UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Fraunhofer IAIS

Semi-Automatic Feature Detection in Volume Data through Anomalies in Local Histograms

Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Jan Christoph Beutgen

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Dr. rer. nat. Stefan Rilling (Fraunhofer IAIS)

Koblenz, im September 2017

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Acknowledgments

First, I would like to thank Prof. Dr. Stefan Müller and Dr. Stefan Rilling for agreeing to be the supervisors of this thesis and for assisting me in the completion with their invaluable expertise.

Secondly, I would like to thank my supervisor Dr. Manfred Bogen and my colleague and friend Jan Bender from Fraunhofer IAIS for their constant support, ideas and valuable input on this topic.

Furthermore, I am obliged to all participants of the evaluation and in particular Prof. Dr. Tezkan, Prof. Dr. Hinzen, Prof. Dr. Lehmann, Dr. Hartwig von Hartmann and Fabian Patterson for facilitating the expert interviews. Additionally, I am very thankful for all the advice and feedback, which I received from the members of the VRGeo Consortium [11].

Finally, I want to thank my girlfriend, my dear family and my friends for supporting me during the last months of this thesis.

Zusammenfassung

In vielen wissenschaftlichen Bereichen fallen heutzutage riesige Datenmengen an, wodurch es mitunter aufwändig und zeitintensiv ist die wirklich wichtigen Bereiche zu detektieren. Insbesondere gilt dies für riesige seismische Volumendatensätze, die für die Erkundung von Öl- und Gasvorkommen benötigt werden. Da die Daten sehr umfangreich sind und die manuelle Analyse sehr viel Zeit in Anspruch nimmt, kann ein semi-automatischer Ansatz zum einen die dafür benötigte Zeit reduzieren, zum anderen aber mehr Flexibilität als ein vollständig automatischer Ansatz bieten.

In dieser Masterarbeit wird ein Algorithmus entwickelt, der interessante Regionen in seismischen Volumendaten über Anomalien in lokalen Histogrammen automatisch detektiert. Des Weiteren werden die Ergebnisse visualisiert und verschiedene Hilfsmittel für die Interpretation der Daten entwickelt. Der Ansatz wird durch Experimente mit synthetischen Daten und Experteninterviews auf der Basis von realen Daten evaluiert. Abschließend werden verschiedene Verbesserungen aufgeführt, die dabei helfen können den Algorithmus in den Interpretationsablauf der Seismologen zu integrieren.

Abstract

In scientific data visualization huge amounts of data are generated, which implies the task of analyzing these in an efficient way. This includes the reliable detection of important parts and a low expenditure of time and effort. This is especially important for the big-sized seismic volume datasets, that are required for the exploration of oil and gas deposits. Since the generated data is complex and a manual analysis is very time-intensive, a semi-automatic approach could on one hand reduce the time required for the analysis and on the other hand offer more flexibility, than a fully automatic approach.

This master's thesis introduces an algorithm, which is capable of locating regions of interest in seismic volume data automatically by detecting anomalies in local histograms. Furthermore the results are visualized and a variety of tools for the exploration and interpretation of the detected regions are developed. The approach is evaluated by experiments with synthetic data and in interviews with domain experts on the basis of real-world data. Conclusively further improvements to integrate the algorithm into the seismic interpretation workflow are suggested.

Contents

1 | Introduction

The exploration and analysis of volume data are difficult and time-intensive tasks. This is due to the fact that modern methods of acquiring huge datasets have surpassed our ability of automatically analyzing these and extracting relevant information. Especially this is true for applications in the field of searching for oil and gas, where huge areas are scanned and processed to a dataset with multiple terabytes of data. This data has to be analyzed by a seismic interpreter, which is also called *expert* in the context of this thesis. The aim of the interpretation is to identify important regions with accumulations of oil and gas. Often no prior knowledge of the structures below the surface is available. Thus domain experts have to orient in an unknown, large-scale dataset and gather supportive background information, before the actual interpretation of the dataset can be performed.

While the traditional workflow was to print the dataset slice by slice on sheets of paper, modern technology has revolutionized the procedure. Sophisticated software suites such as OpendTect [16] or Petrel [41] use proficient techniques to display the dataset in a three-dimensional way. Different tools like complex transfer function editors with multiple dimensions allow to isolate certain structures and assign optical properties to them. This improves the capabilities of data exploration even further. Yet the identification of important regions and the extensive analysis of these are very time-consuming, since aforementioned methods require a lot of manual interaction in multiple domains. For instance, the generation of an applicable transfer function includes a lot of parameter tweaking with continuous examining of the changes in the rendering view. This is especially a challenge, since the location of interesting features is typically not known and a user invests a lot of time exploring currently irrelevant regions.

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. The aim of these is to detect relevant features automatically and present them to a user. However these approaches have multiple challenges. At first, they usually require an elaborate and time-intensive training step, that adapts the algorithm to the specialties of the targeted features. Therefor ground-truth data with millions of interpreted samples has to be available. This is a challenging task in the domain of seismic volume data, which is usually not published by the owning companies. And even if enough data could be gathered, domain knowledge and expertise are required to interpret it for the usage as qualified training data. This process takes time and is expensive [5]. Finally, for every desired feature, the training has to be repeated, always requiring new interpreted data containing the target feature.

While the fully manual approach grants a certain flexibility with the interpretation under the constraint of a bigger time exposure, the fully automatic approach is feature-dependent and thus not generic enough to be used solely. A combination of these two could on one hand lead to results faster, compared to a manual interpretation, and on the other hand remain generic, so that relevant regions can be detected independently from the features contained. Such a semi-automatic approach would detect interesting regions, which could then be interpreted manually. This way the time of screening the dataset and searching for interesting regions could be saved and multiple starting points for the interpretation could be offered directly to an expert.

On a global scale the seismic datasets mostly consist of Gaussian distributed amplitude values. But 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 [26]. 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 [50]. 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 they have a better chance of appearing in the value distribution of the respective region's local histogram. 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 important, since the interpretation of them could be a direct solution to finding hydrocarbons and to defining the lithology [48]. Thus, the detection of the anomalous regions might be a suitable approach for a semi-automatic algorithm.

The aim of this master's thesis is to develop an algorithm, which determines interesting regions automatically by using information of local histograms. Additionally a visualization of the detected regions and multiple tools to support the exploration and interpretation are proposed.

The outline of this thesis is as follows. In chapter 2 the required geological background is introduced. Chapter 3 provides information on the field of volume data in computer science. Related work on the topic of feature detection is discussed in chapter 4. The conception and procedure of the algorithm are documented in chapter 5. Chapter 6 presents the visualization of detected regions and the developed tools for the interaction and interpretation. The algorithm and visualization are evaluated in chapter 7. Finally the conclusion of the approach is given in chapter 8, together with an outlook and suggestions for future work.

2 | Geological Background

The algorithm developed in this thesis is designed to support the interpretation of seismic data, which is performed by the oil and gas industry. Hence this chapter introduces the geological background required in order to follow along. The geological process of deposition in the subsurface and important features within are explained, and the journey of seismic data from the recording to the computer screen of a seismic interpreter is described.

2.1 Formation of Petroleum Deposits

Petroleum, which has already been used by humans for thousands of years, even prior to metals and coal, is still one of the most important commodities in modern industry. Thus the optimization of the search for it is of major importance. One vital step therefor, is to understand the geological processes, which lead to the accumulation in the subsurface. Most of the oil and gas fields were found in sedimentary basins. These develop over tens of millions of years and contain fragmented material, which hardens into layers of rock. This process starts with the subsidence of the land, allowing the sea to extend. This phase is the so-called *transgression*.

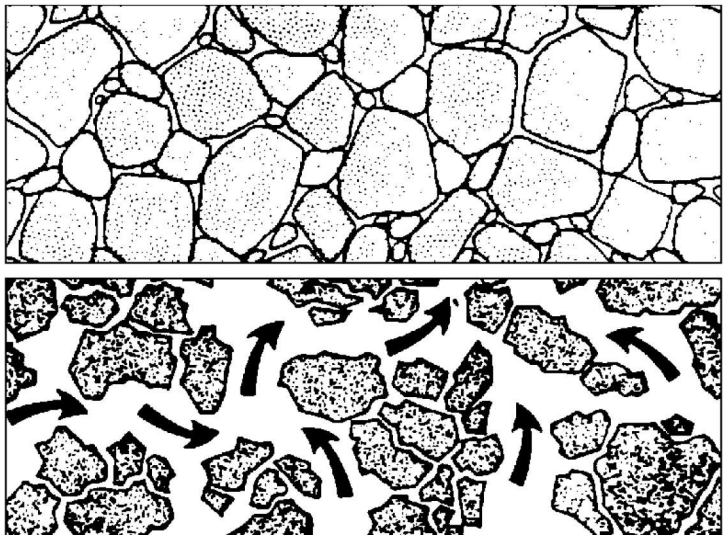


Figure 2.1: Illustration of a porous rock (top) and a permeable rock (bottom) [50]

The surrounding mountains are slowly eroded by wind, rain and ice. Parts of the rock are streaming down rivers and are finally deposited. The amount of sediment, which is washed down the rivers, increases over time and begins to overtake the sea. Further arriving material is then stacked around the edges to form beaches and deltas at the river mouths. As the shoreline grows, the

sea is pushed back again in the phase of *regression* [50]. One period of transgression and regression is called a *sedimentary cycle*. Many of these cycles have to be performed in order to form a sedimentary basin. The lower layers are compressed by those above with increasing weight. The material varies from sand grains and plants, over corals and shells up to ashes from volcanic eruption, which all are transformed in diverse types of rock. These different types vary in their porosity and permeability, which are key factors in the deposition of petroleum. As it can be seen in figure 2.1 a rock is porous, if it has voids and small cavities between individual grains. If these voids are connected, the rock is permeable and fluid is able to pass through it [50].

Oil and gas originate from organic matter, such as the remains of plants and animals. Therefor these have to be accumulated in fine-grained sediments, as for example shale or limestone. They have to be located between two permeable layers, one of the transgressive phase on top and one of the regression below. These circumstances in combination with a deficit of oxygen and a depth with high enough temperatures ($110° - 130°$C) can lead to the transformation of the organic matter inside the layer to oil and gas. Minor fluctuations in transgression and regression are very important for this step, as they bring together potential source rocks and so-called *reservoir rocks*, which have sufficient porosity to contain a significant volume of hydrocarbons. The required permeability occurs either naturally, as for example in sand and limestones, or it can be the effect of later earth movements like faulting and folding. The oil and gas then moves from the source rock to the reservoir rock in a process referred to as *migration*, which is displayed in figure 2.2 [50].

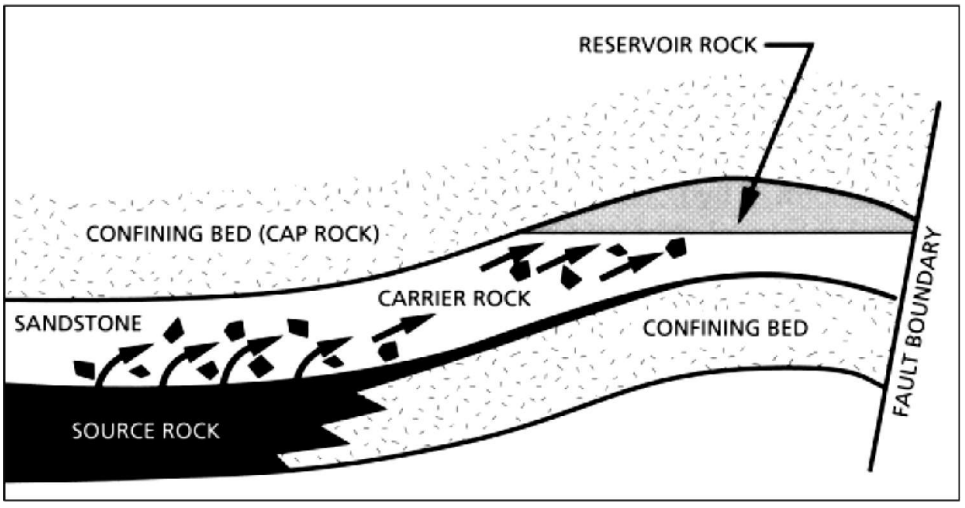


Figure 2.2: Process of oil migration [50]

A vast majority of pores below the water table are filled with water. Thus the migration of oil and gas is related to hydrology. While the water movement is very slow with just a couple of centimeters per year, the path of the migration can easily cover hundreds kilometers. Thus, the whole process can last a very long period of time. Finally, when the oil reaches the reservoir rock, it is stored there. Due to its lower density compared to the water contained in the rock, it moves to the top. This upward movement continues, until a fine-grained impermeable layer is reached. The hydrocarbons are now trapped and cannot move any further. These so-called *petroleum traps* vary in their structure and multiple groups can be distinguished [50].

The first group are the *structural traps*. One representative of these is the *anticlinal trap* which occurs, when rock layers fold into a dome shape. An illustration of this trap can be seen in Figure 2.3. The hydrocarbons can migrate into the dome from all sides and accumulate in the top, where the cap rock stops the migration [50].

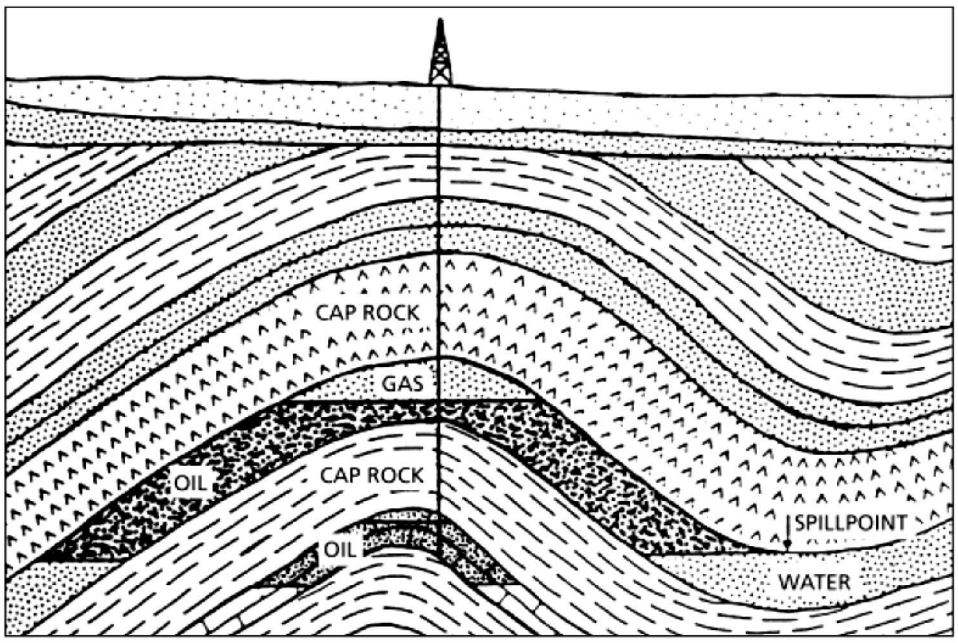


Figure 2.3: Hydrocarbons in an anticlinal trap [50]

Another structural trap is the so-called *fault trap*, which is formed by a *fault plane*, interrupting the direction of migration. A fault plane is a displacement of the earth layers. In addition to the process of migration, an example for this kind of trap can also be found in figure 2.2. The oil cannot proceed its way, as it is stopped by the fine-grained material in the fault or by an impermeable layer on the other side of it. It is mandatory, that the faulting occurs prior to the migration and that it completely seals the reservoir [50].

The so-called *salt dome* is a further popular example for a structural trap. It appears when salt rises vertically, while being less dense than the overlying structures and then makes its way through the sedimentary column. The salt dome, together with the faults and anticlines at the head of it, provides a number of trapping mechanisms [50]. An illustration of these can be seen in figure 2.4.

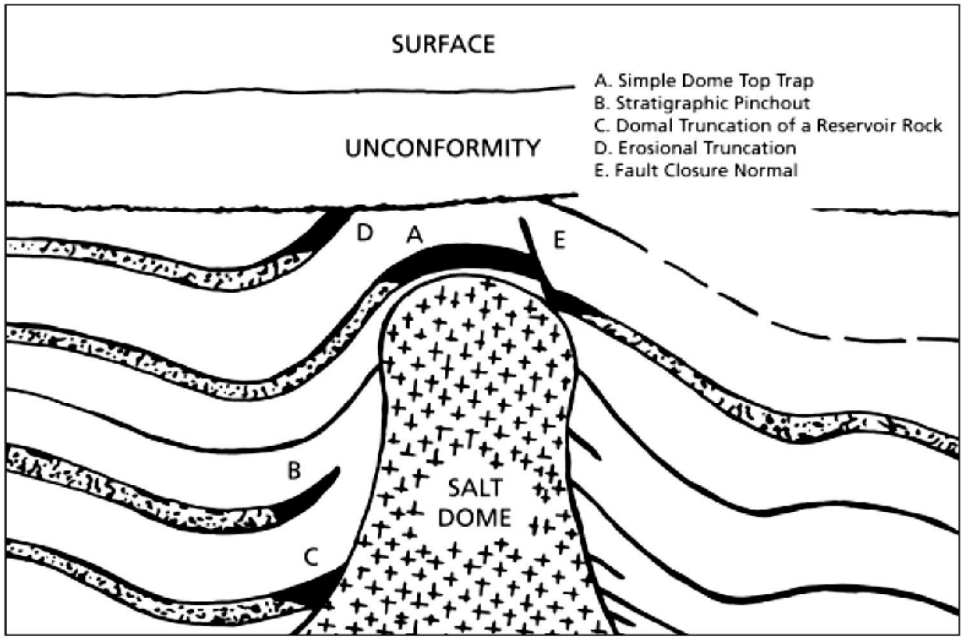


Figure 2.4: Salt dome structure [50]

Besides the structural traps, there are *stratigraphic traps* and *hybrids*, which combine both groups. As these are not of major importance for this thesis, they are not described here. However a detailed description of them can be found in [50].

2.2 Acquisition of Seismic Data

Oil and gas are the result of a combination of less common events, including existence and burial of the source rock, a reservoir in range and a trap in which the hydrocarbons can accumulate. The prediction of where these events have occurred is not easy [50]. Randomly drilling a hole for this purpose is not an option, as it can easily cost between 10 and 50 million dollars and does only provide information at discrete locations [14, 26]. Instead the industry has invented multiple techniques to minimize the drilling requirement in order to make the search for oil and gas more efficient. These are reviewed in this chapter.

One of the main goals of geophysical exploration is to understand and predict the types of rock below the surface. While this was traditionally achieved by mapping the geology and finding relations of various rock units, modern technology now allows to construct models of the subsurface in great detail. The data can then be displayed on a computer screen in order to understand the geologic formation and detect features, that could indicate the presence of oil and gas. One method that allows to acquire the data is based on the use of sound waves, which travel through earth. Hence they are termed *seismic waves*. Similar to ultra sound imaging in medicine, which depicts the human body, the seismic waves can be used to depict the geological structures of the earth. In comparison to the short wave length used in ultra sound imaging, the seismic waves have a longer length, that allows to reach areas many kilometers below the surface. The waves are generated by shooting controlled pulses of sound into the ground. While the waves travel through earth, a part of them is reflected at geological boundaries within the subsurface and send back to the top. The reflected waves are picked up by an array of recording devices. For example in marine environments, the source for the sound are air guns, which fire pulses of compressed air. The reflected waves are recorded with hydrophones floating on the water. In a seismic survey onshore the sound is created by vibrating trucks and geophones, that spread across the surface [26]. The diagram of a marine seismic survey can be seen in figure 2.5.

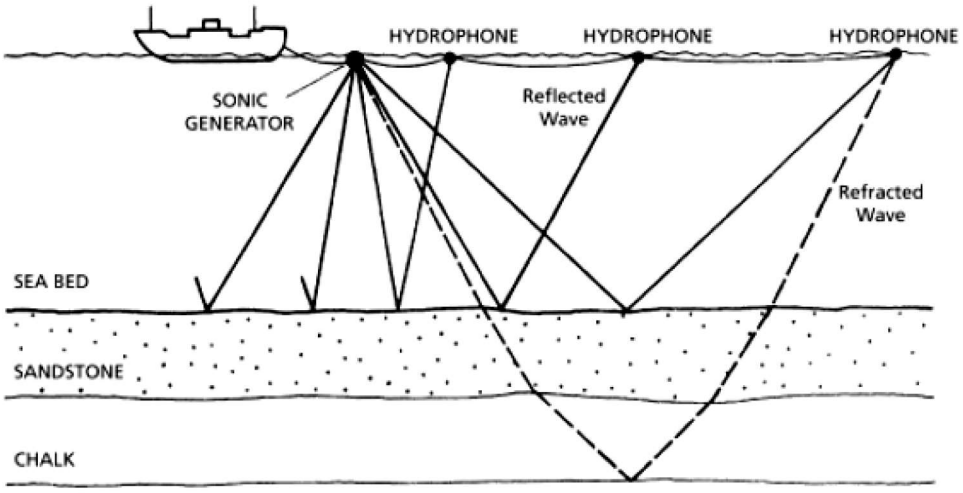


Figure 2.5: Seismic survey with typical reflection and refraction wave patterns [50]

The reflection of a seismic wave is a so-called *event*. By measuring the arrival times of the events at certain distances from the source with neighboring detectors, the travel times of the wave can be converted into depth values. Afterwards the distribution of the geological interfaces can be sys-

tematically reconstructed. This method is qualified for mapping layered sedimentary sequences and is thus frequently used in the search for oil and gas. Additionally the lithology of a rock can be deduced, since the velocity of the seismic wave changes depending on the type of rock that it travels through [26].

One of these measurements corresponds to a seismic trace, which is a one-dimensional, time-dependent function $X(x, y, t)$. While x and y describe the spatial position, t is the travel time in the vertical direction. Note that there are methods in seismic processing to convert this time into a depth value. By combining multiple seismic traces in an array of equivalent distance, a two-dimensional cross-section through the earth can be generated. Extending this idea to the third dimension results in a volume of the subsurface [44]. This volume is a field of scalar values, in which every value represents one seismic amplitude. Each of the cells, which are also called *voxels*, relate to the spatial coordinate of the respective seismic event. A comparison between the different dimensionality of seismic data can be found in figure 2.6.

Figure 2.6: Comparison of 1D seismic signal, 2D seismic slice and a 3D seismic cube of the F3 dataset [15]

The volume datasets differ in the number of bits, used to store the amplitude information. While the processing of the measurements is done using 32 bits per data point, performance restrictions of the first interactive applications in the early 1980's led to the conversion of the preprocessed data to 8-bit [7]. This reduction minimizes the time of all computations and the storage space required for the data. Today datasets with 8, 16 and 32 bit are used. In case of 8-bit data, each voxel can take on a value between -128 and 127. Values that fall above or below the range, must be clipped. The amplitude is typically Gaussian distributed with a large number of zero

amplitudes and only a small number of very low and high amplitudes. The typical amplitude distribution can be seen in figure 2.7.

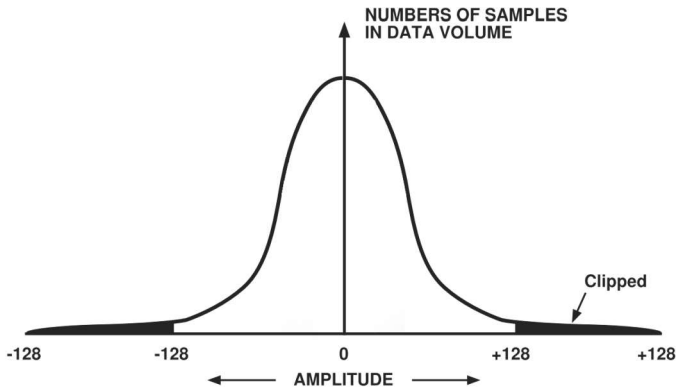


Figure 2.7: Statistical distribution of amplitudes in seismic data [7]

2.3 Seismic Attributes

The term *seismic attributes* describes all different kinds of information, obtainable from the seismic dataset. This can either be through direct measurements, or by applying logical- and mathematical operations to these [47]. One example of a seismic attribute, is the measured amplitude value at every data point. Further attributes can be generated with specific procedures, which typically base the calculation on information gathered during the data acquisition. These can for instance be time, amplitude or frequency values [7]. While attributes derived from time provide structural information, the amplitude- and frequency-derived attributes impart stratigraphic and reservoir information. Further attributes can be generated by utilizing the rate of change of these parameters (i.e. derivations) in time or space.

Seismic attributes can be calculated before or after the *stacking*, which is a process of averaging the data and eliminating offset related information and directional aspects. Thus it is differentiated between *pre-stack* and *post-stack* seismic attributes. The first group generates huge amounts of data and is therefore not practical for a first initial study. Yet, the group contains valuable information about fluid content or the orientation of fractures and is hence very important for a detailed interpretation. The later group offers a more manageable approach for observing the large amounts of data, which is the reason why most seismic attributes are created this way. Obviously, all of the previously mentioned parameters can be combined to generate hybrids. The attributes are computed once in a preprocessing step and are stored as an additional volume for later use. A variety of different

attributes was proposed throughout the years and all of them have their individual strengths and weaknesses. For instance the *instantaneous frequency* can lead to a better perception of faulted zones, while the *phase* gives a detailed visualization of stratigraphic elements. The same seismic slice with different attributes is displayed in figure 2.8. Subsequently a multitude of attributes can be useful for the interpretation of a single dataset. A detailed discussion on different attributes can be found in [47].

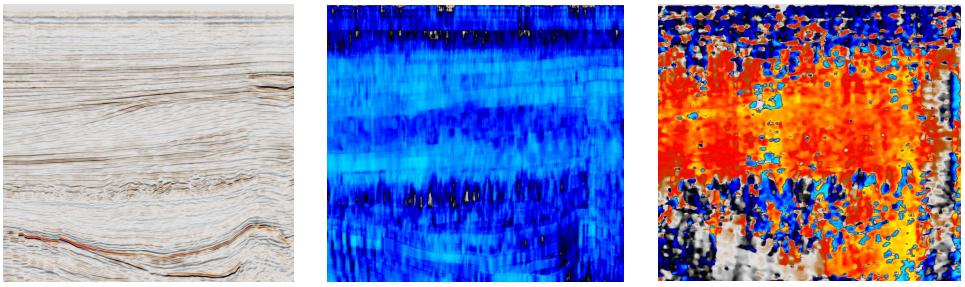


Figure 2.8: Seismic slice of the F3 dataset with different attributes [15]

All of the attributes used in the remainder of this thesis were generated with the open-source seismic interpretation software OpendTect [16].

2.4 Anomalies in Seismic Data

Since the developed algorithm is based on detecting anomalies in different regions of the dataset, this chapter introduces the term *anomaly* in the field of seismic volume data and introduces multiple causes of these. Basically it can be differentiated between two different types of anomalies in the seismic domain. These are *stratigraphic anomalies* and *amplitude anomalies*. Both are explained in the following sections.

2.4.1 Stratigraphic Anomalies

The first type of anomalies are stratigraphic ones. These are multiple kinds of structural features, such as the salt dome that is explained in section 2.1. Another example are *unconformities*, which are of fundamental importance in the stratigraphic studies, as they provide a basis of defining the seismic sequences [2]. In general they represent a period of non-deposition or erosion, followed by resumption of deposition. They can easily be identified through the angular relation between the unconformity plane and the overlying sediments. An example for an unconformity can be found in figure 2.9 Other structural feature of interest are *horizons*, which are horizontal events without geometrical deformation. However, they can also be affected by folding and faulting. The first of these events can happen in various

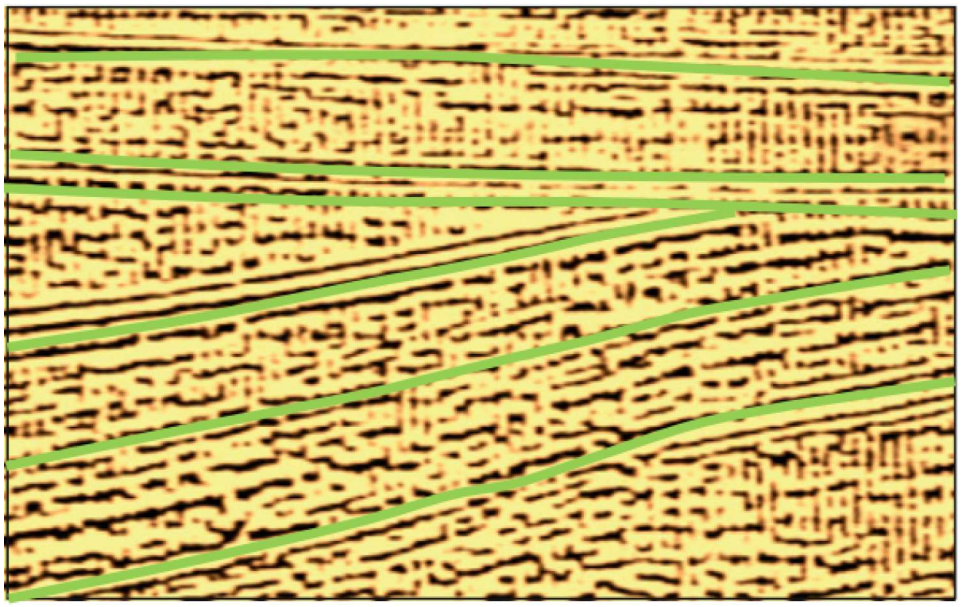


Figure 2.9: Typical example of an angular unconformity [2]

degrees of intensity and lead to the distortion of the horizon to e.g. anticlines. Furthermore faulting, which is a break in the reflection event with an additional shift in the vertical direction, can occur. These horizons are of importance to the stratigraphic interpretation, as they help to understand the geological processes of the earth. The folded and faulted horizons have to be detected and the interferences removed. This process is known as *migration* [2]. Examples for the different horizon types can be found in figure 2.10.

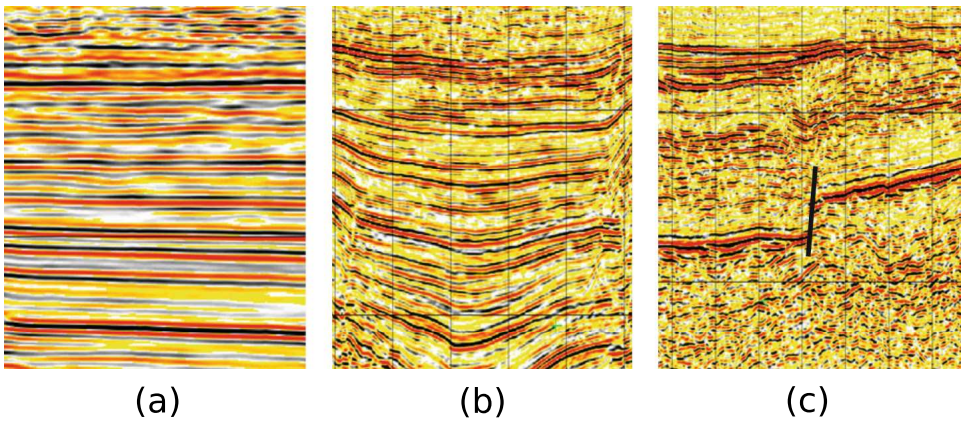


Figure 2.10: Examples of straight horizons (a), folded horizons (b) and one faulted horizon (c) [2]

2.4.2 Amplitude Anomalies

The Schlumberger Oilfield Glossary defines the term *amplitude anomaly* as

> *"an abrupt increase in seismic amplitude that can indicate the presence of hydrocarbons, although such anomalies can also result from processing problems, geometric or velocity focusing or changes in lithology."* [29]

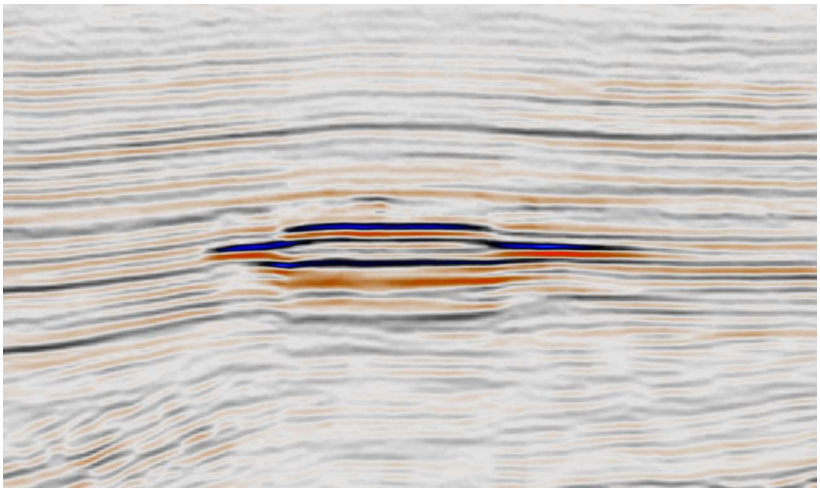


Figure 2.11: Bright spot on a seismic slice of the F3 dataset [15]

Generally those values higher than the average background amplitude levels, can be considered an anomaly [48]. These higher amplitude values are also referred to as *bright spots*, since they correspond to strong reflections and lead to bright colors on seismic sections. One example for such a bright spot can be found in figure 2.11. The spots can be produced by the presence of hydrocarbons in the rock formation and are hence also called *direct hydrocarbon indicators* (DHI) [2]. This is due to the anomalous amplitude change, that occurs at the edge of the accumulation, when the seismic wave leaves the hydrocarbons and moves back into water-filled pores. Thereby the seismic velocity is lowered, which leads to an increase in the reflection coefficient [50]. The interpretation of the DHIs could be a direct solution for the detection of hydrocarbons and could additionally support the investigation of lithology, which are both very important aspects of the interpretation of seismic volume data. Nevertheless these bright spots do not always correspond to the presence of hydrocarbons. Similar effects can be caused by coal seams or porosity changes of different rocks [50]. Another cause is the presence of artifacts, which are naturally not of interest for an interpreter.

3 | Volume Data in Computer Science

After the seismic data is processed, it can be displayed in a seismic interpretation software such as OpendTect [16]. The processes of visualizing the data and the interaction with it are described in this section.

There is a wide variety of techniques for generating images of a volume dataset. Note that these methods are not restricted to the seismic domain, but are frequently used in other scientific fields, as for instance in medical applications. The basic rendering is to map a seismic slice onto a 2D plane. This plane can then be moved through the dataset to display various slices. Additionally the data can be visualized in 3D by using *volume rendering*, which is a term that describes a whole range of techniques, that transform the scalar values of the volume dataset into a colorful visualization. Modern graphics processing units (GPUs) allow to use volume raycasting in real-time, which is the most popular method in volume rendering [17]. This is due to a high visual quality and the ability to adapt the visualization interactively. The principle of raycasting is explained in detail in the next section.

3.1 Volume Raycasting

The physical basis for volume rendering is a combination of light and the effects occurring through the interaction with the medium. These effects are for example emission, absorption and scattering. As the physically correct computation of all of these is very complex, a simplified version called the *emission-absorption model* is used for volume raycasting. In this model the medium can emit and absorb light, while neglecting the scattering aspect. This corresponds to the best trade-off between quality and time of computation. For background information on this model and the definition of the so-called *volume rendering integral* used for raycasting, the reader is referred to the respective chapter in [17].

The basic idea of raycasting is to cast rays from the camera's point of view through the volume. These viewing rays, which are handled independently from each other, are used to compute the volume rendering integral along their way. The definition of the integral is split up into multiple subsequent intervals, realized by sampling each ray at equivalent distances. This principle is clarified in figure 3.1. The contribution of color and transparency in the respective intervals can be evaluated. Therefor the *emission-absorption*

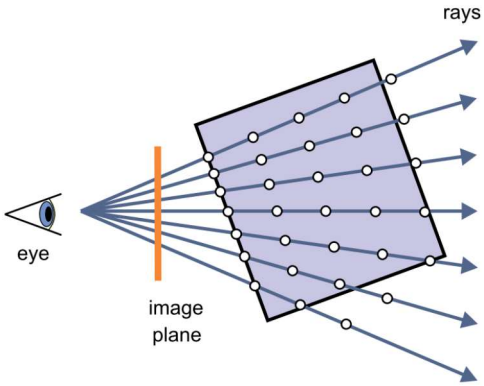


Figure 3.1: Principle of raycasting [17]

model is used to compute a composite color and opacity for each ray. Additionally a single scattering effect is approximated with a local illumination model, such as the *Phong* model [17]. The process of compositing solves the discretized volume rendering integral iteratively. Commonly the method of *front-to-back* compositing is used. It starts at the position of the camera and samples the rays at equivalent distances on their way through the volume. For every sample the color $\hat{C}_i$, consisting of three values for the color channels red, green and blue (RGB), and the opacity $\hat{\alpha}_i$ is calculated with

$$\hat{C}_i = \hat{C}_{i-1} + (1 - \hat{\alpha}_{i-1})\, C_i, \tag{3.1}$$

$$\hat{\alpha}_i = \alpha_{i-1} + (1 - \hat{\alpha}_{i-1})\, \alpha_i \tag{3.2}$$

and the initialization of

$$\hat{C}_0 = C_0, \tag{3.3}$$

$$\hat{\alpha}_0 = \alpha_0. \tag{3.4}$$

Thereby $\hat{C}_{i-1}$ and $\hat{\alpha}_{i-1}$ represent the accumulated color and opacity of the previous calculations. The source color C_i and the opacity α_i of the current sample can be determined by using a so-called *transfer function*, which is explained in section 3.3. Equation 3.1 is applied on every sample, updating the color and opacity along the way. The final results of these values are obtained once the ray has left the volume.

3.2 Histograms

The frequency of all values in the domain can be represented through a discrete function, which is called a *histogram*. In the context of volume data, these frequencies are the total amounts of the respective values in the area. The first step of computing a histogram is to partition the underlying data

space into a fixed number of n bins. Typically the number of bins equals the value range, so that every voxel falls precisely into one bin. The resulting quantized data structure is denoted as histogram H. Further h_i describes the quantity of voxels that have a value in the interval with the index i [39]. A large number of voxels with the same value corresponds to a homogeneous region. Since the scalar values of the amplitude follow a Gaussian distribution, as described in section 2.2, the amplitude histogram is also Gaussian distributed. Note that the histogram provides no information on the location of voxels [37]. The amplitude histogram of the F3 dataset [15] can be found in figure 3.2.

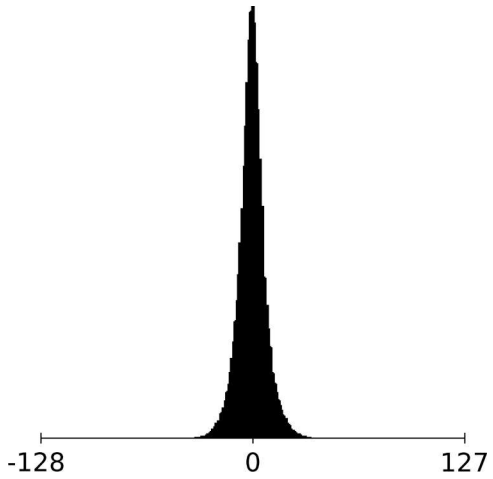


Figure 3.2: Amplitude histogram of the 8-bit (signed char) F3 dataset [15]

A histogram reflects the distribution of values in a global sense. The peaks correspond to those values, that are present in a large portion of the dataset. Subsequently small features might not be noticeable in the global histogram. One solution for this challenge is to generate a local histogram, which is only covering a certain region of the dataset. The chance of detecting small features in the local histogram is much higher, since they are not occluded by the values only present in other parts of the dataset [37].

Another possibility of enhancing the expressiveness of a histogram is to add a second dimension to it, resulting in a 2D Histogram. For example another seismic attribute, which are described in section 2.3, could be used as an additional dimension. The principle behind the 2D histogram resembles the one-dimensional case. A two-dimensional array with size equal to the number of possible values in the respective dimensions is used to store the frequency of every combination of two values. The challenge of the increased dimensionality is to find an efficient visualization of the frequencies. Whereas the one-dimensional histogram relies on a simple bar chart, the value ranges of the two attributes require already two dimensions and

the height of the bars would have to be displayed in the third dimension. However this visualization is not effective due to occlusions [37]. Another approach is to map the value frequency to different gray values. A frequency of zero can be represented by a white color and is thus not visible. The highest frequency value can be displayed in black. Values in between receive a linearly interpolated gray value [37]. Figure 3.3 shows on the left a visualization of the 2D histogram between the amplitude and similarity attributes of the F3 dataset [15]. On the right side is the exact same histogram without mapping the frequency to color. Instead every value that is present in the dataset receives a black color. The huge difference between the two histograms shows, that lots of values can get lost when visualizing the frequency. Thus it can be useful, to abandon the representation of the frequencies and to display all of the present values in the same color. Two-dimensional histograms can be used for a 2D transfer function, as it is explained in section 3.3.2.

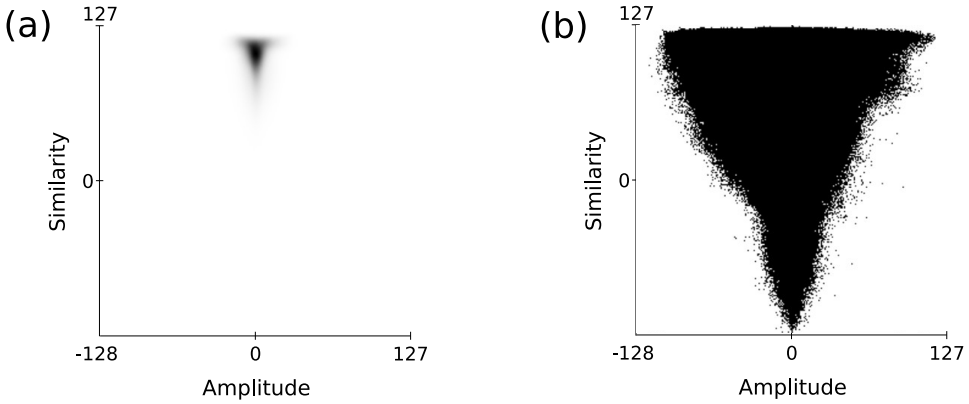


Figure 3.3: Comparison of a 2D histogram with frequency mapping (a) and without frequency mapping (b)

3.3 Transfer Functions

The abstract scalar values of the dataset cannot directly be used to determine the emission and absorption, as required for the volume raycasting. Instead a transfer function has to be applied to define color and opacity for the different data values [17].

3.3.1 One-Dimensional Transfer Functions

The transfer function uses the scalar data values of the dataset and assigns optical properties to them, like a RGB color and an opacity. Thereby the voxels are not mapped individually. Instead a piecewise function specifies a whole intensity range at once. Such a piecewise function can be seen on the

right side of figure 3.4. In the top left corner, the full dataset with its scalar values interpreted as gray levels is displayed. After applying the transfer function, the volume rendering contains colored and transparent parts, as seen in the top right corner.

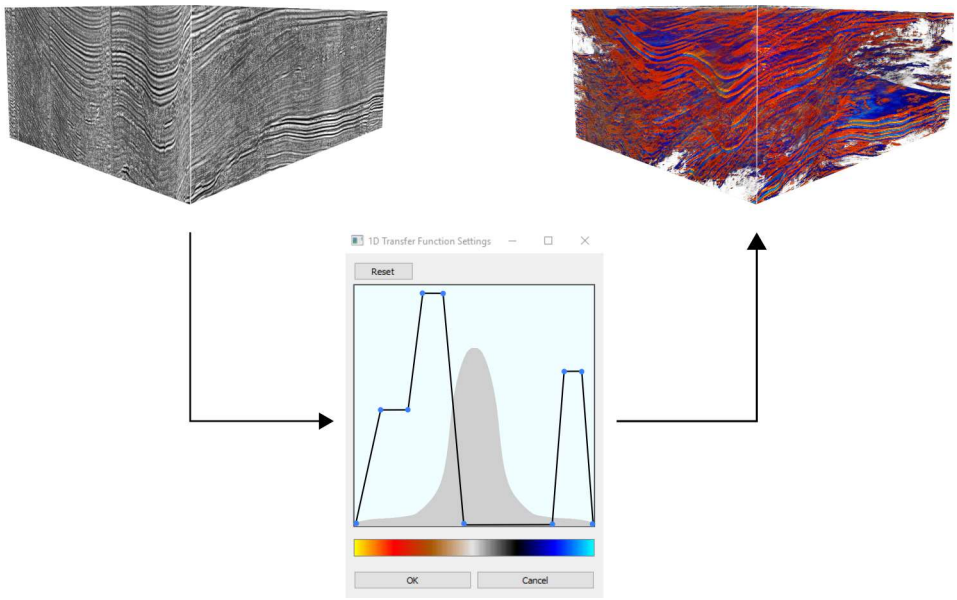


Figure 3.4: Transfer function editor with color table and corresponding volume visualization

The transfer function can be adapted by changing the position of the control points, shown in blue in figure 3.4. Additional points can be added to refine the line segments and existing ones can be removed to coarse them [37]. Their position on the editor's y-axis, determines the opacity of the corresponding scalar value. The opacity of values that lie in between two control points is determined by interpolating between these. Additionally a color has to be defined for the scalar values. This can be done by defining a color table with one entry for each possible value of the data range. Then the scalar values can be used as an index to sample the table and retrieve the respective color. This table should be shown in the editor, as it connects the visualization with the underlying data values. In figure 3.4 the table is displayed on the x-axis of the editor. As explained in [17] the histogram can support the process of transfer function generation and should thus be displayed in the background of the editor.

The transfer function design is a tedious and time-consuming procedure, which requires a detailed understanding of the structures present in the dataset [17]. In order to facilitate this process, changes of the transfer function should be applied to the volume rendering in real-time as visual

feedback. The transfer function can be represented as look-up table in form of a 1D texture that can be evaluated for every sample while performing the volume raycasting. Therefor the scalar value is read out of the volume texture and used as a look up value to determine color and opacity. This information can then be used to perform the front-to-back compositing.

3.3.2 Multi-Dimensional Transfer Functions

The one-dimensional transfer function assigns exactly one color and opacity to a given scalar value. However it is not guaranteed, that different features in the dataset also differ in their intensity. Hence a one-dimensional transfer function cannot always be used to isolate features as required. One possibility to deal with this issue is to expand the transfer function domain and take into account further information of the volume data. This can for example be another seismic attribute, which are described in section 2.3. The aim is to improve the possibility of isolating distinct features in the volume dataset. An example is shown in figure 3.5, where the dentin of the human tooth is to be isolated in a medical dataset. The one-dimensional transfer function, shown on the left side of the figure, fails at the given task as the enamel boundary of the tooth has the same data value. The 2D transfer function uses the gradient magnitude in addition to the raw data values to isolate the dentin successfully, as it can be seen on the right.

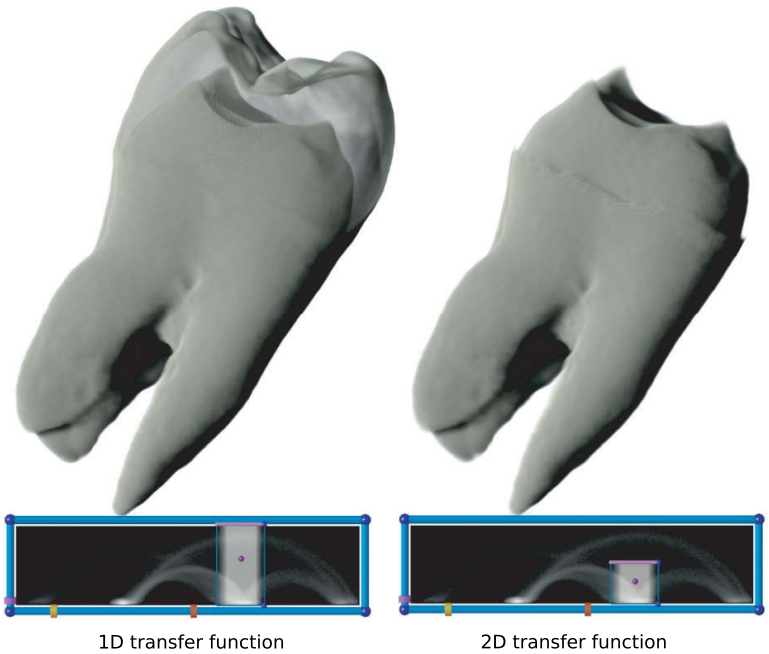


Figure 3.5: Advantage of a 2D transfer function, when isolating the dentin of a human tooth [28]

The drawback of using multiple attributes is that additional dimensions leads to an increase of effort in the transfer function design. Furthermore the implementation of an appropriate user interface for the editor is more challenging, when dealing with multiple dimensions that have to be visualized. The key is to find a balance between sufficient flexibility and reduced interaction effort [37]. Since the option to use control points is not sufficient anymore, another method of selecting the visible values is required. Typically the 2D histogram is visualized, as described in section 3.2 and can be used as basis for an editor [37]. Simple primitives, such as rectangles, trapezoids and triangles, can then be applied to the histogram to select the desired values. These primitives can be modified in position and size. Only values that are on the inside are visible in the volume rendering. Additionally a color and opacity can be assigned to each primitive. Respective values are then visualized with these optical properties. An example of a 2D transfer function editor can be seen in Figure 3.6. In this case the look-up

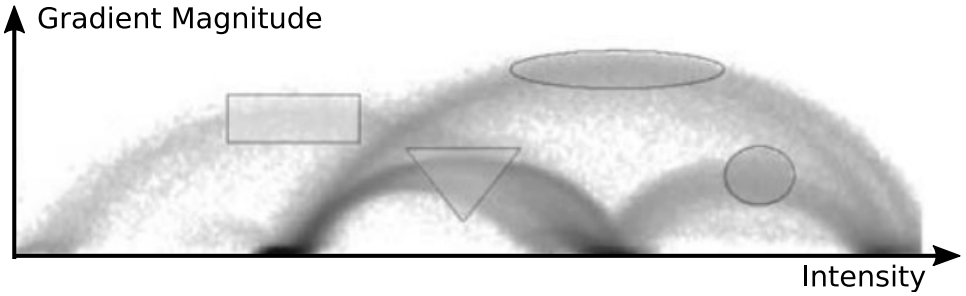


Figure 3.6: 2D transfer function editor [37]

table for the rendering also needs to be two-dimensional, with width and height matching the resolution of the histogram. During the ray traversal the two volume datasets of the attributes have to be sampled at the current position and the combination of the two values is used as coordinate for the look-up table. The stored color and opacity are read out of the texture and applied to the current voxel.

The idea can be extended to beyond two dimensions in the transfer function. Though with every additional dimension the representation of the editor and the interaction with the values get more complicated.

3.4 GPU Computing

The developed algorithm depends on a fast execution. Hence GPU computing with CUDA [13] is an important part of the implementation and is introduced in this chapter.

GPU computing covers a range of approaches to utilize the computational power of GPUs for calculations, which do not necessarily have to be graphics-related. Modern GPUs are highly optimized data-parallel streaming processors. In comparison to the central processing unit (CPU), they do not execute a sequential program. Instead multiple threads run in parallel and cooperate to accelerate the execution. A couple of constraints hamper the application of the GPU for general purposes. Since it was originally designed for accelerating graphical operations, the logic of the computation has to be adapted to data types usable in the context of the graphics API. With the rising popularity of GPU computing, multiple APIs, such as CUDA, were introduced to facilitate the communication with the GPU and to utilize its computational capabilities.

The modern architecture of the GPU contains multiple components that are specialized on allowing easier access to its power for general purposes [40]. CUDA is based on the C programming language, extended by additional keywords for special features of the CUDA architecture [40]. The programming model is based on a combination of CPU and GPU code. As most of the code of an application tends to have a sequential nature, the performance of the GPU can complement these parts. The CPU is therefore called the *host*, while the GPU is referred to as the *device*.

A CUDA program consists of one or more phases without data-parallelism implemented in the host code and phases with parallelization called *kernels*. The kernels are executed on the GPU in numerous threads. All the threads generated by a kernel form a *grid*. When all threads have completed their tasks, the grid terminates and the host continues the execution of the program. Thereby it is important, that all threads must execute the same kernel function. The grid is organized in a two-level hierarchy, which is also illustrated in figure 3.7. As it can be seen, the threads in the grid are arranged in multiple blocks. Note that the number of blocks and threads is typically way higher, but was minimized for the sake of illustration. Hereby it is important, that each block contains the same amount of threads. In the example, grid one consists of a two-dimensional array of 2×2 Blocks, each containing a three-dimensional array of $4 \times 2 \times 2$ threads. In general blocks and threads can be arranged in three dimensions, but not all three of them have to be used. The operations in the kernel can make use of the respective thread number, so that every thread processes the correct portion of data. Therefor built-in variables are provided, containing the index of the active thread or block and the dimensions of the grid and the blocks respectively. The global index of the thread can be calculated with these variables and used to access the correct part of input data, needed to perform the operations.

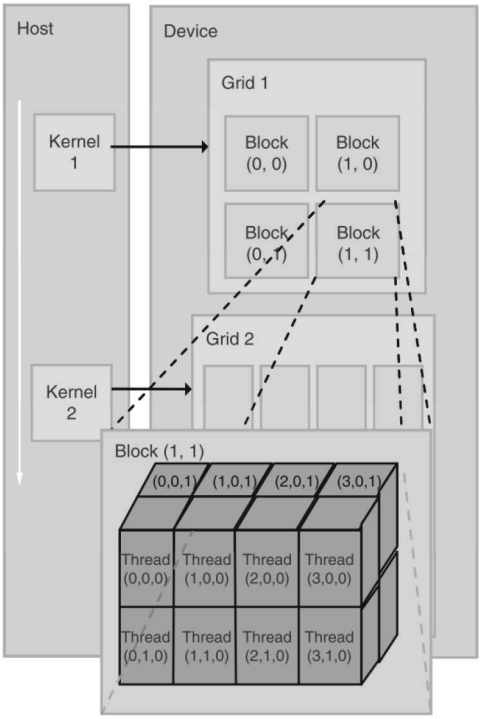


Figure 3.7: NVIDIA CUDA grid architecture [27]

One of the challenges of GPU computing is to design the application in a way, that the limited on-chip memory can store the data needed for the computations. Furthermore the restriction to certain data types and the low-level programming hamper the application design and the parallelization of the computations. But if these obstacles are overcome, the capabilities of the GPU can provide an enormous performance boost. However, the CPU still performs better on sequential tasks than the GPU. Thus GPU computing is not a panacea [27].

4 | Related Work

The idea of detecting important regions and features in volume data is not new. This chapter introduces related approaches, which helps to classify the contributions of this thesis thematically. At first a distinction to other methods of automating the seismic interpretation is done. Afterwards the basics of comparing histograms via distance measures are explained and some examples of the practical application are given.

4.1 Distinction to related approaches

In contrast to the semi-automatic approach developed in this thesis, an alternative is to rely on a fully automatic interpretation. One method for that is the technique known as *Deep Learning*, which has recently gained popularity and is now one of the biggest areas of research in computer science [5]. When it comes to deep learning, the computer learns to solve extremely big and complex problems, achieved by utilizing a so-called *neural network* that mimics the processes of the human brain. The network learns to solve a task by training on classified, ground-truth input data. It uses this data in a complex training step, to learn how to classify the input. This training step is based on the minimization of a cost function with millions of parameters. Thus finding the minimum of this function is not an easy task and requires time. After the ability of decision making was trained, the network can be used to classify unknown, real-world data [5]. Deep learning was mostly used in image processing and pattern recognition tasks, such as the identification of traffic signs in the automotive industry. But the method can also be applied to the use of volume data as shown in [23]. One example of the results on seismic data can be found in figure 4.1, where a neural network was trained to detect fault and channel structures. The images in the top row show the targeted features on time slices of the Parihaka [42] and F3 [15] datasets. Channels are included in the red bounding boxes and faults in the blue ones. The bottom row shows the results of the detection, using the same color coding. One can easily see, that the targeted features were detected by the neural network. Yet there are some false positives in between.

The big challenge of deep learning in the seismic domain is the lack of annotated data for the training process. This is due to the high value of the data to the oil and gas industry. While the companies own a huge amount of datasets, only a couple of them are available to the public. Hence the amount of training data is insufficient, compared to the millions of samples that are usually needed to train a neural network with satisfying results [23].

But the pure collecting of data is not enough. Experts have to interpret and label the data, which is very time-intensive. The shortage of training data might impair the reliability of the detection capabilities of the neural network [5]. Another drawback of deep learning is that the network has to be trained specifically on certain features to detect. The required training data has to be adapted towards the targeted feature. A valid training dataset could for example consist of 300000 samples of seismic slices containing a fault and negative examples without this structure [23]. For every additional feature that the network should be able to detect, new training data has to be generated. However, if a neural network has been trained and is reliable, a human expert would only be required to approve the results of the neural network. This could drastically reduce the time of the interpretation. Thus this approach is very promising. The deep learning approach is

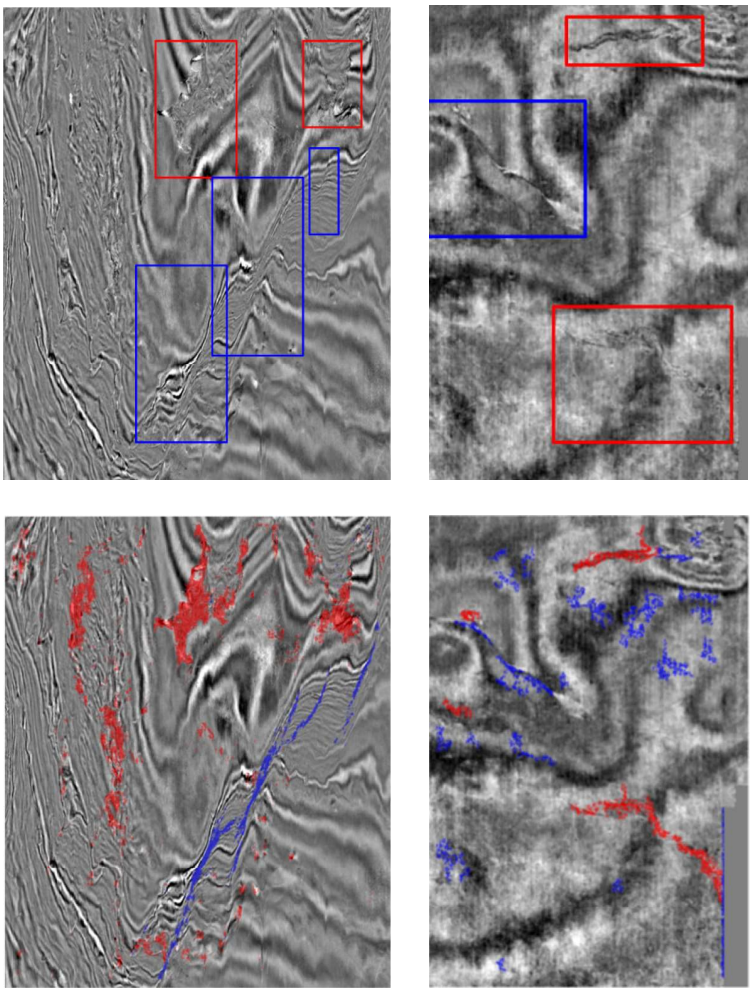


Figure 4.1: Targeted features (top) and detection results of a neural network (bottom) [23]

a *supervised* learning method, as it is based on labeled training data, which can be categorized into known classes. An alternative is to use an *unsupervised* learning algorithm, that utilizes unlabeled data to predict the class membership. This is done by automatically identifying patterns and natural clusters in the data distribution. Unsupervised learning can also be used to detect certain geological features, as it was shown in [32] for the detection of channels, which are concave streams of water and sediment [29]. These channels are important to detect during the seismic interpretation as they can cause the formation of stratigraphic hydrocarbon traps. One enhancement of the unsupervised learning method is to increase the value of the results by incorporating additional a priori information. Such an approach was presented in [30] and covers a whole workflow for using machine learning for seismic quantitative interpretation. Domain knowledge is included in the unsupervised learning to speed up the training process and to allow for a better conclusion on hydrocarbon properties.

But the learning-based techniques are not the only possibility of automating the interpretation. Other automatic approaches were optimized to fulfill specific tasks. One example is the automatic interpretation of all faults, unconformities and horizons in 3D seismic data, as shown in [51]. The idea is to perform a processing procedure to automatically extract all aforementioned structures. Stepwise results can be seen in figure 4.2. Thereby faults are extracted at first, as they complicate the extraction of horizons, which are interrupted by the faults and can otherwise not be identified easily. Then the detected faults are reversed, which can approximate the earth's state prior to the faulting. Afterwards the unconformities and finally the horizons are extracted [51]. The algorithm is mostly based on image processing techniques. Note that the complex processing chain can only be executed in this order and only the three mentioned structures can be found using this technique. One drawback of the method is, that the features cannot always be detected. For example faults and unconformities can only be found if they occur in a specific direction [51]. The limitation of the approach to these specific structures imply, that this approach is not generic. Yet it is useful, when applied during the interpretation of these features. There are other related approaches working in a similar way, but are specialized on different structures. Examples are the automatic detection of fault surfaces [9] or horizons [36].

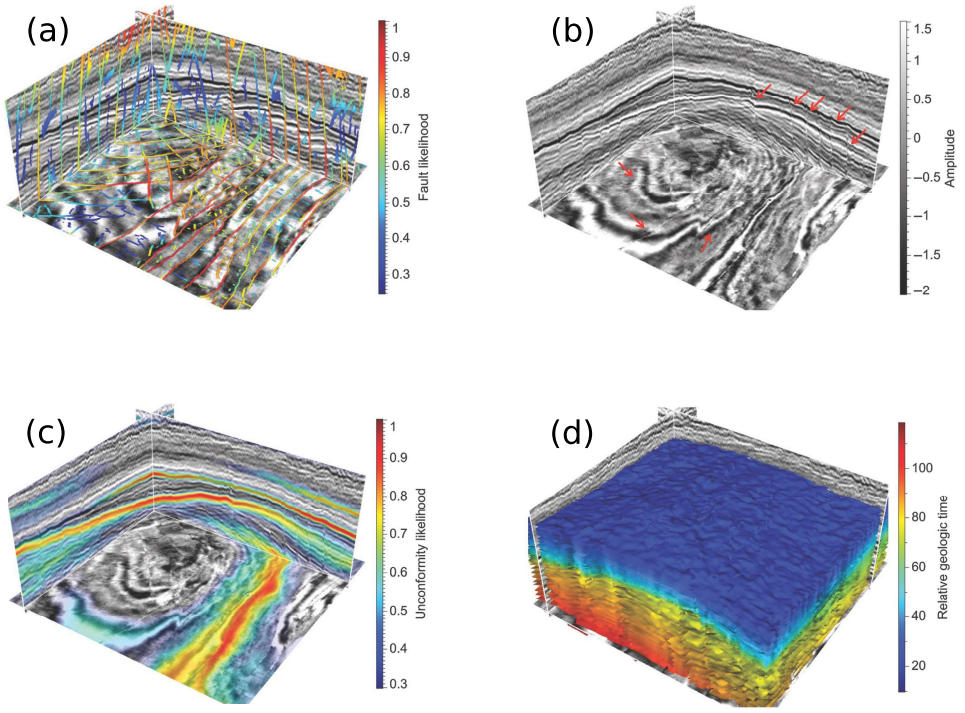


Figure 4.2: Processing chain to detect faults (a) and reverse them (b), find unconformities (c) and model the horizons (d) [51]

4.2 Comparison of Histograms with Dissimilarity Measures

As the approach of this thesis is based on comparing local histograms of different regions, the type of comparison has to be defined. This section introduces the basic principle of calculating the similarity of two histograms and gives an overview of their usage in different scientific fields.

4.2.1 Theoretical Background

Given two histograms, it can be useful to compare their similarity for different reasons. Several measures were proposed throughout the years and an overview of different options is given in this section. A detailed comprehensive survey of histogram dissimilarity measures can be found in [8]. The dissimilarity measures for the comparison of two histograms can be divided into two groups *bin-by-bin* and *cross-bin*. These are explained in this section and some representative examples are given. To ensure the consistency of the notation introduced in section 3.2, the histograms are denoted as H and K and the quantity of voxels in the bin with index i is given by h_i and k_i respectively. The variable n represents the total number of bins.

Bin-by-bin Dissimilarity Measures

The 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. It is important to note, that these measures have the characteristic to differ from perceptual similarity. One specific example can be seen in figure 4.3, where the L_1 distance (comp. eq. (4.1)) is calculated between both of the left and right distributions respectively. The left comparison results in a higher dissimilarity than the one on the right. In contrast the visual perception is that histograms H_2 and K_2 are less similar.

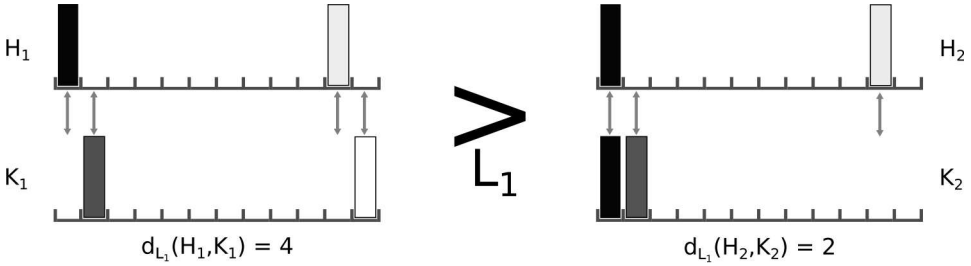


Figure 4.3: Divergence between bin-by-bin dissimilarity and perceptional similarity [39]

The first example for this group is the *Minkowski-form distance* d_{L_r}, which is calculated with

$$d_{L_r}(H, K) = \sqrt[r]{\left(\sum_{i=0}^{n-1} |h_i - k_i|^r\right)}. \tag{4.1}$$

The value of r has to be specified with respect to the type of application. The L_1 distance was often utilized for the comparison of color images [39].

Another prominent example is the *Chi-Square Distance* d_{χ^2}. It can be calculated with

$$d_{\chi^2}(H, K) = \sum_{i=0}^{n-1} \frac{(h_i - m_i)^2}{m_i} \tag{4.2}$$

and

$$m_i = \frac{h_i + k_i}{2}. \tag{4.3}$$

Having its origin in the domain of statistics, this distance measures how unlikely it is that one distribution was drawn from the population represented by the other [39].

Another important representative is the *Jeffrey divergence* d_J, which is not sensitive to the binning of the histogram and is robust with respect to noise. It is defined as

$$d_J(H,K) = \sum_{i=0}^{n-1} \left(h_i \log \frac{h_i}{m_i} + k_i \log \frac{k_i}{m_i} \right), \tag{4.4}$$

where m_i is again calculated with

$$m_i = \frac{h_i + k_i}{2}. \tag{4.5}$$

Cross-bin Dissimilarity Measures

In contrary to the first group, the cross-bin dissimilarity measures also contain terms that compare non-corresponding bins. The results of these dissimilarity measures are perceptually more meaningful [39]. One example is the *Match distance* d_M. It is calculated with

$$d_M(H,K) = \sum_{i=0}^{n-1} |\hat{h}_i - \hat{k}_i|, \tag{4.6}$$

where $\hat{h}_i$ and $\hat{k}_i$ are the frequencies of the bins in the cumulative histogram, in which every bin contains the sum of all bins in the original histogram with a smaller or equal index. More precisely, the frequency $\hat{h}_i$ at the bin with index i, is calculated with

$$\hat{h}_i = \sum_{j=0}^{i} h_j. \tag{4.7}$$

Through incorporating the cumulative histogram $\hat{H}$ into the calculation, the cross-bin characteristic can be ensured since every bin contains information on multiple bins of the original histogram.

Another example for the group of cross-bin dissimilarity measures is the *Kolmogorov-Smirnov distance* d_{KS}, which is calculated with

$$d_{KS}(H,K) = \max(|\hat{h}_i - \hat{k}_i|). \tag{4.8}$$

The resulting dissimilarity is the maximum difference between two corresponding bins $\hat{h}_i$ and $\hat{k}_i$, found in the cumulative histograms $\hat{H}$ and $\hat{K}$ [39].

4.2.2 Feature Detection with Dissimilarity Measures

The idea of detecting certain features by utilizing the dissimilarity measures is not new and was used in image processing tasks before. Some of these approaches and their results are presented in this section.

The work by Rubner et al. [39], which also introduces a new distance metric called the *Earth Mover's Distance*, is a good example for the task of content-based image retrieval. The objective is, given an input image, to detect images with similar content in a database. Therefor a distance measure is calculated between the histogram of the input and histograms of the different images in the database. Comparisons returning a small value are assumed to contain a similar object. The performance of multiple dissimilarity measures on the given task can be seen in figure 4.4. One can see the eight most similar images to the input image of a red car shown on the left. These were selected through resulting in the lowest dissimilarity values, when compared to the input image with the respective metric. The dissimilarity measures have indeed detected a couple of other pictures containing a red car. However, some false detections with a similar color distribution were selected by the algorithm. For example the pictures of the leafs contain a lot of yellow and green colors, that can also be found in the background of the input image.

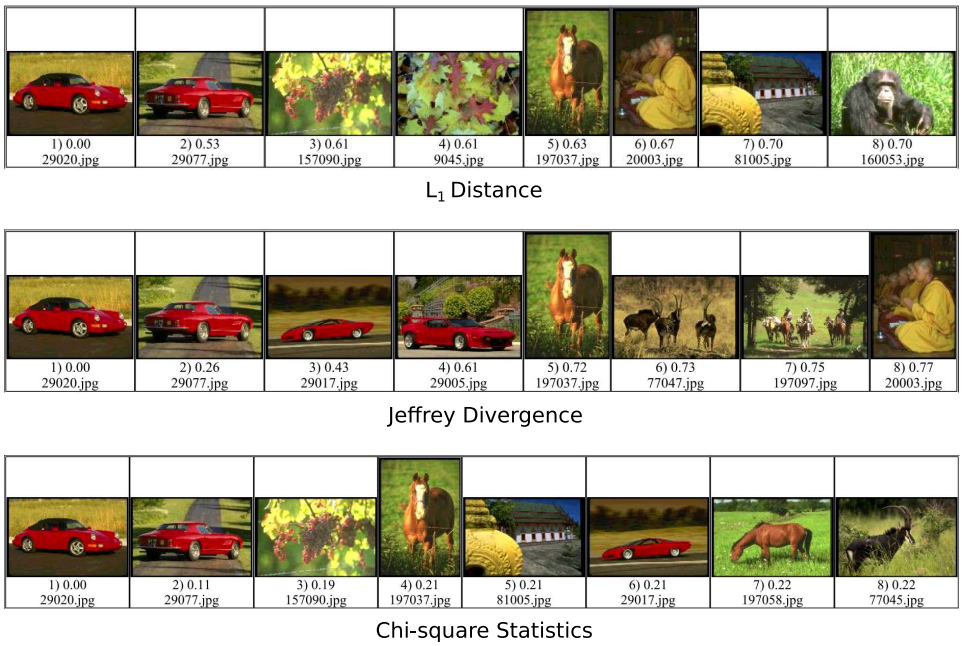


Figure 4.4: The eight most similar images to the leftmost image of a red car, detected with different dissimilarity measures [39]

While this comparison uses dissimilarity measures to find images with similar content, another approach is to search for a high distance between two histograms in order to detect divergences. This idea was utilized in [33] to detect natural image boundaries. The idea is to estimate the probability of the presence of a boundary per pixel. Therefor a local region around each pixel is divided into two halves and the local histograms of these parts are compared using the χ^2 distance (comp. eq. (4.2)). In case of a boundary the textures on both sides are different. Hence a high dissimilarity value indicates the presence of an edge [33]. Results of the boundary detection can

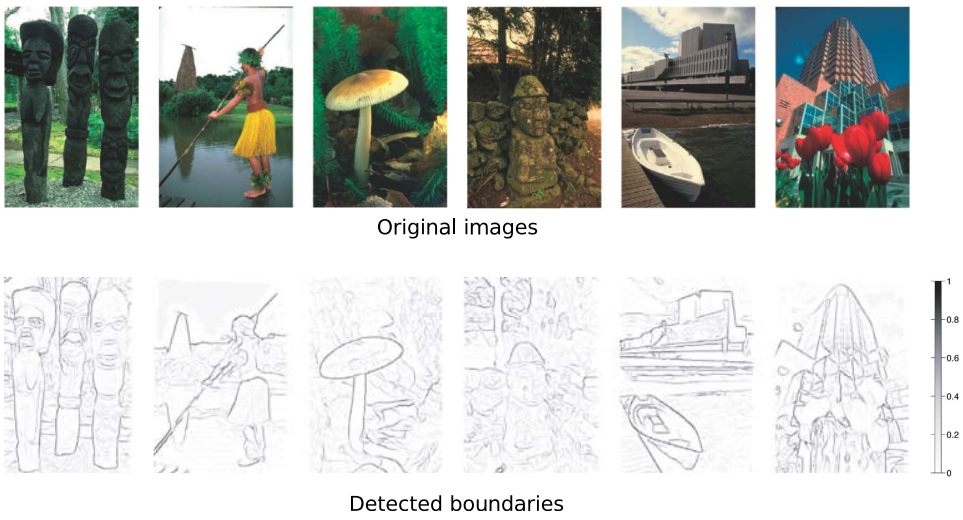


Figure 4.5: Automatic detection of image boundaries using local histograms [33]

be found in figure 4.5. There the top row presents the original images and the bottom row shows the result of the boundary detection. The intensity of the lines corresponds to the estimated probability of the presence of a boundary. Darker values imply a higher chance of having detected a border.

Another approach of searching for outliers in the data was presented by Asha et. al in [3]. Their aim is to detect defects on periodically patterned textures for the application in the automatic quality control of the fabric production. An image taken from the monitored fabric is split into several periodic blocks of equal size. Then the χ^2 distance (comp. eq. (4.2)) of each block's histogram to all other periodic blocks is calculated. The result is a matrix containing the absolute dissimilarity for each of the blocks. Then a hierarchical clustering algorithm is used to extract defective and defect-free clusters. The results of the algorithm can be seen in figure 4.6. On the left, the defect input image can be seen. In the center the detected defects are visualized. The rightmost picture overlays the original image with the defects.

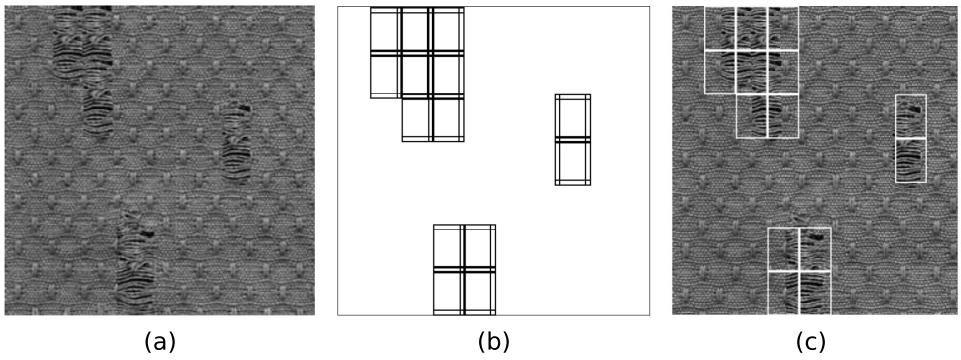


Figure 4.6: Detection of defects in a fabric image [4]

One drawback of the approach is, that the number of defective units must always be less than the number of defect-free blocks. Otherwise the algorithm cannot detect, which of the blocks belong to which category. Furthermore, the test images have to be taken exactly perpendicular to the surface of the fabric. The advantage is however, that quality control in the textile industry normally requires high labor costs and skilled inspectors. These costs can be saved and inspection time can be reduced by using an automated system. Finally the approach does not require a training sequence with defect-free samples, as it is often the case for techniques based on machine learning.

As presented in the previous examples, dissimilarity measures were already used in the domain of image processing and feature detection. However, they were barely used in context of volume data. One approach, proposed by Karimov et al. in [25], uses dissimilarity measures on the result of an automatic segmentation of the volume dataset, to detect potential defects. For each of the segmented objects, one region is derived from structural information, extracted from the results of the initial segmentation. Such a region is identified by using the *skeleton* of the object. This skeleton can for example be generated by iteratively removing surface voxels under geometrical and topological constraints until only the basic structure of the dataset remains. It is used to construct influence zones, for which local histograms are calculated. Then the L_1 dissimilarity measure (comp. eq. (4.1)) is calculated between individual regions to detect those, which have a potential of belonging to a segmentation defect.

The approach reduces the interaction effort, as a user only has to approve the essential corrections, suggested by the algorithm in multiple steps. Otherwise the whole correction would have to be conducted manually [25]. An example can be seen in figure 4.7, which shows the input dataset (a), the results of the initial segmentation (b), one of the suggested correction steps (c) and the corrected segmentation (d).

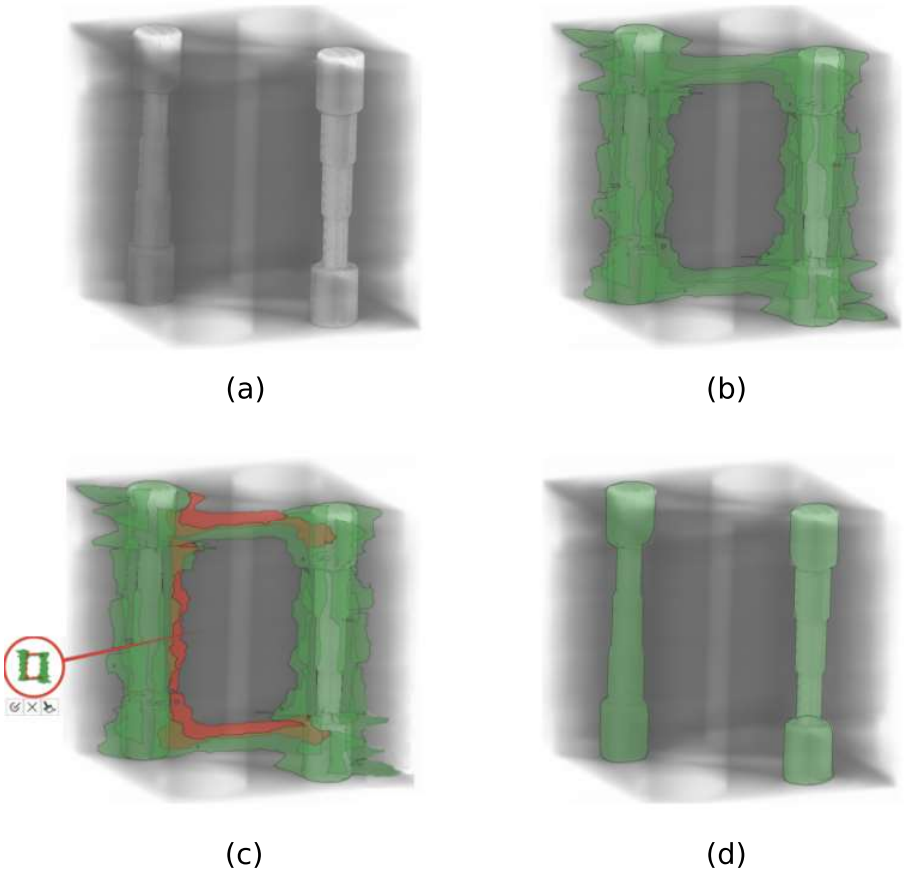


Figure 4.7: Elimination of defects in automatic segmentation [25]

The second approach using dissimilarity measures in the domain of volume data, deals with the segmentation and visualization of electron microscopy data [22]. Cell membranes in the dataset are segmented by using an active ribbon approach, which has to be initialized by a user through painting a rough approximation of the boundary on one of the slices. Combining these 2D segmentations, a centerline of a neural pathway can be traced through the volume [22]. To support the process of initializing the segmentation, the application provides a volume rendering of the dataset. This way the data can be inspected and regions of interest can be identified. The visualization is improved with a special edge enhancement. Using a dissimilarity measure on local histograms, the opacity of the rendering can be adapted to enhance boundaries and to suppress more homogeneous regions. This supports the navigation through the unsegmented dataset and the search for an initialization point of the segmentation [22]. An example of the edge enhancement is displayed in figure 4.8. The top row shows the original dataset as a fully

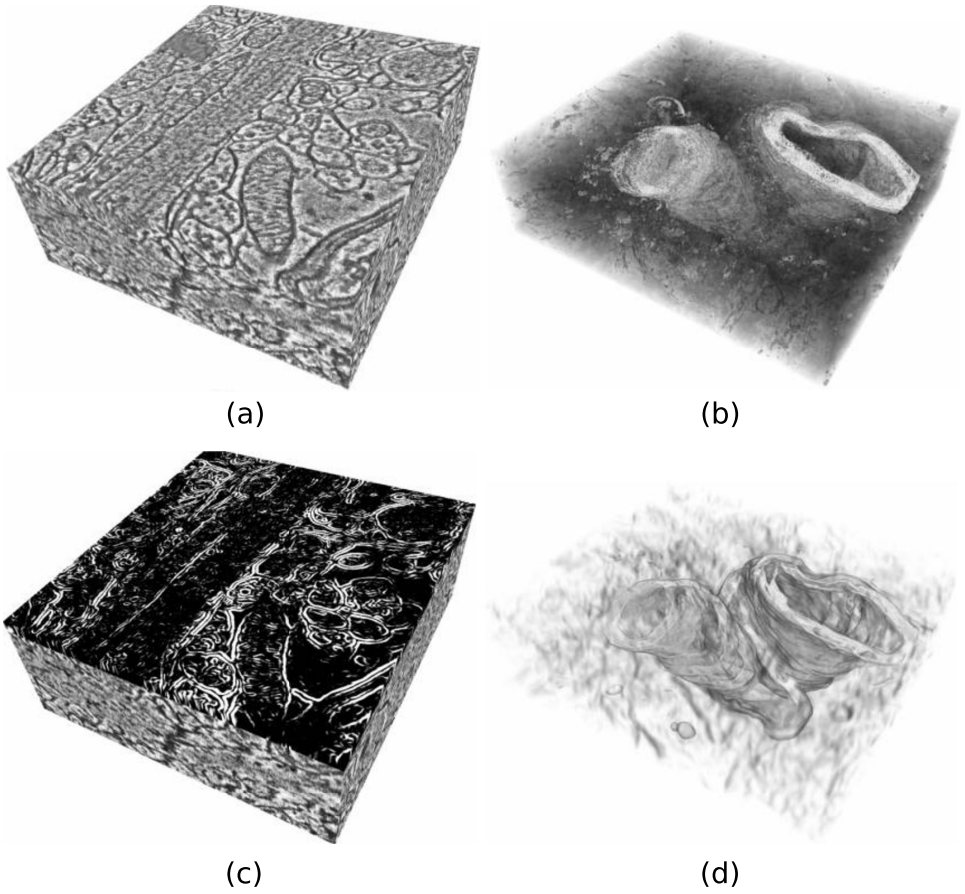


Figure 4.8: Edge enhancement in volume data based on local histograms [22]

opaque block in (a) and with transparency in (b), while the bottom row depicts the strengthened edges of the block in (c) and with transparency in (d). The edge detection algorithm is based on the 2D approach presented in [33], which is introduced earlier in this section. For every voxel a probability of containing a boundary is calculated [21]. A block neighborhood around each voxel is used to calculate the brightness gradient for different directions. The size of the neighborhood can be adjusted to match the resolution of the input data. Then it is separated along the given direction into two halves and a local histogram is calculated for each of the half-spaces. Finally the difference between these two histograms is computed using the χ^2 dissimilarity measure (comp. eq. (4.2)) [22]. If the histograms are dissimilar an abrupt change in the brightness of the volume is detected, indicating the presence of a boundary. If the histograms are similar, both half spaces have no edge between them. Finally for each voxel the maximum difference of all boundary directions is stored and used as the boundary value. During raycasting this value is used to modulate the opacity and color of the given

sample. The advantage of this method of edge detection is, that it works well with noisy data and arbitrary neighborhood sizes. It allows to highlight the boundaries of the axons, which are the target of the segmentation and have to be easily identifiable to set a starting point for the segmentation. While this is normally a time-intensive process, the approach works in semi-automatic fashion. The user simply has to initialize the automatic segmentation. The dissimilarity measures are used to assist the selection of an appropriate starting point by enhancing the edges of different regions.

As described in this chapter, the comparison of (local) histograms with dissimilarity measures were not evaluated in the context of feature detection in volume data. Further it was shown, that a similar approach is able to detect anomalies and important features in the domain of image processing. Thus utilizing a similar technique to detect regions of interest in volume data is a promising approach. The conception of a corresponding algorithm is presented in the following chapter.

5 | Detection of Anomalies

As described before, the aim of this thesis is to develop an algorithm that detects interesting regions automatically and presents them to a user in combination with diverse supporting tools. This chapter lists the requirements and conception of the algorithm and explains the individual steps in detail.

5.1 Conception and Requirements

A couple of requirements for the algorithm have been defined in collaboration with the members of the VRGeo Consortium [11]. At first the speed of the execution is of major importance. The algorithm can only be efficient in providing a first overview, if it can be executed in less than a couple of minutes. Otherwise the benefit of saved time would be reduced by a long runtime. Another major point is the reliability of the detection. Important regions must be detected reliably and entirely. Additionally the contained anomalies have to be of importance for the interpretation task. Furthermore the detection should be feature-independent and include all different kinds of relevant structures. Finally the results of the algorithm have to be reproducible.

The next step is to decide on the type of feature detection. During interpretation the geophysicist is looking for an anomaly in relation to the surroundings [50]. Therefore it might be useful to detect these anomalies by comparing regions in the dataset to their respective neighborhood. The anomalies manifest themselves through a local variation of parameters, relative to some background value [26]. These parameters are for example the different seismic attributes. Hence the local variation might be noticeable in the distribution of data values and thus in the local histogram. As a consequence, the local histogram of each region in the dataset should be compared to the local histograms of the neighborhood, which is the foundation of the developed algorithm.

In section 4.2.1 the basics of dissimilarity measures are explained, which calculate the difference between two histograms. These dissimilarity measures can be used to compare the local histograms in a neighborhood. Different dissimilarity measures are possible and the one to use should be selectable by a user. As it is discussed in section 2.3, different attributes have different strengths and weaknesses. The combination of two attributes could increase the chance of detecting important features. Thus the algorithm should also be able to compare local 2D histograms. Either one or two seismic attributes should be selectable as input for the algorithm. In theory a

comparison of even higher dimensional histograms would be possible, but is out of the scope of this thesis and could be a subject for further research once the approach has proven to be successful.

The next step is to define the type of neighborhood. 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.

In order to increase the chances of comparing the anomalous brick only to surrounding bricks not containing the anomaly, multiple kinds of neighborhood can be defined. The first one takes all possible directions into account and is called the *26 Neighborhood*. Another option is to restrict the comparison to certain directions. For example the *horizontal neighborhood* compares the current brick only to adjacent bricks, which are found in the horizontal x and y directions. Similarly the *vertical neighborhood* uses only bricks in vertical direction t for the comparison. The different types can also be seen in figure 5.1, where the colored brick is compared to the neighborhood. Accordingly the black and white bricks are utilized as neighbors, while the ones not used in the comparison are hidden.

Figure 5.1: Different neighborhood types

It is likely that the type of neighborhood has a huge impact on the results. Different neighborhoods might be capable of detecting different features, depending on the structural spreading. For instance the vertical neighborhood might have a better chance detecting seismic horizons, as they spread in horizontal direction and the values vary in the vertical direction. On the

contrary the horizontal neighborhood might not be as adequate, as there is not a big change of values to be expected in horizontal direction. Thus the type of neighborhood should also be selected by a user.

The neighborhoods explained above are the ones, that are implemented in the application and can be used for the algorithm. These neighborhoods were selected, as they have approved in context of image processing with seismic data. For instance the horizontal neighborhood was used to identify fault surfaces [9] and horizons were detected by using the vertical neighborhood [36].

With the definition of distance measure and neighborhood, the local histogram of every brick 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 algorithm dividable in three sequential steps, which are:

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.

An overview of the algorithm, including the required user input, can be seen in figure 5.2. The three individual steps shown in gray boxes in the figure, are explained in detail in the next sections. The algorithm was presented during the meetings of the VRGeo Consortium multiple times and was evaluated and refined in close collaboration with the delegates of the consortium.

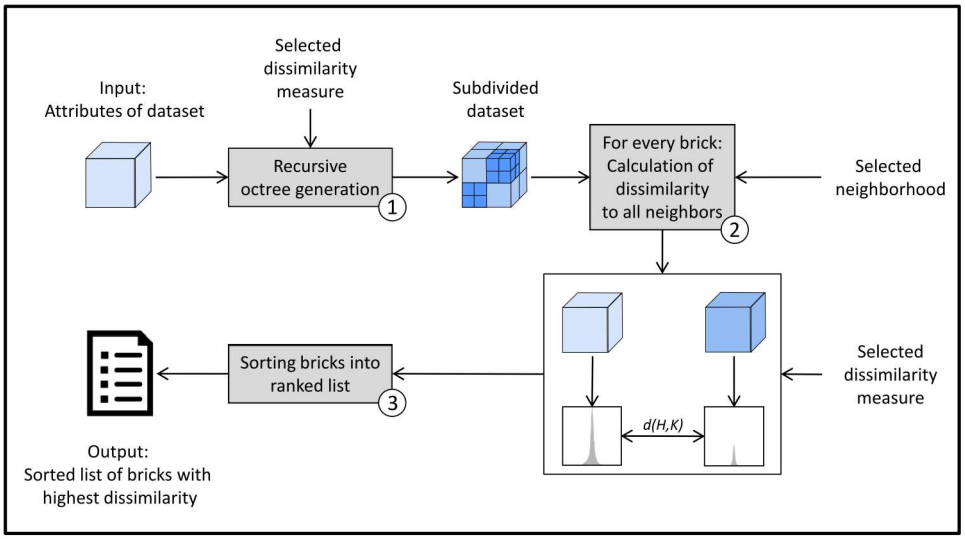


Figure 5.2: Overview of the proposed algorithm

5.2 Recursive Octree Generation

In order to calculate a dissimilarity value to the surroundings for every region, the size of these has to be defined. Thereby the brick size is adapted dynamically to the present data. This has multiple advantages, especially 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 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 manor. Therefor a recursive subdivision of the whole dataset can be done, until the grid fits the local data values optimally. 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. Further subdivision would lead to the division of the anomaly into multiple bricks and should be avoided. The optimal case would be a brick consisting mainly of the anomaly, which is not present in the adjacent bricks. 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 within figure 5.3.

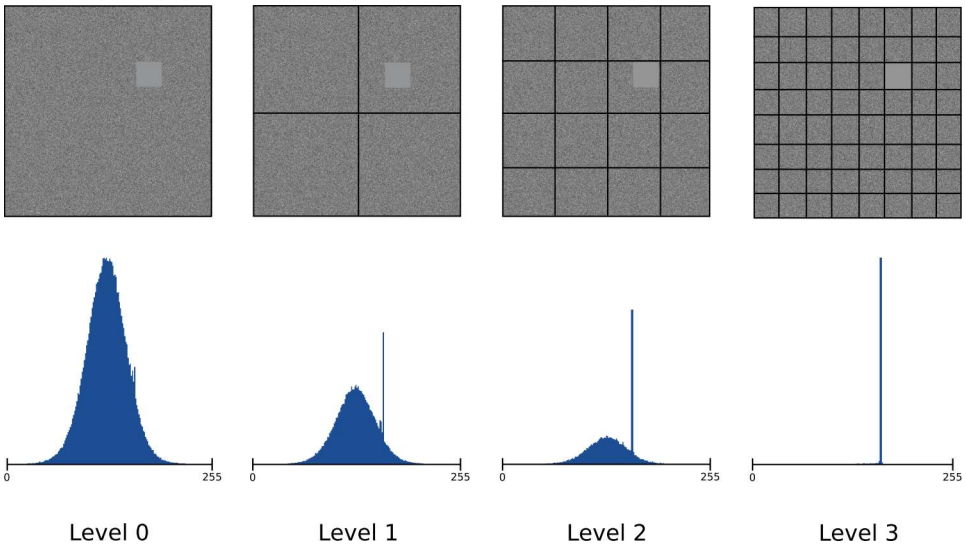


Figure 5.3: Simple example for the isolation of an anomaly through recursive subdivision

For the sake of simplicity the volume dataset is shown from the top, so the third dimension is not visible in the illustration. 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 belong always 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 in four bricks, which reduces the irregularity on a local scale.

A suitable method for the dynamic generation of the bricks is to start with the whole dataset and subdivide it recursively. The so-called *octree*, which is a hierarchical data structure based on the decomposition of the underlying space, is the optimal data structure for the bricks. Recursive subdivision divides the dataset into eight sub-volumes. Every node in the octree has either eight non-overlapping children, or is a leaf node [49]. The subdivision is initialized at the root node of the octree and thus with the whole dataset and the global histogram. The procedure of the subdivision is as follows. The size of the brick to be partitioned is divided by two in x,y and the vertical direction t. This results in eight equally sized sub-bricks, which become the eight children of the current node. For each of these sub-bricks the local histogram covering $\frac{1}{8}$th of the voxels of the parent brick is calculated. Then the user-defined distance measure is evaluated between the local histogram of the sub-brick and the global histogram of the root node. Note that these can either be 1D or 2D histograms, depending on the chosen attributes. The resulting distance is saved in the respective child node. Afterwards the eight children are used as input for the subdivision. Again for every one of these, eight sub-bricks are created and the local histogram and distance to the global histogram are calculated and stored. This procedure is repeated recursively, until a user-defined maximum of octree levels is reached. This value should on one hand be high enough to allow anomalies to occupy a huge part of the bricks, and on the other hand be small enough so that the resulting local histograms are meaningful. Additionally the runtime of the algorithm is highly depending on the number of octree levels. The amount of nodes in the octree grows exponentially with every further level. Thus limited hardware storage and processing power might restrict the possible number of octree subdivisions.

After this step the octree has been recursively subdivided, until the user-defined maximum depth is reached. At this point, every node stores the dissimilarity value of the comparison between its local histogram and the

global histogram. Yet all of the bricks are still of equal size. In order to reduce the brick size in the homogeneous parts, some paths of the octree are merged again. The calculated dissimilarity to the global histogram is used as an indicator. 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. But there is a chance of having detected a local maximum, while further subdivisions would reveal a level with an even bigger dissimilarity. The recursive subdivision of the simple example, provided within figure 5.3, can be seen before and after the merging-step in figure 5.4. One can easily see, that the dataset is only subdivided around the anomalous part.

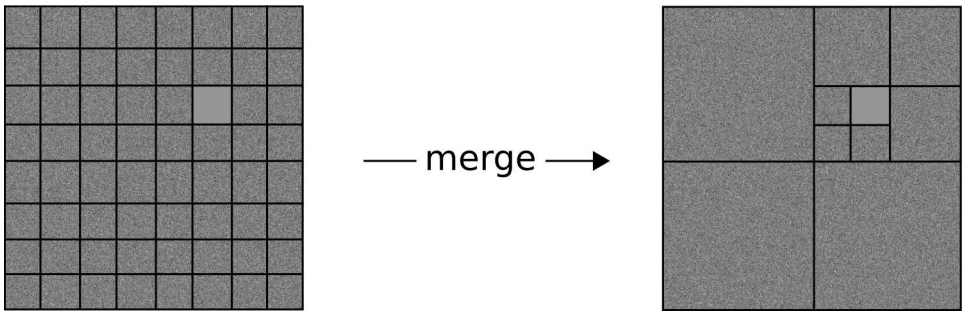


Figure 5.4: Merging process of the recursively generated octree

Note that the octree subdivision must not be carried out by using a normal histogram. Instead relative frequencies have to be used, since the compared histograms do not cover equally sized regions of the dataset. The relative frequencies can be calculated through dividing every frequency in the histogram by the number of voxels inside the brick. More precisely, the frequency $\tilde{h}_i$ of the relative Histogram $\tilde{H}$ at index i is calculated with

$$\tilde{h}_i = \frac{h_i}{x_l \cdot y_l \cdot t_l}, \tag{5.1}$$

whereby x_l, y_l and t_l are the dimensions of the brick, which the local histogram covers.

After the whole octree has been traversed and every path has merged all children below the global maximum level, the dataset is covered with bricks of different sizes. The next step is to figure out, which bricks differ from the adjacent bricks. This procedure is explained in the next section.

5.3 Calculation of Dissimilarities

In the second step, for each of the differently sized bricks in the octree, a value of dissimilarity 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.

Usually the neighborhoods are defined for exactly one adjacent brick in every direction, which holds true for a uniformly sized grid. In the described case, the octree is dynamically generated and the bricks do not have the same size. Therefore the number of adjacent bricks in one direction can vary, depending on the level of subdivision. Hence the neighborhood types have to be adapted in order to fit the dynamic grid size of the recursive octree. The idea is to utilize every brick, which is directly adjacent to the current brick. If the level of octree subdivision in that direction is equal or lower than the level of the current brick, only one directly adjacent neighbor is present. The dimensions of this brick might be bigger than the current one due to less subdivisions in that region. However this does not affect the procedure of the comparison in any way. Otherwise, if the octree in the target direction has a higher level of subdivision, the number of neighbors in that direction is $4^{(L_n - L_c)}$, where L_n is the level of subdivision of the neighbor and L_c is the level of subdivision of the current brick.

The selected neighborhood type can be used to calculate the dissimilarity of every brick. Therefor the chosen distance measure is applied to the local histograms of the current brick and each neighbor individually. The total dissimilarity d of the current brick c calculates with

$$d_c = \frac{1}{n} \left(\sum_{i=0}^{n-1} d(H_c, H_i) \right), \tag{5.2}$$

where n is the number of valid neighbors. Note that the comparison is independent from the amount of adjacent blocks in the neighborhood, as the dissimilarity to each brick is calculated individually and averaged afterwards.

In addition to the total dissimilarity values of the bricks, some information is required for the visualization and needs to be stored during the execution of the algorithm for later use. The usage of this data is described together with the visualization in detail in chapter 6. One of these features is a glyph, that shows the direction of most dissimilarity. To determine this direction, the center points of all neighbors are weighted. This results in one final position in the dataset. The weights are determined by the amount to which the single dissimilarity contributes to the sum of all dissimilarities. Using the averaged point of dissimilarity, a vector from the center of the current

brick to this point of high dissimilarity can be calculated. This vector can be visualized with a glyph, as it is described in section 6.2.

Further information used to enhance the visualization, is the dissimilarity of each bin of the local histogram to the bins of the neighboring histograms. In other terms, the bin dissimilarity describes for every bin, by which portion it contributes to the total dissimilarity of the brick. These bin dissimilarities can be visualized in the local histogram to provide additional information. Furthermore these values can be used to create an automatic transfer function. More details on both features can be found in section 6.3. The values are calculated during the comparison of a brick to the neighbors. In case of the bin-by-bin dissimilarities, one value of dissimilarity is calculated for each pair of bins. Then these values are summed up, which results in the final dissimilarity between the two histograms. During this process the interim results of each of the bins can be stored, prior to calculating their sum. As every comparison between the current brick and one of the adjacent bricks produces one dissimilarity value for each bin, these are added up and normalized to result in the final bin dissimilarity.

Now that the dissimilarities have been calculated, the final step of the algorithm is to determine the most anomalous bricks.

5.4 Sorting of Results

After the previous step, one total dissimilarity value has been calculated for each of the bricks. Finally the task is to search for those with the highest values and to rank them accordingly. All leaf nodes of the octree correspond to bricks, that have been compared to their neighborhood. Therefor these are sorted into a ranked list by their total dissimilarity. The highest ranked brick of the list has the biggest value of dissimilarity, while the last entry of the list is not very different from the neighborhood. This ranked list is the final result of the algorithm and is used for the visualization of the results, as it is described in chapter 6.

5.5 Export and Import of Results

The ranked list of bricks can also be used to export the results of the algorithm. For each of the bricks the size, position and the information needed for the visualization, such as the direction of biggest dissimilarity and the bin dissimilarities, are known. All of these values can be exported and reimported again when restarting the software. That way a user can return to the interpretation at a later point in time, without having to re-run the algorithm.

The possibility to export the results is provided within the GUI and is enabled after the algorithm has finished. During the export, a special file format (*.safd*) is used to store the information needed to recreate the visualization in the software. Afterwards this file can be imported, to recreate the state of the software. This way the interpretation of the results can be continued.

5.6 Generation of a Seismic Anomaly Attribute

Additionally the results of the algorithm can be used to generate a new seismic attribute. This attribute can be understood as a kind of *Anomaly Attribute*. It is generated by mapping the rank of each brick to the value range of the data type. All voxels inside of one brick receive the same value, which allows to recognize the underlying brick structure in the attribute. The export is done by using the *SEG-Y* format [45], which is the open standard for seismic datasets defined by the Society of Exploration Geophysicists (SEG) in 1975. Besides the actual data, it contains a header with useful information, such as the dimensions of the dataset and the coordinate system that it is defined in.

While the export and import functionality described in the previous section allow to return to the results of the algorithm within the proposed application, the generated attribute can be imported in most seismic interpretation software packages. This way the results of the algorithm can be reused in these, to support additional interpretation tasks. Most interpreters use a variety of software packages, as they all have their strengths and weaknesses and are suitable for specific tasks. By allowing the results to be imported in different software packages, the presented algorithm can be integrated into the seismic interpretation workflow more seamlessly. However, information such as the bin dissimilarities and dissimilarity vectors cannot be integrated into a SEG-Y attribute and are only available in the introduced application. Solely the pure dissimilarity values of the bricks can be encoded in the attribute. An impression of the process can be found in figure 5.5. The screenshot on top shows the global visualization, which is explained in the following chapter and is attained directly after the algorithm has concluded. The image below depicts the generated anomaly attribute, imported into OpendTect.

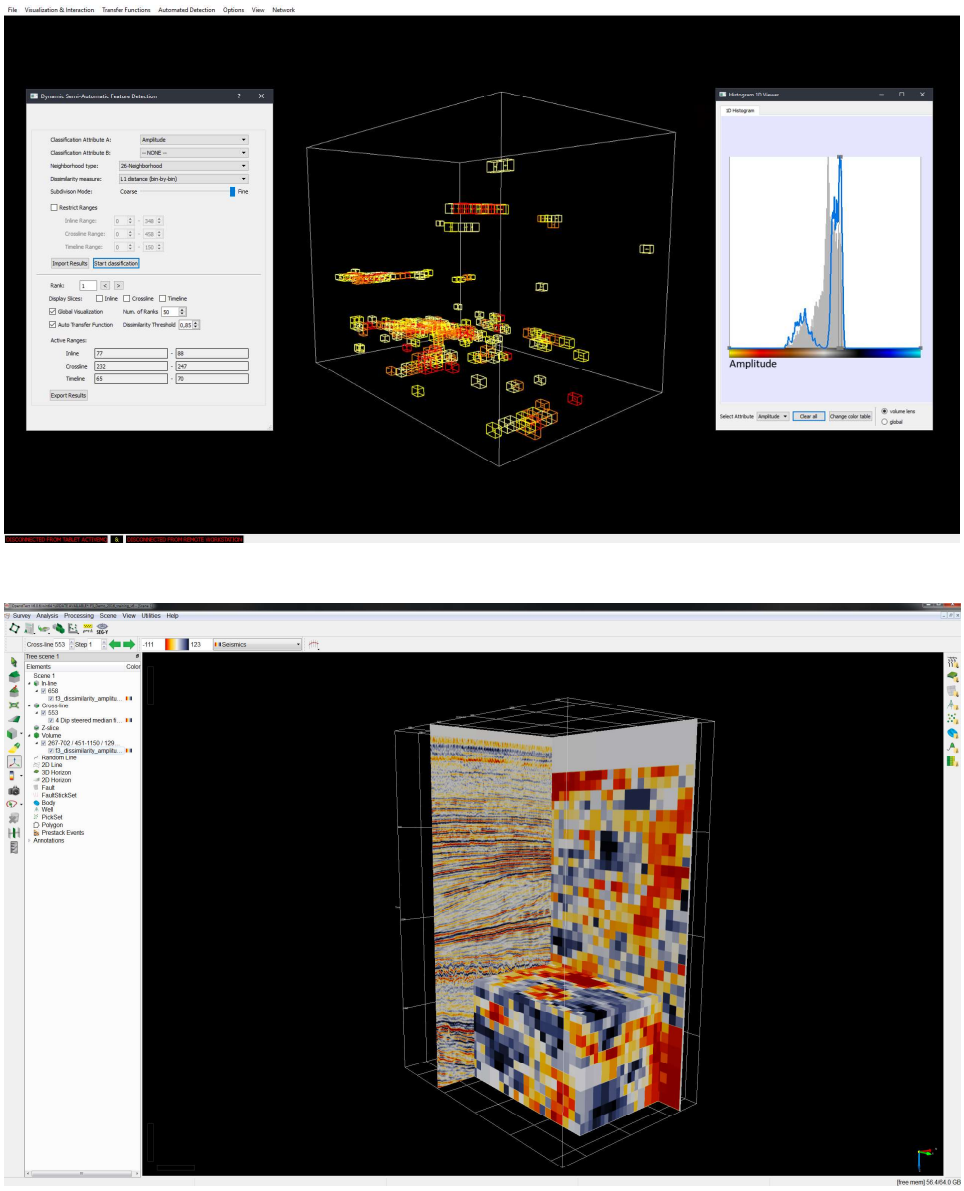


Figure 5.5: Comparison between the results of the algorithm in the proposed application and the generated anomaly attribute imported into OpendTect [16]

6 | Visualization of the Results

In this chapter the different elements of the visualization as well as the provided tools for supporting the interpretation are explained. These are based on the sorted list of ranks, which is created in the last step of the algorithm.

6.1 Overview of the Application

The algorithm and visualization are integrated into an in-house developed seismic interpretation software called the *VRGeo Demonstrator* [11], which is entirely implemented in C++. For the creation of a GUI the QT library [10] was used, which is based on C++ and supports the creation of user interfaces. To integrate the rendering of graphical OpenGL context, the OpenSceneGraph library [43] was integrated, which allows to create real-time graphics applications. An overview of the application can be seen in figure 6.1.

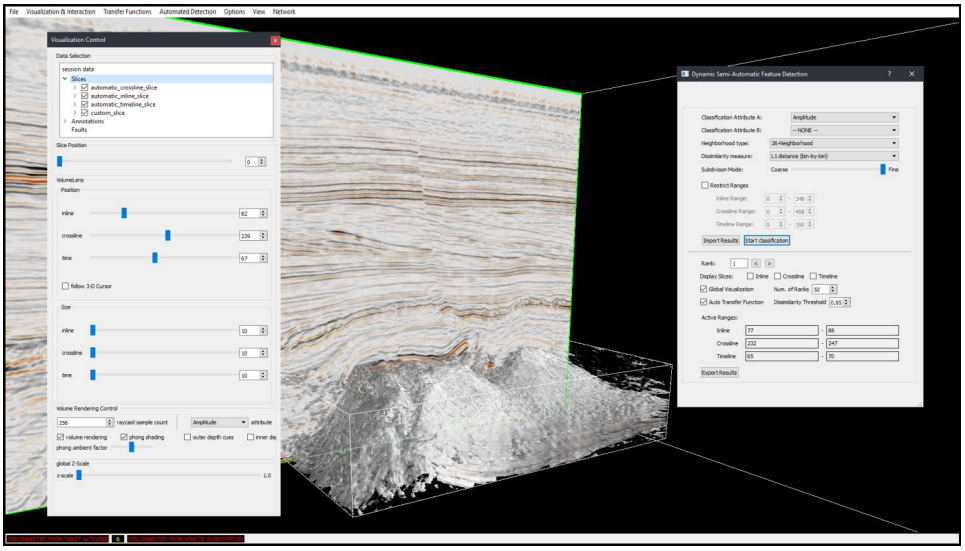


Figure 6.1: Overview of the developed application

When launching the software, a SEG-Y dataset can be selected and imported. As all of the available attributes of the dataset are stored in their own file, the software needs to import all of them one by one. After the import is completed, a user is presented with the *volume view*, which shows a 3D view of the dataset with volume rendering in the center, as it can be seen in figure 6.1. In accordance to the aims of this thesis, the developed algorithm should be executed first, to get an overview of different regions

in the dataset and to be provided with multiple starting points for interpretation. The respective GUI dialog for semi-automatic feature detection can be opened via the menu bar at the top. A screenshot of the user interface can be seen in figure 6.2. There the upper part can be used to define the

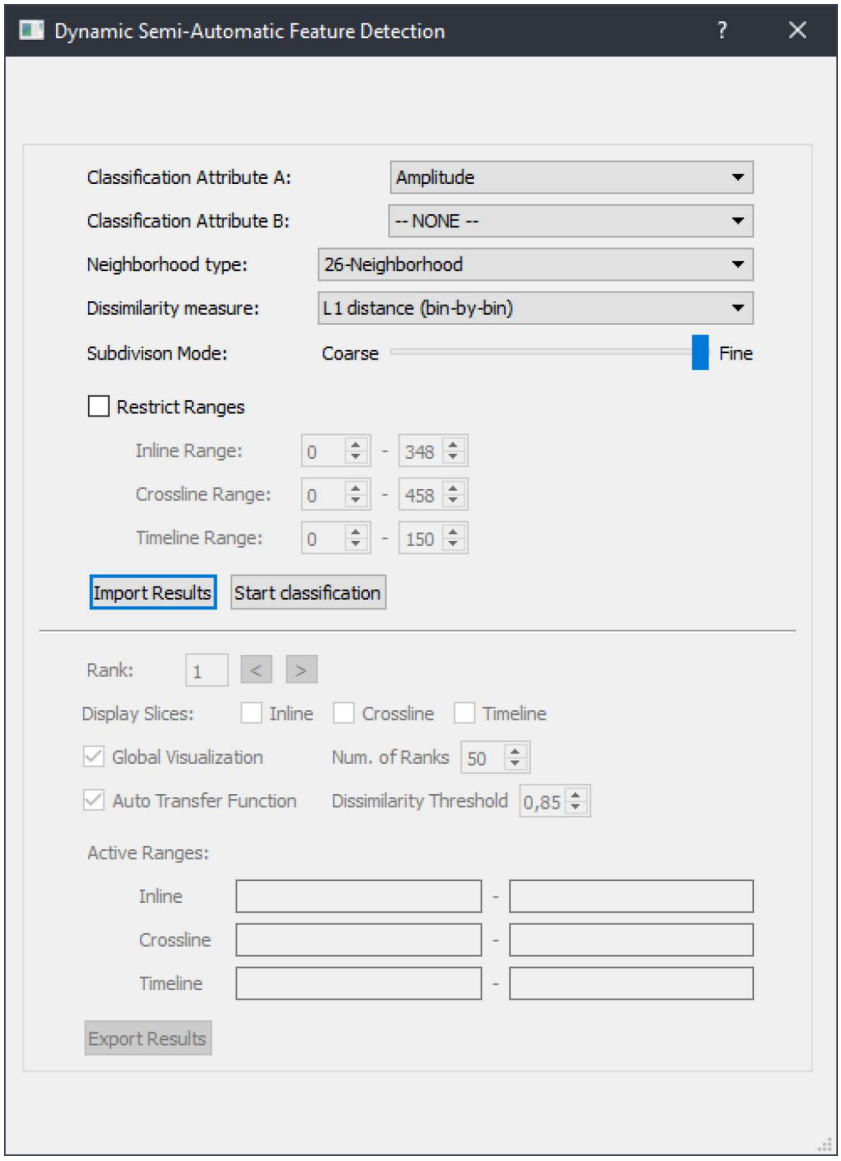


Figure 6.2: User interface of the semi-automatic feature detection

settings for the algorithm, as they are described in chapter 5. The lower part is disabled until the algorithm is executed and can be used afterwards to alter the visualization. The first settings for the algorithm are one or two attributes to use. These are selected with the dropdown-menus at the top. According to that, 1D or 2D histograms are compared. Then a user can

select either the 26, horizontal or vertical neighborhood. Additionally a distance measure can be chosen. The last input is the maximum level of octree subdivision, which can be set with the respective slider. The octree level increases the total amount of bricks, while simultaneously decreasing their size. A higher level of subdivision might lead to the detection of smaller anomalies, that are not found otherwise. However, the runtime of the algorithm grows exponentially with the level of the octree, which should be kept in mind. More information on the performance of the algorithm is given in section 7.1. Optionally the feature detection can be restricted to a certain part of the dataset. This can be done by using the corresponding spin-boxes. Finally the GUI offers the functionality for importing and exporting the results, as it is described in section 5.5. After the algorithm has been executed, the lower part of the GUI is enabled, as it is shown in figure 6.3. The single elements are explained in the following sections. As

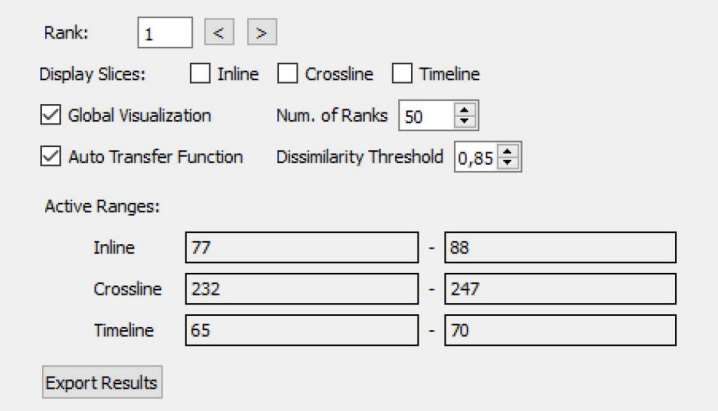


Figure 6.3: Lower part of the user interface after execution of the algorithm

the application has mainly two objectives, to provide a global overview of different structures and to propose starting points for the interpretation, two different types of visualization were implemented. The first one is a global overview, revealing clusters of important bricks in the dataset. The second one is specialized on presenting individual bricks as possible starting points. These two work independently from each other and can also be used in combination. Both visualizations are explained in the following sections.

6.2 Global Visualization

The aim of the global visualization is to provide a first overview of different important regions in the dataset. Thus it is presented directly after the algorithm has concluded. The GUI offers a spin box for the number of ranks (comp. fig. 6.3), which can be used to specify the amount of bricks to be displayed. These are taken from the top of the ranked list and rendered

directly into the volume view. 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. Generally 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 by using a color scheme similar to a heatmap, which is mapped dynamically to the number of visible bricks. Those with the highest dissimilarity are always shown in red. A faint yellow color is used for the last visible entries of the ranked list. The applied heatmap is shown in figure 6.5. It can be noticed that the color scheme is not continuous but consists of four different colors. According to studies on glyph visualization, this type of color scheme allows a clearer differentiation of the respective classes [38]. Possibly important regions in the dataset can easily be identified on the first gaze. An example of the global visualization can be found in figure 6.4.

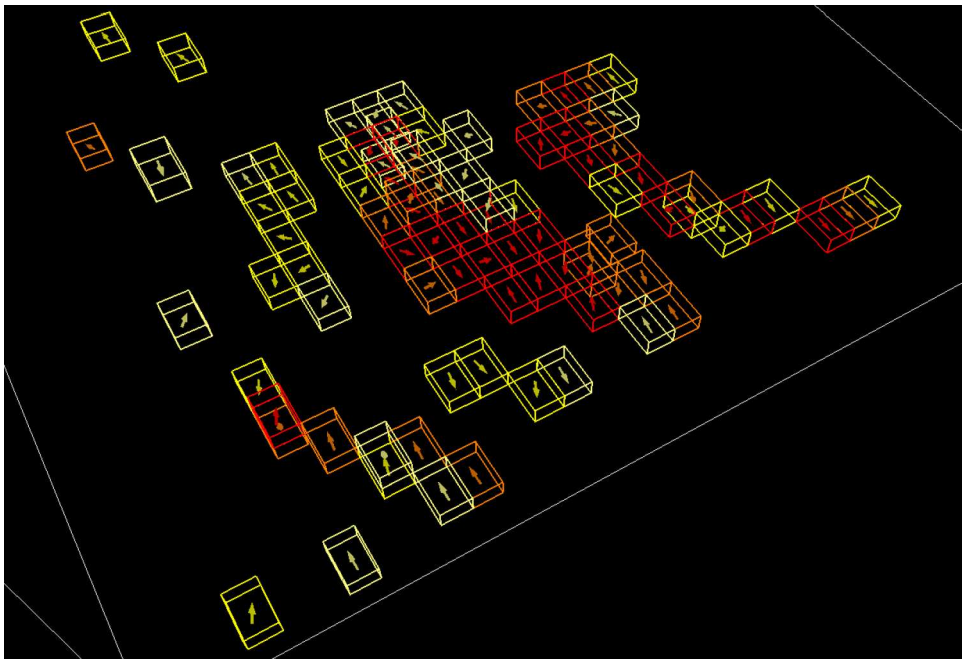


Figure 6.4: Global visualization

Figure 6.5: Heatmap used for the bricks in the global visualization

In order to increase the informational content of the bricks, the direction towards the region with the biggest dissimilarity is represented by a glyphm,

added to the center of each brick. The glyph type fitting the context of a direction best, is an arrow [38]. While the colors of the arrows could in theory be mapped to an additional property, this could also interfere with the colors of the bricks and increase the complexity of the visualization significantly. Instead the arrow receives the same color as the corresponding brick to prevent visual cluttering and to simplify the visualization. The arrow-glyphs can also be seen in figure 6.4.

The proposed workflow is to select a suitable amount of bricks to identify important clusters in the dataset. These can then be examined one by one, including the directions of the contained arrows. At any time slice and volume rendering can be used to inspect the present data values, to get a grasp of different features in this cluster. This way a user can get a basic insight into the dataset and the geological processes, that have caused the structures in this area. In the next step of the interpretation, the focus could be on the highest ranked bricks individually. The appropriate visualization is described in the next section.

6.3 Visualization of Individual Ranks

In addition to the global visualization, multiple tools for the inspection of individual ranks are provided. Note that both visualizations can be used simultaneously. This can for instance be useful to investigate the surroundings of the currently examined brick. The main part of this visualization is the active rank, which is displayed in the GUI (comp. fig. 6.3). To the right are arrows, that can be used to cycle through the ranks and set the focus on a different brick. Thereby the whole visualization is adapted to the current rank in real-time. The single elements are explained in the next sections.

6.3.1 Rendering of Slices

For each of the three possible dimensions, one slice can be added to the visualization. The slices are inserted in the middle of the active brick. As soon as the next rank is activated, they are automatically moved to the center of the next brick. This principle is further elaborated in figure 6.7.

At any given time a user can move the slices dynamically through the dataset with the *Visualization Control Widget*, which can be found in figure 6.6. This widget lists all of the inserted slices at the top. They can be selected and moved through the volume, by using the slider below. While the automatically added slices are named accordingly, an arbitrary number of slices can be inserted manually. This is to adapt the visualization to the specific needs of the interpreter. The slices use the attribute of the algorithm by

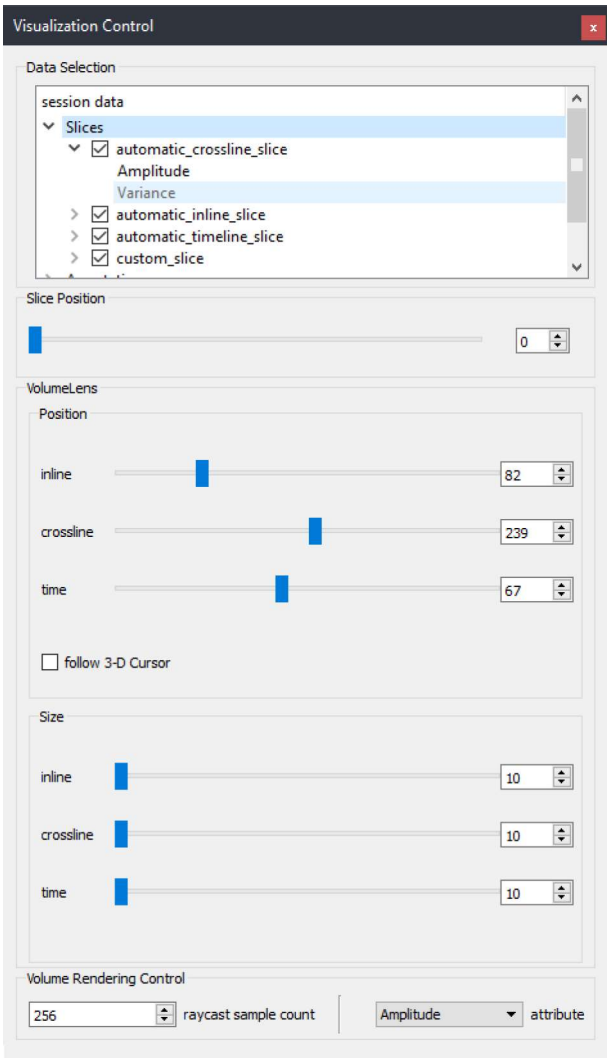


Figure 6.6: Visualization Control Widget

default, but can be adapted dynamically by changing the selected entry in the Visualization Control Widget.

6.3.2 Volume Rendering and Automatic Transfer Function

The part of the volume dataset inside of the active brick is visualized with 3D volume raycasting. By default, the rendering is restricted to the size of the brick, but can be enlarged and moved with the according sliders in the Visualization Control Widget (comp. fig. 6.6). This restricted area of raycasting is referred to as *volume lens* in the context of this thesis, as only the parts covered by it are rendered. This yields the advantage, that the performance intensive volume rendering is only executed in the current

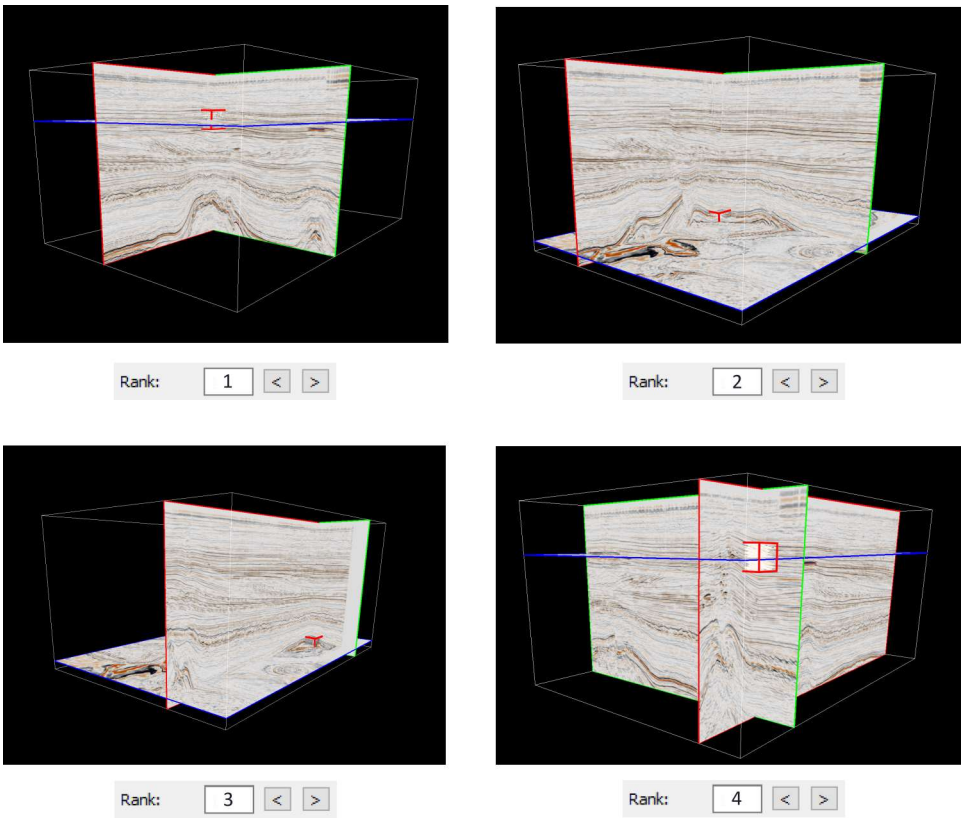


Figure 6.7: Slices of the F3 dataset [15] moving automatically to the active brick

region of interest. Similar to the 2D slices, the volume lens is automatically moved with the adaptation of the active rank. This can can also be seen in figure 6.7.

6.3.3 Histogram and Transfer Function Visualization

As further background information on the active brick can support the interpretation, an additional window displays the local histogram of the active region and the transfer function for the volume rendering. Two types of histogram view can be distinguished, depending on the number of attributes used for the algorithm. The following paragraphs differentiate the two variants.

One-Dimensional Histogram Widget

When only using one attribute for the comparisons of the algorithm, the histogram view is referred to as the *Histogram 1D Widget*. An example of it is shown in figure 6.8. In the center of this window the active brick's local

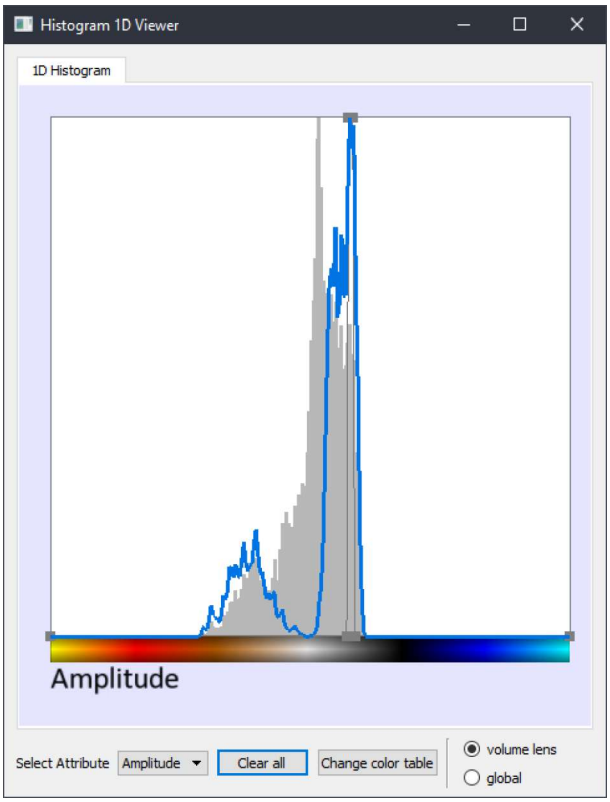


Figure 6.8: Histogram 1D Widget

histogram is displayed. All frequencies are normalized, so that the biggest value reaches the top. The local histogram is dynamically adapted to the size and position of the volume lens. When it is being moved or the size is changed, the local histogram is recalculated in real-time on the GPU, always using exactly the voxels on the inside. This way a user can observe how the value distribution in the nearby areas changes, by sliding the lens through the region.

As described in section 5.3, one normalized dissimilarity value is calculated for every bin of the histogram, describing how it differs from the bins with the same index in the neighborhood. These values are displayed on top of the local histogram in form of a blue curve, which is called the *anomaly curve*. It consists of one point for every bin of the histogram. The position of the respective points corresponds to the bin dissimilarity and the bin index. These points are then connected to result in a curve, which can also be seen in figure 6.8. The aim is to provide additional information on the dissimilarity of the values to support the interpretation. Through the maxima of this curve, the values most responsible for the ranking of the corresponding brick can be easily identified.

The third element of the Histogram 1D Widget is an automatically generated transfer function. Mostly a user might want to investigate just those parts of the active brick, which differ from their surroundings. Hence the calculated bin dissimilarities can be used to generate a transfer function automatically, to emphasize the anomalies and to hide other regions. Therefor the bin dissimilarities have to be normalized to the range of zero and one. Then a threshold can be defined to specify how dissimilar the displayed bins have to be. This threshold is set to 0.85 by default, but can be adapted in the GUI to suit the needs of the interpretation. The transfer function is created by automatically adding control points to the values above the threshold. When the threshold is changed by a user, additional points are added or existing ones are removed. The dissimilarity value additionally serves as an opacity. Thus only the highest dissimilarity values are fully opaque, while others appear transparently. Bins below the threshold are not visible in the volume rendering at all. One example for the automatic transfer function can be found in figure 6.9, where the algorithm was used on the amplitude of the Parihaka dataset [42]. The included horizons are isolated automatically, by using the automatic transfer function. Only values belonging to the horizons are shown in the volume rendering. In addition, the corresponding Histogram 1D Widget can be seen to the right side of the volume. The local histogram, anomaly curve and automatic transfer function are

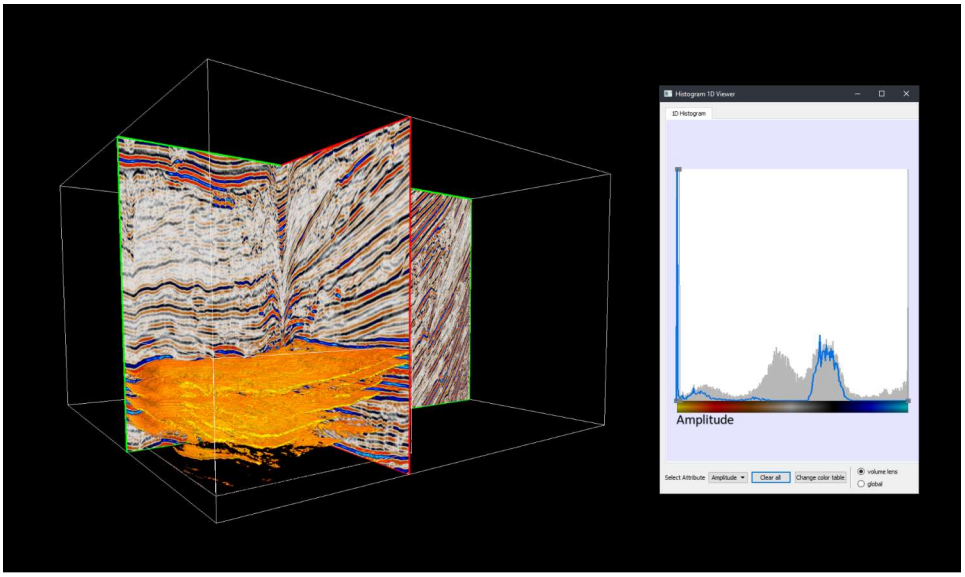


Figure 6.9: Automatic isolation of horizons in the Parihaka dataset [42]

adapted whenever the active rank changes. In addition to the automatically generated transfer function, the widget can be used as a standard editor to adapt the function to the needs of the interpretation. Existing points can be adapted or removed and new ones can be added.

Two-Dimensional Histogram Widget

The 2D version of the widget is similar to the 1D version and contains the same elements. Here the window is referred to as the *Histogram 2D Widget*. An example of it is shown in figure 6.10. The major difference is

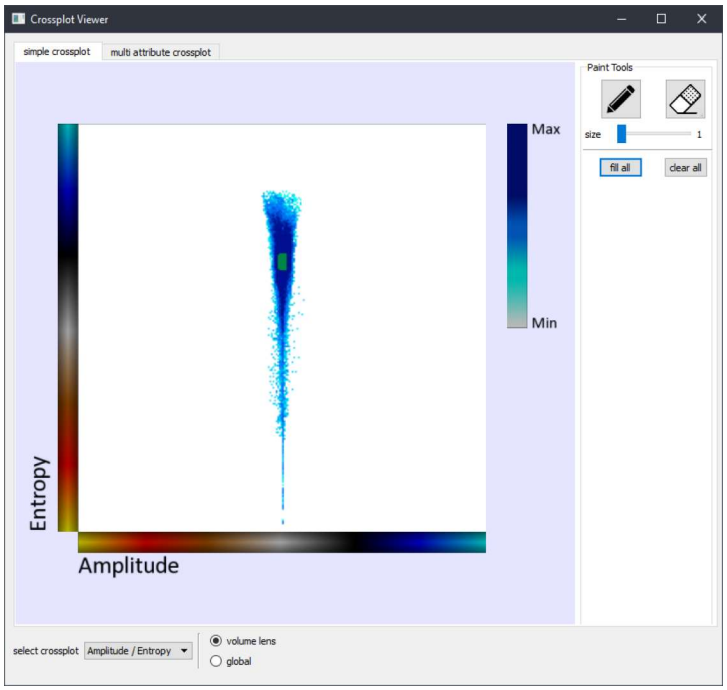


Figure 6.10: Histogram 2D Widget

that a two-dimensional histogram is displayed. As shown in section 3.3.2, integrating the frequencies into the 2D histogram can lead to the loss of non frequent values. The developed algorithm is specifically aiming towards finding non frequent anomalies, which might not be visible in a frequency mapped histogram. Hence it was decided to abandon the representation of the frequencies. Instead a binary mapping is used, which depicts every present value-combination. The 2D histogram is adapted, whenever the volume lens is being moved or its size is changed. Here again the recalculation of the histogram is performed on the GPU in real-time.

The Histogram 2D Widget also shows the bin dissimilarities, as described for the 1D version earlier. On the contrary to the one-dimensional case, the values cannot be represented by a curve due to the two dimensions of the histogram. Alternatively the colors of the visualization can be used for this matter. The most intuitive color scheme would again be a heatmap with red and yellow tones. However, these colors are already in use for the global visualization of the bricks. The color scheme should not be reused to repre-

sent the bin dissimilarities in the histogram, as these values have nothing in common and this might confuse a user. Instead a color mapping from light to dark blue can be used. Dark blue values represent the bins with highest dissimilarity and the light blue colors depict values, which have less variance in the neighborhood. Aside of the histogram, the color table is displayed for reference. The result of the color mapping can also be seen in figure 6.10.

The Histogram 2D Widget also provides an automatic transfer function and editor. Due to higher dimensionality the function cannot be generated with simple control points like in the one-dimensional case. Alternatively a painting metaphor can be utilized, which was introduced in [31]. The editor is based on painting and erasing values. Therefor the Histogram 2D Widget provides a brush with an adaptable size. It can be used to paint those values, that should be visible in the volume rendering. Furthermore a rubber tool can be used to remove previously painted values. The paint is represented by a green overlay color, as it can be seen in figure 6.10.

The method of the automatic transfer function generation is similar to the one-dimensional version. For every bin dissimilarity above the user-defined threshold, the corresponding value is painted automatically and is visible in the volume rendering. The painted spots are adapted, when the active rank or the threshold is changed. Paint brush and rubber tool are available at any time, to adapt the transfer function to the specific needs of the situation. An example for the two-dimensional automatic transfer function can be found in figure 6.11.

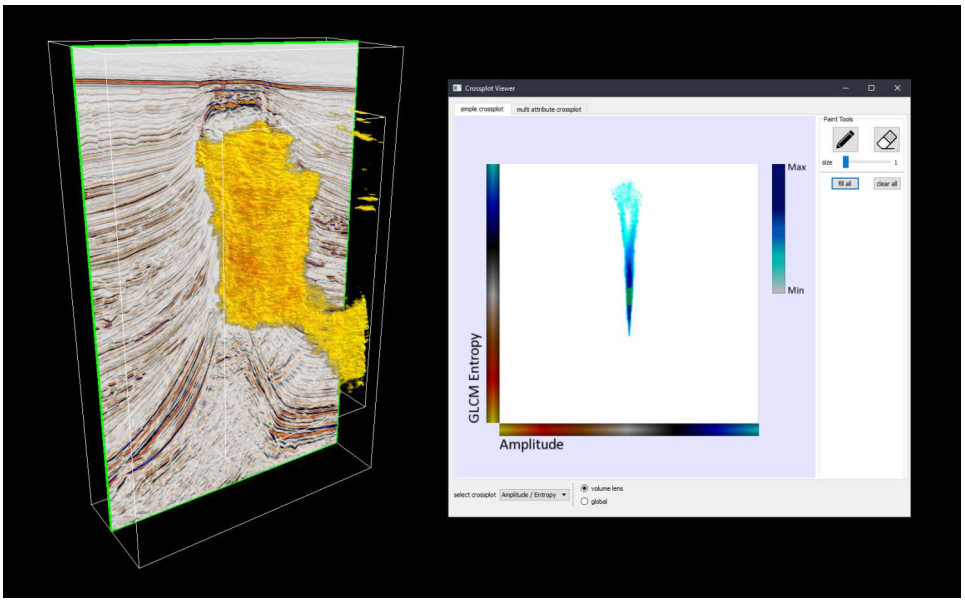


Figure 6.11: Automatic isolation of a horizon in the F3 dataset [15]

As it can be seen, the 2D algorithm was used on the combination of the amplitude and entropy attributes. The automatic transfer function is able to isolate a detected saltdome. All values not belonging to it are automatically hidden in the volume rendering. The corresponding Histogram 2D Widget can be seen on the right side of the volume. The local histogram, anomaly color mapping and automatic transfer function are adapted whenever the active rank changes, similar to the Histogram 1D Widget.

7 | Evaluation

In order to evaluate this new approach, experiments with the algorithm on synthetically generated data were conducted. Additionally multiple expert interviews were realized in collaboration with institutions of the oil and gas industry. These interviews were based on the application of the approach on real-world data. The performance of the algorithm and the results of the evaluation are presented in this chapter.

7.1 Implementation and Performance

One of the requirements defined in section 5.1 is to achieve a fast execution speed of the algorithm. Since the aim is to provide a first overview of different regions in the dataset, a long time-exposure would hamper the benefits of using the algorithm instead of performing a manual analysis. Thus it is vital to utilize the capabilities of hardware acceleration. Hence the algorithm was entirely implemented in CUDA. Nearly all steps can be performed in parallel, as the calculations of the respective bricks do not depend on each other. The subdivision and subsequent merging of the octree is performed layer by layer. Thereby all bricks of the respective layer are processed in parallel. Once the octree is fully generated, the local histograms of the final bricks are stored in the global GPU memory. Then all of the existing bricks access the histograms of their neighbors and the total dissimilarity is calculated for every brick in parallel. Solely the last step of the algorithm, which is the sorting of bricks to obtain the ones with the biggest dissimilarity, is performed sequentially on the CPU.

To verify the compliance of the speed requirement, the runtime was measured under different conditions. The results of the measurements are displayed in the column chart in figure 7.1. All measured times were taken during the execution of the algorithm on the F3 dataset [15] with an extension of $600 \times 950 \times 412$ and a file size of 420 megabytes (MB) per attribute. The utilized workstation has the following specifications:

- Intel Xeon CPU E5-2687W 0 @ 3.10 GHz
- 64 GB RAM
- Nvidia Quadro K6000

All measured times and the number of generated bricks are additionally provided in table 7.1 more detailed. As it can be seen, the algorithm mostly concludes within a couple of seconds. The only higher values are the ones for

Octree Level	1	2	3	4	5
1D runtime in seconds	0.4	0.5	0.6	1.3	12.3
2D runtime in seconds	0.8	1.8	2.3	5.1	49.2
Number of generated bricks	8	64	448	3600	29359

Table 7.1: Comparison of the 1D/2D algorithm runtime and the number of generated bricks for different octree levels

the maximum octree depth of 5 levels, which are 12.3 seconds for the one-dimensional and 49.2 seconds for the two-dimensional case. Since all of the measured times are below one minute, the requirement of a fast execution speed can be considered as fulfilled.

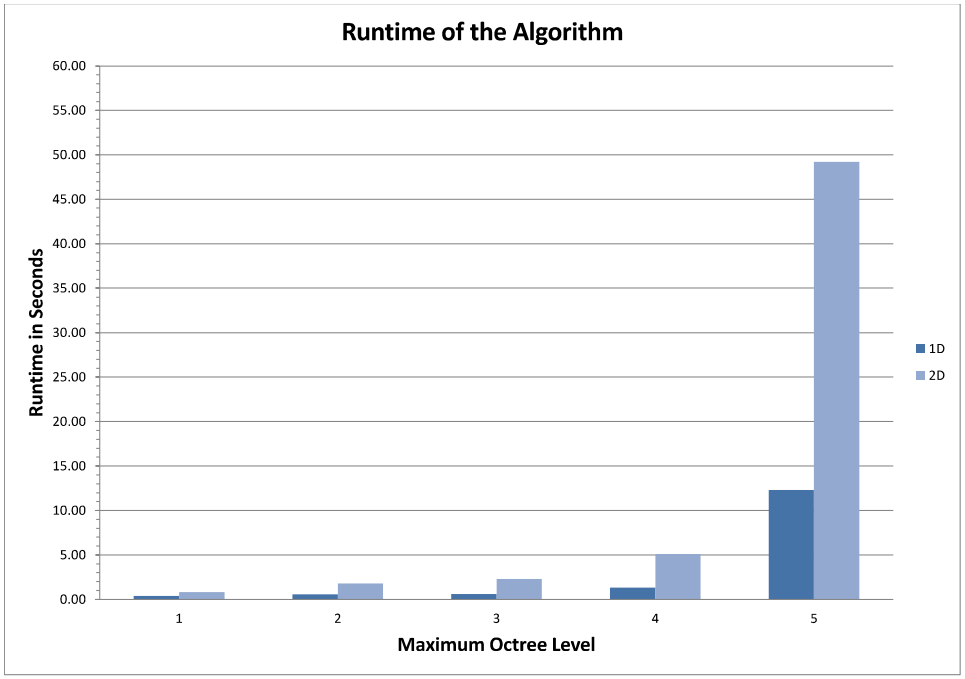


Figure 7.1: Runtime of the algorithm

7.2 Tests with Synthetic Data

The aim of synthetic data is to approximate a real-world situation under certain testing conditions. In this case the desired anomalies can be artificially produced at known locations, allowing the detection capabilities of the algorithm to be evaluated with ground-truth data. The generation of the data and the results of the experiments with these are described in the following sections.

7.2.1 Generation of Synthetic Data

As suggested by experts from Leibniz Institute for Applied Geophysics (LIAG) [18], the synthetic datasets were generated based on one seismic trace, which was extracted from a real-world dataset. This way a valid seismic signal was the foundation of the data, which is important to guarantee the validity of the test results. The seismic trace was duplicated and spread across the whole dimensions of the synthetic dataset. Then a Gaussian noise was added to the amplitude values, to increase the variance of the data. Finally a random vertical offset was applied to recreate the layering characteristic of seismic data, which is not totally homogeneous and parallel. The result of these steps is a synthetic template dataset, that follows the typical normal distribution of the amplitude. In the next step the anomalies were added, by modifying the values at certain locations manually.

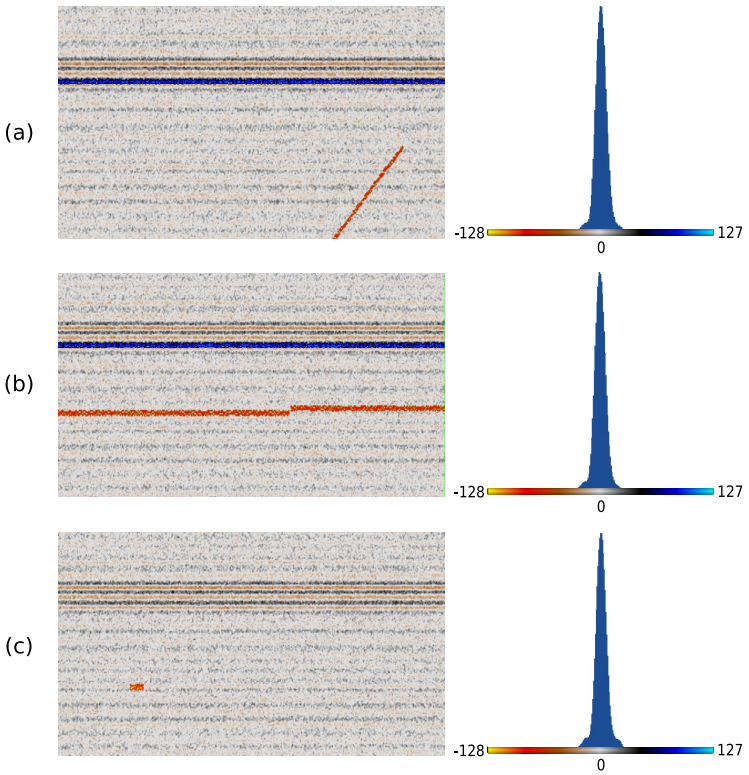


Figure 7.2: Synthetically generated tilted horizon (a), faulted horizon (b) and bright spot (c)

A total of three datasets were created, containing the following anomalies:

(A) One tilted and one straight horizon

(B) One faulted and one straight horizon

(C) Three bright spots

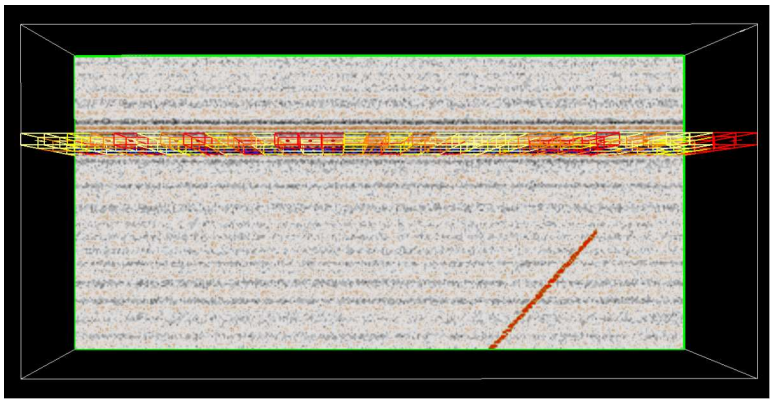
Figure 7.2 shows slices of the synthetically generated datasets and the contained anomalies. On the right side of the slices the global histograms of the dataset are shown. As it can be seen, all of the three datasets still follow the Gaussian distribution of the amplitude. Thus the anomalies cannot be detected on the global scale.

The synthetic datasets were generated using python and the open-source library *Segpy* [34], which supports the import, adaptation and export of SEG-Y datasets.

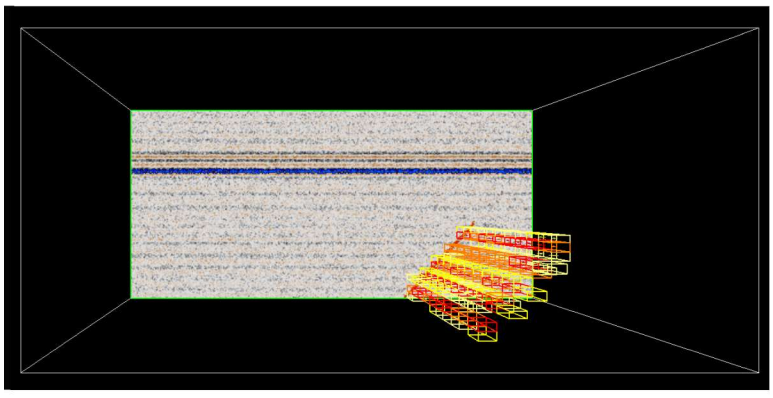
7.2.2 Results

In this section the results of the experiments with the synthetic datasets are presented, differentiated between the three datasets.

(A) - Tilted Horizon



(a)



(b)

Figure 7.3: Global visualization of the 26 neighborhood (a) and the horizontal neighborhood (b)

The first synthetic dataset contains one tilted horizon and one straight horizon. Experiments with the algorithm have shown, that the 26 neighborhood is only capable of detecting the straight horizon, while the tilted one is not found. A screenshot of the global visualization can be found in (a) of figure 7.3. The detection of the tilted horizon is not possible with this neighborhood, as the anomalous values are also present in many nearby bricks.

As opposed to this, the values of the upper horizon vary in all vertical and diagonal directions. Equal results are obtained when using the vertical neighborhood. Just the horizontal neighborhood is able to detect the tilted horizon, due to the obvious value change in this direction. However, the upper horizon is not found anymore. A screenshot of the detection of the tilted horizon can be found in (b) of figure 7.3.

(B) - Faulted Horizon
The second synthetic dataset contains one straight horizon and one horizon interrupted by a fault. The 26 neighborhood and the vertical neighborhood are able to detect both horizons, due to the change of the value distribution in the corresponding directions. But the faulted region itself is only detected by the horizontal neighborhood. This is due to the strong difference of the values at the fault plane. However, the other parts of the horizons are merely detected, since the value distribution is equal in the horizontal direction. The global visualization of the detected fault can be found in (a) of figure 7.4. The screenshot in (b) shows the corresponding glyph visualization. It can be seen, that the arrows point into the direction of the fault plane and thus towards each other. This is a good example of how the arrows can indicate the presence of anomalous structures.

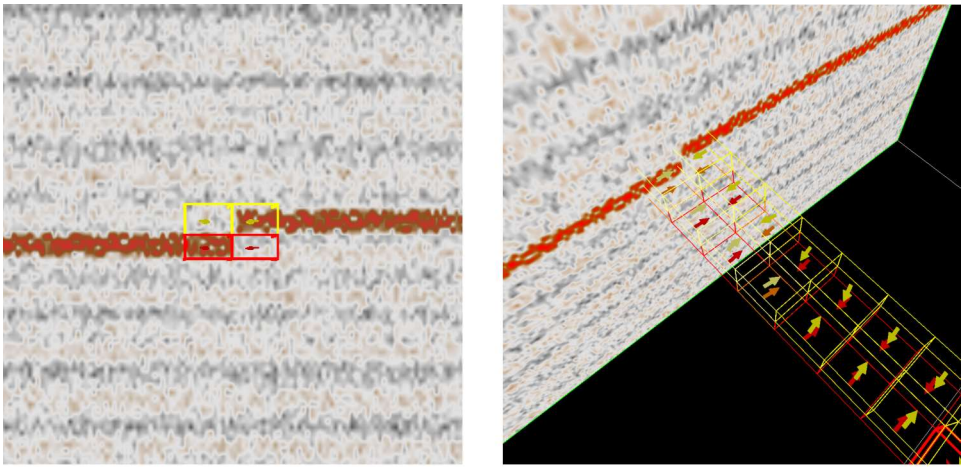


Figure 7.4: Global visualization of the detected fault in the horizon

(C) - Three Bright Spots
The last synthetic dataset used for the experiments contains three bright spots. Two of the three are detected by all of the neighborhoods. One of these can be seen in figure 7.5. Note that only the brick in the center was rated with a high rank, while the other two bricks were displayed in black manually for the sake of the illustration. The local histograms of the three bricks can be found below. One can easily see, that the histogram in the center differs clearly from both neighboring histograms. It contains negative amplitude values represented by yellow and red color. These are caused by the bright spot and are not present in the other two histograms. Hence the dissimilarity measures are able to recognize the brick as an anomaly, when comparing it to the neighbors.

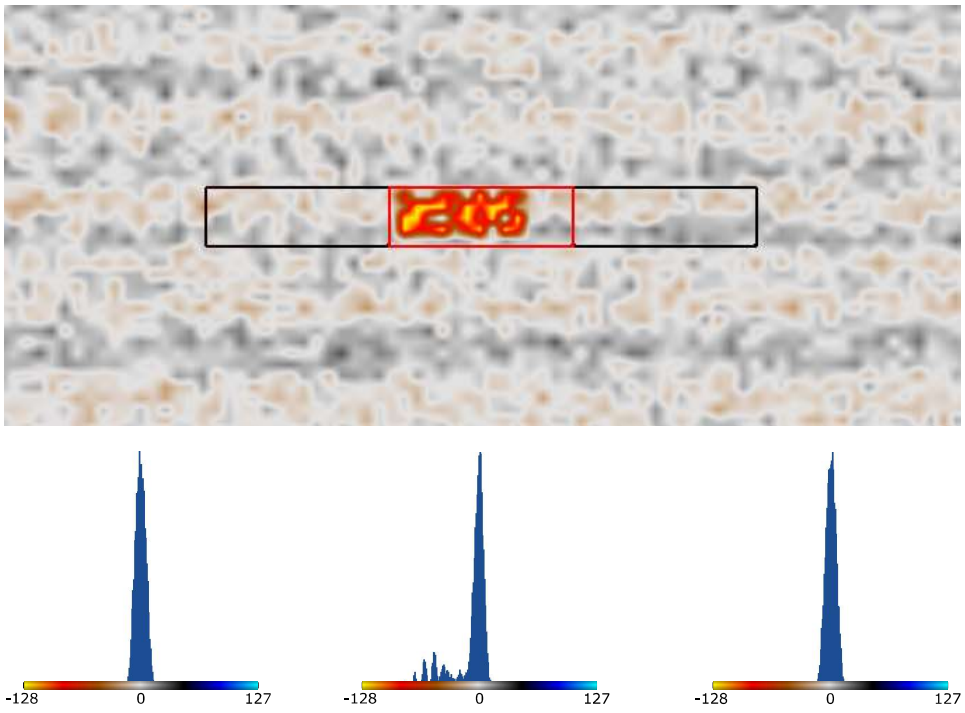


Figure 7.5: Detected bright spot and local histograms of the three bricks

While the second detected bright spot is similar to the one presented, a different case has occurred regarding the third bright spot in the dataset. It is positioned directly on the edges of four adjacent bricks. Thus the amount of anomalous voxels distributes into four local histograms, instead of one. Hence when comparing one of the bricks to the bricks in the neighborhood, the values are present in both histograms. This leads to the bright spot not appearing to be an anomaly on this local scale. A screenshot of the

situation can be found in figure 7.6. Here again the bricks were displayed in black manually, to provide a better understanding of the issue. The unfavorable position of the bright spot can be seen on top. Below are again the local histograms of the three bricks of the upper row. Here, especially in contrast to figure 7.5, the local histograms are merely distinguishable. The histogram in the center only contains a couple of anomalous values in the positive amplitude range. Yet the values can also be found in the left histogram. They are not present in the right histogram, as the brick does not contain parts of the bright spot. The distribution of the anomalous values into multiple bricks reduces their total frequency and thus the detectability in the histogram. Some ways of solving this issue were developed during the expert interviews and are given in the next section.

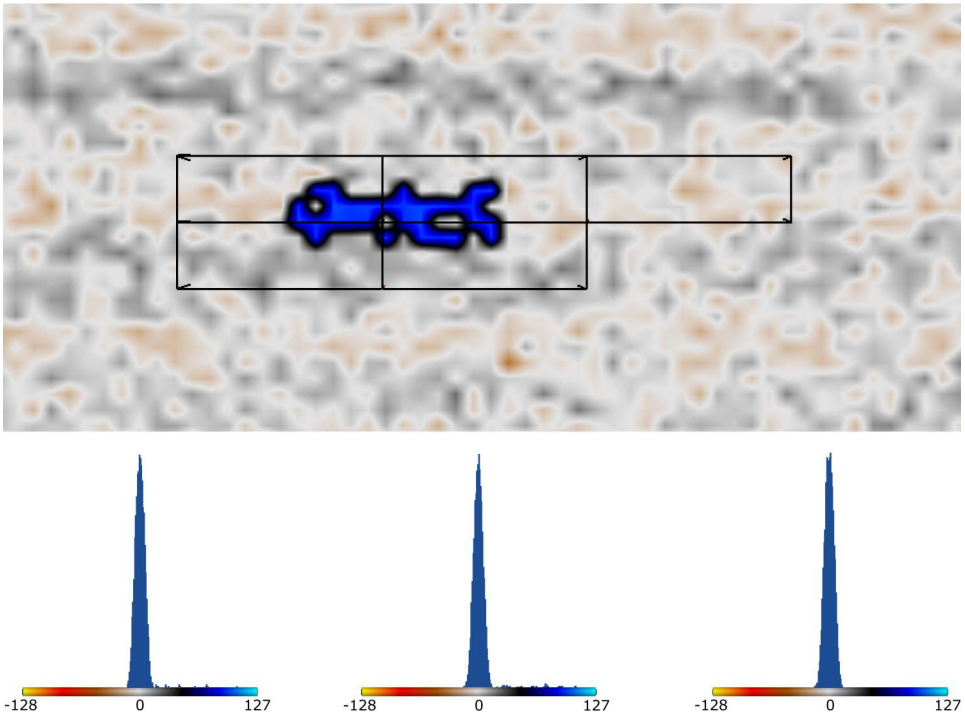


Figure 7.6: Bright spot split into multiple bricks and local histograms of the top row

In addition to the presented experiments, different dissimilarity measures were tested on the various neighborhoods. These tests have revealed, that the type of dissimilarity measure has almost no impact on the results. Just some of the bricks are ranked on a slightly different spot. The overall results however are very similar. The reason for that might be, that only basic dissimilarity measures were used for the first experiments. There is a chance

that more proficient dissimilarity measures, such as the Earth-Movers Distance [39], yield different results. This topic might be an interesting direction for further research.

7.3 Expert-Interviews

In addition to the experiments with synthetic data, multiple expert interviews were conducted with members of the VRGeo Consortium [11] and further institutions from the oil and gas industry. These kind of interviews are a frequently used technique in qualitative research and provide an insight into the thoughts of domain experts, allowing to benefit from their experience. As opposed to this, a quantitative evaluation cannot provide this kind of information [20]. The developed algorithm is a new approach and a qualitative study is more useful when little is known, to get a first idea of the benefits [6]. Thus a qualitative interview was chosen to gather information on the strengths of the developed algorithm and to find elements for further improvement.

The interview was structured in a flexible way to improve the quality of the resulting data. On one hand, the conversation had to be guided by defined questions. On the other hand, it had to be dynamically adaptable to allow the experts to develop their own ideas on how the algorithm could be beneficial. Thus the method of a *semi-structured* interview was utilized, as it combines specific questions with an open discussion [20].

A *topic guide*, which is often used in these kind of interviews, was prepared. It consists of a list of key questions, which were to be covered. Additionally it could be extended with different prompts during the respective interviews, encouraging the participant to talk about various ideas and specific issues [6]. This guide can be found in the appendix of this thesis. Note that the same topic guide was used among all participants. Only the utilized datasets were exchanged between the respective interviews.

The aim was to determine the benefits of the algorithm and the presented visualization. However, the usability and user experience of the software were not part of the evaluation. Thus the interviews were conducted by a single interviewer, who also controlled the software and lead the discussion. Although statistical representativeness is not part of qualitative research implicitly [6], the sample was taken in a systematic way to ensure that multiple perspectives have an influence on the results. Hence the selected participants work in diverse geological fields. Furthermore experts with a background in research, as well as employees of industrial companies were involved. A total of 12 experts were interviewed. Half of these work for in-

dustrial companies and institutes like ExxonMobil [12] and DMT [19]. The other half are involved in research facilities, like the department for geophysics of the University of Cologne [35], the Earthquake Station in Bensberg [46], the Leibniz-Institute for Applied Geophysics (LIAG) [18] and the RWTH Aachen [1]. The interviews took around 60 minutes each. Two different datasets, which are the freely-available Parihaka [42] and F3 [15] volumes, were used in the interviews. Each participant was given one of these datasets, so that both were used equally often. This way the results were not biased by the characteristics of one dataset.

7.3.1 Procedure

In the first step of the procedure, introductory slides were shown to each participant. These did contain the basic idea and theory of the algorithm, as well as an explanation of the different elements of the visualization. Furthermore the procedure of the live demo was presented. Finally the slides included additional information on the utilized dataset, for instance the dimensions and geographical location, as this knowledge is of importance for the analysis.

After the short introduction, the live demo of the software was conducted. The results of the algorithm were prepared and exported in advance, as described in section 5.5. This way the repetitive execution of the algorithm was avoided. The results were imported one after another and shown to the participant. A total of five results were used. Among these were different neighborhood types and attributes to compare their performance. As the experiments with the synthetic datasets have shown, the dissimilarity measures do not influence the results considerably. Therefore the measures were not part of the interviews. Instead all of the different results were generated using the L_1-distance (comp. eq. (4.1)). During the live demo, the focus was on the elements of the visualization. At first, the global visualization was inspected in order to detect clusters of important bricks. The experts were asked to validate, whether the corresponding regions are of importance for the interpretation. Secondly, the experts focused on the top ranks and examined the corresponding regions for interesting structures individually. Lastly, the local histogram with the anomaly curve, the direction of the arrow-glyphs and the visible structures in the volume rendering had to be evaluated.

The last step of the interview were a couple of general questions that focused on the benefits of the approach and the integration into the day-to-day workflow of an interpreter.

The aim of the expert interviews was to find answers to four main questions:

1. Can the algorithm be used to receive a global overview of different structures?
2. Does the algorithm detect interesting anomalies and rank these accordingly?
3. How can the algorithm be integrated into the seismic interpretation workflow?
4. Which fields of the seismic interpretation can be supported with the help of the algorithm?

The results of this evaluation are presented in the following section.

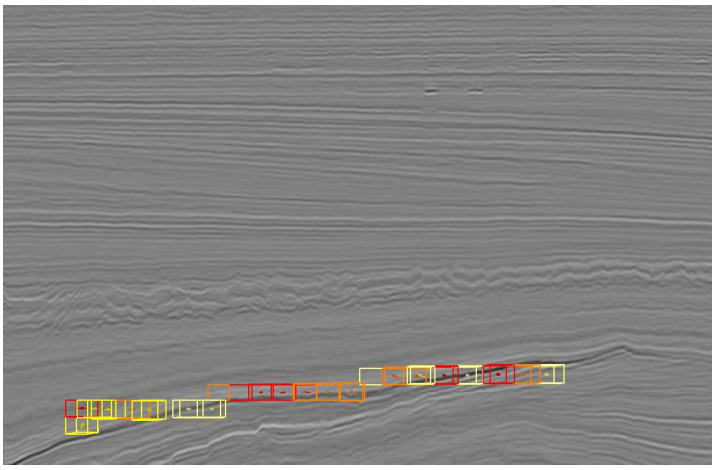
7.3.2 Results

The interviews have provided valuable insights and the received feedback was very positive. In general the approach was rated to be very promising. The experts approved that using histograms for detection of anomalies is a novel idea and was priorly not used in the context of volume data. A comprehension of the results, enriched with examples of the structures detected during the interviews, is given in this section. Therefor the four main questions of the evaluation are used as a framework.

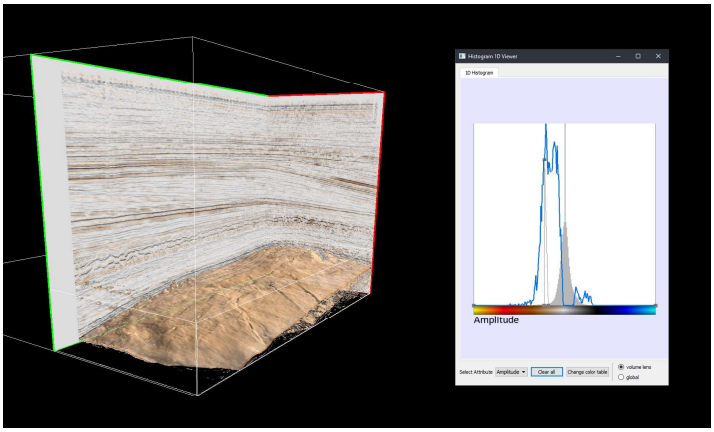
1. Global overview of different structures

The first aim of the expert interviews was to determine, if some regions and structures can be identified by only investigating the colored bricks in the global visualization and their clustering. The experts were able to assume the presence of different structures and important regions directly.

The spatial brick arrangement corresponds to the shape of the underlying structures. One example therefor is the horizon that can be seen in (a) of figure 7.7. The horizon moves downwards to the left, which is represented in the positioning of the bricks. It was detected in the F3 dataset with the 26 neighborhood and the brick with the rank number 1 was contained in the center. Using the vertical neighborhood, the horizon was detected equally successful. Only the horizontal neighborhood was not able to find it, as the value distribution does not change in that direction. With the help of the automatic transfer function, the detected horizon could be isolated entirely in the volume rendering, as shown in (b) of figure 7.7.



(a)

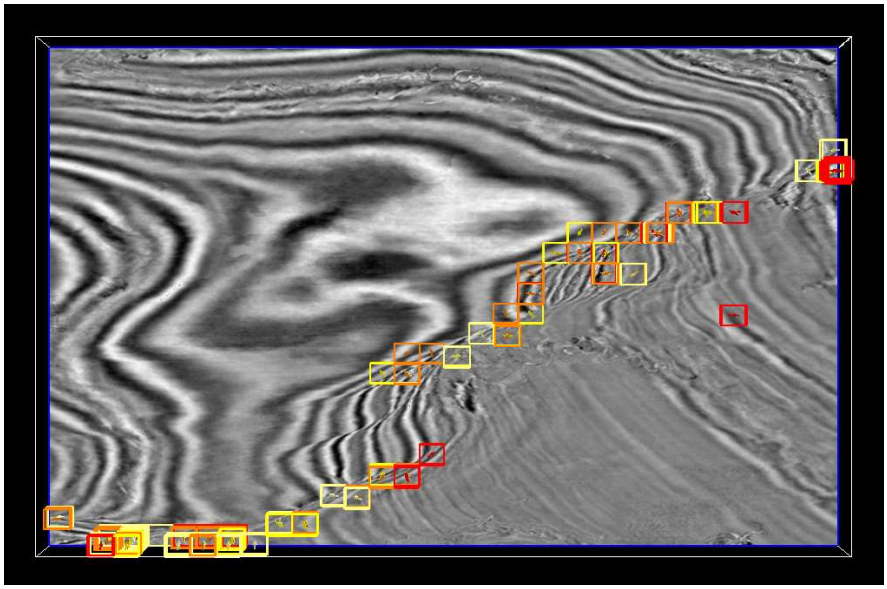


(b)

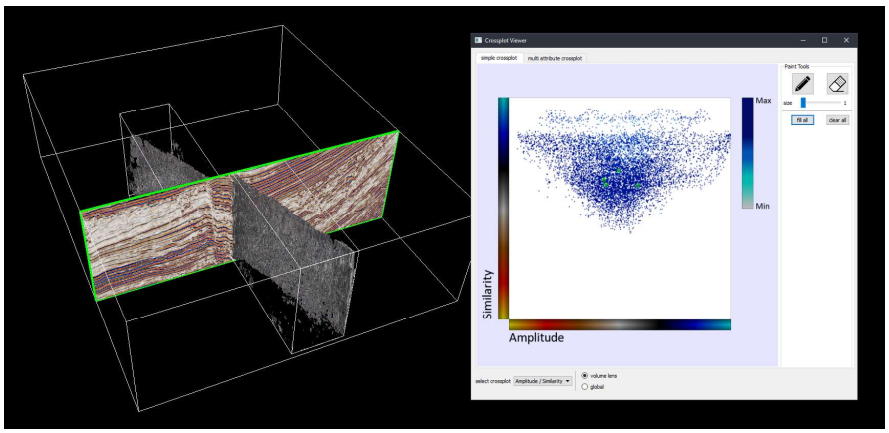
Figure 7.7: Horizon detected by the algorithm (a) and volume rendering with automatic transfer function (b)

The experts also found the 2D comparison, which detects anomalies in a combination of two attributes, to be very important. When features can be found in multiple attributes particularly well, this also increases the chances of detecting them as an anomaly. Further there are attributes, which are specifically designed for highlighting certain features. These attributes can be used in combination to detect exactly these features with the algorithm. For instance the dip attributes can be used to find unconformities and the similarity attributes can be used to detect faults [29]. One example therefor is given in figure 7.8, where two faults were found in the Parihaka dataset with the combination of the amplitude and similarity attributes and the horizontal neighborhood. These faults cannot be detected, when using only

one of the given attributes. Here again the global visualization allows to identify the structure directly, as it can be seen in (a). The clustering of the bricks corresponds to the location of the respective fault planes. In (b) the isolation of the structures with the automatic transfer function is shown.



(a)



(b)

Figure 7.8: Detected fault in the Parihaka dataset (a) and isolation of the structure with the automatic transfer function (b)

An even further increase of the number of possible attributes was desired by the experts. The principle of the algorithm can also be applied to histograms with a higher dimension. Yet one needs to keep in mind, that every further dimension increases the required memory and the runtime of the algorithm significantly. Another promising suggestion by one of the experts was to

use a different type of global visualization. The information of the arrow-glyphs could be used to create isolines between the bricks. Then the bricks could be removed and the isolines would mark the border between different structures. Therefor an algorithm would have to be developed, that chooses the optimal path of the lines between the bricks depending on the direction of the arrows. This would most likely be the path, on which the arrows of the adjacent bricks point towards each other, indicating a change of region between them.

2. Detection of interesting anomalies

In addition to the impressions of the global visualization, the top ranks were investigated individually during the interviews. All of the experts agreed on the algorithm detecting very promising regions on the top ranks, which would be of interest for the seismic interpretation. Consequently the algorithm gives further advice for the decision-making during the analysis of a new dataset.

At first, the experts found the algorithm to detect strong reflections very well. These can for instance be small bright spots. An example of a bright spot detected by the algorithm, can be found in figure 7.9. In (a) the bright

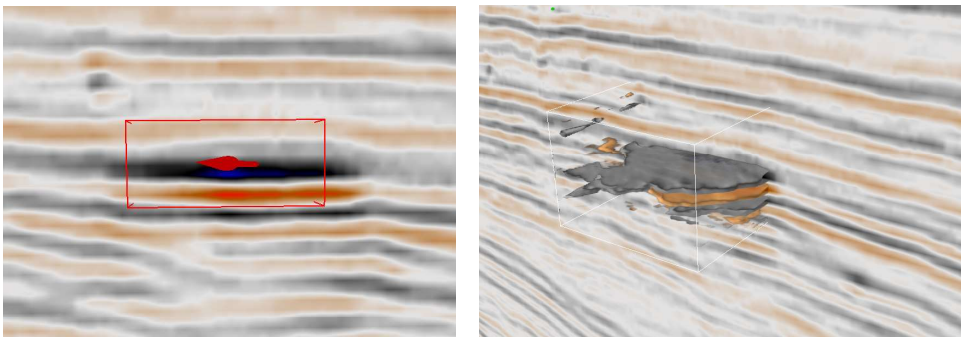


Figure 7.9: Bright spot detected by the algorithm in the F3 dataset

spot with the corresponding brick is shown. The automatic transfer function was able to isolate the spot in the volume rendering, as it can be seen in (b). The anomaly was ranked fourth and found on an inline slice by using the horizontal neighborhood. In figure 7.10 the histogram of the brick with the bright spot and the histogram of the region directly to the left of it can be seen displayed in gray. One can easily see, that the local histogram of the bright spot in (a) looks very different to the one in (b). It has additional peaks in the smaller positive and negative ranges, which are not present in the other one. These amplitude ranges were also rated to be the ones that

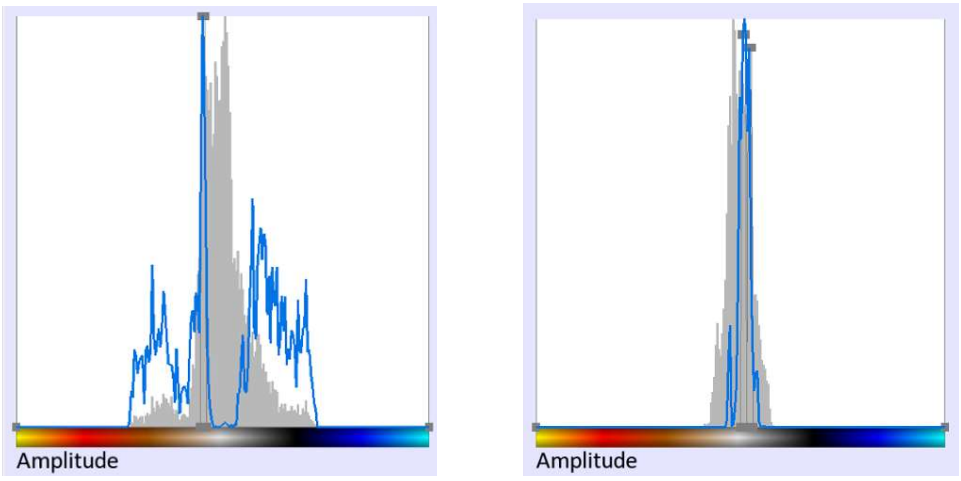


Figure 7.10: Local histogram of a brick including a bright spot (a) and local histogram of a neighboring brick without anomaly (b)

are dissimilar to the neighborhood, as it can be seen through the peaks of the blue anomaly curve. As opposed to this, the histogram in (b) follows the typical Gaussian distribution of the amplitude values. Furthermore the peak of the anomaly curve in (b) is also around zero, as these amplitudes are not present in the anomalous brick. The region to the left of the detected brick was solely chosen as an example. Other adjacent bricks have similar histograms through an equal value distribution, as it can be seen on the slice in figure 7.9.

But the detection of strong reflections is not the only advantage of the algorithm. Changes in the layering of the dataset are also detectable. One example therefor is the peak of an anticlinal structure, found on an inline slice of the F3 dataset by using the 26 neighborhood. The corresponding brick was rated on rank 10 and can be seen in figure 7.11. While the horizontal neighborhood did also find the structure but ranked it differently, the vertical neighborhood was not able to detect it. This might be due to the fact, that the 26 neighborhood includes a huge portion of the Gaussian distributed bricks of the neighborhood into the comparison, to which the anomalous part is very different. This amount is smaller for the horizontal direction, leading to a lower rank. As the F3 dataset is part of the training of the open-source software opendTect [16], well-data is freely available and can be used to draw conclusions on the material in the seismic. These wells are drilled paths into the earth. They come with a *well log*, that contains measurements of physical quantities in or around the well [29]. One of these wells runs directly through the detected brick of the anticlinal structure.

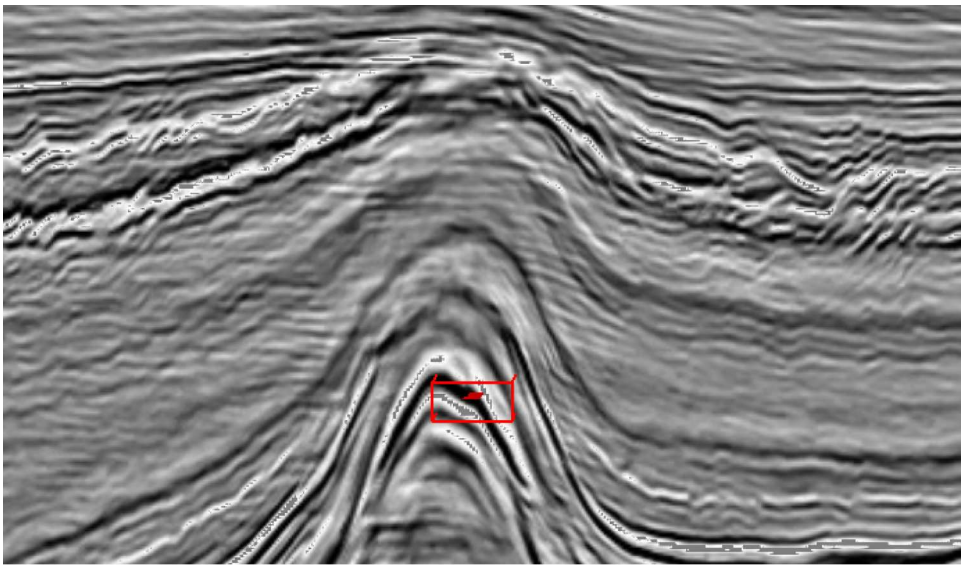


Figure 7.11: Detected anticlinal structure in the F3 dataset

When overlaying the slice with the well log, the region of the detected brick is covered with a local gamma ray maximum. This is visualized with a brown color in (a) of figure 7.12. In (b) the corresponding well log is displayed. The high gamma ray indicates the presence of oil and gas in that area, as it was noted by the experts. Above the detected brick is a so-called *gas chimney* structure, which is a subsurface leakage of gas from a poorly sealed hydrocarbon accumulation [29]. This shows that the region detected by the algorithm is promising for further interpretation.

Another example for structural anomalies are folded horizons, which are found to some extend. The amount of folding can lead to the horizon being located on the border of bricks. Thus the detection of the whole horizon might not be possible in these parts. A folded horizon for which that is the case can be found in (a) of figure 7.13. The horizon is detected on a crossline slice of the F3 dataset with the vertical neighborhood. As it can be seen, the left part of it was sorted into higher ranks without issues, as the horizon is only present in the upper bricks. The bricks below are also rated higher, since the vertical neighborhood determined that these bricks differ locally from the bricks above. A bit of the right side was also detected, as it is only present in the lowest brick. However, the whole part in the middle was not discovered, since it runs along between the bricks and influences the value distribution of both histograms. The difference between the histograms is not significant enough, to be detected by the algorithm. In this situation the automatic transfer function is very useful. Although the middle part of the horizon was not detected directly, the anomalous values are present

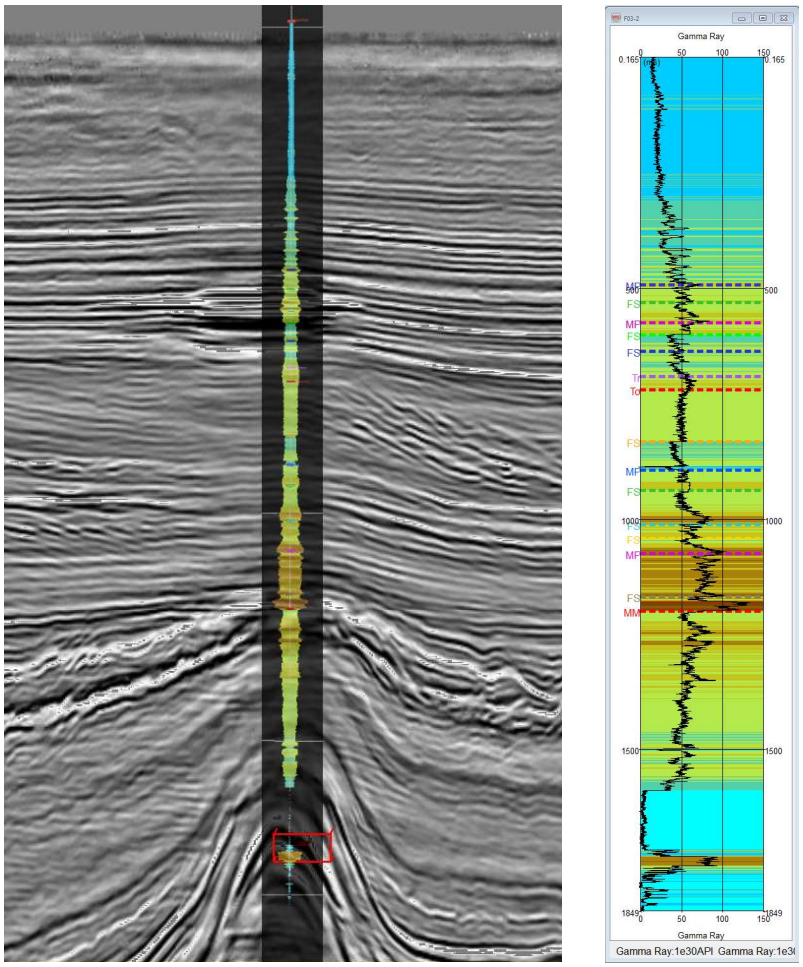
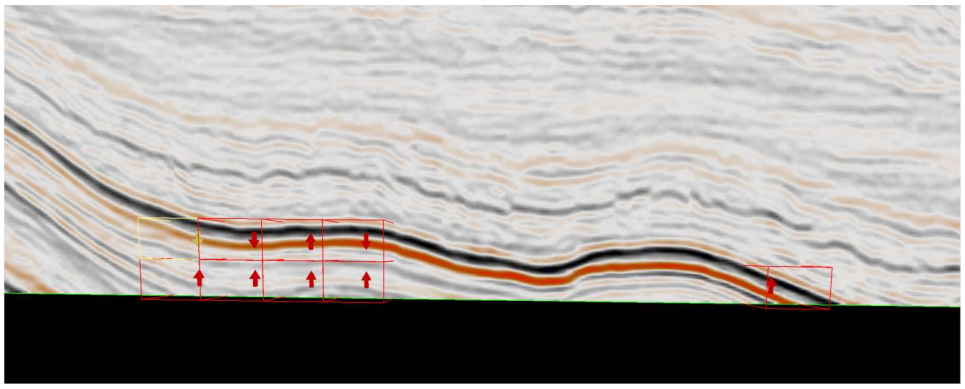


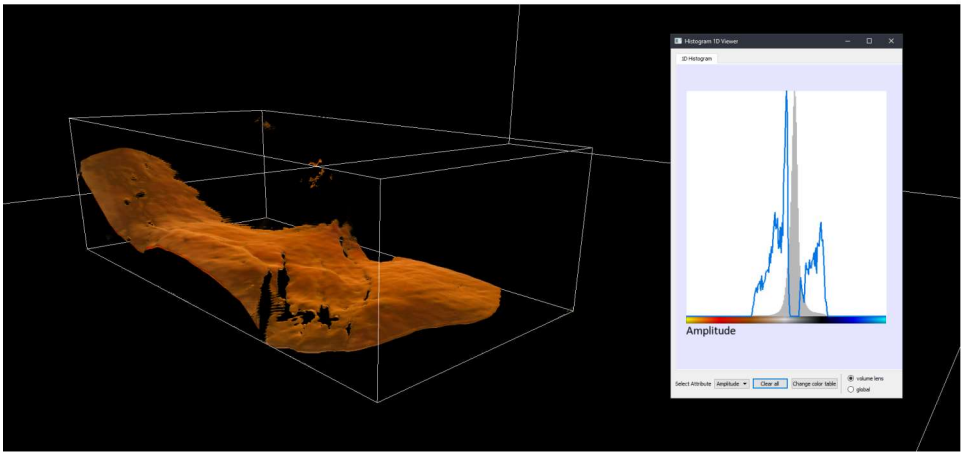
Figure 7.12: Overlay of well on detected anticlinal structure (a) and corresponding well log (b)

and equal to the rest of the horizon. Therefore when extending the volume lens across the area, the whole horizon appears in the volume rendering, as it can be seen in (b) of 7.13. This feature was rated to be very convenient by the experts. In addition to displaying not detected parts of one single structure, the same principle can be used to find similar structures in other parts of the volume, that can also be isolated with the same transfer function.

As it could be seen in the previous examples, the detectable feature-types depend heavily on the utilized neighborhood. In general, the horizontal neighborhood is not suitable for structures, which spread in the same direction. This is due to the fact, that the data values do not change much there. For example horizons are more likely to be found, when using the vertical



(a)



(b)

Figure 7.13: Folded horizon not detected entirely (a) and isolation of the complete structure with the automatic transfer function (b)

neighborhood. Similarly the horizontal neighborhood is more adequate for features like faults, which cause a change of values in the horizontal direction. The 26 neighborhood appears to be an allrounder. It is able to detect most feature types, if the corresponding structure is positioned accordingly. Yet it has a bigger chance of missing certain features, which only spread in one of the 26 directions. In that case the more specialized neighborhoods perform better. Thus the 26 neighborhood is suited best for a first overview of important structures, while the other neighborhood types provide more value, when targeting specific structures.

Another important requirement for the algorithm was to be unaffected by artifacts. One of these artifacts is a so-called *multiple*, which is seismic energy that was reflected several times and appears as an additional event [29].

The expert interviews have shown, that the algorithm is not misguided by these. This might be the case, since they do not affect the value distribution of the histogram by a noticeable amount.

Some improvements for the visualization were suggested by the experts. It should be possible to show the bricks and histograms directly adjacent to the one in focus for reference, even if they are not rated to be anomalous. The experts were particularly interested in the neighboring regions. This was especially important in cases, in which only one brick was detected to contain anomalies, while the whole neighborhood did not seem to deviate from the normal distribution. The knowledge of the ranks of the surrounding bricks, and precisely to find out why these were graded differently, could improve the understanding of the results. Thus it should be possible to show the bricks temporarily. An additional window could then display all of the histograms in the neighborhood, allowing a user to compare them. One further improvement could be the combination of particular results of the algorithm. This could happen either by using a logical operator on multiple results, or by combining the results visually with a special kind of visualization.

As the experiments with synthetic data and the results of the expert interviews have shown, occasionally features are not detected when they are situated on the border of the bricks. Further subdivision cannot be used to cope with this issue, since the borders of the bricks remain in place. Two different solutions can be considered to solve this issue. The first one would be to allow an overlap of the bricks, so that the features located on the border are then present in both and might be detected this way. The octree data structure would have to be adapted accordingly, because it only supports non-overlapping, space-filling regions by nature. Another solution would be to implement a sliding detector, which has a certain size and moves through the dataset voxel by voxel. For every step, a full comparison between the current local histogram and the ones in the neighborhood would have to be performed. This way the exact step, in which the sliding brick covers the anomaly perfectly, would result in the highest dissimilarity. Yet this extension would increase the runtime of the algorithm significantly.

3. Integration into the seismic interpretation workflow

Besides the results of the algorithm, the question on how it can be integrated into the seismic interpretation workflow was of major importance. As it was shown, the algorithm can be executed in a short period of time and the global visualization provides valuable information on the importance of different regions in the data. The experts were directly able to assume the presence of certain structural features, just from looking at the global

visualization. Additionally the highest ranked bricks do contain important structures, which are suitable as starting points for a first analysis. The arrow-glyphs inside of the bricks were very useful for identifying the origin of the biggest change in the data distribution. Furthermore the automatic transfer function was able to isolate the target structure in nearly all of the cases and can be used to detect similar structures in other regions of the dataset. Thus the algorithm is especially useful when starting the interpretation of a new, unknown dataset.

The idea of generating an anomaly attribute from the results of the algorithm was appreciated by the experts. It allows the seamless integration of the algorithm into their workflow. The experts stated that they typically use a handful of different software packages depending on the current task. By creating a SEG-Y anomaly attribute, the results of the algorithm can be imported and used in different software packages fluently.

4. Fields of application

One objective of the algorithm was to provide a first overview of important structures in a new dataset, which is already fulfilled. Beyond that, the expert interviews were used to ascertain, if additional tasks of the seismic interpretation can benefit from the results of the algorithm. The experts determined, that the algorithm could provide value to two different fields of application.

At first, it could be used to detect and extract big structures like saltdomes or horizons. This would help to prevent the tedious manual extraction on a per-voxel basis, which is typically done today. One requirement would be, that the features are detected entirely and definite, which is currently not always the case. Yet the combination of the ranked bricks and the automatic transfer function can be used to extract the structures completely.

The other possible field of application is the detection of direct hydrocarbon indicators (DHIs). These bright spots are small regions with a high amplitude value. They indicate the presence of oil and gas, as explained in section 2.4.2. Thus the localization of these would support the search for hydrocarbons. As it can be seen in the example of figure 7.9 and the experiments with synthetic data in section 7.2.2, the DHIs can be detected by the algorithm, depending on their size and position. Since these spots are typically very small, the number of subdivisions in the octree was sometimes not adequate, so that the DHIs did not occupy enough space in the brick to affect the local histogram sufficiently. The maximum level of octree depth would have to be higher, which leads to bricks of a smaller size. Then the DHIs could be detected more reliably. However, this would have the

side-effect of increasing the runtime of the algorithm exponentially. Besides the bright spots being too small for the current size of the bricks, they can also be located directly on the border of them, as the experiments with synthetic datasets have shown. The same situation was described in case of the horizons before and the solutions of using an overlap or a sliding detector could also solve this issue for the detection of the bright spots.

Finally the experts suggested to apply the algorithm on data from different domains, such as ground-penetrating radar or meteorological data. This could help to get a better understanding of how the algorithm works and how it can be improved.

Summary

In summary the expert interviews have revealed, that the algorithm is capable of detecting important anomalies in the dataset. At first the algorithm can be executed very fast as presented in section 7.1, which makes it applicable during an initial screening of an unknown dataset. Additionally it was shown, that the detected results are feature-independent, allowing the application in a multitude of possible scenarios. Furthermore the results of the algorithm are reproducible, since the histograms and dissimilarity measures are always calculated identically. Thus almost all of the defined requirements of section 5.1 are satisfied. Solely the reliable detection of features in their entirety cannot always be guaranteed yet. However the proposed improvements are very promising to resolve these limitations.

8 | Conclusion and Outlook

In this thesis, an algorithm was developed for the detection of anomalous regions in a seismic volume dataset. The basis for the algorithm is the comparison of local histograms in a neighborhood, to detect the diverging parts. A software was developed, which implements the algorithm and visualizes the results. Furthermore it offers multiple tools to support the interpretation of the data. The algorithm was evaluated in multiple ways. At first, experiments with different synthetic datasets were performed. Further an expert-interview was conducted, in which the software was applied on real-world seismic datasets. The 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.

As the approach is a proof of concept, the integration into the seismic workflow is not clarified yet. In addition to the pure detection of anomalous regions, it could be useful in two particular tasks of the interpretation.

At first, the extraction of structural features like salt domes or horizons could be performed automatically, which could save the time of extracting the structures manually. However, the algorithm does not always detect these features entirely. This is due to the possibility of a structure being situated on the border of two bricks. The issue can probably be solved, by allowing the bricks to overlap or alternatively by using a sliding brick detector, which moves forward voxel by voxel and does a full neighborhood comparison at every step. These improvements could ensure the full detection of those structures.

The other possible field of application is the detection of direct hydrocarbon indicators (DHI). As these spots typically consist of merely a couple of voxels, the subdivision of the octree is too coarse, to identify all of the spots reliably. If the octree would be subdivided further, these DHIs could be detected at a higher probability. Contingently the proposed features of overlapping bricks or the sliding brick operator can also facilitate the detection of these.

Until now only some basic distance measures and a selection of neighborhood types were tested. Moreover only a small amount of attributes and combinations of these were used. Since the number of possible input parameters is enormous, an extensive long-term study on different distance measures, neighborhoods and attributes could provide additional ideas on how to improve the algorithm further. This study should be performed together with

seismic interpreters, who integrate the developed application into their daily workflow.

Another interesting idea might be, to combine the algorithm with other methods of automatically interpreting data. For example deep learning, which was introduced in section 4.1, could be combined with the results of the algorithm. Thereby a neural network could be used to give a first estimation on the feature type contained in the detected anomalous bricks. This could enhance the informational content of the global visualization. Alternatively the regions detected by the algorithm could be used as training data for an unsupervised machine-learning approach, similar to the one presented in [32]. To generate the training data, the parts of the slices within the highest ranked bricks could be exported to image patches. Then the machine-learning algorithm could be used to detect natural patterns in the data and sort the bricks into different classes, which could correspond to different geological features. This way the content of the detected regions could directly be classified to support the interpretation even further. The main benefit would be, that the unsupervised learning does not require labeled training data. Instead the training can directly be performed with the extracted results of the algorithm. Through the pre-selection of relevant data, the time of the training and required disk space can be reduced and there is a chance of receiving more meaningful results [24, 30].

In summary the proposed algorithm is promising and has aroused the interest of domain experts. The optimization and further development of the algorithm according to the feedback received, as well as a detailed evaluation of the approach are subject to further research.

List of Figures

Bibliography

[1] RWTH Aachen. Faculty of Georesources and Materials Engineering, 2017. Available at `http://www.fb5.rwth-aachen.de/cms/~hgv/Georessourcen/`, accessed 13th September 2017.

[2] H.N. Alsadi. *Seismic Hydrocarbon Exploration: 2D and 3D Techniques.* Advances in Oil and Gas Exploration & Production. Springer International Publishing Switzerland, 1st edition, 2016.

[3] V. Asha, N. U. Bhajantri, and P. Nagabhushan. Automatic Detection of Defects on Periodically Patterned Textures. *Journal of Intelligent Systems*, Volume 20(No. 3), 2011.

[4] V. Asha, N. U. Bhajantri, and P. Nagabhushan. GLCM based Chi-square Histogram Distance for Automatic Detection of Defects on Patterned Textures. *International Journal of Computational Vision and Robotics (IJCVR)*, Volume 2(No. 4), 2011.

[5] Jan Bender. Deep Learning using three dimensional Volume Data. Technical report, Cologne University of Applied Sciences, 2017.

[6] Nouria Bricki and Judith Green. A Guide to Using Qualitative Research Methodology, 2007.

[7] Alistair R. Brown. *Interpretation of Three-Dimensional Seismic Data.* The American Association of Petroleum Geologists and the Society of Exploration Geophysicists, 6th edition, 2004.

[8] Sung-Hyuk Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, Volume 1(No. 4), 2007.

[9] Israel Cohen, Nicholas Coult, and Anthony A. Vassiliou. Detection and extraction of fault surfaces in 3d seismic data. *GEOPHYSICS*, Volume 71(No. 4), 2006.

[10] The Qt Company. Qt, 2017. Available at `https://www.qt.io/`, accessed 13th September 2017.

[11] VRGeo Consortium, 2017. Available at `http://www.vrgeo.org/`, accessed 13th September 2017.

[12] ExxonMobil Corporation, 2017. Available at `http://corporate.exxonmobil.com/`, accessed 13th September 2017.

[13] NVIDIA Corporation. Cuda, 2017. Available at `https://developer.nvidia.com/cuda-zone`, accessed 13th September 2017.

[14] Mark Cronshaw. Value of Information and other Decision Analytic Techniques for Optimization of Seismic and Drilling. Society of Petroleum Evaluation Engineers (SPEE) in Denver, 2010.

[15] dGB Earth Sciences. Netherlands offshore f3 block, 2017. Available at `https://www.opendtect.org/osr/pmwiki.php/Main/NetherlandsOffshoreF3BlockComplete4GB`, accessed 13th September 2017.

[16] dGB Earth Sciences. Opendtect, 2017. Available at `https://www.opendtect.org/`, accessed 13th September 2017.

[17] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, and Daniel Weiskopf. *Real-Time Volume Graphics*. A K Peters Ltd., 2006.

[18] Leibniz Institute for Applied Geophysics (LIAG). Hannover, Germany, 2017. Available at `http://www.liag-hannover.de/`, accessed 13th September 2017.

[19] DMT Group, 2017. Available at `http://www.dmt-group.com/de.html`, accessed 13th September 2017.

[20] Siw Elisabeth Hove and Bente Anda. Experiences from conducting semi-structured interviews in empirical software engineering research. In *Proceedings of the 11th IEEE International Software Metrics Symposium*, 2005.

[21] Wen-mei W. Hwu. *GPU Computing Gems Emerald Edition*. Morgan Kaufmann Publishers Inc., 1st edition, 2011.

[22] Won-Ki Jeong, Johanna Beyer, Markus Hadwiger, Amelio Vazquez, Hanspeter Pfister, and Ross T. Whitaker. Scalable and Interactive Segmentation and Visualization of Neural Processes in EM Datasets. *IEEE Transactions on Visualization and Computer Graphics*, Volume 15(No. 6), 2009.

[23] Ying Jiang. Detecting geological structures in seismic volumes using deep convolutional neural networks. Master's thesis, RWTH Aachen, 2017.

[24] Peter Joslin. *Recognition and Investigation of Temporal Patterns in Seismic Wavefields Using Unsupervised Learning Techniques*. PhD thesis, Institute for Geoscience, University of Potsdam, 2009.

[25] Alexey Karimov, Gabriel Mistelbauer, Thomas Auzinger, and Stefan Bruckner. Guided Volume Editing based on Histogram Dissimilarity. *Computer Graphics Forum*, Volume 43(No. 3), 2015.

[26] Philip Kearey, Michael Brooks, and Ian Hill. *An Introduction to Geophysical Exploration*. Blackwell Publishing, 3rd edition, 2002.

[27] David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., 1st edition, 2010.

[28] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional Transfer Functions for Interactive Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, Volume 8(No. 3), 2002.

[29] Schlumberger Limited. Oilfield glossary, 2017. Available at `http://www.glossary.oilfield.slb.com/`, accessed 13th September 2017.

[30] Yexin Liu. Applications of Machine Learning for Seismic Quantitative Interpretation. In *Geoconvention, Calgary, Canada*, 2017.

[31] Ross Maciejewski, Yun Jang, Insoo Woo, Heike Janicke, Kelly Gaither, and David Ebert. Abstracting Attribute Space for Transfer Function Exploration and Design. *IEEE Transactions on Visualization and Computer Graphics*, Volume 19(No. 1), 2013.

[32] Amir Hossein Mardan, Abdolrahim Javaherian, and Marzieh Mirzakhanian. Channel detection using unsupervised learning algorithms. In *The 17th Iranian Geophysical Conference*, 2016.

[33] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 26(No. 5), 2004.

[34] Sixty North. Segpy 2, 2017. Available at `https://github.com/sixty-north/segpy`, accessed 13th September 2017.

[35] University of Cologne. Institute of Geophysics and Meteorology (IGM), 2017. Available at `http://www.geomet.uni-koeln.de/`, accessed 13th September 2017.

[36] Daniel Patel. *Expressive Visualization and Rapid Interpretation of Seismic Volumes*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2009.

[37] Bernhard Preim and Dirk Bartz. *Visualization in Medicine: Theory, Algorithms, and Applications*. Morgan Kaufmann Publishers Inc., 1st edition, 2007.

[38] Timo Ropinski, Steffen Oeltze, and Bernhard Preim. Survey of Glyph-based Visualization Techniques for Spatial Multivariate Medical Data. *Computers & Graphics*, Volume 35(No. 2), 2011.

[39] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The Earth Mover's Distance As a Metric for Image Retrieval. *International Journal of Computer Vision*, Volume 40(No. 2), 2000.

[40] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming.* Addison-Wesley Professional, 1st edition, 2010.

[41] Schlumberger. Petrel, 2017. Available at `https://www.software.slb.com/products/petrel`, accessed 13th September 2017.

[42] SEG. Netherlands offshore f3 block, 2016. Available at `http://wiki.seg.org/wiki/Parihaka-3D`, accessed 13th September 2017.

[43] OpenSceneGraph Professional Services. Openscenegraph, 2015. Available at `http://www.openscenegraph.org/`, accessed 13th September 2017.

[44] Pedro Mario Silva, Marcos Machado, and Marcelo Gattass. 3D Seismic Volume Rendering. In *8th International Congress of The Brazilian Geophysical Society*, 2003.

[45] Society of Exploration Geophysicist (SEG). *SEG Y rev 1 Data Exchange format*, 1st edition, May 2002.

[46] Earthquake Station. Bensberg, Germany, 2017. Available at `http://www.seismo.uni-koeln.de/`, accessed 13th September 2017.

[47] M. Turhhan Taner. Seismic Attributes. *CSEG Recorder*, Volume 26(No. 7), 2001.

[48] Luis Vernengo, Trinchero Eduardo, and Satinder Chopra. Deciphering Seismic Amplitude Language. *AAPG EXPLORER*, Issue January, 2017.

[49] Jane Wilhelms and Allen Van Gelder. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics (TOG)*, Volume 11(No. 3), 1992.

[50] Rick Wilkinson. *Speaking oil and gas / written for BHP Billiton Petroleum Pty Ltd by Rick Wilkinson.* BHP Billiton Petroleum Pty Ltd., 3rd edition, 2006.

[51] Xinming Wu and Dave Hale. Automatically interpreting all faults, unconformities, and horizons from 3D seismic images. *SEG Interpretation*, Volume 4(Issue 2), 2016.

Appendix

Fraunhofer IAIS

SEMI-AUTOMATIC FEATURE DETECTION TROUGH ANOMALIES IN LOCAL HISTOGRAMS

Expert-Interview Topic Guide

DATE:
NAME:
INSTITUTION:
USED DATASET:

PROCEDURE (~60 MIN):

- Slideshow: Introduction and goals of the project (~15 min)
- Interview on multiple results of the algorithm with different settings (each ~5 minutes = 25 minutes)
    - Amplitude, 26-Neighborhood
    - Amplitude, Horizontal Neighborhood
    - Amplitude, Vertical Neighborhood
    - Amplitude + Dip (2D) , 26- Neighborhood
    - Amplitude + Similarity (2D) , 26- Neighborhood
- Discussion and further questions (~15 min)

GUIDING QUESTIONS:

- Ask the following questions for every of the five result files:
    - How does the global visualization support the detection of structures / patterns?
    - Which interesting features are found in the top ranks (e.g. Channel, Fault, Saltdome, ...) ?
    - Does the direction of the arrows in the global visualization provide useful information?
    - Which additional information does the visualization of the anomalies provide?
    - Does the automatic transfer function help to isolate important structures?

- General questions for the final discussion:
    - Are there important structures in the dataset, which have not been detected by the algorithm?
    - How can the algorithm be used in the seismic interpretation workflow?
    - Which tasks could benefit from the algorithm?