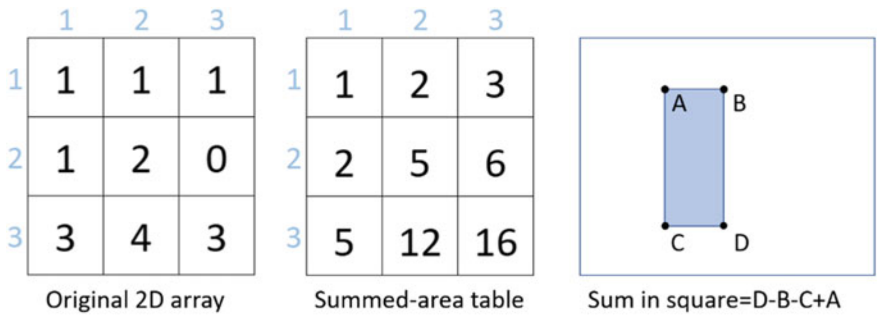
The sequence of numbers is split into two sequences or channels, called Low and High such that in small neighborhoods on either sequence, the variance is minimized. An example of such a sequence is the sequence we used previously: [0 0 1 1 1 1 6 7 6 7] where [0 0 1 1 1 1] and [6 7 6 7] are each well described by a single Gaussian distribution (with mean = 0.5, sdev = 0.5 and mean = 6.5, sdev = 0.5 respectively) and therefore stored in separate channels. The result is shown below. An underscore is shown if the data value has been stored in the other sequence.

$$\begin{aligned}\text{data} &= \left[0\ 6\ 0\ 0\ 1\ 1\ 1\ 1\ 6\ 7\ 6\ 7\right]\\ \text{low} &= \left[0\ \_\ 0\ 0\ 1\ 1\ 1\ 1\ \_\ \_\ \_\ \_\right]\\ \text{high} &= \left[\_\ 6\ \_\ \_\ \_\ \_\ \_\ \_\ 6\ 7\ 6\ 7\right]\end{aligned}$$

Details of how this splitting is calculated is described in Sect. 2.2. By repeating the procedure above for the low and high sequence, and treating underscore as the value 0, we are able to get the mean and variance in constant time of any subsequence of low and of high. However, one step in the calculation is to divide by the number of values in the subsequence (n). With one distribution, we used the length of the sequence, but now we can have empty values (_) that should not be counted. To know the real count we create yet a summed area table (lowcount) of the set values in low:

$$\begin{aligned}\text{lowset} &= \left[1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\right]\\ \text{lowcount} &= \left[0\ 1\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 7\ 7\ 7\ 7\right]\end{aligned}$$

With this sequence we can find the count of low elements in any subsequence and also of high since highcount = sequencelength − lowcount. If lowcount or highcount



$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 1 & 1 & 1 \\ 2 & 1 & 2 & 0 \\ 3 & 3 & 4 & 3 \end{array} \qquad \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 1 & 2 & 3 \\ 2 & 2 & 5 & 6 \\ 3 & 5 & 12 & 16 \end{array}$$

Fig. 7 2D Summed area table calculation

equals to 0, then we know that the sequence is described by one distribution and not two.

A summary of the procedure is as follows: take the original data (data), split it in two sequences (Low and High), calculate the sum and sumsquared for each sequence, resulting in four arrays, and also calculate the lowcount, resulting in five arrays altogether. The memory cost for this is described in Sect. 2.3.

The method can easily be expanded to store more than two distributions. We have therefore named the method Multimodal Summed Area Tables (MSAT).

Extending to 2D and 3D

Extending to 2D is shown in Fig. 7. Instead of taking the difference of two values, we now add or subtract the four corner points of the area we want. For a volume, this extends to the eight corner points of the cube we have in interest.

2.2 *How to Separate Data into Low and High Channel*

Several steps are required to separate the data into the Low and High arrays. To decide if a sample goes to Low or High, all samples within a fixed radius around the sample are investigated. In Fig. 8 left, in the middle of the blue square is an example of a sample to be evaluated. The square represents the evaluation neighborhood. In the right figure, in green is the histogram of the samples. The solid curves in red and blue shows the histogram fitted with two Gaussian distributions. The distributions were fitted using the Python Gaussian Mixture Model algorithm (sklearn.mixture.GaussianMixture) from the scikit-learn package (https://scikit-learn.org/). The sample value (vertical cyan line) itself is checked against the two distributions. If it statistically fits better the distribution with the lowest mean, it is stored in Low; else, it is stored in High. The sample in Fig. 8 is just inside of the gray square. Its value is very high and has a best fit with the rightmost distribution and is stored in the High array.

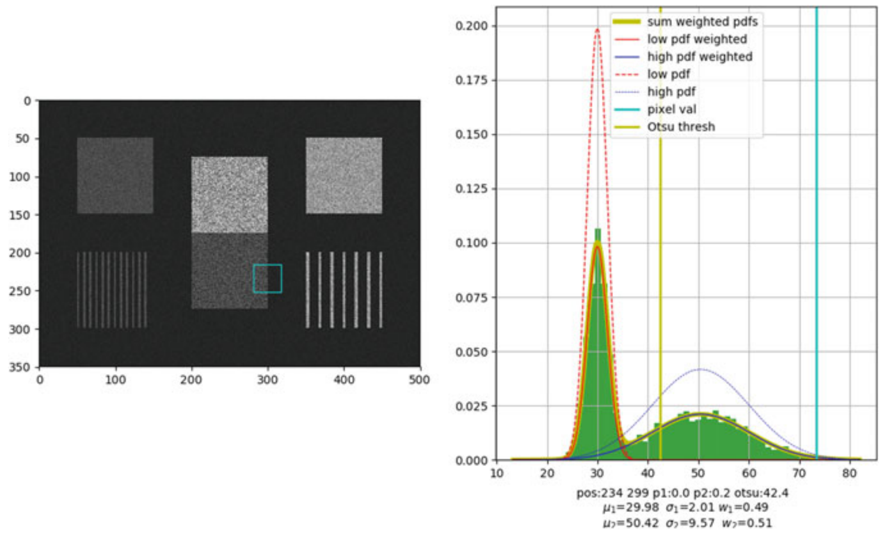


Fig. 8 Left. A fitting neighborhood seen in the blue square and to the right is the histogram of the region (green) and two fitted Gaussian distributions (red and blue solid curves). The vertical blue line is the sample value in the center of the blue square

In a way, this is an adaptive thresholding of the data into a part with low values and a part with high values. However, when inside a homogeneous area best described by a single Gaussian, this method still force-fits two Gaussians and effectively splits half the values of the homogeneous material into low and the other half into high. To avoid splitting areas of a single Gaussian, we extended the method so that it for each sample first tries to fit a single Gaussian, and then two Gaussians. If the single Gaussian reports the best fit, then the sample is stored in an array called *OneDistr*(ibution) (Fig. 9).

Then after all samples have been fitted, the samples in *OneDistr* are segmented into connected components and the values in each component is moved over to either l*ow* or *high* depending on where the values are most similar. This is just a coarse heuristic and not the optimal one. Other ways to move the *OneDistr* values into low or high should be investigated. This process is done only once per dataset

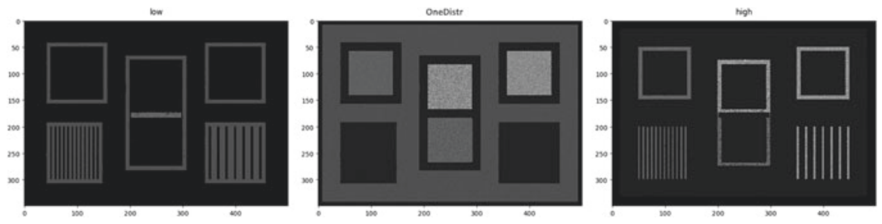


Fig. 9 The low, *OneDistr* and high array

and results in a binary mask of the same size as the dataset. The value in the mask mask is set to 1 if the corresponding value in the data belongs to low, and 0 if it belongs to high. The preprocessing time is a limitation in the current implementation. Sequentially calculating the Gaussian Mixture Model in Python takes about 15 min for a single slice of resolution 1000 $\times$ 1000. It is slow due to the iterative approach of the expectation maximization algorithm. Calculating GMM can be performed per sample in parallel and therefore using an existing or implementing a new GMM that runs on the GPU should reduce the preprocessing time significantly while giving accurate results.

An alternative approach was tried where first a normality test was performed (the Shapiro–Wilk test [12]) to check if the distribution looks like a Gaussian. If not, then we assume there are two distributions and find the separating threshold using the Otsu method [13]. This reduced the time about 25$\times$ but gave a less accurate result. The Otsu method often failed in finding an optimal threshold for separating two distributions. A failure example can be seen in in Fig. 8 right. The yellow vertical line is the threshold calculated by the Otsu method and it is too high, resulting in values from the right distribution to become part of the left one.

The process of separating the values into two channels can be considered a minimization problem, where the goal is to minimize the variation in each channel. The radius that is used should be small, so that it encompasses as few materials as possible, because this makes the fitting easier, but at the same time it must be large enough so that there is enough data to perform a fitting. We have manually set the radius for each dataset, but ideally this should be automatic. It is also possible that varying the radius across a dataset would create even better results. An interesting parallel can be found to the work by Lauritzen and McCool [14]. In their previous work [11] they used one Gaussian distribution for storing distances to light-receiving geometry (the shadow map). This gave good results but produced shadow artifacts in neighborhoods with distance distributions that would be better described by more than one distribution. To alleviate the problem, they split the shadow map into several channels. Different from our work, their work is focused on shadows cast from meshes.

2.3 Memory Consumption

When storing mean and variance, we replace the original data with two arrays, each of same size as the original data. However, the storage requirements are more than twice of the original data as more bits per value is needed. The *sum* array stores sums and ends up reaching high values. This is even more pronounced for the *sum2* array that stores the sum of the squared values. Assume the dataset has bit depth of *b* (for volume datasets this is typically 8, 12 or 16 bits). Which bit depth must then *sum* and *sum2* have to avoid overflow? Given a volume of size n^3, where each sample has maximum value with all bits set (2^b), then the highest value of the *sum* volume would be the sum of n^3 samples each with value 2^b, i.e. $n^3 * 2^b$. The number of bits to express this value is:

$$\mathrm{SumBitDepth} = \log_2\left(n^3 * 2^b\right) = \log_2\left(n^3\right) + \log_2\left(2^b\right) = \boxed{3\log_2(n) + b}$$

The maximum value for the *sum2* dataset would be the square of that of the *sum* array: $(2^b)^2$. The number of bits to express this value is:

$$\mathrm{SumSqrBitDepth} = \log_2\left(n^3 * \left(2^b\right)^2\right) = \log_2\left(n^3\right) + \log_2\left(2^{2b}\right) = \boxed{3\log_2(n) + 2b}$$

Having a 12 bit volume of size 1024^3 results in SumSqrBitDepth = $3\log_2(1024) + 2 * 12 = 30 + 24 = 54$ bits, and SumBitDepth = $3\log_2(1024) + 12 = 42$ bits, and with an 8 bit volume we get SumSqrBitDepth = 46 bits and SumBitDepth = 38 bits.

When storing two distributions, we need twice the space as described above and also an array for storing the count: countBitDepth = $3\log_2(1024) + 1 = 23$. I.e. for an 8 bit volume, we would need $(46 + 38) * 2 + 23 = 191$ bits, which is almost a 24 times increase in storage (191/8 = 23,875).

The original volume sizes are already large and challenging, and a 24-fold increase in size will not help much. Also, it is more cumbersome and slower to represent numbers with a higher number of bits than what is supported by the underlying hardware. Luckily, there exist different methods for reducing the storage requirements of summed area tables. One example is Urschler et al. [15] who divide the sums into blocks and subtract a fitted function from the data before summing.

2.4 Types of Data Supported

As demonstrated in Sect. 1.2, decimating data using a bimodal distribution gives better results than using a unimodal distribution. The synthetic dataset we used, consisted of regions where each region represented a material having values from a Gaussian distribution. If the materials have values that are not Gaussian distributed, or if the intermixing of materials is such that it is hard to separate them from each other using GMM, then our approach will not work well. An example of such an intermixing is if one material contains thin strands of another material. Then, during fitting, the number of samples from the material in the thin strands is not high not enough to detect its distribution.

An example of a data type that does not consist of materials with Gaussian distributions is seismic data. Seismic data is the reflection signal of pressure waves sent into the ground. Consecutive samples along the depth axis represent the signal of a soundwave and is undulating like a sine wave. The probability distribution of a sinus shaped waveform is in fact bimodal with peaks at the minimum and maximum of the wave as this is where the derivative is lowest. When perturbations are introduced to the sine wave, the peaks will smooth out and move towards the center. When the neighborhood is large enough, the peaks will merge into one Gaussian-like peak.

This is also the case for seismic data. Our approach will therefore likely work well for seismic data too. For more information, see Sect. 3.6.

3 Results

3.1 MSAT for More Accurate Volume Rendering

MSAT can be used to quickly evaluate all samples in the frustum behind a pixel. Conceptually, this can be done by traversing view-plane aligned slabs of data within the pixel's frustum from front to back using a user selected step size which defines the slab thickness. The color contribution of each slab must be calculated based on a bimodal or unimodal distribution and added to the accumulated color. With this approach, the step size affects the quality of the result, but even with a coarse step size, all samples in the frustum are evaluated. Error arises when the bimodal distribution reported for a slab deviates from the actual distribution in the slab. In an adaptive rendering mode, this error can be estimated based on the variances returned. If only one distribution is returned, we are in a homogeneous material. If two variances are returned, this indicates that there are at least two materials in the slab. In this case, the slab can be further split up until each piece has a single distribution or until a max number of splits has been performed. The more splits performed, the more accurate the result will be. When two variances are returned and one (or both are) high, then high variance can come from a homogenous material with high variance or it can indicate that the slab contains more than two materials. Then further splitting can be performed. As described in Sect. 1.3, not only the average color within a region, but the ordering of colors within a region affects the resulting color. This approach addresses this problem by adaptively subdividing regions into pieces with similar properties.

As the SAT tables are aligned with the volume borders, the slabs cannot be aligned with an arbitrarily angled viewing plane but must be aligned with one of the three side surfaces of the volume. The volume plane most parallel to the viewing plane should be selected and for a step size of 2, this results in the blue slabs shown in Fig. 10. For an adaptive mode, one can imagine that the yellow slab is first evaluated. If it is not homogeneous, it is further split first into the first blue slab that is then further split into the two gray slabs. In general, one can consider several different subdivision schemes.

3.2 MSAT for Interactive Classification

With the ability to, in constant time, retrieve the one or two Gaussian distributions approximating the values in an arbitrary box-shaped region, we can create a novel

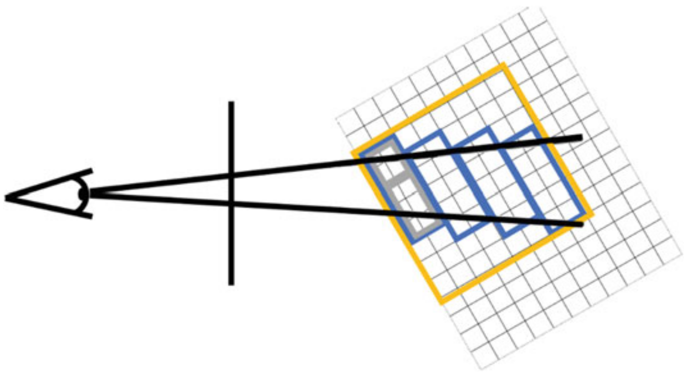


Fig. 10 Slab-based ray casting for the frustum behind a pixel

and powerful transfer function able to detect and segment out features in a volume. Existing 1D and 2D transfer functions plot the histogram or some scatterplot of the data respectively. The user can then detect interesting areas by looking for peaks in the histogram or clusters in the scatterplot and see what this corresponds to in the volume by assigning color to that part of the transfer function. Figure 11b shows the histogram of the synthetic dataset used in this chapter shown in (a). There are three peaks in the histogram at value 30, 50 and 100. Assigning different colors to each of the three visible distributions resulted in Fig. 2b (red color for values 0–33,

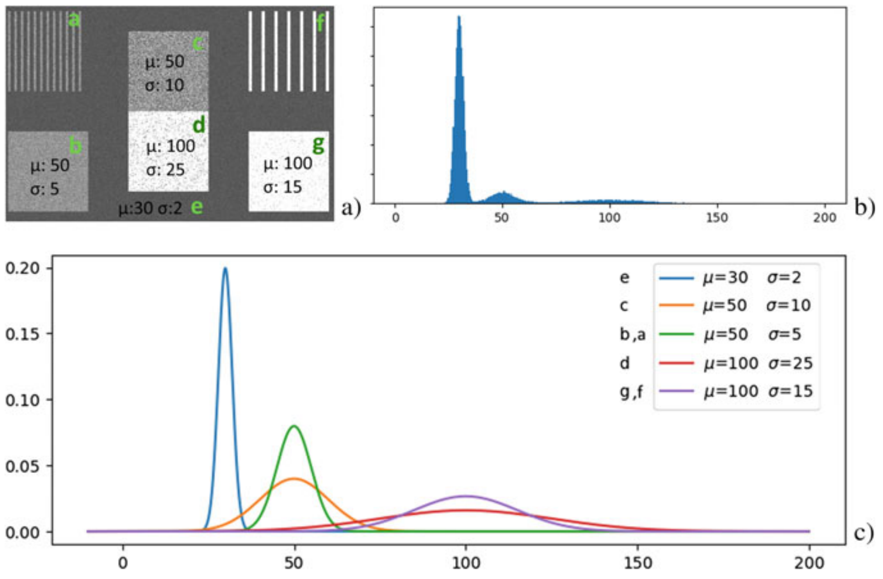


Fig. 11 **a** The materials and their distributions. **b** Histogram of pixel values. **c** Actual material values of the individual materials labelled in **a**

green to values 34–66, and blue to values 66 and higher). The distribution of the values for the regions labelled a–g in Fig. 11a is shown in (a) and (c). For region a and f, the distributions describe the higher valued vertical lines drawn on top of the background values. These background values are from distribution e.

Kniss et al. [16] presented a 2D transfer function where each sample is plotted in a scatterplot with x coordinate equal to the sample value and y coordinate equal to the sample's gradient magnitude. This results in an increase in separability compared to standard 1D transfer functions. It can be used to find boundaries between materials but is not suited to find material interiors and requires each material type to have almost constant interior values to work optimally. A further improvement was presented by Patel et al. [17]. This method can classify and separate materials having distinct internal variance and mean. However, in neighborhoods where more than one material is present, such as on the edges between two materials, the method fails. In addition, the method requires reprocessing of the whole volume when the radius that the mean and variance is calculated for, changes. In contrast, MSAT does not require reprocessing and works for two materials. In the two next sections, we present two new and powerful transfer functions built on the MSAT method.

3.2.1 Mean–Variance Transfer Function

If it would be possible to plot each sample into a scatterplot with mean at the x-axis and standard deviation at the y-axis, where the mean and standard deviation represents the distribution of the material the sample belongs to, then all samples from the synthetic dataset in Fig. 12a would have been clustered around the five colored circles shown in Fig. 12b.

If we calculate a single mean and variance in a box around each sample as was done by Patel et al. [17], except that they used a spherical neighborhood, (and which gives the colors shown in Fig. 4a) then we get the results shown in Fig. 13. For our synthetic dataset, a scatterplot with mean along the x-axis and standard deviation along the y-axis is shown for a square box shaped neighborhood of radius 1, 2 and 5 in Fig. 13a–c respectively. As the radius increases, increasingly focused clusters

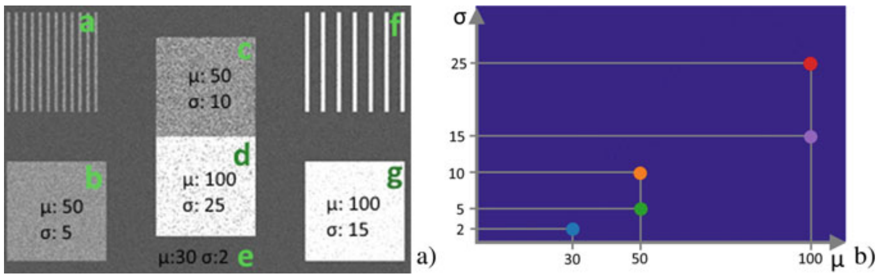


Fig. 12 Right: Conceptual scatterplot of a mean/variance transfer function of the synthetic dataset (left)

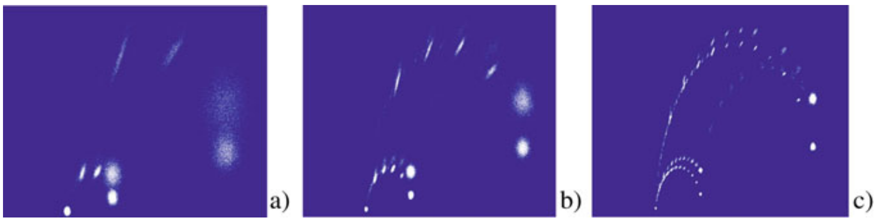


Fig. 13 Scatterplot of the synthetic dataset using one Gaussian distribution for neighborhoods of radius 1, 2 and 5 in **a**, **b** and **c** respectively. Mean is mapped to the x-axis and standard deviation is mapped to the y-axis

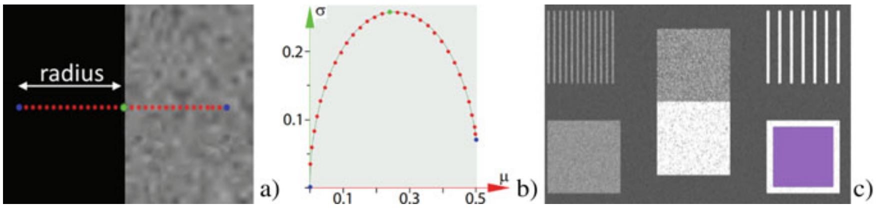


Fig. 14 Boundary regions are not included when using a single Gaussian distribution for classification

show up that correspond to the five circles drawn in Fig. 12b. Points at other positions are also visible, and in Fig. 13c these other points are distributed so that they form arcs between the 5 clusters.

These arcs come from points on the boundary between different materials. Consider the area between two materials as shown in Fig. 14a. If we plot the μ and σ for each of the red dots in (a), we get the values plotted in (b). The leftmost sample in Fig. 14a in blue has a neighborhood of black material only and the (μ, σ) equals that of the black material which is (0, 0). The rightmost sample in blue has a neighborhood containing the gray speckled material only and the (μ, σ) is therefore that of the gray speckled material. When going from left to right along the points, μ increases since less black and more gray material is included in the neighborhood and σ increases since there is more variation in the values. At the center green point, an equal amount of each material is in the neighborhood around the point and σ is at maximum. Going further to the right, σ starts decreasing as the distribution starts becoming homogeneous. For the interested, the analytical formula of the arc-shape of the scatterplot has been derived in [17]. Since the (μ, σ) is wrongly calculated for samples in boundary regions, boundary regions of objects are not captured when using the scatterplot as a transfer function by assigning a color to a cluster representing a material. This is shown in Fig. 14c where the pixels corresponding to the cluster of material g (see Fig. 12) has been colored violet. An uncolored border around the material of thickness equal to the radius is seen. The vertical stripes in region a and f will also be very hard to identify as they are thin. The estimation of

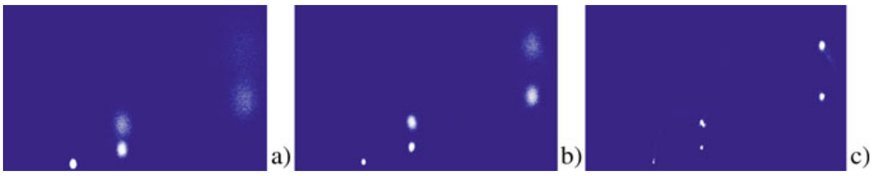


Fig. 15 Scatterplot of the dataset using two Gaussian distributions for radius 1, 2 and 5 in **a**, **b** and **c** respectively. Mean is mapped to the x-axis and standard deviation is mapped to the y-axis

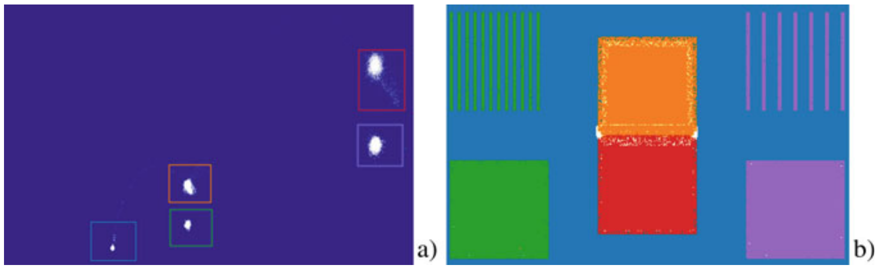


Fig. 16 **a** Mean-variance scatterplot with selection shown as colored rectangles, and in **b** is shown the result. White represents unselected samples. Radius = 4

(μ, σ) for samples in a material which is close to another material is affected by the values from the other material and becomes wrong.

When retrieving two mean and variances instead of one, using MSAT, the two-material boundary problem disappears and the scatterplots for radius 1, 2 and 5 shown in Fig. 15 are much cleaner.

By assigning colors using the scheme of Fig. 12 to each cluster as shown in Fig. 16a, we get the result shown in **b**. Each sample produces two tuples of (μ, σ) values. To find out which of the two distributions the sample most likely belongs to, we calculate the probabilities for the sample value to belong to each of the distributions and assign the sample to the distribution that returns the highest likelihood.

The white points in Fig. 16b represent samples that have not been colored by the selection shown in (a). To better see the points that are outside the selection rectangles we show the scatterplot with high contrast in Fig. 17a. These outside points correspond mostly to the two white spots on the corners of the squares where material c (orange) and d (red) meet. The two white spots arise because there are 3 materials in the neighborhood but the two tuples of (μ, σ) values are not able to describe 3 materials. Fitting two Gaussians to a distribution with three Gaussians will result in a bad fit. An example is shown in Fig. 17b for the neighborhood around the two white spots. Here one Gaussian (blue curve) is fitted correctly to material d, but the other Gaussian (red curve) is fitted to cover both material c and material e and has a mean value just in between the mean of each of the materials.

The wrongly colored pixels and the uncolored white pixels on the inside border around material c (orange) and d (red), are due to overlapping tails of the distribution

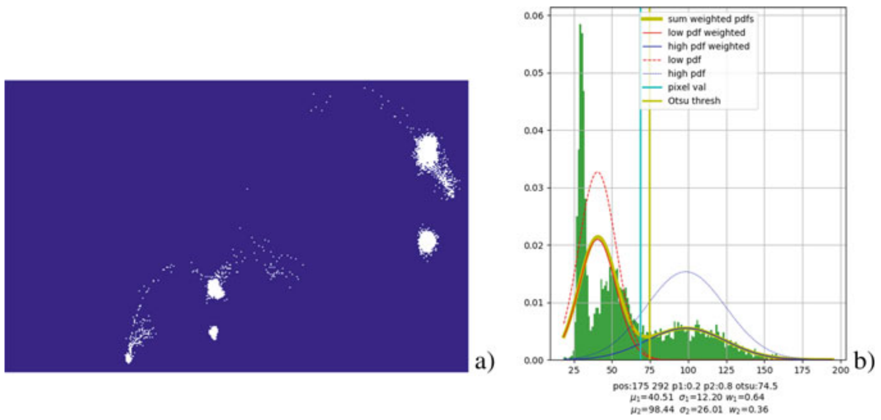


Fig. 17 **a** Scatterplot at maximum contrast shows some points outside the 5 clusters. Some of them arise from areas that have more than 2 distributions in their neighborhood. **b** Shows the histogram and badly fitted bimodal fitting for a neighbourhood containing the 3 materials c, d and e (see Fig. 12a)

of the material in the square with the material outside the square. Figure 8 shows distribution d and e, and Fig. 11 shows all distributions. When separating samples into either Low or High (Sect. 2.2), a hard thresholding is performed. In Fig. 18 left is shown two overlapping distributions in orange and green and their summed distribution in blue. The red stippled vertical line shows the optimal threshold value for separating the distributions. Samples lower than this value have highest probability to belong to the left distribution and are put into the Low array even though they may belong to the tail of the right (green) distribution and vice versa. The hard thresholding puts samples belonging to "the tail" into the wrong channel/array. Therefore inaccuracies show up on the border of materials with a high degree of overlap in their distributions. This inaccuracy might be possible to improve by using more information than the threshold alone. This can be achieved by performing something like a watershed, level set or edge based segmentation and then looking at the histogram inside each segmented part.

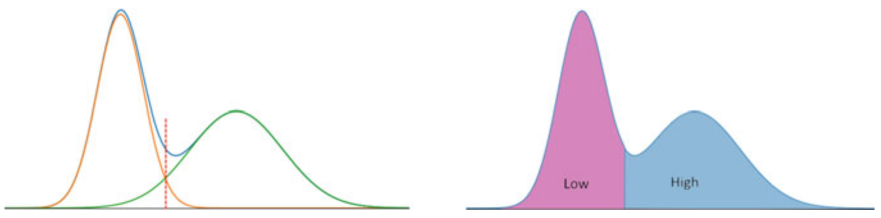


Fig. 18 Left: Two overlapping distributions (orange and green) and their separating threshold (red stippled line). Right: The resulting parts of the distributions stored in the Low and High array

This hard thresholding during preprocessing also results in that the positions of the clusters do not have the exact coordinates as they theoretically should according to Fig. 12. This is because the mean and variance are calculated from the Low and High values. Figure 18 right shows the distribution stored in Low to the left, and in High to the right, for a region containing two materials. Each distribution has lost one of its tails and gained one of the tails from the other distribution. The mean and variances will therefore not be exactly the same as the actual material distributions. For our synthetic dataset, the measured mean positions of the clusters are at 33, 66 and 142 (instead of 30, 50 and 100 respectively), while the standard deviations are at 3, 9, 16, 22 and 35 (instead of 2, 5, 10, 15 and 25 respectively).

3.2.2 Mean-Mean Transfer Function

Another mode of the transfer function is achieved by, given a radius, plotting the two means reported for each sample up against each other. This results in a scatterplot as shown in Fig. 19. The x-axis is the mean of the material that the sample is (probability-wise) in, and the y-axis is the mean of the other distribution. Going back to Fig. 8, the sample in the center of the selection box belongs probability-wise to the right distribution having mean = 50.42, while the left distribution has mean 29.98. This sample would therefore be plotted as a point with x coordinate 50.42 and y coordinate of 29.98. When creating such a scatterplot we see clusters appearing.

By assigning colors to some of the clusters, as shown with colored rectangles in Fig. 20a, we get the coloring shown in (b). To better analyze these results, we have

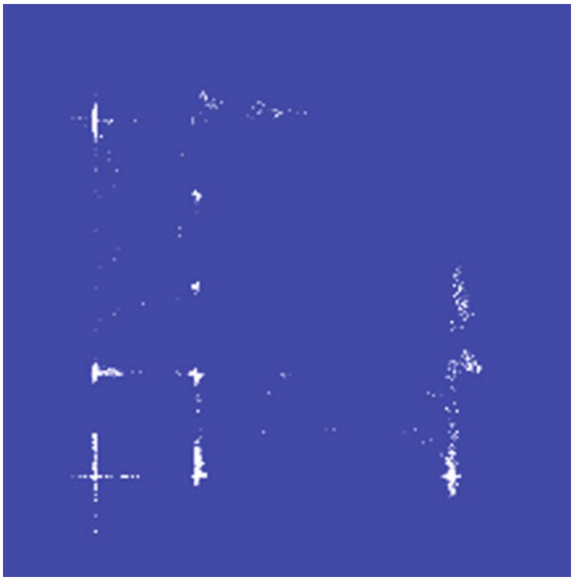


Fig. 19 A mean-mean transfer function for Radius of 5

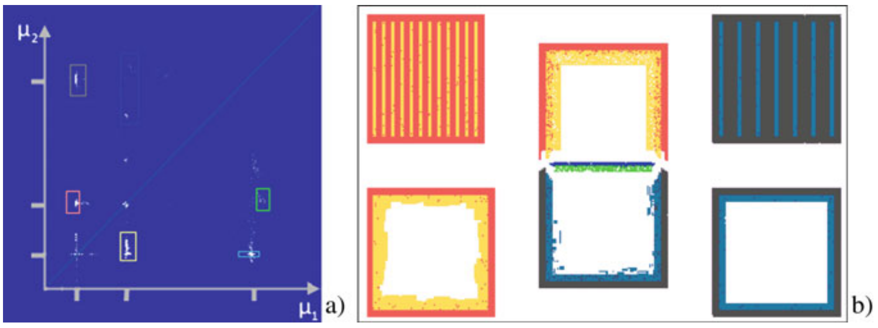


Fig. 20 **a** Mean-mean scatterplot with selection, and in **b** is shown the result. White represents unselected samples. Radius = 4

annotated the scatterplot in (a) with axes, a diagonal line, and tick marks for the positions corresponding to $\mu = 30$, 50 and 100. Consider the yellow rectangle in Fig. 20a. It selects samples that are inside materials with $\mu = 50$, while also having samples with $\mu = 30$ within its radius. Materials a, b and c (see Fig. 11), all have $\mu = 50$. Their interior parts that borders against the background which has $\mu = 30$ gives the yellow areas shown in Fig. 20b. If we look on the opposite side of the diagonal in (a) for samples inside $\mu = 30$ materials which border against $\mu = 50$ materials, we see a cluster which represents the other side of the yellow border and is shown in red in (b). The blue and gray smaller squares on each side of the diagonal in (a) follow the same scheme as described above, but for the values $\mu = 30$ and 100.

Finally, we have colored in green the samples inside $\mu = 100$ that border against $\mu = 50$, and oppositely for bright blue. These borders are only found between square c and d. We have now considered all combinations of different material borders. The other clusters shown in the scatterplot do not represent any significant features. This type of transfer function can be useful e.g. for identifying samples that belong to one material that are embedded inside another material, or to find areas where two specific materials are touching each other.

3.3 *MSAT for Edge Preserving Smoothing*

Edge preserving smoothing is the process of smoothing away noise in an image without removing details. This is achieved b not by smoothing over edges or sharp features. The result is that texture or noise disappears, while the sharpness of the image is kept. An example is shown in Fig. 21. Examples of algorithms performing such smoothing is bilateral filtering [18] and anisotropic diffusion [19]. With our method, after the initial preprocessing has been performed, we can in real-time perform something similar to edge preserving smoothing by displaying the mean value of the pixel according to a user defined radius which can be interactively

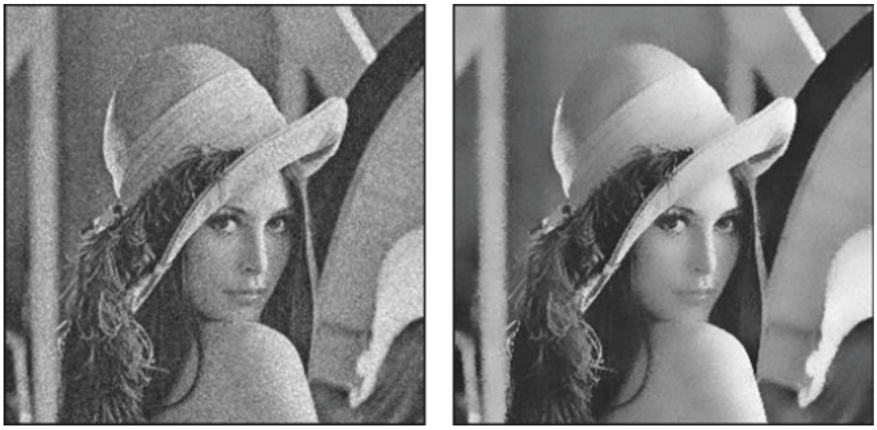


Fig. 21 Left. Original image with noise. Right. Image after edge preserving smoothing with reduced noise

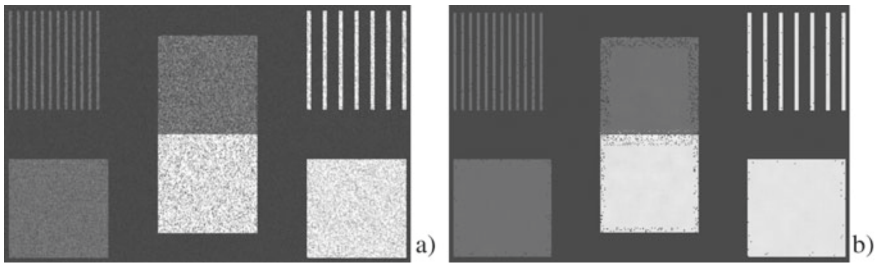


Fig. 22 **a** Our synthetic dataset having materials with internal variance which in this example can be considered as noise. **b** Showing mean value for radius = 5 simulating edge preserving smoothing

set. Radius of 0 results in no smoothing, and higher radius gives higher degree of smoothing. An example using MSAT is given in Fig. 22 for the synthetic dataset and in Fig. 23 for a CT scan of a core sample taken from a drilled well. The method could be performed during raycasting resulting in edge preserving smoothing directly on volume datasets.

3.4 *MSAT for Calculating Shadows in Volumes*

Shading and shadows help us understand the shape and relative spatial position of objects. It also results in more realistic renderings as demonstrated in Fig. 24.

Shading is relatively simple to calculate as it is based on the surface normal on an object, which in volumes can be approximated by the gradient value. On the other hand, the degree of shadow at a sample, i.e. the amount of light arriving on it, is

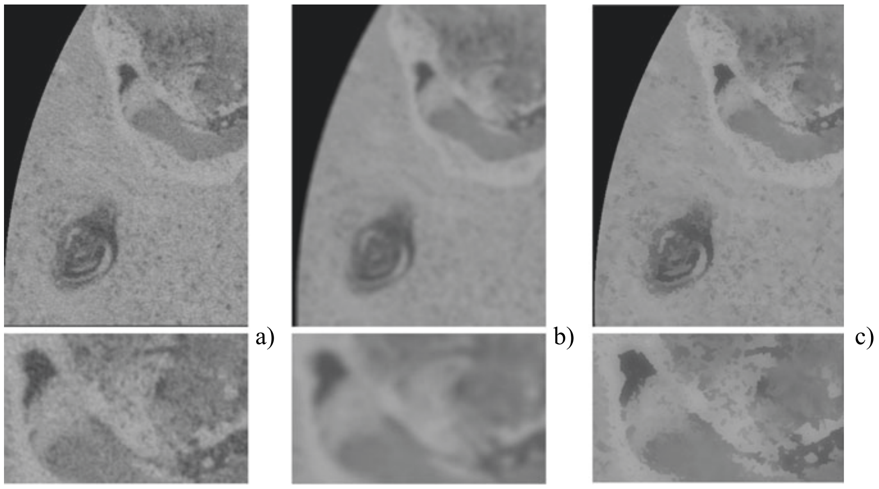


Fig. 23 Top **a**: Slice of a CT scan of a core sample. **b** Smoothing with radius = 4 in all 3 dimensions of the dataset. **c** Showing mean value for radius = 4 in 3D, simulating edge preserving smoothing. Bottom: zoom-in on the slice shown above

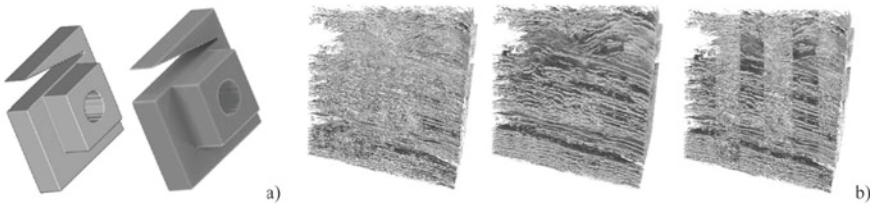


Fig. 24 **a** A geometric object rendered with Phong surface shading to the left and with shadows to the right. **b** A seismic volume rendered with Phong surface shading to the left and with shadows in the middle. The right image shows alternating stripes from the two left renderings for comparison. Phong shading gives a flat appearance and reduces insight into the volume [20]

dependent on the opacity value of any sample that is between the sample and a light source. When the light source is a point, fast methods exist, but this creates hard and nonrealistic shadows. Brute force methods for soft realistic shadows will for each sample in the volume cast rays out in all directions from the sample to see if any rays are hitting a light source and also taking the light attenuation along the ray into account. This is slow, and approximate methods have been suggested that assume that the data is in a uniformly and ambiently lit room. Then, the light arriving at a given sample is calculate by taking the average degree of opacity of all samples within a certain radius around it. This value is then used as the shadow since opaque samples stop light from arriving to the center sample. Summed area tables have been used to calculate this value in many works (e.g. [21–23]). However, in these works, when the transfer function is edited, opacities of samples change, and the SAT of

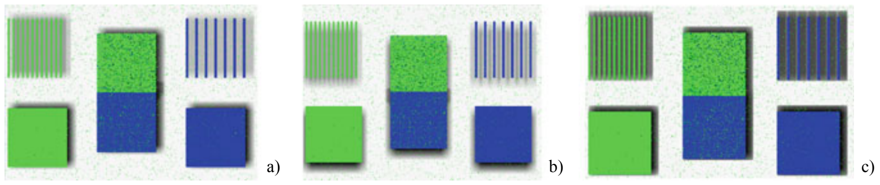


Fig. 25 **a** Light from lower left. **b** Light from top. **c** Left from lower left as in **a**, but calculated using only one distribution

the whole volume must be recalculated. With our method, no SAT recalculation is required as we can estimate the opacity based on the bimodal distribution. Creating volumetric shadows based on a single Gaussian distribution was explored by Shih et al. [24], then recalculation is not required after changing the transfer function. However, the shadows become more inaccurate than with a bimodal distribution as is demonstrated in the next section.

In Fig. 25 we demonstrate in 2D a synthetic setup where a shadow is cast from a slice onto a white background behind the slice. We are using the slice having same values and coloring as Fig. 2b, but with the red set to transparent so that the white background becomes visible. In Fig. 25a we simulate shadows cast onto the white background from the colored slice in front, based on a light source from the bottom left corner. In (b) the light comes from top direction. The way we simulate the light source position is by offsetting the evaluated neighborhood in the opposite direction from where the light source is. In (c) we show again the light from bottom left, but this time the shadow is calculated based on a single distribution instead of two distributions and one can see how much more incorrect the shadows become.

3.5 MSAT for Frequency Filtering of Volumes

MSAT approximates the average color of any box-shaped neighborhood given a transfer function. This results in more accurate volume rendering as described earlier. Another use is to set the viewer's attention to a given distance by gradually blurring out data the further it is from this distance. In [25], blur is used to simulate a camera with a focus plane to set attention to an aneurism in blood vessels, see Fig. 26. Letting the user interactively set the focus plane may improve understanding of depth.

Another use is to perform interactive band-pass filtering of the volume to for instance sharpen edges. This is done by taking the difference of a strong blur and a weak blur. The MSAT method approximates the Difference of Gaussians algorithm except that the filters are top hat shaped (boxes) and not Gaussian shaped. For seismic data one could calculate spectral decomposition in real-time (Fig. 27) at almost no processing cost. Spectral decomposition [26] is an efficient way to visualize channels in seismic data.

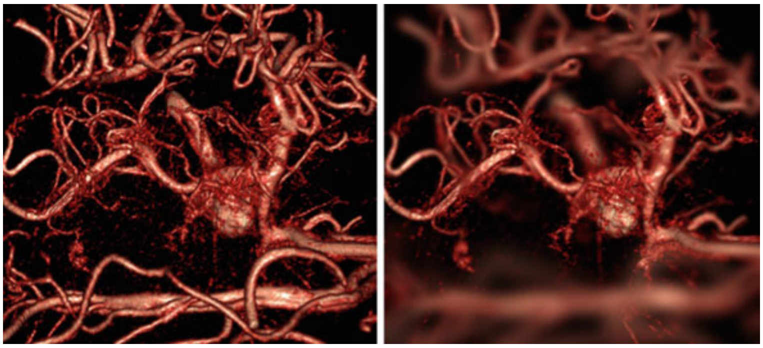


Fig. 26 Left: Volume rendering of blood vessels. Right: The same volume rendering, but with a focus plane for giving focus of attention to a specific depth [25]. This focus plane simulates the depth of field introduced by the lens of the eye, or the lens of a camera

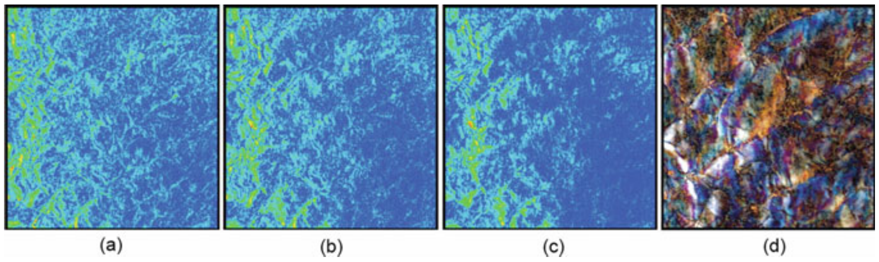


Fig. 27 Combining the response for 24 Hz (**a**), 32 Hz (**b**), and 40 Hz (**c**), of a horizontal slice to the red, green and blue color channel respectively (**d**). From Henderson et al. [26]

3.6 MSAT Adapted for Seismic Data

This section shows that the distribution of values in local neighborhoods of seismic data is typically bimodal and as the neighborhood increases, the two peaks move closer together until the distribution looks one-peaked.

A seismic signal along a horizontal line through the dataset is called a trace. This signal typically wiggles back and forth in a wave shape. Due to this, a decimated voxel, may cover an area with a two-peaked distribution of values. This distribution is very badly approximated by its average value as the value falls between the two peaks. Storing an additional variance value represents the two-peaked distribution as a one-peaked distribution and is also a bad approximation (similar to box f in Fig. 11a).

Distributions of seismic data around small neighborhoods are typically two-peaked. To show why consider Fig. 28a. We model the oscillating nature of a seismic trace (reflection amplitude along depth axis) using a sine function shown at the top. The distribution of values of a sine wave by projecting the samples onto the y-axis

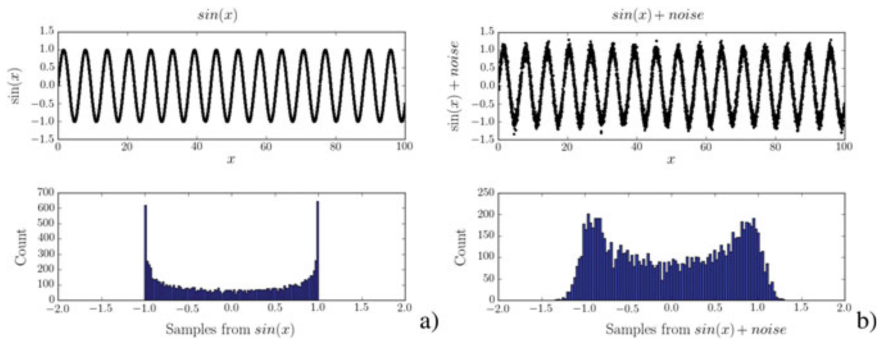


Fig. 28 The distribution of a sine wave without noise (**a**) and with noise (**b**). Figures from [27]

is shown at the bottom. It has two clear peaks at the edges and minimal value in the center. However, a seismic trace is not a perfect sine wave, its maxima and minima vary as it crosses horizons (reflectors) of different strengths. A coarse approximation of this variation can be achieved by adding noise to the sine wave as shown in consider Fig. 28b top. The resulting distribution shown at the bottom has smoother peaks which get closer to each other the more noise is added. This two-peaked distribution can be approximated much better with a bimodal Gaussian distribution (although this is not the actual underlying distribution), than using a single mean value or a single Gaussian distribution. By considering the derivative of the sine function after rotating it 90°, it can be shown that the expression of the distribution of (a) is $P(y) = \frac{1}{2\pi\sqrt{1-y^2}}$ and that of (b) is achieved by convolving the expression with a Gaussian function. For large enough neighbourhoods, the peaks will merge into a one-peaked distribution, which is what you see in practice for the distribution of a full seismic dataset.

The separation of data into Low and High channel should be modified for seismic data as it would probably work better and faster by separating all the values of a trace below 0 (no reflection energy) into Low and the ones above into high, instead of using GMM fitting.

4 Conclusions and Future Work

We have presented a novel method that returns the bimodal distribution of any box-shaped neighborhood of a dataset in constant time. This single functionality opens up for surprisingly many applications. This includes more accurate volume rendering, better transfer functions, edge preserving smoothing, rendering shadows, sharpening edges, calculating blur and performing spectral decomposition, all in real-time.

A volume renderer that evaluates all samples will not be a scalable approach as volume sizes increase faster than computing power. Therefore existing "brute force" volume renderers skip samples and will therefore not represent all the data in the rendering. On the other hand, existing algorithmic volume renderers decimate the data in such a way that the colors do not correctly represent the underlying data. Our method addresses these issues, however currently the limitations are the preprocessing time for separating data into two separate channels, and the storage requirements for the SAT tables. The former problem should be easy to solve as it is parallelizable while the latter one requires more research. Creating a multiresolution representation of the SAT tables using e.g., a mipmap or an octree could be a possible solution to the size problem.

Future work could be to improve on the heuristic for separating samples into the Low and High channel as discussed in the end of Sect. 3.2.1. Generalizing the method to support trimodal and higher modal distributions in a memory efficient manner would be interesting to look at, and to evaluate if it is possible to have an adaptive method that varies the number of modes (the number of Gaussians for representing the histogram) created in the preprocessing step.

References

1. J. Kruger, W. Westermann, Acceleration techniques for GPU-based volume rendering, in *VIS '03: Proceedings of the 14th IEEE Visualization* (2003)
2. H. Younesy, T. Möller, H. Carr, Improving the quality of multi-resolution volume rendering, in *EUROVIS'06* (2006)
3. A. Dempster, N. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. Ser. B (1977)
4. R. Sicat, J. Krüger, T. Möller, M. Hadwiger, Sparse PDF volumes for consistent multi-resolution volume rendering. IEEE Trans. Vis. Comput. Graph. (2014)
5. T. Fogal, J. Kruger, Tuvok, An architecture for large scale volume rendering, in *Proceedings of the Vision, Modeling, and Visualization Workshop* (2010)
6. NVidia, *NVIDIA IndeX Whitepaper,* Berlin (2013)
7. I. Wald, G. Johnson, J. Amstutz, C. Browniee, A. Knoll, J. Jeffers, J. Gunther, P. Navratil, OSPRay-A CPU ray tracing framework for scientific visualization. IEEE Trans. Vis. Comput. Graph. (2016)
8. I. Wald, S. Woop, C. Benthin, G. Johnson, M. Ernst, Embree: a kernel framework for efficient CPU ray tracing. ACM Trans. Graph. (2014)
9. F. Crow, *Summed-Area Tables for Texture Mapping* (Siggraph, 1984)
10. M. Olano, D. Baker, LEAN mapping, in *I3D '10: Proceedings of the Symposium on Interactive 3D Graphics and Games* (2010)
11. W. Donnelly, A. Lauritzen, Variance shadow maps, in *I3D 06. Proceedings of the Symposium on Interactive 3D Graphics and Games* (2006)
12. S. Shapiro, M. Wilk, An analysis of variance test for normality (complete samples). Biometrika **52** (1965)
13. N. Otsu, A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. (1979)
14. A. Lauritzen, M. McCool, Layered variance shadow maps, in *Graphics Interface* (GI) (2008)
15. M. Urschler, A. Bornik, M. Donoser, Memory efficient 3D integral volumes, in *ICCV* (2013)
16. J. Kniss, G. Kindlmann, C. Hansen, Multidimensional transfer functions for interactive volume rendering. Trans. Vis. Comput. Graph. (2002)

17. D. Patel, M. Haidacher, J. Balabanian, E. Gröller, Moment curves, in *IEEE Pacific Visualization* (2009)
18. C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in *Sixth International Conference on Computer Vision*, Bombay (1998)
19. P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, in *Proceedings of IEEE Computer Society Workshop on Computer Vision*, Miami Beach, Florida (1987)
20. D. Patel, S. Bruckner, I. Viola, E. Gröller, Seismic volume visualization for horizon extraction, in *Pacific Visualization* (2010)
21. J. Diaz, P. Vazquez, I. Navazo, F. Duguet, Real-time ambient occlusion and halos with Summed Area Tables. Comput. Graph. (2010)
22. P. Schlegel, M. Makhinya, R. Pajarola, Extinction-based shading and illumination in GPU volume ray-casting. IEEE Trans. Vis. Comput. Graph. (2011)
23. M. Ament, C. Dachsbacher, D. Weiskopf, Low-pass filtered volumetric shadows. IEEE Trans. Vis. Comput. Graph. (2014)
24. M. Shih, S. Rizzi, J. Insley, T. Uram, V. Vishwanath, M. Hereld, M.E. Papka, K.L. Ma, Parallel distributed, GPU-accelerated, advanced lighting calculations for large-scale volume visualization, in *2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV)*
25. M. Schott, A. Grosset, T. Martin, V. Pegoraro, S. Smith, C. Hansen, Depth of field effects for interactive direct volume rendering, in *Eurovis* (2011)
26. J. Henderson, S. Purves, G. Fisher, C. Leppard, Delineation of geological elements from RGB color blending of seismic attribute volumes (Leading Edge, 2008)
27. StackExchange 2014. https://stats.stackexchange.com/questions/126273/probability-distribution-for-a-noisy-sine-wave. Accessed Mar 2021

Visualization of Large Scale Reservoir Models

Loïc Ceballos, Bruno Conche, Guilhem Dupuy, and Daniel Patel

Abstract In this chapter we introduce a method for rendering large reservoir grids that do not fit into GPU memory. We use a multiresolution method where the reservoir is represented in multiple increasingly decimated versions. We decimate shapes of the cells in the reservoir such that important features are preserved, and we decimate property values in a manner that makes it possible to detect when finer versions of the reservoir must be used for achieving correct visualization of the cell values.

1 Introduction

For more accurate appraisal of potential oil fields, or for improved extraction in production fields, the flow of oil, gas and water in the subsurface is simulated to better predict reservoir content and optimal extraction strategy. A simulation models the subsurface by coarsely subdividing it into parts, or cells. The simulation then calculates for discrete timesteps, scalar properties such as temperature, pressure, and content of oil, gas and water, and vector properties such as flow direction, for all the cells.

It is common to use hexahedral cells in reservoir grids and to represent these with a cornerpoint datastructure, which will be defined later. Using hexahedral cells have

L. Ceballos: Work was done when affiliated with INT.

L. Ceballos
INT, Pau, France

B. Conche · G. Dupuy
TotalEnergies, Pau, France
e-mail: bruno.conche@total.com

G. Dupuy
e-mail: guilhem.dupuy@total.com

D. Patel (✉)
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_6

several advantages. They can be aligned with subsurface features such as (vertical-like) layers and (horizontal-like) faults. Compared to unstructured grids, these reservoir grids have a local structure in that they possess a logical i, j, k ordering locally. This results in locally structured linear systems which are easier to solve during simulation, than fully unstructured systems [1]. What is particular about reservoir grids is that they contain geometric discontinuities along faults. Also, the data in the cells exhibit value discontinuities across the faults and across layers. It is important to correctly display these geometric and value discontinuities during rendering of a reservoir.

With the evolution of simulation accuracy, finer and finer subdivisions of the reservoir are being produced. This requires more computing power to visualize the simulation results. Therefore, fine scale reservoir models are often too large to be visualized on regular workstations with current solutions. Consequently, coarser models are used as a compromise to allow visualization. This can lead to inaccuracies and, ultimately, misinterpretation of the reservoir dynamics.

Total has been a world leading oil and gas company and is a major integrated player in low-carbon energies. Total's Exploration and Production group needed a solution that would allow them to visualize large 3D geological reservoir models with more than one billion cells while maintaining the resolution necessary to accurately predict hydrocarbon recovery. Total expressed a wish for visualizing large reservoirs on regular workstations and laptops. To achieve this, we create and use a multiresolution data structure. Lower resolutions naturally will have less details. Therefore, downsampling of geometry has been designed to preserve discontinuities, which are important in reservoirs. Downsampling of attribute data is done by storing extra metadata that gives hints on the representativeness of the downsampled data so that it is possible to detect if higher resolution data should be loaded.

The focus of our rendering is to be able to read out quantitative information on the values of the cells. Therefore we draw the exterior surface of each cell with a color according to an attribute and a color table selected by the user. We also render a wireframe outline of each cell to differentiate neighboring cells of same color. In addition, it is possible to toggle on or off the visibility of cells based on a user defined interval for a specific attribute value. As cells have an (i, j, k) index according to an internal parameterization of the reservoir, it is also possible to select intervals on the indices for showing parts of the reservoir.

2 Related Work

Visualizing unstructured grids. All unstructured cell shapes can be represented using tetrahedra as they are the smallest volumetric building blocks. Therefore, a vast literature exists on rendering tetrahedral meshes. For an overview of early research, see the survey by Silva et al. [2]. For rendering tetrahedral meshes, one solution is to rasterize the tetrahedra one by one, ordered from back to front relative to the viewpoint. Such approaches are called object-order approaches. Much research revolved around how to perform the back-to-front ordering of the tetrahedra fast. A different approach was suggested by Garrity [3] which stores in a preprocessing step the neighbor connectivity information for each tetrahedron. This made it possible to trace rays through the mesh since one could, when exiting the ray through the surface of a tetrahedron, look up which tetrahedron the ray would enter. Methods building on this approach are called image-order approaches. A GPU implementation was presented by Weiler et al. [4]. While the problem of object-order approaches is the time-consuming object ordering, the disadvantage of image-order approaches is the preprocessing and the extra memory required to store the neighbor connectivity. Muigg et al. [5, 6] present interactive ray-casting of unstructured grids composed of cells that are not constrained to consist of a single cell type. Grids can mix cells from a fixed set of cell types such as tetrahedra, hexahedra, octahedra, and prisms [5], or can comprise a mixture of completely general non-convex polyhedral cells [6].

Compared to these methods, ours is tailored for out-of-core rendering of hexahedral meshes with focus on maintaining geometric and value discontinuities.

Visualizing geoscientific reservoir grids. Early work on visualizing unstructured 3D grids from the oil and gas domain was presented by Giertsen [7]. He used a method that samples the data using a plane that sweeps over the spatial domain which subdivides the grid into planar units that can be rendered and aggregated into a 3D volume. The PhD thesis by Frank [8] also addresses unstructured grids in the context of reservoirs and earth models.

In addition to our approach, two other multiresolution methods for reservoir data with hexahedra cells that honor geometric discontinuities from faults, have recently been published [9, 10]. Hexashrink [9] uses Wavelet transforms for downscaling and creating the multiresolution representation. Their representation is general and meant for storage and transfer of data as well as visualization, but it is not optimized for visualization as ours is. For instance, they do not present methods for identifying and combining different resolutions for the purpose of accurate rendering of the reservoir based on available GPU memory and the viewpoint, as we do. Abraham and Celes [10] present a rendering solution using multiresolution reservoirs. Downsampling of geometry is done based on a target surface such as the outer boundary surface of the whole reservoir. Since the boundary geometry is static, they can project attribute values on it by using 2D texture mapping. However, this method does not support visualization of arbitrary interior cells based on their cell values, as ours do.

The following methods also visualize reservoirs, but either use clusters of computers or require that the full reservoir fits in GPU memory. Abraham and Celes [11] present a distributed solution for visualization of large reservoir grids by employing a cluster of computers. Each computer renders a part of the reservoir, and the renderings are combined and visualized on the client computer. Franceschin et al. [12] present a method for rendering reservoir attribute values on arbitrarily shaped surfaces that cut through the reservoir. They utilize regularities in the way reservoir grids are represented to compactly store them on the GPU. A fragment shader is used which, for each pixel, locates the reservoir cell that the corresponding surface position resides in and colors it accordingly. That work is extended by Campagnolo and Celes in [13] for performing volume rendering on reservoir grids.

3 Representation of Reservoir Grids

To digitize the physical properties existing at a specific time in the subsurface space where a hydrocarbon reservoir resides, the space is subdivided into smaller parts called cells. For each cell, a single value representing each physical property of interest is stored. A property, such as temperature will in reality vary at different locations inside a cell, but is approximated with the average value for the whole cell. To produce a model that best represents reality, the reservoir must be subdivided such that this average value is a good representative for the actual varying physical values in that cell. This is the case if the actual values do not have a too high variation around the average value. To achieve such a subdivision, one can either use small cells, or carefully shape the cells so that it is more likely that they cover areas where the properties vary little. As properties usually change less inside identical subsurface materials than across materials, the cell shapes can be made to align with the materials. Material types often change across layers and faults. The subsurface typically has horizontal-like layers of materials, while faults are vertical-like discontinuities where the material on one side has moved relative to the other side. It is thus not ideal to have a cell that crosses layers or faults. This can be avoided by shaping the cells such that their sides are aligned with layer and fault borders and do not cross them. There exist several data structures for expressing spatial subdivisions. One example is a regular grid data structure that subdivides space into 3D rectangles (cuboids). Cuboid sides can only be aligned with horizontal or vertical surfaces and not with arbitrarily angled layer and fault surfaces, and is therefore not well suited for describing subsurface structures. A special datastructure, called the cornerpoint grid has been developed for representing subsurface grids. Cornerpoint grids have the ability to align the cell shapes with layers and faults. The cells are general hexahedra (see Fig. 1 left) and not limited to cubes as for the regular grid.

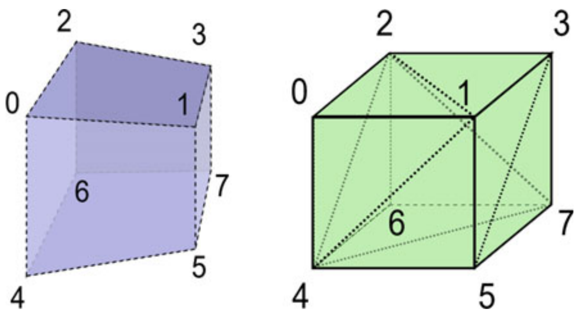


Fig. 1 Left: A hexahedral cell. Right: In our approach we represent each face with two triangles

Each cell is defined by eight 3D cornerpoints as shown in Fig. 1. The 3D coordinates are defined by specifying a line in 3D, called a pillar, and a z-coordinate. The 3D position on the line having this z coordinate defines a cornerpoint. Several cornerpoints will use the same pillar. A pillar is defined by specifying the coordinates of two 3D points that it intersects.

In Fig. 2b is shown a cornerpoint grid in 2D for simplicity. It has 4 green cells and 3 blue pillars. Each pillar is defined by two crosses as shown in the figure. Along each pillar are marked points. Point 1 and 2 on pillar 1 together with point 1 and 2 on the left side of pillar 2 define a cell. The cell above is similarly defined by points 3 and 4 on pillar 1 and 2. Two cells are also shown between pillars 2 and 3. Figure 2a demonstrates this datastructure allows for aligning cells next to each other for modelling layers of distinct properties and for terminating a layer (red layer). The red layer is pinching out between pillar 2 and 3 by having identical corners at left side of pillar 3 (index 2). Between pillar 3 and pillar 4, the red layer continues in a pinched-out manner being completely flat and having zero extent. Continuing the layer using flat cells ensures that the depth-wise indexing of cells is consistent with the layer that the cell is in, i.e., in Fig. 2a the cells in the green layer are all cell number 1 counted from bottom, the red cells are all cell number two counted from bottom, and the blue cells are all cell number 3 counted from the bottom. Figure 2b shows that gaps between cells in the height direction is possible, and that cells that are neighbors along the x axis do not need to share faces. This latter property makes it possible to model faults.

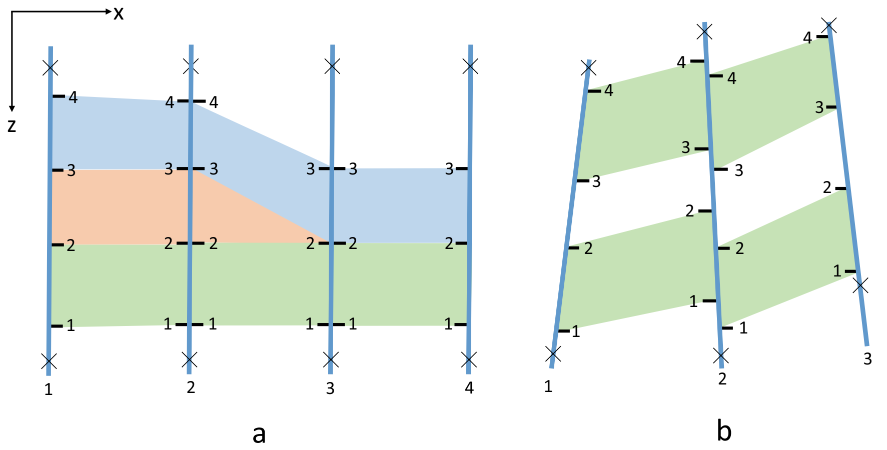


Fig. 2 2D cornerpoint grids seen from the side. The z-axis represents depth. The blue lines are pillars. In 3D, there would also be a *y*-axis moving into the picture. **a** Three layers turning into two layers due to a pinch-out of the red layer. **b** An artificial example of cell discontinuities and gaps that can be expressed with the datastructure

In memory, the grid cells are ordered with the x-axis index (i-direction) cycling fastest, then the y- and z-axis indices (j- and k-direction). All cell-wise property data follows this ordering.

For more information about cornerpoint grids, see Chap. 3 in the book [14] by Lie which is available through open access.

4 Rendering of Reservoir Grids

There are many approaches to render a reservoir grid. A simple rendering approach is to draw the boundary surface of each cell in the reservoir. Graphics processing units (GPU) will automatically handle correct visibility ordering as long as the cells are nontransparent. To support semitransparency, the cells must be rendered back-to-front. A cell has 6 faces, each of which can be rendered using two triangles (Fig. 1 right). The grid can thus be rendered by creating the 12 triangles for each cell, uploading the triangles and the cell properties that have been calculated by the simulator to the GPU, and render all triangles. The triangles can be colored according to a selected simulation property, and selective rendering of cells with specific properties can be performed.

4.1 Rendering Reservoirs that Do not Fit into GPU Memory

The simple rendering approach just described assumes that the GPU has enough memory to store all the reservoir data. Due to the increasing size of grids handled by simulators, it is likely that there will always exist grids that are too large to fit in graphics memory of the latest graphics hardware.

The approach can naively be extended to support large geometries by rendering the reservoir over several passes. Each pass transfers a part of the reservoir from CPU memory to the GPU memory to fill the GPU up completely and renders that part to the screen by correctly combining it with what has been rendered in a previous pass. Data must also be loaded from disk to CPU memory in the case that the reservoir also does not fit CPU memory. This is repeated until all parts of the reservoir have been rendered. This will naturally be slower than rendering a reservoir that fits the GPU memory, because the transfer of data from (disk to) CPU to GPU takes time and must be performed several times per frame (image), whereas when the geometry fits on the GPU there will be no data transfer per frame.

A faster approach is to be able to render the whole reservoir even when it does not fit on the GPU without requiring several rendering passes. This is achieved by storing the whole reservoir in several versions, each with different memory size and thereby different accuracy. The original reservoir which has maximum accuracy is used as base for deriving lower resolution versions by performing downsampling. A downsampled version is created by grouping together neighboring cells of the

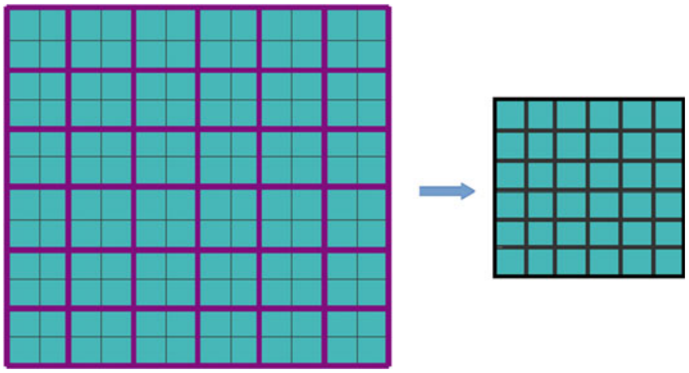


Fig. 3 Downsampling of 2 × 2 cells (left) into single cells (right) for a 2D example. In 3D, 2 × 2 × 2 cells will be downsampled to single cells

higher resolution grid and approximating their boundaries with a single cell. A 2D schematic is shown in Fig. 3 where a grid of resolution 12 × 12 (left) is downsampled to a coarser grid of resolution 6 × 6 (right) by combining 2 × 2 cells into single cells. The downsampled version is again downsampled using the same procedure. The original base reservoir version is called level 0 and downsampling it creates a level 1 reservoir which is again downsampled to level 2 and so on. This makes it possible to use the lowest level that fits into the GPU memory, or to use an even coarser version if one wants to fulfill a high framerate requirement.

Instead of storing each reservoir level monolithically in such a way that a level must be loaded as a whole to the GPU before it can be rendered, each level is subdivided into smaller blocks so that parts of a level can be loaded. This makes it possible to upload and combine parts from different levels for e.g., using low-resolution blocks to render data far away from the viewpoint where cells will be small due to perspective, and high-resolution blocks to render data close to the viewpoint where more details are visible. Each level is subdivided into blocks which have a predefined identical size. This size is equal on all levels. The block size is chosen so that data transfer is efficient as it is faster to transfer data in large chunks. The actual size must be decided by performing tests on the relevant hardware that the software is to be run on. Optimal block size can be different when transferring from disk to CPU compared to transferring between CPU to GPU, in this case an overall compromise must be taken.

An artificial small example is shown in Fig. 4. With a block size of 6 × 6 × 5, a reservoir of size 12 × 12 × 10 cells is perfectly divided into four blocks. Usually, the reservoir dimensions are not evenly divisible by the block size. In this case, blocks at the end along any of the three axes in the reservoir will contain the remaining cells and thus have a smaller size than the predefined size.

The reservoir with all blocks and levels can conceptually be represented as a tree structure as shown in Fig. 5, where each block is shown as a blue node. The root node represents the whole reservoir downsampled so it fits a single block. Each node has

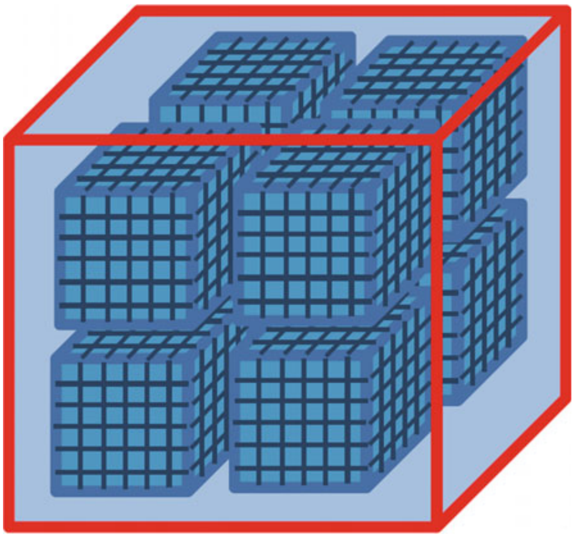


Fig. 4 Subdivision of a resolution level of size 12 × 12 × 10 cells into blocks of size 6 × 6 × 5, resulting in 8 blocks

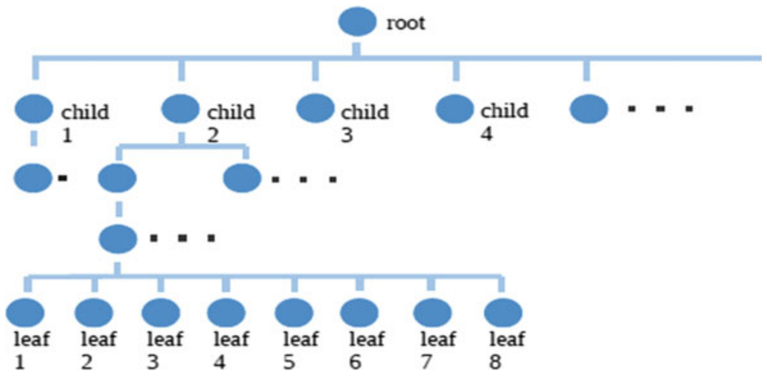


Fig. 5 Octree datastructure

eight child nodes which are displayed on the level below. They represent the nodes it was created/downsampled from (except from nodes containing faults, which are created directly from the level 0 cells, this is described in detail later). Thus, nodes of same reservoir level are positioned in the same level in the tree, and the original finest resolution level (level 0) blocks are shown at the bottom as leaf nodes. In practice, this tree is constructed bottom up based on the original reservoir. This spatial data structure is called an octree [15] since each node has eight children. The octree is used to quickly identify the appropriate blocks at the appropriate resolution during rendering.

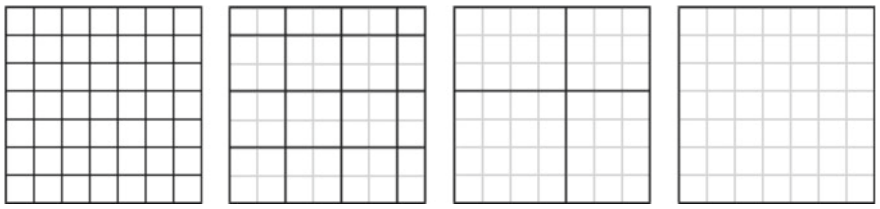


Fig. 6 Downsampling of a grid with odd number of cells in each dimension (7 × 7)

When performing a subdivision on a grid which has an odd number of cells along any of the dimensions, the last cells of the dimension with the odd number which are remaining after divisions by 2 will not be downsampled in the direction of the axis with the odd number of cells. Such blocks have a different size than the predefined block size. A 2D example of the downsampling of a 7 × 7 grid is shown in Fig. 6.

When a cell is downsampled into a larger cell, a new cell shape, and new property values of the cell must be calculated which in a good way represent the original values. This will be described in the following sections.

4.2 *Decimation of Cell Shapes*

The decimation (downsampling) of 8 cells into one coarser cell is performed by taking the convex hull of the 8 outer corners of the cells. In Fig. 7, 2D examples are shown. Detail is lost as the decimated shape is represented using fewer vertices. When a fault is present (right), the subsampling will smooth it out and completely erase it. Fault discontinuities are key features in reservoirs, therefore, a fault-preserving decimation is used.

To describe the fault-preserving decimation, we give a 2D example as in shown Fig. 8 left. The figure consists of 4 × 4 cells of level 0 and a fault (black thick line). The image represents a horizontal layer in the reservoir as seen from above. With

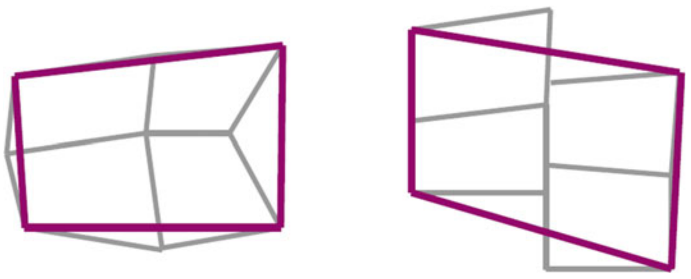


Fig. 7 2D examples of cell decimation. In gray is shown original cells, in magenta is shown the decimated cell. To the right is shown cells on each side of a vertical fault

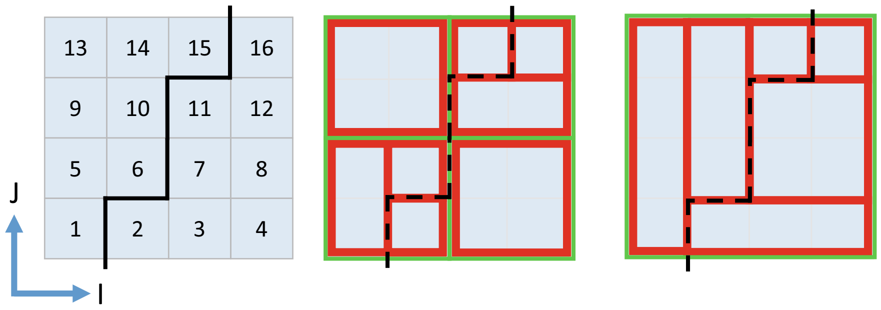


Fig. 8 Left: level 0 (16 cells) with a fault passing through (black line). Middle: level 1 with 8 cells drawn in red. Right: level 2 with 6 cells drawn in red. Green boxes show the cell boundaries for each level where level 1 has 2 × 2 green boxes and level 3 has one

no fault present, the cells at level 0 would have been decimated to 4 cells at level 1 as shown in green in the middle figure, and the cells at level 1 would have been decimated to the one cell at level 2 as shown in green in the right figure. However, to preserve the fault, our algorithm creates level 1 cells by collapsing level 0 cells so that the fault is not crossed, as shown in red in Fig. 8 middle. The 16 cells have been decimated to 8 cells. The next level decimation is shown in red in the right image. The cells at level 0 are used for creating all decimation levels when faults are present instead of using the cells from the previous level.

The algorithm works by merging cells first along the I axis, then the J axis and finally the K axis such that the resulting cell is always cuboid shaped, see Fig. 9. For each direction, merging continues until the coarser cell boundary is reached (green), or a fault is reached (black). The cells that are considered for merging are the ones that have not been merged already. The procedure for the cell expansion shown in the middle of Fig. 8 is now described for some of the cells. Cell 1 is first attempted expanded in I direction, but this is not possible due to the fault. Then expansion in J direction is attempted, which is performed until resolution boundary is reached. Cell 2 cannot expand in I direction due to resolution boundary, and also not in J direction

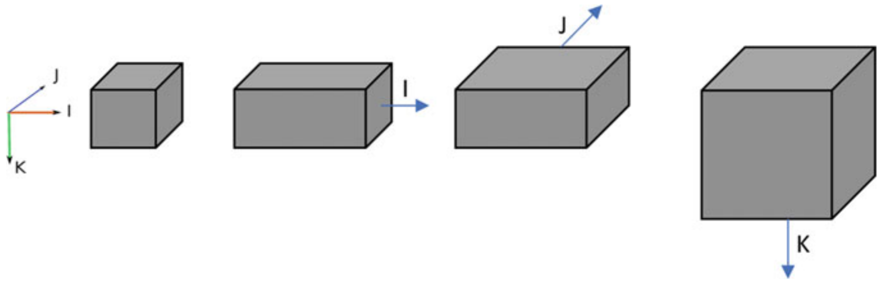


Fig. 9 For decimation, the expansion of a cell by merging it with other cells is done first in I direction, then in J and finally in K direction

due to fault boundary. Cell 3 can expand both in I and J direction until resolution boundary is reached. Cell 4 is skipped as it was already included (during expansion of cell 3), this applies also for cell 5. Cell 6 can neither expand in I or J direction. This algorithm is heuristic and likely not optimal; however, it produces satisfying results. For each reservoir, we perform the algorithm in a preprocessing step and store the result on disk.

4.3 Decimation of Cell Values

This section will describe three different ways to calculate decimated cell values and discuss the advantages and disadvantages of each method.

4.3.1 Storing Average Values

When decimating the properties of a $2 \times 2 \times 2$ cell neighborhood into one cell, one can use the average value as the new cell value. However, this can result in artifacts appearing as sudden change of cell color when there is a switch between two levels in the visualization, e.g., during a zoom-in. When looking at the top-surface of the reservoir as shown in Fig. 10, coarser cells will have their property values affected by nonvisible cells situated directly below the visible ones as described in the next paragraph.

This artifact can be explained by considering a reservoir property that is mapped to a black-green-white gradient color table. In Fig. 11, when the viewer's distance from the cells is increased such that a lower resolution cell is shown instead, the color

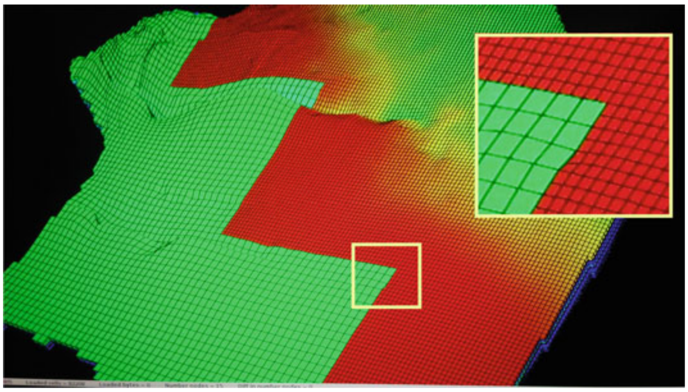


Fig. 10 Property decimation artefact shown at green–red border. Blocks at two different levels of resolution are visualized based on distance to viewer. A zoom-in is shown for the yellow rectangle. The green cells at the border are of a lower resolution than the red cells which is also indicated by the black outlines around each cell

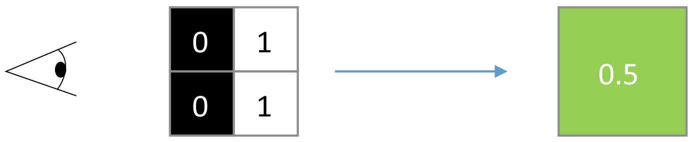


Fig. 11 Nonvisible cells affect the color of visible cells when decimated

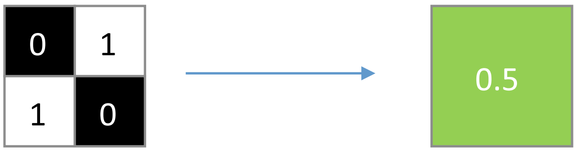


Fig. 12 Using a black-green-white color table, the decimated cell should have been gray, but is instead given the color in the center of the color table (green). This is because averaging cell values is not the same as averaging colors (except for when the color table is a linear transition between two colors)

will suddenly change from black to green even though the surface that the user is looking at is black regardless of the viewing distance. This is because the four cells are decimated into one cell with value 0.5, which is the average value. However, only the two front cells with value 0 mapped to black are visible.

Another error arises when visible cells have different values such as shown Fig. 12. If the fine-resolution cell to the left is decimated by averaging using the same color table as above, the result will be a green decimated cell instead of a gray one. Decimating colors instead of values would resolve this problem, however it is not a practical approach as all property values inside the octree would then have to be recalculated whenever the color table changes. In addition, it would not solve the previous problems shown in Figs. 10 and 11.

We have shown that storing the average value when decimating results in artifacts. We will discuss two possible solutions: storing histograms and storing min–max values.

4.3.2 Storing Histograms

One way to achieve more accurate downsampling and color representation in volume data is to store the histogram of values during decimation. Alternatively, by storing the mean and variance of the histogram instead, which can be considered a compact approximation of the histogram, space is saved, and the color calculation is sped up [16], however this reduces accuracy. When storing a full histogram, the number of bins must be chosen. More bins increase accuracy at the cost of the size of the dataset. In addition, calculating the cell color requires iterating over all bins and performing a weighted blending with the color and size of each bin, which increases rendering time. The histogram accurately represents the underlying value distribution when

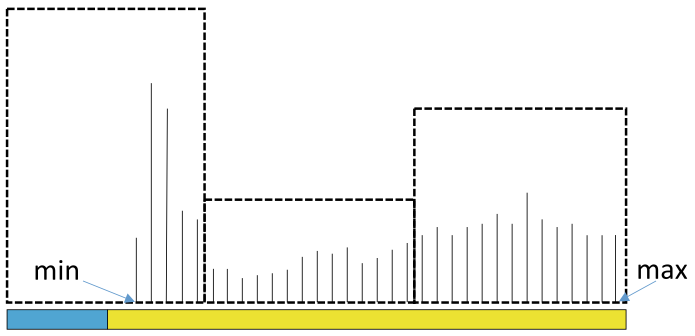


Fig. 13 Approximate histogram with three bins shown with stippled lines made from the full resolution histogram of the underlying data shown as thin vertical lines

having a bin count large enough to accommodate unique values in each bin. For this, an n-bit cell property requires a bin size of 2^n. E.g., an 8-bit property would require storing 256 values for each decimated cell. To use less storage space, smaller bin counts are required. This results in different values ending up in the same bin as exemplified in Fig. 13. The figure shows a hypothetical histogram with only three bins that describe the value in a decimated cell. Full resolution histogram of the underlying values is shown with vertical lines. The color table that is used is shown horizontally at the bottom. It is not possible to accurately represent the cell's color using three bins as the color is not uniform in the leftmost bin. The cell should be colored yellow as only yellow values are present in the underlying data, but according to the coarse histogram, the color blue will be mixed in too, since the leftmost bin is covering both yellow and blue portions of the color table.

4.3.3 Storing Min–Max Values

The min–max solution stores, in addition to the average value, the minimum and the maximum value of the 8 decimated property values from the finer-resolution level. This requires much less storage than a full histogram. Storing min and max makes it possible to detect when it is wrong to use the average value as color. When the colors vary within the min–max interval, we know that the color estimate from the average value is wrong. Then this can be indicated during rendering, or one can render at finer resolution. If the color is constant in the interval, it is possible to identify the correct color. The latter is the case for the scenario in Fig. 13 where the cell can be colored yellow.

The min and max values are only needed for nonleaf cells (nodes) and will therefore not constitute a too large increase in storage of the octree. Details of how the min–max values are used during rendering for identifying which nodes must be refined, is described in the next section.

5 Rendering Large Reservoirs Using GPU Steered Level-of-Detail

Now that the octree structure and decimation of cell shape and cell values are established, we will describe how to use this for rendering large reservoirs.

To achieve fast framerates, we avoid any multipass methods that require several CPU to GPU data transfers per frame and instead limit ourselves to representing the whole reservoir using the available memory on the GPU. This will drastically reduce data transfer times and increase framerates. We need to identify a subset of the blocks, no larger than what can fit on the GPU, which will in a good way represent the reservoir. This can be done by only choosing blocks that are visible from the current viewpoint, and by choosing an appropriate coarseness of these blocks that does not reduce accuracy too much. The coarseness can depend on e.g., closeness to viewer and/or how well the decimated cells match the reservoirs actual colors and shape (level 0).

5.1 GPU Steered Level-of-Detail Selection

For filling up the limited GPU memory wisely, we want to only use blocks that are visible from the current viewpoint, at a coarseness that does not reduce accuracy too much. We will be using feedback from the rendering pipeline to identify which blocks are visible. The accuracy of a block depends on its coarseness (level), its data values and the color table that is being used. The accuracy can be more specifically defined as the "visual difference" between rendering the block and rendering the corresponding original level 0 cells. Since each decimated cell stores min and max values, we can get an indication of accuracy using only the data in the block itself. To find out if the colors of the decimated cells in a block matches the actual colors at the reservoir's original resolution, the color table inside the cell's min–max interval is investigated. Using the average value results in correct color only when all values in the min–max interval have the same color as shown in in Fig. 13. We call such cells for KC (Known Color) cells. One special case of a KC cell is when the decimated values are all equal resulting in min = max. We call all cells that are not KC for UC (Unknown Color) cells. 2D examples of an UC cell is shown in Figs. 11 and 12.

By using various techniques on the GPU which we will explain, we can quickly identify which regions of the reservoir are visible, and which parts of the visible regions are of type UC and of type KC. This information is produced by rendering a fast and coarse reservoir from the current viewpoint. The information is then used to select a subset of nodes to be uploaded to the GPU and rendered, which results in a more accurate rendering.

The idea is to first upload to the GPU only the root node, which represents the whole reservoir at maximum coarseness, and render it. After the rendering, algorithms on the GPU will have collected information about the degree of visibility and the UC and KC pixel count for all the 8 child nodes of the root node even though the child nodes are not on the GPU. This identifies which nodes should be loaded to the GPU and displayed in the next pass. Based on the pixel visibility counts, a subset of the child nodes is uploaded to the GPU. Then after rendering the nodes on the GPU, visibility of grandchildren is estimated. This continues until the GPU memory is full. The approach makes it possible to decide which nodes should be uploaded to GPU but also which nodes which should be removed from GPU memory. Removal candidates are nodes that have not been visible for several frames. Coarsening candidates are nodes with a low pixel count. Refinement candidates are octants with a pixel count larger than some threshold as well as octants with high UC count. If GPU memory is full, nodes with high UC count can be rendered with a color indicating that their real color is unknown. This iterative refinement also ensures that the viewer is given a rendering of the reservoir quickly, possibly with low accuracy, which is then gradually improved.

5.2 *Visibility Counting*

When rendering the colored pixels to screen, they are at the same time rendered to a hidden buffer where their node ids are rendered. Then these ids are counted by visiting each pixel and incrementing the corresponding id counter, in effect creating a histogram of ids.

Different possibilities exist for performing this count. Using the GPU is a natural choice as the counting can be done in parallel. We have chosen the method described by Scheuermann et al. [17] which uses OpenGL vertex shaders and additive blending.

5.2.1 Rendering Ids

Figure 14 shows the steps of visibility counting. In Fig. 14a is shown an octree node, it has a unique id and consists of e.g., $50 \times 50 \times 50$ cells (individual cells not shown). When the node is not a leaf node, it points to finer scale representations corresponding to its octants shown in Fig. 14b. Each of these 8 parts is again represented by a node with unique id and the same number of cells. Unique node ids are shown in Fig. 14b with a color coding used for representing the ids also used in Fig. 14c–f. During rendering, when setting the color of the screen pixels, pixels are at the same time set in an off-screen texture (OpenGL Render Target) that stores node-ids. This is done at practically no additional cost. In this off-screen texture, the node-id of the octant is stored and is shown in color in Fig. 14c. Notice that the ids of the children are rendered when rendering the parent node without needing to render the children nodes. This is possible because the parent knows that for its 8 octants there exist higher resolution children representations.

After rendering, we proceed to go over all pixels in the off-screen texture and count the number of pixels for each child node id. Each node id and its count is shown at

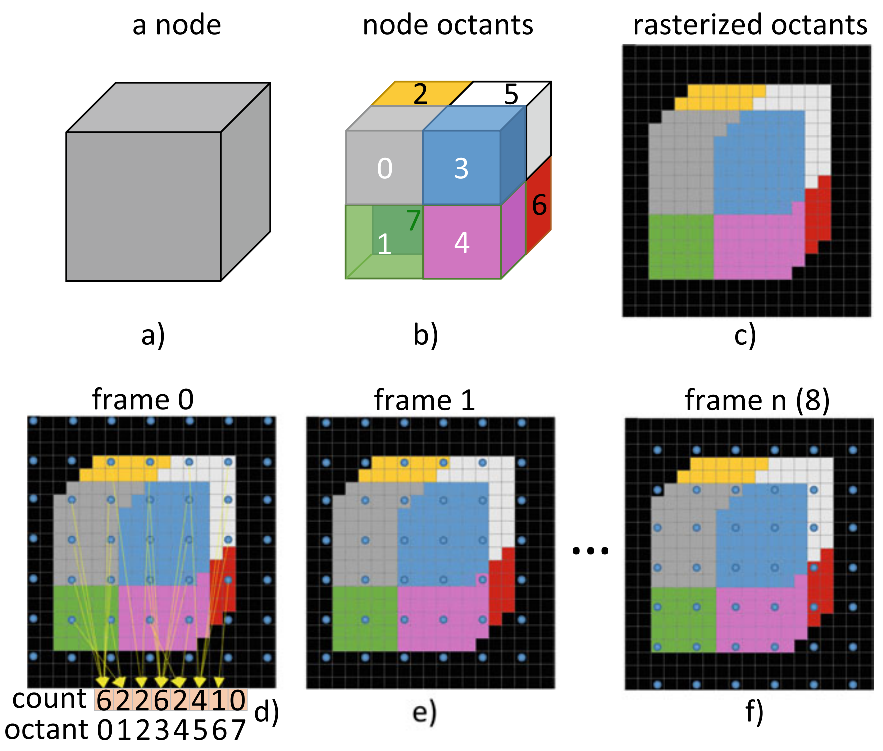


Fig. 14 Visibility counting

the bottom of Fig. 14d. As counting all pixels in an image is computer intensive (e.g., an image of size 1000 * 1000 has 1 million pixels that must be counted), we divide this calculation over several frames as shown in Fig. 14d–f. A regular sampling is done for each frame, and this sampling is shifted by 1 pixel at the next frame. The pixels counted/sampled in each frame are shown with blue circles in Fig. 14d. In our implementation, the number of frames needed for covering all pixels can be set by the user with a constant (`Count`). It is set to $3 \times 3 = 9$ for the example in Fig. 14 and to $5 \times 5 = 25$ in the implementation. To keep track of how many pixels belong to each node-id, there is a separate counter per node id. Each counter is stored in a specific position in a texture residing on the GPU (shown as the red array at the bottom of Fig. 14d).

Counting is distributed over several frames, so during interaction with the reservoir model, such as rotation, the system does not have an accurate node-count of the current rendered image, but an "average" count of the nodes visible in the last `Count` frames. When interaction is stopped, the count will converge and is correct after `Count` frames.

5.2.2 Counting Ids

For counting the node ids in all pixels, a vertex shader is applied on a grid of point primitives where each point is covering a pixel (shown as circles in Fig. 14d–f). The vertex shader translates the point primitives from their pixel location (shown as arrows in Fig. 14d) to the node counter location using a predefined one-to-one node-id to node-counter mapping.

The points translated by the vertex shader are given color (1, 0, 0) and OpenGL blending is turned on and set to additive. This results in an increment of 1 in the red channel of the node-counter. High valued counts are supported as the framebuffer has 32-bit floating point precision. However, the blending stage is a potential bottleneck in our algorithm as simultaneous additions to a specific texture position cannot be performed in parallel. This may make the processing time of our algorithm slower when many pixels have the same node-id. Spreading the calculation over several frames and the sample points over space as we do, reduces the problem. By stopping the counting after a certain threshold has been reached, we can also reduce the problem further.

Figure 14 is simplified as it does not differentiate between UC and KC count. In the actual implementation, during rasterization, each id-pixel in Fig. 14c is tagged to be of either type UC or KC, in addition to storing a node-id. The tagging is done by the shader that renders cells, based on comparing the min–max value of the cell with the color table. The pixel where the count is stored, is storing the UC and KC count in the red and blue channel respectively.

5.2.3 Addressing

All nodes in GPU memory are rendered. Eight counters are allocated to each node so that individual octants (which correspond to child nodes) are counted. These 8-tuples are stored next to each other in texture memory as shown in the row named `Accum` in Fig. 15. The UC and KC counter for its 8 octants are shown in red and blue respectively. The position of a specific node's counters is calculated based on the node-id. For storing all the node counters, we need to know how much texture memory to allocate. For large reservoirs, the number of node-ids can become large. For instance, having a reservoir grid of size 2000^3 with nodes of size 40^3 results in $2000^3/40^3 = 50^3 = 125{,}000$ leaf nodes. The number of nodes including all resolution levels is slightly higher ($50^3 + \lceil 50^3/8 \rceil + \lceil 50^3/8^2 \rceil + \lceil 50^3/8^3 \rceil + \cdots + \lceil 50^3/8^6 \rceil =$ 142,859). For any octree, based on a geometric series calculation, the internal nodes constitute about 1/7th of the leaf nodes. However only a fraction of the nodes is being rendered at any given time. Therefore, by remapping the node-ids from an id that is unique for the whole octree to an id that is unique among the nodes currently uploaded on the GPU, we reduce the texture address space. E.g., if node ids 2135 and 1111 are the only nodes on GPU, they are mapped to the ids of 1 and 2. The CPU maintains the mapping $2135 \Leftrightarrow 1$ and $1111 \Leftrightarrow 2$ while the GPU only knows about id 1 and id 2. Then if the node with id 2135 is removed from the GPU and

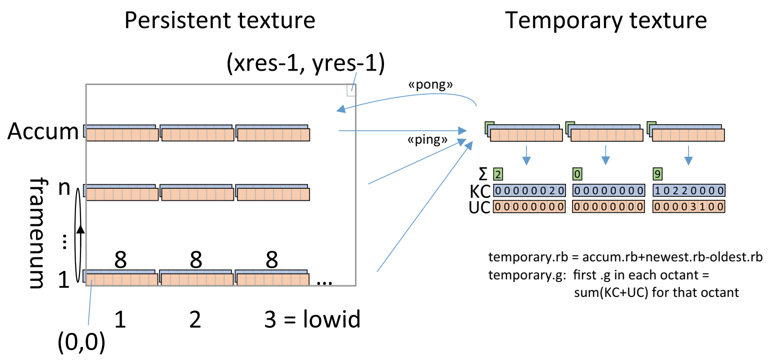

Persistent texture
Temporary texture
(xres-1, yres-1)
«pong»
«ping»
Accum
framenum
n
...
1
8
8
8
1
2
3 = lowid
(0,0)
Σ 2 0 9
KC 00000020 00000000 10220000
UC 00000000 00000000 00003100
temporary.rb = accum.rb+newest.rb-oldest.rb
temporary.g: first .g in each octant = sum(KC+UC) for that octant

Fig. 15 Textures for storing and calculating node-counts

another node with id e.g., 12,334 is added, then GPU id 1 is used for the new node (12,334 ⇔ 1) so that the GPU ids stay as low as possible. We call this GPU "alias" id for `lowid` (see the numbered columns named `lowid` at bottom of Fig. 15).

5.2.4 Representing Node-Counts in a 2D Window-Sized Texture

The visibility count for each node is calculated and stored in the texture shown to the left in Fig. 15. The texture is an RGB texture with dimensions equal to the window size as shown in the upper right corner. The actual needed space for storing the node-id counters has no logical correlation to the window size but at the time of implementation, the library we used that abstracted OpenGL did not support better alternatives such as using OpenGL's Shader Storage Buffer Objects.

To reduce overhead per frame, we distribute the calculation over several frames by counting only a subset of pixels each frame, so that after so that after `Count` frames, all pixels have been counted. If we keep accumulating the count from each frame, the values will increase indefinitely. To avoid this, we must subtract from `Accum` the count which was taken `Count` frames ago. Therefore, we store the count of all the last n (=`Count`) frames in a cyclical buffer. This cyclical buffer resides in the first n rows of the Persistent texture as depicted with a cycle in Fig. 15. All counters in `Accum` are initially set to 0 and for each frame we add to `Accum` the pixelcount of the current frame and subtract the pixelcount of the oldest frame in the cyclical buffer. An alternative that requires no cyclical buffer would be to use the count result for the current frame only, as an estimate of the visibilities. This estimate is representative as the sampling is evenly dispersed over the image.

Due to GPU restrictions in our implementation, simultaneous reading and writing to the same texture is not supported, therefore we must store the result of the addition and subtraction in a temporary texture (Fig. 15, Right) and then copy it back to the `Accum` row in the Persistent texture. Processing using values from one texture and storing the result in another is often referred to as "ping-ponging". The name arises from the process of calculating back and forth between two textures over several passes, for instance for performing smoothing on an image texture. In our case, we are "pinging" a sum into a row in a temporary buffer and "ponging" the row part back to `Accum`.

The main calculations have been done now and the `Accum` row having full node-count information can be copied to the CPU. The original node-ids are found using the `lowid` mapping. By summing up the 8 UC and 8 KC counts we get the number of visible pixels of that node, while each of the 8 positions store the UC and KC count for the corresponding child-node of that node. Notice that the child nodes themselves are not resident in GPU memory. Now we can identify child-nodes that should be loaded to the GPU based on high visibility or high UC count, and nodes in GPU memory that should be removed due to low visibility.

5.2.5 Extra Processing on the GPU for Identifying Interesting Nodes

Often only a small subset of the rendered nodes needs to have the resolution changed or be removed. Therefore, we use the GPU to identify such interesting nodes to reduce the sequential search through the node info on the CPU. To do this we extend the "ping" stage to also calculate the sum of the 8 UC and 8 KC counts, constituting the node's visible pixels. The sum is stored in the green channel in the first index of each octant as shown right in Fig. 15. Bottom right of Fig. 15 shows example counts for the three first node ids. The second node has a sum of zero, indicating that it is not visible at all. By copying this row to the CPU, we have access to these precalculated sums.

We also extend the "pong" stage to identify nodes with nonzero count. Table 1, row 1 and 2, shows an example-count for 20 nodes. To identify the visible nodes (nodes with count larger than zero) on the CPU, one would need to iterate over all 20 nodes. If the GPU could identify them and produce a table as shown in the row called "Sorted", the CPU would save time. In our solution, a compromise is done where the GPU performs a partial sorting. An example of a "4-sorting" is shown in the bottom row in Table 1 where the list is divided into clusters of size four and each is sorted, creating 5 sorted lists. The GPU can create these 5 lists in parallel, and the CPU will save some time as it will visit fewer empty elements.

The size of each cluster has been set to 16 and is called `cs`, see Fig. 16. The "cs-sorted" list is stored in the green channel. The first index points to the first nonempty

Table 1 Visibility counting

Count	3	0	5	2	9	0	0	0	234	0	14	0	0	0	0	0	0	0	0	29
Node-id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
On GPU	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Sorted	{1, 3, 4, 5, 9, 11, 20}																			
4-sorted	{1, 3, 4}				{5}				{9, 11}				{}				{20}			

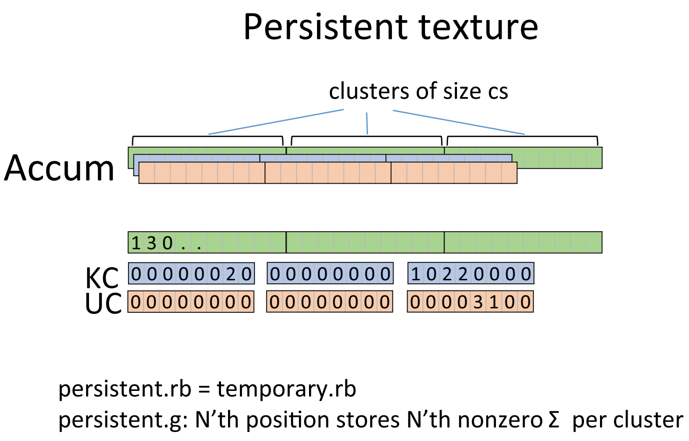

Persistent texture
clusters of size cs
Accum
1 3 0 . .
KC 0 0 0 0 0 0 2 0 | 0 0 0 0 0 0 0 0 | 1 0 2 2 0 0 0 0
UC 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 3 1 0 0
persistent.rb = temporary.rb
persistent.g: N'th position stores N'th nonzero Σ per cluster

Fig. 16 Layout of the `Accum` row in the Persistent texture

node (with value 1), the second points to the second nonempty node (3) and when there are no more nonempty nodes the following indices store the value 0. The first cluster does this for the first `cs` nodes, the second cluster does it for the following `cs` nodes and so on. To access this list, it must be copied from GPU to CPU.

An actual snapshot of the Persistent texture taken while the program is running is shown in Fig. 17. The top row having green pixels is the `Accum` row, while the 25 rows below are the cyclical buffer with counts for the 25 previous frames where red pixels represent cells with unknown color (UC).

This approach can be extended so that the GPU calculates in parallel a small list specifying which nodes should be removed and which should be added. This would relieve the CPU of sequential work and minimizes data transfer between CPU and GPU. It can be achieved if a list of all nodes together with a Boolean value specifying if the node is in GPU memory or not, is maintained on the GPU. The list is shown in the row "on GPU" in Table 1. The GPU can with this list, find nodes that are candidates for removal by identifying nodes that are on the GPU but with visibility below some minimum threshold (on GPU = 1 and Count < minimumVisibility) such as node 6 in Table 1 for minimumVisibility = 1, and nodes that are candidates for uploading to the GPU by identifying nodes that are not on the GPU but have visibility over a certain threshold (on GPU = 0 and Count > threshold) such as node 9 in Table 1 for threshold = 200. After the CPU has decided which nodes it should

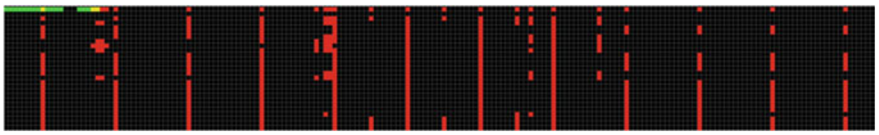


Fig. 17 Screenshot of the Persistent texture. Top row shows the Accum row. The following rows represent the cyclical buffer

remove and add, it would have to update the “on GPU” list. Using parameters such as minimumVisibilty, threshold, and more detailed ones such as “minimum UC count” and for how many frames a node must have been below visibility threshold before removing it, the user can fine tune the way nodes are refined. The responsiveness and speed of convergence of the system can be tuned by setting parameters such as `cs` and `Count`.

6 Results

Figure 18 shows a rendering of a large reservoir containing 600 million cells which is rendered at interactive framerates.

In Fig. 19 is shown a screenshot of the whole GUI including a rendering of the Johansen reservoir which is publicly available [18]. The GUI allows for choosing what reservoir property to visualize, for assigning a color table to the property, and for selectively visualizing cells that have properties within a certain range. It is also possible to only show cells which are in specific ranges of the I, J and K cell indices.

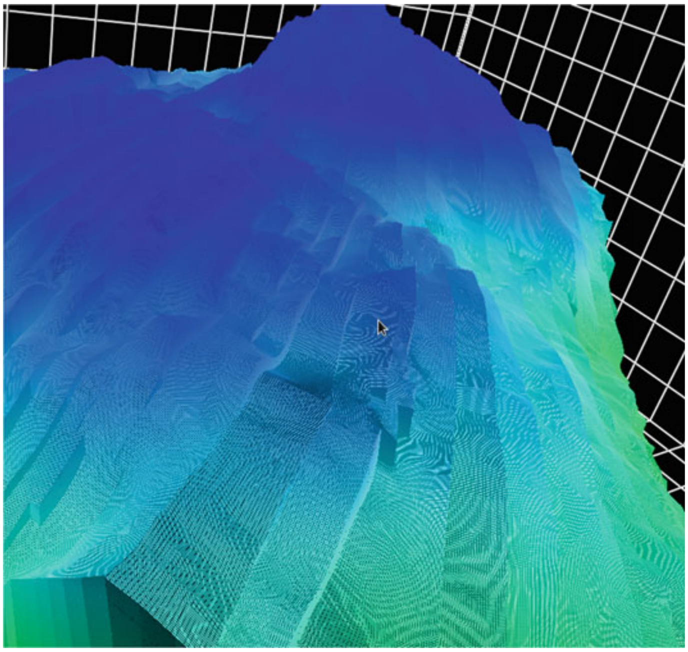


Fig. 18 Rendering of a reservoir with 600 million cells

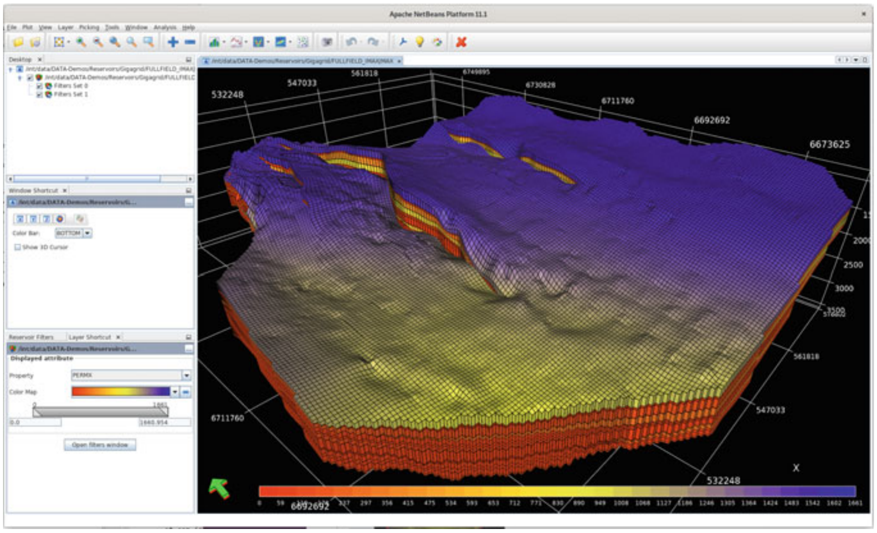


Fig. 19 The user interface with cells colored according to permeability, color table is shown at the bottom

6.1 Rendering Cell Outlines

In addition to rendering each cell with a specific color, the cell outlines are also rendered with black lines. This is useful for showing the cells' shapes and sizes but is only correct for level 0 cells. Higher level cells are larger as shown in Fig. 10. To address this, artificial outlines are added to higher-level cells in the rendering process so that the correct number of cell outlines are shown. When the viewing distance to the cells is large enough, the cell outlines will be so dense that the cell color is not discernible. To avoid this, a rendering mode can be used where the outlines of the cells at the level above are shown instead, and this transition is smoothed by gradual fading in and out of the outlines. The cell outlines are rendered in a fragment shader. Using the barycentric coordinates of each triangle makes it possible to display a grid on each cell face according to the level of the cell. The thickness of the lines can also be controlled and vary according to the distance to the viewer.

6.2 Adding Shadows for Improved Depth Perception

Shadows help understand the relative positioning of objects and the distance between them. Calculating shadows can be computing intensive and reduce the framerate. We used a fast and well-established method that only creates shadows from nearby geometry, called screen space ambient occlusion (SSAO) [19]. This method reduced the framerate insignificantly. Figure 20 left, shows a rendering of a reservoir without

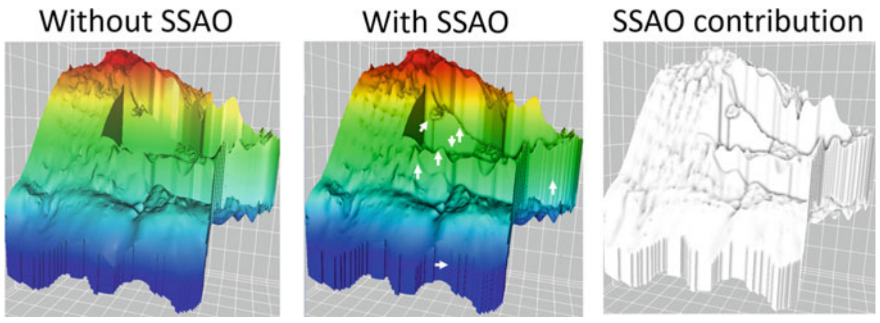


Fig. 20 Adding shadows using screen space ambient occlusion (SSAO). White arrows in the middle figure point to areas where depth and occlusion is better perceived due to shadows being cast

SSAO shadows. The middle figure shows a rendering with SSAO. The shadowing calculated from the SSAO are shown in the right figure. Areas benefitting from SSAO are indicated with white arrows in the middle figure.

7 Conclusions

In this chapter we have introduced a method for rendering large reservoir grids that do not fit GPU memory. We have reviewed the cornerpoint grid datastructure that is able to represent the faults and layers of a reservoir. We have shown how the octree datastructure is useful for representing the reservoir in a granular manner with multiple levels of resolution. For downsampling the original grid to lower levels, we have shown how grid cell shapes must be preserved across faults, and how the minimum and maximum property values of finer resolution cells should be stored in coarser resolution cells. A GPU accelerated method that compares the min and max values of cells with the color table is used for deciding which nodes in the octree should be rendered. This approach selects nodes from the octree for optimizing the accuracy of the rendered reservoir while using a limited amount of GPU memory.

References

1. P. Jenny, C. Wolfsteiner, S. H. Lee, L.J. Durlofsky, Modeling flow in geometrically complex reservoirs using hexahedral multiblock grids. SPE J. (2002)
2. C. Silva, J. Comba, S.P. Callahan, F. Bernardon, A survey of GPU-Based volume rendering of unstructured grids. Brazil. J. Theor. Appl. Comput. (2005)
3. M.P. Garrity, Raytracing irregular volume data, in *Proceedings of the 1990 Workshop on Volume Visualization* (1990)
4. M. Weiler, M. Kraus, M. Merz, T. Ertl, Hardware-based ray casting for tetrahedral meshes, in *IEEE Visualization* (2003)

5. P. Muigg, M. Hadwiger, H. Doleisch, H. Hauser, Hybrid unstructured and structured grid raycasting. IEEE Trans. Visual. Comput. Graph. (2007)
6. P. Muigg, M. Hadwiger, H. Doleisch, E. Gröller, Interactive volume visualization of general polyhedral grids. IEEE Trans. Visual. Comput. Graph. (2011)
7. C. Giertsen, Volume visualization of sparse irregular meshes. IEEE Comput. Graph. Appl. (1992)
8. T. Frank, Advanced visualisation and modeling of tetrahedral meshes. Ph.D. thesis, Institut National Polytechnique de Lorraine (2006)
9. P. Jean-Luc, D. Laurent, P. Frédéric, B. Lauriane, C. Lenaic, S. Schneider, M. Antonini, HexaShrink, an exact scalable framework for hexahedral meshes with attributes and discontinuities: multiresolution rendering and storage of geoscience models. Comput. Geosci. (2019)
10. F. Abraham, W. Celes, Multiresolution visualization of massive black oil reservoir models. Visual Comput. (2019)
11. F. Abraham, W. Celes, Distributed visualization of complex black oil reservoir models, in *Eurographics Symposium on Parallel Graphics and Visualization, EGPGV* (2009)
12. B. Franceschin, F. Abraham, L.F. Netto, W. Celes, GPU-based rendering of arbitrarily complex cutting surfaces for black oil reservoir models, in *Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)* (2019)
13. L.Q. Campagnolo, W. Celes, An experimental study on volumetric visualization of black oil reservoir. Comput. Graph. (2020)
14. K.-A. Lie, *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave* (Cambridge University Press, 2019)
15. H. Samet, An overview of Quadtrees, Octrees, and related hierarchical data structures, in *Theoretical Foundations of Computer Graphics and CAD. NATO ASI Series* (1988)
16. H. Younesy, T. Möller, H. Carr, Improving the quality of multi-resolution volume rendering, in *EuroVis 2006* (2006)
17. T. Scheuermann, J. Hensley, Efficient histogram generation using scattering on GPUs, in *Symposium on Interactive 3D graphics and games (I3D)* (2007)
18. The Johansen Data Set [Online]. Available: https://www.sintef.no/projectweb/matmora/downloads/johansen/. Accessed Mar 2021
19. Hardware accelerated ambient occlusion techniques on GPUs, in *Symposium on Interactive 3D graphics and games (I3D)* (2007)

When Visualization and Virtual Reality Made a Paradigm Shift in Oil and Gas

Endre M. Lidal, Tor Langeland, and Jens Grimsgaard

Abstract In the late 1990s, Virtual Reality (VR) and advanced visualization technology created a paradigm shift in the work processes in the petroleum industry. Major contributions to this shift came from the results of the R&D VR project presented in this chapter. The chapter describes the development of a VR platform for production, especially well planning, and exploration reconnaissance. The platform supported cross-disciplinary work inside a CAVE setup which substantially reduced the time to plan well trajectories.

1 Introduction

In the late 1990s, Virtual Reality (VR) and advanced visualization technology created a paradigm shift in the work processes in the petroleum industry. Major contributions to this shift came from the results of the R&D VR project presented in this chapter.

This project was initiated in the Norwegian oil company Norsk Hydro and was given the name *HydroVR*. In 2007, Norsk Hydro and Statoil merged into one company called StatoilHydro. The project and corresponding application was renamed *SHIVR* (StatoilHydro Integrated VR). There will be references to both company names in this chapter. Finally, in 2018, Statoil changed name to Equinor.

E. M. Lidal
Ulriken Consulting, Bergen, Norway

T. Langeland (✉)
NORCE Research, Bergen, Norway
e-mail: tola@norceresearch.no

J. Grimsgaard
Equinor, Bergen, Norway
e-mail: jegri@equinor.com

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_7

In this chapter, we give the background for this successful project, how it was carried out, show examples of developed tools that changed work processes, and share lessons learned. We believe that experiences from this project will be valuable input for other software projects in the oil and gas business. For further technical descriptions of the application, we also recommend our previously published article [9].

2 Background

From the mid-1980s, graphics workstations became an important tool for experts working with oil and gas exploration and production (E&P). However, these workstations had shortcomings, especially when it came to viewing and interacting with 3D data on small screens with poor resolution. In late 1980s, the foundations of VR were made, and the beginning of the 1990s saw the introduction of VR technology with large-screen display systems, utilizing stereo for 3D effect. The automotive and the aerospace industries were some of the early adaptors of this new technology [1]. Inspired by their work and promising results, Norsk Hydro, in collaboration with the Norwegian research institute Christian Michelsen Research (CMR), now merged into NORCE, conducted in 1997 a pre-study on adapting VR to Oil and Gas industry. The pre-study concluded with the recommendations for establishing a CAVE laboratory for E&P related applications.

In Norway, research in the Oil and Gas business is mainly funded through the oil-field license agreements. Approximately 2% of invested capital of a license shall be reinvested in research. The research funds are administered by the operators and are primarily focused on the R&D upstream. In the mid-90s a certain part of the research funding Norsk Hydro controlled, was allocated to so-called “Blue Sky”-projects. For these projects, the Research Department could select the content themselves without intervention from the business units. This gave the research department a high degree of freedom and the opportunity to initiate what was looked upon as high-risk projects.

In the light of the conclusion of the pre-study mentioned above, a “Blue Sky” project proposal was submitted. The project proposal stated that the goal was to make use of VR to “radically improve the critical operations within exploration and production, increase quality, and improve cost efficiency”. The expected results were to:

1. Generate new prospects
2. Secure well placements
3. Increase recovery
4. Improve precision
5. Reduce turnaround time for interpretation, modelling and analysis
6. Improve multidisciplinary communication
7. Improve communication between E&P’s core disciplines and management, partners, government, and community

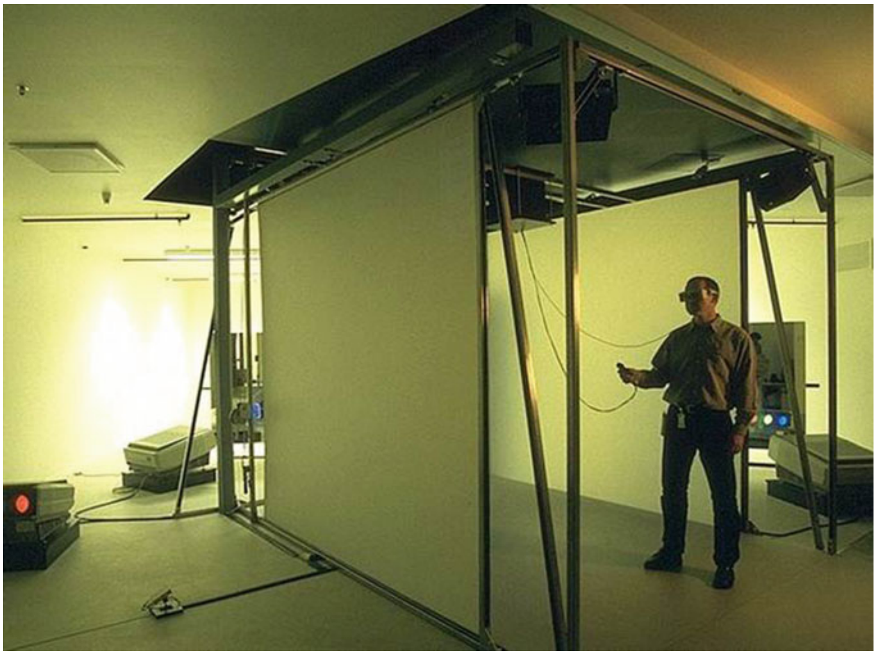


Fig. 1 The CAVE-installation that was installed at Norsk Hydro's research center in Bergen

8. Profile Norsk Hydro as an innovative and leading company in use of new IT in petroleum disciplines

Norsk Hydro decided to initiate the VR-project with "Blue Sky"-funding and was therefore among the very first companies that started looking at the possibilities for utilizing the VR technology for E&P work processes. When the VR project started, in the summer of 1997, it was given an initial funding of $2,5M.

Following the recommendations of the pre-study report, Norsk Hydro invested in a CAVE laboratory. In the fall of 1997, a CAVE™ was installed at Hydro's Research Centre in Bergen. The CAVE, or Cave Automatic Virtual Environment, had been invented by the University of Illinois, Chicago [3]. It consisted of one front wall (3 $\times$ 3 m^2), two equally sized sidewalls and a projected floor, see Fig. 1.

The walls were back-projected to avoid shadows from the user. The users wore stereo glasses[1] to experience a true 3D vision of the displayed models. The glasses of the main user, the pilot, were tracked[2] in such a way that the 3D perspective was always based on the pilot's head position. The pilot also held a tracked 3D mouse, called a Wand. The Wand gave the pilot an opportunity to interact with the VR-application and the displayed objects. The CAVE provided a very strong feeling of being immersed and present in the data (see Fig. 2). This required high and steady

[1] 3D effects are created by displaying different images for each eye.

[2] A sensor attached to the glasses continuously sends position and orientation to the computer.

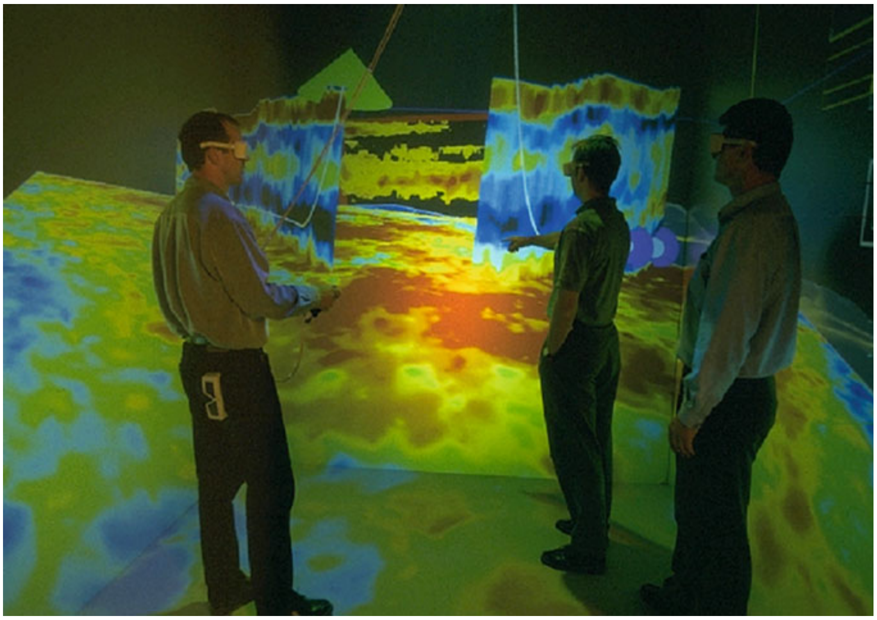


Fig. 2 The CAVE provides a strong feeling of being immersed in the data. The users wear glasses to experience true 3D vision. The leftmost user is the pilot, controlling the application with use of the 3D mouse

frame rate,[3] because the user needed to feel that the visualized scene was updated according to his or her movements without significant delay. This required a frame rate of at least twelve frames per second, which is in today's standard very low.

There are two major drawbacks of the CAVE. First, the CAVE requires a lot of installation space, as one can see from Fig. 1. Even though the CAVE setup utilizes mirrors to allow a shorter distance between the projectors and the CAVE walls, the required installation space is still many times the area within the CAVE. The second drawback of the CAVE is high costs, even though the computer hardware costs are much lower now than at the time the CAVE was installed at Norsk Hydro.

At the time, no commercial software for advanced visualization, in any scientific domain, supported the CAVE platform. Reasons for this were that the hardware for running the CAVE was very expensive and specialized, and implementing visualization functionality for CAVEs was very different from implementing the same functionality on workstations. Since no commercial off-the-shelf solutions existed, all the software for the CAVE had to be developed. This task was assigned to CMR, and this became the start of a close collaboration between the software researchers at CMR and the E&P experts at Norsk Hydro.

Developing all the software from scratch would have been time consuming, resulting in an undesirable long time before proof of concept could have been made.

[3] Frame rate is the number of times the computer updates the scene per second.

Luckily, some academic research institutions had already developed VR libraries and applications. Norsk Hydro and CMR visited several of these, among others the National Center for Supercomputing Applications (NCSA). NCSA had developed a VR application for biological applications [2]. After studying the design of this application and the results it could produce, and comparing this to the needs of our application domain, we decided to purchase its program source code. This source code provided a starting point for our own VR software framework and for several prototype tools, e.g., for volume visualization of seismic data. This made it possible to have the first version of the CAVE E&P software, called HydroVR, up and running within a few months and the users at Norsk Hydro could start gaining experience with use of VR in their work processes.

3 Early Focus and Development

It was important for the project to identify specific workflows and tasks, for which the technology could be developed and tested. Two clearly defined workflows were acknowledged as "low-hanging fruits":

1. Well planning
2. Exploration reconnaissance

We very quickly saw a need to involve the end-users. A tight collaboration with these ensured engagement and ownership. In addition, we also saw the need for a dedicated team in the Research Department at Norsk Hydro with responsibility for:

1. Defining new functionality for the VR-application
2. Following up the development at CMR
3. Testing the functionality on real data
4. Implementing the results in new workflows
5. Operating as pilots in work sessions with cross-disciplinary teams from various business units
6. Collecting feedback from the teams to improve existing or define new functionality

3.1 Well Planning

Prior to the VR-project, well planning was performed in a sequential flow, see left part of Fig. 3. Each discipline had its own constraints and goals, so the traditional well planning process was a time consuming, iterative process of exchanging data and models back and forth between the different involved disciplines. The planning procedure could take 3–4 weeks before the well was agreed upon.

The CAVE could easily accommodate 6–8 people, so typically, we would have a multi-disciplinary group with geologists, geophysicists and a drilling engineer

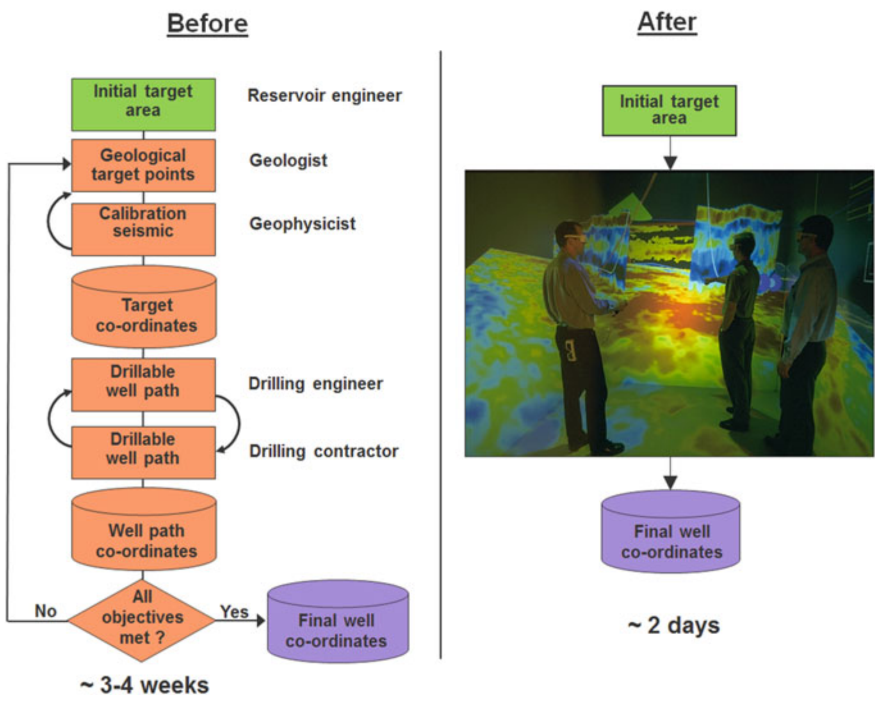


Fig. 3 Schematic of the well planning work process prior to and after introduction of VR-technology

working together, discussing problems and finding the solutions (see the right part of Fig. 3). In other workflows, we also involved reservoir engineers and production engineers. All the involved data types were rendered in the same coordinate system in a common view. Each discipline could show, using their specific data, their constraints, challenges, goals and opportunities. At the same time, they could all see the bigger picture by also considering the information from the other involved disciplines. In addition to adding increased value to the work sessions, this also enhanced creativity and understanding. Gathering the various disciplines in the same room and co-visualizing all the necessary data in one virtual world accelerated the well path planning tremendously. Figure 4 illustrates some of the disciplines collaborating in the CAVE.

The data used in the well planning workflow would typically be seismic data and corresponding attributes, interpreted horizon surfaces, existing wells and logs, and interpretation of fault surfaces. The well planners positioned the well target points in 3D and the computer calculated a well path between the target points, as shown in Fig. 5. The output well path file would then be sent to the well contractor for QC. The well contractor made adjustments to the path if the proposed curvature was not compliant.

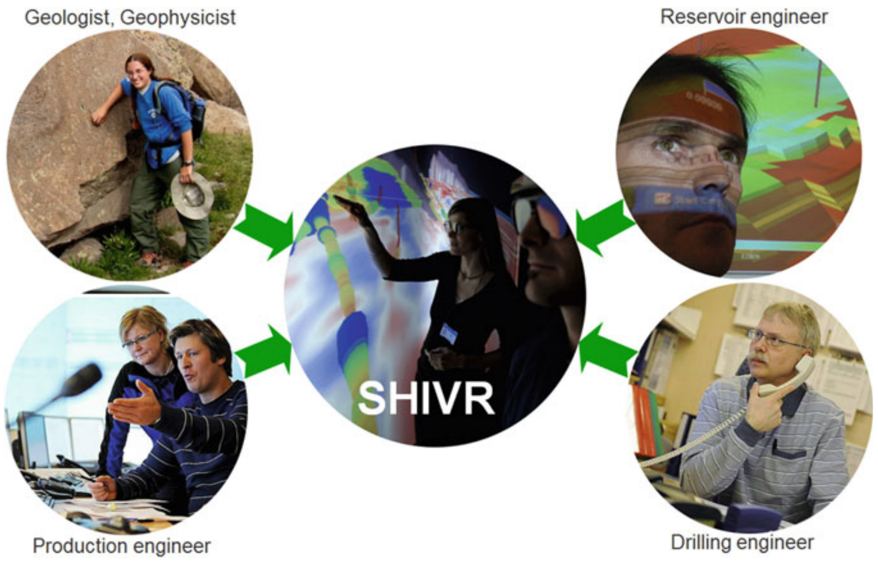


Fig. 4 Cross-disciplinary collaborative work in SHIVR

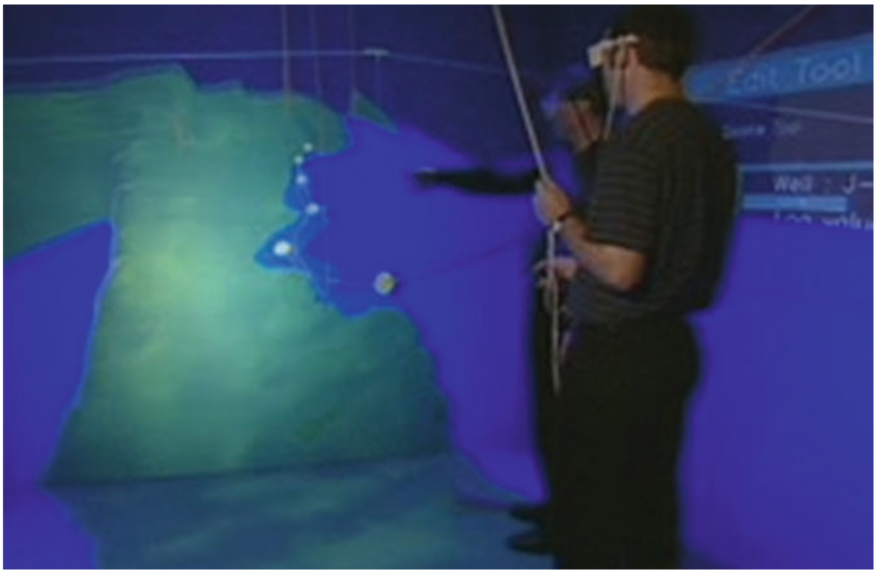


Fig. 5 Planning a well path in the CAVE

Early experience showed that using SHIVR in the CAVE reduced the well planning workflow from 3–4 weeks to 2–3 days. Even more important, the quality and the production rate from the final well paths improved considerably. The combination of large screen visualization and cross-disciplinary collaboration enhanced the creativity for defining these well paths. One example is from the Oseberg oil field in the North Sea, west of the Norwegian coast. The Oseberg field consists of a formation group from Middle Jurassic time, called the BRENT. The Brent group consists of five formations: Broom (named Oseberg on the Norwegian continental shelf) at the base, then Rannoch, Etive, Ness and Tarbert at the top (take the first letter of each formation and you get the name BRENT). These are deposits of a prograding delta system, which is shallowing upwards in the stratigraphy. The sand in the Ness represents paleo fluvial channel deposits from a coastal river plane. These channel deposits are patchy and not always well connected. Early wells targeting the Ness-sands had a sand zone of 35% prior to the use of the CAVE. Improved models from seismic inversion and the use of SHIVR increased the sand zones in the new wells with up to 50%. With the new inversion cube, there was a higher confidence that low impedance in the Ness zone represented sand. The CAVE environment offered a visual arena where new well paths could be interactively planned by applying natural gestures in the 3D model space where all relevant subsurface data was available. Four new Ness-wells, planned in the CAVE, had a total increase of 750.00 Sm^3 oil, representing a value of $80M (using $16.5/barrel, which was a typical price for the late 90s) [11].

The production units also took advantage of the CAVE during the actual drilling of the wells. The well planning team came together in the CAVE to discuss the latest drilling results and the implications these had for the understanding of the field. One member of the dedicated research team was given the task to update the project with the latest logs, well paths, and interpretations, in addition to act as the pilot in the CAVE sessions.

3.2 Exploration Reconnaissance

The other workflow where VR was tested with positive results was the early exploration phase. When exploring a new area, seismic data is collected. In many cases, such an area has limited data coverage and therefore the geological knowledge of the area is limited. Gathering a team of experienced geologists and geophysicists in the CAVE to discuss the data was extremely valuable. The interpreters may have varied but complementary background and experience and the CAVE functioned as a media for brainstorming, discussions, and creativity.

In the VR-application, dedicated functionality for seed-picking and growing of seismic anomalies reveals geological features in the seismic data, as shown in Fig. 6 for a deep-marine fan system. Horizons can be automatically tracked and seismic attributes can be generated and displayed as textures on these horizons. The next section provides more details on each of these functionalities.

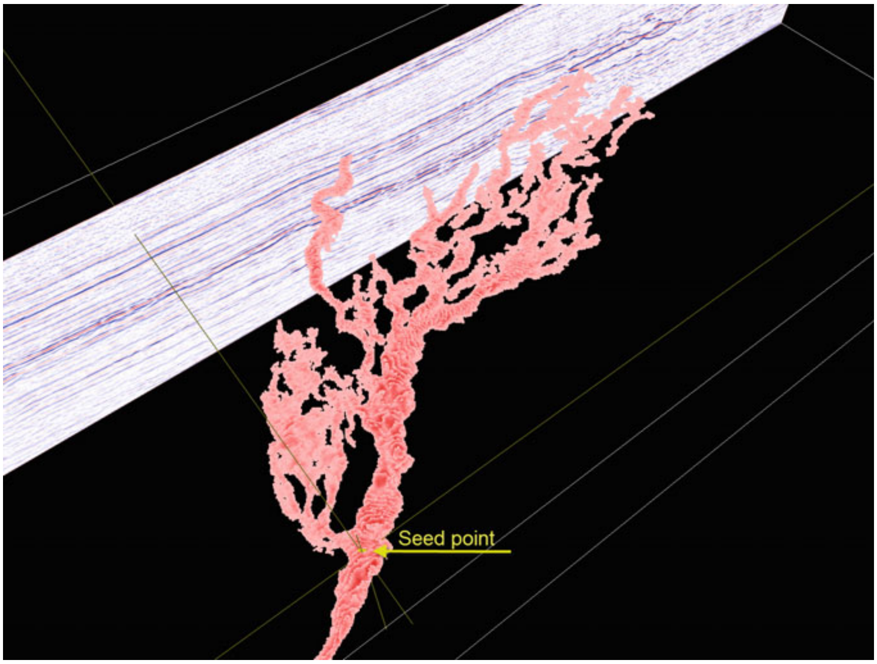


Fig. 6 Automatic grown deep-marine fan system based on one seed point

This data reconnaissance gave the exploration team a flying start, enabling them to focus their work at a very early stage. Bear in mind that all exploration projects have limited time and resources to establish an understanding of the area to decide whether a continued investment and exploration in the area should be performed. Early focus improves the foundation of such decisions.

4 Developing the VR Application

To achieve early results, it was important to select an efficient software development approach. As mentioned earlier, VR software for biological applications was acquired to give the project a flying start. Additionally, the overall software architecture that was selected facilitated effective software development.

This architecture consists of three main components (see illustration in Fig. 7):

- The first component is the kernel, which controls the program flow and manages input and output devices.
- The second component is the tools, which are plug-ins to the kernel. The tools control visualization of one or more data types, e.g. volume visualization tools, or they support work processes, e.g. the tool for well planning.

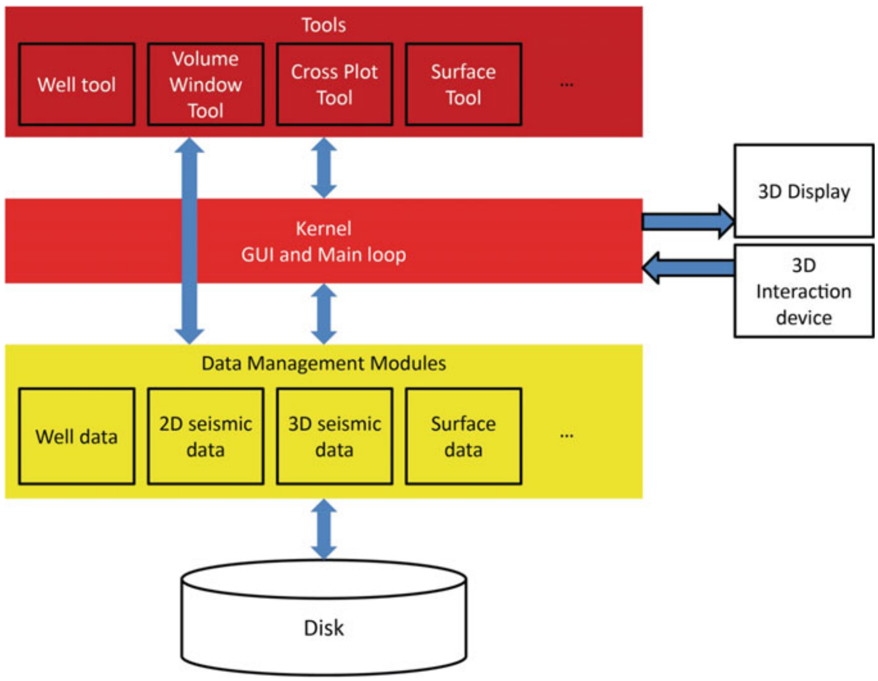


Fig. 7 The SHIVR architecture consists of three main components, the kernel that controls the program flow and manages the input and output, the data management modules, and the tools

- The third component is the data management modules, which are plug-ins for storing and managing data. By separating the tools from the data management, we have ensured that data types can be shared between many different tools.

One way of viewing this architecture is that the kernel represents a "VR operating system" where one can plug in different applications (tools) supporting different data types and work operations. Once the kernel was up and running, new tools could be prototyped quite quickly. This facilitated a very effective collaboration between the E&P experts at Norsk Hydro and the software experts at CMR. The E&P experts could provide ideas and specifications for tools supporting their specific requirements. These specifications were then elaborated in meetings with the team from CMR. The software development process consisted of frequent iterations and close collaboration between developers and the end users. The users received early prototypes they could try out, making it possible to give useful feedback to the development team. This process resulted in tools tailored to the user's data, methods and workflows.

As the project progressed, users representing different disciplines became interested in trying VR for their work processes, and the number of tools grew steadily. We developed approximately 40 tools throughout the project's lifespan. The remaining part of this section describes some of the important tools developed in relation to the identified workflows described the previous section.

4.1 Production and Well Planning

The first production work process we targeted in the VR project was well planning. With the Well Tool, the user can accurately place target points with the Wand to define a well path. Each target point contains a direction vector (tangent vector), which the user can modify. The Well Tool communicates with external well planning software, giving the user information about the well path properties such as curvature or dogleg severity.[4] When setting thresholds for the dogleg severity, the tool indicates automatically the parts of the path that does not meet the drilling constraints.

To support the well planning process, the Well Tool can be combined with other tools visualizing geophysical and geological data for guiding the placement of the control points. This includes tools for identifying target structures in the seismic data. Visualization of surfaces representing the top and bottom of the targeted reservoir body aids to the work process, and visualization of faults assists the well planning team to secure a minimum distance to the faults. Figure 5 shows users planning a well in the CAVE.

4.1.1 Well Visualization (Logs, Production History, Completion, Real Time Drilling Data)

The Well Tool also includes visualization of different log types measured along the well path, as seen in Fig. 8. This makes it possible to compare the information from the well with the seismic data and the geo-models. Images of well core samples can be displayed, attached to the correct position along the path. The tool can also be used for visualization of CSD[5] well-completion data collected from a web service, as shown in Fig. 9.

Other functionality related to wells includes the Fence Tool for visualizing different types of seismic information (seismic attributes) along the well profile. The Well Tool was later extended with visualization of real-time drilling data for monitoring the drilling of the wells. For wells in production, the Well History Tool can show the production and injection history as animated barrel symbols. The size and color of the barrels show the amount and type of production or injection. Figure 10 shows an example of the Well History Tool.

[4] Dogleg severity is the measure of how rapid the inclination and/or azimuth of a borehole changes, typically measured in degrees per 100ft or 30 m.

[5] CSD - The Completion String Design system from the software vendor Completion Services is a repository system for storing completion strings and well history tracking.

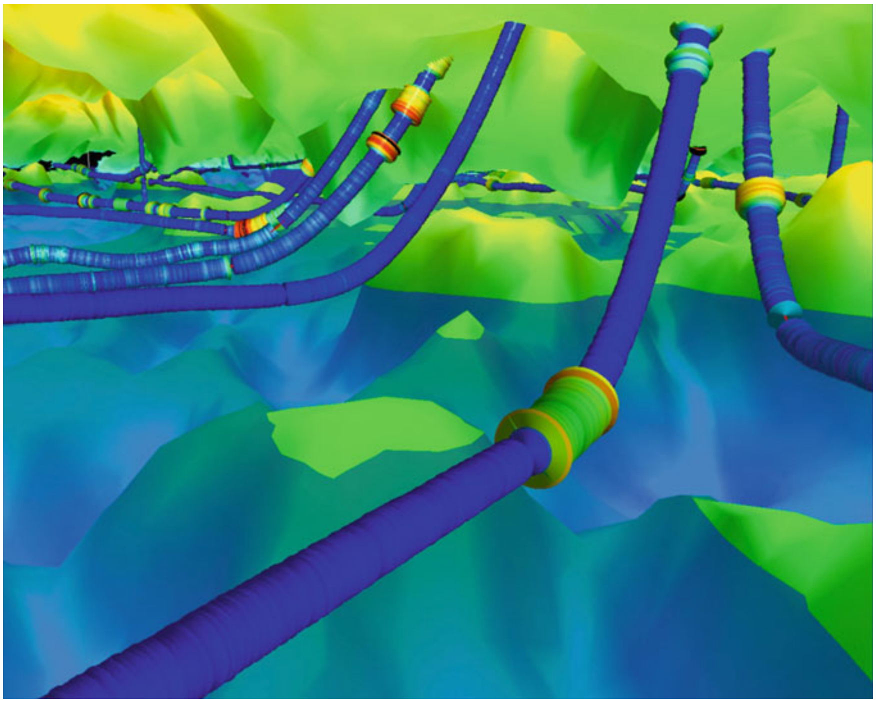


Fig. 8 This image shows several horizontal production wells in a field's reservoir zone. The measured gamma log is displayed as colored disks along each well path. Larger disk sizes, combined with warmer colors (yellow/red), represent a higher gamma value indicating shale, while smaller disk sizes, in cooler colors (blue), indicate sand

4.1.2 3D Survey Visualization—Volume Probe Visualization

The Volume Window Tool was used for visualization of 3D seismic data sets, also called seismic cubes. This tool was among the very first tools to be developed, and it was an important tool in well planning. By giving the user interactive control of the translation from data values to displayed colors and opacities (transfer function), it was possible to reveal geological patterns and structures, which were difficult to identify on seismic slices, e.g., channels and fan systems. Figure 11 shows how adjusting the transfer function revealed geological structures. This supported the well planners in positioning the target points for the well path.

At the start of the VR project, only small seismic data cubes could be displayed with sufficient frame rate because of performance limitations of the graphics hardware that was available. The Volume Window Tool was therefore designed as a probe showing only the part of the seismic data cube being inside a 3D window. The user could downsize the 3D window to obtain sufficiently high frame rate. The probe could then be moved around inside the seismic cube to explore it.

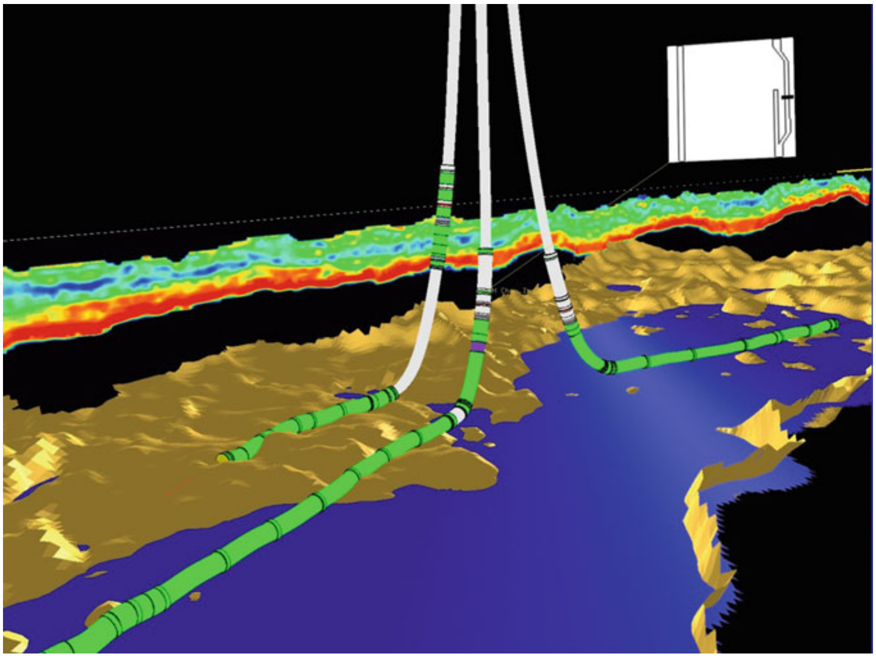


Fig. 9 The Well Tool can visualize well-completion data from the CSD (Completion String Design™) database. This image shows three wells with their CSD data mapped along the well path. The small window on the upper right hand side provides more detailed information about one of the completion components

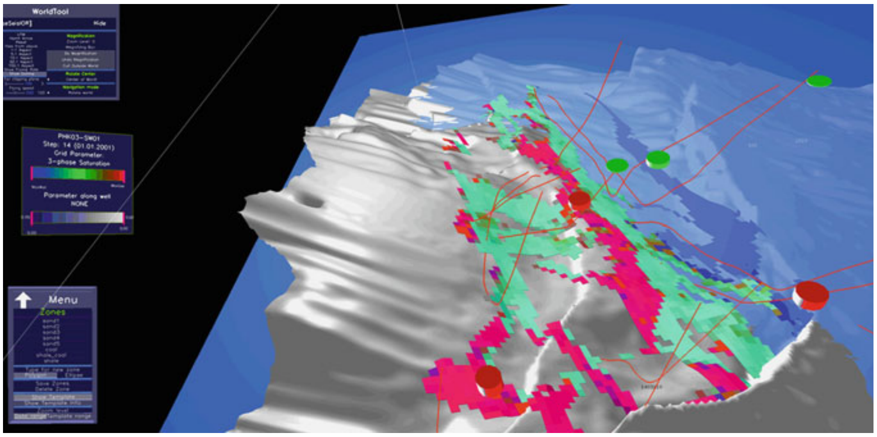


Fig. 10 The Well History Tool visualizes the production and injection history for the wells. This image shows the base reservoir as a solid grey surface and the oil water contact level (OWC) as a light blue semi-transparent surface. The pink/red/green surfaces are the result of classifications where the red color indicates sand bodies filled with gas. The barrel at the top of the well path visualizes the production history. Wells with green colored barrels produce mainly oil and wells with red barrels produce mainly gas

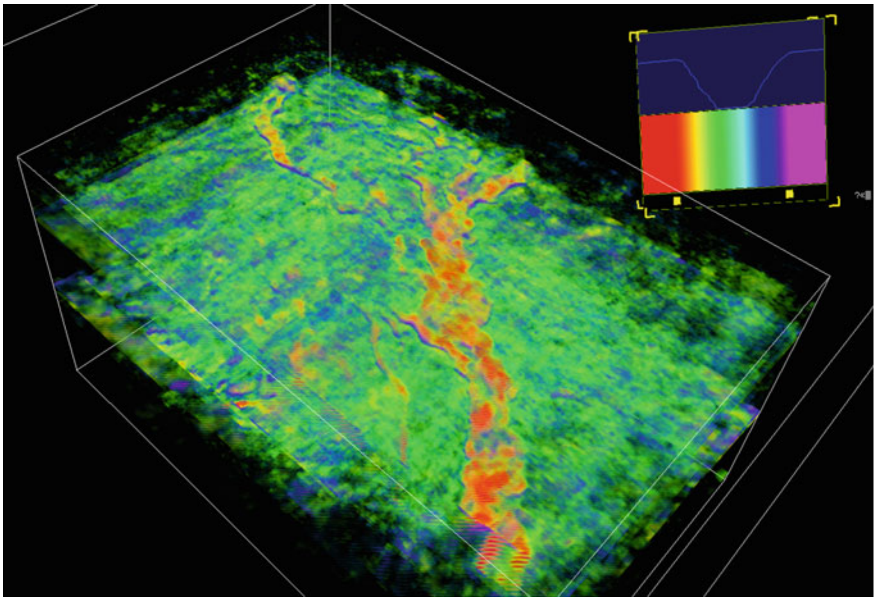


Fig. 11 Volume visualization of the same deep marine system as shown in Fig. 6 with applying the blue opacity curve drawn above the rainbow color table

Improvements in the graphics hardware performance made it possible to visualize larger seismic cubes with high frame rate. Multi-resolution techniques add to the usability for visualizing large amounts of data: Initially, a coarse version of the volume is loaded and rendered quickly. In the background, the computer calculates more and more detailed versions of the visible parts of the volume. The view is automatically updated when higher resolutions are available. This enables the user to quickly open large volume files and browse through them without having to wait for the complete volume to be loaded.

Another improvement in the Volume Window Tool was the introduction of multi-attribute visualization. The users are able to make visualizations where the resulting view is a combination of several seismic attributes. Separate transfer functions are assigned to each seismic attribute and by manipulating these transfer functions, the user can highlight different aspects of the data sets and thereby explore more subtle properties in the data. The increasing usage of 4D seismic data has made this functionality even more interesting, e.g. for studying changes in the reservoir during production.

4.1.3 Growing Tool with Surface Extraction

Volume growing is another way of exploring a seismic cube. This functionality was already available in E&P software (e.g. Paradigm™ VoxelGeo® (Paradigm Ltd.,

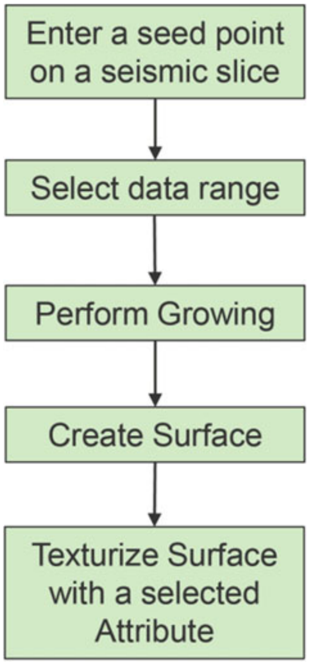

Enter a seed point on a seismic slice
Select data range
Perform Growing
Create Surface
Texturize Surface with a selected Attribute

Fig. 12 The typical workflow for the Growing Tool with surface extraction

2021)), but it was believed that the large display and 3D interaction in VR would improve the usability of the method.

A typical work session in the CAVE started with exploring the 3D seismic cube using one or more Volume Window Tools. By adjusting the transfer functions, the user may see structures (s)he wants to explore more closely. To try to isolate e.g. a possible channel system, he then starts the Growing Tool. He marks a seed point inside the interesting volume using the Wand. The user can then specify the thresholds for the growing process based on the actual data values for the selected seed point. The growing process results in objects representing connected voxels[6] in the cube.

An important success criterion for this tool is the high degree of interactivity. The user can control the connectivity (how many neighbors to check for each voxel) and the data thresholds limiting the vertical and horizontal connectivity. If the growing process fails, e.g. by creating too large bodies, the growing process can easily be undone to the point where the process failed and the thresholds adjusted before restarting the growing process. It is also possible to specify additional seed points. This way the user can end up with one or several bodies representing e.g. a complex channel system.

We also added functionality to the Growing Tool for surface extraction based on the grown volumes (min, max, top or base surface). This is useful e.g. for generating input to geo-models. Figure 12 shows the typical workflow for this operation. The

[6] A **voxel** is a volume element, representing a value on a regular grid in three dimensional space.

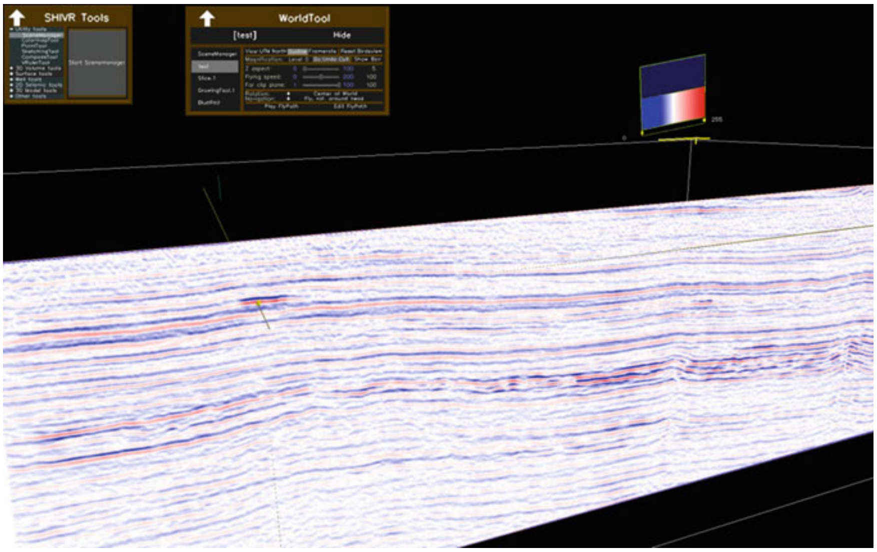


Fig. 13 A seed point is set in a positive (red) anomaly

first step is to set a seed point on a seismic slice (see Fig. 13). Then the user sets the data range for the threshold amplitude values and performs the growing. The result of the growing is a point-cloud where every point is connected to the initial seed point (see Fig. 14). The user selects the surface type she wants to extract and the system creates a surface of the grown points. Figure 15 shows a surface extracted based on the maximum seismic reflection value for each z-value (depth) in the grid. In this example, the extracted surface reveals a paleo channel system with an associated splay system to the left. The final step is to map a selected seismic attribute onto the new surface. In Fig. 16, the Seismic Reflection attribute is applied as the texture map on the extracted channel surface. From this example, one could interpret the red colored values to represent the best porosity area, which would represent the best reservoir quality.

As with the Volume Window Tool, we extended the Growing Tool to handle multi attribute data sets. The user can control the growing algorithm by specifying threshold values for a set of attributes. This enables improved segmentation of bodies in the data. These bodies can later be used as targets for well planning. Functionality for calculating volumes of the grown bodies gives valuable input to the geo- and reservoir-modeling.

4.1.4 Cross Plot Tool

The Cross Plot Tool is a tool for cross plotting two or four attributes, and for performing classification based on zones defined by the user. As for the Growing Tool,

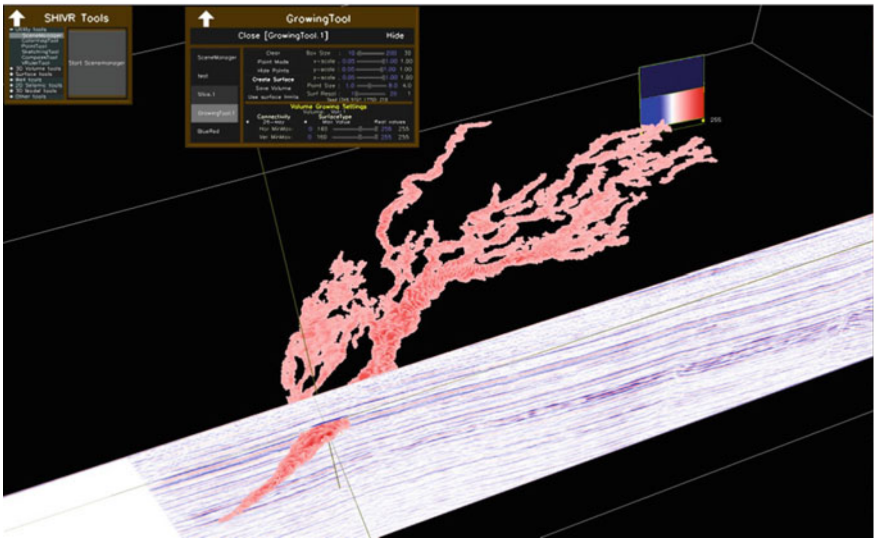


Fig. 14 The result of the growing is a cloud of red points, where each point is connected to the initial seed point. In this example, the selected thresholds for the growing are amplitude values 160 and 255

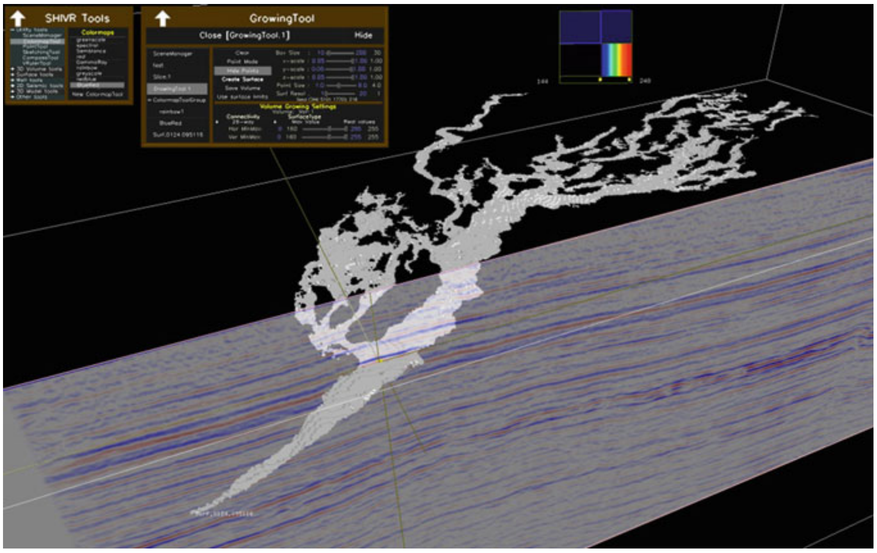


Fig. 15 The user has set "Surface Type" to "Max Value" in the Growing Tool GUI and SHIVR extracts a surface based on the maximum seismic reflection value for every z-value (depth). The extracted surface reveals a paleo channel system with an associated splay system to the left. The seismic has been given a semi-transparent appearance to better show the extracted surface

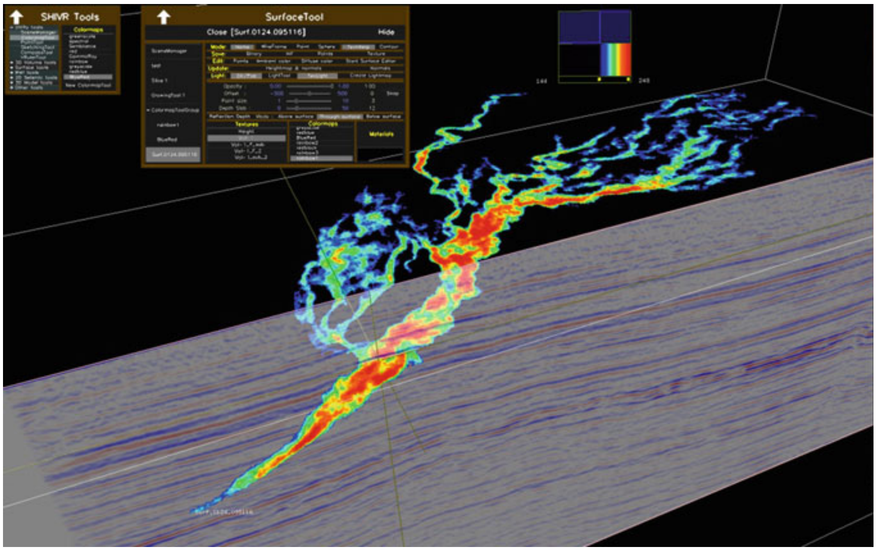


Fig. 16 From the Surface Tool GUI, the user has now applied a texture on the generated surface. The texture is the Seismic Reflection attribute. The color map selected for this attribute texture is compressed around the positive values, showing the high positive attribute values as red and low positive values as blue. In this example, one could interpret the red to represent the best porosity areas, which would represent the best reservoir quality

the basic methods of the Cross Plot Tool are also available in other E&P software. It is also in this case the high degree of interactivity, which makes the VR version an interesting alternative to the commercially available versions. The algorithm is implemented on the GPU,[7] making it fully interactive.

The first version of the Cross Plot Tool handled two attributes. This can e.g. be acoustic impedance and the ratio between the shear and pressure wave velocities. A 2D scatter-plot (cross plot) of the two attributes is displayed in the CAVE. A volume probe can be positioned and scaled inside the seismic cube. In the cross plot, points are displayed in every position representing attribute values that are present inside the probe. The user can get a picture of how the two attributes related to each other in different parts of the volume by moving the probe through the data volume. Figure 17 shows how the CrossPlot Tool is utilized to isolate geological bodies.

Different areas of the cross plot represent different rock properties. By correlating the data with e.g. findings in exploration wells, the user can draw polygons or zones in the cross plots representing different rock types. In the 3D volume, the voxels representing the zones are displayed with corresponding color. By interactively moving and reshaping the polygons while seeing the resulting volumetric bodies in 3D,

[7] GPU—Graphics Processing Unit is a specialized electronic circuit in computers and other technical equipment to handle and manipulate the graphics output.

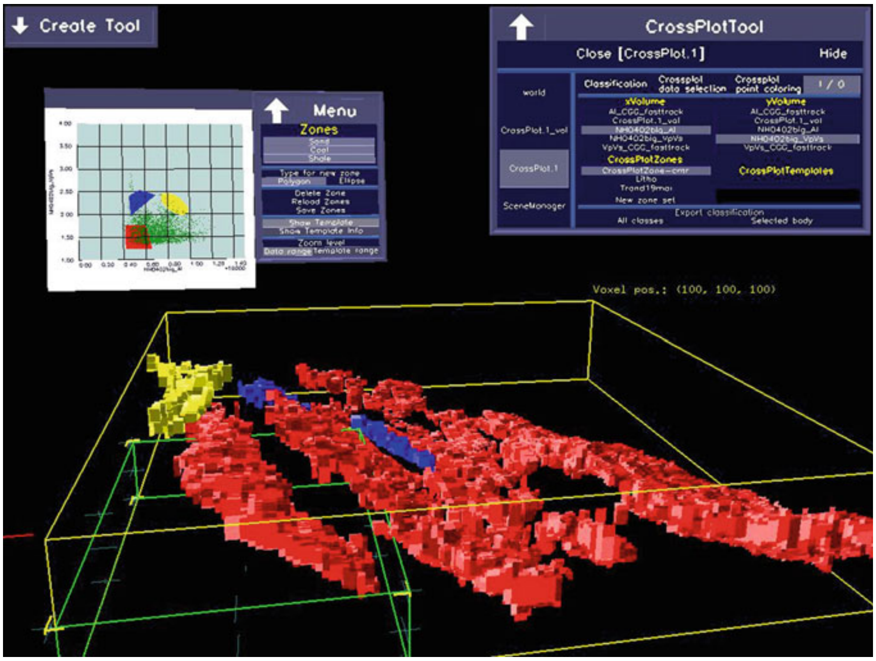


Fig. 17 The Cross Plot Tool is utilized to explore the correlation of two or four different volumetric parameters. In this image, acoustic impedance is mapped against Vp/Vs in a 2D cross-plot. The user draws polygons in the cross-plot and the corresponding voxels are rendered in the volume rendering tool. The polygons drawn in this image is nicely segmenting the geological bodies of this dataset

the user can perform hypothesis testing in real time, making this a useful tool to understand the geology.

The Cross Plot Tool was extended with a dual mode for classification based on four volumes in two cross plots. This improves the possibilities for exploring hypotheses about where oil-filled sands are located, and for studying changes in the reservoir during production. It can e.g. be used for localizing areas where the geological model contains properties which are not compatible with changes or absence of changes in the 4D seismic data. The analysis can be used to update the geologic model.

4.1.5 Model Visualization (Geo-models, Reservoir Simulations)

At an early stage it became clear there was a potential added value if one could visualize in the CAVE the geo-models and reservoir simulations together with the seismic data and well data. We added several new tools to handle these data types.

The Surface Tool is used for visualizing horizons and faults. It has support for texturing the surfaces with height map data and images. It is also possible to texture

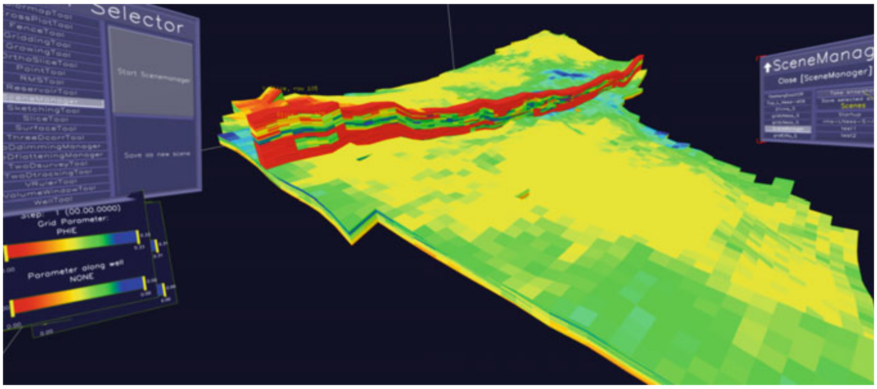


Fig. 18 An RMS-model with the porosity parameter visualized

the surfaces with the seismic data it intersects. In the same way that hardware performance had to be considered in the original design of the Volume Window Tool, existing hardware at the time the Surface Tool was designed put constraints on the complexity of surfaces that could be displayed with sufficient frame rate. To overcome this problem, we added support for Continuous Level of Detail (CLOD) [15]. This means that the software continuously monitors the frame rate and adapts the resolution of the displayed surface to maintain a sufficient frame rate. We utilized a software library developed by Kongsberg SIM.[8] SHIVR offers functionality for adjusting surfaces vertically, e.g. to match the surface to findings in well logs. There is also some simple surface editing functionality.

The Reservoir Tool visualizes RMS™ reservoir models and Eclipse™ reservoir simulation results, see an example in Fig. 18. The tool offers extensive functionality for adapting the visualization of the geo-model and reservoir data to user specific requirements, including box and polygon clipping, and clipping based on parameter value range. The tool can also extract iso-surfaces and grid aligned slices for irregular grids, and the results from the reservoir simulations can be visualized as animations. The possibility to co-visualize and compare the geo-models and reservoir simulation results with other data types, such as seismic data, was an important motivation for developing the Reservoir Tool.

4.2 Exploration Reconnaissance

Several of the tools described for production and well planning have also been important for exploration reconnaissance when 3D seismic data is available, especially the Volume Window Tool, the Growing Tool and the Cross Plot Tool. In the following, we will only be focusing on specific tools developed for exploring 2D seismic.

[8] Kongsberg SIM is a company in the Kongsberg Group.

4.2.1 2D Survey Visualization

The seismic data available in an early exploration phase is generally limited to 2D seismic surveys. These lines are often quite long, generating a grid that covers the area to investigate. It can be challenging for the geophysicist to build a mental overview of the various possible geologic interpretations, especially compared to a dense 3D seismic survey. The large-screen, stereo visualization system SHIVR is an ideal tool for exploring such data, because one can easily obtain an overview and at the same time observe the seismic details.

A survey base map is the easiest way to get an overview of one (or several) 2D seismic surveys. The base map is, in many petroleum software tools, visualized in a separate window, thus the user needs to switch the attention back and forth between the seismic data window and the base map window. In SHIVR we chose to implement an integrated view-visualization, where we merged the base map with the 2D seismic data in one common 3D visualization (see Fig. 19). The user can be immersed in the seismic data and at the same time, be aware of the surrounding seismic lines and toggle them on and off without losing the focus on the seismic data.

In SHIVR, as with other seismic visualizations, the user specifies a color map that maps the seismic amplitude values to a grayscale or color representation. The 3D co-visualization environment in SHIVR support visualization of other data types together with the 2D seismic survey. Well paths, with well logs, and interpreted horizons are examples of such co-visualized data types. Visualizing the 2D seismic lines in a large screen 3D environment also opened up for creativity in the visualization and interaction with the data. We observed that the users, instead of studying the seismic line as a vertical profile, many times placed the section flat down. Figure 20 shows an example of this. The perspective horizontal tilt of the seismic was not possible in other petroleum software at the time and it resembled the way the geophysicist studied long 2D seismic paper sections before the introduction of computers in seismic interpretation.

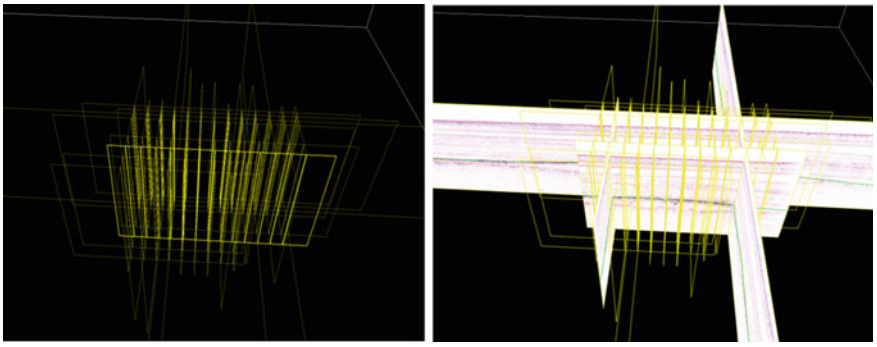


Fig. 19 Left image shows the seismic base map displayed in SHIVR. In the right image, the seismic lines are turned on

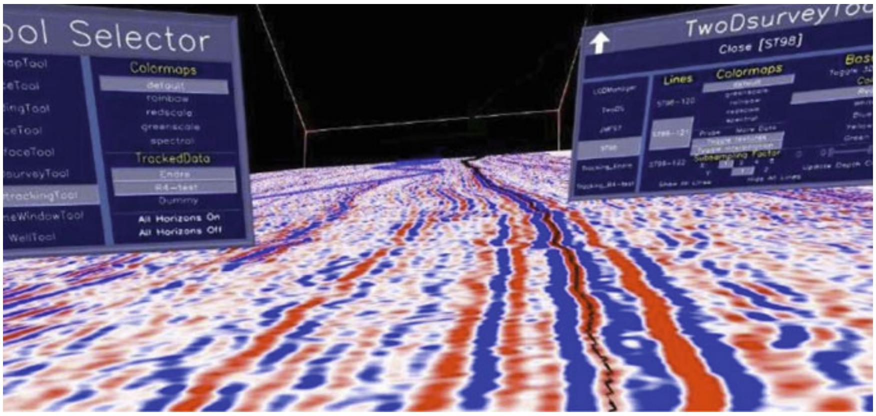


Fig. 20 A tilted view of the 2D seismic line with one tracked horizon (black line)

For seismic surveys shot onshore, there is also usually a digital elevation model (DEM) available. Co-visualizing this DEM with the survey base map is advantageous because the elevation model can give an understanding of the subsurface. Since the DEM will occlude the seismic lines in a naive co-visualization, the user was given the option to give the DEM surface a negative offset so the seismic lines in the base map became visible. The relationship between the DEM and the seismic became easier to perceive.

Functionality for emphasizing the interesting zones in the seismic was implemented. This method is called dimming. The purpose is to highlight important parts of the 2D seismic data to enhance the cognitive focus of the users. In SHIVR the user can assign different color maps to different sections of the seismic line, for instance apply a red-white-blue color map in a possible reservoir zone and a grayscale color map for the less interesting sections. The user can also make the less interesting sections semi-transparent to further hide it, hence thereof the name dimming. Figure 21 shows an example of this.

4.2.2 2D Seismic Interpretation in SHIVR

As seen, SHIVR's large-screen display is well suited for visualizing and inspecting 2D seismic lines. Therefore, the users soon requested methods for quick and easy interpretation. We build a SHIVR tool where the experts can track horizons directly onto the 2D seismic lines. The tool provides the user with different interpretation possibilities, from manually drawing the horizon onto the seismic line, to full automatic tracking based on user-defined seed-points and tracking parameters. The different modes can be combined on the same tracked horizon in such a way that the last manually tracked point becomes the seed-point for auto-tracking. The user runs

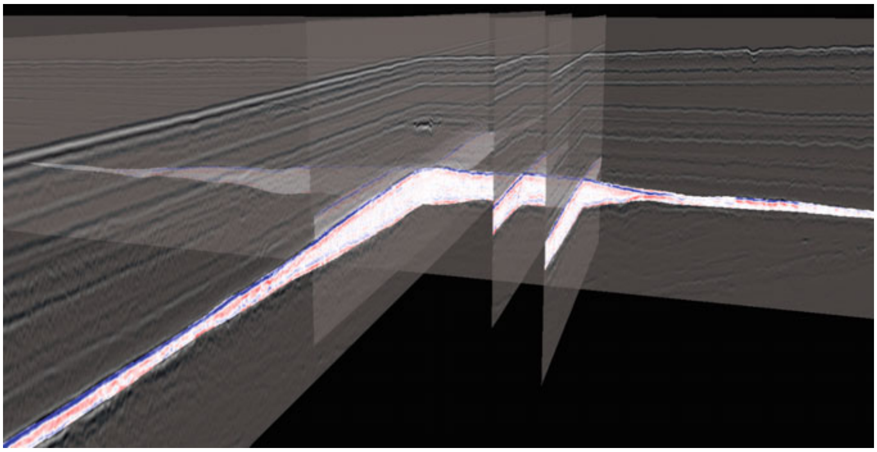


Fig. 21 Selected lines from a 2D survey using a blue-red color map in the reservoir zone and a semi-transparent greyscale in the non-interesting zone

the auto-tracking in reverse or uses an eraser-mode to remove unwanted tracking. Figure 22 shows the tracking of a horizon.

A successful invention we made was to couple the auto-tracking with the navigation in SHIVR, letting the user fly along the 2D seismic line while tracking. This coupling makes it easier for the interpreter to validate the auto-tracking, as she does not have to move her head to follow the tracking. The flying works for both vertically aligned and horizontally tilted seismic lines. A tracking ribbon that can show the depth values or the seismic values along the tracked line can also be used to enhance the tracked event.

When exporting the generated interpretation, we chose to use a file format frequently used in the oil and gas industry. This ensured that the data type produced in SHIVR could be read and studied in other E&P software. Using a standardized file format also simplified the import of 2D seismic interpretation into SHIVR.

We also added flattening functionality for 2D seismic lines. If one assumes that a surface was flat at the time of deposition, it helps the interpreter to view the underlying seismic and horizons relative to this horizon. The functionality is common in all standard interpretation software and is called *flattening*. Even though it is not purely geological accurate; for instance, it ignores compaction of shale rock, it still gives valuable understanding of the geologic processes and the underlying topography at a given geological time. Figure 23 shows the effects of flattening. The lower, green surface in the top figure is flattened and the resulting flat surface is shown in blue color in the bottom figure. Please notice how the 2D seismic sections are changed after the flattening.

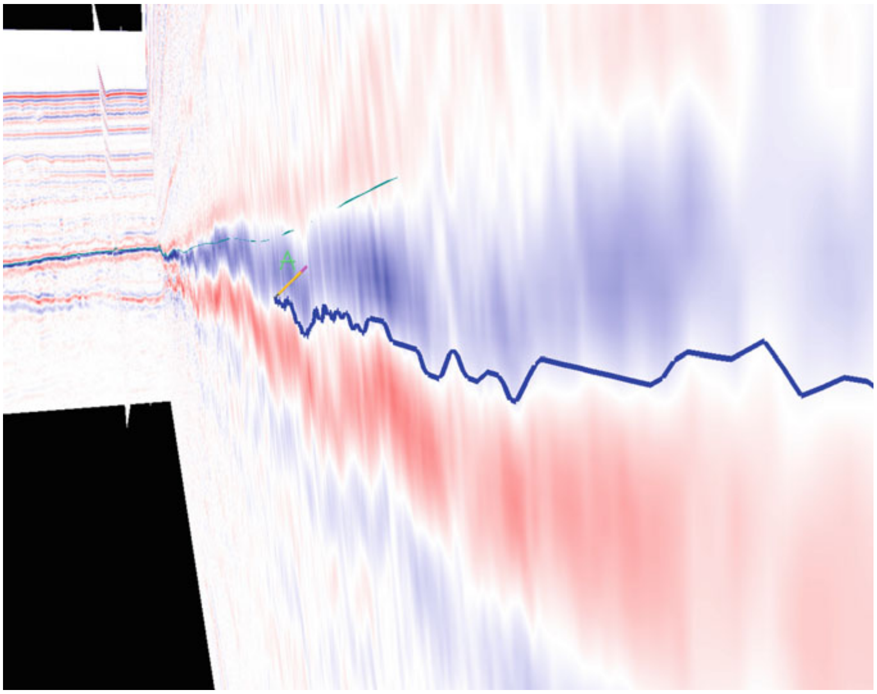


Fig. 22 Automatic tracking, following a zero-crossing event. While the auto-interpretation proceeds, the distance and position to the pen is constant giving a sensation of flying along the line

4.3 Remote Collaboration

Inspired by the successful interdisciplinary well planning work done in SHIVR, we wanted to take the interdisciplinary collaboration one step further. We recognized that sometimes one or more of the experts needed in the interdisciplinary work sessions were located at different physical locations. That meant that they had to fly in or had to participate over phone conference. Norsk Hydro had by now installed large-screen displays in several of their office locations. We extended SHIVR with support for remote collaboration where several large-screen installations can participate in one VR work session. To facilitate the feeling of close cooperation we use avatars, i.e., virtual characters, to represent the remote user's actions and movements. In this way, the others can see where a remote participant is looking and pointing, see an example of this in Fig. 24. We based our solution on a software library called CAVERNsoft developed by University of Illinois, Chicago [8].

Successful remote collaborative work sessions were executed between offices in Bergen and Oslo, Norway and Houston, Texas. These work sessions were proof of concept showing that the international offices could access experts in the main offices to efficiently discuss problems and challenges without having to travel.

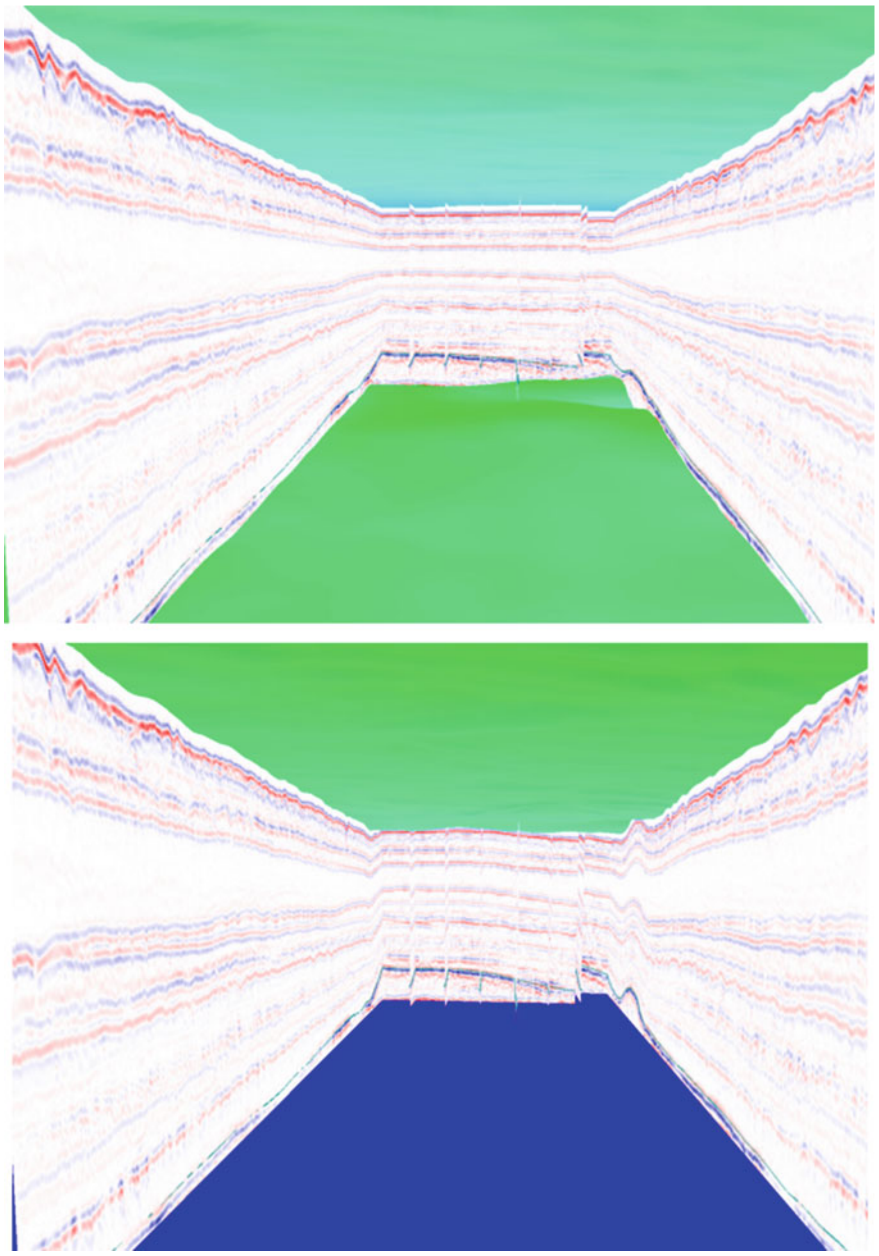


Fig. 23 The lower interpreted surface (see top picture) has been flattened (see bottom picture). The seismic data, interpretations and other surfaces are affected correspondingly in the process

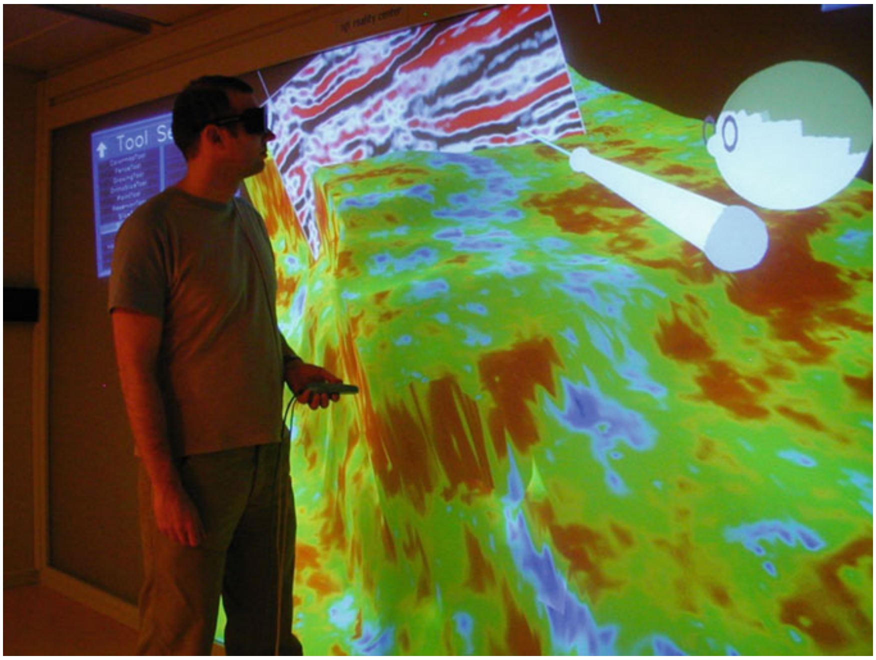


Fig. 24 A collaborative work session. The avatar represents the remote participant

4.4 Importing Data into SHIVR

So far, we have described some of the tools we developed and how they enhanced, and in many cases redefined, the E&P work procedures. To complete the picture, we wish to give a brief description of the third main component of the SHIVR architecture, the data management modules.

Most of the data types are imported into SHIVR through data files. Each data type has a great variety of available file formats, mainly because each petroleum software vendor tends to define their own proprietary file formats. Therefore, we developed a large number of different file readers, e.g. SHIVR supports 18 different file formats for importing interpretation. A substantial part of the development time has been spent on decoding proprietary file formats in order to be able to import the data into SHIVR, as few programming libraries for the file formats are available. Generally, data files on text format are preferred over binary files, as the text files are easier to interpret.

A challenge of getting the imported data types co-visualized is to have them all defined in a common coordinate system. For SHIVR we have chosen the UTM (Universal Transverse Mercator) coordinate system. This is a 2D Cartesian coordinate system making it possible to represent the location of any point on the earth. The earth is divided into a number of grid zones, so all data types in a dataset must be defined in the same UTM grid zone to be co-visualized correctly in SHIVR.

5 Integrating the VR Application with the Company's Software Portfolio

In the first phase of the project, only limited functionality was developed for preparing the data to be used in the VR sessions. Because of this, the user had to prepare the data sets manually by copying the required files into SHIVR specific directory structures and manually edit SHIVR startup and configuration files.

After VR had proven a success, the number of SHIVR users increased. All the manual setup-tasks became a severe obstacle for the average user, preventing a broad implementation of SHIVR in the organization. To mitigate this situation, a simple graphical user interface for semi-automatic writing of configuration files was developed. In addition, we started to integrate data from the portfolio of software tools and solutions already used in the company. We have used three different strategies to bridge the gap between SHIVR and the other relevant software tools, and the data generated by these tools (also illustrated in Fig. 25):

1. Connect SHIVR to another running application and obtain the data via message passing between the applications.
2. Connect SHIVR directly to a data server/repository through a defined interface.
3. Export data from other applications on formats that SHIVR can read.

We will exemplify these approaches below.

5.1 Connecting to Another Running Application

Simulators are the typical example of software applications SHIVR connects to using this approach. The simulators extend SHIVR with functionality needed for completing work processes in VR. We have usually worked with the simulator developers to extend the simulators with a SHIVR interface.

5.1.1 Dogleg Severity Calculations During Well Planning

The well planning work process in SHIVR was the first process to utilize this type of application-to-application communication. As described in the previous section, the Well Tool communicates with external well planning software, e.g. the RMS Wellplan application developed by Roxar/Emerson. When the user adds a new well-path target point, it is sent to this external software tool. The well planning software then calculates and returns to SHIVR, in real-time, well path parameters, such as

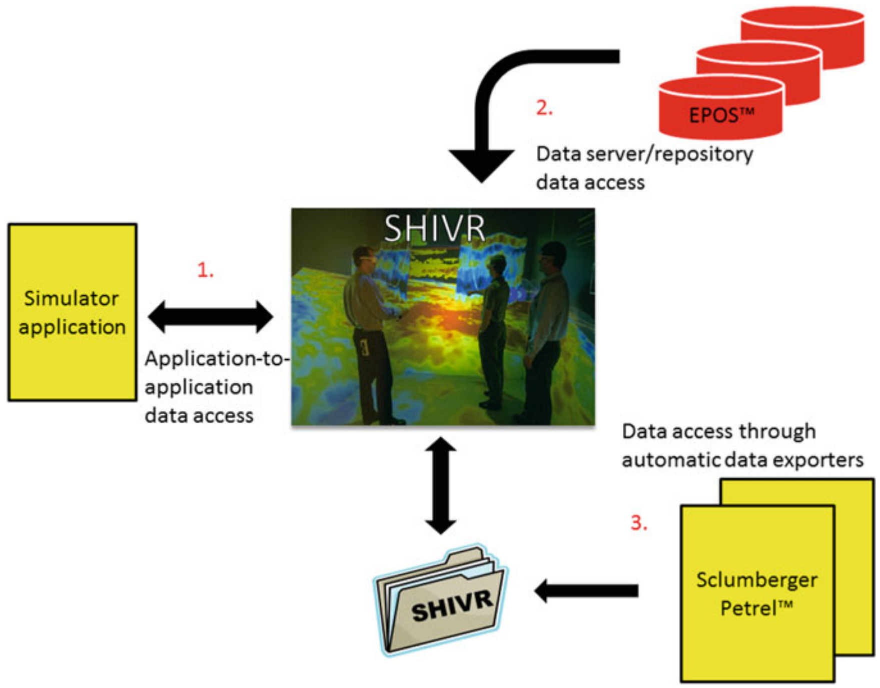


Fig. 25 The three data access methods utilized by SHIVR

curvature and dogleg severity. SHIVR can then display these parameters along the well path, to guide the user in the planning process.

5.1.2 Migration Simulation—Control and Visualization in SHIVR

MPath™ was a petroleum migration simulator developed by the Permedia Research Group (bought by Landmark/Halliburton) [16]. As with other 3D data, the migration simulation results were well suited for visualization in SHIVR. Seed points for the simulation could be set in SHIVR and MPath returned incremental results in real-time as they were calculated. In this way, the user could cancel a simulation if the seed point was put in a wrong spot without having to wait for the simulation to complete.

To realize this application-to-application communication, the MPath developers wrote a small network module that linked MPath and SHIVR. Since the communication is based on network communication, neither this module nor MPath had to run on the same computer as SHIVR. This also meant that the simulator could run on a different operating system than the SHIVR installation.

5.2 *Connecting to a Data Repository*

A data repository is a system for storing and managing datasets. A wide range of technical solutions is available for building these repositories. In SHIVR, we have built support for several such repositories.

5.2.1 Data Sharing Repository Systems

Petroleum software vendors have spent substantial effort on building systems for storing, managing and accessing E&P data. These systems enable the users to decide for themselves what work procedures and software tools they will use. The Epos™ system from Paradigm (EPOS Paradigm Ltd., 2021) is one such repository system that SHIVR can connect to. Paradigm provides a programming toolkit that contains the necessary functionality for accessing the data in Epos. We used this toolkit to extend SHIVR with an Epos data-reader plugin. This plugin can access data from an Epos data repository located anywhere within the network. From an end-user's perspective, this data access happens automatically. The data types are available in the SHIVR menus just as if they were located on the local disk.

5.2.2 Web Service Systems

Another example of data share technology is through *web services*. This is data sharing between computers using the Internet communication protocols. The Completion String Design (CSD) system from the software vendor Completion Services is a repository system for storing completion strings and well history tracking. The system shares this data through a web service. As for the Epos system, accessing the data is automatic and transparent for the end-user, as long as SHIVR is configured with the Internet address to the web service.

5.3 *Import Data from Other E&P Applications*

Many relevant data sets from proprietary software applications (source applications) cannot be accessed through an open interface. The third interfacing strategy implemented in SHIVR is to automate the copying of data files and creation of configuration files in situations like this. It should be noted though that there are some drawbacks associated with this approach because of the data duplication. Firstly, it wastes disk space. More importantly, it also represents a data management challenge if the data are modified either during a SHIVR work session or in the source application. Thus, we recommend this approach for data that are only visualized and not modified.

5.3.1 Utilizing a Preprocessing Program

One variant of this approach is to implement preprocessing programs. One such program was created, in cooperation with Simula Research, for accessing datasets from the PetroMod™ application from Schlumberger. The end-user specifies the file location of the PetroMod projects and the program copies the data files to the SHIVR specific file directory structure and generates the necessary configuration files automatically. The user has to run the preprocessing program every time someone modifies the data in PetroMod.

5.3.2 Export Plugins in Third-Party Software Tools

If the source application allows it, one can build an export plugin that prepares the SHIVR data directly from within the source application. The Petrel E&P Software Platform by Schlumberger supports this through the Ocean plugin system, and Statoil implemented a Petrel plugin for extracting data on files readable from SHIVR.

6 Success Criteria and Lessons Learned

In this section, we have summarized important criteria for the success of the VR research project and some lessons learned.

6.1 Main Benefits Achieved Through Use of VR and Large Screen Visualization

Based on the experiences from the two workflows well planning and exploration reconnaissance, the main benefits achieved through use of VR and large screen visualization are:

- Co-visualization of cross-disciplinary data in the same virtual world gives improved understanding of constraints, challenges, goals and opportunities across the disciplines.
- Stereoscopic large screen visualization gives increased 3D understanding of spatial data.
- Tracking of the position and orientation of the user's head and wand, combined with stereo, gives improved spatial control and manipulation.

These benefits were expected based on experiences from other industries. However, we can add the following additional important benefits:

- Enhanced cross-disciplinary collaboration and increased efficiency: As an example, we have described how gathering the various disciplines involved in well planning processes in the VR room and co-visualizing all the necessary data in one virtual world gave a substantial acceleration of the well path planning.
- Increased quality: We have also seen that the quality of the well paths planned in the VR improved considerably compared to well paths planned in traditional work flows for the same oil field.
- Increased creativity: We have noted that the combination of large screen visualization and cross-disciplinary collaboration enhanced the creativity in teams solving complex problems.

6.2 Risk Willing Management

The "blue sky" research project structure (high risk/value), where seed funds were (and are) available, was an important enabler for the project. Initiating this particular research project also depended on key managers in the research organization that were willing to take the risk when the large investments for the VR infrastructure and graphics compute resources were proposed. The implementation of the VR application in the company and the commercialization of the technology have proven to return this investment many times over.

6.3 Learning from Other Industries

It was important for the project's early success to learn from other industries. Choosing to explore the use of VR-technology turned out to be the right choice at the right time. Additionally, acquiring the source code for the VR software for biological applications developed by NCSA gave the project a flying start.

6.4 Close Collaboration with Developers and User Involvement

Geographical proximity between the project teams at Statoil and CMR ensured effective collaboration. Since CMR's knowledge within geology, geophysics and, the O&G work processes was limited, it was important with frequent iterations and close collaboration between the two teams. The Statoil team received early prototypes they could try out, making it possible to give useful feedback to the CMR team. The results were tools tailored to the user's data, methods and workflows. User involvement during the specification and development phase guaranteed that the users took ownership in the VR software when it was implemented in the organization.

6.5 Use of Pilot Operators

Since the CAVE-installation with following equipment had a high degree of complexity, dedicated SHIVR “pilots” from the research team were available for assistance during the VR sessions. These pilots acquired important knowledge of how to operate SHIVR in an optimal way and were able to transfer this knowhow between the projects. During the VR sessions, the pilot also gathered feedback and comments from the teams. This was useful input to the software specification and development work.

6.6 Integration of the VR Application with Existing Software Portfolio

To achieve as broad as possible implementation of SHIVR in the organization, it has been important to integrate the product with existing software portfolio in Statoil. It must be simple to transport data easily and quickly between SHIVR and the other relevant software applications. The interfaces need to be maintained continuously because of regular software updates.

We have spent significant time developing importers for the various data types needed by the tools in SHIVR. In many cases, we have had to reverse-engineer proprietary file formats with little or no file data descriptions. Many file formats are ambiguously defined so it is generally not possible, or desirable from a cost–benefit point of view, to support all variations. A lesson learned is that additional development time and resources often had to be allocated to improve a file reader due to the end-users need to import a new file format or a variation of an existing file format.

6.7 The Difficult Balance Between Development of New Functionality and Bug-Fixing

To get the most out of the project budgets, focus was primarily on developing new functionality. Only the most serious bugs were prioritized. The advantage of this approach was that we were able to develop and test many good ideas, but the list of minor bugs grew long. When the users started utilizing the VR technology in their daily E&P work processes, a task force was initiated to stabilize the application and improve user friendliness. We believe that more focus on bug fixing from the start of the project would have resulted in lower total costs for maintenance of the VR application over the project period. However, finding the right balance is not trivial.

6.8 Remote Collaboration—Great Potential but Low Usage

Despite of the successful proof of concept of remote collaboration, this functionality was not used much. We believe the main reasons were:

- We overestimated the need for collaborative sessions for accessing research expertise. Apart from the exploration environment in Oslo, the number of external VR-installations outside of Bergen was limited. Since the Oslo office had the necessary exploration expertise locally, the need for connection to Bergen was only on rare occasions.
- Setting up a collaborative session is cumbersome and time consuming. All data for the work sessions has to be duplicated on all collaborating sites. This design was necessary to cope with limited network bandwidth. The consequence of time-consuming preparations is that ad-hoc collaborative VR-sessions were difficult to perform.

6.9 Adapting to Major Hardware Developments During the Project Period

There has been a tremendous development in graphics hardware performance since this project started. Today's commodity systems outperform the graphics capabilities of the original special-purpose graphics supercomputer manyfold, at a fraction of the price. We succeeded to a certain degree in adapting SHIVR to the new technologies, partly because of the modular design of SHIVR where the interface to the graphics hardware is isolated to a limited part of the system kernel. However, it was a challenge to utilize all new features and possibilities offered by new hardware.

Additionally, new display systems consisting of one large screen became available. Such systems gave sufficient sense of immersion in the 3D data even though there was only one display wall and no projected sidewalls or floor as in the CAVE. Advantages with these display systems compared to the CAVE included room for more people, possibility for more light intensive projectors, and significantly lower costs. Statoil installed a number of such VR rooms. These displays were driven by computers equipped with high-end graphics hardware and running Linux or Microsoft® Windows®. Since the display system configuration in SHIVR was taken care of by a third-party library, adapting to different display configurations was unproblematic. The result was that more users in Statoil had access to running SHIVR in VR. Today, only one such room is operation.

6.10 Increasing Request for a Desktop Version of the VR Software

During the project period, there was an increasing request for a desktop version of SHIVR. The main reason is that SHIVR has useful functionality, which does not require running in a VR environment, namely its functionality for co-visualizing and comparing data from different disciplines. We made some adaptations of the user interface making it possible to run SHIVR on a desktop instead of in a VR environment. One lesson learned is that it is very difficult to design one single user interface that is fitted for both desktop and VR. The desktop version of SHIVR is somewhat clumsy to use compared to standard desktop applications.

6.11 Commercialization

SHIVR was commercialized in 2000 as Inside Reality. Schlumberger later purchased the commercial rights, and incorporated the VR technology into the company's E&P software portfolio. For Norsk Hydro (at that time) the VR software became available in a commercial software package, with improved stability and maintenance as a major advantage.

However, Norsk Hydro recognized that with the commercial version, they were restricted in implementing new ideas and new functionality. Since there was a wish to continue their research and explore new VR functionality, this work had to be performed on the internal VR-application. Even though Schlumberger had the commercial rights for the technology, Norsk Hydro and CMR secured the rights to continue their development. Such a continuation was extremely valuable for the VR project and for other research projects. New ideas and improvements could still be implemented and tested.

6.12 The Value of VR Technology for Branding

SHIVR and its VR-technology was successfully used for promoting Norsk Hydro and Statoil as a high-technology company. For a long time, also after the project finished, the technology was used in large conference exhibitions as a visual showcase for important results. The use of a large 3D screen at the company's booth always attracted delegates. The company received many visitors, and a demo in the VR room was a standard ingredient in the guided tours.

7 From CAVE to Head Mounted Displays

The CAVE and the VR walls were space-consuming, difficult to operate and required maintenance. Also, other commercial geoscientific software started to feature visualization. Therefore, CAVEs and VR walls slowly disappeared in oil & gas corporations.

Then, the head mounted displays (HMD) made their entry into the consumer market. They were affordable, easy to setup and easy to use. Compared to a CAVE setup which cost somewhere between two to four hundred thousand Euro, an HMD costs around one thousand Euro, which is two to four hundred times less.

There are many different sources of spatial data within Statoil (Equinor) stored in different locations. There was an interest in gathering all these different data types and visualizing them together. Motivated by this, The SHIVR code was ported to run on an HTC Vive Pro 101 HMD [6]. The porting was performed by an internal team that had no prior experience with the code base consisting of around 400.000 lines of C++. They stated that the code was of good quality and that it therefore did not take much time to port it.

The basic concept of running a CAVE and an HMD is the same as they both are display technologies. The position and orientation of a tracked head is used to render a scene as it would be seen from a user's left and right eye. The two renderings are then displayed on the corresponding eyes. In a CAVE, this is achieved using active or passive 3D glasses. For HMD's, each eye sees a separate screen that the rendering is displayed on, and a lens is used to move the focus point so that the image seems to be farther away.

7.1 Experiences Using SHIVR in HMD

Users running SHIVR in an HMD stated that it felt more realistic compared to the previous VR solution and that it gave a total feeling of immersion in the data. However, when moving quickly around in the VR world, many users experienced motion sickness. Also, some users got headaches after wearing the HMD for more than a few minutes. Instead of using a mode of movement where the user moves/flies in the direction he/she is pointing, a mode where the user must grab onto points in the scene and drag the scene closer was used. This reduced motion sickness but was only practically useful for moving around short distances.

Trying to overcome these physical side effects from a programmer's perspective are challenging. They are different from technical and well-defined problems such as fixing bugs and adding new functionality. The HMD problems are individual and may not be experienced by the developer who is trying to reproduce and address them. This may be because often, the more times one is exposed to VR, the more one gets used to it and experiences fewer side effects. Therefore, one is dependent on assessing people's reactions to changes that are supposed to reduce motion sickness and headache.

There is a lot of active research on how to overcome these side effects [13] [14]. This includes software adaptions such as limiting the user's ability to perform changes in movement (acceleration), and hardware upgrades such as increasing the field of view or increasing the update frequency of the display.

8 Conclusion

In this chapter, we have presented a VR research project that contributed to a paradigm shift in the work processes in the petroleum industry. The targeted work processes were production, especially well planning, and exploration reconnaissance. Software tools supporting these work processes in VR were developed and tested.

The goal of the project was to radically improve the critical operations within E&P, increase quality, and improve cost efficiency. The project succeeded in fulfilling this goal. The most important results were increased recovery and more efficient work processes.

We have shared important experiences and lessons learned from the project. We would especially mention the enhanced cross-disciplinary collaboration. Gathering the various disciplines involved in a work process in the VR room and co-visualizing all the necessary data gave a substantial increase in efficiency. The results from work processes performed in VR improved considerably compared to traditional workflows, and the combination of large screen visualization and cross-disciplinary collaboration enhanced the creativity in teams solving complex problems.

After 12 years of development, the VR research project concluded in 2008. The resulting software was operationalized, and became part of Statoil's software portfolio. Statoil and CMR continued to collaborate on a maintenance project to keep the software operational. Minor developments were also carried out.

Acknowledgements We would like to thank all involved people in the management and in the project teams at Norsk Hydro, Statoil (now Equinor) and CMR (now NORCE Research). Especially we emphasize the roles of Mons Midttun, Christopher Giertsen, Rolf Helland, Gunnar Halvorsen, Tyge Løvset and John Pixton in making this a successful project. We would also like to thank SGI (which ceased existing in 2009) for technical advice and the support in the first phase of the project.

Appendix

Below is a list, in alphabetical order, of the tools in the current version of SHIVR.

1. ColormapTool: Edits transfer functions translating values to color and opacity.
2. CrossPlotTool: Crossplots data in two or four volume datasets and performs classification.

3. FaciesProbabilityTool: Performs quality control of classified volumes by resampling well logs to the volume data grid, and then compare the logs and the volume classification.
4. FenceTool: FenceTool: Displays a vertical cut plane through a volume dataset along a well path. Can also display well logs on the fence.
5. Geomodeller: Depth adjusts horizons to match updated information concerning the geology, and updates the seismic volume accordingly.
6. GeoTimeManager: Synchronizes animation of time dependent surfaces, and applies transformations defined by the horizons on seismic data.
7. GrowingTool: Performs single and multi-attribute volume growing. Surfaces can be generated from the grown volumes. Surface growing is also supported.
8. LODManager: Controls dynamic level of details for the SurfaceTool.
9. MPathTool: Connects to the migration simulation program MPath (developed by Permedia Research Group) and visualizes the results while a simulation is running.
10. PointTool: Displays points from xyz text files in various rendering modes. Point colors are calculated from depth or attribute value.
11. ReservoirTool: Visualizes RMS and Eclipse geomodels through volume and slice visualization and animation of results from Eclipse reservoir simulations.
12. RMSTool: Manages data for the ReservoirTool. Parses a directory structure of RMS projects and lets the user choose realization, parameter and grid segment from a structured menu.
13. SceneManager: Saves and restores SHIVR user sessions.
14. SketchingTool: Creates sketches including text and symbols, e.g. to annotate displayed data.
15. SliceTool: Visualizes slices through volumetric data, e.g. seismic cubes. The slices can be freely moved and scaled. Supports multi-attribute visualization.
16. SurfaceEditor: Edits grid surfaces, either by editing a selection of points or by NURBS modeling.
17. SurfaceTool: Visualizes surfaces. Supports a wide range of different data formats. The surfaces can be textured with height map data, images or intersected volumetric data.
18. ThreeDcorrTool: Correlates markers on a set of wells, and generate surfaces based on the resulting correlation lines.
19. TimeManager: Synchronizes the animation of various tools.
20. TwoDdimmingManager: Emphasizes interesting zones in 2D seismic data by dimming the less interesting sections.
21. TwoDflatteningManager: Lets the user select a horizon for flattening, and then transforms the data in seismic 2D lines correspondingly.
22. TwoDsurveyTool: Visualizes survey base maps to get an overview of one or several 2D seismic surveys.
23. TwoDtrackingTool: Tracks horizons on 2D seismic lines.
24. VolumeMaskTool: Defines regions, which for example can be used for limiting the classification volume and crossplot data volume in the CrossPlotTool.

25. VolumeRenderTool: Uses shader technology and multi resolution techniques for fast volume data rendering.
26. VolumeRgbBlendTool: Performs multi-attribute visualization where the weights of the red, green, and blue color channels are mapped to attribute values.
27. VolumeUncertaintyTool: Visualizes uncertainties in volume datasets.
28. VolumeWindowTool: Visualizes seismic data using a 3D window that can be scaled and moved around within the seismic cube. Supports multi-attribute visualization.
29. WellHistoryTool: Visualizes and animates injection and production data for wells. Production data can also be interpolated onto surfaces.
30. WellTool: Visualizes wells with logs. Supports planning of new wells and monitoring of the drilling process. Supports visualization of well completion and well core samples.
31. WorldTool: Manages transformations of spatial data and offers several different modes for navigation through the data.
32. 2DViewer: Visualizes and explores 2D sections of seismic and simulated data, projected well paths and logs, surfaces, and linear polygons. The visualizations are organized as a Multiple Document interface (MDI) enabling the possibility to study several sections simultaneously.

References

1. J. Adam, Virtual reality is for real. Spectrum **30**(10), 22–29 (1993)
2. R. Brady, J. Pixton, G. Baxter, P. Moran, C. Potter, B. Carragher, A. Belmont, Crumbs: a virtual environment tracking tool for biological imaging. Biomed. Vis. 18–25 (1995)
3. C. Cruz-Neira, D. Sandin, T. DeFanti, Surround-screen projection-based virtual reality: the design and implementation of the CAVE, in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (ss. 135–142) (ACM, Anaheim, 1993)
4. M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, M. Mineev-Weinstein, ROAMing terrain: real-time optimally adapting meshes, in *Proceedings of the Visualization '97* (IEEE, 1997), pp. 81–88
5. H. Hauser, Generalizing Focus+Context Visualization, in *Scientific Visualization: The Visual Extraction of Knowledge from Data.* ed. by G.-P. Bonneau, T. Ertl, G. Nielson (Springer, Berlin, 2006), pp. 305–327
6. HTC, *HTC Vive*, Mar 2021. Retrieved from https://www.vive.com/eu/product/#vive%20series
7. IEEE, *P3079/D02 IEEE Approved Draft Standard for Head Mounted Display (HMD) Based Virtual Reality (VR) Sickness Reduction Technology* (2020)
8. J. Leigh, A. Johnson, T. Defanti, CAVERN: a distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments. Virtual Real. Res. Dev. Appl. **2**, 217–237 (1997)
9. E. Lidal, T. Langeland, C. Giertsen, J. Grimsgaard, R. Helland, A decade of increased oil recovery in virtual reality. Comput. Graph. Appl. **27**(6), 94–94 (2007)
10. Paradigm Ltd., *The web page for the structural and stratigraphic interpretation software VoxelGeo by Paradigm Ltd* (2021). Retrieved from https://www.pdgm.com/products/voxelgeo/
11. M. Midttun, R. Helland, E. Finnstrom, Virtual reality—adding value to exploration and production. Lead. Edge **19**(5), 538–544 (2000)

12. EPOS Paradigm Ltd., *The web page for EPOS Data Management and Interoperability by Paradigm Ltd* (2021). Retrieved from https://www.pdgm.com/products/epos/
13. Porcino, T., Trevisan, D., & Clua, E. (2020). Minimizing cybersickness in head-mounted display systems: causes and strategies review. *22nd Symposium on Virtual and Augmented Reality (SVR),*. Porto de Galinhas, Brazil.
14. L. Rebenitsch, C. Owen, Review on cybersickness in applications and visual displays. Virtual Real. (2016)
15. S. Röttger, W. Heidrich, P. Slussallek, H.-P. Seidel, Real-time generation of continuous levels of detail for height fields, in *Proceedings of the 6th International Conference in Central Europe on Computer Graphics and Visualization* (1998), pp. 315–322
16. The Permedia Research Group, *The web page of the petroleum migration simulator included in the Permedia*™ *Petroleum Systems Software (formerly MPath)* (2012). Retrieved December 2012, from http://www.permedia.ca/products/basin-migration.html

Evolution of VR Software and Hardware for Explosion and Fire Safety Assessment and Training

Chad Jarvis, Veronika Solteszova, Dag Magne Ulvang, Djurre Siccama, and Daniel Patel

Abstract Building and maintaining a custom-made Virtual Reality (VR) system is expensive and time consuming. The recent availability of affordable and capable head mounted displays (HMD) and graphics cards along with powerful 3D game engines has created new opportunities for implementing VR systems. In this chapter we describe the historical development of software and hardware for VR systems and applications from the 90s to today, as well as the advantages, disadvantages and challenges that the recent developments have introduced. The development is described through the journey of porting a VR system from a custom made game engine displayed on a backprojected screen to using an inexpensive off-the-shelf system with the Unity game engine displayed in an HMD.

1 Introduction

Several VR systems used in the areas of oil and gas exploration and medical visualization had been developed by Christian Michelsen Research (CMR), which is now

Chad Jarvis, Veronika Solteszova: Contribution described here was done when working at NORCE Research.

C. Jarvis (✉)
GraphCore, Oslo, Norway
e-mail: chadj@graphcore.ai

V. Solteszova
Equinor, Bergen, Norway
e-mail: ves@equinor.com

D. M. Ulvang · D. Siccama
Gexcon AS, Bergen, Norway
e-mail: dagu@gexcon.com

D. Siccama
e-mail: Djurre.Siccama@gexcon.com

D. Patel
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_8

part of Norwegian Research Centre (NORCE). One of these systems was for risk assessment, risk communication and safety training in fire and explosion scenarios. The VR hardware consisted of a large screen display and shuttered glasses for stereo viewing, including positional and rotational tracking. The software was an in-house developed custom 3D engine. Custom 3D engines are time consuming to develop and maintain, and the VR hardware we used took much space and was costly. However, the recent availability of affordable but advanced VR HMDs, combined with powerful 3D game engines such as Unity [1] and Unreal Engine [2], has created a new opportunity in VR technology.

To test the potential of this new technology we ported the functionality of our custom VR system into the Unity game engine and used the Oculus Rift DK2 [3] for rendering and tracking. In this chapter we will discuss our previous experience with VR technology and then compare and contrast it with the new technology.

2 Background

HMDs have been used extensively for military training purposes [4] and are entering the public domain for medical training [5]. Recently, companies are increasingly embracing VR for training their personnel in fields such as safety and security, sales, engineering, hospitality, HR, leadership and education [6].

In 2004, CMR started development of a VR solution called VRSafety [7, 8], funded by Equinor and Norsk Hydro. VRSafety was developed for evacuation and safety training and for discussing and reviewing structural changes in existing or planned industrial environments. Computational Fluid Dynamics (CFD) simulations calculated in external software were imported and visualized to simulate fire, gas leak and explosion scenarios inside the models. To visualize the simulation results, we implemented volume rendering and isosurface rendering. To discuss mitigation steps, we made it possible to do basic editing of the geometry model. In effect, we had created a digital twin with respect to visual appearance, and visualization of fire, gas and explosion scenarios that could be used for learning and training on e.g. hazard identification and to find evacuation routes. As spatial understanding and the sense of presence is important, particularly in training scenarios, we implemented support for 3D immersion using VR. In addition to training, experts and non-experts could use the solution to communicate hazard and risk topics, e.g., by showing how much larger an explosion would be if a certain opening would be closed off with a wall.

VRSafety had implemented a real time connection to the FLACS (Flame Acceleration Simulator) CFD simulator [9], this enabled defining explosion, gas leak and fire scenarios in VR and immediately starting a FLACS CFD simulation allowing real-time visualization of the calculated results in the VR environment. However, real-time visualization of the simulation output was not practically useful due to long computation times. It took in the order of seconds to simulate a single timestep, which could be between a fraction of a second to a few seconds long depending on the type of simulation. So in most cases the scenarios to be discussed were computed in advance before running the VRSafety application.

2.1 Evolution of VR Hardware and Software

Before starting the development of VRSafety, CMR had already implemented another 3D engine for supporting VR applications in oil and gas exploration and production, called SHIVR. This 3D engine, which is described in the chapter titled "When Visualization and Virtual Reality made a Paradigm Shift in Oil and Gas", was developed for UNIX running on an SGI ONYX visualization system. The graphics rendering of the 3D engine was built on top of the low-level OpenGL library [10], and it supported immersive visualization in CAVE setups [11] through the use of the (discontinued) CAVELib library. CAVElib handled the graphics display on each wall of the CAVE, including computation of correct angles and projections. For tracking the user, hardware technology (discontinued) was used which consisted of wired electromagnetic sensors (Ascension Flock of Birds [12]). The engine had collaborative capabilities where several instances of the software could run at different sites, enabling geographically dispersed users to meet and collaborate within the virtual environment.

Interfacing OpenGL directly is a time-consuming way of programming computer graphics applications, as almost every high-level functionality has to be implemented from scratch. Therefore we based the 3D engine for the VRSafety application on a higher graphics abstraction level provided by the OpenGL Performer scene graph API by SGI (now discontinued). On top of OpenGL Performer we implemented our own navigation system, event system, and XML meta-language for controlling behavior in the virtual environment [13]. As SGI started experiencing financial difficulties, the support on the hardware and libraries we were using became uncertain. Therefore, we switched from SGI's OpenGL Performer scene graph to OpenScenegraph [14] and ported our 3D engine from UNIX to Microsoft Windows. Due to the rapid development of the game industry, we were now able to replace the expensive SGI Onyx OS and hardware systems with a high-end PC and commodity graphics cards costing a fraction of the price, without losing performance. In addition, we no longer needed a separate server room for the large SGI Onyx rack setup. At the same time, we exchanged CaveLib by VRjuggler [15] which had no license fees associated with it. We were also able to get rid of wired tracking by replacing the Flock Of Birds tracking system with the IO Tracker [16] tracking system which was using infrared cameras, emitters and reflective markers.

VRSafety was designed for running in a CAVE environment, but during development we were running the application in a downscaled immersive VR setup consisting of a $4.6\,m \times 1.6\,m$ back-projected screen, powered by two partly overlapping Barco Galaxy NW-7 projectors. The overlap was smoothed using edge blending, resulting in a resolution of 2048×1600 120 Hz. Stereo was achieved using active stereo shutter glasses from Nvidia that were synchronized with the display through an infrared signal. Positional and rotational tracking of the glasses and of the pointing device was achieved with the IO Tracker system using reflective markers. The markers were tracked by infrared emitting and receiving cameras positioned around the bevel of the display, see Fig. 1.

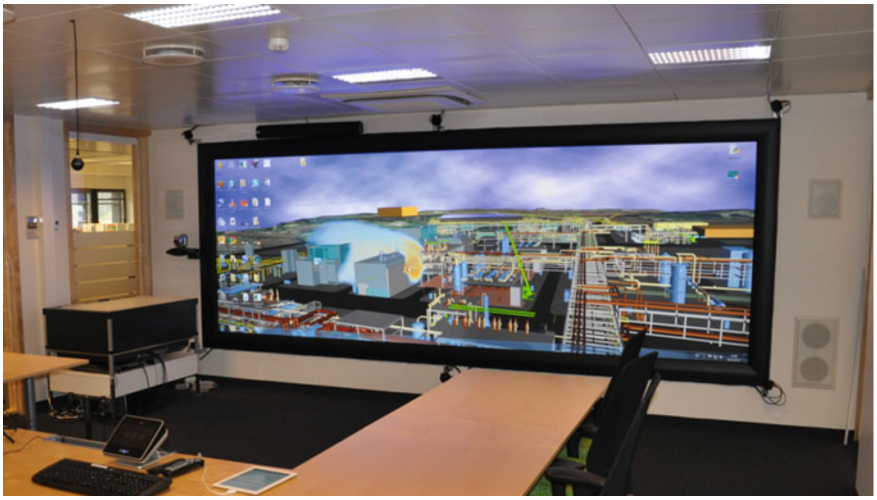


Fig. 1 Immersive display environment consisting of a back projected screen with infrared tracking cameras around the screen bevel

2.2 Experiences with VRSafety

The new VRSafety setup was considerably improved, but still consisted of expensive hardware that required a custom-made visualization room, with an additional projector room located behind the screen. In addition, an initial manual calibration process had to be performed by trained personell, both for the projectors and the tracking system. The setup was ideal for simultaneously immersing multiple people in the virtual environment, and for facilitating collaborative discussion sessions between experts from different disciplines. In our particular setup, we only had the front wall projected, resulting in a lower degree of immersion than in a multiwall CAVE. We used this setup as it could also be used as an ordinary large screen for meetings, and it was more affordable than a CAVE setup. Our immersive work sessions did usually not produce motion sickness among the participants. Compared to the currently available rendering engines and frameworks, requests for added functionality was time consuming to implement due to our in-house developed framework. The framework also naturally did not benefit from bug fixes and feature updates from external actors.

3 Head Mounted Displays and Game Engines

3.1 Evolution of HMDs

Because of the recent interest in VR from companies such as Facebook, HTC, Samsung and Sony, affordable HMDs have now become commercially available. These devices offer similar features, such as six degrees of freedom and a wide field of view (FOV). Sony's device is geared toward the PlayStation game console whereas the others are run by connected PCs or inbuilt mobile devices. Now that head mounted displays are commercially available and there are several companies competing in the marked, the hardware is rapidly improving. HMDs available in the 90s suffered from a very narrow field of view. Some had angular tracking, but to also have positional tracking, external sensors had to be attached such as the wired electromagnetic Flock of Birds sensor. Since the first prototypes of HMDs, the field of view has increased dramatically, and 6 degrees of freedom tracking is supported both on the HMD and on the hand-controllers. Tracking has evolved from using several external cameras for detecting the position and orientation of a device, called outside-in tracking, to using cameras embedded in the device, called inside-out tracking. The latter solution requires less hardware setup and calibration, and does not limit the movement to a confined area within the cameras' view. Most HMDs require cables, but cableless HMDs are appearing. One example is the Oculus Quest device which is based on a stand-alone Android mobile device with inside-out tracking. Such systems allow the user to move freely around in the physical environment. The disadvantage with using mobile computation of VR is that it is less performant than an HMD connected with cables to a powerful computer. It is possible to send images wirelessly from a powerful computer to an HMD, but this adds some latency between a user's action and an updated rendering. If latency can be reduced to become unnoticeable, one will be able to achieve high quality rendering on lightweight wireless HMDs which could improve the user experience dramatically. Another recent development is eye tracking integrated into the HMD which makes it possible to track what the user is looking at. This feature is already available in the VIVE Pro Eye device [17]. Eye tracking can open up for more intuitive interaction with the VR world, and for higher framerates or better-quality renderings by focusing the rendering computation to the area that the user is looking at. This is called foveated rendering [18].

3.2 The Oculus Rift HMD

We used the Oculus Rift DK2 HMD, as this was the only commercially available HMD at the onset of the project. The lenses used in the Oculus Rift creates two distortions: pincushion distortion and chromatic aberration. These distortions are corrected for by convoluting the image with a barrel distortion and distorting the red, blue, and green components of the image to cancel out the chromatic aberration. Both the 3D position and angles of the DK2 is tracked. The position is measured by a stationary infrared camera observing at 60 FPS infrared (IR) emissions from an array of IR-LEDS on the DK2. The angles are measured with an accelerator in the DK2.

3.3 Unity and Unreal Engine 3D Software

To find out which 3D engine we should port VRSafety to, we evaluated the Unreal and Unity game engines because of their state-of-the-art features and liberal end user licenses. The Crytek CryEngine also offers state-of-the-art features but is only allowed to be used for game development [19]. The engine cannot be used for scientific simulations and serious games as the end user license agreement states: "Under this Agreement the following will not be considered Games: military projects; gambling; simulation (technical, scientific, other); science; architecture; pornography; Serious Games".

Unity and the Unreal Engine are multi-platform game engines. They provide advanced functionality that was lacking in our previous system such as a powerful scene editor for adding geometry, interactive landscape shaping, (dynamic) foliage and visual environmental and weather effects. Additionally, all objects can be scripted for adding behavior to them.

Both game engines are reasonably priced with scalable fees according to either game revenue or development licenses used. Unreal has a 5% royalty fee for revenues above 1 million USD per product [20]. If the product is distributed through the Oculus store however, the royalty free revenue limit is raised to 5 million USD [21]. In 2019 Unreal removed the restriction regarding using the engine for gambling-related activities, for military activities with live combat, in nuclear facilities, or in critical aircraft software.

Unity has three licenses. With the personal license, products created with Unity can be used, distributed and sold without fees by entities earning less than 100,000 USD per year. Entities earning less than 200,000 USD can use the Plus licence for 35 USD per month or 299 USD per year. Entities earning more than 200,000 USD must use the professional license which will cost 150 USD per month per developer or 1800 USD per year [22]. In 2019 Unity removed the restriction regarding using the engine for gambling-related activities. The licenses described here are from 2021 and may change from year to year.

4 Porting from VRSafety to a Modern Head Mounted Display and Game Engine

In the following text we will refer to the new platform which we port VRSafety to, as VRFlacs. The porting consisted of two main tasks, transferring assets such as geometry models to a game engine, and implementing/porting the software functionality found in VRSafety.

4.1 Porting of Geometry Models

The geometry constituting the industrial model was originally in a CAD format but was provided to us as an OpenScenegraph Binary file (IVE) of 668 Mb. The geometry consists approximately of 5 million triangles

Both game engines support the FilmBox format (FBX) [23], while Unity additionally supports Collada (DAE) [24]. Therefore we focused on converting the geometry to FBX. We identified two geometry software packages which advertised the ability to convert from OpenScenegraph exportable file formats to FBX. Sketchup Pro [25] could import DAE and export FBX, but the software crashed during import. Blender [26] could transform both Autodesks [27] 3ds Max format (3DS) and DAE format to FBX, but Unreal failed to load the FBX file created both from 3DS and from DAE. It appears that Unreal's FBX importing was not suitable for large CAD files. Although, we were pleased with the Unreal Engine editor and impressed with the visual realism of the Unreal Engine, due to the failure to import our model, we decided to continue with the Unity engine. We did not investigate the many third party importer plugins for Unreal, nor the latest version of the engine, which might have solved this problem.

Unity successfully imported the FBX file, however, the original smooth shading was turned into flat shading. Loading the DAE format that had been exported from OpenScenegraph maintained the smooth shading, however all textures had been lost during conversion, but it was relatively easy to reassign the textures thanks to the Unity editor allowing for editing geometry. In addition, many surfaces flickered due to overlap with other surfaces with different colors (Z-fighting). This issue was also quickly resolved in the Unity editor by deleting one of the overlapping surfaces. Figure 2 shows an overhead view of the model imported into Unity after being updated in the Unity editor.

4.1.1 Adding Terrain, Sea and Sky

Compared to VRSafety which lacks a graphical editor, Unity supports many advanced features that can be created easily within the integrated graphical scene editor. Unity also has an assets store where one can purchase textures, models and visual effects. Our original model used in VRSafety was positioned inside a large sky-textured box that rendered the sky. It also included geometry representing the surrounding terrain. As Unity supports several sky-models and allows for interactive terrain sculpting and the adding of foliage, we replaced the original sky and terrain with Unity-made models. In addition, we added animated trees and an animated sea into the model. Using the interactive editor in Unity, we were able to create surroundings that more accurately represented the real surroundings of the model, see Fig. 3 bottom.

Unity supports several extensions for improved rendering quality, such as advanced shading models with shadow casting, and dynamic content such as animated environmental effects in water, trees and leaves. Adding too many of these features quickly degraded the framerate to below comfortable levels for VR. Rendering our scene in

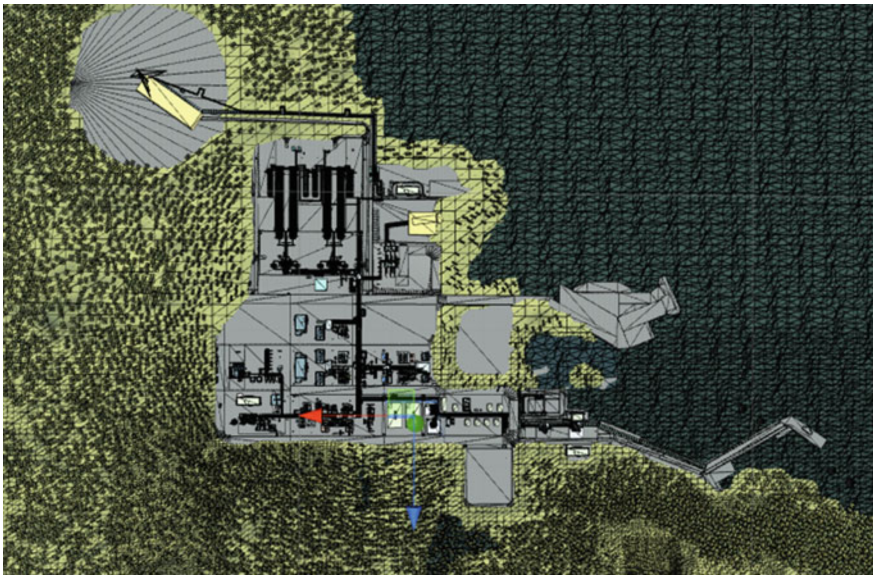


Fig. 2 Top view of the industrial model in the Unity editor. The sea and the surrounding terrain with foliage were created in the Unity editor

Oculus Rift with the basic Phong shading model as used in our previous VRSafety system, and having disabled animation of water and trees, resulted in interactive framerates at just over 70 FPS on an Nvidia GTX580 graphics card from 2011. This is an outdated graphics card, but provides at least a reference number for the framerate.

4.2 Implementing Software Functionality

Adding functionality such as navigation and collision detection in Unity was simply a matter of importing an asset and clicking a check box. Other functionality available in VRSafety, such as moving objects around in the scene, has not been implemented in Unity, but by inspecting the game engine design and scripting functionality we believe this would be easy to implement. VRSafety supported a two-way interactive connection with the FLACS CFD fire, leak and explosion simulator. However, this was not practically useful due to slow simulation times, therefore it was not ported to VRFlacs. For visualizing the simulation output, VRSafety had implemented an isosurface renderer for showing boundary surfaces for a user defined value and attribute, and a volume renderer for showing all values of an attribute mapped with a color table. In VRFlacs, we implemented only a volume renderer since it could be used to render isosurfaces also.

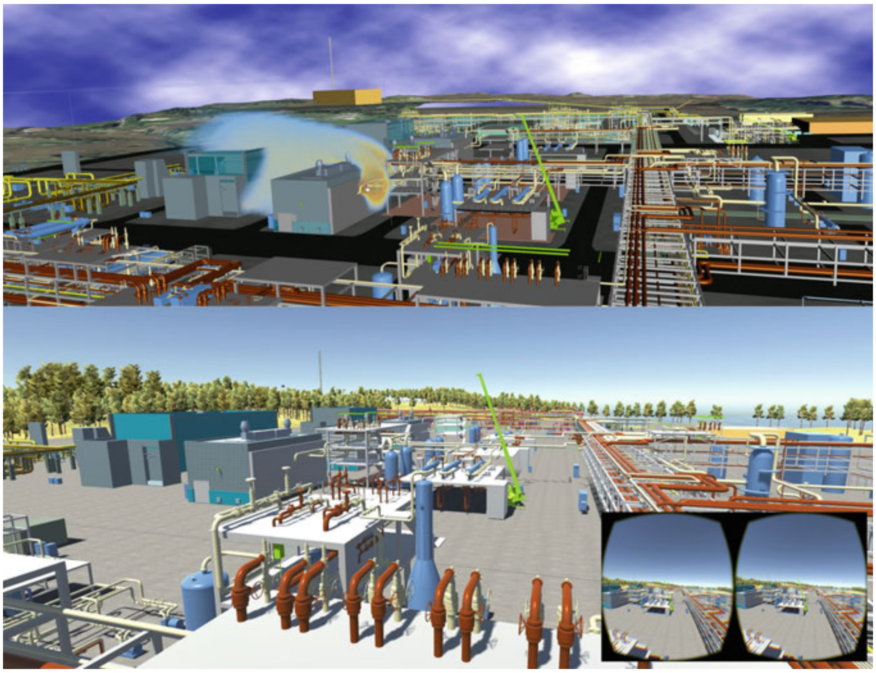


Fig. 3 Top: Rendering from VRSafety showing geometry and volume rendering of a gas leak. Bottom: Rendering from VRFlacs using Unity showing geometry with trees, shadows and water in the distance. Bottom Right inset: Stereoscopic rendering generated for the DK2

4.2.1 Volume Rendering

The CFD simulator used for computing gas dispersion and gas explosions generates a 3D volumetric dataset for each attribute and timestep. Volume rendering is a suitable method for visualization of such datasets. As Unity does not have built-in volume rendering capabilities, we implemented this ourselves. There are parameters that must be set before starting a simulation. The timestep size and spatial resolution of the simulation grid is manually set to achieve sufficient accuracy and to capture relevant features. Simulations such as explosions that start and end in a fraction of a second require smaller timesteps than e.g. a slow burning oil fire. The attributes to be output could be temperature, pressure, gas composition or a multitude of other attributes that are calculated by the simulator. As the volumetric data output consumes much memory, a decision to only store e.g. every 2nd or 4th timestep to the VR environment can be taken.

To show the data quantitatively, a slice through the volume at a user-defined position can be displayed. The user chooses which scalar simulation value to display, for instance temperature, and chooses a colormap for translating the scalar value to a color. This is useful for reviewing the simulator output.

For giving a qualitative impression of a fire scenario, the fire is rendered in a realistic manner. To achieve this, the simulator must calculate temperature and soot particles per volume unit (i.e. per voxel). The renderer then maps temperature to a wavelength spectrum using Planck's law of black-body radiation [28], and soot is mapped to opacity. The wavelength spectrum is further mapped to an RGB color value using the CIE 1931 color spaces [29] which define quantitative links between a wavelength spectrum and physiologically perceived colors in human vision. The RGB color represents chromaticity only (Planckian locus). To add lightness to the color according to how much energy is radiated from the point, the color is modified using the power of the spectrum. An example is shown in Fig. 5, bottom.

Gas leaks are mostly invisible, so in order to visualize them in VRSafety and in VRFlacs in Unity, we used a non-physically based mapping from gas density to color, which gave the leak a cloud-like appearance. Such a rendering from VRSafety is shown in Fig. 3, top. We can also show the extent of the gas leak by volume rendering a single isosurface for a user defined gas density, for instance for lethal or combustible levels of gas, giving the impression of a growing (semitransparent) bubble around the critical area.

In Unity, we implemented slice-based volume rendering [30]. This represents a volume as a set of semitransparent slices stacked behind each other, always facing the viewer, see Fig. 4 for a 2D schematic view. Each slice is a textured rectangle represented in Unity as a Quad GameObject. A higher number of slices improves the accuracy of the rendering at the cost of reduced framerate. Parts of slices that are behind existing scene geometry are automatically hidden due to depth buffering, and this ensures that the volume rendering is integrated with the geometry rendering. Because of this, only opaque geometry is supported inside the volume rendering.

To increase quality without decreasing the framerate, we use a smaller distance between slices when they are closer to the viewer instead of having slices evenly spaced. This works since data closer to the viewer is more visible which affects the resulting rendering more than data further away. We used an initial distance of 0.5 voxels for the first slice (Nyquist sampling rate) and increased this with 1% per additional slice. The user can speed up the volume rendering at the cost of reduced quality by increasing the initial and thereby the following slice distances. Each slice represents the optical properties of a slab around the slice. This is depicted in Fig. 4, where slices are shown with stippled vertical lines, and slabs with solid vertical lines. For simplicity, we use a constant distance between slices in the illustration. The blue rotated square represents the bounds of the volume data and the black circle represents opaque geometry inside the volume. The color and transparency of a given pixel on the texture of a slice depends on the data in the volume at the 3D position of the pixel, and how this value is mapped to color and opacity. Since the slice represents a slab of a specific thickness, this thickness will affect the transparency of the pixel. The thickness is affected by a slab's intersection with other geometry in the volume, and by its intersection with the volume bounds. In Fig. 4 is shown line intervals of different colors representing slab thicknesses. To improve rendering quality, the predefined slab thickness (sampling distance) is reduced to fit the front of the volume bounds (green), the back of the volume bounds (orange) or the front of geometry

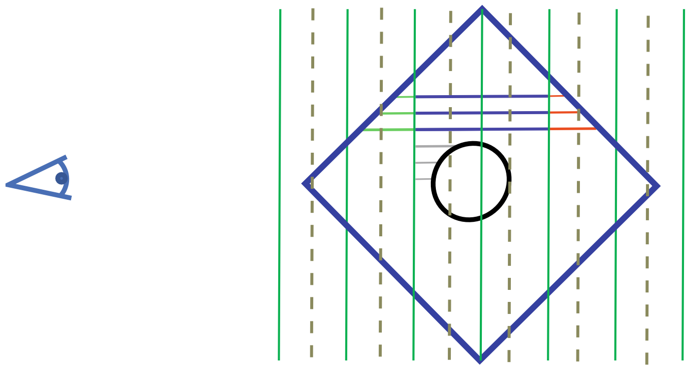


Fig. 4 The eye represents the viewing direction. Blue square shows volume bounds, black circle represents opaque geometry inside the volume rendering. Stippled vertical lines represent slices, green vertical lines represent slab borders. Horizontal lines represent different slice thicknesses due to intersection with volume bounds (green and red) or geometry (grey)

(gray). In blue is shown a few internal intervals where no adjustment is needed. There is also no adjustment behind the geometry since this area will not be visible as we only support opaque geometry. When not taking these distances into account, artifacts as shown in Fig. 5 top will be visible which are more pronounced when using larger slice distances. Figure 5 top shows jagged artifacts in square numbered 1) and 2) arising from not taking into account the distance to geometry and to front of volume bounding box respectively. Bottom figure shows the rendering when these distances are taken into account.

To calculate the distance to geometry, we accessed the depth buffer after rendering the scene geometry. The distance to the cuboid volume bounds was calculated analytically. We experienced that the programming interface for low level access to graphics features such as the depth buffer was not always direct and efficient in Unity. This resulted in a reduced framerate as compared to e.g. an OpenGL implementation.

4.2.2 Particle System Rendering

We faced several problems when visualizing the simulations using volume rendering. The graphics-intensive volume visualization reduced the framerate to below the recommended framerates for an optimal VR experience. In addition, simulations take up much space in memory which limits the length and size of the simulations that can be displayed. Finally, there is the problem of simulations that have long timesteps, e.g. the simulation of a long running fire. Since the fire lasts longer and is less dynamic than an explosion, the timestep of the simulator is set to one second to save memory. However, animating a fire which changes appearance only every second breaks the realism. In order to address the issues of memory usage and nonsmooth animation of fire, we looked at how fire and explosions are expressed in games with limited computing and memory consumption. One common technique in games to visualize

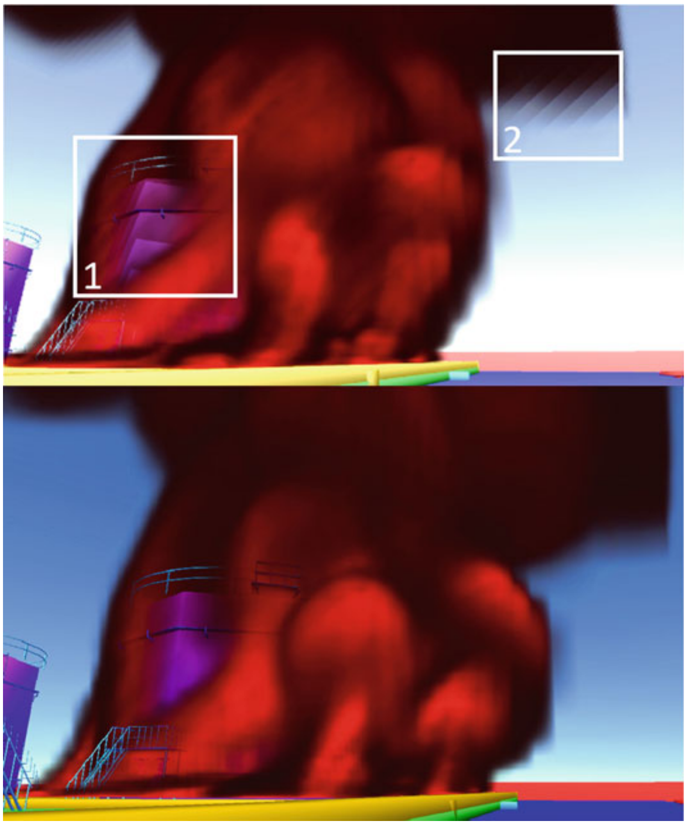


Fig. 5 Slice-based volume rendering shown together with geometry in Unity. Top: Jagged artifacts when not taking into account distance to geometry (1) and to volume bounding box (2). Colors have been exaggerated to better show the artifacts. Bottom: Volume rendering when taking these distances into account (The two images show data from two slightly different timesteps)

fires is to use particle systems [31] where each particle is a rising billboard having predefined fire animations playing on the surface of the billboard. A billboard is a 2D view-aligned textured surfaces with transparency masking. The animated billboard functionality is integrated in Unity's particle system module via the Texture sheet Animation module [32]. Several predefined fire particle systems can be directly downloaded through the Unity Asset Store, which made it easy for us to implement this functionality.

For a specific training scenario where accurate simulations have been performed, we simply positioned particle systems for fire at the positions of the fires, and set their parameters to approximatively match the simulation results. We also calculated and showed the accumulated radiation received by the player. This was calculated based on the volumetric output from the simulator. See Fig. 6 for renderings of fire and smoke using particle systems. To automatically get more accurate renderings

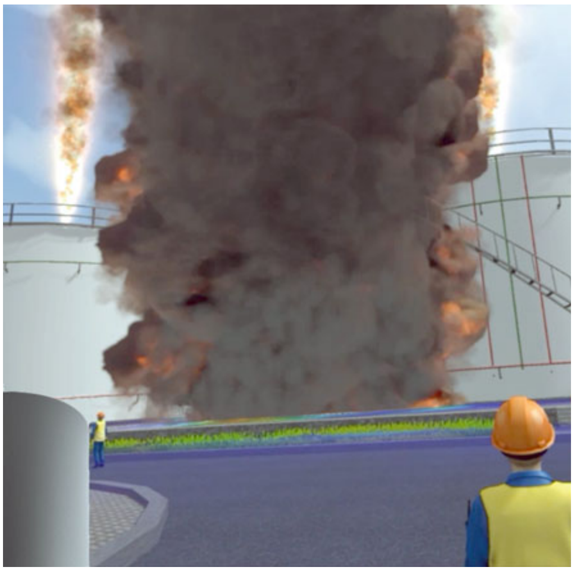


Fig. 6 Particle systems used to visualize fire and smoke. In the top left corner is visualized an ignited gas leak. Colors are exaggerated in the figure to better show the results

of fire using particle systems without manually setting parameters for the particle systems, we also experimented with limiting the particle movements based on the simulation data. This approach looked promising, but we did not have time to explore this sufficiently.

4.2.3 Support for Multiple Users

In VRSafety, local collaboration was possible simply because the participants were physically present in the same room and could see and talk to each other. In Unity, the application was running in an HMD, which is designed for one person only. To support multiple collaborators, the multiplayer functionality supported in Unity was used, where several instances of the application, each running on an individual computer, are synchronized. In this mode, the users can see the avatars of each other, see Fig. 7.

5 Modes of Work in VR

VRSafety supported a desktop mode where a single user would typically review and inspect simulation results on a desktop computer by looking at the volume rendering animations, using quantitative slice visualization at specific timesteps and using a probe to read out simulation values at specific positions in space. For discussing the simulation results with a group of people, VRSafety could be run in VR where

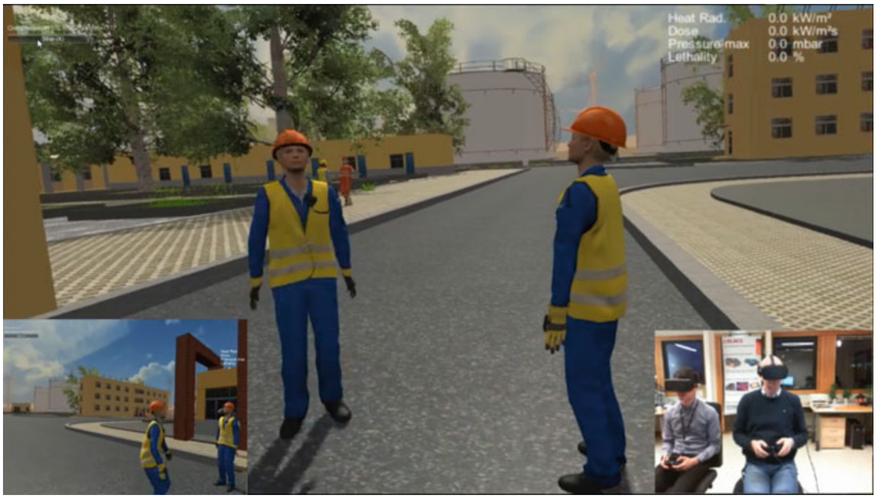


Fig. 7 Multiuser mode. Two users (bottom right) are immersed. Third-person view seen by each user is shown in bottom left and main image respectively

the same operations could be performed as in desktop mode. A different mode of operations was to run VRSafety in VR for the purpose of replaying a realistic recreation of a simulated situation and immersing a group of participants into the situation. This was useful for efficient communicating the consequences of leak, explosion and fire situations. In this mode, VRSafety needed an operator to guide the participants through the scenario.

In Unity, there was built-in support for navigation by walking and running, for animation of realistic characters and for creating game logic that triggers events based on a user's actions in the world. Together with the high immersion created by the HMDs, this made it easy to create a VR training mode where the user learns by being exposed to various situations in the virtual world which must be handled correctly. This mode of work enables safe and affordable training on dangerous situations. Figure 8 shows one such scenario that we created, where the user's actions affected how events unfolded. These scenarios supported multiple immersed users as shown in Fig. 7.

6 The Immersive Experience in VRFlacs Compared to VRSafety

Those that had used the VRSafety application in our back projected single wall setup with 3D glasses found the experience with the Oculus Rift far more immersive. Several people trying VRFlacs with an HMD found it so immersive that they expected to see their hands and arms when they moved them. Newer HMDs from Oculus (Rift S) now include two tracked hand controllers which makes this possible.

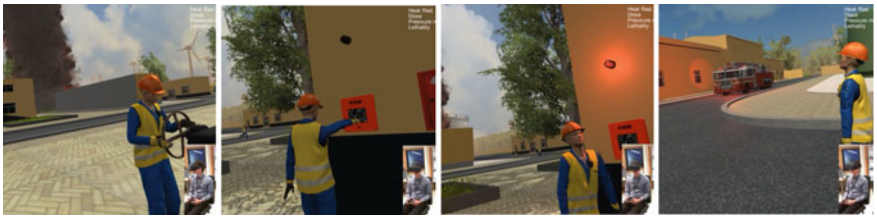


Fig. 8 A scenario playing out from left to right. A user detects a fire, localizes the valve and closes the gas supply to the fire. The user sets of fire alarm. Fire alarm is set of. The fire truck arrives

We did face some new challenges using an HMD for VR. Initially, a first-person perspective was used when being immersed. After an approximately 30 min of immersion, the main author experienced nausea. This lasted for over an hour. Reducing negative sideeffects from VR is an actively researched field [33]. Limiting the user's ability to perform changes in movement, and using HMD's with increased field of view or increased framerates are examples of techniques that help. In our case, we switched from a first-person to a third-person perspective which gave fewer viewpoint changes when navigating in VR and this reduced nausea.

A problem we faced with the complex terrain we added along with shadows and a dynamic ocean, is that it had a dramatic effect on the framerate of the application which was particularly noticeable during rotational head movements. Lag in rotational head movements was not a problem in the VRSafety solution since the rendering on the large-screen display did not have to change much when the user rotated his/her head. The low frame rate in our new VRFlacs framework not only damages the immersive experience but also contributes to motion sickness.

We believe that with a more modern GPU along with a better understanding of the Unity engine, we can achieve a sufficiently high framerate. For now, we have created two version of our demonstration: one with a complex terrain and ocean and one with a simple terrain and no ocean. In addition, we do not use volume rendering in immersive training scenarios. Instead we use the approximate particle system rendering.

Another challenge with the DK2 HMD compared to a large-screen setup is that the HMD blocks the view of the physical room one is situated in. Thus, collaborative sessions where several users see each other and e.g., point at parts of the model using hand gestures is not supported. Current HMDs come with tracked hand controllers, which will make it possible to show the participants as avatars with arms. Technology is developing that captures subtle body language and face mimic which is an important part of communication. Recently the VIVE Facial Tracker [34] has been released for the consumer market. It is is a device that works with HMDs and tracks the movement of the lower face including the mouth. This can substantially increase the ability to capture face mimic. There are now technological advances where the hand controllers might be replaced by sensors around the wrist that register the finger positions using cameras [35] or by reading the electric signals from muscles through the skin [36].

Table 1 Comparing the software aspects of the solutions. “+” is better than “(+)”, which is better than “−”

	Custom Engine (VRSafety)	Commercial Game Engine
Importing large geometry	+	(+)
Fast implementation	-	+
Rendering quality	-	+
Adding “standard” functionality	-	+
Adding “nonstandard” functionality	+	-
Pricing scheme	Free	Per dev. licence / Per sale

Motion sickness and the reduced ability to communicate with collaborators in the same room are probably the two largest challenges using HMDs compared to using shared stereoscopic screens or CAVEs. We have summarized in Table 1 the differences between our previous system (VRSafety) that used a large stereoscopic screen with shuttered glass and the current one (VRFlacs) that used the Oculus Rift. In Table 1, we compare the software aspects of the two solutions. We use the term “standard” functionality for functionality that is common in games such as realistic environments, collision detection, animation and multiplayer support. Conversely, with “nonstandard” functionality we refer to features not common in games such as volume rendering, which we could not implement as efficiently as we wished due to lack of enough low-level graphics control in Unity. In sum, we spent a substantially less amount of time on implementation in Unity than with OpenSceneGraph in VRSafety. This is also thanks to Unity’s integrated developer environment, good debugging abilities, and the interpreted C# language that does not require time consuming compilation. This is reflected by giving the category “Fast implementation” a + for Unity in the Table. VRSafety was developed in-house and did not have a purchase cost, while Unity has a cost per license. The license cost far outweighs the extra hours required for implementing a custom engine. The risk of using an externally provided game engine for a domain that it is not exactly designed for is that only after investing a certain (possibly large) amount of time, one may realize that a specific functionality is not possible to implement in a satisfactory manner. This is less likely to happen for a custom-made engine where one has more control.

Table 2 shows the differences between presenting VR through large screens and in CAVEs (Large-screen VR) compared to using HMDs (HMD VR). HMDs are more affordable than large-screen solutions. For fast rotational head movements in an HMD, a high framerate is required to not experience lag. However when the display is not attached to the head, this is not a problem. Perhaps partly related to the rotational framerate, motion sickness was less of a problem on large-screen VR in our experiences. When considering only the hardware, it is easier to collaborate with multiple people in the same room for Large-screen VR solutions than for HMDs as HMDs block the view of the room and the participants.

Table 2 Comparing the hardware aspects of the old and new solution

	Large-screen VR	HMD VR
Hardware price	high	low
“Rotational framerate”	high	medium/low
Motion sickness	low	medium/high
Collaboration support	high	medium/low

7 Discussion and Conclusions

VRSafety was developed as a tool for training and safety assessment of industrial environments. The costs for developing and maintaining a custom 3D graphics engine made it challenging to convince companies to pursue further investments. The recent availability of inexpensive HMDs with six-degrees of freedom tracking and a wide FOV, along with the availability of advanced 3D engines have rekindled our interest in VR as a viable platform for immersive training.

Our initial experience using the Oculus Rift DK2 device has been positive, as exploring the model of an industrial complex has never felt so immersive. Additionally, using Unity made it simple to reproduce many of the features we have in our custom 3D engine with much less effort, and the high visual quality provided by Unity generated a much more realistic visualization than we achieved with our 3D engine. The implemented volume rendering in Unity did not achieve high enough framerates for being used in a HMD in a training scenario. A faster but less accurate particle system rendering was instead used to give a qualitative impression of fire.

The major problem using the Oculus Rift DK2 was the motion sickness. This is a serious issue which needs to be carefully addressed. This may be as simple as avoiding low frame rates and lag to more restrictive solutions locking the user to a fixed inertial reference frame such as the cockpit of an automobile.

References

1. The Unity Engine. https://unity3d.com/unity. Accessed Mar 2021
2. The Unreal Engine. https://www.unrealengine.com/. Accessed Mar 2021
3. Oculus VR. https://www.oculus.com/. Accessed Mar 2021
4. A. Rizzo, A. Hartholt, M. Grimani, A. Leeds, M. Liewer, Virtual reality exposure therapy for combat-related posttraumatic stress disorder. IEEE Comput. **7 (2014)**
5. M. Bressler, A virtual reality training tool for upper limb prostheses. Masters thesis (2013)
6. Warp vr customer stories and use cases. www.warpvr.com/customer-stories. Accessed Mar 2021
7. J. O. Erdal, T. Langeland, D. Patel, I. Eliassen, FAVEum framework architecture for virtual environments applied to urban modelling, in *Next Generation 3D City Models* (2005), pp. 298–300

8. S. Hoiset, E. Glittum, Risk and safety training using virtual reality (VRSafety), in *SPE International Conference on Health, Safety, and Environment in Oil and Gas Exploration and Production* (2008)
9. FLACS Software, GEXCON. https://www.gexcon.com/products-services/FLACS-Software/22/products-services/flacs-software/. Accessed Mar 2021
10. The Industry's Foundation for High Performance Graphics. https://www.opengl.org/. Accessed Mar 2021
11. C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, The CAVE: audio visual experience automatic virtual environment, in *Communication of the ACM 35* (1992)
12. Ascension Flock of Birds. https://est-kl.com/manufacturer/ascension/flock-of-birds.html. Accessed Mar 2021
13. D. Patel, I. Eliassen, T. Langeland, FAVE, a framework architecture for virtual environments, in *The 3rd CAVE-Programming Workshop* (Helsinki, Finland, 2003)
14. OpenSceneGraph. http://www.openscenegraph.org/. Accessed Mar 2021
15. A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, C. Cruz-Neira, Vr juggler: a virtual platform for virtual reality application development, in *Proceedings IEEE Virtual Reality*, vol. 2001 (2001), pp. 89–96
16. T. Pintaric, H. Kaufmann, Affordable infrared-optical pose-tracking for virtual and augmented reality, in *Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop, IEEE VR* (2007)
17. Vive PRO Eye HMD. https://enterprise.vive.com/us/product/vive-pro-eye-office/. Accessed Mar 2021
18. A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, A. Lefohn, Towards foveated rendering for gaze-tracked virtual reality. ACM Trans. Graph. **35** (2016)
19. CryEngine. https://www.cryengine.com/ce-terms. Accessed Mar 2021
20. Unreal License. https://www.unrealengine.com/en-US/eula. Accessed Mar 2021
21. UnrealEULA. https://developer.oculus.com/documentation/unreal/unreal-oculus-license/. Accessed Mar 2021
22. Unity License. https://unity3d.com/legal/terms-of-service. Accessed Mar 2021
23. Fbx Format. http://help.autodesk.com/view/FBX/2017/ENU/. Accessed Mar 2021
24. Collada Format. https://www.khronos.org/collada/. Accessed Mar 2021
25. Sketchup. https://www.sketchup.com/products/sketchup-pro. Accessed Mar 2021
26. Blender Software. https://www.blender.org/. Accessed Mar 2021
27. 3DS Format. https://www.autodesk.no/products/3ds-max/overview. Accessed Mar 2021
28. R. Loudon, *The Quantum Theory of Light (third ed.)* (Cambridge University Press, 2000)
29. T. Smith, J. Guild, The C.I.E. colorimetric standards and their use. Trans. Opt Soc., **33** (1932)
30. J. E. Swan II, R. Yagel, Slice-based volume rendering. Technical Report OSU-ACCAD-1/93-TR1 (Ohio State University, Jan 1993)
31. W. T. Reeves, Particle systems—a technique for modeling a class of fuzzy objects. Trans. Graph
32. Unity Manual: Texture Sheet Animation module. https://docs.unity3d.com/Manual/PartSysTexSheetAnimModule.html. Accessed Mar 2021
33. T. Porcino, D. Trevisan, E. Clua, Minimizing cybersickness in head-mounted display systems: causes and strategies review, in *2020 22nd Symposium on Virtual and Augmented Reality (SVR)* (2020)
34. VIVE Facial Tracker. https://www.vive.com/eu/accessory/facial-tracker/. Accessed Mar 2021
35. F. Hu, P. He, S. Xu, Y. Li, C. Zhang, Fingertrak: continuous 3d hand pose tracking by deep learning hand silhouettes captured by miniature thermal cameras on wrist, in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2020)
36. B. Mu, R. De Nardi, R. Newcombe, R. King, E. Gander, R. Wang, Armband for tracking hand motion using electrical impedance measurement. Facebook Technologies (2019)

Groupware for Research on Subsurface CO_2 Storage

Daniel Patel, Tor Langeland, Saman Tavakoli, and Morten Fjeld

Abstract Directed towards the multidisciplinary research field on carbon capture and storage (CCS), this paper reports empirically on progress towards a groupware system (collaborative software) and experiences learned when creating groupware for CCS research. As subsurface CO_2 storage requires collaboration across a wide range of CCS disciplines, we describe the collaborative challenges faced and how our proposed groupware addresses these, and present criteria for CCS groupware and a field study over three iterations evaluating how well various software systems fulfill those criteria. In iteration one, we evaluate a commercial and feature-rich stand-alone application used together with a conferencing application. Based on this evaluation, in iteration two we develop and evaluate a custom-built lightweight web application. In the third and last iteration, we present a prototype that uses the most useful features from the two evaluated solutions. Finally, we give a suggestion of how to implement a groupware fulfilling all requirements identified for a successful CCS groupware.

1 Introduction

With growing concern for global warming, the possibility of storing CO_2 in subsurface layers has gained increasing interest from governments, industry, and research [1]. This paper presents groupware for a carbon capture and storage (CCS) research program named SUCCESS. SUCCESS [2] aims to establish knowledge on how to

D. Patel (✉)
Rapid Geology AS and Western Norway University of Applied Sciences (HVL), Bergen, Norway
e-mail: daniel.patel@hvl.no

T. Langeland
NORCE Research, Bergen, Norway
e-mail: tola@norceresearch.no

S. Tavakoli
Luleå University of Technology, Luleå, Sweden
e-mail: saman.tavakoli@ngi.no

M. Fjeld
University of Bergen, Bergen, Norway
e-mail: morten.fjeld@uib.no

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_9

safely store CO_2 in the subsurface by identifying suitable CO_2 storage locations; investigating CO_2 behavior during injection and storage; and estimating the effects of subsurface CO_2 leaks. The researchers were organized into seven groups to cover: CO_2 storage (1), CO_2 fluid flow (2), CO_2 sealing (3), CO_2 monitoring (4), effect of CO_2 on marine environments (5), CO_2 injection (6) and CO_2 education (7). During the five first years of the SUCCESS project, three collaborative challenges were identified, which, if addressed through creating suitable groupware, could stimulate the research.

R1. Remote collaboration. The partner institutions were in different locations, requiring mechanisms for facilitating good communication between distant researchers.

R2. Cross-discipline collaboration. Long-term and safe storage of injected CO_2 requires collaboration between a wide range of research disciplines (e.g. fluid injection, CO_2 flow, reservoir sealing).

R3. Efficient data sharing and handling. Long-term and safe storage of injected CO_2 requires sharing and understanding of a wide range of diverse data types arising from field studies, simulations and laboratory experiments. Handling of multiscale and multimodal data types as well as effective use of large and multidisciplinary datasets is required.

These challenges were identified by the group leaders based on the experiences and observations made during the project. Before starting spending time on addressing them, it was verified that also the researchers in CCS supported these challenges through a survey described in this chapter. The challenges have been formulated so that they can also be read as functionality requirements, i.e. groupware supporting remote collaboration, cross-disciplinary collaboration and efficient data sharing and handling could help reach the goals of SUCCESS. Each requirement is further subdivided into more concrete requirements in Sect. 3.2.

Handling geological data is challenging due to the the multitude of modalities that exist, the size of these modalities, and their multidimensional and multiscale nature. There are 3D geometries which can range from a detailed model of an outcrop (a mountain side) to elevation models covering vast areas of land. A different modality is large volumetric datasets of subsurface measurements. Here each data point may store measurements of multiple physical properties (e.g. pressure or magnetism). In addition the data can be time varying when the data covering the same area were collected at multiple time intervals, in this case the data is 4D. Often the datasets need to be visualized together with each other to be able to get an overview. This is called covisualization.

Interactive visualization of multiple datasets combined with remote collaboration has proven to enable interdisciplinary collaboration in fields such as automotive design, aviation design and medicine, resulting in better data utilization and faster product development [3]. Interactive covisualization and remote collaboration have also successfully been applied in the oil and gas industry as will be described in more detail in related work. This may indicate that these concepts could be effective in the field of CCS, which also deals with subsurface structures. Based on this research, our idea was to create groupware acting as a “virtual CO_2 lab” which would contain all existing data from the SUCCESS project. With the groupware, users users could access remotely from their geographic locations and visualize their data together with that of other researchers. This could enable remote collaboration (R1) and cross-disciplinary discussions (R2). To achieve simultaneous visualizations of various CCS data, the solution would have to support handling the diverse data used by the SUCCESS researchers (R3). The project team responsible for creating this groupware (referred to as “groupware team” or “we” in this chapter) consisted of two computer scientists and one geoscientist and were not affiliated with the CCS research in SUCCESS.

1.1 Related Work

Computer-supported cooperative work in oil and gas industry. Successful CSCW has been performed through the SHIVR software [4]. It resulted in improved workflows in Equinor, leading to a considerable decrease in costs related to drilling. The project integrated the visualization and analysis of a wide range of data types (e.g. geological data, reservoir data, well data, and production data) in one tool for cross discipline collaboration. Well planning was reduced from weeks to days because well engineers and geophysicists could bring their knowledge into a common work environment and understand each other’s criteria for designing an optimal well path. This enabled them to reach a design combining all requirements much faster than previous sequential processes. Additionally, the SHIVR software supported remote collaboration and Virtual Reality (VR) so that several instances of SHIVR could run at different locations in such a way that remote participants could take part. The field of Integrated Operations [5, 6] in the oil and gas domain addresses how to tightly integrate the versatile operations and disciplines needed for oil extraction. Information technology is used to increase collaboration between disciplines such as drilling, production and reservoir management. Also, remote collaboration is facilitated by continuous surveillance and video links to improve the communication between offshore oil platforms and land-based offices. Integrated Operations are particularly useful for both oil and gas as well as mining facilities far from centres of population which also may have hostile geographical conditions [7]. Oil companies have made significant efforts and investments in integrated operations. This has led to reduced costs (e.g. reduced personnel requirements on offshore installations) and more efficient work processes.

Computer-supported cooperative work for research teams. Several works look at CSCW for research teams. Young and Lutters [8] presented a study on interdisciplinary research in Land Change Science (LCS). LCS studies human use of land and is an interplay between various fields ranging from politics to biosphere and atmosphere sciences. The work discusses how standardization of data and procedures is successful in more homogeneous fields of study where data is comparable, but is challenging in broad fields such as LCS which makes it difficult to create tools that foster collaboration. They found that positioning LCS research and geographical observations into a GIS system, worked as a common denominator for the field, and created a GIS tool which shows areas and points on a map where LCS metadata exists. The system supported text search on the metadata and statistical calculations for comparing LCS studies. Similarly, in our work, we found that a GIS system enriched with metadata worked as a good common denominator for the field of CCS. However, we had additional 3D subsurface data to be visualized.

In The Upper Atmospheric Research Collaboratory (UARC) [9], instrument sites situated in remote areas such as on Greenland were used to study upper atmospheric events, for instance solar winds. As such sites are difficult to reach, tools for access to the instruments were made which enabled interaction and collaboration over real-time data. To design the system, laboratories and instrument sites were visited to collect use cases from the scientists about the details of their practice. The goal was to allow an already collaborating community to work together more flexibly. The technology used was Java applets accessed through Web browsers.

The Collaboratory for Multi-scale Chemical Science (CMCS) [10] is a collaborative study of combustion research. Combustion processes are complex multi-scale phenomena spanning more than nine orders of magnitude in length scales. Their work presents tools for presenting provenance graphs on the data, search tools, annotation tools, access to chemistry databases, 3D visualization of molecules as well as chat and discussion functionality. Their system was developed through an iterative development and deployment process driven by guiding use cases and feedback from pilot user groups.

The Biological Sciences Collaboratory (BSC) [11] is a Web-based collection of collaborative tools for structural biologists. It enables the sharing of biological data and analyses through capabilities such as metadata capture, electronic laboratory notebooks, data organization views, data provenance tracking, access to protein and genome databases and starting networked supercomputing jobs.

The Elettra Virtual Collaboratory (EVC) [12] supports scientific collaboration in experiments with ultra-bright light sources. The main focus of the collaboratory is to provide remote access to expensive and hard-to-duplicate equipment. EVC is divided into three phases: data collection where collaborators can access the experimental station equipment, data analysis, where collaborators can share experimental data and run scientific programs to analyze those data, and experiment standby, where only sharing of data is allowed.

Jacoby et al. [13] present a cloud based 2D map visualization system for exploring algae blooms by running user defined image processing scripts on satelite imagery. They describe the steps taken when transforming a workflow consisting of a collection of scripts that initially had to be run locally, to a cloud based solution. They first containerized the scripts so it would be simpler to install the workflow on a local computer. Then these containers were made to run on a high performance computing cluster within their organization. Finally they adapted and moved the containers to run in the cloud on Amazon Web Services. They compare the tradeoffs for each of these four approaches for deploying and running their software.

These collaboratory tools have tailored functionalities for their respective domains, and can not easily be adapted for the CCS domain. We are not focusing on sharing of instrumentation, or access to generic and structured databases on e.g. genomes and molecules, but sharing of site specific data and in particular, visualization of 3D subsurface geodata. Several tool have integrated chat, groups, collaborative writing, videoconferencing and notebooks. Today, professional standalone tools offer this functionality and we therefore decided not to integrate such functionality into our work.

A review of collaborative geospatial tools for environmental challenges was performed by Palomino et al. [14]. They look at thirty-one existing collaborative geospatial tools and apply a cluster analysis to create a typology of these tools. Interestingly, none of these tools support visualization of subsurface data which is needed for CCS research. This is a strong indication that collaborative tools for subsurface geoscience is lacking and underlines the relevance of our work.

Using visualization to support collaboration. Visualizing data can be an effective and intuitive way to communicate information within and across disciplines. It is a natural way to achieve collaboration. In 2011, Isenberg et al. [15] classified collaborative visualization as an emerging field. In their work, they reviewed previous definitions of collaborative visualization and provided this encompassing definition: "Collaborative visualization is the shared use of computer-supported, (interactive,) visual representations of data by more than one person with the common goal of contribution to joint information processing activities." The importance of collaborative visualization in the field of geoscience has been demonstrated in two works by MacEachren et al. [16, 17] where they show how visualization can support construction, communication and revision of geoscientific knowledge. Kirby and Meyer [18] describe in their work about visualization collaborations how visualization researchers help domain experts organize, categorize, present, and explore their data.

Evaluating computer-supported cooperative work. Many of the CSCW tools described above were not evaluated, or only lightly evaluated. This is not uncommon for CSCW tools as it is challenging to design and evaluate them.

Olson et al.[19] list up outcomes that are indications of a successful CSCW tool. Examples are new scientific discoveries, new modes of working, more papers made, less duplication of work, positive attention from the public, new funding initiatives, new software built and new collaborations formed. The paper also describes factors that increase the likelihood of success, and this includes the degree of common ground between the groups such as whether the groups share the same vocabulary, the same management, or the same working style. In addition, agreement among participants as to what platform to use and if technical support resides at each location increase the likelihood of success. Bos et al. [20] state that for distributed research centers it can be challenging to accelerate collaboration through groupware due to the difficulty to communicate knowledge (as opposed to data), and in particular cutting edge knowledge. It can also be a problem that scientists like to work independently. This might be why groupware to a higher degree has become standard in e.g. Integrated Operations than in research. Sedlmair et al. [21] give very concrete and useful advice for designing successful groupware. Their focus is on how to effectively apply visualization techniques to specific application areas. The paper describes 32 pitfalls that should be avoided to produce a useful tool. Many tools have been designed that ultimately failed due to going into some of these pitfalls and they give some examples of this. We believe this paper is not well known in the field of CSCW and provides valuable advice in the process of developing groupware in general.

As related work show, there are challenges in creating successful groupware for research, which we also experienced in our work as described further in the chapter.

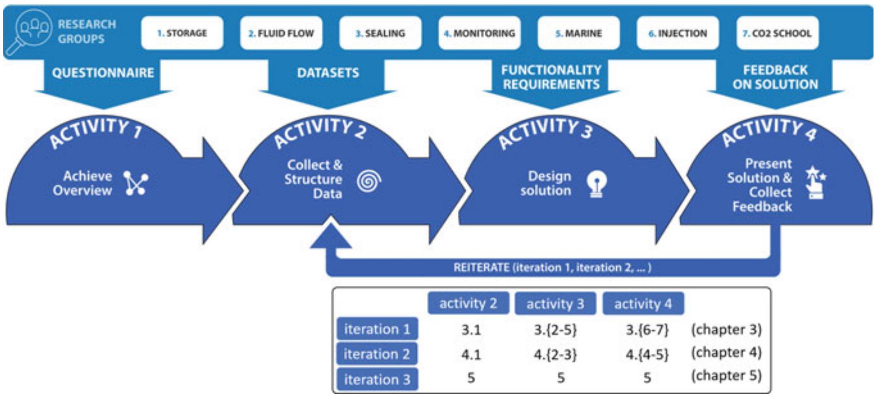
1.2 Systematic Approach

To guide the design of the CCS groupware, we obtained requirements from the SUCCESS research groups through a questionnaire, interviews and by collecting the research data they were working on. Achieving an overview of the research of several research groups, each with different scientific goals, geographical locations and datasets was challenging. Therefore, inspired by ISO 9241-210: Human-centred design for interactive systems [22], we developed a systematic approach consisting of four activities organized in a workflow as shown in Fig. 1. In Activity 1 we gained an overview by conducting a questionnaire to understand the degree of collaboration in SUCCESS and to get feedback on our idea of creating a “virtual CO_2 lab”. In Activity 2 we performed interviews and collected data, research questions and groupware functionality requirements from the SUCCESS research groups. With the help of this input, we designed a groupware solution in Activity 3. When software had been established, we received feedback in Activity 4 by making our solution available to the research groups and by giving demonstrations, talks and posters. The feedback was then analyzed so that the solution could be adjusted and improved. The feedback was also meant to identify missing datasets that should be included in the software database. Data collection and design was therefore reiterated by going back to activity 2, in effect starting a new development iteration. Parallel with this process, we had regular steering committee meetings with representatives from the research groups for receiving advice and feedback. We performed three iterations in the workflow; described in three separate subchapters. In the first iteration we tried out existing geoscientific software supporting 3D covisualization used together with a conferencing application. In the second iteration, we produced a custom-built lightweight web application which only supported 2D visualization. Finally, in iteration three, we created a web application supporting 3D covisualization inspired by the most useful features from the two previously evaluated solutions.

2 Achieving Overview (Activity 1)

The groupware team performed a questionnaire designed to gain insight into the degree of collaboration within and across the research groups and to get feedback which could guide our efforts when designing the groupware. The questionnaire was made as a web form and sent to 78 researchers constituting the core research personnel in SUCCESS. The respondents specified information about themselves in terms of their name, university affiliation, academic position (Ph.D. student, Post-doc or researcher) and the main research group they belonged to. To map out the collaboration between the researchers, each respondent listed names of who they had made papers, abstracts, presentations or proposals with (i.e. strong collaboration), who they had performed scientific research or had scientific discussion with (i.e. a weaker and undocumented collaboration), and who they had not, but would like to collaborate with. Review of the answers revealed that discerning between strong and weak collaboration provided no significant information. Therefore, we combined then for expressing collaboration in general. We received 22 answers. As some questions defined symmetric relations such as "to collaborate with", the respondents would in effect also be reporting on behalf of others. This resulted in a mapping of collaboration between 72 individuals. We analyzed the data using a graph visualization shown in Fig. 2 with the Tulip software [23].

In some cases, individuals were reported as collaborators that were not part of any of the seven groups. These were external industrial collaborators and have been tagged as "Industry/External" in the graph. As shown in Fig. 2, the questionnaire feedback indicates that collaboration takes place between all research groups and that no group is particularly isolated from the others. Naturally, there are more relations



	activity 2	activity 3	activity 4	
iteration 1	3.1	3.{2-5}	3.{6-7}	(chapter 3)
iteration 2	4.1	4.{2-3}	4.{4-5}	(chapter 4)
iteration 3	5	5	5	(chapter 5)

Fig. 1 Workflow (top): our approach consisting of four activities for creating a groupware system: (1) achieving overview, (2) collecting data, (3) designing a solution, and (4) presenting the solution and receiving feedback from research groups. Activities 2–4 were repeated in three iterations to gradually improve the groupware solution as shown in the framed bottom figure. Each of the chapters 3 to 5 correspond to one iteration, each chapter is divided into subchapters corresponding to activities

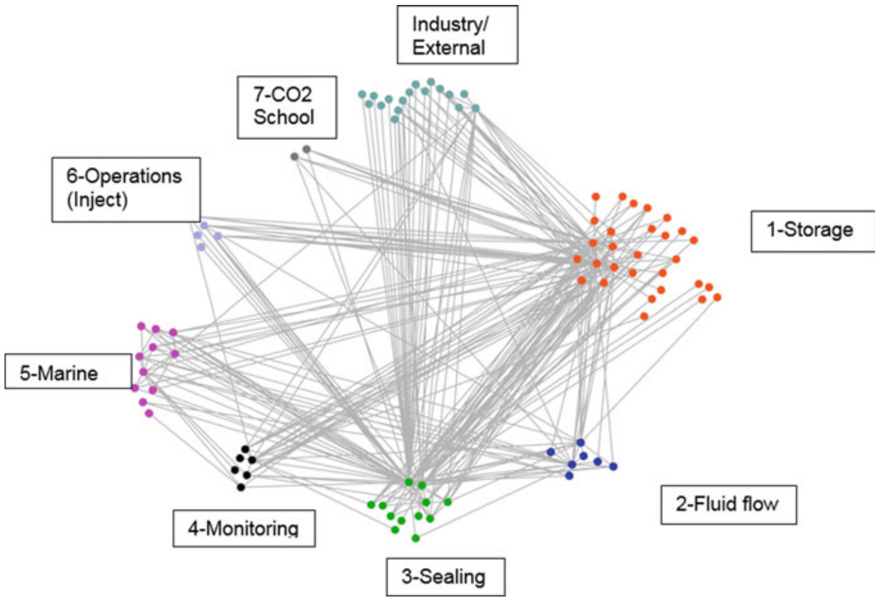


Fig. 2 Result of questionnaire revealing collaboration among researchers in SUCCESS. The points represent individual researchers and lines represent collaboration between researchers. Researchers have been grouped and coloured according to the groups they belonged to, shown in boxes

between large groups. Figure 3, right, shows a directed graph of who researchers are not, but would like to collaborate with. We observed that many arrows pointed towards external collaborators in industry (external/industry), indicating that the researchers wanted more collaboration with industry partners. We had no data on collaboration or collaboration wishes from the industry collaborators towards research groups as a complete overview of the external collaborators was lacking.

From Fig. 2 we can see that collaboration is present across all research groups, while Fig. 3 shows that some responders want more collaboration, e.g. between responders in group 1 (storage), and group 3 (sealing), and in general with the industrial collaborators. These observations support the validity of challenges R1 (remote collaboration) and R2 (cross-discipline collaboration). The community had a wish for more collaboration and groupware facilitating this could be a good catalyst. We also asked the following question to get feedback on challenge R3 (efficient data sharing and handling) "Do you think that you would collaborate more if sharing and integration of data with other researchers were easier? Please elaborate on this or suggest other mechanisms that could increase collaboration in SUCCESS". Answers ranged from supporting the challenge (e.g. respondent 1 and 2), to having a work situation where collaboration is not applicable (respondent 3): Respondent 1: "Virtual communication has to be improved to make regular meetings with researchers at different locations within and across research groups more easily accessible and efficient …*Easy and digital access to other peoples' data would definitely improve collaboration* and scientific output and increase the value of the data significantly.";

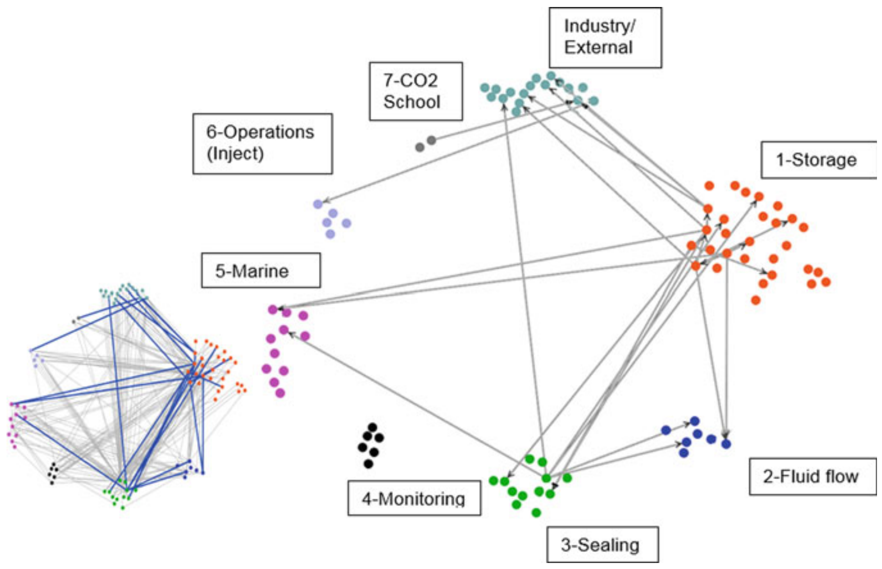


Fig. 3 Result of questionnaire revealing wished, but not existing collaboration among researchers. Left: wished collaboration relations shown as blue edges, actual collaboration shown as grey edges. Right: wished collaboration relations further detailed using directed arrows

respondent 2: "…we will do modelling of under-pressure in Adventdalen, and that will bring us in close cooperation with a number of other researchers (from other places). In other words, *working on the data from Svalbard brings us together*."; respondent 3: "I doubt it. My research has been quite far removed from what others are doing and *collaboration has hence been difficult*". The questionnaire response indicates that the SUCCESS researchers support the three challenges R1–R3. Further discussion of the questionnaire is performed in the discussion chapter.

3 Iteration 1: 3D Commercial Software as Groupware

After having gained an overview and verified the challenges, we transitioned to Activity 2 (Fig. 1) and started to collect data. Creating a solution for covisualization of spatial data from the ground up is complex and time consuming [4]. Therefore we first tried to identify if there was existing software that could be suitable to use.

3.1 Collecting and Structuring CCS Datasets (Activity 2)

In CCS research, data are collected in the field. The data are then processed and modeled to facilitate the interpretations. To plan for safe storage of CO_2 in the subsurface, researchers must access existing geoscience data from the potential storage site or measure it if it doesn't already exist. The data is then analyzed and modelled.

Analysis and modelling of the data can include seismic processing (e.g. stacking, and migration) followed by seismic interpretation to identify the boundaries of the reservoir where the CO_2 is planned to be injected and reservoir simulation for predicting how the CO_2 will flow and behave in the reservoir. The result of the analysis is new derived data. Both the measured and the derived data can be of value to other researchers.

There was no common database or overview of which data existed in the research groups. Therefore, our geoscience expert visited each group and talked with the group leader to understand the topics they were working on and to bring back the data they had. Each visit would typically last between four hours to one full day. Subsequent information exchange took place to get all data.

Birnholtz and Bietz [24] state that developing an effective CSCW system requires a good understanding of the use of data in practice. They divided the problem of sharing scientific data into three categories: willingness to share, locating shared data, and understanding the maturity, reliability and context of the data. For addressing these issues, the geoscientist in our groupware team visited each research group and performed interviews to acquire an overview of research questions in SUCCESS and collect the data associated to the research. As described by Bietz [25], to enable sharing of data e.g. through a common software interface, it is advantageous to standardize the description of the data by creating a common metadata format. During the visits, it was found that data exist in highly varying forms. Examples were mathematical models, descriptions of laboratory experiments, studies of chemical reactions as well as geological and geophysical data collected from different sites. Creating a unified and coherent software platform covering all these data types would be challenging. As data visualization is important in CCS research, we categorized the data in three classes according to which degree the data can be positioned in 3D space. The classes were spatial data, which has a well-defined geographical position, semi-spatial data, which is also spatial but can either not be associated with a specific geographical position or represents hypothetical unverified scenarios for specific geographical positions, and non-spatial data which is data with no positional information. The central types of spatial data that we identified during data collection are topography, 2D cross-sections, wells, 3D volumes, reservoir grids and separator surfaces. Topography describes the shape of the landscape. 2D cross-sections show measurements into the depth along lines on the top surface and are displayed as 2D images, see Fig. 6 (Seismic + resistivity slice). Wells are measurements along the path of a drilled well (a 2D curve). Cartesian 3D volumes are measurements taken at regular intervals in a 3D cuboid area. Reservoir grids are simulated values at irregular intervals in a 3D volume. Separator surfaces define the boundary between different rock types or describe faults. These types are general and also used in other geoscience domains such as oil and gas exploration. We limited our focus to spatial data as it has well defined extents and positions relative to each other and therefore naturally lends itself to 3D visualization and covisualization with other spatial data. However, by being able to display additional textual information for the spatial data, it would be possible to create a link to e.g. the storage location of related semi, and nonspatial data.

We found that the spatial data in SUCCESS was mainly focused around three CCS sites. Two of them, the Snoehvit field and the Sleipner field were operated by Equinor and were subsea CO_2 reservoirs related to offshore production platforms. As collecting data from under the sea bed is more challenging (except for seismic data) compared to on land, the field had a lower diversity of data types. Also, since the field was in commercial operation, much of the existing data had usage restrictions. The third site, the Longyearbyen CO_2 Laboratory (LYB) situated at Svalbard was noncommercial and included a rich and varied set of available data. In addition, several SUCCESS research groups worked with these data (e.g. see comment by interview responder 2). Another respondent stated "Working on one specific case will increase collaboration. If all partners in a case study try to solve a rather clear problem, then the collaboration may happen much quicker and more effective.". For these reasons, the data from LYB was chosen to be integrated into the groupware. We decided to first attempt to create a working solution for this subset, which if successful could be expanded with more data, rather than attempting to create a general solution supporting all data at once (Fig. 4).

We structured the collected research data from LYB in a Microsoft Excel table with one row assigned for each dataset (see Fig. 5), where each column described different attributes of the dataset. The attributes were data-type (a short description of what data it is); category (geological, geophysical or environmental); short description of the data (info); storage location of the data (folder); file names of the data, reports or publications (additional_files); data format (format); date of measurement; geographical start position (start_n, start_e); end position if applicable (end_n, end_e); institute that owns the data (owner); name and email of contact person that produced or knows the data well (contact_person, email); link to a relevant website for the data (website); whether the data has third party juridical confidentiality limitations, e.g. from an oil company (confidentiality) and if the research institute that manages the data is willing to share it with others (sharing).

We chose to use the Excel format as basis for the database since it is human-readable and can be sent around to researchers as an overview of the data, as well as updated by them. When adding datasets, researchers would also send accompanying files containing the data itself. We experienced that the Excel format was easily

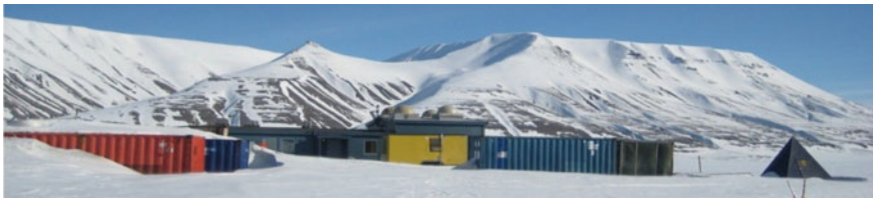
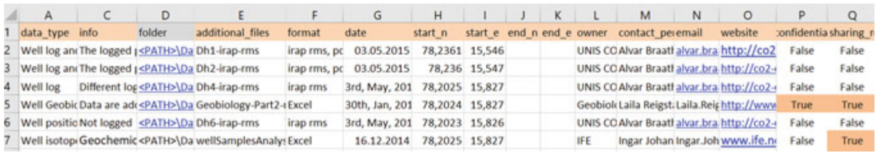


Fig. 4 UNIS Science Park in Longyearbyen. Site for assessing where CO_2 can be injected into a sealed subsurface formation consisting of porous sedimentary rocks. The black pyramid structure to the right shows the top of a well (Dh1)



	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	data_type	info	folder	additional_files	format	date	start_n	start_e	end_n	end_e	owner	contact_pe	email	website	confidentia	sharing_r
2	Well log an	The logged	<PATH>\Da	Dh1-irap-rms	irap rms, pc	03.05.2015	78,2361	15,546			UNIS CO	Alvar Braatl	alvar.bra	http://co2	False	False
3	Well log an	The logged	<PATH>\Da	Dh2-irap-rms	irap rms, pc	03.05.2015	78,236	15,547			UNIS CO	Alvar Braatl	alvar.bra	http://co2-	False	False
4	Well log	Different log	<PATH>\Da	Dh4-irap-rms	irap rms	3rd, May, 201	78,2025	15,827			UNIS CO	Alvar Braatl	alvar.bra	http://co2-	False	False
5	Well Geobic	Data are adc	<PATH>\Da	Geobiology-Part2-	Excel	30th, Jan, 201	78,2024	15,827			Geobiol	Laila Reigst	Laila.Reig	http://www	True	True
6	Well positic	Not logged	<PATH>\Da	Dh6-irap-rms	irap rms	3rd, May, 201	78,2023	15,826			UNIS CO	Alvar Braatl	alvar.bra	http://co2-	False	False
7	Well isotop	Geochemic	<PATH>\Da	wellSamplesAnaly	Excel	16.12.2014	78,2025	15,827			IFE	Ingar Johan	Ingar.Joh	www.ife.n	False	True

Fig. 5 Overview of the collected data organized in an Excel table. Each row represents a spatial dataset having a 3D position on Svalbard; columns describe properties of the datasets

extensible and did not impose strict rules on how to organize the data. It allowed us to evolve the representation from an unstructured human readable one towards a structured, both machine, and human readable format.

3.2 Groupware Functionality Requirements (Activity 3)

Visualizing the different research groups' data at correct geographical positions makes data discovery and identifying correlation between datasets easier, while visualizing derived data gives the research groups an opportunity to communicate their research results. This visualization would be performed in the groupware. Achieving it required functionality for tasks such as data import, large data handling, interactive covisualization and a mechanism for remote collaboration. Based on dialog with the SUCCESS researchers during activity 1 and 2, we elaborated on R1–R3 and created the description shown in Table 1.

3.3 Selection of Software to Evaluate

Using the requirements for the groupware, we started evaluating which existing geoscientific software we should then present to the SUCCESS researchers for a more thorough evaluation. We initially considered a broad range of software, this was based on feedback from SUCCESS researchers during the interviews in the data collection phase, from feedback in steering committee meetings, from searches with relevant keywords on the web, and from software described in CCS literature.

Nimtz et al. reported in 2010 [26] that an ideal tool for modelling and visualization of the data from CO_2 storage projects does not exist. The following evaluations of existing software indicates that this is still the case.

From information provided by software vendors on their website and in technical notes and by from published evaluations and use cases, we narrowed down 17 tools to three, Petrel v2015, SKUA v14.1 and ENCOM PA v12 (now PA Explorer), as most suitable as the groupware. These were the newest versions of the software at the time of evaluation. We then had to exclude Encom PA from our evaluation due

Table 1 Groupware functionality requirements

R1: Remote collaboration
(1a) **Support for simultaneous users**. Groupware for scientists to simultaneously work in, should support session sharing through e.g. screen and sound sharing
(1b) **Ease of install and use**. Researchers that are able and willing to use the system is a requirement for achieving remote collaboration
(1c) **Ease of acquirement**. Cost of software must be within budget bounds. Several of the commercial software vendors offered affordable academic licenses. Other vendors only offered licenses tailored for high revenue oil and gas companies
R2: Cross-discipline collaboration
(2a) **Covisualization**. Covisualization of data from different collaborators can increase collaboration and may reveal unknown correlations between spatially overlapping data
(2b) **Scale ranges**. CO_2 data span large scales ranging from nanometers to kilometers. Support for multi scale visualization is important
(2c) **Spatial navigation**. Efficient spatial navigation is crucial for being able to browse the data. Examples are jump-to-object functionality allowing the user to automatically navigate to an object by selecting it from a textualist,and the ability to define flight paths that can later be used in collaborative sessions
(2d) **Interior visualization**. If the data contains solid 3D models, or if the integration of multiple data prevents seeing certain parts of models, it must be possible to cut through the model or to hide occluding geometry
(2e) **Interactive annotation**. Interactive annotation of data by defining curves and surfaces to e.g. indicate boundaries and layers is useful during discussions to express ideas
(2f) **Interactive performance**. Lagging user interfaces and low framerates reduce usability and perceived quality of software. Interactive performance is challenging to achieve, particularly for visualization of large datasets, as it requires raw compute power and/or advanced visualization algorithms
R3: Efficient data sharing and handling
(3a) **Import**. It must be possible to import a wide range of data types
(3b) **Export**. It must be possible to export the data being shown and it must be possible to take screenshots for e.g. use in reports
(3c) **Large data**. Improved measurement technologies have enabled a growth in data acquisition. Large datasets represent challenges related to storage, processing and visualization both in terms of available memory and available processing power (which is related to point 2f)
(3d) **Ownership**. For collaboration across institutions, managing data ownership and rights for use of data is important. It is time-consuming and can also be juridically challenging to assess the data use rights for other parties
(3e) **Linking data to other data and documents**. It should be possible to create looks between data, e.g. from graphical objects to relevant documents describing them, or from graphical objects to other graphical objects. A specific requirement by researchers was to be able to open a corresponding well core photo by clicking on a part of a well in the 3D view

to a licensing cost beyond the project budget. The remaining software was evaluated by integrating a representative selection of large, multi-scale and multidisciplinary data from LYB.

Our evaluation revealed strengths and weaknesses in both tools with respect to our needs. For instance, while we experienced volume rendering capabilities to be better in Petrel, interactivity during surface rendering was slightly better in SKUA. In our datasets, we had many 2D surfaces and vertical sections but little 3D data that required volume rendering, which is one reason why we found SKUA marginally better suited for our purposes and ended up choosing it. See Fig. 6 for renderings of LYB data in the SKUA software.

We realized after the evaluation that the evaluated software is highly specialized, whereas our functionality requirements are more of a general art targeted towards efficient presentation, sharing and browsing of various data types.

It was challenging to perform a fair evaluation of all available software due to several factors. Industry-grade geoscientific 3D visualization software typically contain a very large amount of functionality. This functionality is hard to get an overview of and master in limited time. Another factor was costly licenses, although some software suites had reduced price for academic use. A third factor was the availability of extension plugins with associated fees, sometimes from third party venders. These could potentially improve a software suite; however this is costly and time consuming to evaluate. In general, we experienced that it was hard to get an overview of the functionalities offered in geologic software.

3.4 Description of 3D Platform

After having chosen SKUA as the groupware, we integrated the majority of the collected LYB data into it. SKUA is a stand-alone application with an interface consisting of a graphical view and a user interface panel (Fig. 8 right). Inside the graphical view, the user can navigate around in the 3D world. The user interface panel shows a textual list of all graphical items, from which each can be selected and e.g. toggled on or off.

In Fig. 7 center, a well log visualization is shown. This consists of different petrophysical measurements along the well called attributes. SKUA allows to map one attribute to a color shown on the well cylinder and additional attributes that are shown as graphs next to the cylinder. Here the rock type (lithology) is color coded and visualized on the well cylinder and electrical resistivity, gamma radiation and seismic properties are shown as vertical graphs along the cylinder. The rock type is derived from collected well cores during drilling. These cores have also been photographed and are part of the database. Three well core photos are shown with arrows indicating their positions in the well. Unfortunately, we were not able to integrate these photos into SKUA.

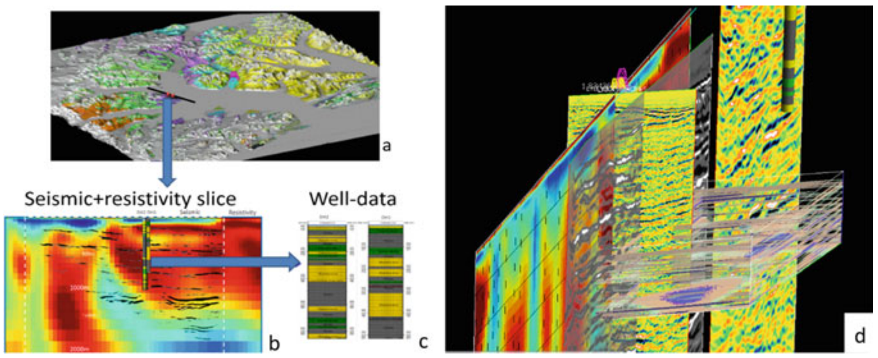


Fig. 6 **a**–**c** Multiscale covisualization of geological and geophysical data. **d** Different modalities visualized in relation to each other

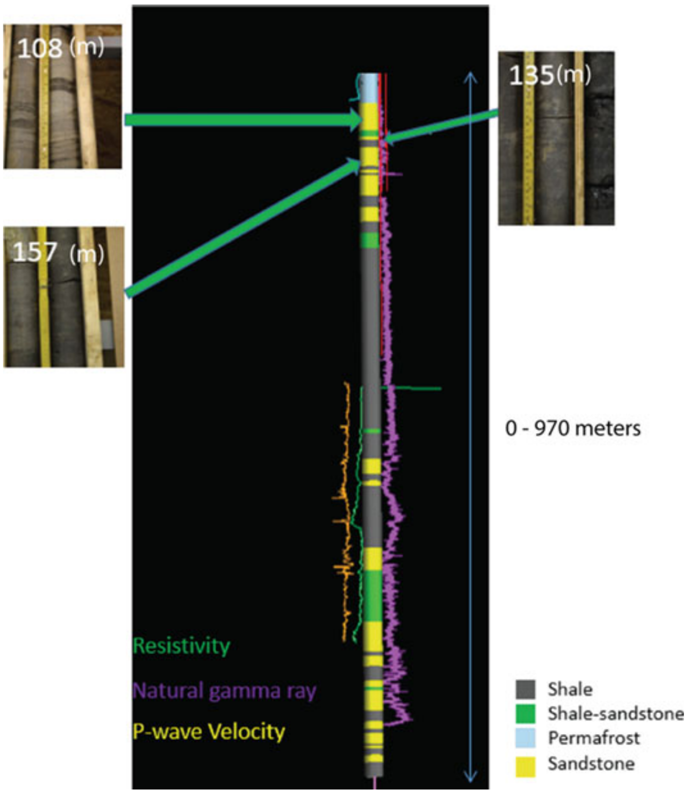


Fig. 7 Numbered images are well core photos and numbers represent the depth of the core samples. Center image shows lithology variations along the well (see legend bottom right), vertical colored lines show variations of the P-wave velocity, natural gamma ray and resistivity along the well (legend bottom left)

3.5 Evaluation of 3D Platform

We evaluated the solution according to the criteria in Table 1 and have summarized the results in the following paragraphs. The results are also summarized in the column named "3D" in Table 2.

Remote collaboration (R1). SKUA acted as a stand-alone application that had been installed on a powerful computer which also contained all the data. The application was therefore not available on the computers of the SUCCESS researchers. Support for simultaneous users (1a) was established by using an application enabling generic sharing of the desktop of the computer SKUA was installed on. Few existing 3D visualization tools support remotely connected simultaneous users directly. Therefore, we used the software Virtual Arena (by Visual Solutions). Virtual Arena makes it possible to share applications over internet by letting remotely connected users take control of the mouse and keyboard while receiving a video stream of the application. Only one user at a time can be in control and can transfer the control to another user. In addition, all participants in the session can be captured with camera and microphone and displayed. Figure 8 shows a screenshot of a collaborative SKUA session as seen from one of the collaborators. Concerning ease of install and use (1b), the sharing application was easy to install. It required a download followed by a few simple steps. Installing SKUA required more steps, including setting up a license server, but this was a one-time operation performed by the groupware team. The software contained a lot of functionality, much of which was not relevant to the users. This resulted in a complex user interface which made it hard for the researchers to use the software. This was solved by having us guide the researchers when they were using it. The external application that enabled screensharing of SKUA was affordable and easy to acquire (1c) but SKUA itself was costly, which may be a barrier when on a low budget. Overall, the criteria for (1) were technically covered as functionality was present for making remote collaboration.

Cross-discipline collaboration (R2). Covisualization (2a) was handled well; for examples see Fig. 6d. Scale ranges (2b) were handled well as demonstrated in Fig. 6a–c. Spatial navigation (2c) was possible but cumbersome in SKUA as e.g. "jump to location" functionality was lacking. Instead, this was handled by zooming out for a full overview and toggling the object of interest on and off from the list of all 3D objects, to identify its location. We were also not able to define flight paths. Interior visualization (2d) was achieved by using clipping planes, semitransparency and toggling of geometry. The software did not support interactive annotation (2e). Interactive performance (2f) was achieved for moderately sized geometry, however we experienced a framerate lag when showing topography of large detailed areas. To address this, we made a low-resolution topography version for the full area around LYB, and a detailed high-resolution topography confined to a small area around LYB which we could switch between. Such custom terrain resolution handling would not have been necessary if the tool had supported large terrains using e.g. level of detail techniques [27]. However, the versions of SKUA and Petrel we tested did not support this. Overall, the criteria for 2 were covered well except for 2c and 2e. Navigation

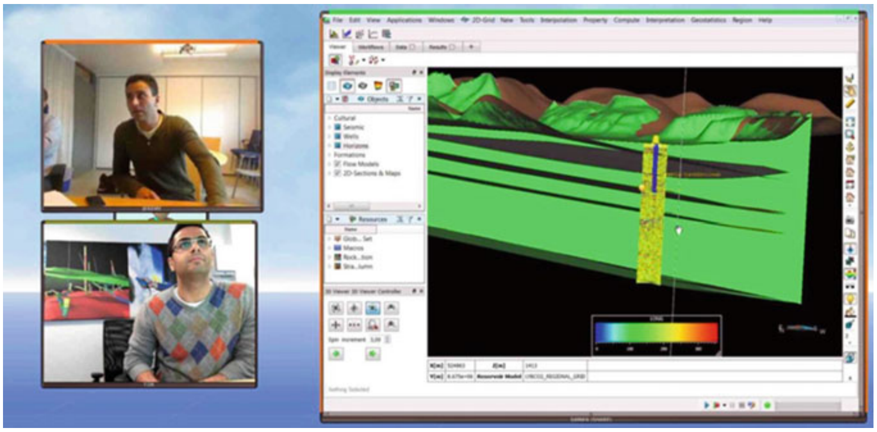


Fig. 8 Remote collaboration between two participants using Virtual Arena sharing software. To the left, the participants; to the right, the shared SKUA application

(2c) was problematic and interactive annotation (2e) was not possible, but likely not critical as researchers could instead express their ideas orally and by pointing. In general, the tool should be able to be used to interactively covisualize data for cross-discipline collaboration.

Efficient data sharing and handling (R3). Data Import (3a) was handled well, except for nonspatial data such as documents. However, we could refer to nonspatial data by filling in text fields associated to the visualizations. Export (3b) was handled, but not optimally as data had to be saved on the computer running SKUA, then manually sent to the remote user using e.g. email. Large data (3c) were handled moderately well; we experienced no limitations on file sizes, but SKUA did not support advanced memory management techniques for supporting large data. Ownership (3d) and linking data (3e) was solved by a functionality in SKUA where one could add textual metadata to the visualizations which could be displayed in the user interface. Getting access to this link was however not straightforward as it was hidden in a dialog that required nonintuitive GUI navigation to access. Overall, the criteria for (3) were not covered too well. Export of data (3b) to remote participants was cumbersome. Getting access to data ownership (3d) and linking (3e) associated with the datasets was possible but hard. Also, clicking on a segment of a well and showing well core photos of that segment was requested functionality from researchers, but was not supported. Although features were present for data sharing and handling (3), it was inefficient to access some of the features.

Summary of all criteria. In summary the technology is in theory present in the evaluated solution to cover all criteria to a sufficient degree. Next, we describe the community response to the SKUA solution.

3.6 Response on the 3D Platform (Activity 4)

How response was collected. Going into activity 4 in our methodology, we presented our solution and collected feedback from the research groups. This was achieved by inviting the research leaders of the groups and all interested researchers to our collaboration room. In total 9 people showed up and at least one representative from each group was present. We demonstrated the solution by presenting the incorporated data in SKUA. The demonstration lasted for about 30 min followed by about one hour with interactive discussions around the datasets. The reactions were overall positive.

New collaborations and new data. The meeting sparked discussions between e.g. the research leaders of the sealing research group (3) and the marine research group (5), as the covisualizations made them discover that each had measurements along wells close to each other that were not integrated into SKUA, and which should be exchanged between the groups. The meeting also resulted in follow-up remote collaboration sessions with a researcher who was in possession of additional data at the location where the SKUA database contained multiple seismic sections and well logs. The researcher wanted to covisualize and interpret the data together. In addition, the visualizations and the data collected resulted in starting a new project on visualizing volumetric CT scans of parts of the well logs.

High interest in database, low interest in solution. Except from the collaborations described above, there was less follow-up activity around the proposed solution after the meeting took place than we anticipated. Most of the SUCCESS researchers did not request to use the solution. On the other hand, several researchers were interested in the Excel overview of the geoscientific data. We shared the overview of the data by sending the Excel table to those who requested it, which often resulted in a subsequent request for either parts of the data itself, or for screenshots of visualizations of specific datasets. The researchers then expressed a wish to access both database and visualizations without having to contact the groupware team each time.

Possible reason for low interest in solution. We believe there was low interest in the solution because the threshold for use (i.e. requirement 1b—Ease of install and use) was too high. It was required to make appointments for running the software and an install of software for remote users (VirtualArena). Secondly, the solution lacked functionality for exporting data directly to remote users, as the software with the data and the user were on separate computers (requirement 3b—Export).

Possible reasons why we failed to anticipate the low interest. We underestimated the high importance of requirement 1b (Ease of install and use) compared to the other requirements. The software was evaluated by a geoscience expert familiar with complex geoscientific software. Therefore, the high complexity as seen from a user may not have been identified. In addition, as the software was installed on the expert's computer, the steps of booking a meeting and installing the remote software, which a new user would have to go through, may also have been given too little emphasis.

Changes we planned to make according to the feedback. The 3D solution's complex user interface and lack of accessibility (1b) resulted in a too high threshold to using it, also data could not be easily downloaded (3b). We were not able to solve these issues in the existing solution, so we started to create a new solution focusing particularly on these two issues. In addition, as there was a high interest in the Excel sheet which gave an overview of all data, we also tried to allow for good overview of all the data in the new solution. Such a solution could replace the need for the 3D solution in certain cases or allow for both solutions to be used in conjunction.

3.7 Updated Version of Tested Software

Several newer versions of SKUA have been released with improved functionality since we performed our evaluation. In the latest version (SKUA-GOCAD 18), the new functionality include support for area of interest polygons which may address requirement 2e (interactive annotation), support for large data and utilization of high-performance computing platforms which may address requirement 3c (large data), as well as the ability to run the application as a service on the cloud which may partly address requirement 1b (ease of install and use). The application is still feature rich and therefore complex to use, so we believe that the "use" part of requirement 1b is still not sufficiently supported. In addition, the software is hard to customize and costly. SKUA is powerful geological interpretation software while our requirements point towards a lightweight data browsing tool, which still results in a mismatch between what SKUA offers and what we want for the groupware.

4 Iteration 2: 2D in Web Browser as Groupware

Based on the feedback from the 3D solution, we started a new iteration to address the identified weaknesses. First, we collected the new data that had been identified (activity 2). Then for activity 3, our idea was to implement an easy-to-use web application with direct download capabilities. This could reduce the threshold for researchers to use the software and allow them to download data directly. For giving the users an overview of the data, we created a 2D map visualization of data positions shown next to a textual list of all datasets currently in the map view. This 2D tool could be use instead of, or before using the 3D tool. It could for instance be used to detect, by browsing the data, who one wants to initiate a collaborative 3D session with later.

4.1 Collection of More Data (Activity 2)

As stated earlier, during demonstrations of the 3D groupware, discussions between the sealing research group (3) and the marine research group (5) revealed that data was missing. The missing data refers to measurements along wells close to each other which is particularly interesting as they could be combined and correlated. In addition, volumetric CT scans of parts of the well logs were identified and collected.

4.2 Description of 2D Web Platform (Activity 3)

Our 2D map visualization produces an overview of the localizations of the data, as shown in Figs. 9a and 10. The solution was made using free and open software. We used a Python Apache server and JavaScript client code. We made support for automatically populating the database by importing the Excel file. This was performed by creating a script on the server that read and transformed the Excel file and the data files it pointed to, into an SQL database. This allows multiple datasets to be imported automatically at once instead of manually importing each dataset through a complex user interface, requiring different actions for different datasets, as was the case in SKUA. Also, we could easily update the script whenever we added new attribute types in the Excel table. Map visualization was implemented using the Leaflet library [28], which supports texture overlays, vector graphics maps and Web Map Services [29].

Positions of datasets are shown with a marker in the map view (Fig. 10a). When hovering over a marker, a short textual description is shown. When clicking on a marker, the dataset at that position is selected, and the dataset attribute information is shown in a separate window including links to data files which can be downloaded (see Fig. 10b). For data measured along a line, such as cross-sections, the line itself is drawn as shown in black in Fig. 9a. Each dataset is also shown as an item in a list (Fig. 10a, left), allowing data to be accessed by two different means. The list shows all data visible on the map and is updated when the user zooms and pans the map. In addition, the list can be filtered according to the dataset metadata by using a textual search field. Overlapping data are indicated by displaying a number inside the position marker showing how many datasets overlap. If the marker is clicked, individual markers for each dataset are spread out in a spiral around the original position. The system can automatically zoom in on a dataset selected in the list. This makes it easy to navigate to datasets. Seismic sections and other data that can be represented as images are stored in image format, so they can be viewed directly in the browser (Fig. 9c). In addition the original underlying data can be downloaded for import to e.g. interpretation software. Several map providers can be chosen among (Open Street Map, Open Surfer, Norwegian government map, and Bathymetry sea depth map). Map overlays are supported from external Web Map Service data sources such as geology maps and hydrocarbon prospect borders. Thus, the use of open web standards allows us to load data from external sites and visualize the SUCCESS data in a map view together with e.g. bathymetry, geology and prospect borders.

The website makes it possible for scientists to explore and get overview of the datasets by e.g. searching for keywords in metadata or by finding data in proximity to the data they are working on, as well as identifying related articles and possible coop-

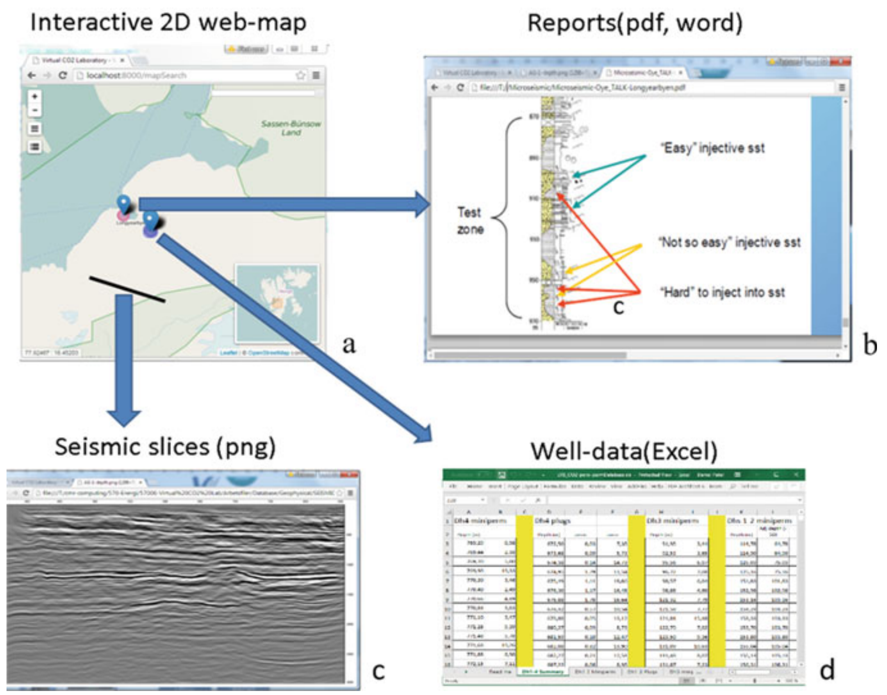
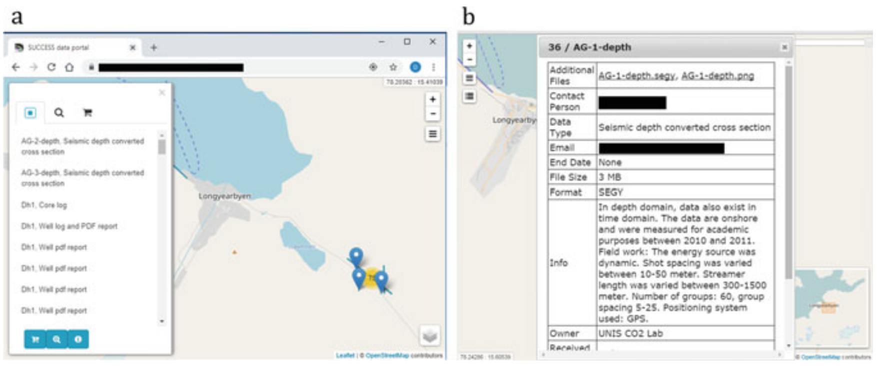


Fig. 9 **a** Data is annotated on a 2D map as points (e.g. wells), or lines (e.g. cross-sections). The corresponding data is shown directly in the browser such as reports (**b**), seismic cross-sections (**c**) or can be downloaded and opened in external software, such as well-data (**d**)



[redacted]

AG-2-depth, Seismic depth converted cross section

AG-3-depth, Seismic depth converted cross section

Dh1, Core log

Dh1, Well log and PDF report

Dh1, Well pdf report

Dh1, Well pdf report

Dh1, Well pdf report

Dh1, Well pdf report

36 / AG-1-depth

Additional Files	AG-1-depth.segy, AG-1-depth.png
Contact Person	[redacted]
Data Type	Seismic depth converted cross section
Email	[redacted]
End Date	None
File Size	3 MB
Format	SEGY
Info	In depth domain, data also exist in time domain. The data are onshore and were measured for academic purposes between 2010 and 2011. Field work: The energy source was dynamic. Shot spacing was varied between 10-50 meter. Streamer length was varied between 300-1500 meter. Number of groups: 60, group spacing 5-25. Positioning system used: GPS.
Owner	UNIS CO2 Lab
Received	

Fig. 10 **a** A scrollable list of all data visible on the map is shown in the left pane allowing the user to either select a dataset from the textual list or from the map. **b** When clicking on a dataset on the map or in the list, more information about the data shows up such as owner of data, email, data size, a short description, and a list of files associated with the data such as documents and measurements

eration partners. In contrast to the 3D solution, the 2D web solution does not require installation, and runs on any computer and OS having a web browser. It also runs on tablets and mobile phones. This may foster collaboration as it can be quickly accessed during meetings with other researchers. A modified version of this solution where data with ownership constraints has been removed is available to the public at the website URL success.webfarm.cmr.no and will be maintained until at least year 2023.

4.3 Evaluation of 2D Platform

We have evaluated the solution according to the criteria in Table 1 as described in the following paragraphs. This is also summarized in the column named "2D" in Table 2.

Remote collaboration (R1). Simultaneous users (1a) was not supported. There were no limitations on how many could access the web site at the same time, but the users were not able to communicate with each other. As the software existed as a web page, no install procedure was required, and use was simple (1b). Powerful graphics hardware was not necessary, and the solution worked on standard computers and on tablets and phones. The system was also simple to use compared to the 3D solution due to its limited functionality and adherence to a standard web user interface. No acquisition was required (1c) as operating systems come with preinstalled web browsers. Overall, the criteria for (1) were covered well except for (1a).

Cross-discipline collaboration (R2). 3D covisualization (2a) was not possible as 3D rendering was not supported. The position of the data on the map was shown, and for e.g. seismic cross-sections, a line was drawn which resulted in limited covisualization regarding positioning. Scale ranges (2b) were not supported since 3D covisualization was not supported. Spatial navigation (2c) was supported, but only in 2D. The user could easily pan around and zoom into the 2D map and could select and search for datasets from a list and center the map position of the selection. Interior visualization (2d) was not supported as only 2D data cross-sections are visualized and not all the datatypes. However, for the data that could be shown as 2D images, it was useful to be able to visualize each dataset individually. This was more cumbersome in SKUA. Interactive annotation (2e) was not supported. Performance was interactive (2f) as only basic visualizations were shown. Overall, the criteria for (2) were not covered well, however the functionality that was supported worked smoothly and was user friendly.

Efficient data sharing and handling (R3). Importing datasets (3a) was well supported. The Excel datasets file was read directly by our software and imported into the web solution, and any datatype with a geographic position could be uploaded. Exporting datasets (3b) was well supported; users could download data easily and directly to their computer from the website. Handling of large data (3c) is in general more problematic for web solutions as all data initially resides on the server and must be downloaded to the client and this may take time. Handling of ownership (3d) was very good. The ownership metadata (shareable, owner, restrictions) was easy to access. Linking data (3e) to publications or to other visualizations was handled well. By using URL links to data residing on our server, we were able to let the user download any type of data, and to display data such as image files and pdf documents directly in the browser. Publications that had no clear geographic location were positioned at the location of the SUCCESS administration in Bergen. This allowed us to add all reports produced in SUCCESS into the solution. Specific reports could be retrieved by using the text search field. Overall the criteria for (3) were handled sufficiently.

Summary of all criteria. This solution was primarily designed to address the shortcomings of the 3D solution without reimplementing what already worked in the 3D solution. It thus lacks the important functionalities of 3D visualization, covisualization and simultaneous users (Column “2D” in Table 2). However, the 2D solution has a user-friendly interface, is easily accessible as a web page and works on both new and older computer hardware. In addition, datasets can be downloaded from the web page easily. These factors may increase the likelihood that the tool will be used more than the 3D solution.

4.4 Response on the 2D Platform (Activity 4)

The solution was tested by the SUCCESS research leaders. Although the 2D solution lacked 3D visualization abilities, it was considered as more valuable than the 3D solution since it was easily accessible. Access to the data is now easier with the help of 2D platform. Researchers regardless of their locations and institution have access to data and results from other project members. This should facilitate communication across research disciplines within SUCCESS. The solution was rolled out and is now publicly available containing all the nonconfidential results of the SUCCESS project. For Norwegian state funded projects, the results must be made public by law, and this solution was selected as the means for making the SUCCESS data publicly available. The solution was unfortunately rolled out at the end of the SUCCESS project and the researchers did therefore not have much opportunity or motivation for testing it out.

4.5 Comparison of 2D Web and 3D Standalone Platform

The evaluation of the 2D and 3D solutions according to the requirements in Table 1 is summarized in Table 2. The 3D solution’s complex user interface and lack of accessibility (1b) resulted in a too high threshold to start using it, also data could not be easily downloaded (3b). This was the motivation for creating the 2D solution where

Table 2 Comparison of 2D web and 3D standalone solution

Collaborative Challenge	Requirement		3D	2D
R1: Remote collaboration	1a	Simultaneous	Y	N
	1b	Ease of install and use	Y->N	Y
	1c	Acquirement	P	Y
R2: Cross-disciplinary collaboration	2a	Covisualization	Y	P
	2b	Scale ranges	Y	N
	2c	Navigation	P	Y/P
	2d	Interior visualization	Y	N/P
	2e	Interactive annotation	N	N
	2f	Interactive performance	Y/P	Y
R3: Efficient data sharing and handling	3a	Import	Y/P	Y
	3b	Export	P->N	Y
	3c	Large data	P	P
	3d	Ownership	P	Y
	3e	Link data	P	Y

Y = Yes, N = No, P = Partly. Arrow in 1b and 3b represents that the evaluation was modified based on the response on the 3D solution

the threshold for use (1b,1c) and data export was improved. Other places where the 2D solution stood out positively was ease of navigation (2c), although limited to 2D. Also, data import (3a) was simplified by automatically parsing the Excel table. This eliminated the time-consuming import steps needed in SKUA and allowed for complete regeneration of the database in the case there had been changes on several datasets in the Excel table. Export (3b) was much simpler than for 3D as data was downloaded directly from the browser, also ownership information (3d) and links from data to other data and reports (3e) were supported and easily accessible compared to the 3D solution. However, the 2D solution also had shortcomings compared to the 3D solution. It did not support simultaneous users (1a) nor did it support cross-discipline collaboration through visualization of data (2a, 2b, 2d). These shortcomings made it hard to perform in-depth discussions and comparisons of data.

Although the 2D solution was considered most favorable by the SUCCESS leaders, using the 3D software was a necessary preliminary step for achieving overview and cataloging data. This involved extraction of geolocation of the data to understand the actual data content to correctly classify it and to evaluate its content, quality and maturity. In other words, a 3D platform for integration and hence better understanding of the multi-dimensional (2D, 3D and 4D) data may be required. The 3D solution resulted in new collaborations that may not have been found when showing the data in a 2D view as e.g. data overlap and correlation in depth cannot be detected in 2D. Therefore, using the 2D solution first to detect interesting data and then transition over to 3D could be viable way of combining the strengths of both solutions. However, this was not explored due to time constraints.

5 Iteration 3: 3D in Web Browser as Groupware

Neither the 2D nor 3D solution fulfilled all the requirements that were made. Since the 2D web solution has shown to be a promising platform, we thought that extending this solution to cover more of the overall functionality required, would be a good choice. An important question in this regard is how easy it is to support 3D subsurface visualization in a web browser. A possible extension of our 2D solution would be to display the 2D web in one window, and a synchronized 3D rendering mode could be shown in a separate window or tab when running on a graphics capable device. This question was attempted answered by two consecutive Master's students that we were supervising, and we report the results in this section.

For rendering 3D in web browsers, there exists several programming libraries. Web applications built on Google Earth [30], NASA World Wind [31] and Cesium [32], render globes with overlays and geometry on the globe surface and therefore seem like a good match, but unfortunately they do not support subsurface visualization. Ziolkowska and Reyes [33] note that these solutions "were not originally developed with the purpose of representing data below the Earth's surface". After testing several web-based virtual globe programming libraries including the ones described, we experienced that navigating below the surface of such globes is problematic, as support for this is not made. Adding geometry below the surface can be done, but the subsurface geometry becomes overdrawn by the globe geometry. Solving this problem would require access to low-level rendering functionality not clearly exposed in the software libraries. We found no suitable virtual globe programming libraries for web that could be extended to render subsurface geology.[1] A STAR report on web-based visualization from 2016 [35] shows a wide variety of visualization solutions available on web using WebGL and HTML5, but does not give examples of web based subsurface visualization. Due to lack of appropriate libraries, we created from scratch, subsurface visualizations by using the X3DOM open standard [36, 37]. X3DOM hides the details of graphics rendering in high-level, declarative XML syntax. Using X3DOM, we were able to make a proof of concept web demonstrator [38]. X3DOM contained versatile building blocks and was easy to use due to its declarative programming model. The visualizations we implemented were 3D terrain visualization with level of detail support [27], cross-sections and rendering of well data. See Fig. 11 for a visualization of these data types. Like the 2D solution, this solution was also built with dual indexing, either from graphics directly (by right clicking) or from a textual list of datasets that can be shown on the left side of the screen. The user can navigate around in the 3D topography and jump to datasets from a textual list. As data often overlap each other or obstruct view to data behind, we implemented functionality to toggle visualizations on and off by left clicking directly on the visualization, which was missing in SKUA. When toggled off, a red translucent sphere is shown instead, indicating that there is data at this point. Wells that have physical measurements are colored along the well path. A legend

[1] Cesium has since this work was done, implemented some support for subsurface visualization [34].

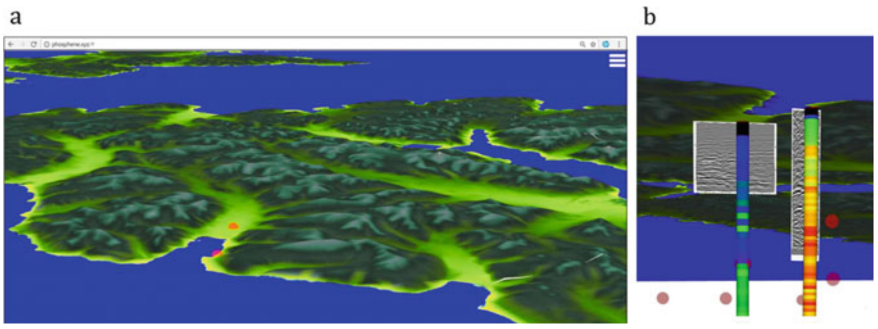


Fig. 11 Web solution with 3D capabilities. Left: **a** area around Longyearbyen. Right: **b** two seismic depth-sections and two wells. The color variation along the left well represents "self-potential" measurements while the right well represents "Gamma radiation"

describing the mapping between color and value can be shown by right-clicking on the well visualization.

This solution demonstrated the feasibility of rendering 3D subsurface data in a web browser without plugins. The solution was built on a web framework called Angular (version 4) [39] which we had some problems integrating with X3DOM. To overcome these problems, the solution became less modular than we had anticipated. We wanted to be able to define subsurface objects declaratively using HTML-like tags so that the 3D subsurface entities would be modular and easily reusable. In a follow up Master's thesis [40] we succeeded in this by using React (version 16) [41]. Compared to the 3D client solution, there will be limitations on the size of the datasets that can be interactively visualized at once due to varying rendering capabilities on client machines and low data transfer speeds over internet. However, the follow up Master's thesis demonstrated a simple technique for handling large datasets in web browsers by limiting the user to see subsets of the large datasets. For instance, a volumetric cube of byte-sized data values with resolution $1000 \times 1000 \times 1000$ would require the downloading of 1Gb of data, and would require processor intensive volume rendering for visualizing it. Letting the user instead only see a 2D slice of size 1000×1000 and allowing the user to interactively choose which of the 1000 slices to see would not allow for covisualization of all data within the volume simultaneously, but still give the user access and overview of the full dataset. This approach was demonstrated in the Master thesis. The code from these projects are available for download [42, 43].

5.1 *Results if the 2D Web and 3D Web Solutions Were Combined*

In the second iteration we made the 2D web solution which, compared to the 3D standalone solution had major improvements regarding ease of use (1b and 1c), and data access (3b, 3d and 3e). On the other hand, the 2D solution had no 3D visualization capabilities. If there would have been time for another iteration we would, based on the knowledge gained, first have combined the 2D and 3D web solutions we implemented by e.g. showing a synchronized 2D and 3D view side by side. The added 3D rendering capability would improve several of the 2D web solution scores shown in Table 2. Requirement 2b (Scale ranges), 2c (Navigation), 2d (Interior visualization) and 3c (Large data) would now be fulfilled to a higher degree. Still unfulfilled would be 1a (Simultaneous) and 2e (Interactive annotation). Point 1a could be addressed by integrating collaboration directly into the web solution using the open Web Real-Time Communication (WebRTC) API [44]. WebRTC has been successfully demonstrated in a collaborative web-based environmental data visualization solution [45] where it is used to synchronize 2D views between web users at different locations. WebRTC has also been used for synchronizing the viewing of 3D X3DOM objects [46]. Then only requirement 2e (Interactive annotation) would be missing. As stated in Sect. 3.5, interactive annotation for the purpose of collaborating and expressing ideas can be achieved in by sharing the pointer position of the people taking part in the session, together with sharing of sound, possibly using an external application or a phone. Thus, although we did not succeed in addressing all requirements in Table 1, we believe we have shown that it is achievable to do so in a easily accessible web application.

6 Results and Findings

6.1 Results

The results produced in this project in terms of concrete artifacts such as databases, software, websites and open source code are described in the following bullet points.

- We collected data from research groups and created a database which was positively received and used by SUCCESS members.
- For performing covisualization of the data in the database, we evaluated software alternatives and identified suitable software (SKUA), which we integrated all data into. We made SKUA accessible to remote users and we organized meetings which produced new collaborations.
- We found that the 3D SKUA software was inaccessible and hard to use and therefore created an accessible and easy to use solution which however was limited to 2D visualization. This opened up for a work mode using the 2D solution for discovery of data, articles and collaborators based on data overlap detectable in the map view or using free text search. After identifying interesting data or collaborators, a collaborative SKUA session could be initiated for detailed follow up scrutiny.
- We made the 2D solution publicly available at success.webfarm.cmr.no, containing a subset of data consisting of nonconfidential results of the SUCCESS project.

- The 2D web solution had limitations on data visualization. We evaluated the possibility to show 3D subsurface data in a web browser and created functional prototypes for this [40, 42]. We successfully demonstrated that it is technically possible to covisualize 3D subsurface data in a web application without requiring any plugins.

6.2 Findings

CCS research groupware in context of CSCW related work. As described in related work, there exist well designed tools that have increased collaboration in the oil and gas industry, as well as in research groups that have shared and well defined goals. We have shown that CCS research is more heterogeneous than other projects where collaborative tools have been applied successfully. Although the overall goal of SUCCESS was storage of CO_2, each group worked with distinct subgoals and workflows. SUCCESS was a large research project and it has been shown [8] that workflows within research are more fluid, as solving new problems requires novel thinking and creativity, and that large research projects often have more high level goals.

CCS groupware requirements. We have identified a combination of groupware functionalities needed in CCS research (Table 1) that no solutions today offer. We also found that we lack mechanisms for determining when a functionality is sufficiently implemented, and how effective it is relative to the other functionalities with respect to facilitating collaboration. For instance, we found, only at the end of iteration 1, that threshold for use (requirement 1b) was too high and hindered adoption, and that ability to download data (requirement 3b) was considered as a more central functionality than the others. The first observation is in line with Isenberg et al. [15] who points out that for groupware to be useful in practice, it must be quick to set up and not require considerable overhead to organize.

Towards web based open standards. We have taken steps in the direction of using modular, web based and open standards for creating groupware for 3D subsurface data in CCS research.

We were not able to identify commercial software adequately fulfilling our requirements. It is also worth to note the challenge in getting an overview of functionalities in commercial software for subsurface geology, as described in Sect. 3.3. We also identified a lack of software libraries addressing subsurface visualization. At the same time we see that open libraries now exist for web based visualization and web based synchronization. These two technologies in combination can be used as building blocks for creating software addressing our requirements, as described in Sect. 5.1, and may produce effective groupware. We demonstrated using the open web frameworks Angular and React for making reusable subsurface visualization components with the open 3D graphics standard X3D. As the subsurface entities are standard in any geological domain, this is useful not only for CCS research.

Palomino et al. showed in their review [14] that use of geospatial data above the surface is evolving from proprietary desktop software and data formats toward open source code and data formats. Further, it has evolved towards web-based tools and APIs to create dynamic web visualizations shared by collaborative teams. An example of this is Web Map Service [29], which is a standard protocol for serving georeferenced map images over the Internet. There now exists several free Web Map Service databases, and free web-enabled software for visualizing map data. Geospatial data below the surface is still being handled by proprietary standalone applications as described in this paper, and we believe a similar development for this data would substantially increase the opportunities for collaboration.

Having web services that provide subsurface data in an open format together with visualization components that can be used from web browsers that read and visualize the subsurface data will have several advantages. Separating data provider and data visualizer breaks down the problem in two components, each which can be solved by separate initiatives. Having a framework that supports independent visualization components for each datatype, makes it easier to create tailored web software, and to expand software by creating new visualizations. Open data format makes it easy to share and reuse data across applications and over distances.

Suggestions on improving data collection. The data collection phase in this project was time consuming as it was sequentially performed by one person and included travelling to several sites. It would be more efficient if the data collection could be a cooperative effort from the start. We believe this could be possible by using concepts from distributed version-control systems. To be able to accommodate for new types of data discovered during the data collection, we used a simple Excel sheet where each row represented a dataset while the columns represented various attributes of the datasets such as geographical position and an URL to the raw data. With this structure, it was easy to add new datasets by adding columns, and to add new attributes by adding new rows. We also made a script for transforming the Excel sheet into an SQL database used by our software as described in Sect. 4.2.

Using distributed version-control, collaborators could fork out (copy) an existing Excel sheet, add data themselves, and then an administrator could audit the changes to ensure relevance and correctness of the additions and merge all or parts of them. There could be several levels of administrators, each responsible for verifying and merging the submitted data from their own groups. New data would then move in a tree structure from the data contributors towards the main database. Users could also take a copy of the database and add their own, possibly confidential, data and visualize it locally.

7 Discussion and Summary

7.1 Discussion

No matter how well a CSCW software solution is implemented, there will exist external factors that can inhibit collaboration. We list up and discuss such factors here.

Scientific research environment. Collaboration may not be of high priority due to predefined goals, tools and collaboration setups specified in project proposals before the project started. This is supported by two responders on the questionnaire stating, "For Ph.D. students it would be easier if collaboration across Institutional boundaries and/or scientific disciplines was defined from the start." and "Collaboration should be integrated into the structure of the center. Making data sharing easier will not help if there are no incentive to collaborate". Another reason for reduced collaboration is that research is highly competitive and collaborating with groups that publish in similar domains can be problematic.

Researcher mindset. It is possible that collaboration is down-prioritized due to researchers' high degree of specialization and strong focus on doing research and publications in their own field of work. This was also indicated by questionnaire respondent 3 (see section "Achieving an overview"). Another respondent stated that "apart from the seminars and annual meetings, me and my group have not had any major collaboration with other groups in SUCCESS, although I have commented on that several times in the meetings that we need to improve this but unfortunately nothing has been done in this regard. The wish to focus on publishing may interfere with the wish for collaboration." There are several reasons for not collaborating, as described in the Editorial [47], such as to improve one's own competitive position and to avoid dependency on colleagues completing their part of the work on time and with sufficient quality.

Data confidentiality. Several research groups had interesting but confidential datasets which they could not share with other groups. Such data could be third-party data that they did not own, or data that they had acquired internally but wanted to analyze and publish papers on before sharing.

7.2 Summary

The work presented in this chapter was directed towards the multidisciplinary research field on carbon capture and storage (CCS). We have presented results including reports on a groupware system and experiences learned when creating groupware for CCS research. As CO_2 storage requires collaboration between several research disciplines, the contributions presented here are inherently crossing boundaries. The chapter has presented groupware design choices along these steps: (1) Review of work from the three large fields of CSCW, visualization and subsurface geology with respect to collaboration; (2) Introduction to the multidisciplinary research field of CCS, describing its collaborative challenges and our attempt to address these by providing groupware to the researchers; (3) Presenting a four-activity strategy to systematically get overview of the research questions and datasets, and to design a CSCW solution; (4) Presenting the various data types in CCS and selected a basis suitable for CSCW; (5) Presenting a set of requirements for CCS collaboration software and evaluated existing and custom created software against these criteria; (6) Implementing and rolling out a 2D web solution; (7) Demonstrating the different strengths of the 2D web and 3D standalone solution; (8) Suggesting a 3D web solution that will sufficiently support all criteria. (9) Suggesting research directions and technologies for enabling collaboration in subsurface projects. We believe the experiences reported here can be valuable in large CCS projects and for subsurface projects in general.

References

1. R.S. Haszeldine, Carbon capture and storage: how green can black be? Science **325**, 1647–1652 (2009)
2. C. Krafft, Success annual report (2016)
3. F. Dai, *Virtual Reality for Industrial Applications* (Springer Science & Business Media, Berlin, 2012)
4. R. Helland, E.M. Lidal, J. Grimsgaard, T. Langeland, C. Giertsen, A decade of increased oil recovery in virtual reality. IEEE Comput. Graph. Appl. **27**, 94–97 (2007)
5. T. Rosendahl, *Integrated Operations in the Oil and Gas Industry: Sustainability and Capability Development: Sustainability and Capability Development* (IGI Global, 2012)
6. N. U. of Science and Technology, Center for integrated operations in petroleum industry (2018). http://www.iocenter.no/
7. A.R. Edwards, Integrated operations (IO) in mining and oil and gas, what can we learn from each other?, in *Society of Petroleum Engineers*, *SPE Middle East Intelligent Oil and Gas Conference and Exhibition* (2015)
8. A.L. Young, W.G. Lutters, (Re)defining land change science through synthetic research practices, in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW'15 (ACM, New York, NY, USA, 2015), pp. 431–442
9. G.M. Olson, D.E. Atkins, R. Clauer, T.A. Finholt, F. Jahanian, T.L. Killeen, A. Prakash, T. Weymouth, The upper atmospheric research collaboratory (UARC). Interactions **5** (1998)
10. J.D. Myers, T.C. Allison, S. Bittner, B. Didier, M. Frenklach, W.H. Green, Y. Ho, J. Hewson, W. Koegler, L. Lansing, D. Leahy, M. Lee, R. McCoy, M. Minkoff, S. Nijsure, G. von Laszewski, D. Montoya, C. Pancerella, R. Pinzon, W. Pitz, L.A. Rahn, B. Ruscic, K. Schuchardt, E. Stephan, A. Wagner, T. Windus, C. Yang, A collaborative informatics infrastructure for multiscale science, in *Proceedings of the Second International Workshop on Challenges of Large Applications in Distributed Environments*, CLADE 2004 (2004)
11. G. Chin Jr, C. Lansing, Capturing and supporting contexts for scientific data sharing via the biological sciences collaboratory, in *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW'04 (2004)
12. R. Ranon, L. Chittaro, R. Pugliese, The elettra virtual collaboratory: a CSCW system for scientific experiments with ultra-bright light sources, in *Proceedings of HCITALY 2003* (2003)

13. D. Jacoby, A. Wynden, B. Gao, F.C. Giannini, M. Costa, Y. Coady, Geospatial computing collaborations, in *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)* (2019)
14. J. Palomino, O.C. Muellerklein, M. Kelly, A review of the emergent ecosystem of collaborative geospatial tools for addressing environmental challenges. Comput. Environ. Urban Syst. **65** (2017)
15. P. Isenberg, N. Elmqvist, J. Scholtz, F. Cernea, K. Ma, H. Hagen, Collaborative visualization: definition, challenges, and research agenda. Inf. Visual. **10** (2011)
16. A. MacEachren, I. Brewer, Developing a conceptual framework for visually-enabled geocollaboration. Int. J. Geograph. Inf. Sci. **18** (2004)
17. A. MacEachren, M. Gahegan, W. Pike, Visualization for constructing and sharing geo-scientific concepts. PNAS **101** (2004)
18. R. Kirby, M. Meyer, Visualization collaborations: what works and why. IEEE Comput. Graph. Appl. **33** (2013)
19. J. Olson, E. Hofer, N. Bos, A. Zimmerman, G. Olson, D. Cooney, I. Faniel, A theory of remote scientific collaboration. Sci. Collab. Internet **10** (2008)
20. N. Bos, A. Zimmerman, J. Olson, J. Yew, J. Yerkie, E. Dahl, G. Olson, From shared databases to communities of practice: a taxonomy of collaboratories. J. Comput. Mediated Commun. **12**, 652–672 (2007)
21. M. Sedlmair, M. Meyer, T. Munzner, Design study methodology: reflections from the trenches and the stacks. IEEE Trans. Visual. Comput. Graph. **18** (2012)
22. Human-centred design for interactive systems (2019). https://www.iso.org/standard/52075.html
23. D. Auber, Tulip: a huge graph visualisation framework, in *Graph Drawing Software* ed. by P. Mutzel, M. Junger. Mathematics and Visualization (Springer, Berlin, 2004), pp. 105–126
24. J.P. Birnholtz, M.J. Bietz, Data at work: supporting sharing in science and engineering, in *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP'03 (ACM, New York, NY, USA, 2003), pp. 339–348
25. M. Bietz, Standardization in large-scale collaborative science. Workshop on sharing, re-use and circulation of resources in cooperative scientific work, in *Proceedings of the ACM 2014 Conference on Computer Supported Cooperative Work* (ACM, Baltimore, MD, USA, 2014)
26. M. Nimtz, M. Klatt, B. Wiese, M. Kühn, H.J. Krautz, Modelling of the CO_2 process-and transport chain in CCS systems-examination of transport and storage processes. Chem. Erde-Geochem. **70**, 185–192 (2010)
27. D. Luebke, M. Reddy, J.D. Cohen, A. Varshney, B. Watson, R. Huebner, *Level of Detail for 3D Graphics* (Morgan Kaufmann, 2003)
28. V. Agafonkin, Leaflet—an open-source javascript library for mobile-friendly interactive maps (2018). https://leafletjs.com/
29. O.G. Consortium, Web map service (2019). https://www.opengeospatial.org/standards/wms
30. T.G. Blenkinsop, Visualizing structural geology: from excel to google earth. Comput. Geosci. **45**, 52–56 (2012)
31. L. Valentini, M.A. Brovelli, G. Zamboni, Multi-frame and multi-dimensional historical digital cities: the como example. Int. J. Digital Earth **7**, 336–350 (2014)
32. M. Krämer, R. Gutbell, A case study on 3d geospatial applications in the web using state-of-the-art WebgGL frameworks, in *Proceedings of the 20th International Conference on 3D Web Technology* (ACM, Heraklion, Greece, 2015), pp. 189–197
33. J. Ziolkowska, R. Reyes, Geological and hydrological visualization models for digital earth representation. Comput. Geosci. **94**, 31–39 (2016)
34. Cesium blog: underground support (2020). https://cesium.com/blog/2020/06/01/cesium-june-release/
35. F. Mwalongo, M. Krone, G. Reina, T. Ertl, State-of-the-art report in web-based visualization. Comput. Graph. Forum **35** (2016)
36. W. Consortium, Getting started with X3D (2017). http://www.web3d.org/getting-started-x3d
37. X3DOM, X3DOM at a glance (2017). https://www.x3dom.org

38. G. Malt, A. Geitung, H. Soleim, D. Patel, Visualizing 3d geology in web browsers using X3DOM, in *2017: Norsk Informatikkonferanse*, Norway, Oslo (2017)
39. Google, One framework. Mobile and desktop (2017). https://angular.io/
40. T.F. Eriksen, A modular approach for creating web applications with 3D geology using react and X3DOM. Master's thesis, Western Norway University of Applied Sciences and University of Bergen, Bergen, Norway (2019)
41. Facebook, A javascript library for building user interfaces (2019). https://reactjs.org/
42. O. Malt, Source code (2018). https://bitbucket.org/oysmal/
43. T. Eriksen, Source code (2019). https://github.com/tomfjug/Visualizing-3D-geology-with-React-and-X3DOM
44. WebRTC (2019). https://webrtc.org/
45. J. Lukasczyk, X. Liang, W. Luo, E.D. Ragan, A. Middel, N. Bliss, D. White, H. Hagen, R. Maciejewski, A collaborative web-based environmental data visualization and analysis framework, in *Workshop on Visualisation in Environmental Sciences (EnvirVis) (2015) at EuroVis* ed. by K. Rink, A. Middel, G.H. Weber (Eurographics Association, 2015)
46. H. Andrioti, A. Stamoulias, K. Kapetanakis, S. Panagiotakis, A.G. Malamos, Integrating WebRTC and X3DOM: bridging the gap between communications and graphics, in *Proceedings of the 20th International Conference on 3D Web Technology* (ACM, 2015)
47. Editorial, Who'd want to work in a team? Nature **424**, 1 (2003)

Subsurface Evaluation Through Multi-scenario Reasoning

Ingrid Chieh Yu, Irina Pene, Crystal Chang Din, Leif Harald Karlsen, Chi Mai Nguyen, Oliver Stahl, and Adnan Latif

Abstract Interpretation of the subsurface in order to find out where hydrocarbons are located is a challenging task for explorationists. They need to be creative and come up with innovative ideas when defining and assessing new prospects, especially nowadays when the easy to find, big fields have been already discovered. The challenges related to prospect assessment are (1) the geodata is uncertain, intermittent, sparse, multiresolution, and multi-scale, and (2) the explorationists often limit themselves to assess few possible scenarios. In this chapter, we try to address these challenges by looking into logic-based techniques for subsurface modeling. We demonstrate how the inherent complexity in geology, such as spatial and temporal aspects, can be formally captured and reasoned about using the strength of different formalizations. We consolidate the various modeling techniques into a methodology for creating multiple geological scenarios and conceptual reasoning about their merits by answering how? and why? early phase qualitative prospect assessment questions. This logic-based technology enables explorationists to express interpretive uncertainty as discrete scenarios with branches of potential alternative interpretations. We use two use cases related to subsurface evaluation to show the applicability of our methodology.

I. C. Yu (✉) · I. Pene · C. C. Din · L. H. Karlsen · C. M. Nguyen · O. Stahl · A. Latif
Department of Informatics, SIRIUS SFI, University of Oslo, Oslo, Norway
e-mail: ingridcy@ifi.uio.no

I. Pene
e-mail: pene@ifi.uio.no

C. C. Din
e-mail: crystald@ifi.uio.no

L. H. Karlsen
e-mail: leifhka@ifi.uio.no

C. M. Nguyen
e-mail: chimn@ifi.uio.no

O. Stahl
e-mail: ostahl@ifi.uio.no

A. Latif
e-mail: adnanl@ifi.uio.no

D. Patel (ed.), *Interactive Data Processing and 3D Visualization of the Solid Earth*,
https://doi.org/10.1007/978-3-030-90716-7_10

1 Introduction

Oil companies seek to limit the risks of exploration drilling by determining the presence, types, and volumes of hydrocarbon present in a prospect before drilling. The main variables that need to be estimated are probability of geologic success (Pg), Minimum Economic Field Size (MEFS) and associated probability of economic success (Pe). Advances in digital tool support for quantitatively-oriented geodata interpretation over the past decades have improved geoscientists' capacity to delineate closed structures and potential traps indicating possible hydrocarbon deposits in the subsurface [26].

Estimating the Pg relies on correct assessment of chance factors associated with petroleum system components e.g., whether source rock is present and mature, the presence and quality of reservoir rock, the presence of an effective seal and whether or not there are hydrocarbon accumulations in subsurface structures observed in the data. These factors are very difficult to estimate on the basis of measurement-oriented data only. Geoscientists therefore supplement these quantitative methods with qualitatively oriented forms of reasoning, such as using pen and paper along with drawing and presentation tools for developing and communicating multiple hypothetical geological scenarios, which we refer to as *geological history reasoning*.

A key challenge, for exploration in particular and subsurface evaluation in general, is that geodata is "uncertain, intermittent, sparse, multiresolution, and multi-scale" [8]. Geodata underdetermines concrete models and interpretations [18]. This means that even though geodata may refute particular interpretations of the subsurface, they can never verify the correctness of a single interpretation. As such, it is common for different geoscientists to hold widely differing interpretations of available data [3, 4].

Geological history reasoning makes active use of data underdetermination as a source of knowledge to develop and assess multiple hypothetical scenarios of the subsurface. These scenarios will explain whether, where, and (importantly) why there can be hydrocarbon accumulation in a prospect, displaying the available data in terms of sequences of interrelated geological processes [15]. Despite its centrality in exploring new hydrocarbon resources, this qualitative form of reasoning remains almost utterly unsupported in the digital tool set currently available [19]. Geoscientists often end up limiting themselves to only assessing a few possible scenarios due to the lack of a methodology for constructing alternative scenarios and evaluating their merits.

There have been some recent developments in addressing this problem. Natali et al. [16] present a tool for sketching and visualizing geological models by sequentially defining stratigraphic layers. The technique proposed allows geologists not only to produce different scenarios of the subsurface by using sketches, but also to communicate their concepts. Lidal et al. [12] present a graphical system to capture and visualize different subsurface interpretations for discussion and analysis. This tool offers a complete pipeline from conceptual models and ideas in sketches to synthesize a 3D geometrical model. Both approaches focus on visualization processes to communicate the results of geological reasoning. However, neither of these approaches offer ways to support the systematic and rigorous analysis of multiple geological scenarios.

In this chapter we experiment with logic-based techniques for subsurface modeling, giving examples on how the inherent complexity in geology such as spatial and temporal aspects can be formally captured and reasoned about using the strength of different formalizations. In particular, we demonstrate the use of abstraction and how formal modelling gives a precise and human-readable representation of domain knowledge. Later, we show how we bring together the various models in a novel tool-based approach [7] that constructs multi-scenarios to support geologically-oriented subsurface evaluation. In this work, we combine techniques from knowledge representation with formal methods (mathematical approaches that support the rigorous specification, design and verification of computer systems) to the exploration domain. This logic-based technology enables explorationists to express interpretive uncertainty as discrete scenarios with branches of potential alternative interpretations. With this approach, common-sense explorationist domain knowledge and rules of thumb are explicitly represented in the tools together with collected O&G data. Our goal is to develop a tool that supports explorationists to systematically describe, construct, and compare different scenarios in early-phase exploration prospect assessment. The applicability will be demonstrated on two use cases related to subsurface evaluation.

2 Subsurface Evaluation in the Oil and Gas Exploration

Play-based subsurface evaluation is one of the predominantly used workflows in exploration. The overall focus of this workflow is to understand the main ingredients of the petroleum system at a basin-scale and then assess how these components come together to define discrete petroleum plays (play is a geographically delineated area where several geological factors are present so that producible petroleum could be proven [17]). This assessment is then used to evaluate the subsurface at a prospect scale. This workflow has a pivotal role in understanding and accessing the hydrocarbon generation, migration from source to the reservoir, accumulation and trapping mechanisms.

Shell [22] define a similar play-based exploration methodology in terms of an exploration pyramid with three distinct levels of evaluation, i.e., Basin focused, Play focused, and Prospect focused evaluation.

The main components for the basin focused evaluation are tectonostratigraphic framework, plate tectonics and reconstructions, basin evolution, structural and stratigraphic elements, and potential petroleum systems. These elements are then used to define discrete plays present in the petroleum system. These plays are then further analysed for stratigraphic framework, exploration history, and volumetric scope. The probability of success is then calculated, and plays are ranked accordingly.

Narrowing down from basin-scale to discrete plays, the prospect evaluation phase focuses on assessing a particular play in a specific part of the basin. Several geological

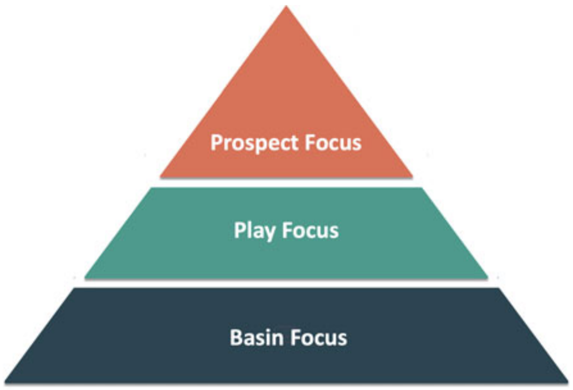


Fig. 1 Illustration of the play-based exploration process [22]

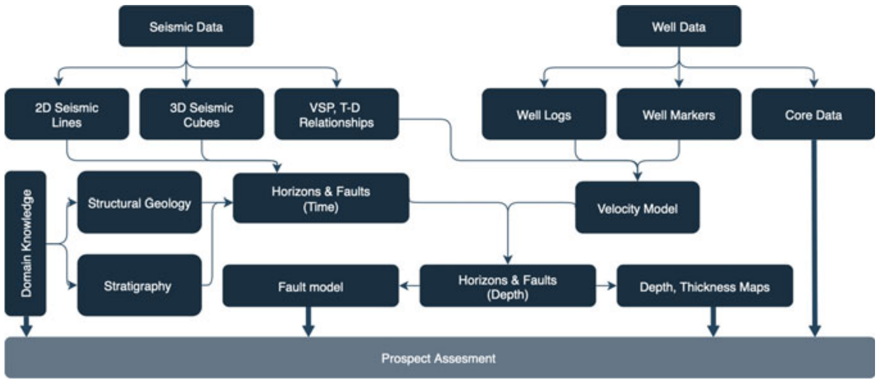


Fig. 2 A generalized workflow for using geological and geophysical data in subsurface evaluation

attributes are considered in this part of the workflow for understanding the subsurface on a much finer scale. These attributes include structural and stratigraphic configuration, DHIs (direct hydrocarbon indicators), facies, migration pathways, trapping and sealing capacity of the reservoir formation at a particular location. This evaluation is mostly based on structural and stratigraphic interpretation of geological and geophysical data. The output of the play focus phase is a portfolio of prospects. In the prospect focus phase, a set of drill-ready prospects with information on probabilistic volumes, risks, dependencies and estimated probability of success are provided (Fig. 1).

Figure 2 shows a generalized workflow for subsurface evaluation by integrating and interpreting various types of geological data. Seismic data plays a key role in this workflow for finding leads and defining prospects.

Advances in seismic interpretation have improved the capacity to delineate closed structures and potential traps over the past decades. However, seismic provides limited information about the presence and maturity of source rock, presence and quality of reservoir rock and the presence of an effective seal and have limited capacity to substantiate the presence of hydrocarbon accumulations in these structures.

Several tool-supported methods exist for Explorationists to address these aspects. The most common methods for such assessment are the basin and petroleum system modelling. Both apply numerical techniques to forward simulations (modelling of geological processes in sedimentary basins over geological time spans). However, as these are data-based simulation methods, their results are never better than the data they are based on. As such, there is a need to supplement measurement-oriented methods with more qualitatively oriented methods. Many factors such as presence and maturity of source rock, presence and quality of reservoir rock and the presence of an effective seal are challenging to estimate based on measurement-oriented data only. To compensate, throughout the entire subsurface evaluation cycle, Explorationists use pen and paper along with digital drawing and presentation tools for developing and communicating multiple hypothetical geological scenarios. Novel tools remain to be developed for this type of conceptual reasoning for answering *how* and (importantly) *why* there can be hydrocarbon deposits in a prospect through many-faceted narratives.

3 Conceptualized Geological History

To reconstruct the geological history of the earth one must understand all the geological events that contributed to the shaping of the subsurface, based on the geological time scale. A series of techniques and principles from structural geology, tectonics, stratigraphy and paleontology are used to describe the sequence of these events. Structural geology characterises the geometry of the structures, their deformation and tectonics explains the geological context in which deformations occur. Geological structures control the migration, trapping and escape of hydrocarbon fluids. For example, folds and faulted blocks can form traps that accumulate oil and gas.

Geoscientists use three main ingredients when trying to reconstruct the history of an area [21]; *facts*, *hypothesis* and *laws*. A fact can be seen as something that is considered being true, for example, the dip direction of a fault plane (if Fig. 3 has west-east orientation, then "fault `f1` is dipping to the west" is a fact). A hypothesis is an assumption geologists make during the reasoning process to reduce the number of uncertainties. A law is a fundamental concept that is always verified by experiments and is aligned with our understanding of the world. We use them to determine how and when the rocks have been created and how they changed through time.

There are many principles or laws that apply in geology e.g.:

- The Law of Superposition
- The Law of Original Horizontality
- The Law of lateral Continuity
- Cross-cutting relationship
- The Principle of Faunal Succession.

By combining facts, hypotheses and laws, a model is derived that provides a reasonable and coherent interpretation of the observed facts. A key challenge for exploration is that geological data are uncertain, intermittent, sparse, multi-resolution, and multi-scale. This means that even though geological data may refute particular interpretations of the subsurface, the correctness of a single interpretation cannot be verified. As such, it is common that different geoscientists hold different interpretations of available data.

In the following subsections, we walk through some examples that concretize the use of these concepts and how they take part in the process of discovering hydrocarbon accumulation. First, we start with an example of a static evaluation of a petroleum system, assuming that the present-day configuration has not been changed after the start of migration. We focus mainly on the structural elements and not on the processes that lead to the present configuration of the area. In the second example, we will discuss some of the geological processes that have changed the configuration of the area under evaluation during different geological time periods and how these changes affect the presence of hydrocarbon accumulation. To not overwhelm with too detailed domain knowledge, we have deliberately simplified aspects of geology, e.g., hydrocarbon generation.

These examples will form the basis for the technical sections on using logical and formal approaches to achieve automated multi-scenario reasoning.

3.1 Example 1—Petroleum System

Before diving into the reasoning process, we will talk shortly about the petroleum system and the elements and processes that one should consider to discover oil and gas. This concept is very well known and used, and we will not bring any changes to its definition. Our focus is on how geologists are reasoning about the existence of a petroleum system when evaluating an area of interest and how they cope with the uncertainties.

Petroleum system is the framework used in substantiating the possibility of hydrocarbons accumulation in an area. For oil and gas to accumulate, a series of elements and processes must exist and be correctly placed in time and space. The elements of a petroleum system include source rock, reservoir rock, cap rock and overburden rock. Below we give a short description relevant for our example:
Source rock refers to organic-rich rocks that have generated or are capable of generating and expelling hydrocarbons. They are deposited in many deposition environments including deep marine, lacustrine or deltaic.
Reservoir rock represents rocks that are porous and permeable to allow accumulation and drainage of hydrocarbons. All types of rock can act as a reservoir rock. The most common reservoir rocks are represented by sedimentary rocks, mainly sandstone, limestone and dolostone.

Cap rock is defined as a relatively impermeable rock that seals the top of a reservoir, impeding hydrocarbons to migrate/escape. The most common cap rocks are represented by shale, anhydrite or salt.
Overburden rock overlies the source rock, contributing to its maturation.

Petroleum systems have two processes, trap formation and generation-migration-accumulation of hydrocarbons:
A *petroleum trap* refers to a geological structure in which reservoir and cap rock provide the right settings for hydrocarbons to accumulate and preserve. Three main types of traps are classified as structural, stratigraphic and combined.
Generation-migration-accumulation. Migration of hydrocarbons is a very important process but not very well understood. Many questions need to be answered when evaluating migration. In our example, we will focus on identifying:

- migration pathways
- geological units with good permeability that can act as carrier/reservoir beds
- geological units with low permeability that can act as seals
- properties of faults that can influence the migration and accumulation of hydrocarbons.

For detailed reasoning about the presence of a petroleum system in a specific area, we have chosen the case study illustrated in Fig. 3. The geological settings for this case study comprise a series of four rotated fault blocks, each block consisting of three geological units. We assume that the *source rock* is present, has generated and expelled oil and gas and is located to the left, outside the figure. Furthermore, the *traps* are structural, fault-dependent, relying on faults being sealing or on lateral seal being present in order to hold hydrocarbons. All the geological units that make up the fault blocks are deposited in a marine environment, as a submarine fan

The submarine fan, illustrated in Fig. 4, has an approximate W–E orientation, having the following logical distribution, from proximal to distal: a feeder channel, distributary channels, lobes and basin plain. Geological units coloured in yellow represent the possible *reservoir units* deposited in a feeder channel, distributary channel or lobe, the ones in green represent the *seal(s)* (base, top, lateral) deposited in a basin plain environment. For simplicity, we assume that units deposited in the feeder channel, distributary channels and lobes are porous and permeable, acting as reservoir/carrier beds. The units deposited in basin plain settings are considered non-porous and non-permeable, acting as a seal. *Migration pathways* are represented by a combination of geological units (carrier beds) and faults. For pathways to be effective, carrier beds should be permeable and the faults should be non-sealing.

In this use case, additional information is provided by the well X (Fig. 3), drilled in the area of interest. Well X did not encounter any hydrocarbons but the cores and logs acquired tell us that the reservoir is present, of good quality and deposited in a distributary channel.

All the observations, facts and hypotheses stated above are combined for answering the question: "*Where and under what conditions can hydrocarbon accumulations be found?*". Figure 5 shows a possible series of steps that a geoscientist needs to follow. First, we need to understand why well X was dry. Since we know that the source

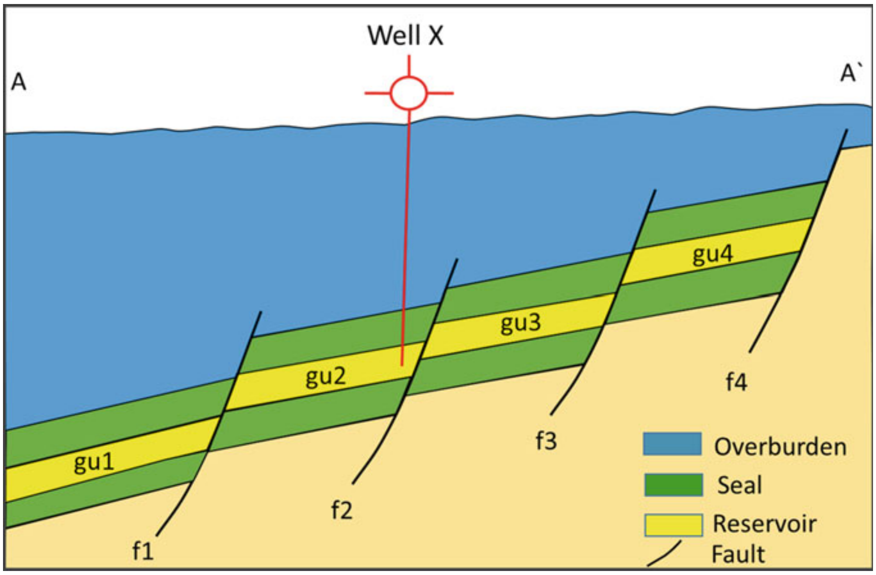


Fig. 3 The static geological setting of rotated fault blocks

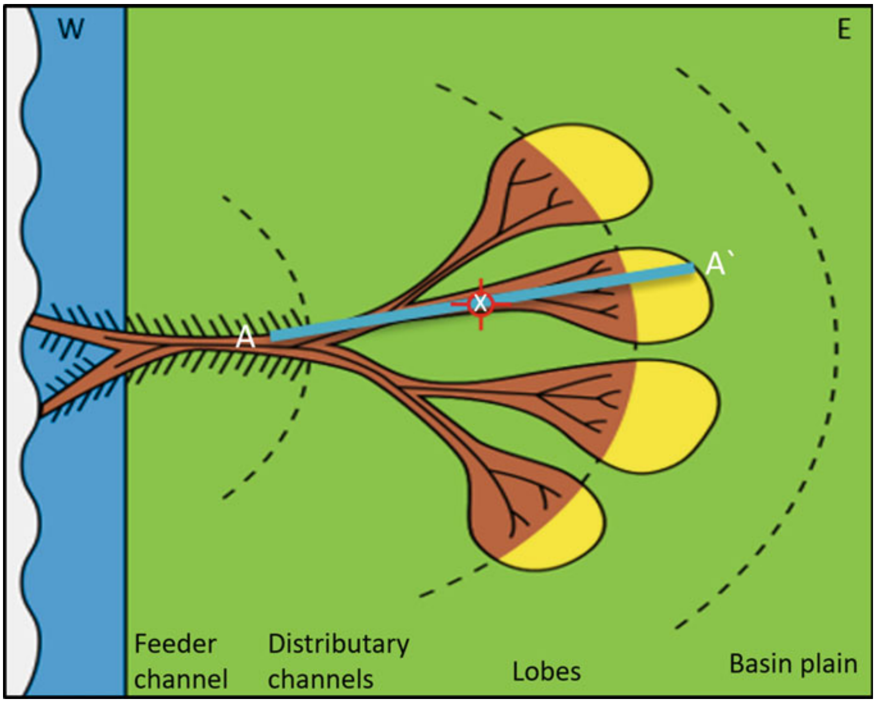


Fig. 4 Submarine fan elements; Figure 3 is shown as the blue line A–A

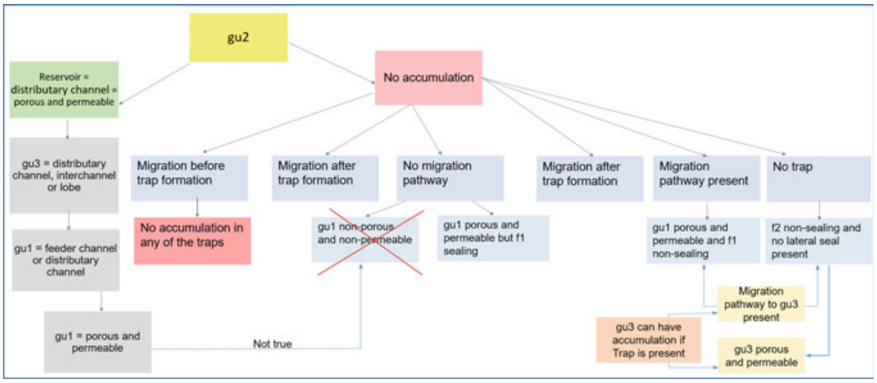

gu2
Reservoir = distributary channel = porous and permeable
No accumulation
gu3 = distributary channel, interchannel or lobe
Migration before trap formation
Migration after trap formation
No migration pathway
Migration after trap formation
Migration pathway present
No trap
No accumulation in any of the traps
gu1 non-porous and non-permeable
gu1 porous and permeable but f1 sealing
gu1 porous and permeable and f1 non-sealing
f2 non-sealing and no lateral seal present
gu1 = feeder channel or distributary channel
Migration pathway to gu3 present
gu3 can have accumulation if Trap is present
gu1 = porous and permeable
Not true
gu3 porous and permeable

Fig. 5 A geologist's manual reasoning

rock is present and has generated and expelled hydrocarbons and reservoir is present and of good quality we can conclude that the failure can be attributed to:

- absence of migration pathways
- absence of a viable trap.

Absence of migration pathways from the source rock into the reservoir (gu2) means:

- gu1 is non-porous and non-permeable or/and
- Fault f1 is sealing

In this case, no accumulation will be present in gu2.

Absence of a viable trap means:

- Fault f2 is non-sealing and no lateral seals present
- No lateral seals present means that gu3 (the lateral contact of gu2) is porous and permeable.

If the absence of trap is the only failing element for well X being dry, we can continue reasoning about the possibility of having hydrocarbon accumulation in gu3. Following the logical distribution of the submarine fan's elements and knowing that gu2 is deposited in a distributary channel we can deduce that gu1 can only be deposited as distributary or feeder channel, and gu3 can be either in a distributary channel or a lobe. We can state that gu3 is a reservoir and of good quality, so in order to have hydrocarbons accumulated, we need a trap to be present. For gu3 to be trapped we need either f3 to be sealing or an impermeable geological unit acting as a lateral seal. This kind of reasoning is neither complete nor fail-free and the success is depends on the expertise of the interpreter.

3.2 Example 2—Erosion

The subsurface is not static, it changes through time. Therefore, when evaluating the petroleum system for a particular area, understanding when and how geological processes have changed the subsurface is crucial. Erosion is one of the processes that impact the most the presence or absence of a petroleum system. Erosion of some geological units may lead to the destruction of migration pathways, seals or trap integrity. The time of erosion should be correlated with the time of migration.

Generally, erosion is defined as the process in which some earth material is removed from one location and deposited somewhere else by different transport agents such as wind, water or ice. For some geological units to be eroded, exposure (above sea level) must happen first. Exposure can be due to other geological processes such as uplift or tilted block faulting. Tilted block faulting, which is associated with extensional tectonic events, results in the creation of normal faults, downwards movement of the hanging-walls and rotation. While fault blocks rotate and tip, erosion occurs. Sediments resulted from erosion are redeposited into the accommodation space created by the downward movement of the blocks. During the erosion of the exposed units, no deposition occurs, resulting in a gap in the geologic record, forming an unconformity.

We use the cross-section illustrated in Fig. 6 to exemplify how geologists reason about the sequence of events that affected the subsurface at a specific location. Later we will show how this can be done by an automated reasoning engine.

A geologist may ask the question "*Is it possible to have hydrocarbons accumulation in* gu5 *and/or* gu7 *and how?*". To answer such a question, the geologist will undertake a petroleum system analysis of the area. The combination of facts and observations with domain knowledge leads to the following understanding:

- Source rock is represented by gu6, age Jurassic and is composed of shale;
- Reservoir is represented by gu5 and gu7, age Triassic and Jurassic, and is composed of sandstone;
- Properties of gu1, gu2, and gu3 are unknown;
- Trap is of stratigraphic type, onlap.

With the assumption that the source rock has generated and expelled hydrocarbons, the reservoir units are porous and permeable and the seals have very low permeability, then the only element that remains to be discussed is the presence of migration pathways. Looking at the present-day configuration of the area of interest a possible path for hydrocarbons to migrate, from the source rock to the reservoir, can be represented by: gu6 (source rock) - f1 - gu2 - gu5 - gu7. For migration to happen, fault f1 must be non-sealing, and gu2 must be permeable, which contradicts with the facts. However, the lack of a migration pathway at the present time does not necessarily imply its absence throughout the entire geological time. We need to reconstruct the geological history of the area in order to investigate a possible pathway(s) at a different geological time. By analyzing Fig. 6, we can come up with the following possible interpretation:

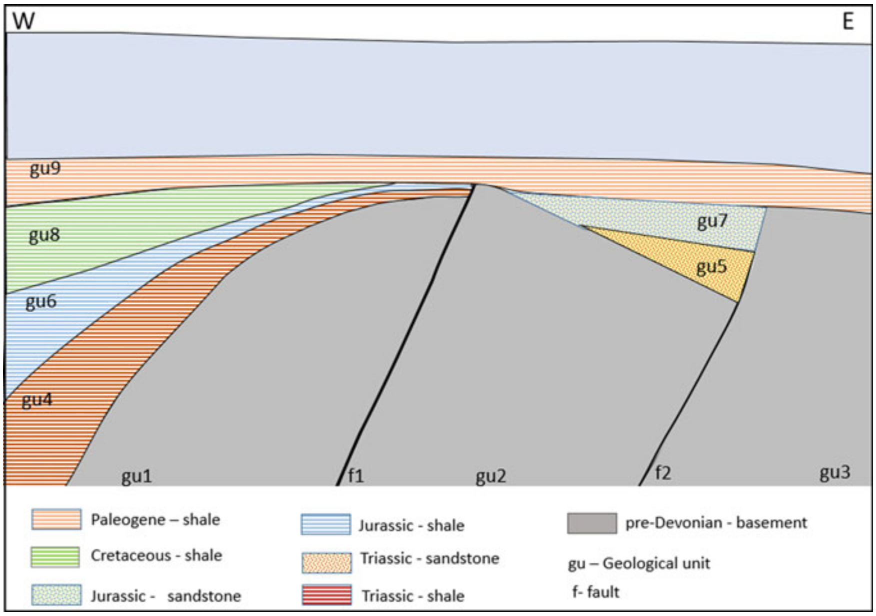


Fig. 6 Geological cross-section (for reasoning about geological processes that could have taken place)

- gu1, gu2 and gu3 represent the pre-faulting units
- f1 and f2 are the result of faulting
- gu2 is dipping to the east, indicating tilting and rotation during faulting
- gu5 and gu7 represent the syn-faulting units
- gu9 represent the post-faulting unit
- The age of faulting is equal with the age of the syn-faulting geological units, in this case Triassic-Jurassic
- There is a hiatus (missing geological units) on the western part of the gu2 due to either non-deposition or erosion.

The missing units are an important part of the petroleum system analysis. If the hiatus is due to erosion, it means that these units have been deposited at a certain time but eroded later. If, for example, these units were deposited during Triassic and Jurassic and eroded during the mid-Cretaceous time and migration of hydrocarbons started early Cretaceous, they could have acted as carrier beds and/or reservoir units if permeable and porous. This demonstrates that a critical task for a geologist is to correctly place these missing units in *space* and *time*.

4 Logical Frameworks

This section is dedicated to giving some basic understanding of the logics used in subsurface modeling and automated reasoning.

4.1 Description Logics

Description logics [1] is a family of logical languages tailored for formalizing human knowledge and automated reasoning by computers. The different languages are tailored for different purposes, balancing the trade-off between high expressivity and high computational cost of reasoning. Several description logics have standardized implementations known commonly as OWL [20] which have been successfully applied to formalize domains such as medicine and biomedicine [6, 9, 24] and engineering domains [11, 23, 25]. The main constructs in these languages are *individuals*, *classes*, and *properties*. Individuals denote concrete objects (such as the geological unit gu2). Classes intuitively denote sets of individuals sharing common properties. Examples of classes are Fault and GeologicalUnit. To state that gu6 is a source rock, one simply writes *SourceRock* (gu6). Properties either denote relationships between individuals (e.g. *hasAge* (gu6, Jurassic) or *above* (gu9, gu8)) or relates an individual to a concrete data value such as numbers, dates, text, etc. (e.g. *hasDepth* (gu9, 1500)). With these basic constructs, one can create more complex classes by making expressions that combine classes with other classes or properties. For example, to express the class of all individuals having a nextTo-relationship to some GeologicalObject, one writes $\exists nextTo . GeologicalObject$. To make the class of all individuals that belongs to both the class A and B, one writes $A \sqcap B$ (for example anticline belongs to both $Trap \sqcap GeologicalStructure$, or if we want the class of individuals that belongs to either one (or both) of the classes we can write $A \sqcup B$.

Classes can then be related via subclassing, that is, one can state that one class A is a subclass of another B, by writing $A \sqsubseteq B$. Intuitively, this states that any individual that belongs to the class A should also belong to the class B. For example, one could state that all sandstones are rocks with the statement $Sandstone \sqsubseteq Rock$. With subclassing, one can create complex class hierarchies describing the relationships between the terms within a domain. For example, to state that all geological objects are either faults or result of a deposition, we can write:

$$GeoObject \sqsubseteq Fault \sqcup \exists isDepositedIn . DepositionalEnvironment$$

A set of description logic statements is called an *ontology*. Typically, an ontology contains a set of statements describing relevant parts of a domain, such as geology, medicine, or biology. One of the main reasons for building such an ontology is to have computers do reasoning on its statements. For example, the computer can infer that object o is either a fault or is a result of a deposition. If we, either directly or through inference from other statements, know that *o* is not a fault, the computer can then deduce that it must be a rock. Such reasoning can therefore be used to infer all implicit knowledge in data as well as checking the consistencies of data.

Note that an ontology does not need to be completely correct, nor does it need to capture all details of the domain it describes. An ontology should be viewed as a model of the domain, i.e. it described the relevant parts of the domain up to a level of detail necessary for the application of the ontology. For example, if one wants to model geology for the purpose of determining where migration and accumulation can happen, the ontology needs to capture the spatial relationships between geological units and faults, the porosity and permeability, etc. but might not need to model all possible materials geological unit can be made of.

Below is a small example ontology describing geological units and their properties:

$$\begin{aligned}
GeoUnit \sqsubseteq\ & \exists partOf.GeologicalLayer \sqcap \\
& \exists hasPorousity.Porousity \sqcap \\
& \exists hasPermeability.Permeability \\
Seal \sqsubseteq\ & (GeoUnit\ \sqcap \exists hasPermeability.\{nonPermeable\}) \sqcup \\
& (Fault \sqcap \exists hasSealingCapacity.\{sealing\}) \\
Reservoir \sqsubseteq\ & GeoUnit \sqcap \\
& \exists hasPorousity.\{porous\} \sqcap \\
& \exists hasPermeability.\{permeable\} \sqcap \\
& \forall connectedToMigrationDir.Seal
\end{aligned}$$

The above ontology states the following: A geounit is part of a geological layer, it has some porousity and some permeability; a seal is either a geounit that is non-permeable or a sealing fault; and finally, a reservoir is a geounit that is porous and permeable and only connected to seals in the migration direction.

4.2 Rewriting Logic

Rewriting logic [14] was introduced in the 1990s and has been applied in various domains, such as automated deduction, software and hardware specification and verification, security, real-time and cyber-physical systems, probabilistic systems, and bioinformatics. Rewriting logic can be understood as having a computational side and a logical side. On the computational side, rewriting logic is a semantic framework in the sense that it can represent different software and hardware modeling languages, algorithms, protocols, and mechanisms of parallelism. Moreover, these representations can be executed and analysed. On the other hand, rewriting logic can be seen as a logical framework where different logics and automated deduction procedures can be represented and reasoned about. These two sides complement each other and for many applications, both sides can be represented even though the applications may fall more naturally towards one of the sides. In this chapter, we will focus on the computational side of rewriting logic and talk about it as a semantic framework for representing and reasoning about concurrent systems having states and where the system evolves by means of transitions.

Formally, a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ consists of an equational theory (Σ, E) [5] and a set of rewrite rules R. Σ is an equational signature and E a set of Σ-equations. The equational theory specifies the statics, i.e., the structure and is entirely user definable. Semantic framework, (Σ, E) specifies the static aspects of distributed states (in contrast to formulas in a logical framework). Σ is a set of declarations of sorts, subsorts, and typed operators, including the state constructors that build up a distributed state out of simpler state components. For example, an operator f declared with the syntax op $f : s_1 \dots s_n \to s$, tells us that the operator has n arguments, $s_1 \dots s_n$ are the sorts of the arguments, and s is the sort of its result value. Operators can also define constants. For example, we can define constructors of different environments in a submarine fan:

```
sort SubmarineFanEnv.
op   feederChannel       : → SubmarineFanEnv.
op   distributaryChannel : → SubmarineFanEnv
op   interChannel        : → SubmarineFanEnv.
op   lobe                : → SubmarineFanEnv.
op   lobeFringe          : → SubmarineFanEnv.
op   basinPlain          : → SubmarineFanEnv.
```

E is a set of confluent and terminating equations with the form eq $t = t'$ and ceq $t = t'$ if $cond$, where the latter expresses conditional equations, which means that cond must be evaluated to true in order for the equation to be applied. Distributed states can be seen as terms (algebraic expressions) built up by operators in Σ. Two states t and t' describe the same state if one can show the equality $t = t'$ using the equations E. Rewrite rules$R$ specify the dynamic aspects of a distributed system, i.e. what are the possible local transitions that take the system from one distributed state to another distributed state. Equations in E can be thought of as specifying the structure over which rules R operate.

Rules in rewriting logic can be expressed with the form rl $[l] : t \Rightarrow t'$ and crl $[l] : t \Rightarrow t'$ if $cond$, where the latter is a conditional rewrite rule. The rule specifies the system's legal transitions from an instance of t to the corresponding instance of t', where l is a label. Conditional rules apply only if their conditions hold. Note that unlike equations that specifies equality between t and t', $t \Rightarrow t'$ describes the state change. For example an area with unconsolidated sediment with certain porosity (t), can because of compaction during burial which moves grains closer together, result in the sediment having lower porosity (t'). Expressed using rewrite rules, t and t' specify different states of that area of sand and unlike equations, t' cannot without further ado become t (equations hold the same meaning on both sides of the equality sign).

The dynamic aspect of rewriting logic captures local concurrent transitions. With several rules performed at the same time, true concurrency is represented.

In the following, we show some simple rules to exemplify how rewriting can be used to change objects' states as well as the state (or known as configuration) of the computational system:

```
rl [compaction] :
        compaction(GU)
        ⟨GU : GeoUnit | Type: sandstone,Porosity: porous, Permeability: permeable ⟩
        ⟹
        ⟨GU : GeoUnit | Type: sandstone,Porosity: nonPorous,
         Permeability:nonPermeable ⟩
```

This rule provides a simplified description of the effect of compaction on sandstones. The rule expresses that if there is a geological unit of type sandstone that is porous and permeable and that this unit can be subjected to compaction, then when compaction occurs, the porosity and permeability of the geological unit will change to non-porous and non-permeable. Here, objects are distinguished by angle brackets. The rule is precise in communicating the abstraction, the prerequisites and the effects of what we mean by compaction of sandstones. Assume as input we have a concrete sandstone ⟨ gu3 : GeoUnit | Type: sandstone, Porosity: porous, Permeability: permeable ⟩with identity gu3 and a message compaction(gu3). This input then *matches* the left-hand side of the arrow (GU $\mapsto$ gu3). The result of this rule application is a sandstone ⟨gu3 : GeoUnit | Type: sandstone, Porosity: nonPorous, Permeability: nonPermeable ⟩. Moreover, the message is removed from the system. Similarly, rules can also be used to create new objects:

```
rl [faulting]  :
        faulting(GU, F)
        ⟨GU : GeoUnit | Type: T ⟩
        ⟹
        ⟨GU(1) : GeoUnit | Type: T ⟩  ⟨GU(2) : GeoUnit | Type: T ⟩
        ⟨F : Fault | ⟩
```

Ignoring details such as type of fault, orientation and spacial information, faulting fractures a geological unit into two units (GU(1) and GU(2)) and creates a new unique fault object F. Note that the rule uses a variable T to denote that faulting disregards the type of sedimentary rock, but requires GU, GU(1) and GU(2) to have the same sediment type.

5 Modeling Geology in Space

Spatial understanding of the subsurface plays a critical role in interpreting the geological system. Most geological processes are spatial in nature, i.e. they affect how the geological elements resides in the subsurface and changes how each element relate to each other (e.g., faulting or uplifting changes the geometrical configuration). The geometry of geological objects is also crucial in determining their role in the petroleum system, such as a dome-shaped structure can function as a trap for hydrocarbon accumulation and an impermeable rock covering the top of a permeable rock can act as a seal. To be able to formally reason about the evolution of geological systems, we see that it is necessary to find an appropriate abstraction and develop a formal language that is able to capture the necessary domain-specific spatial information. This section will outline what types of formal spatial reasoning we need and intuitively how this is formalized in our framework.

5.1 Spatial Formalization and Reasoning

To capture the necessary spatial information described above, we need a spatial language for describing the relationships between the different elements within a petroleum system, such as depositional layers, geological units and faults. We could of course simply introduce one relation per relationship directly, e.g.

- *domeShape*(X)
- *impermeableRockCovering*(X, Y)

However, when designing a formal language one generally wants to introduce as few foundational language constructs as possible, and build as much abstractions as possible over these foundational constructs. The reason for this is that each such construct needs to be given an explicit semantic meaning in terms of code or other external means, and needs to be explicitly handled by any software using the language. In our setting, such foundational constructs would be base relations which we then use to define other relations.

To capture geological spatial relations, we use the system developed in [10] for formalizing spatial relations, which consists of two different notions, *roles* and *relations*. Roles are simply names given to the parts of objects relevant for the relationships we want to express. Examples of roles are *boundary*, *interior*, and *topmost point*. We then e.g. use *boundary(X)* to denote the boundary-part of an object *X*. Relations are used to relate objects and their parts. We minimize the number of base relations, which are used as building blocks for constructing more complex ones. Examples of such base relations are given below:

- *overlaps(X, Y)*—states that *X* overlaps *Y*, that is, there is some part common to both *X* and *Y*
- *before(X, Y)*—states that *X* comes before *Y* in the canonical direction, e.g. *Y* is east of *X*

Together with the roles described above, a complex relation such as connectedness can be expressed by *connected(X, Y) = overlaps(boundary(X), boundary(Y))*, in which an object *X* is connected to an object *Y* if their boundaries overlap. We can then express that *X* is connected to *Y* in the eastward direction (i.e. object *Y* is east of object *X*) by *connectedEast(X, Y) = connected(X, Y) and before(X, Y)*. By combining statements with *and*, we state that both relations must hold. We can also negate a statement by using *not*, e.g. to state that gu1 is not connected to gu2 we can write *not connected(*gu1, gu2*)*. To express that a unit is dipping to the east, we can state that its topmost point is to the west: *dippingEastward(X) = overlaps(topmost(X), west(X))*. In a similar fashion, we can define *above* to express that an unit is above another through the composition of roles *top* and *bottom*, which denote the top boundary and bottom boundary of a unit, respectively.

These definitions, relating more abstract relations to the base relations, can now be made before any reasoning or formal analysis is done. Thus, the reasoning and analysis can be restricted to base relations. For these base relations, only very simple reasoning is necessary, such as symmetries (e.g. $overlaps(X, Y)$ implies $overlaps(Y, X)$) and transitive closures (e.g. $before(X, Y)$ and $before(Y, Z)$ implies $before(X, Z)$). This simplifies the overall architecture and makes it easier to extend the framework with additional relations.

5.2 Modeling the Geological Spatial Relations

To answer the questions related to migration described in Example 2 in Sect. 3.2 we need to identify possible migration pathways. A migration pathway can be represented by a *connected* sequence of geological units and faults with certain properties. For example, gu6 is connected to gu2 via f1 in the eastern direction, which can be formalized by the relation introduced in the previous section: *connectedEast*(gu1, gu2).

In order to capture specific geological structures, we need to take into account the dipping direction of geological units, and total versus partial coverage of geological units. Thus, statements such as "gu2 is east dipping" and "gu5 is above the eastern top of gu2" can be formalized as *dippingEastward*(gu2) and *above*(gu5, gu2) *and overlaps*(*east*(gu2), gu5), respectively.

Modeling the Submarine Fan. The spatial language can also be used in describing the relations between different elements in a submarine fan (see Fig. 4). Each element, such as distributary channel and feeder channel, is represented by a class, i.e. DistributaryChannel and FeederChannel. A geological unit gu2 located inside a distributary channel can be captured by *DistributaryChannel*(gu2).

In order to capture the logical distribution of different elements in a submarine fan from proximal to distal, we use the spatial relation *connectedEast*. An example

$$\mathit{FeederChannel} \sqsubseteq \forall \mathit{connectedEast}.\ (\mathit{FeederChannel} \sqcup \mathit{DistributaryChannel})$$

describes that all geological units in the feeder channel must be connected (eastwards) either other units in the feeder channel or units in the distributary channel.

We can now express similar axioms for the other pairs of adjacent elements to capture the rest of the spatial structure.

6 Modeling Geological Processes and Time

Geological processes are events that change the subsurface over time. When a geologist talks about a particular process, the cause and effect of various geological configurations subjected to that process are described within a temporal setting.

The evidence accumulated over the last twenty years strongly supports the claim that in rewriting logic what is represented and its representation is often isomorphic structures [14]. We exploit this property and exemplify how geological processes can be captured through rewriting logic.

6.1 The Structure of System Configurations

Rewriting rules specify the system's legal transitions from one configuration to the other. We define what a system configuration should contain based on the domain of the system being modelled. In order to model the geological processes and petroleum system using rewriting logic, our system configuration contains three main domain parts describing the geological time, the geological objects, and the spatial relations between the geological objects. In addition, the configuration also carries information that affects and directs the computation, such as interpreted seismic data and knowledge acquired during runtime. In this section, we show the syntax used to describe the various components. For readability, we have purposefully reduced the complexity of the terms and rules. All the capitalized terms are variables.

Geological time T is represented by the term geoTime(T). We use the International Chronostratigraphic Chart (ICS) as the model of time e.g., Jurassic, Paleogene, etc. For linking a specific process in time, such as timing for migration, we use the convention migrationTime(T).

For geological object, we limit ourselves to the objects that are relevant for our use cases; faults and geological units. A fault is expressed by the term

⟨ID : Fault | SealingCapacity: SC⟩

where SealingCapacity can have values sealing or non-sealing. Each fault has an unique identity ID. A geological unit is expressed by the term

⟨ID : GeoUnit | Type: TP, Permeability: PMT, Porosity: PRT, Accumulation: B, GeoTime: T⟩

where Type describes the type of rock, e.g., sandstone or shale, Permeability can either be permeable or non-permeable, Porosity values can be porous or non-porous, Accumulation can be true or false, and GeoTime describes when the rock was deposited. Each geological object has a unique identity captured by ID. A geological unit contains multiple properties; however, within a specific rewrite rule, we only show those properties that are relevant for its application, i.e., properties and their values, for the rule to apply. A set of spatial relations S is presented by the term spatial(S), in which S may contain the horizontal and vertical relations between different objects, i.e., geological units and faults, and the dipping directions of faults. We use the language described in Sect. 5 to encode S.

Data stemming from an interpreted seismic picture, which the rules may operate on, is wrapped in the term interpretation(I). For example, it may contain information about faulting periods, time gaps between two geological units, or a set of geological units L that were high during a time period TP, i.e., wasHigh(L, TP). Note that this information may be derived from a reasoning engine and not necessarily provided by a user.

6.2 Time Advance

Although time is advancing continuously, our time modeling is discrete. We focus on capturing the relation between discrete changes rather than presenting a continuous evolvement. We capture the possible geological processes that happened within each geological time and the corresponding consequences in terms of how the geometrical structure changes at the end of each geological time steps. This is like taking a sequence of snapshots, and a use of smaller time steps will create more snapshots.

The geological time which we model belongs to the past history. What we can see from the seismic picture today is a consequence of a sequence of geological processes that happened from Cambrian time up to now. For the big events, there are well-studied theories for how the subsurface evolved and the order of events, however, for local regions in the subsurface, it is not always clear what order of events formed the unconformities and their impact on the petroleum system at the various times. This way of backward thinking, constructing potential histories to match current observations, is quite different from the forward-thinking traditionally found in computer-based simulations where the application has been to make predictions about some future behaviour.

So how do we model the geological processes through time? Based on the given domain knowledge, the information extracted from the seismic picture, and assumptions about the subsurface in the past, we simulate the geological processes from Cambrian, and then Ordovician up to Quaternary. During each geological time, we simulate a sequence of tectonic events and processes modeled in rewriting rules. Figure 7 shows an illustration of how geological processes and time advance is organized. The circles represent the states, i.e., a snapshot of the subsurface at one moment in time. The blue box represents the conditional check for a geological process. A state can only be transited from one to the other if the corresponding conditions of the rewrite rule are satisfied. When all the possible events have taken place (simulated) during a geological time, time will be advanced to the next geological period. For example, if the current time is Triassic, the next time to be advanced to will be Jurassic. This is illustrated by the big rectangle, representing the duration of a geological time that contains a sequence of local rule applications. The red box represents the conditional check for time advancing (captured by the conditional rewrite

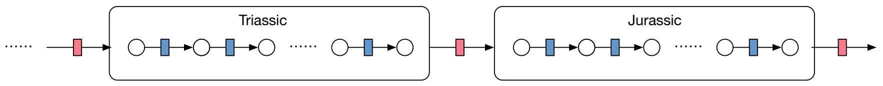


Fig. 7 Illustration for the application of rewrite rules demonstrated how geological processes and time advance is organized

rule for time advance), ensuring that all rule applications will eventually converge to an interpretation that fits the current geological observation.

```
crl [time-advance] :
    geoTime(T) interpretation(I) C ⟹ geoTime(advance(T,geoTimeTable)) interpretation(I)
    C
    if timeConstraints(T,I, C) .
```

where C is a placeholder for geological objects and spatial relations in the system configuration. Function timeConstraints(T, I, C) ensures that all necessary depositions and erosions have been considered within the current time period T before advance(T, geoTimeTable) advances time to the next geological period.

6.3 *Geological Processes*

For understanding the subsurface, geologists identify the geological objects and make assumptions about the processes that shaped them. Each process acts in a certain period of time, applies changes to some geological objects or create new objects, and thus change the geological settings at that particular time. To capture geological processes, such as *deposition*, *uplifting*, *faulting*, *tilting*, and *erosion* we use rewriting logic. *Erosion* and *faulting* are good examples for explaining how we model processes using rewriting rules.

To make the rules readable and understandable by non computer scientists, a flowchart is being used to illustrate the modeling of each geological process. In the flowchart, the oval boxes represent the starting and the ending of the process, the rhomboid boxes represent the status of the subsurface at one moment, the diamond boxes represent the decision points, the arrows direct the flow of the process, and the rectangular boxes represent a set of operations. When all the conditions inside a diamond box are met, the execution flow will follow the arrow with label `true`, otherwise, the arrow with label `false`. For our rules, it is only when all the conditions in the diamond boxes are met, we apply the operations defined in the rectangular box and derive a new subsurface state.

The flowchart represents an erosion process (Fig. 8), expressing that for a given geological time period T, the subsurface state contains a set of spatial relations S, a geological structure L, and a geological unit GU that is the top most unit of L. If L is high, i.e., exposed, during time period T, and GU does not exist today, then GU will be eroded during time T and the rule will update the spatial relations accordingly.

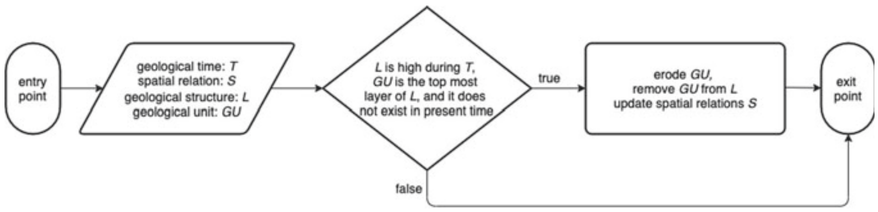

entry point
geological time: *T*
spatial relation: *S*
geological structure: *L*
geological unit: *GU*
L is high during *T*, *GU* is the top most layer of *L*, and it does not exist in present time
true
false
erode *GU*, remove *GU* from *L* update spatial relations *S*
exit point

Fig. 8 The flowchart of the erosion process

A rewrite rule for the erosion process corresponding to the flowchart above can be written as:

```
crl [erosion] :
    geoTime(T)  interpretation(wasHigh(L, TP), I)  spatial(S)
    ⟨GU : GeoUnit | ⟩
    ⟹
    geoTime(T)  interpretation(wasHigh(L, TP), I)  spatial(updateSpatial(GU, I, S))
    if occurs(T, TP) and isTopMostOf(GU, L) and notExists(GU) .
```

We demonstrate the application of this rule on Example 2 from Sect. 3.2. On top of the western part of the gu2 (Fig. 6) there is a hiatus (missing geological units), so TP is mapped to the time period Triassic-Cretaceous, assuming the current time T is Jurassic. This hiatus can be either due to partial erosion or non-deposition. Since we are in the correct time period (checked by function occurs(T, TP)) and a gu has been deposited on top of gu2 before Jurassic (checked by function isTopMostOf(GU, L)) and does not occur in the seismic section (checked by function notExists(GU)), it is reasonable to assume that gu has been eroded. The resulting spatial information will then correspond with what we see today, where gu has been removed from the configuration (not occurring on the right-hand side of the rule), and a new spatial relation is established (by the execution of function updateSpatial(GU, I, S)).

A similar approach is done for faulting. The big difference is that a new fault object is created and a geological unit is split in two by the newly created fault. A rewrite rule for faulting can be written as:

```
crl [faulting] :
    geoTime(T)  interpretation(faulting(id: F, timePeriod: TP), I)
    spatial(S)  C
    ⟹
    ⟨F : Fault | ⟩
    geoTime(T)  interpretation(I)
    spatial(updateSpatial(F, I, S))   updateGeoUnits(F, I, C)
    if occurs(T, TP) .
```

The message faulting(id: F, timePeriod: TP) contains information inferred from the input data, (e.g. input seismic section); the existence of a fault F at the present time is the result of a faulting process that took place in a period of time TP. The time period for faulting is set to be equal to the deposition time of the sediments present in the accommodation space created by the movement of the hanging wall. This conditional rewrite rule can only be applied if the current time T is in the time period TP, i.e., occurs(T, TP). The consequences of this rule application is that the fault F is created, the geological unit cut by the fault F is separated into two distinct units by the execution of the function updateGeoUnits(F, I, C), and the spatial relations between F and its surroundings are updated by the function updateSpatial(F, I, S) according to what we see today. Note that the function updateGeoUnits(F, I, C) also remembers the geological unit from which the two geological units were split from.

6.4 Modeling Petroleum System

Petroleum system has been discussed in some of the previous sections. Therefor, in this section we will show how the process of migration and accumulation of hydrocarbons can be captured by a rewrite rule.

The process of hydrocarbons migration–accumulation requires a source rock that has generated and expelled hydrocarbons, a migration pathway and a trapped reservoir unit. The runtime discovered migration pathway from a source rock SRock to a geological unit GU within a specific time T is represented by the term pathway(SRock, GU, T). The process of hydrocarbon migration and accumulation is captured by the conditional rewrite rule:

```
crl [migration-accumulation] :
  ⟨GU : GeoUnit | Permeability: permeable, Porosity: porous, Accumulation: B⟩
  pathway(SRock, GU, T1)  migrationTime(T2)
  ⟹
  ⟨GU : GeoUnit | Permeability: permeable, Porosity: porous, Accumulation: true⟩
  pathway(SRock, GU, T1)  migrationTime(T2)
  if (T1==T2 or isAfter(T2, T1)) and isTrapped(GU, T1) .
```

We demonstrate the application of this rule on Example 2. Assume that in the system configuration we have established that (1) gu7 was permeable, porous, and trapped during Paleogene (verified by the function isTrapped(gu7, Paleogene)), (2) a pathway from the source rock gu6 to gu7 exists in Paleogene (captured by the term pathway(gu6, gu7, Paleogene)) and (3) that the earliest possible hydrocarbon migration time from the source rock is Paleogene (captured by migrationTime(Paleogene)). With these preconditions it is possible for hydrocarbons to accumulate in gu7 during Paleogene time. Note that the rule only applies if migration happens after trap formation (ensured by functions isAfter(T2, T1) and isTrapped(GU, T1) where time T2 is after a time T1).

7 GeMS: A Geological Multi-scenario Reasoning Framework

In this section, we present GeMS, a tool for constructing geological multi-scenarios and its scalable architecture. The methodology of multi-scenario generation combines the formal modeling techniques showcased in Sects. 5 and 6 and we demonstrate its application by revisiting examples from Sect. 3.

7.1 The GeMS Methodology

Figure 9 illustrates the workflow of GeMS and its components. The *input data* contains geological observations (i.e., observable from the seismic sections), facts (e.g., information from the wells) and assumptions and interpretations of the subsurface. From the input, we have some understanding of the spatial condition of the subsurface and have knowledge about certain geological events, which will be used during the workflow to guide the scenario generation from ancient time up to now. However, there are still much data that is unknown that will influence the possible outcome of the constructed geological scenarios. For instance, the sealing capacities, permeabilities and porosities in the considered area may be unknown at the time of investigation, impacting the process of migration and accumulation of hydrocarbons. For seismic, one may observe a hiatus; however, it is unknown how the hiatus is formed. The unknown factors play an essential role as they define variations in the narratives (explaining the when and how subsurface evolved).

The domain knowledge is formalized by classes and properties using description logic. This formalization together with the spatial language is an integrated part of the framework. In the *Scenario pre-processing* phase, classes and properties are instantiated based on the given input data and the spatial relations between the geological entities (individuals) are captured by the spatial language. The individuals or properties of individuals that are under-specified, which we call ‘*unknowns*’, are identified based on the logical formalization of geological knowledge. These unknowns will be automatically concretized with potential values defined in the ontology, resulting in multiple automatically generated interpretations. The output of this phase is a set of *proto-scenarios*, which will be further investigated by *Scenario generation*.

The dynamic aspect of geology is formalized using rewriting logic. We developed a rewrite theory for geology, $\mathcal{R}_{geo}$, that contains various geological processes.

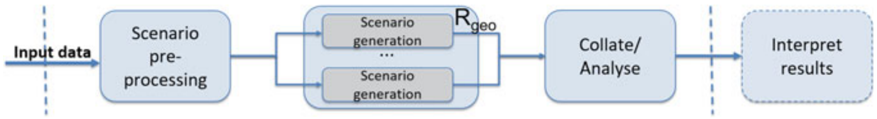


Fig. 9 The workflow of GeMS

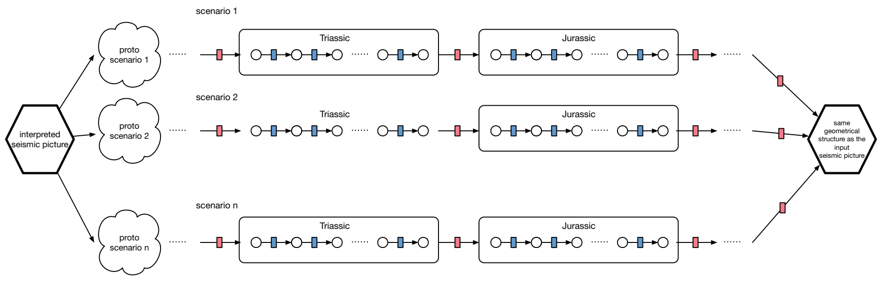


Fig. 10 Illustration for multi-scenario execution

In scenario generation, we use Maude [2], an engine capable of executing rewriting logic theories to generate multi-scenarios by instantiating $\mathcal{R}_{geo}$ with proto-scenarios. The execution will follow the geological time advancement described in the previous section. Each simulation results in a potential final geological scenario. GeMS ensures that the geometrical structure derived upon termination is consistent with the input interpreted seismic section, independent of the variations. We illustrate this invariant in Fig. 10.

When the number of uncertainties ('unknown') increases, GeMS is able to generate variations and support the geologist by pointing out and explaining scenarios that might easily have been overlooked otherwise. The value of using GeMS is that, the geologists can explore, explain and constrain scenarios. They can test and compare the result of different assumptions (by geologist with different experiences), see the variations and adjust their understanding of the prospect. It serves as a decision support early in their work process. We will show how the constructed scenarios can be analysed/presented through the use cases in Sect. 7.3.

7.2 A Scalable and Flexible Architecture

Due to a very high number of alternative scenarios can be generated by GeMS, the logic simulation is quite computational intensive. In GeMS, scalability can be realized by distributing the proto-scenarios over available machines and by running the Maude simulation in parallel.

The pipeline needs to be flexible enough so that the implementation of its components can be adapted through time, and it should be easy to control the communication with the different subsystems. To achieve these goals, the chosen architecture consists of one controller-application that handles all communication with the outside world. It controls the whole system and workers that are responsible for performing the computations. The more worker nodes are available in the system, the more computations can be run in parallel. A worker knows the controller by configuration and automatically contacts and registers with the controller machine when started. Workers can be added dynamically to an already started computation run.

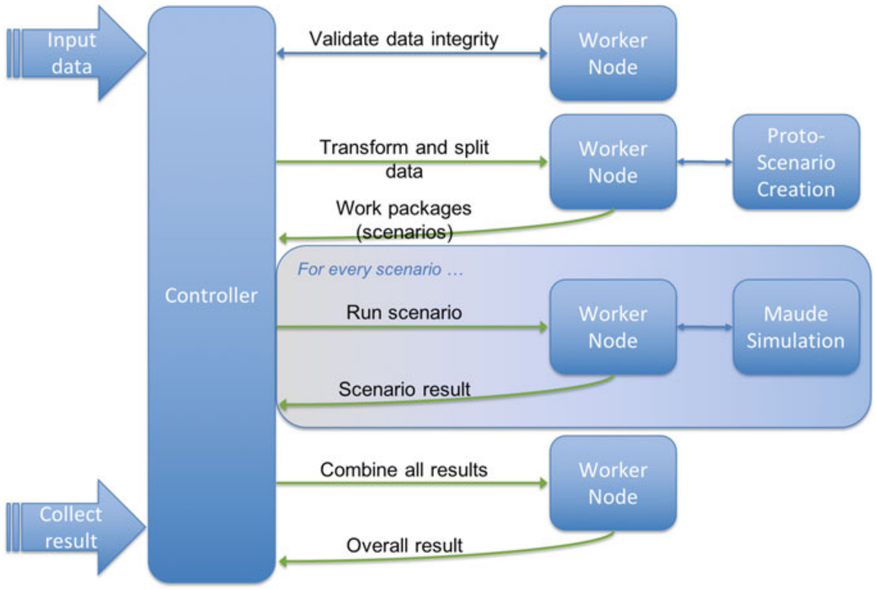
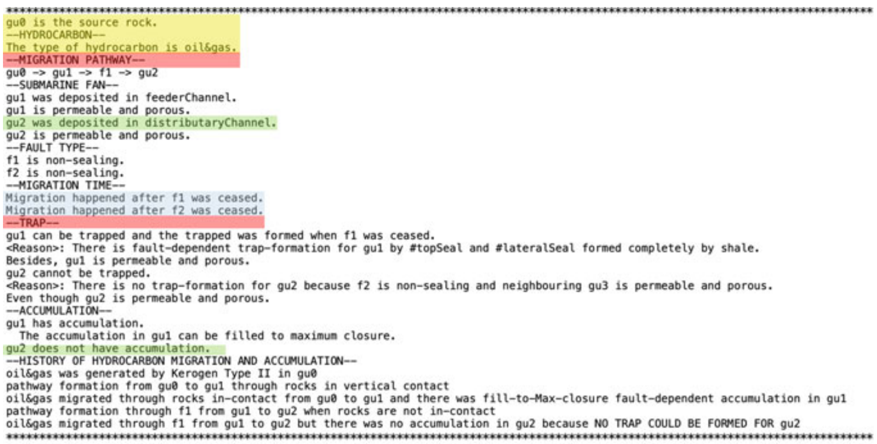


Fig. 11 The pipeline's sequence of interactions from start to finish

Figure 11 gives a detailed impression of the pipeline's functionality and interactions. The controller is developed to be completely agnostic of the information that is computed and uses REST web services to communicate with the worker machines as well as to provide an interface to users and applications that want to use the services the system provides. When the controller picks up an incoming computation request, it first expands the input data into proto-scenarios and then runs distributed computations on several worker machines. The task of generating proto-scenarios is delegated to an available worker machine, which then converts the input data into proto-scenarios using a program which extracts knowledge about the spatial relations of the input data and domain knowledge about the *unknown* elements in the data from an ontology that holds this knowledge. The enriched data is then given to an available worker machine and run by the Maude rewriting engine. After completing the simulation, the result is collected by the controller and the analysis is presented to the user.

Both the controller and worker implementation are available as Docker images. The usage of a containerized deployment enables usage in cloud environments as well as on local machines, either with one container per virtual or physical machine or with multiple containers deployed on the same computer. Containers, especially worker machines, are independent of the technology and programming languages used by GeMS. Should future use-cases require different methodologies or different software, this will not affect the machines running the pipeline as all necessary software are encapsulated inside the containers.



```
**************************************************************************************************************************
gu0 is the source rock.
--HYDROCARBON--
The type of hydrocarbon is oil&gas.
--MIGRATION PATHWAY--
gu0 -> gu1 -> f1 -> gu2
--SUBMARINE FAN--
gu1 was deposited in feederChannel.
gu1 is permeable and porous.
gu2 was deposited in distributaryChannel.
gu2 is permeable and porous.
--FAULT TYPE--
f1 is non-sealing.
f2 is non-sealing.
--MIGRATION TIME--
Migration happened after f1 was ceased.
Migration happened after f2 was ceased.
--TRAP--
gu1 can be trapped and the trapped was formed when f1 was ceased.
<Reason>: There is fault-dependent trap-formation for gu1 by #topSeal and #lateralSeal formed completely by shale.
Besides, gu1 is permeable and porous.
gu2 cannot be trapped.
<Reason>: There is no trap-formation for gu2 because f2 is non-sealing and neighbouring gu3 is permeable and porous.
Even though gu2 is permeable and porous.
--ACCUMULATION--
gu1 has accumulation.
  The accumulation in gu1 can be filled to maximum closure.
gu2 does not have accumulation.
--HISTORY OF HYDROCARBON MIGRATION AND ACCUMULATION--
oil&gas was generated by Kerogen Type II in gu0
pathway formation from gu0 to gu1 through rocks in vertical contact
oil&gas migrated through rocks in-contact from gu0 to gu1 and there was fill-to-Max-closure fault-dependent accumulation in gu1
pathway formation through f1 from gu1 to gu2 when rocks are not in-contact
oil&gas migrated through f1 from gu1 to gu2 but there was no accumulation in gu2 because NO TRAP COULD BE FORMED FOR gu2
**************************************************************************************************************************
```

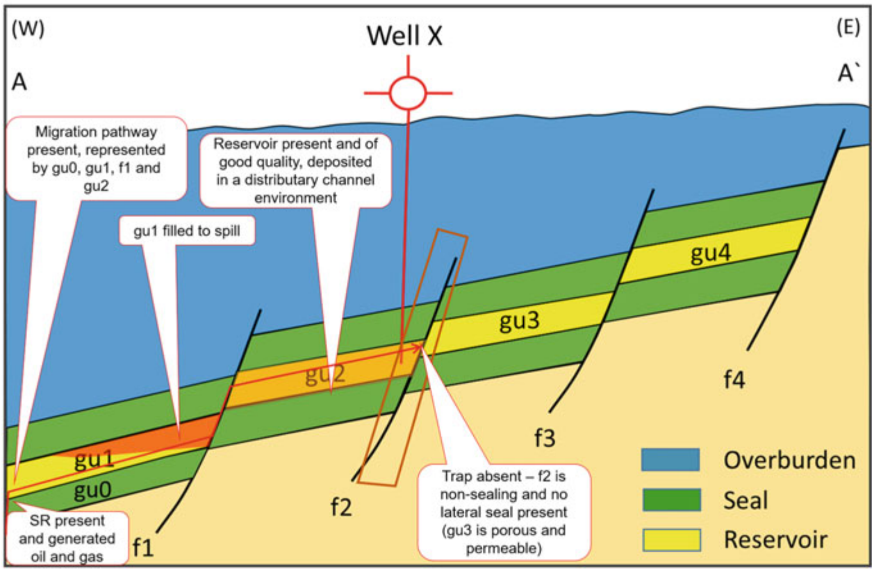

(W)
A
Well X
(E)
A`
Migration pathway present, represented by gu0, gu1, f1 and gu2
Reservoir present and of good quality, deposited in a distributary channel environment
gu1 filled to spill
gu4
gu3
gu2
gu1
gu0
f1
f2
f3
f4
SR present and generated oil and gas
Trap absent – f2 is non-sealing and no lateral seal present (gu3 is porous and permeable)
Overburden
Seal
Reservoir

Fig. 12 An example of scenario explanation and its graphical representation. For the textual explanation, yellow marks the observations, green the facts, blue geologists' assumptions and red are the generated interpretations by GeMS

7.3 Geological Scenarios

In this section we revisit the use cases described in Examples 1 and 2 from Sect. 3 and show how GeMS is able to construct multi-scenarios that explains how the subsurface has evolved and how the evolution affects the petroleum system.

Table 1 An example of the output scenario from Triassic to Jurassic

	Triassic	Jurassic
Geological processes	creation of f1 deposition of gu4 creation of f2 gu2 was tilted deposition of gu5 deposition of gu10 on gu2	deposition of gu6 deposition of gu7 gu2 and gu10 were tilted gu1, gu4 and gu6 were uplifted gu10 was eroded
Geological objects	gu1 $\sim$ gu5 and gu10 f1 was created f2 was created	gu1 $\sim$ gu7 f1 was active f2 was active
Spatial relations	gu1 is west dipping ... gu2 is east dipping ... gu3 has flat top gu4 is DirectlyAboveAll gu1 gu5 is DirectlyAboveEast gu2 gu10 is DirectlyAboveWest gu2 gu1 is NextTo f1 ... gu1 is ConnectedTo gu2 through f1 ... gu4 is ConnectedTo gu10 through f1 ... f1 and gu10 are Connected gu5 and gu10 are Connected	gu1 is west dipping ... gu2 is east dipping ... gu3 has flat top gu4 is DirectlyAboveAll gu1 ... gu5 is DirectlyAboveEast gu2 ... gu1 is NextTo f1 ... gu1 is ConnectedTo gu2 through f1 ... gu6 is west dipping gu6 is DirectlyAboveAll gu4 gu6 is NextTo f1 gu6 is NextTo gu4 gu6 is ConnectedTo gu2 through f1 gu7 is DirectlyAbove gu2 gu7 is DirectlyAboveAll gu5 gu7 is NextTo f2 gu7 is ConnectedTo gu3 through f2

The geological units and their spatial relations are the consequences of the geological processes. Since gu10 (western part of gu5) was eroded during Jurassic, all its spatial relations are also removed. Note that for saving the space, we only present some of the spatial relations in Table 1

7.3.1 Petroleum System

In this section, we will use Example 1 from Sect. 3.1 to explain what data comes into each component of the toolchain and how the question: "*Why there is no accumulation in gu2?*" can be answered using automated reasoning. The input data is represented by the facts stating that gu2 has no accumulation and has been deposited in a distributary channel, the observation about the presence of a source rock that has generated oil and gas and the assumption that the migration happened after the trap has formed. However, we do not have information related to the presence of a migration pathway and the presence of a viable trap. For this question, GeMs computed several potential scenarios. Figure 12 shows one scenario (as textual and graphical representations) where the migration pathway exists and is made up of gu0, gu1, f1 and gu2 but the trap is missing, because the fault f2 is non sealing and there is no lateral seal present. This lack of seal explains why Well X was dry.

7.3.2 Subsurface Evolution

This section will focus on the geological processes that have taken place during Triassic–Jurassic time period and how they have changed the area under investigation. Table 1 displays one scenario computed by GeMS when applied to Example 2. This scenario can be explained as follow: in Triassic a faulting process has started with creation of faults f1 and f2, splitting of basement into three different units (gu1, gu2 and gu3), tilting and downwards movement of gu1 and gu2, and syn-faulting deposition of gu4, gu10 and gu5. Note that we named gu10 as the western part of gu5. In Jurassic, the faulting process has continued, as well as tilting of gu2, the result being the erosion of gu10 and deposition of gu6 and gu7. Figure 13 captures graphically this sequence of geological processes for each of the two time periods. Continuing this scenario to present day, and by connecting to a petroleum system analysis, GeMS can actually discover that if f1 is non-sealing and gu2 is porous and permeable at Paleogene, then we can have a migration from gu6 into gu7.

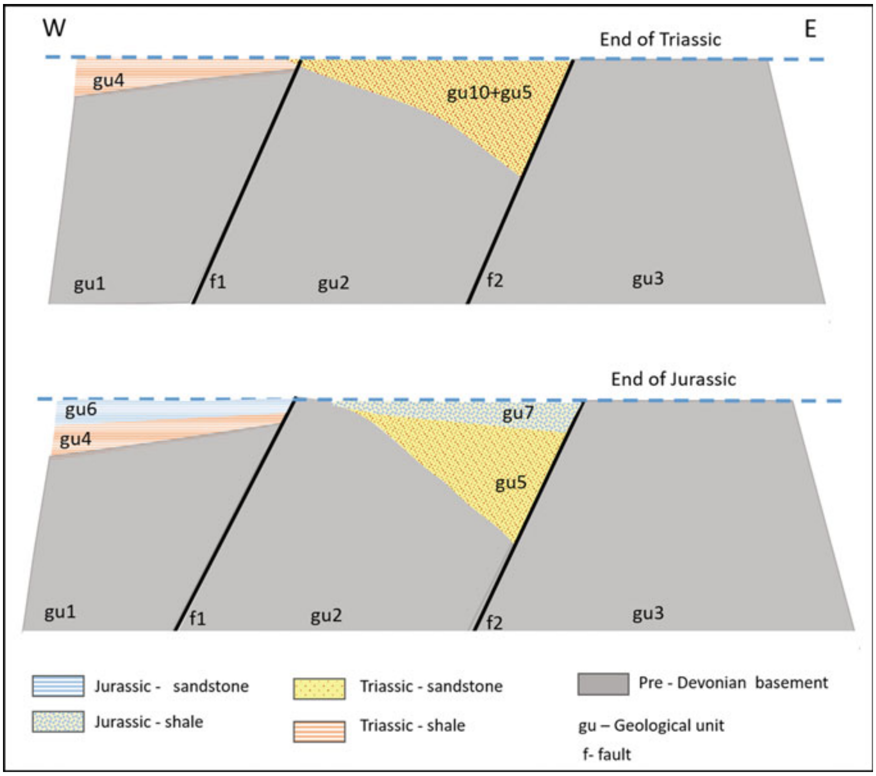


Fig. 13 Graphical representation of the scenario displayed in Table 1

8 Technology Directions

Recent advancements in computation, network and storage have led to numerous opportunities to improve the subsurface evaluation workflows. Further, the volatility and uncertainty in the oil and gas industry have forced exploration and production companies to find improved and cost-effective solutions by automating the subsurface evaluation workflows.

When it comes to digitalization, traditionally, the focus has been on purely data-driven workflows. Although geological reasoning is the most crucial factor that defines exploration success, little attention has been given to exploit digitalization opportunities in reasoning-based evaluation. In geological reasoning workflows, explorationist still rely on ad hoc manual work practices and tools and use pen and paper along with computer drawing and presentation tools to develop and communicate multiple hypothetical geological scenarios of the prospect. This leaves them with little to no efficient means to make the fullest use of state-of-the-art digital technologies to communicate and systematically compare and assess different hypothetical geological scenarios before deciding which scenario to pursue when assessing exploration prospects.

A recent study [13] of 97 wells drilled in the UK sector of the North Sea from 2003 to 2013 showed that some of the major reasons for unsuccessful exploration were the fact that several of the prospects relied too heavily only on data (seismic DHIs and amplitude). Regional play-based work for setting and context were repeatedly missed, and pre-drill analysis was often excluded the full range of possible outcomes.

The industry's current trend on moving from a physics-driven world to a purely data-driven world for a complex domain like Geoscience has proved inefficient. We believe that a significant success factor will be to combine geological reasoning-based evaluation with the insights derived from the Geological, Geophysical, and Petrophysical data.

The geological multi-scenario reasoning methodology described in this chapter is a step in this direction. It provides a hybrid approach by combining the data-driven seismic interpretation (faults and horizons) with geological reasoning (based on the encoded geological rules). It can significantly help subsurface experts to think out of the box to consider several geological models rather than relying on a single model. Further, it will enable Geoscientists to consider a full range of possible scenarios and corresponding outcomes beyond what is possible within human capacity.

The methodology presented in this chapter provides a mechanism to formulate, systematically compare and assess different conceptual (Geological) models. Since this work is a PoC (proof of concept), the data model for representing geological scenarios is limited to only a few prospect evaluation aspects. In the future, this model can be extended to cover the broader spectrum of prospect evaluation in the context of geological reasoning.

The key to a successful implementation of such methodology is integrating with existing subsurface evaluation tools and workflows currently being used for prospect evaluation. In this PoC, input data for the reasoning engine is the horizons and faults extracted from the interpretation software (e.g., Petrel E&P Software Platform by Schlumberger). The GeoAssistnat system is then able to process files (.xyz files; surface location (x,y) and depth (z)) and deduce geological information. In the future, a pipeline could be established for direct integration with interpretation software. The interpretation software can provide a mechanism for both the input and displaying output (visualization and mechanism for interpreting results) of the reasoning engine.

Acknowledgements The authors would like to thank our partners Equinor, Schlumberger and NTNU for the discussions and feedback on this work.

References

1. F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi et al., *The Description Logic Handbook: Theory, Implementation and Applications* (Cambridge University Press, Cambridge, 2003)
2. L. Bachmair (ed.), *Rewriting Techniques and Applications. 11th International Conference, RTA 2000, Proceedings*, Norwich, UK, 10–12 July 2000. Lecture notes in computer science, vol. 1833 (Springer, Berlin, 2000)
3. C.E. Bond, Uncertainty in structural interpretation: lessons to be learnt. J. Struct. Geol. **74**, 185–200 (2015)
4. C.E. Bond, A. Gibbs, Z. Shipton, S. Jones, What do you think this is? "Conceptual uncertainty" in geoscience interpretation. GSA Today **17**(11), 4–10, 12 (2007)
5. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C.L. Talcott (eds.), *All About Maude—A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*. Lecture notes in computer science, vol. 4350 (Springer, Berlin, 2007)
6. G. O. Consortium, Gene ontology annotations and resources. Nucleic Acids Res. **41**(D1), D530–D535 (2012)
7. C.C. Din, L.H. Karlsen, I. Pene, O. Stahl, I.C. Yu, T. Østerlie, Geological multi-scenario reasoning, in *32nd Norsk Informatikkonferanse, NIK*, UIT Norges Arktiske Universitet, Narvik, Norway, 25–27 Nov 2019 (Bibsys Open Journal Systems, Norway, 2019), p. 2019
8. Y. Gil, S.A. Pierce, H. Babaie, A. Banerjee, K. Borne, G. Bust, M. Cheatham, I. Ebert-Uphoff, C. Gomes, M. Hill, J. Horel, L. Hsu, J. Kinter, C. Knoblock, D. Krum, V. Kumar, P. Lermusiaux, Y. Liu, C. North, V. Pankratius, S. Peters, B. Plale, A. Pope, S. Ravela, J. Restrepo, A. Ridley, H. Samet, S. Shekhar, Intelligent systems for geosciences: an essential research agenda. Commun. ACM **62**(1), 76–84 (2018)
9. C. Golbreich, S. Zhang, O. Bodenreider, The foundational model of anatomy in owl: experience and perspectives. J. Web Semant. **4**(3), 181–195 (2006)
10. L.H. Karlsen, M. Giese, Qualitatively correct bintrees: an efficient representation of qualitative spatial information. GeoInformatica **23**(4), 689–731 (2019)
11. E. Kharlamov, T. Mailis, G. Mehdi, C. Neuenstadt, Ö. Özçep, M. Roshchin, N. Solomakhina, A. Soylu, C. Svingos, S. Brandt et al., Semantic access to streaming and static data at siemens. J. Web Semant. **44**, 54–74 (2017)
12. E.M. Lidal, H. Hauser, I. Viola, Geological storytelling—graphically exploring and communicating geological sketches (2012)
13. C. Mathieu, Moray Firth—central north sea post well analysis (Oil & Gas Authority, 2015)
14. J. Meseguer, Twenty years of rewriting logic. J. Logic Algebraic Program **81**(7–8), 721–781 (2012)
15. E. Monteiro, T. Østerlie, E. Parmiggiani, M. Mikalsen, Quantifying quality: towards a posthumanist perspective on sensemaking (Springer International Publishing, Cham, 2018), pp. 48–63

16. M. Natali, T.G. Klausen, D. Patel, Sketch-based modelling and visualization of geological deposition. Comput. Geosci. **67C**, 40–48 (2014)
17. NPD, Play definition (2019). https://npd.no/en/facts/geology/plays/
18. N. Oreskes, K. Shrader-Frechette, K. Belitz, Verification, validation, and confirmation of numerical models in the earth sciences. Science **263**(5147), 641–646 (1994)
19. T. Østerlie, E. Parmiggiani, E. Monteiro, Information infrastructure in the face of irreducible uncertainty, in *5th Innovation in Information Infrastructures (III) Workshop*, 7th–9th Nov, Rome (2017)
20. B. Parsia, P. Patel-Schneider, B. Motik, OWL 2 web ontology language structural specification and functional-style syntax, in *W3C, W3C Recommendation* (2012)
21. P. Rey, Introduction to structural geology. http://geosci.usyd.edu.au/users/prey/Patrice_Intro_to_SG.pdf
22. R.D. Shell, Play based exploration: a guide for AAPG's imperial barrel award participants (Shell Exploration and Production, Rijswijk, 2013), pp. 1–51
23. M.G. Skjæveland, A. Gjerver, C.M. Hansen, J.W. Klüwer, M.R. Strand, A. Waaler, P.Ø. Øverli, Semantic material master data management at Aibel, in *International Semantic Web Conference* (P&D/Industry/BlueSky, 2018)
24. B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, C.J. Mungall et al., The obo foundry: coordinated evolution of ontologies to support biomedical data integration. Nat. Biotechnol. **25**(11), 1251–1255 (2007)
25. Y. Svetashova, B. Zhou, T. Pychynski, S. Schmidt, Y. Sure-Vetter, R. Mikut, E. Kharlamov, Ontology-enhanced machine learning: a Bosch use case of welding quality monitoring, in *International Semantic Web Conference* (Springer, Berlin, 2020), pp. 531–550
26. A.K. Turner, C.W. Gable, A review of geological modeling (2007)