Technology
Arts Sciences
TH Köln

Fraunhofer
IAIS

Intuitive Erstellung von Störungsgeometrien in dreidimensionalen seismischen Volumendaten

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Medientechnologie

vorgelegt von
Jan Bender

Erstgutachter: Prof. Dr.-Ing. Arnulph Fuhrmann
Institut für Medien- und Phototechnik, TH Köln

Zweitgutachter: Dr. rer. nat. Stefan Rilling
Fraunhofer IAIS

Köln, im Juni 2016

Bachelorarbeit

Titel: Intuitive Erstellung von Störungsgeometrien in dreidimensionalen seismischen Volumendaten

Gutachter:

- Prof. Dr.-Ing. Arnulph Fuhrmann (Technische Hochschule Köln)
- Dr. rer. nat. Stefan Rilling (Fraunhofer IAIS, St. Augustin)

Zusammenfassung: Seismische Daten spielen in der Energiewirtschaft eine bedeutende Rolle. Unternehmen der Öl- und Gasindustrie oder aus dem Bereich der Geothermie betreiben einen großen finanziellen und organisatorischen Aufwand für die Akquirierung der Daten aus dem Erdinneren. Für den wirtschaftlichen Erfolg dieser Unternehmen ist dabei die Analyse und Interpretation der Daten von enormer Wichtigkeit.

In der vorliegenden Arbeit wurde ein Verfahren, zur Analyse und Interpretation seismischer Volumendaten, entwickelt. Insbesondere liegt dabei der Fokus auf der Erstellung von Geometrien innerhalb der Volumendaten. Mit diesen Geometrien können geologische Experten sog. *seismische Störungen* im Erdinneren analysieren, um Aufschluss über den möglichen Ertrag an Rohstoffen, die diese Störungen liefern könnten, zu erlangen. Das besondere an dem hier entwickelten Verfahren ist, dass es in verschiedenen visuellen Darstellungen der Volumendaten eingesetzt werden kann.

Stichwörter: Volumenrendering, seismische Störungen, Interaktion, Picking, Painting.

Datum: 29. Juni 2016

Bachelors Thesis

Title: Intuitive creation of fault geometry in threedimensional seismic volume data

Reviewers:

- Prof. Dr.-Ing. Arnulph Fuhrmann (Technische Hochschule Köln)
- Dr. rer. nat. Stefan Rilling (Fraunhofer IAIS, St. Augustin)

Abstract: Seismic data plays an important role in the energy industry. Companies of the oil and gas industry or in the field of geothermal energy conduct a large financial and organizational effort for the acquisition of data from the interior of the earth. Therefore, the analysis and interpretation of the data has a great importance for the economic success of these companies.

In the present work, a method for analyzing and interpreting seismic volume data, was developed. In particular, the focus is on the creation of geometries within the volume data. With those geometries, geological experts are able to analyze so called *seismic faults* to get information about the possible return, of this seismic fualt, of raw materials . The special feature of the developed method, is that it can be used in various visual representations of the volume data.

Keywords: volumerendering, seismic faults, interaction, picking, painting.

Date: 29. Juni 2016

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

.....

(Ort, Datum) (Unterschrift)

Danksagung

Hiermit möchte ich mich ganz herzlich bei allen bedanken, die mich bei der Anfertigung dieser Bachelorarbeit unterstützt haben.

Mein Dank geht dabei ganz besonders an Dr. Manfred Bogen und Dr. Stefan Rilling vom Fraunhofer IAIS in Sankt Augustin für die fachliche Hilfe, die ständige Unterstützung und das konstruktive Feedback während der Erstellung dieser Arbeit, sowie an meinen Kollegen Jan Beutgen für die wertvollen Anregungen und die ständige Motivation.

Weiterhin möchte ich mich bei Prof. Dr. Arnulph Fuhrmann für die Betreuung dieser Arbeit seitens der Technischen Hochschule Köln, die hilfreichen Ratschläge und das ehrliche Feedback bedanken.

Nicht zuletzt möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mich immer, und in den letzten Monaten ganz besonders, unterstützt haben.

Inhaltsverzeichnis

1 Einleitung **2**
1.1 Motivation . . . 2
1.2 Ziele . . . 3
1.3 Aufbau der Arbeit . . . 4

2 Grundlagen **5**
2.1 Geologischer Hintergrund . . . 5
2.1.1 Energiegewinnung aus der Erde . . . 5
2.1.2 Datenakquise . . . 7
2.1.3 Seismische Horizonte und Störungen . . . 10
2.1.4 Interpretation von Daten und Störungen . . . 11
2.2 Volumenrendering . . . 14
2.2.1 Funktionsweise . . . 14
2.2.2 Transferfunktionen . . . 18
2.2.3 2D-Slices und direktes Volumenrendering . . . 19
2.3 Die Software OpenSceneGraph . . . 21
2.3.1 Moderne Rendertechniken mit OpenGL . . . 22

3 Anforderungen **24**

4 Stand der Technik **26**
4.1 Techniken für Volumeninteraktion . . . 26
4.2 Techniken in geologischen Softwarepaketen . . . 30
4.3 Bisherige Techniken der SeisViz3D Software . . . 33

5 Schnittpunktberechnung in Volumendaten **35**
5.1 Berechnung auf 2D-Slices . . . 35
5.2 Berechnung im direkten Volumenrendering . . . 37
5.2.1 Konzeption . . . 37
5.2.2 Implementierung . . . 39
5.2.3 Softwarearchitektur . . . 49

6 Vereinfachte Erstellung von Störungsgeometrien **52**
6.1 Idee . . . 52
6.2 Implementierung . . . 52
6.3 Integration in SeisViz3D Software . . . 53

7 Ergebnisse **56**

8 Zusammenfassung und Ausblick **61**

Abbildungsverzeichnis **67**

Literaturverzeichnis **72**

Anhang **74**

1 | Einleitung

1.1 Motivation

In der Informationsindustrie dreht sich alles um Daten. Durch technologischen Fortschritt sind diese Daten in der heutigen Zeit wesentlich größer und komplexer als noch Jahre zuvor. Diese Größe und Komplexität bringen jedoch nicht nur Vorteile mit sich. So ist es schwer, die Daten gekonnt zu strukturieren und zu verarbeiten, damit sie vernünftig analysiert und interpretiert werden können und somit eine Aussagekraft erhalten. Solche Daten werden auch als *Big Data* bezeichnet und sie sind für den wirtschaftlichen Erfolg eines Unternehmens von immer größerer Bedeutung [1].

Big Data fallen in sehr vielen unterschiedlichen Unternehmungs- und Anwendungsbereichen an: Wirtschaft, Marktforschung, Telekommunikation oder auch in der Wissenschaft wie z.B. Kernphysik, Klimaforschung oder Geologie. Meistens handelt es sich bei diese Daten um zuvor gewonnene Messdaten, die dann auf eine geeignete Art und Weise weiterverarbeitet werden müssen. Was „geeignet“ bedeutet, hängt von der Art der Daten ab. In der Wirtschaft werden meistens personenbezogene Daten erhoben diese müssen dann gefiltert, strukturiert und bewertet werden. Bei Messdaten in der Wissenschaft geht es vor allem darum, Zusammenhänge zu erkennen. Dies wird oftmals dadurch erreicht, dass dem Nutzer, der die Daten letztendlich interpretiert und bewertet, ermöglicht wird, die Daten visuell zu betrachten und mit ihnen interagieren zu können.

Die Daten, um die es sich in dieser Arbeit handelt, sind seismische Daten, d.h. sie stellen den Aufbau des Erdinnern dar, welcher zuvor durch akustische Wellen erforscht wurde (vgl. Kapitel 2.1.2). Zur Interpretation können diese Daten auf einem Bildschirm als 3D-Volumen in Echtzeit visualisiert werden (vgl. Kapitel 2.2). Die Analyse der Daten hat meistens das Ziel, mehr über die Beschaffenheit des Erdinneren in Erfahrung zu bringen, die Strukturen der einzelnen Erdschichten genauer zu betrachten und womöglich Hohlräume (sog. Störungen, vgl. Kapitel 2.1.3) zu finden, da sich dort im Laufe der Zeit mit hoher Wahrscheinlichkeit Erdöl, Erdgas oder heißes Wasser gesammelt hat. Diese Rohstoffe gilt es nun zu fördern, da fossile Energiegewinnung immer noch ein wichtiger Faktor in der Energiewirtschaft unserer Zeit ist.

Die Probleme, die sich hier ergeben, sind den enormen Datenmengen geschuldet. Dadurch dass die Störungen meist nur einen kleinen Teil des Gesamtvolumens ausmachen, muss das Erscheinungsbild der Volumendaten je nach Bedarf angepasst werden können. Außerdem müssen die Störungen,

wenn gefunden, auf ihren möglichen Ertrag analysiert werden. Hierfür werden spezielle Interaktionsmöglichkeiten benötigt.

In dem normalen Arbeitsablauf von Geologen und Geophysikern muss eine solche Analyse möglichst schnell, aber dennoch präzise durchgeführt werden. Es stellt sich also u.a. die Frage nach einer Methode, die es ermöglicht, solche Störungen schnell und einfach analysieren zu können.

1.2 Ziele

In dieser Arbeit soll eine Methode entwickelt werden, die den Nutzern der seismischen Daten dabei helfen soll, Störungen zu analysieren. Dabei steht eine einfache Bedienbarkeit und eine flexible Anwendungsmöglichkeit im Vordergrund.

Das Ziel der entwickelten Methode soll es sein, Störungen möglichst schnell und intuitiv analysieren und interpretieren zu können. Außerdem soll die Analyse der Störungen problemlos in verschiedenen visuellen Darstellungen und Auflösungen der seismischen Daten möglich sein.

Es gibt zwei unterschiedlichen Arten der Darstellungen, die häufig für die Interpretation seismischen Daten genutzt werden:

- **2D-Slices Rendering** - Hierbei wird das Volumen in einzelnen Scheiben (engl. *Slices*), auch Sektionen genannt, dargestellt. Die Scheiben können in jeder Achsenrichtung (x, y, z) erstellt werden und das Volumen traversieren. Um das zu ermöglichen wird eine 2D-Geometrie in der Größe des Volumendatensatzes, auf einer vorher definierten Achse, erstellt und eine 2D-Textur auf diese Geometrie gerendert [45].

- **Direktes Volumenrendering** - Bei dieser Methode wird der Volumendatensatz als Ganzes gerendert. Dabei werden mithilfe einiger moderner Rendertechniken die Volumendaten in einer 3D-Textur gespeichert. Zusätzlich wird jedem *Voxel* (Pixel im dreidimensionalen Raum) ein Opazitätswert gegeben, wodurch es möglich ist, mittels Transferfunktionen Teile der Volumendaten auszublenden (vgl. Kapitel 2.2.2) [27].

Beide Arten der Darstellung werden in Kapitel 2.2.3 noch ausführlicher erläutert. Auf die konkreten Anforderungen, die an die zu entwickelnde Methode gestellt werden, wird in Kapitel 3 genauer eingegangen.

1.3 Aufbau der Arbeit

Nachdem die Motivation und das Ziel dieser Arbeit aufgezeigt worden sind, werden in Kapitel 2 einige Grundlagen erklärt, sodass dem Leser ein gewisses Grundverständnis von den Themen, die in dieser Arbeit behandelt werden, vermittelt wird. Zunächst werden die geologischen Grundlagen näher erläutert, da diese Arbeit ausschließlich im Kontext der seismischen Interpretation entstanden ist. Danach werden einige Techniken vorgestellt, die für die technische Umsetzung dieser Arbeit genutzt wurden: Einige moderne Rendertechniken von *OpenGL* [3], die Szenegraphen-Architektur der Software *OpenSceneGraph* [4] und die Technik des Volumenrenderings an sich.

Im Hauptteil der Arbeit wird zu Beginn von Kapitel 3 eine kurze Anforderungsanalyse erstellt und in Kapitel 4 wird dann ein Überblick über den aktuellen Stand der Technik gegeben im Hinblick auf die bereits vorhandenen Interaktionsmöglichkeiten mit Volumendaten. Anschließend wird in Kapitel 5 & 6 die Implementierung der Arbeit und Einbettung in die bestehende Software thematisiert. Dabei wird die Implementierung in zwei Teilpakete aufgeteilt, da im ersten Teilpaket eine Funktionalität geschaffen wird, die notwendig ist, um die finale Methode im zweiten Teilpaket implementieren zu können.

Zum Schluss werden in Kapitel 7 die Ergebnisse dieser Arbeit gezeigt und in Kapitel 8 wird nach einer kurzen Zusammenfassung aufgeführt, welche Möglichkeiten sich durch die neue Methodik bieten.

2 | Grundlagen

In diesem Kapitel werden die Grundlagen im Bereich Geologie, Visualisierung und Software erläutert, welche nötig sind, um den Hauptteil dieser Arbeit nachvollziehen zu können.

2.1 Geologischer Hintergrund

Da sich diese Arbeit mit der Darstellung und Interaktion von und mit seismischen Daten beschäftigt, wird in diesem Unterkapitel zunächst auf die geologischen Hintergründe eingegangen. Die Fragen, warum seismischen Daten so wertvoll sind und wie diese generiert und interpretiert werden, sollten nach diesem Unterkapitel geklärt sein. Außerdem werden wichtige Begriffe erläutert.

2.1.1 Energiegewinnung aus der Erde

Der Grund weshalb überhaupt ein solcher Aufwand um die Visualisierung von seismischen Daten betrieben wird ist der, dass unter der Erdoberfläche wertvolle Ressourcen und Rohstoffe vorhanden sind, die es zu beschaffen gilt. Nun ist das Hauptproblem, diese Vorkommen zunächst zu finden, bevor sie gefördert werden können, und genau hier setzt die seismische Datenvisualisierung und Interpretation ein.

Zwei dieser wertvollen Rohstoffe sind u.a. Öl und Gas. Solche Vorkommen entstehen über tausende von Jahren hinweg, meist durch organisches Material (Tier- oder Pflanzenkadaver), welche einst auf dem Grund des Meeres gelegen haben und mit der Zeit von mehreren Gesteinsschichten überlagert wurden. Diese Gesteinsschichten übten mit der Zeit einen enormen Druck auf die organische Masse aus, wodurch sich die Temperatur erhöhte. Unter der Einwirkung der Kombination von Druck und Hitze auf die organische Schicht entstanden über die Zeit hinweg die wertvollen Rohstoffe Erdöl und Erdgas [57].

Die Gesteinsschichten, in denen diese Vorkommen entstehen, werden *Quellgestein* genannt. Diese liegen aber oftmals so tief, dass es für den Menschen unmöglich ist, sie zu erschließen. Nun ist es aber naturgemäß so, dass die Erdkruste ständig in Bewegung ist. Durch diese Bewegung werden manche Gesteinsschichten manchmal porös oder brechen sogar auseinander. Man spricht in diesem Fall von sog. *Störungen* (vgl. Kapitel 2.1.3). Sind solche Störungen vorhanden oder ist die Gesteinsschicht über dem Quellgestein so porös, dass sie durchlässig geworden ist, sucht sich das Erdöl bzw. Erdgas seinen Weg nach oben, bis eine weitere undurchdringliche Gesteinsschicht

oder ein sog. *Reservoir* erreicht wird [57].
Dieser Vorgang wird auch als *Ölmigration* bezeichnet und wird in Abbildung 2.1 noch einmal bildlich dargestellt.

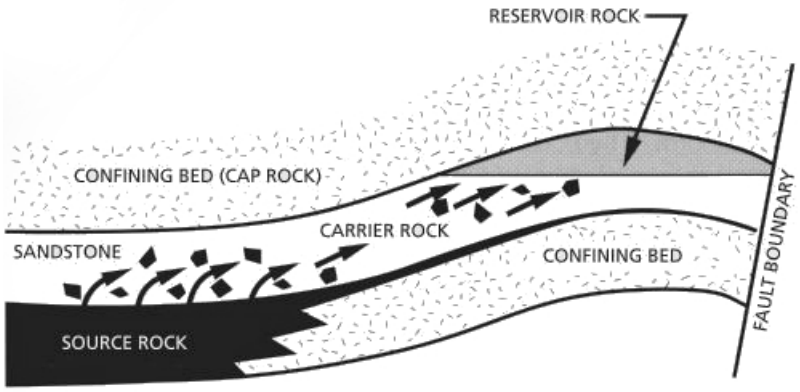

Abbildung 2.1: Ablauf der Ölmigration [57]

Auf der Suche nach Öl- und Gasvorkommen sind deshalb die Stellen in der Erde interessant, an denen Störungen vorkommen, da sich dort Hohlräume im Gestein bilden können, in denen sich dann Öl oder Gas ansammeln kann.

Neben Öl- und Gasvorkommen hält die Erde noch eine weitere wichtige und vor allem zukunftsreiche Ressource bereit: Heißes Wasser. Die sog. *Heißwasseraquiferen* (Gesteinskörper mit Hohlräumen, welche zur Leitung von Grundwasser geeignet sind) werden dabei oftmals zufällig bei der Erschließung einer neuen Störung entdeckt. Hat die Störung nämlich eine große Reichweite in die Tiefe und befindet sich dort Grundwasser, wird die Fluidbewegung auf diese Störungsbahn fokussiert. Da das Grundwasser aufgrund der Erdwärme in größerer Tiefe heißer ist als das Gestein in niedrigerer Tiefe, heizt sich das Gestein auf und macht es für die geothermische Nutzung interessant [41].

Die Aufgabe besteht also darin, Störungen zu finden, dort zu bohren und die Rohstoffe zu fördern. Nun sind solche Bohrungen ein äußerst kostspieliges Unterfangen und zudem sehr schädlich für die Umwelt. Es gilt also, das Risiko einer Fehlbohrung so gering wie möglich zu halten. Deshalb wird vor der Bohrung eine Analyse des Erdinneren von dem Gebiet gemacht, wo Vorkommen vermutet werden, um an Informationen über möglich vorhandene Störungen zu kommen und somit das Risiko auf eine Fehlbohrung zu minimieren [57].

2.1.2 Datenakquise

Die Analyse des Erdinneren erfolgt mittels Schallwellen. So wie Ultraschall dazu verwendet wird, um Formen und Umrisse vom Inneren des menschlichen Körpers darzustellen, werden sog. *seismische Wellen* dazu verwendet, um die geologischen Strukturen des Erdreiches zu visualisieren. Wo jedoch bei dem Ultraschallverfahren sehr hoch frequentierten Schall (kurze Wellenlänge) benutzt wird, um einige Zentimeter tief in den Körper einzudringen, wird bei der seismischen Exploration niederfrequenter Schall verwendet (lange Wellenlänge), um bis zu 10 Kilometer unter die Erdkruste zu gelangen [57].

Ziel einer solchen Analyse oder Messung ist es, ein möglichst genaues Modell von dem Bereich zu erstellen, in dem Störungen vermutet werden, um so die besten Bohrpositionen herauszufinden. Diese Bereiche können auf dem Land sein, oftmals sind sie jedoch auch mitten im Meer. Die Messverfahren für beide Bereiche unterscheiden sich jedoch nicht grundlegend voneinander. In Abbildung 2.2 & 2.3 werden beide Verfahren visuell verdeutlicht.

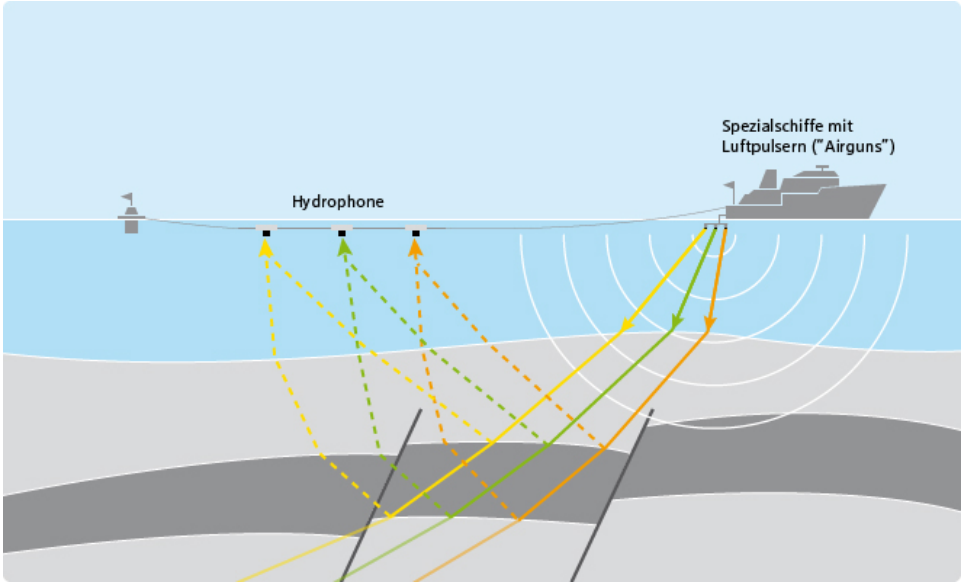

Abbildung 2.2: Aufnahme von seismischen Daten auf dem Wasser [6]

Auf dem Wasser werden spezielle Schiffe eingesetzt, welche mit sog. *airguns* ausgerüstet sind. Diese sind durch Druckluftimpulse für die Erzeugung der Schallwellen verantwortlich. Das Ganze kann in etwa mit einem Steinwurf ins Wasser verglichen werden: Dort breiten sich die Wellen kreisförmig aus und werden zurückgeworfen, wenn sie auf ein Hindernis treffen. Bei der

seismischen Exploration geschieht dasselbe mit Schallwellen. Abhängig von der physikalischen Eigenschaft der Erdschichten wird ein Teil der Schallwellen reflektiert und ein anderer Teil gebrochen, welcher weiter in tiefere Erdschichten vordringt. Die Ausbreitungsgeschwindigkeit und Stärke der reflektierten und gebrochenen Strahlen hängt dabei von der Beschaffenheit der Erdschicht ab. Anhand dieser beiden Parameter wird der Reflexionskoeffizient bestimmt [6] [57].

Die reflektierten Strahlen werden an der Oberfläche von Hydrophonen (Aufnahmegerät), die hinter dem Schiff in regelmäßigem Abstand hergezogen werden, aufgenommen, um später analysiert und weiterverarbeitet zu werden. Die ausschlaggebenden Parameter sind hier zum einen der Reflexionskoeffizient, der die Information über die Charakteristik der Erdschicht bereithält und der zeitliche Abstand der reflektierten Wellen zueinander, mit dessen Hilfe sich eine Aussage über die Tiefe der Erdschicht machen lässt.

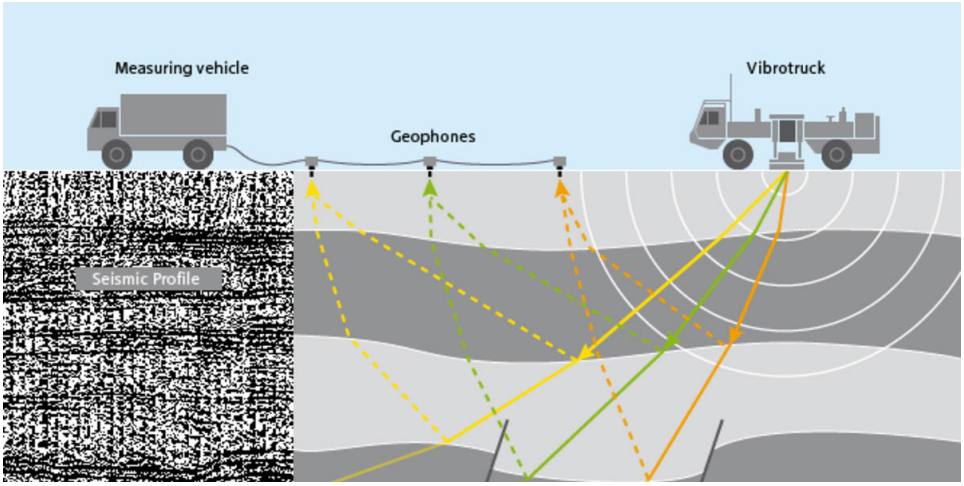

Abbildung 2.3: Aufnahme von seismischen Daten auf dem Land [6]

An Land unterscheidet sich die Methodik der seismischen Exploration nicht grundlegend von der auf dem Wasser. Statt Schiffen werden hier Lastkraftwagen eingesetzt. Der Schall wird hier nicht mittels Druckluftimpulsen, sondern durch vibrierende Masse erzeugt, welche von einem sog. *Vibro-truck* ausgesandt wird [6]. Diesem LKW fährt ein zweiter voraus, der die Messinstrumente (in diesem Fall keine Hydrophone sondern Geophone) hinter sich herzieht. Das Aufnahme- und Analyseverfahren unterscheidet sich sonst nicht von dem bereits oben beschriebenem.

Anhand der Daten, die von den Hydro- bzw. Geophonen gemessen wurden, können durch den Reflexionskoeffizienten und die Distanz, die der Dauer eines jeden Impulses zwischen Aussenden und Eintreffen auf das Aufnahmegerät zugrunde liegt, ein Amplitudenwert errechnet werden, der eindeutig einer Erdschicht zugewiesen werden kann und diese in ihrer Beschaffenheit und Lage beschreibt [39].

Nach der Messung liegen die Daten in einer Rohfassung vor und sie müssen vor der Weiterverarbeitung noch bereinigt werden. Bei dieser Bereinigung werden Störfaktoren wie z.B. unnatürliche Reflexionen (*ghosting*), die einem rauen Seegang und der damit einhergehenden unruhigen Wasseroberfläche geschuldet sind, entfernt. Anschließend werden die Daten zu einer sog. Seismik zusammengefasst [57].

Die Zusammenfassung der Daten hängt von der Art der Anordnung der Aufnahmegeräte ab. Sind diese in einer Linie angeordnet, so wie es Abbildung 2.2 & 2.3 zunächst vermuten lassen, wird ein 2D-Schnittbild (auch *Slice* oder Sektion) der Erdschichten, unmittelbar unterhalb der Linie von Aufnahmegeräten, erzeugt.

Eine effizientere Methode der Messung, die ebenfalls viel mehr Informationen bietet, ist es die Aufnahmegeräte gitterförmig, in einem *Array*, anzuordnen (s. Abbildung 2.4).

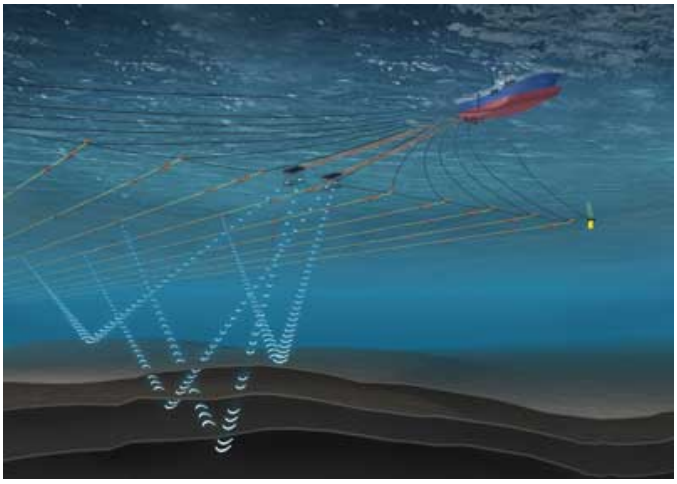

Abbildung 2.4: Aufnahme von seismischen Daten mit Hydrophon Array [33]

Durch dieses Aufnahmeverfahren werden mehrere 2D-Schnittbilder aneinandergereiht, wodurch die Informationsdichte enorm wächst [57]. Bei der, durch diese Messung entstandenen Seismik kann auch von einem dreidimensionalen Datensatz bzw. *Volumen* gesprochen werden.

Dieses Volumen hat die Größe (x, y, z), wobei x und y das Ausmaß der vermessene Oberfläche beschreiben und z die Ausdehnung in die Tiefe. In der seismischen Interpretation werden diese Ausrichtungen auch als *inline*, *crossline* und *timeline* bezeichnet, sodass gilt: $(x, y, z) \rightarrow (i, c, t)$.

2.1.3 Seismische Horizonte und Störungen

Horizonte und *Störungen* sind wichtige Begriffe in der Öl- und Gasindustrie und essentiell für die vorliegende Arbeit.

Der Begriff Horizont kommt aus dem altgriechischen von dem Wort *horízein* und bedeutet „begrenzen“ . Normalerweise ist damit die Grenzlinie zwischen Himmel und Erde gemeint [17].

Der „seismischer Horizont“ beschreibt dagegen eine Gesteinsoberfläche im Erdinneren, an der eine seismische Reflexion (vgl. Kapitel 2.1.2) stattfindet. Dies ist immer genau dann der Fall, wenn eine Gesteinsschicht in eine andere über geht und sich deshalb die physikalischen Eigenschaften der Schichten verändern (Dichte, Porosität etc.), wodurch die seimsichen Wellen reflektiert bzw. gebrochen werden. [47].

Der Begriff „seismische Störung“ oder auch Verwerfung genannt, ist nun bereits des Öfteren gefallen, nichts desto trotz soll hier noch einmal eine kurze Definition vorgestellt werden, da sich ein Großteil der seismischen Interpretation mit der Exploration und Deutung von seismischen Störungen beschäftigt und sich auch diese Arbeit mit einem Verfahren für eine möglichst einfache und schnelle Interpretation auseinandersetzt.

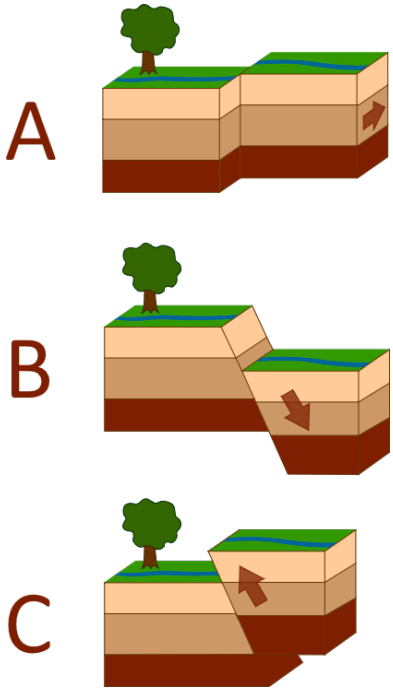

Abbildung 2.5: Oben: Blattverschiebung, Mitte: Abschiebung, Unten: Aufschiebung [24]

Störungen sind tektonische Bruchstellen im Gestein [57] und werden *Bruchflügel*, *Blöcke* oder *Schollen* genannt. Diese entstehen durch Druck- oder Zugkräfte, resultierend aus plattentektonischen Vorgängen und können sich zwischen einigen Zentimetern und mehreren Kilometern ausdehnen. Sie befinden sich meist in den oberen, noch relativ kühlen Bereichen der Erdkruste, da dort das Gestein spröde ist und leichter bricht. Es gibt unterschiedliche Arten von Störungen, die sich in der Richtung ihrer Ausbreitung unterscheiden. Horizontal gelegene Störungen werden *Blattverschiebung* genannt, während in horizontaler Richtung die Störung entweder *Auf-* oder *Abschiebung* genannt wird, je nachdem, ob sich die Erdkruste staucht und sich die Scholle anhebt oder ausdehnt und sich die Scholle absenkt [12]. In Abbildung 2.5 werden die verschiedenen Arten der Störung dargestellt.

In der Realität ist es meistens so, dass nicht eine einzige Störungsart vorhanden ist, sondern eine Mischform. Dadurch ist es möglich, dass in manchen Fällen große Hohlräume im Erdinneren entstehen, in denen sich dann wiederum Rohstoffe aus tieferen Bereichen der Erde sammeln können (vgl. Kapitel 2.1.1). Aus diesem Grund ist eine vernünftige Interpretation der Störungen so wichtig, um den Verlauf, die Größe und die Gesteinsschichten in näherer Umgebung zu bestimmen und so auf mögliche Vorkommen schließen zu können [57].

2.1.4 Interpretation von Daten und Störungen

Nachdem die Daten gesammelt und bereinigt worden sind, ist der nächste Schritt, diese zu interpretieren. Die Aufgabe übernehmen Geologen und Geophysiker, welche im Folgenden auch als Experten bezeichnet werden.

Bisher liegen die Daten in einer Rohfassung vor, die nur die Skalarwerte der gemessenen Amplituden und Positionen enthält. In dieser Form können die Daten praktisch nicht ausgewertet werden, da sich allein anhand der gemessenen Zahlenwerte noch keine Aussagen treffen lassen. Um eine umfassende Auswertung von seismischen Daten vornehmen zu können, bedarf es zweier Komponenten:

Zum einen müssen die Daten in geeigneter Form, d.h. inhaltlich passend, visualisiert werden, zum anderen müssen Interaktionsmöglichkeiten mit der visuellen Darstellung der Daten geschaffen werden, die den Experten eine korrekte Interpretation ermöglichen.

Diese Aufgaben werden von modernen seismischen Interpretationsprogramme realisiert. Oftmals haben diese Programme eine Vielzahl von Möglichkeiten, was Interaktion und Visualisierung anbelangt.

Gängige Interaktionen, die Experten für die seismische Interpretation benötigen, sind z.B. das Auswählen von wichtigen Elementen innerhalb der visualisierten Daten, das Setzen von Markierungen in den Daten, um sich charakteristische Stellen zu merken, oder das Einzeichnen von beispielsweise einem Störungsverlauf [57]. Die Methoden „Auswählen“ und „Setzen“ werden im Folgenden als *picking* [55] und das „Einzeichnen“ als *painting* [38] bezeichnet.

Speziell bei der Interpretation von Störungen hat es sich bewährt, sog. *Störungsgeometrien* zu erstellen. Diese sollen in etwa den räumlichen Verlauf der Störung nachempfinden, damit so eine Aussage über den möglichen Ertrag dieser Störung gemacht werden kann. Die Geometrien werden dabei anhand von *Störungslinien* erstellt, wie in Abbildung 2.6 zu sehen ist. Um eine solche Linie zu erstellen, setzt der Experte Punkte in eine Sektion, wo er einen Störungsverlauf vermutet. Diese Punkte werden dann zu einer Linie verbunden. Anschließend wird die Sektion durch das Volumen bewegt und in regelmäßigen Abständen wird das Verfahren wiederholt. Die Linien werden dann miteinander verbunden bzw. trianguliert [21], sodass eine Fläche, die Störungsgeometrie, entsteht [31].

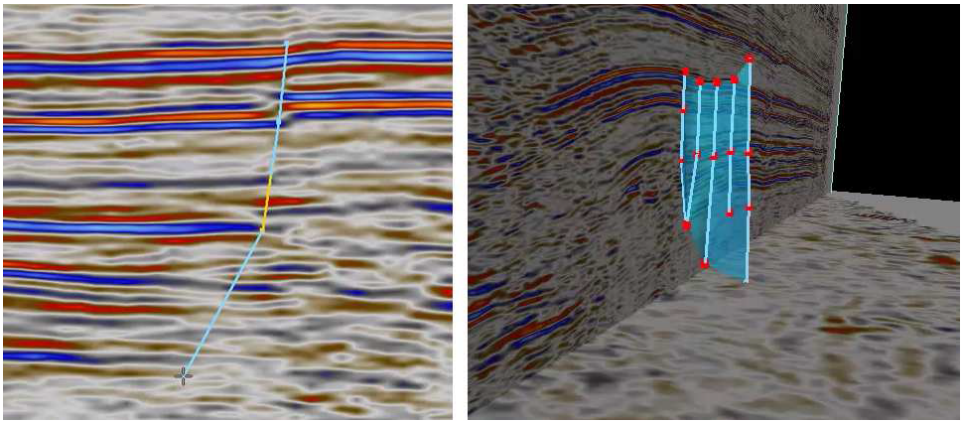

Abbildung 2.6: Links: Störungslinie in einer 2D-Sektion; Rechts: Störungsgeometrie bestehend aus mehreren Störungslinien. Entnommen aus [31]

Die Interpretation von Störungen ist mit diesem Verfahren nur über die Darstellung des Volumens mittels 2D-Schnittbildern möglich, da das Setzen der Punkte im dreidimensionalen Volumendatensatz nicht trivial ist. Die jeweilige Möglichkeit der Interaktion ist also stark abhängig von der Art der Visualisierung der Daten [58] und kann je nach Darstellung sehr komplex und schwierig anzuwenden sein. Wie es dennoch möglich ist, die beiden Methoden *picking* und *painting* in anderen Visualisierungen zu verwenden,

wird in Kapitel 4.1 beschreiben, da diese Methoden essentiell wichtig für die intuitive Erstellung von Störungsgeometrien sind.

Doch was genau sind eigentlich diese 2D-Schnittbilder oder Sektionen und dreidimensionale Volumendaten?

Wie bereits in Kapitel 2.1.2 erwähnt, gibt es unterschiedliche Arten, seismische Daten zu messen und genau so, wie bei diesen Messverfahren entweder ein 2D-Schnittbild oder ein dreidimensionaler Datensatz zustande kommt, so gibt es auch in der Visualisierung die Option seismische Volumendaten entweder mittels 2D-Schnittbildern oder durch direktes Volumenrendering das komplette dreidimensionale *Volumen* darzustellen.

Was genau darunter zu verstehen ist, wird in Abbildung 2.7 zum besseren Verständnis noch einmal visualisiert.

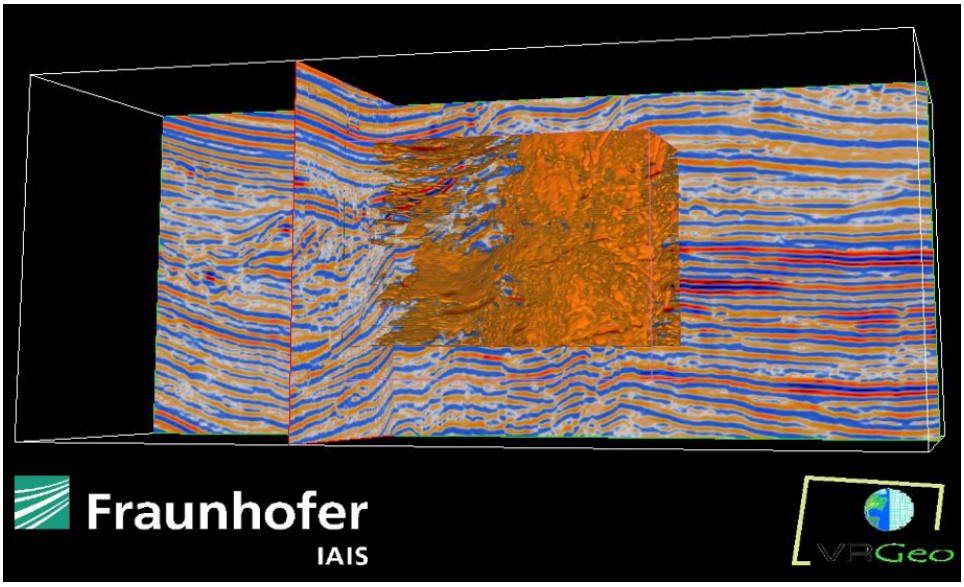

Abbildung 2.7: Darstellung von Volumendaten mittels 2D-Slices (inline und crossline). In der Mitte sind die Daten als Volumen mittels direktem Volumenrendering dargestellt [11]

In Kapitel 2.2.3 werden *2D-Slices* und das *direkte Volumenrendering* noch genauer betrachtet, sowie Vor- und Nachteile aufgezeigt. Doch um diese besser nachvollziehen zu können, wird im folgenden Kapitel zunächst die allgemeine Technik der Darstellung von Volumendaten erläutert.

2.2 Volumenrendering

Das folgenden Kapitel beschreibt die Technik des Volumenrenderings. Diese Technik ist ein Oberbegriff für das weite Feld der Visualisierung von dreidimensionalen Skalardaten und wird in unterschiedlichen Bereichen eingesetzt. Sie wird u.a. für Strömungssimulationen benötigt, die in Bereichen der Automobil- und Flugindustrie zur Visualisierung von Luftströmungen zum Einsatz kommen oder in der Wettersimulation, für die Erstellung von Klimamodellen [43]. Ein weiteres Gebiet ist der Einsatz in der Medizin, wenn mittels der Computertomographie (CT) ein 3D-Scan von Teilen des menschlichen Körpers erstellt wird. Aber auch in der Geologie oder Mathematik wird die Technik verwendet, um große Datenmengen übersichtlich darzustellen [27].

2.2.1 Funktionsweise

Die dreidimensionalen Daten, welche es zu visualisieren gilt, werden zunächst in kleine Blöcke rasterisiert, wodurch ein dreidimensionales Gitternetz entsteht. Dieses Netz ist vergleichbar mit der Anordnung der *Pixel* auf einem Monitor. Wegen der zusätzlichen dritten Dimension wird hier nicht mehr von Pixeln gesprochen; stattdessen werden die gerasterten Blöcke als *Voxel* (volumetric pixel) bezeichnet (s. Abbildung 2.8) [18].

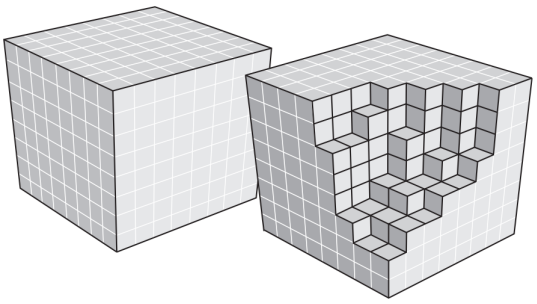

Abbildung 2.8: Ein diskreter Räumlicher Datensatz nach Rasterisierung. Links ist der vollständige Datensatz dargestellt. Rechts ist zum besseren Verständnis der Datensatz nur zum Teil dargestellt um die Idee zu visualisieren, die einen Voxels als Würfel beschreibt, der eine Einheit im dreidimensionalen Raum repräsentiert. Entnommen aus [18]

Beim Volumenrendering wird nun versucht, jedem Voxel ein korrektes Lichtverhalten zuzuordnen. Dabei kommen lichttechnische Größen wie Emission, Absorption, Streuung und Abschattung zum Einsatz. Da die realistische Berechnung des Lichtverhaltens für jeden Punkt sehr rechenaufwendig ist,

werden zur Vereinfachung verschiedene optische Modelle verwendet. Das am häufigsten genutzte Modell ist das sog. *Absorptions-Emissions Modell* [18].

Die Idee hinter diesem Modell ist, dass jedem Skalarwert ein Absorptions- und Emissionskoeffizient zugewiesen wird. Während des Renderings wird ein Strahl $\vec{x}(t)$ vom Augpunkt aus in die Szene geschossen, welcher durch die Distanz zum Auge t parametrisiert wird. Der Emissionskoeffizient $c(t)$ und Absorptionskoeffizient $\kappa(t)$ werden beide anhand des Skalarwertes $s(\vec{x}(t))$ des aktuellen Punktes, welcher von dem Strahl getroffen bzw. durchlaufen wird, festgelegt. Es gilt also:

$$c(t) := c(s(\vec{x}(t))) \qquad \text{und} \qquad \kappa(t) := \kappa(s(\vec{x}(t))) \tag{2.1}$$

In Abbildung 2.9 ist die Grundidee des Modells noch einmal bildlich dargestellt. Das emittierte Licht eines Punktes wird über die Distanz $d = t$ zum Auge hin immer weiter absorbiert, sodass am Ende nur noch ein Teil des emittierten Lichts das Auge erreicht. Die Berechnung dieses Teils erfolgt über die Integration der Absorption in den Grenzen 0 bis d.

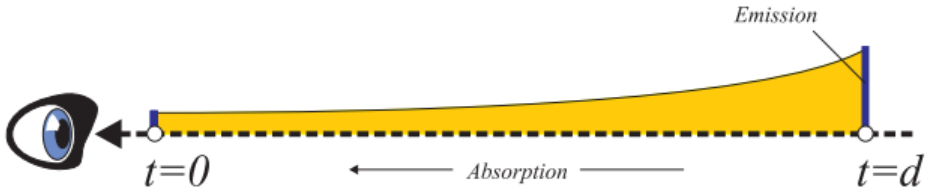

Abbildung 2.9: Absorption des Punktes an der Stelle $t = d$. Entnommen aus [27]

Abbildung 2.9 zeigt allerdings nur, wie die Emission eines Punktes an einer bestimmten Stelle des Strahls berechnet wird. Um die Emission von allen Punkten, die auf diesem Strahl liegen, zu berechnen, muss zusätzlich obiges Integral nochmal über die Distanz $d = t$ integriert werden, damit die Emission jedes Punktes auf dem Strahl berücksichtigt wird und am Ende der richtige Wert am Auge ankommt. Diese Berechnung führt zu dem sog. *Volumenrendering-Integral*:

$$C = \int_0^n c(t) \cdot e^{-\int_0^t \kappa(\hat{t})\, d\hat{t}} dt \tag{2.2}$$

Dabei ist C das Licht, das auf den Augpunkt trifft, und n die Anzahl der Abtastpunkte des Strahls (auch *Samples* genannt).

Da die Lösung der beiden Integrale sehr rechenintensiv ist, wird dieses Verfahren approximiert und eine Diskretisierung des Integrals vorgenommen.

Es gibt unterschiedliche Methoden, wie diese Approximation durchgeführt wird; die häufigste ist das sog. *Front-to-Back Compositing* [27]. Dabei wird die finale Farbe durch das Aufaddieren der Farb- und Opazitätswerte der vorangegangenen Voxel auf dem Strahl bestimmt. Das Wichtige hierbei ist, dass die Berechnung der Farbe an dem Punkt beginnt, wo der Strahl den Augpunkt verlässt und nach hinten fortgeführt wird. Die Gleichung lautet folgendermaßen:

$$C'_i = C'_{i-1} + (1 - A'_{i-1})C_i$$

$$A'_i = A'_{i-1} + (1 - A'_{i-1})A_i \tag{2.3}$$

Dabei entsprechen C'_i und A'_i dem Farb- bzw. Opazitätswert des aktuellen Voxels. C'_{i-1} und A'_{i-1} sind die Werte des vorherigen Voxels auf dem Strahl. Die Anfangsbedingung lautet $C_0 = 0$ und $A_0 = 0$. Die Farbe bestimmt sich also aus dem aktuellen Farbwert des Voxels, gemischt mit den vorherigen Farbwerten, die durch das gleichzeitige Aufsummieren der Opazitätswerte angepasst wurden [27]. Das Verfahren ist nochmal in Abbildung 2.10 verdeutlicht.

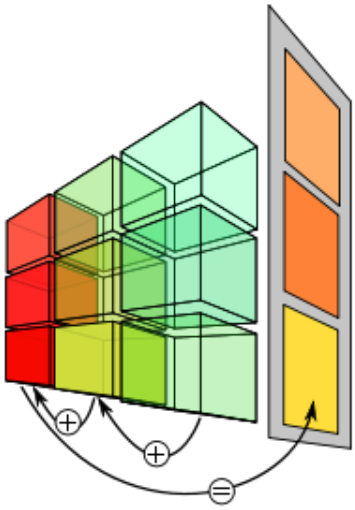

Abbildung 2.10: Front-to-Back Compositing - Die finale Farbe wird von vorne nach hinten berechnet [2]

Ein großer Vorteil dieser Methode ist die sog. *Early-Ray Termination* [18]. Diese Technik bietet eine extreme Beschleunigung für die obige Berechnung, da durch die Reihenfolge des *Front-to-Back Compositings* die Opazitätswerte der bereits betrachteten Voxel aufaddiert werden können. Erreicht diese

Summe einen Wert ≥ 1 wird der Strahl abgebrochen, da eine Opazität größer als 1 undurchlässig ist und somit alle weiteren Punkte auf dem Strahl nicht mehr gesehen werden können, was weitere Berechnungen redundant macht.

Ein Verfahren, mit dem sich die oben beschriebene Rechnung realisieren lässt, ist das *Raycasting* [18]. Bei diesem Verfahren werden von jedem Pixel aus Strahlen in die Szene geschickt. Diese Strahlen werden in bestimmten Abständen (*Sample Rate*) abgetastet. Bei jeder Abtastung wird der Skalarwert der acht benachbarten Pixel betrachtet und mittels *trilinearer Interpolation* dem aktuellen Voxel zugewiesen. Anhand dieses Skalarwertes und der aktuell ausgewählten *Transferfunktion* (vgl. Kapitel 2.2.2) wird dem Voxel dann ein passender RGBA-Wert (Farbkanäle: Rot, Grün, Blau und Opazität: Alpha) zugewiesen. Das Raycasting-Prinzip wird in Abbildung 2.11 verdeutlicht.

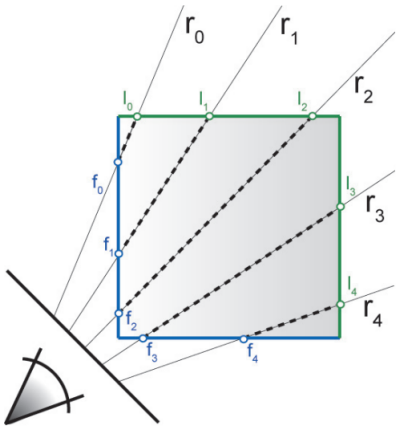

Abbildung 2.11: Raycasting-Verfahren. Entnommen aus [27]

Ein großer Vorteil dieses Verfahrens ist, dass die gesamte Berechnung auf der GPU durchgeführt werden kann. Der Grund dafür ist, dass die Strahlen unabhängig voneinander sind und die Berechnung deshalb parallel stattfinden kann. Die GPU bietet die Möglichkeit der Parallelisierung; dabei werden die Daten als 3D-Textur in den Speicher der GPU geladen, wo der Zugriff auf die benötigten Werte dann parallel stattfinden kann [18].

2.2.2 Transferfunktionen

Wie im vorangegangen Kapitel erwähnt erhalten die Voxel ihre Farb- und Opazitätswerte, welche für das Front-to-Back Compositing benötigt werden, anhand ihres Skalarwertes und der angewandten Transferfunktion. Im Folgenden wird beschrieben, was Transferfunktionen sind und wozu diese benötigt werden. Man unterscheidet dabei zwischen ein- und mehrdimensionalen Transferfunktionen.

Transferfunktionen lassen sich grob zwei Aufgaben in der Darstellung von Volumendaten zuweisen: Im ersten Schritt sind sie dafür zuständig Materialien zu *klassifizieren* [45]. Diese Klassifikation findet im Falle von seismischen Daten anhand der gemessenen Amplituden statt. Hohen Amplituden, die besagen, dass ein harter Übergang der Gesteinsschichten stattfindet (lockere Gesteinsschicht in feste Schicht) wird ein anderes Material zugewiesen als niedrigen Amplituden, die einen weicheren Übergang und demnach andere Gesteinsschichten beschreiben [19]. Im zweiten Schritt weisen sie den zuvor klassifizierten Materialien optische Eigenschaften zu, d.h. RGB-Farbwerte und den Opazitätswert α [45].

Die Hauptaufgabe von Transferfunktionen ist es, zu entscheiden, welche Daten aus dem gesamten Volumendatensatz sichtbar sind und welche nicht. Dadurch können bestimmte Strukturen des Datensatzes bzw. Regionen, die von großem Interesse sind, hervorgehoben und von den restlichen Daten abgegrenzt werden [45]. In der Geologie können dadurch z.B. einzelne Horizonte herausgestellt oder in der Medizin Knochen oder Organe dargestellt werden, indem die restlichen Daten um die interessanten Regionen herum ausgeblendet werden. Das Erstellen einer passenden Transferfunktion ist eine der häufigsten Interaktionstechniken im Volumenrendering [45].

Für die Erstellung gibt es verschiedene Arten von Editoren. Bei eindimensionalen Transferfunktionen hat sich der Ansatz eines Editors bewährt, der aus zwei Achsen besteht. Auf der x-Achse ist das gesamte Farbspektrum abgebildet und auf der y-Achse die Opazität von $0-1$. Über das Setzen von Punkten wird in dem Editor eine Kurve erstellt, die je nach Opazitätswert die Voxel mit der betreffende Farbe ausblendet oder beibehält. Ein solcher Editor mit dem angepassten Volumenrendering ist in Abbildung 2.12 dargestellt; hier sind die negativen (links) und positiven (rechts) Amplituden maximal sichtbar, während die Nullamplitude zum Großteil ausgeblendet ist (deshalb sind diese im Rendering nur ganz schwach und trüb sichtbar).

Das Anwenden und Finden einer passenden Transferfunktion ist ein zeitlich sehr aufwendiges Unterfangen und wird noch aufwendiger, wenn mehrdimensionale Transferfunktionen auf das Volumen angewandt werden [46]. Der Un-

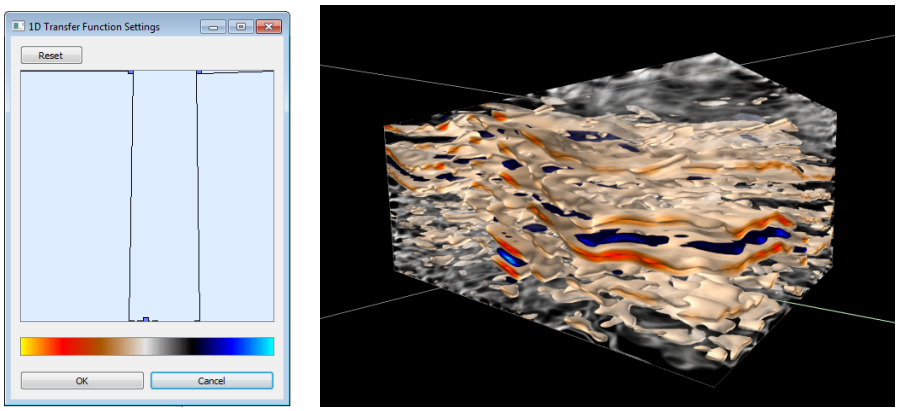

Abbildung 2.12: Links: Editor zur Erstellung von eindimensionalen Transferfunktionen; Rechts: Adaptiertes Volumenrendering

terschied von mehrdimensionalen Transferfunktionen zu eindimensionalen liegt darin, dass Daten, welche dieselbe Amplitude haben, einzeln voneinander isoliert werden können. Dies ist nämlich oftmals ein großes Problem bei eindimensionalen Transferfunktionen. Unterschiedliche Strukturen im Volumen können manchmal dieselben Amplituden haben und bei der Anwendung einer eindimensionalen Transferfunktion ausgeblendet werden, obwohl dies vom Experten für eine der beiden Strukturen nicht gewollt war. Dieses Problem lösen mehrdimensionale Transferfunktionen [45]. Diese Funktionen fügen weitere Eingabewerte hinzu, welche die Sichtbarkeit jedes Voxels beeinflussen. Mit jedem hinzukommenden Wert bzw. jeder hinzukommenden Dimension steigt die Wahrscheinlichkeit, dass sich Strukturen mit gleicher Amplitude voneinander unterscheiden und somit voneinander isoliert werden können [18]. Editoren, mit denen sich mehrdimensionale Transferfunktionen erstellen und verändern lassen, sind wesentlich komplexer, da für jede Dimension die Sichtbarkeit der Amplitudenwerte einzeln eingestellt werden muss. Ein vielversprechender Ansatz ist die Visualisierung mehrerer Dimensionen mittels Paralleler Koordinaten (s. Abbildung 2.13) [8].

2.2.3 2D-Slices und direktes Volumenrendering

Wie nun bereits mehrfach erwähnt, gibt es verschiedene Möglichkeiten, Volumendaten anzeigen zu lassen. Die wohl am häufigst benutzten Methoden sind die Darstellung des Volumens in 2D-Slices oder als dreidimensionalen Datensatz, meistens in Kombination mit einer angewandten Transferfunktion. Wie das direkte Rendering des dreidimensionalen Datensatzes (im Folgenden auch dVR genannt) technischen realisiert wird, sollte an dieser Stelle

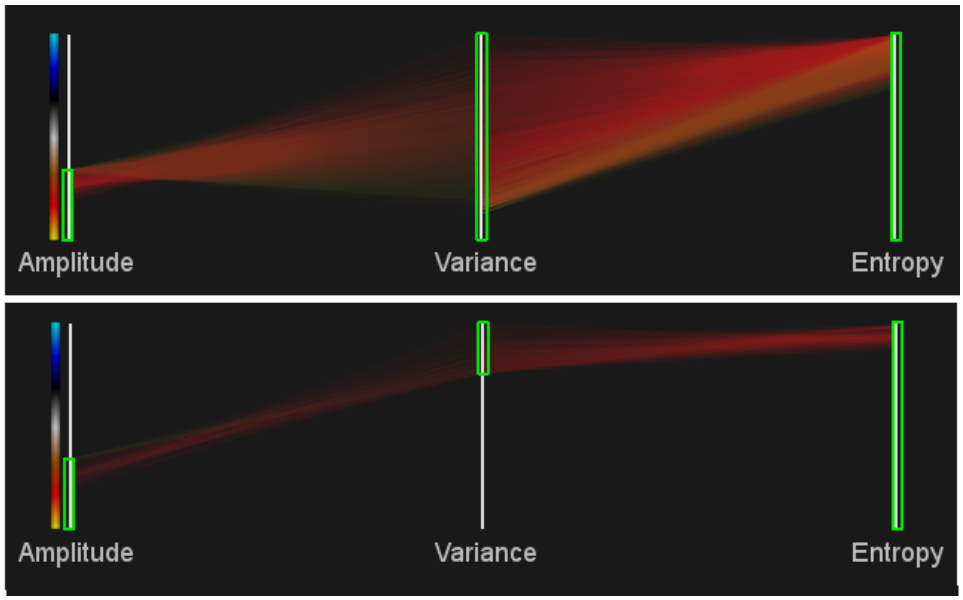

Abbildung 2.13: Editor für mehrdimensionale Transferfunktionen mittels Paralleler Koordinaten. Entnommen aus [8]

soweit verstanden worden sein. Die Darstellung der Daten als 2D-Slices ist wesentlich unkomplizierter. Hierbei wird eine *quad-Geometrie* an einer beliebigen Stelle im Datensatz erstellt und die passende 2D-Textur aus derselben Stelle im Datensatz auf die Geometrie „gemappt“. Da sich diese Arbeit um die Interaktion mit beiden Arten der Darstellung beschäftigt, wird im Folgenden ein kurzer Vergleich vorgenommen.

Mit dem bisherigen Wissen könnte angenommen werden, dass die Darstellung von Volumendaten mittels direktem Volumenrendering immer die bessere Alternative sei, weil der Informationsgehalt gegenüber 2D-Slices wesentlich höher ist. In der Praxis ist es allerdings so, dass viele Experten, egal ob in der Medizin oder Geologie, häufiger die slice-basierte Darstellung nutzen [45] [55] [58]. Der Grund dafür ist meistens, dass es im direkten Volumenrendering zu wenig bzw. zu ungenaue Interaktionsmöglichkeiten gibt [58]. In komplexen Datensätzen haben Experten im dVR oft das Problem, die Daten zu finden, die sie suchen, und nach dem Finden geeignet zu interpretieren, da es ihnen an Möglichkeiten mangelt mit diesen zu interagieren [44]. Für 2D-Slices ist es wesentlich einfacher, präzisere Interaktionstechniken zu entwickeln. Zudem eignen sich besser, um sehr feine Details zu erkennen, während Daten im dVR besser dazu geeignet sind, um Strukturen in Relation zu anderen Daten zu erkennen und dem Experten eine Übersicht zu geben [55]. Dies wird in Abbildung 2.14 verdeutlicht.

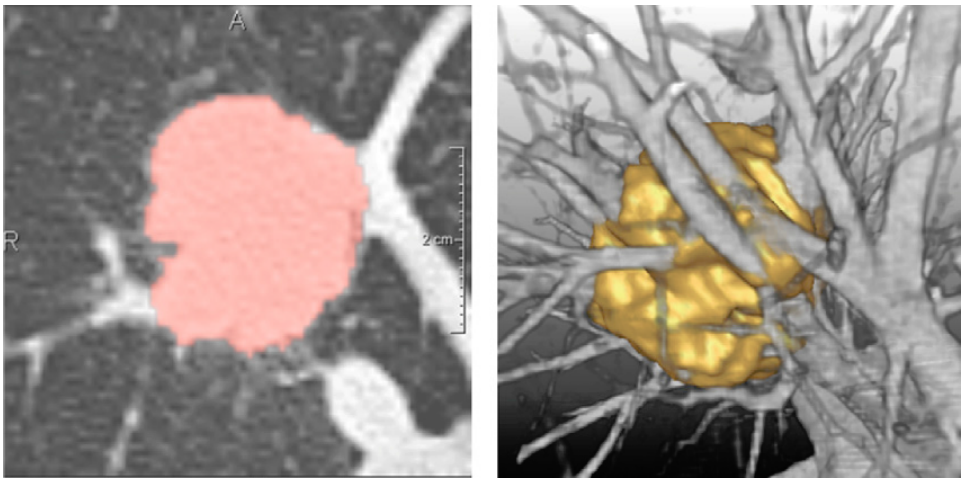

Abbildung 2.14: Links: Die Umrisse eines Lungenknotens sind auf der 2D-Slice klar zu erkennen und lassen sich vom Rest abgrenzen. Rechts: Im direkten Volumenrendering lässt sich besser die Verwachsung desselben Knotens mit dem Bronchialsystem abschätzen. Entnommen aus [45]

Mit immer leistungsstärkerer Hardware und fortschreitender Technik wie z.B. das Visualisieren des direkten Volumenrenderings in 3D Stereo [53] bietet das dVR den immer stärker werdenden Anreiz für Experten, diese Technik häufiger und ausgiebiger zu nutzen.

Da dies ein sehr aktuelles Problem ist, gilt es hier anzusetzen und die Möglichkeiten des dVR für Experten zu erweitern. Mit diesem Problem beschäftigt sich auch die vorliegende Arbeit; insbesondere liegt hier der Hauptaugenmerk auf die noch fehlenden bzw. zu wenig vorhandenen Interaktionsmöglichkeiten. In Kapitel 3 wird darauf weiter Bezug genommen.

2.3 Die Software OpenSceneGraph

Bisher wurde nur über die unterschiedlichen Arten der Darstellung von Volumendaten gesprochen und nicht, wie dies auf Programmier-Ebene umgesetzt wird. Im Falle dieser Arbeit läuft die software-seitige Realisierung über ein 3D Grafik Framework namens *OpenSceneGraph (OSG)* [4].

OSG ist vollständig in Standard *C++* geschrieben und nutzt für die grafische Darstellung die moderne Renderbibliothek *OpenGL* [3]. OSG stellt eine API zur Verfügung, die es ermöglicht, eine grafische Anwendung mittels *Szenengraph* zu programmieren. Ein Szenengraph beschreibt dabei die Struktur einer Szene und die Beziehungen einzelner Elmente zueinander. Jedes Element der Szene wird dabei in dem Graphen als *Knoten* bzw. *node*

definiert. Typischerweise ist so ein Graph hierarchisch aufgebaut, beginnend mit dem sog. *root-node*. Danach folgen eine Reihe von *group-nodes*, die alle mehrere *child-nodes* haben können und in *leaf-nodes* enden, welche keine weiteren „Kinder“ haben und das Ende der jeweiligen Verzweigung des Graphen repräsentieren [54]. Das Besondere an den Beziehungen der einzelnen *nodes* zueinander ist, dass die Kindknoten eines Parent-Knoten als eine Einheit betrachtet werden, was bedeutet, dass eine Transformation (z.B. Translation oder Rotation), die auf einen Parent-Knoten angewandt wird, einen kaskadierenden Effekt hat und automatisch auch auf die Kindknoten übergeht. Ein solches Verhalten wird auch als *Transformationshierarchie* bezeichnet [34].

Die Visualisierung der Szeneelemente in Echtzeit durch die Verwendung von OpenGL ermöglicht den Zugriff auf viele moderne Rendertechniken, die zum Teil in dieser Arbeit ebenfalls ihre Verwendung gefunden haben, und im folgenden Kapitel kurz näher erläutert werden.

2.3.1 Moderne Rendertechniken mit OpenGL

Für die Implementierung der in dieser Arbeit vorgestellten Technik ist es notwendig, dass einige Rendertechniken von OpenGL in OSG genutzt werden. Die wichtigste davon sind die sog. *Framebuffer Objects* kurz *FBOs*. Mit FBOs lassen sich Objekte im Hintergrund rendern und in Texturen abspeichern, die dann für weitere Berechnungen oder Kompositionen genutzt werden können [48]. Wie genau das funktioniert, wird im Folgenden beschrieben.

In OSG funktioniert ein normaler Renderdurchlauf (*Renderpass*) so, dass die Elemente einer Szene, die dargestellt werden soll, als Kindknoten der Hauptkamera angehängt werden. Die Kamera visualisiert in jedem Frame die Elemente anhand Vertex- und Fragmentshader in der Anwendung und zeigt sie auf dem Bildschirm an. In manchen Fällen ist es aber hilfreich, die Szene zunächst *off-screen*, also außerhalb des Bildschirms, zu rendern und das Ergebnis für weitere Berechnungen zu nutzen, die für die finale Darstellung auf dem Bilschrim nötig sind. Um das zu erreichen, wird sog. *Multipass-Rendering* benötigt, also mehrere Renderdurchläufe in einem Frame [48]. Ein zusätzlicher Renderpass kann entweder vor der finalen Visualisierung stattfinden (*Prerender*) oder danach (*Postrender*). Bei vielen Techniken werden zusätzliche Renderduchläufe dafür genutzt, die komplette Szene mehrmals, aber jedes mal unterschiedlich, darzustellen (z.B. in einem Durchlauf nur die Oberflächennormalen der Objekte und im nächsten die Tiefenwerte der Szene) [32]. Da in diesen Fällen immer die gleiche Szene mit demselben Shader gerendert wird, muss eine Methode her, die es ermöglicht, die verschiedenen Ausgaben des Shaders zwischenzuspeichern, damit sie nach dem Renderdurchlauf nicht verloren gehen. Diese Möglichkeit bietet eine Kom-

bination aus den Technik: *Render To Texture (RTT)* und *Multiple Render Targets (MRT)*. Mittels RTT ist es möglich, die Ausgaben eines Shaders in einen Buffer zu speichern und damit eine Textur zu befüllen. MRT ermöglicht weiterhin, die verschiedene Ausgaben des Shaders zur selben Zeit in unterschiedliche Texturen zu schreiben [48].

In OSG werden FBOs erstellt, indem zunächst eine Kamera initialisiert wird, welcher die Eigenschaft des Pre- oder Postrenderings zugewiesen bekommt, und danach ein FBO angehängt wird. Diesem FBO wird dann ein Color Buffer zugewiesen, die Auswahl geht dabei von `Color_Buffer_0` bis `Color_Buffer_15`, da die maximale Anzahl von MRTs in OSG auf 16 limitiert ist; `Color_Buffer_0` enthält dabei die Ausgabe des Shaders an der Position 0, `Color_Buffer_1` die Ausgabe von Position 1, usw. sowie eine zuvor erstellte Textur, in die das Ergebnis des Color-Buffers geschrieben wird. Diese Textur kann dann beispielsweise als Input für andere Shader dienen [54].

3 | Anforderungen

Im bisherigen Verlauf dieser Arbeit sollte das Basiswissen zur Akquirierung und Visualisierung von seismischen Volumendaten sowie die technische Umsetzung klar geworden sein. Es sollte außerdem die Wichtigkeit von seismischen Störungen bei der Erschließung neuer Öl- und Gasvorkommen festgestellt worden sein und wie essentiell die Analyse dieser Störungen ist. Die Analyse steht und fällt jedoch mit den Darstellungs- und Interaktionsmöglichkeiten, die der Experte für die seismischen Daten zur Verfügung hat.

Wie bereits in Kapitel 2.2.3 erwähnt wurde, herrscht ein Mangel an Methoden für die Interaktion mit seismischen Volumendaten, die sowohl auf 2D-Slices als auch im direkten Volumenrendering funktionieren. In Kapitel 2.1.4 wurde außerdem gesagt, dass die Interpretation von Störungen mittels Geometrien erfolgt.

Die geläufigste Interaktionsmethode mit Volumendaten ist das direkte *picking* von Daten mit einem Eingabegerät (z.B. Maus oder Stift), da es equivalent zu dem natürlichen Verhalten des Menschen ist: *Auf etwas zu zeigen das man sieht* [55]. Eine intuitive Methode zur Analyse von Störungen würde demnach so aussehen, dass die Störungsgeometrie vom Experten an der Position erstellt wird, auf die der Experte in dem Moment mit dem Eingabegerät zeigt. Da Störungen aber nicht auf einen einzigen Punkt beschränkt sind, ist es mit dem *picking* alleine noch nicht getan. Stattdessen wäre es sinnvoller, den Verlauf der Störung direkt nachzuzeichnen (*painting*) und anhand dieser Linie die Geometrie zu erstellen.

Als Resultat einer Zusammenarbeit mit der Öl- und Gasindustrie (ExxonMobil [20], Statoil [50], innerhalb des Projektes *VRGeo* [10]) und der Geothermie-Forschung (Leibniz Institut für angewandte Geophysik - LIAG [22], im Projekt *SeisViz3D* [9]) wurde eine Anforderungsanalyse für besagte Methode zur Analyse von seismischen Störungen erstellt:

Anforderungen an eine Analysemethode:

1. Die Methode soll sowohl in der Darstellung mit 2D-Slices als auch im direkten Volumenrendering anwendbar sein.
2. Die Einfachheit der Methode soll dadurch gewährleistet sein, dass der Experte durch eine einzige Interaktion das auswählt, was er sieht [55].
3. Die Analyse von Störungen erfolgt durch das Erstellen von Dreiecksgeometrien innerhalb eines Volumendatensatzes.

4. Die Erstellung der Geometriepunkte soll durch die Interaktionsmethoden *picking* und *painting*, die direkt auf den Daten ausgeführt werden, möglich sein.

Eine Methode, die diese Anforderungen erfüllt, ist in bisherigen Softwarepaketen für die seismische Interpretation noch nicht vorhanden (vgl. Kapitel 4.2. Der Bedarf der Experten ist dementsprechend hoch, besonders bei der Möglichkeit, intuitiv mit dem Volumenrendering interagieren zu können. Da das *picking* die am häufigsten benutze Interaktionsmethode ist, gibt es (vor allem im medizinischen Bereich) bereits verschiedene Ansätze, die der hier gewünschten Methodik nahe kommen. Einige davon werden im folgenden Kapitel kurz vorgestellt.

4 | Stand der Technik

In diesem Kapitel werden zunächst einige bereits vorhandene Techniken für das *picking* von Volumendaten erläutert und es wird dargelegt, warum die meisten dieser Ansätze der obigen Anforderungsanalyse nicht entsprechen. Danach werden einige gängige geologische Softwarepakete vorgestellt, welche von vielen Experten genutzt werden. Zum Schluss wird kurz auf die Software, in der die geforderte Methode integriert werden soll, Bezug genommen und erklärt mit welchen Methoden die Störungsinterpretation dort bisher durchgeführt wurde. Die Software trägt den Namen *SeisViz3D* [53].

4.1 Techniken für Volumeninteraktion

Im Vergleich zu dem *picking* von Volumendaten im direkten Volumenrendering ist die Methode, wenn auf 2D-Sektionen angewandt, eine eher triviale Angelegenheit.

Nach Wiebel et al. [56] können die 2D-Slices in einem Volumendatensatz als einfache viereckige geometrische Objekte angesehen werden, wodurch sich die *picking*-Technik relativ einfach realisieren lässt. Bei der Auswahl von geometrischen Objekten im dreidimensionalen Raum wird nach Batagelo et al. [7] am Häufigsten der Ansatz verfolgt, einen Strahl (auch *picking-ray* genannt) in die Szene zu schießen und mit einem einfachen Schnitttest (Strahl trifft auf Dreiecks-Mesh [13]) die getroffene Geometrie zu bestimmen und durch geeignete Koordinatentransformation den 3D-Punkt in der Szene zu rekonstruieren.

Im direkten Volumenrendering sind dagegen keine Geometrien vorhanden, weshalb obiges Verfahren dort nicht angewandt werden kann.

Ein häufiger Ansatz bei der Rekonstruktion eines bestimmten 3D-Punktes in Volumendaten ist das Prinzip, zunächst ebenfalls einen *picking-ray* zu erzeugen und den nächsten sichtbaren Voxel auf diesem Strahl als 3D-Punkt zu rekonstruieren. Gobetti et al. [25] führten eine Technik ein, die heute noch ihre Verwendung findet. Dabei wird der Strahl anhand des in Kapitel 2.2.1 beschriebenen Compositings nach einem Voxel durchsucht, dessen Opazitätswert mindestens so groß ist wie der vom Nutzer vorher festgelegten, Schwellwert (*threshold-picking*). Diese Technik kann vereinfacht werden, indem der Schwellwert auf 0 gesetzt wird und somit der erste sichtbare Voxel auf dem Strahl ausgewählt wird. Dieser Ansatz wird auch *first-hit* genannt.

Beide Verfahren haben ihre Nachteile: Das Setzen eines Schwellwertes für die Opazität ist nicht trivial und so kann es passieren, dass der gewählte Wert zu hoch ist und somit semi-transparente Regionen nicht ausgewählt werden, obwohl sie sichtbar sind. Dahingegen kann es bei der *first-hit* Methode passieren, dass je nach angewandter Transferfunktionen bestimmte Regionen nicht vollständig ausgeblendet werden und noch sehr schwach sichtbar sind. Diese würden dann trotzdem ausgewählt werden, obwohl die gewünschte Position dahinter liegt. Die Problematiken werden in Abbildung 4.1 dargestellt.

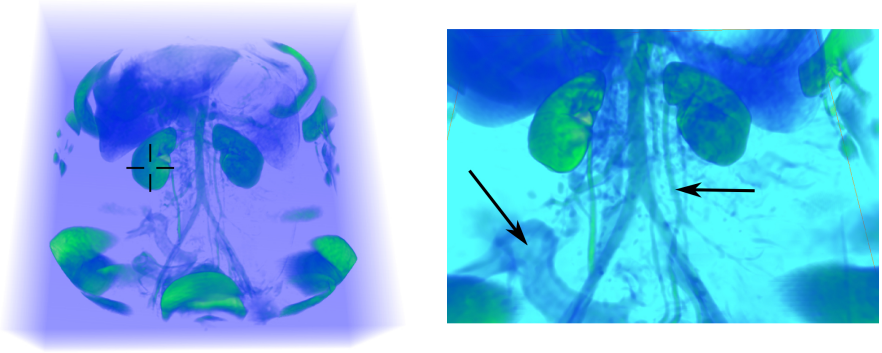

Abbildung 4.1: Links: Bei zu niedrigem Schwellwert werden Daten ausgewählt, die nur ganz schwach sichtbar sind, statt klar erkennbarer Regionen; Rechts: Ein zu hoher Schwellwert ignoriert womöglich wichtige Strukturen in den Daten, die nicht komplett opak sind. Entnommen aus [55]

Wiebel et al. [55] haben sich dieser Technik angenommen und sie verbessert. Dabei wird der Opazitätswert jedes Voxels auf dem *picking-ray* mit dem dahinter liegenden verglichen. Es wird außerdem angenommen, dass die gewünschte Position, die ausgewählt werden soll, der Bereich ist, an dem der größte Opazitätssprung auf dem Strahl stattfindet (s. Abbildung 4.2). Dieses Verfahren wird häufig eingesetzt, hat jedoch den Nachteil, das der gesamte Strahl abgetastet werden muss, um nach dem größten Opazitätssprung zu suchen, wodurch eine Beschleunigung des Renderings mittels *Early-Ray-Termination* (vgl. Kapitel 2.2.1) nicht mehr möglich ist.

Ein anderer Ansatz, welcher häufig in der Medizin seinen Einsatz findet, legt den Fokus nicht so sehr auf die Sichtbarkeit der Voxel, sondern bezieht die Skalarwerte der Daten in das *picking*-Verfahren mit ein. Toennis und Derz [51] verwenden dafür benutzerdefinierte Werte, um die Region im Datensatz auszuwählen, deren Skalarwerte am ehesten den Werten des Benutzers nahe kommen. Im Falle von seismischen Daten könnte man für dieses Verfahren die gemessenen Amplituden nehmen, jedoch gäbe es dann

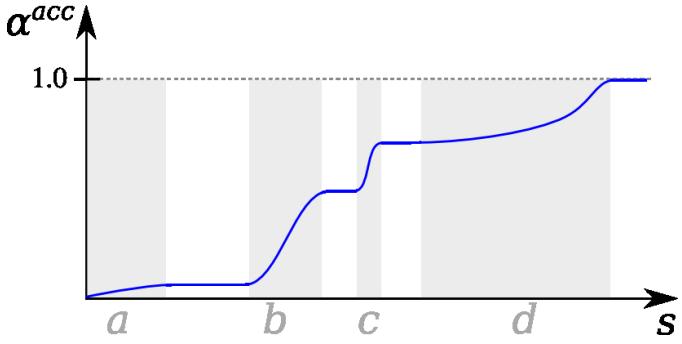

Abbildung 4.2: Auf der x-Achse ist das Sampling des picking-rays dargestellt und auf der y-Achse die Opazität. Der höchste Opazitätssprung findet im Intervall zwischen b und c statt. An dieser Stelle werden die Daten vermutet, die der Nutzer auswählen möchte. Entnommen aus [55]

das Problem, dass bei einer angewandten Transferfunktion (was in der seismischen Interpretation sehr oft der Fall ist) auch nicht sichtbare Daten in den Fokus fallen würden.

Kohlmann et al. [30] haben ein Verfahren entwickelt, bei dem die Rekonstruktion der 3D-Position im Volumen unter Zuhilfenahme von Metadaten des Datensatzes berechnet wird. Häufig wird dieses Verfahren auf medizinische Datensätze des Dateityps „DICOM“ angewandt, da es neben den Volumendaten zusätzliche Informationen zu Körperteilen wie Arterien, Organen oder Knochen bereithält und so von dem Nutzer ausgewählt werden kann, welche Regionen er auswählen möchte [30] [45]. Im Falle von seismischen Daten findet dieses Verfahren allerdings keine Verwendung, da diese im Dateityp „SEGY“ abgespeichert werden, welches keine Metadaten enthält.

Eine andere Technik, die das *picking* mit der *painting*-Methode kombiniert, wurde von Owada et al. [38] entwickelt. Dabei wird die Kontur eines Objektes oder Struktur im Datensatz „nachgezeichnet“ und somit ausgewählt. Das Objekt bzw. die Kontur wird anschließend in den meisten Fällen vom Rest der Daten mittels Transferfunktion isoliert (s. Abbildung 4.3). Im Falle seismischer Daten könnte dieses Verfahren benutzt werden, um einzelne Horizonte zu *picken* und freizustellen. Da dies jedoch nur bedingt für die Störungsinterpretation hilfreich wäre, wird dieser Ansatz nicht weiter verfolgt.

Peng et al. [42] nutzen ein Verfahren, bei dem die Auswahl eines Punktes in dem Volumen durch den Schnittpunkt zweier Strahlen bestimmt wird. Der Nutzer muss dabei mit der Maus zweimal von unterschiedlichen Ansichten

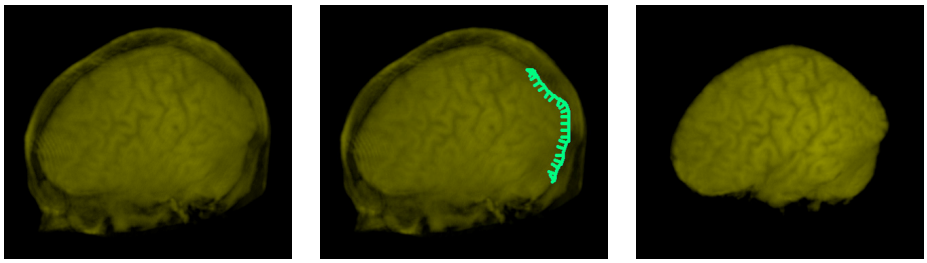

Abbildung 4.3: Links: Medizinische Volumendaten eines Kopfes mit angewandter Transferfunktion; Mitte: Kontur des Gehirns wird nachgezeichnet; Rechts: Gehirn wird von umliegenden Daten (Knochen etc.) isoliert dargestellt. Entnommen aus [38]

aus auf die gewünschte Region in den Daten klicken. Dadurch werden zwei *picking-rays* losgeschickt, an deren Schnittpunkt dann die 3D-Position berechnet wird. Nachteil dieser Methode ist, dass zum einen die gewünschte Region von beiden Ansichten aus sichtbar sein muss, was je nach Komplexität der Daten nicht unbedingt gegeben ist, und zum anderen ist die Auswahl eines Punktes über das zweimalige Anklicken nicht besonders intuitiv.

Das Verfahren, welches in dieser Arbeit für die Anwendung in seismischen Volumendaten entwickelt wurde, verfolgt eine Kombination aus den Ansätzen *threshold-picking* und *first-hit*. Der Fokus bei seismischen Daten liegt immer auf klar sichtbaren Strukturen, weshalb der *first-hit*-Ansatz eigentlich ausreichend wäre. Es kann jedoch in einzelnen Fällen passieren, dass der Nutzer eine Transferfunktion nicht ganz korrekt einstellt und dadurch einzelne Amplituden ganz schwach dargestellt werden. Da bei seismischen Daten allerdings schon ein sehr geringer Opazitätswert zu sichtbaren Daten führt, kann der Schwellwert für das *threshold-picking* dementsprechend niedrig gesetzt werden.

In der in dieser Arbeit entwickelten Methode wurde der Schwellwert auf $T = 0.05$ mit $T \in [0, 1]$ gesetzt, da damit gute Ergebnisse erzielt wurden und sehr schwach sichtbare Daten ignoriert und alle klar sichtbaren Strukturen ausgewählt wurden. In Kapitel 5.2.2 wird die Findung dieses Wertes noch näher erläutert.

4.2 Techniken in geologischen Softwarepaketen

Auf dem Markt existieren viele unterschiedliche Softwarepakete, die von Experten für die seismische Interpretation genutzt werden. Sie unterscheiden sich dabei von der Art und dem Umfang der Funktionalitäten, die je nach Aufgabe in der seismischen Interpretation benötigt werden. Im Folgenden werden die gängigsten Softwarepakete kurz vorgestellt und im Hinblick auf die jeweilige Art der Störungsinterpretation erläutert.

Die Software *Petrel* von dem Unternehmen *Schlumberger* unterstützt kein direktes Volumenrendering. Stattdessen werden die Horizonte und Störungen als Geometrien dargestellt. Die Modellierung der Störungsgeometrien erfolgt auf der Darstellung der Daten als 2D-Slice. Dabei werden die Horizonte auf der 2D-Slice markiert. Diese Markierungen werden dann in die Tiefe des Volumens verfolgt und anhand der verfolgten Punkte werden die Geometrien der Horizonte automatisch erstellt. Die Störungen können dann ausfindig gemacht werden, indem nach einem Versatz zwischen den Geometrien gesucht wird (s. Abbildung 4.4) [49].

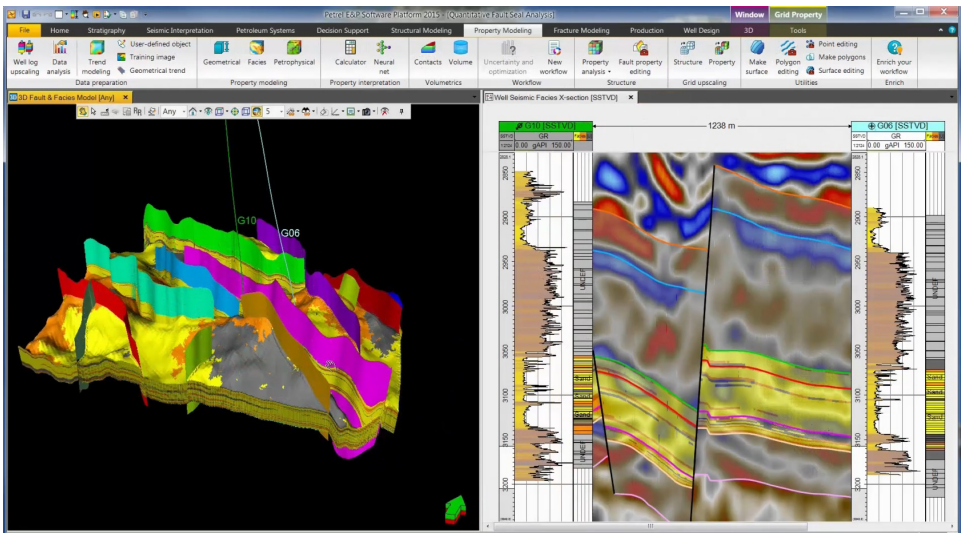

Abbildung 4.4: Software Petrel. Links: Horizontgeometrien im Volumen; Rechts: Markierte Horizonte auf der 2D-Slice mit sichtbarem Versatz [49]

Das automatische Verfolgen von Horizonten wird auch *Advanced Horizon Tracking* [29] genannt. Dieses Verfahren wird sehr häufig für die Störungsinterpretation genutzt, so auch von der Firma *Paradigm*. *Paradigm* stellt eine Vielzahl von unterschiedlichen Softwarepaketen zur Verfügung, die modular, je nach Aufgabe in der seismischen Interpretation, aufgeteilt sind. Einige Pakete sind in der Lage, die seismischen Daten mittels direktem Volumenrende-

ring zu visualisieren; das Paket *Fault Extraction*, das für die Interpretation von Störungen genutzt wird, basiert jedoch auf der Darstellung einzelner Sektionen. Hier wird ebenfalls das oben genannte Verfahren genutzt: Die Horizonte werden auf den Sektionen markiert und von der Software automatisch getrackt. Durch verschiedene Algorithmen aus der Bildverarbeitung werden dann, anhand der getrackten Horizonte, die Störungsgeometrien erstellt [40].

Die frei verfügbare Software *OpendTect* verfährt ebenso wie *Paradigm* mit der Störungsinterpretation; das Ergebnis ist in Abbildung 4.5 zu sehen [35].

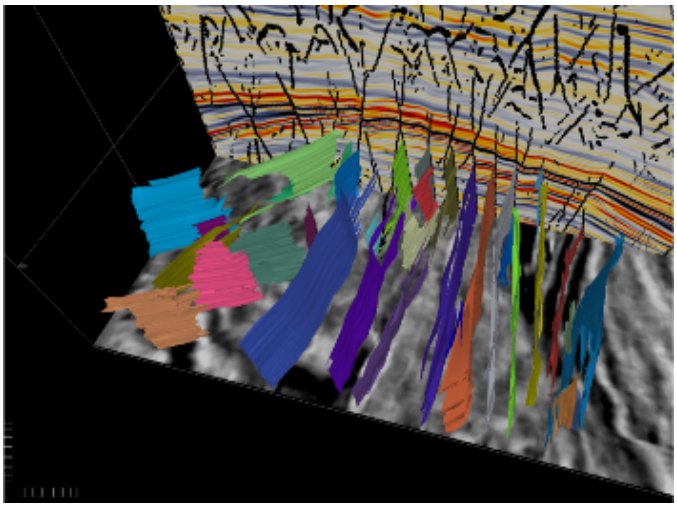

Abbildung 4.5: Automatisch erstellte Störungen anhand getrackter Horizonte aus OpendTect [35]

Die Software *Paradise* von *Geophysical Insight* ist eine mächtige Software im Bereich Visualisierung und unterstützt das direkte Volumenrendering neben einer Vielzahl von weiteren Visualisierungen. Eine direkte Störungsinterpretation mit der Erstellung von komplexen Geometrien ist allerdings nicht möglich [36].

Eine Software, die ebenfalls das dVR unterstützt, ist *GeoTeric*. Bei dieser Software funktioniert die Störungsinterpretation so, dass auf eine 2D-Slice bestimmte Bereiche markiert und somit einzelne Amplitudenwerte ausgewählt werden. Anhand dieser Amplitudenwerte wird dann der Horizont im direkten Volumenrendering verfolgt und somit Störungen sichtbar gemacht (s. Abbildung 4.6) [23].

Die Software, welcher der Methode dieser Arbeit, zumindest im Bereich der Störungsinterpretation auf Sektionen, am nächsten kommt, ist der *Decisi-*

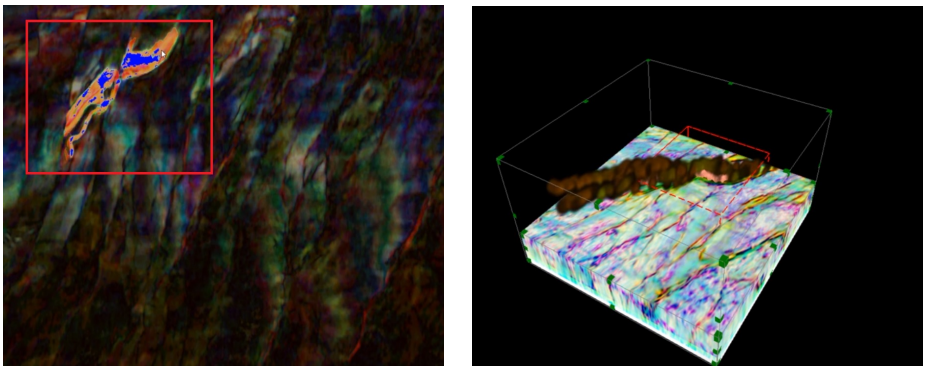

Abbildung 4.6: Software GeoTeric. Links: Markierter Bereich auf 2D-Slice; Rechts: Verfolgung des Horizonts im dVR [23]

onSpace Desktop von *Landmark Halliburton*. Hier wird die Störungsgeometrie intuitiv erstellt, indem der Nutzer einfach mit dem Eingabegerät auf die Sektion den Verlauf der Störung einzeichnet. Dadurch entsteht eine Linie, welche dann interpoliert wird, wodurch in regelmäßigen Abständen Geometriepunkte auf der Linie erzeugt und anschließend trianguliert werden, wodurch eine Fläche entsteht (s. Abbildung 4.7).

Dieses Verfahren funktioniert jedoch nur auf 2D-Slices und nicht im Volumenrendering [28].

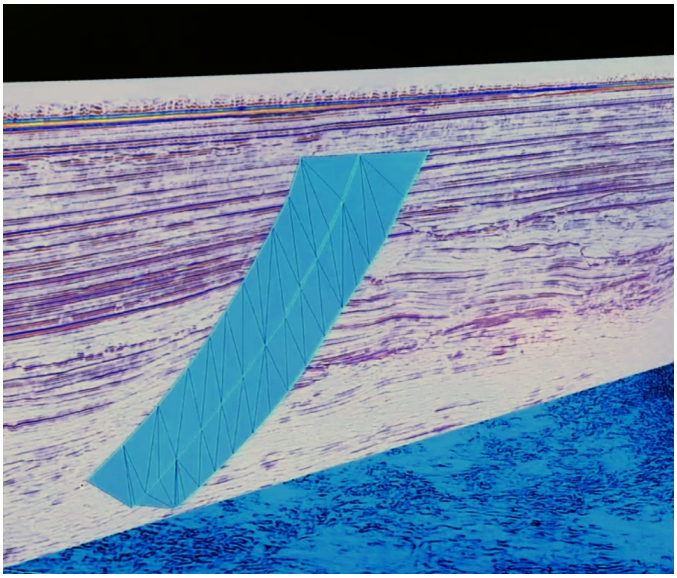

Abbildung 4.7: DecisionSpace Desktop: Störungsgeometrie wird anhand eingezeichneter Linien automatisch erstellt [28]

4.3 Bisherige Techniken der SeisViz3D Software

Die Software *SeisViz3D* bietet diverse Visualisierungs- und Interaktionsmöglichkeiten. Ein besonderer Aspekt, von der sie sich von vergleichbarer Software abgrenzt, ist die Möglichkeit, seismische Daten in Stereo 3D zu betrachten [53]. Jedoch unterstützt sie bislang keine der oben genannten Methoden. Das *Picken* der Volumendaten funktionierte bisher so, dass der Nutzer der Software einen 3D-Cursor in der Szene (grüner Würfel, s. Abbildung 4.8) manuell mit dem Mausrad bewegen konnte. Dabei hat sich der 3D-Cursor bei Drehung des Mausrades nach vorne, auf einem Strahl entlang, tiefer in die Szene hinein bewegt und bei Drehung nach hinten bewegte sich der Cursor auf demselben Strahl zurück.

Das Setzen der Geometriepunkte für die Störungsgeometrien konnte immer an der aktuellen Position des 3D-Cursors getätigt werden und war dadurch eine sehr zeitaufwendige Aktion, da für jeden Punkt zunächst getestet werden musste, ob sich der 3D-Cursor genau auf den sichtbaren Daten, also in der korrekten Tiefe, innerhalb der Szene befindet. Dieser „Tiefentest“ (s. Abbildung 4.8) wurde vom Nutzer auf Basis der Verdeckung des Cursors durch die Daten durchgeführt: Wenn die Geometrie des Cursors sichtbar war, lagen die Daten dahinter; war der Cursor nicht sichtbar, lagen die Daten davor; war die Geometrie halb verdeckt, konnte vom Nutzer angenommen werden, dass sich der Cursor auf den Daten befand und an dieser Stelle der Punkt für Störungsgeometrie gesetzt werden.

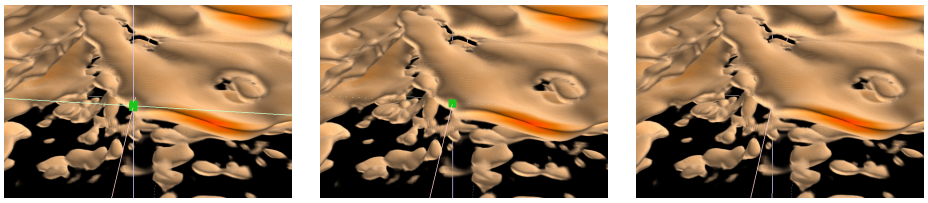

Abbildung 4.8: Links: 3D-Cursor vor den Daten (Falsch); Mitte: 3D-Cursor genau auf den Daten, da halb verdeckt (Richtig); Rechts: 3D-Cursor hinter den Daten (Falsch)

Die letztendliche Erstellung der Geometrie erfolgt durch die automatisch Triangulierung der zuvor erstellten Punkte. Dafür werden in der Software bekannte Triangulationsalgorithmen wie *Greedy-Projecton* [16] und *Delaunay* [14] verwendet. Diese Algorithmen wurden auch für die in dieser Arbeit entwickelte Methode genutzt.

Die beiden nachfolgenden Kapitel beschäftigen sich mit der Implementierung der Methode für die intuitive Erstellung von Störungsgeometrien. Dabei wird zunächst in Kapitel 5 die Implementierung des ersten Teilpaketes der Methode erläutert, welches eine notwendige Komponente für die Realisierung des zweiten Teilpaketes in Kapitel 6 ist.

5 | Schnittpunktberechnung in Volumendaten

Das erste Teilpaket kann als Vorarbeit für die intuitive Erstellung von Störungsgeometrien angesehen werden und es beschäftigt sich mit der Lösung des Problems, wie die 3D-Position des Cursors im Raum korrekt bestimmt wird. Es wird davon ausgegangen, dass der Experte während der seismischen Störungsinterpretation die Geometrie anhand der visualisierten seismischen Daten erstellt. Deshalb kann auch angenommen werden, dass eine zufriedenstellende Position des 3D-Cursors derselben Position entspricht, an der sich die ersten sichtbaren Daten im Volumen befinden.

Da eine Anforderung an die Methode die Funktionalität sowohl in der 2D-Slices-Darstellung als auch im direkten Volumenrendering beinhaltet, wird im Folgenden zunächst die Umsetzung der Methode auf den Sektionen und anschließend im dVR erklärt.

5.1 Berechnung auf 2D-Slices

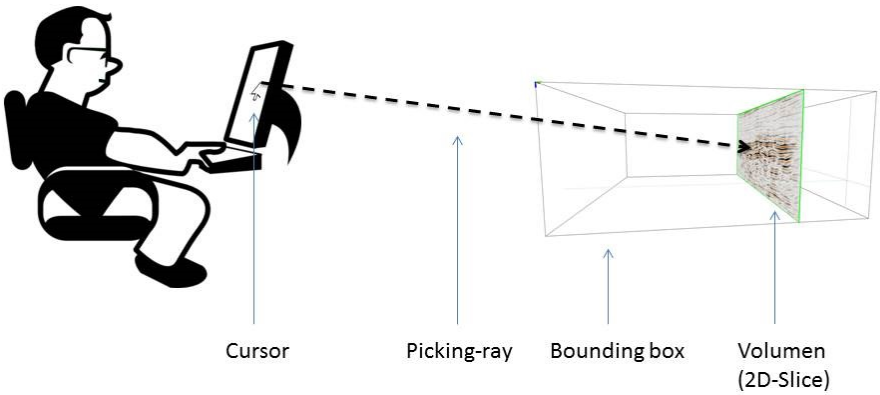

Abbildung 5.1: *Picking-ray* auf 2D-Slice

In Kapitel 2.2.3 wurde gesagt, dass die Visualisierung von Volumendaten mit 2D-Sektionen technisch wesentlich einfacher umzusetzen ist, als die Darstellung im dVR. Dieses Verhalten spiegelt sich auch in den Interaktionsmethoden wieder.

Das Ziel bei der Darstellung in 2D-Sektionen ist es, die Punkte für die Störungsgeometrie direkt auf der Sektion zu erstellen, auf die der Nutzer in dem Moment mit dem Eingabegerät zeigt.

Wie bereits in Kapitel 2.2.3 erwähnt, können 2D-Sektionen als einfache viereckige Geometrie angesehen werden [56]. Das bedeutet, dass die Störungsgeometrie vom Nutzer unmittelbar auf der Oberfläche der Slicegeometrie erstellt werden soll. Der Ansatz, welcher hier verfolgt wird, hat einige Ähnlichkeiten mit den oben genannten Techniken (vgl. Kapitel 4.1), denn auch für diese Methode wird ein *picking-ray* verwendet, also ein Strahl, dessen Ursprung sich aus der aktuellen 2D-Position des Eingabegerätes (z.B. Mauszeiger) und dem Tiefenwert des Augpunktes zusammensetzt, und von dort aus senkrecht in die Szene geschossen wird. Um das oben genannte Ziel zu erreichen, muss nun der Schnittpunkt dieses Strahls mit der Slicegeometrie gefunden werden (s. Abbildung 5.1). Da es sich dabei um eine relativ simple Schnittpunktberechnung zwischen Strahl und Ebene handelt, wird auf die Berechnung nicht weiter eingegangen.

In OSG gibt es für diese Aufgabe bereits vorgefertigte Klassen [37]. Der *LineSegmentIntersector* von OSG kümmert sich um die korrekte Erstellung des *picking-rays*. In Kombination mit dem *intersection visitor* sind alle grundlegenden Funktionen vorhanden, die für die Schnittpunktberechnung mit der Oberfläche der Slicegeometrie benötigt werden. Dieser *visitor* durchläuft dabei alle vorhandenen *nodes* im Szenegraphen und überprüft, ob ein Schnittpunkt mit dem *picking-ray* vorhanden ist oder nicht. Ein Problem, welches durch das Durchlaufen der *nodes* jedoch auftritt, ist, dass in der Szene auch noch andere Geometrien als die 2D-Sektion vorhanden sind. Das Volumen hat beispielsweise eine sog. *bounding box* oder *Hüllkörper* (s. Abbildung 5.1), welche die Grenzen des Volumens festlegen und in anderen Bereichen oftmals auch als *hit-box* zur Kollisionserkennung bekannt ist [52]. Zwar sind die Flächen dieser Box nicht sichtbar, werden aber dennoch von dem *picking-ray* als Geometrie interpretiert. Es gilt also, für dieses Verfahren einen Weg zu finden, zwischen den Geometrien zu unterscheiden. Dies kann dadurch gelöst werden, indem jedem *node* im Szenegraphen individuelle Parameter wie z.B. einen Namen zugewiesen wird und indem man den *visitor* alle Geometrien ignorieren lässt, die nicht diesem Parameter entsprechen.

Falls ein *node* solche Parameter nicht besitzt, gibt es in OSG noch die Möglichkeit, sog. *camera masks* zu erstellen und dem *node* zuzuweisen. Der *visitor* bekommt dann eine Freigabe für eine bestimmte *mask* und erkennt alle Geometrien, die dieser angehören. *Camera masks* werden noch genauer in Kapitel 5.2.2 erläutert.

Wenn ein Schnittpunkt mit der Slicegeometrie gefunden wurde, wird der Schnitttest abgebrochen und der gefundene Punkt gespeichert. Dieser ist bislang im lokalen Koordinatensystem der gefundenen Geometrie gegeben und muss noch anhand der Position der Slice im Raum und der Größe des Volumens in das passende Koordinatensystem der Szene umgerechnet werden. Anschließend wird der 3D-Cursor auf diese Position gesetzt und er ist somit automatisch immer auf der Sektion, auf die der Nutzer mit dem Eingabegerät aktuell zeigt.

5.2 Berechnung im direkten Volumenrendering

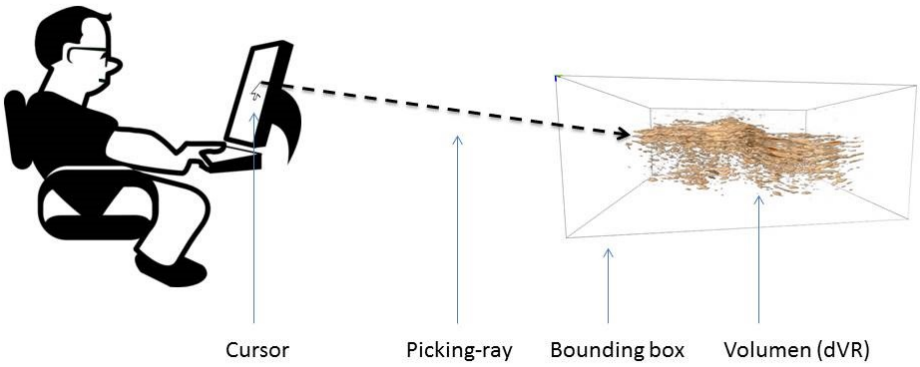

Abbildung 5.2: *Picking-ray* auf direktem Volumenrendering

Wie bereits erwähnt, ist die Schnittpunktberechnung im direkten Volumenrendering alles andere als trivial. Das hat den Grund, dass im dVR keine Geometrien vorhanden sind, weshalb obiger Schnitttest hier nicht funktioniert. Außerdem muss beachtet werden, dass bei der Anwendung einer Transferfunktion die nicht sichtbaren Daten nicht in die Berechnung der 3D-Cursor Position mit einfließen dürfen.

5.2.1 Konzeption

Dadurch, dass ein einfacher Schnitttest mit einer Geometrie nicht möglich ist, und auch das Wissen um die Größe und Position der *bounding box* des Volumens bei angewandter Transferfunktion nicht hilfreich ist und zu keinem zufriedenstellenden Ergebnis führt, muss eine andere Größe gefunden werden, anhand derer die Position des 3D-Cursors im Raum bestimmt werden kann.

Der grundlegenden Ansatz hinter der Methode ist es, die *Texturkoordinaten* des Volumens als maßgebende Größe hierfür zu nutzen. Texturkoordinaten sind die normalisierten (x, y, z) Positionen der Daten im Raum, als RGB-Farben dargestellt. Die Koordinaten enthalten also Werte zwischen 0 und 1; dabei entspricht die Position $(0, 0, 0)$ einer schwarzen Farbe und die Position $(1, 1, 1)$ ist weiß. Dazwischen entsteht ein farblicher Verlauf [27]. In Abbildung 5.3 ist zu sehen, wie eine Würfelgeometrie in Texturkoordinaten dargestellt ist. Links sind die *front-faces* des Meshes dargestellt (linke obere Ecke weiß), in der Mitte sind die *back-faces* visualisiert (untere rechte Ecke schwarz) und ganz Rechts ist die Differenz der beiden Fabwerte zusehen. Diese Differenz wird u.a. auch dafür benutzt, um die Richtung der Strahlen für das *Raycasting* zu bestimmen [27].

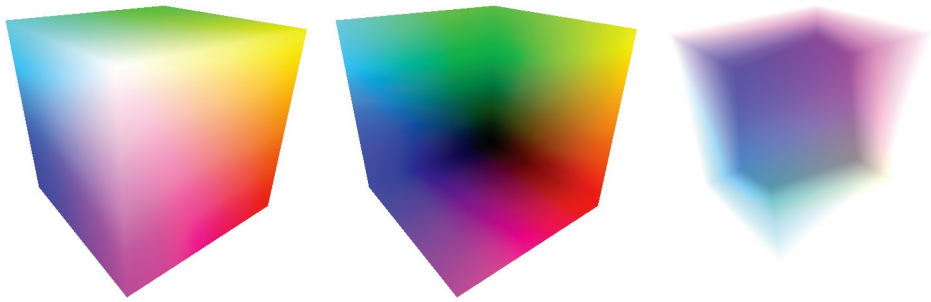

Abbildung 5.3: Würfelgeometrie in Texturkoordinaten. Links: front faces; Mitte: back faces; Rechts: Ergebnis von Mitte - Links. Entnommen aus [27]

Durch die 3D-Texturkoordinaten besteht nun die Möglichkeit, jeder Position innerhalb des Volumens einen individuellen und einzigartigen RGB-Farbwert zuzuordnen. Ein weiterer Vorteil ist, dass dieser Ansatz selbst bei angewandter Transferfunktion noch funktioniert, da der Transparenzwert jedes Voxels unabhängig von seiner Texturkoordinate ist und somit unsichtbare Voxel auch nicht in ihrem RGB-Farbwert der Texturkoordinate dargestellt werden. Die Voxel, welche durch die Transferfunktion isoliert wurden, bleiben hingegen sichtbar (s. Abbildung 5.4).

Der weitere Verlauf dieses Ansatzes ist es, den nun existierenden RGB-Texturkoordinatenfarbwert des Voxels auszulesen, der sich genau „unterhalb“ (d.h. auf dem *picking-ray*) des Cursors des Eingabegerätes befindet. Anschließend soll anhand dieser Farbwerte durch geschickte Transformation auf die Position des Voxels im korrekten Koordinatensystem rückgeschlossen werden, so dass der 3D-Cursor auf diese Position gesetzt werden kann. Diese Methode wird im weiteren Verlauf auch als *cursor snapping* bezeichnet.

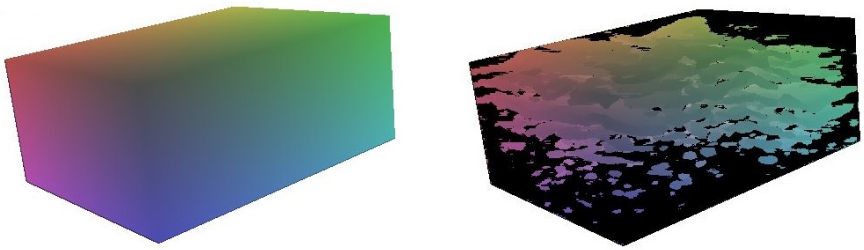

Abbildung 5.4: Volumendatensatz in Texturkoordinaten. Links: Ohne angewandte Transferfunktion; Rechts: Mit angewandter Transferfunktion (Volumen vor schwarzem Hintergrund)

5.2.2 Implementierung

Nachdem der Ansatz hinter der Methode grob vorgestellt wurde, wird nun detaillierter auf die Umsetzung und Implementierung eingegangen. Außerdem wird eine Methode vorgestellt, mit der obiger Ansatz performanter gestaltet werden kann.

Erstellung von Texturkoordinaten

Wie im vorangegangen Kapitel bereits erwähnt wurde, ist es essentiell, das Volumen in Texturkoordinaten darstellen zu können. Die Texturkoordinaten der Volumengeometrie werden im *Vertex-Shader* berechnet mithilfe des OpenGL Befehls `gl_MultiTexCoord`. Diesem Befehl wird eine Zahl (*Unit)* mitgegeben, welche dem Index der *Uniform Variable* entspricht, von der die Texturkoordinaten errechnet werden sollen [15]. Da die Darstellung der Volumengeometrie in Texturkoordinaten auch für das *Raycasting* sehr wichtig ist, müssen die Informationen in Texturen gespeichert werden, damit diese daraufhin im *Raycasting-Shader* genutzt werden können. Dafür werden in OSG zwei Kameras angelegt, eine für die Visualisierung der *front-faces* in Texturkoordinaten und die andere für die Darstellung der *back-faces*. Die zu rendernden Geometrien werden dann den Kameras als Kindobjekt angehängt und je nach Kamera werden entweder die front- oder back-faces der jeweiligen Geometrie dargestellt. Dies lässt sich bei OSG über den *polygon mode* einer Geometrie festlegen. Mit dem Befehl `OSG::PolygonMode::FRONT` werden die front-faces dargestellt und mit `OSG::PolygonMode::BACK` die back-faces.

Um die *face*-Texturen zu erzeugen, müssen diese zunächst als 3D-Textur in OSG angelegt werden. Mit dem Befehl `setRenderTargetImplementation (FRAME_BUFFER_OBJECT)` bekommen die Kameras den Hinweis, die aufgenommenen Bilder offscreen d.h. für den Nutzer nicht sichtbar, in die erstellten Texturen zu rendern. Nachdem die Texturen befüllt sind, können sie als

Shader Eingaben verwendet werden. Dazu werden sie als *Uniform* an den *Fragment-Shader* übergeben. Für die korrekte Abtastung der Textur müssen die Abtastpositionen in Texturkoordinaten angegeben werden; diese werden erneut mit dem Befehl `gl_MultiTexCoord` im *Vertex-Shader* berechnet und übergeben. Danach stehen die Texturen als RGBA-Wert im Shader zur Verfügung. Für das Raycasting-Verfahren ist die Differenz zwischen der back- und front-faces-Textur bedeutend, da anhand dieser Differenz die Richtung des einzelnen Strahls festgelegt wird. Für die von der entwickelten Methode benötigte Darstellung des Volumens in Texturkoordinaten wurde die front-face-Textur verwendet, da bei der Methode der Fokus auf dem ersten von der Kamera ausgehenden, sichtbaren Sample liegt, welche von der front-face-Textur geliefert werden.

Da der Raycasting Shader allerdings bisher für die finale Visualisierung der seismischen Daten zuständig war und diese Aufgabe auch beibehalten soll, muss für die Visualisierung der Volumendaten in Texturkoordinaten eine zusätzliche Ausgabe des Shaders erzeugt werden. Wie bereits in Kapitel 2.3.1 erwähnt, funktionieren mehrere Shaderausgaben in OpenGL über *Multiple-Render-Targets (MRT)*. Da die Ausgabe des Volumens in Texturkoordinaten jedoch nur für die Berechnung der Cursor-Position genutzt und für den Nutzer der Software nicht sichtbar sein soll, muss diese zusätzliche Ausgabe des Shaders offscreen, mittels der *Render-To-Texture (RTT)* Methode, in eine Textur gerendert werden. Dafür wird ein FBO erstellt und der Kamera angehängt. Ein Problem besteht nun darin, dass eine Kamera nicht gleichzeitig on- und offscreen rendern kann. Da die Hauptkamera jedoch dazu benötigt wird, das seismische Volumen darzustellen, muss eine zweite Kamera (im Folgenden auch *Texturkamera* genannt) mit den identischen Eigenschaften der Hauptkamera erstellt werden, die das FBO als *Render-target* bekommt und anhand des zweiten Color-Buffers `COLOR_BUFFER1`, die zweite Shaderausgabe in eine zuvor erstellte Textur rendert. Die Texturkamera bekommt außerdem den Prerender-Parameter `setRenderOrder(osg::Camera::PRE_RENDER)`, wodurch der zusätzliche Renderpass vor dem eigentlichen Hauptrenderpass für die Darstellung der seismischen Daten ausgeführt wird.

Anpassung des Raycasting-Shaders

Das Raycasting der Volumendaten nutzt als Optimierungsmethode die *Early-Ray-Termination* (vgl. Kapitel 2.2.1). Dabei wird der Strahl bei einem Opazitätswert von 1 abgebrochen, um Rechenleistung zu sparen. Da die Anforderung an die gewünschte Methode war, den Cursor an die Position des ersten sichtbaren Voxels bzw. Samples des Strahls zu setzen, muss der Opazitätswert des aktuellen Samples nicht 1 erreichen; es reicht theoretisch, wenn der Farbwert in Texturkoordinaten des ersten Samples genommen wird, dessen

Opazität $\neq 0$ entspricht. Jedoch kann es dabei passieren, dass bei einer nicht ganz korrekten Anwendung einer Transferfunktion Teile der seismischen Daten, die für das Auge kaum auszumachen sind, von der Methode als sichtbar eingestuft werden und den Cursor auf diese Position setzen. Wie bereits in Kapitel 4.1 erwähnt, wird deshalb in dieser Methode das Verfahren des *threshold-pickings* nach Gobetti et al. [25] verwendet. Dabei wird erst der Voxel ausgewählt, dessen Opazitätswert den vorher festgelegten Schwellwert übersteigt, und anschließend dessen Texturkoordinaten als RGB-Farbwert gespeichert.

Die Festlegung dieses Schwellwertes ist speziell bei seismischen Daten keine allzu große Schwierigkeit, da in der seismischen Interpretation die Transferfunktionen meist so gewählt werden, dass sie einzelne Farbbereiche komplett ausblenden, weshalb der Schwellwert sehr niedrig gewählt werden kann, damit womögliche Fehler bei der Anwendung der Transferfunktion ignoriert werden. Der Schwellwert wurde in dieser Methode auf den Wert $T = 0.05$ gesetzt, da dieser sich in einigen Testdurchläufen bewährt hat.

Bei den Tests wurde absichtlich im Editor für 1D-Transferfunktionen ein Farbwert nicht hundertprozentig ausgeschaltet, was durch einen schnellen Klick mit der Maus durchaus passieren kann. In Abbildung 5.5 ist oben die korrekte Transferfunktion und daneben das Ergebnis bei einem Opazitätswert von $T = 0.0$ zu sehen: Der 3D-Cursor befindet sich auf den klar sichtbaren Daten, also der gewollten Position. In der Mitte befindet sich eine kleine Abweichung der Transferfunktion und daneben wieder das Ergebnis bei einem Opazitätswert von $T = 0.0$: Der Cursor befindet sich auf den vorderen, nur sehr schwach sichtbaren Daten (gut zu erkennen an den Linien, welche die Tiefenhinweis für den 3D-Cursor liefern). Unten findet sich dieselbe Abweichung in der Transferfunktion; das Ergebnis ist diesmal jedoch mit einem Opazitätswert von $T = 0.05$ dargestellt. Es ist zu sehen, dass die schwach sichtbaren Daten von dem 3D-Cursor ignoriert werden und er auf die klar sichtbare Struktur, wie im obersten Bild, *snappt*.

Dieser Wert sollte allerdings nicht als allgemeiner Standard angesehen werden, sondern bezieht sich allein auf die hier verwendete Software zusammen mit dem eigenen Editor für 1D-Transferfunktionen.

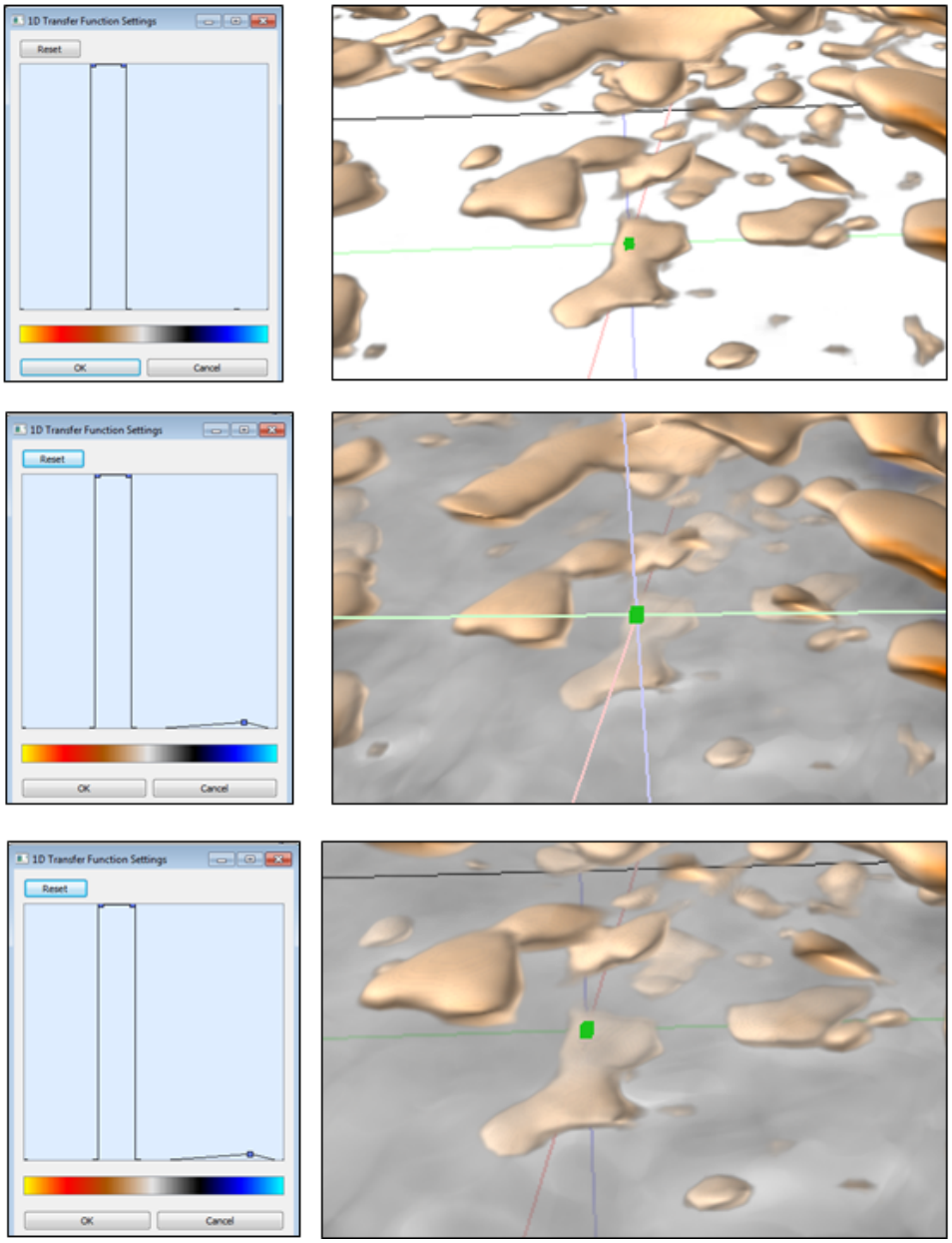

Abbildung 5.5: Oben: Saubere Transferfunktion bei $T = 0.0$; Mitte: Unsaubere Transferfunktion bei $T = 0.0$; Unten: Unsaubere Transferfunktion bei $T = 0.05$

Umrechnung von Farbwerten in 3D-Position

Nachdem nun immer der erste sichtbare Voxel der Volumendaten dargestellt in einem Farbwert, der seinen Texturkoordinaten entspricht, mittels FBO in einer Textur gespeichert wurde, ist die nächste Aufgabe, an genau diesen Farbwert zu gelangen, der von dem *picking-ray* getroffen wird.

Da in OSG nicht direkt auf die Informationen, die in einer Textur gespeichert sind, zugegriffen werden kann, gibt es einen Datentypen namens *Image*. Dieser Datentyp entspricht einem Array, wobei ein Feld dem Farbwert (RGBA) eines Pixels der Textur entspricht. Das Array hat also eine Länge von $TextureHeight$ x $TextureWidth$ x 4. Über den Befehl `setImage(...)` wird eine Textur einem Image angehängt und damit kann auf den Farbwert eines bestimmten Pixels der Textur zugegriffen werden.

Allerdings steht OSG hier vor einem Problem: Eine Textur kann <u>entweder</u> einem Image <u>oder</u> einem FBO angehängt werden. Da das FBO benötigt wird, um die Ausgabe des Raycasting-Shaders zu rendern, und das Image benötigt wird, um auf die Werte dieser Ausgabe zugreifen zu können, muss eine Lösung für dieses Problem gefunden werden.

Eine Lösung, die OSG zur Verfügung stellt, ist es, die Ausgabe des Shaders über das FBO nicht in eine Textur, sondern direkt in das Image zu rendern. Somit wird die Ausgabe direkt in das Datenarray des Images geschrieben und zugreifbar gemacht. Jedoch ist dies nicht die performanteste Lösung, denn die Texturkamera (an welcher das FBO hängt) muss den gesamten Bildschirm rendern, d.h. die volle Auflösung; bei dem Rechner, auf dem die Methode entwickelt wurde, entspricht dies einer Auflösung von 1680x1050; das Array des Images hat also 7056000 Stellen, welche in jedem Frame neu befüllt werden. Da dies sehr rechenaufwendig ist, wurde ein Verfahren zur Beschleunigung der Methode entwickelt, welches im folgenden Unterkapitel 5.2.2 vorgestellt wird. Im weiteren Verlauf kann davon ausgegangen werden, dass die Daten im Array vorliegen.

Um den korrekten Pixel bzw. die richtige Stelle im Array des Images herauszufinden, werden die normalisierten 2D- Koordinaten der Maus benötigt. Diese werden mithilfe der aktuellen Auflösung des Bildschirms berechnet und anschließend werden die normalisierten Koordinaten mit der Größe der Textur multipliziert, um den Pixel, auf dessen Oberfläche sich aktuell der Cursor befindet, ausfindig zu machen. Ist die Position des Pixels klar, kann einfach auf das Array des Images zugegriffen und der Farbwert an der Position ausgelesen werden (s. Abbildung 5.6).

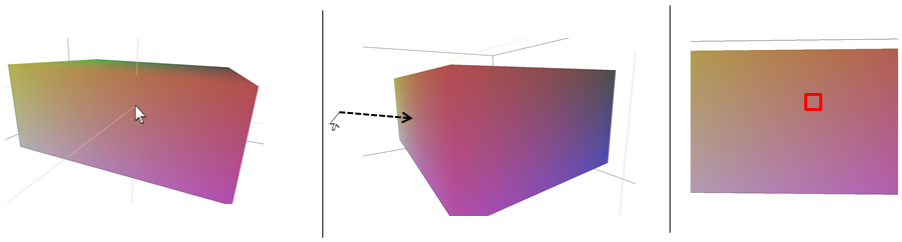

Abbildung 5.6: Links: Maus-Cursor über Volumen in Texturkoordinaten; Mitte: *Picking ray* des Maus-Cursors; Rechts: Getroffener Farbwert in rotem Rechteck

Dieser Farbwerte entspricht einer Texturkoordinate des Volumens und muss nun mit der tatsächlichen Auflösung des Volumendatensatzes verrechnet werden, um an den 3D-Punkt im Volumen zu gelangen. Diese Auflösung ist in den Messdaten der Seismik enthalten und abhängig von der Art der Messung. Um eine allgemeingültige Größenordnung zu erhalten, gibt es die sog. `BIN_SIZE`, welche der Schlüssel ist, um die Auflösung des Datensatzes in Meter umzurechnen. Dieser Schlüssel unterscheidet sich je nach Datensatz. Im Datensatz, der in dieser Arbeit verwendet wurde beträgt er 14,98 in x-Richtung (*Inline*), 15,00 in y-Richtung (*Crossline*) und 1,00 in z-Richtung (*Timeline*). Um an die Größe des Datensatzes zu gelangen, muss die Auflösung mit diesem Schlüssel multipliziert werden. Eine Erleichterung ist es, dass alle Positionierungen im Weltkoordinatensystem der Szene dem Maßstab in Metern angepasst sind, jedoch mit einem Unterschied: Der Ursprung $(0, 0, 0)$ befindet sich in der Mitte der Szene. Dahingegen hat das Volumen den Punkt $(0, 0, 0)$ in der vorderen oberen rechten Ecke und den Punkt $(1, 1, 1)$ diagonal gegenüber. Es muss also eine Transformation über die Hälfte der Volumenauflösung stattfinden, die jeden Punkt im Intervall $[0, 1]$ auf einen Punkte im Intervall $[-0.5, 0.5]$ abbildet. Nach dieser Berechnung kann der 3D-Cursor auf den soeben berechneten Punkt gesetzt werden.

Somit erscheint der 3D-Cursor nun immer auf dem ersten sichtbaren Sample der Volumendaten. Da es dadurch so scheint, als würde der 3D-Cursor magnetisch von den Volumendaten angezogen werden, kann diese Methode auch als *cursor snapping* bezeichnet werden.

Dadurch, dass die Schnittpunktberechnung an diesem Punkt sowohl auf 2D-Slices als auch im dVR funktioniert, kann die erste Anforderung der Anforderungsanalyse aus Kapitel 3 als erfüllt angesehen werden.

Verfahren zur Beschleunigung der Methode

Wie oben beschrieben liegt der Verlust an Performance (die Framerate der Software sinkt auf 6FPS, was eine präzises Arbeiten unmöglich macht) hauptsächlich an der Befüllung des Datenarrays des Images in jedem Frame. Für die Lösung dieses Problems muss also die Größe des Arrays reduziert werden, d.h. eine niedrigere Auflösung der Textur muss gefunden werden. Dies ist durchaus durchführbar, denn für die Benutzung der Methode wird die volle Auflösung nicht benötigt. Es reicht, wenn die Farbwerte der Textur noch klar zu unterscheiden sind, um die Position des 3D-Cursors zu setzen.

Es muss also eine zweite Textur des Volumens in Texturkoordinaten erstellt werden, mit niedrigerer Auflösung, die das Gleiche anzeigt, wie die originale Textur mit hoher Auflösung. Um das zu erreichen, wird zunächst ein Image erstellt, das eine niedrigere Auflösung hat (in diesem Fall beträgt die Auflösung ein Viertel der Ursprünglichen). Nun wird obiges Verfahren so angepasst, dass das FBO, welches das hochauflösenden Bild bisher direkt in das Datenarray des Images gerendert hat, nun das Bild in eine Textur schreibt. Anschließend wird diese hochauflösende Textur mit einer orthographischen Kamera, erneut mittels FBO, in das zuvor erstelle Image mit niedrigerer Auflösung gerendert. Der Vorteil ist nun, dass die Länge des Arrays auch nur noch ein Viertel der ursprünglichen Länge beträgt, was die Methode erheblich beschleunigt. Um das dieses Verfahren zu visualisieren, wurden in Abbildung 5.7 die beiden Texturen beispielhaft auf zwei quad-Geometrien dargestellt

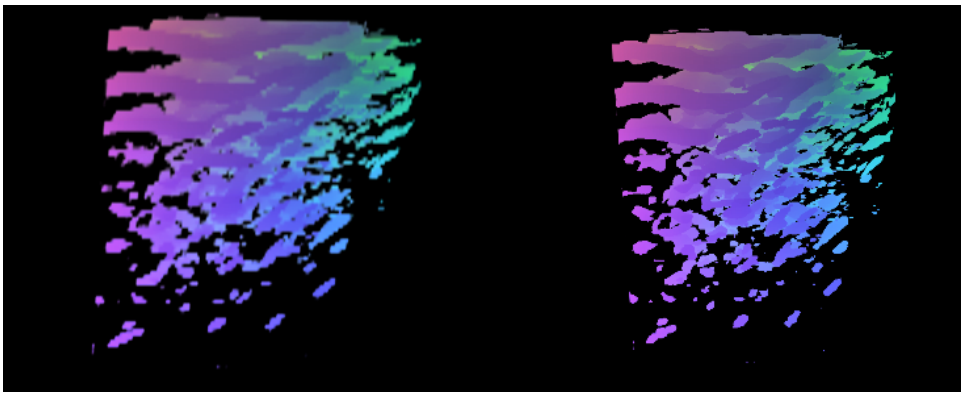

Abbildung 5.7: Volumen in Texturkoordinaten. Links: Niedrige Auflösung; Rechts: Hohe Auflösung

Der Zugriff auf die Farbwerte an der Stelle der Position der Maus, sowie obige Umrechnung der Farbwerte in eine 3D-Position im Raum kann damit nun über die niedrig aufgelöste Textur stattfinden.

Mit diesem Verfahren konnte die Methode bereits ein gutes Stück beschleunigt werden: Ist die *cursor snapping* Methode deaktiviert, läuft die Software bei einer relativ konstanten Framerate von 60 FPS. Wird die Methode aktiviert, bricht die Framerate bei der Variante ohne Beschleunigung auf 6 FPS ein (Framedauer: 157,62 ms). Bei der Variante mit Beschleunigung hält sich die Framerate hingegen bei soliden 45 FPS (Framedauer: 22,47 ms).

Eine weitere Idee, um die Methode zu beschleunigen und ein präziseres Arbeiten mit ihr zu ermöglichen, war es, die Mausposition in einer Textur zu verfolgen. Das funktioniert, indem die Textur mit transparenten Werten befüllt wird und nur die Pixel an der aktuellen Mausposition einen sichtbaren Farbwert erhalten. Der Gedanke dahinter ist dieser, dass die Textur für die Verfolgung der Maus mit der niedrig aufgelöste Textur des Volumens in Texturkoordinaten abgeglichen werden soll, so dass ausschließlich der Teil des Volumens auf der Textur dargestellt wird, über dem sich aktuell der Mauszeiger befindet; denn das allein ist die relevante Region, um die 3D-Position des Cursors zu berechnen.

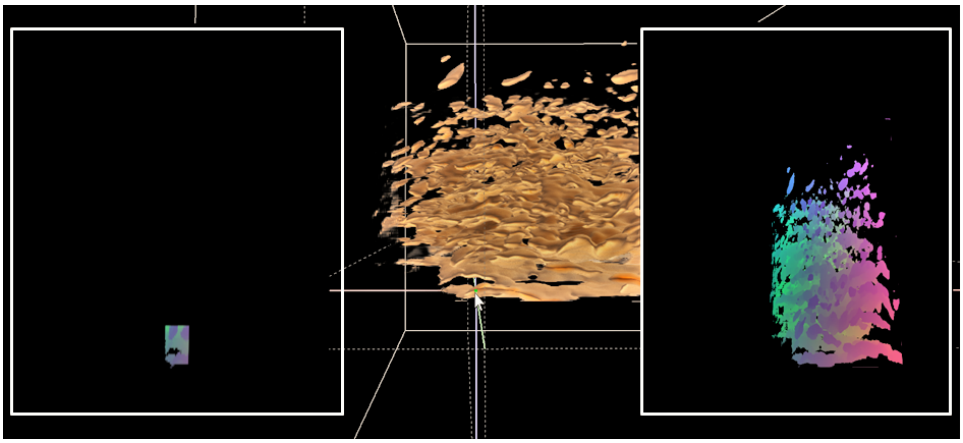

Abbildung 5.8: Links: Ausschnitt von niedrig aufgelöster Textur an aktueller Mausposition; Rechts: Hochauflösende Textur des ganzen Volumens

Für die Verfolgung der Mausposition auf der Textur wird zunächst eine quad-Geometrie erstellt, die den gesamten Bildschrim ausfüllt, aber für den Anwender der Software nicht sichtbar ist (die Sichtbar- und Unsichtbarkeit von bestimmten Objekten in der Szene wird im folgenden Kapitel 5.2.2 erklärt). Anschließend wird für die Geometrie noch eine Textur mitsamt Image initialisiert. Das *picken* dieser Geometrie erfolgt, wie schon in Kapitel 5.1 beschrieben, mit einem Schnitttest zwischen Strahl und Geometrie. Dieser Schnittest liefert den Schnittpunkt in den lokalen Koordinaten der Geome-

trie. Die Koordinaten müssen anschließend in Texturkoordinaten umgerechnet werden, damit ein Zugriff über das Image in die Textur erfolgen kann. Ganz ähnlich wie im vorangegangen Kapitel, kann dann anhand der aktuellen und normalisierten Position der Maus ein beliebiger Farbwert in die Textur geschrieben werden. Eine Besonderheit hierbei ist, dass nach jedem Frame die letzten Einträge aus dem Array des Images gelöscht werden, damit in jedem Frame nur die aktuelle Mausposition auf der Textur angezeigt wird. In Abbildung 5.8 ist dies nochmal zu sehen (zur Veranschaulichung wurde im rechten Bild nicht nur die Pixel unterhalb des Mauszeigers dargestellt, sondern zusätzlich noch weitere benachbarte Pixel damit eine größere Fläche sichtbar ist).

Das komplette Beschleunigungsverfahren ist in Abbildung 5.9 noch einmal grafisch dargestellt.

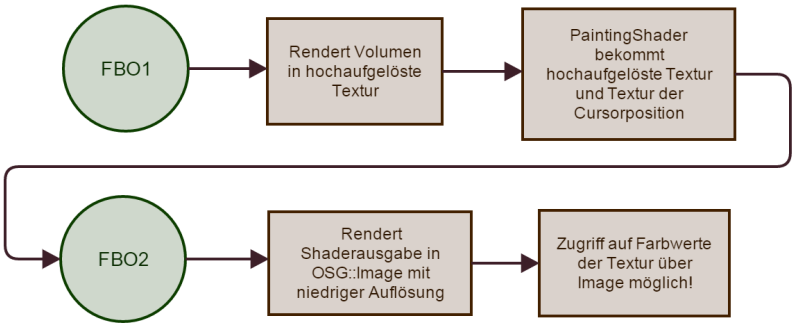

Abbildung 5.9: Verfahren zur Beschleunigung der Methode

Verwendung von OSG Camera-Masks

Ein thematischer Aspekt, der bei der Entwicklung dieser Methode noch eine Rolle gespielt hat, ist die Verwendung von sog. *Camera-Masks*. In OSG ist es möglich, solche Masks zu erstellen. Sie dienen dazu, für bestimmte *Nodes* des Szenegraphen individuell festzulegen, ob diese von einer Kamera „gesehen“ werden oder nicht. Die Masks repräsentieren jeweils einen einzigen Hexadezimalwert; um mehrere parallel zu benutzen, müssen sich diese in der Bitreihenfolge voneinander unterscheiden.

Im Falle dieser Methode ist es beispielsweise so, dass die Geode, welche Volumendaten beinhaltet, noch weitere Nodes als Kinder angehängt hat.

Diese sollen aber von der Texturkamera, die das Volumen in Texturkoordinaten darstellt, nicht gerendert werden, sondern nur die Geode für die Volumendaten. Um das zu erreichen, werden dieser Geode mit dem Befehl `setNodeMask(MAIN_CAMERA_MASK | SNAP_PRERENDER_MASK)` zwei Camera-Masks zugeteilt, was bedeutet, dass sie für beide Camera-Masks sichtbar ist. Gleichzeitig wird über den Befehl `setCullMask` der Kamera mitgeteilt, welche Mask sie rendern soll. In diesem Fall bekommt die Hauptkamera die `MAIN_CAMERA_MASK`, da diese Camera-Mask alles beinhaltet, was letztendliche auf dem Bildschrim dargestellt wird, und die Texturkamera erhält die `SNAP_PRERENDER_MASK` damit sie nur die Geode der Volumendaten rendert.

Gleiches passiert bei der Erstellung der quad-Geometrie für die Beschleunigung der Methode (s.o). Um sicherzustellen, dass diese Geometrie nicht sichtbar ist und dem Anwender die Sicht versperrt, wird diesem Node ebenfalls die `SNAP_PRERENDER_MASK` zugewiesen und die orthographische Kamera wird für diese Camera-Mask freigeschaltet. Hierbei ist jedoch noch zu beachten, dass der Schnitttest mit dieser quad-Geometrie bei einer angewandten Camera-Mask nun nicht mehr ohne Weiteres durchgeführt werden kann, da er standardmäßig nur auf sichtbaren Geometrien funktioniert. Stattdessen muss für den *intersection visitor*, der den Schnitttest durchführt, explizit eine sog. *traversal-mask* erstellt werden, welche dem Hexadezimalwert der Camera-Mask entspricht, in der sich die Geometrie befindet, mit welcher der Schnitttest durchgeführt werden soll. Das Ganze ist in folgender Abbildung noch einmal visualisiert.

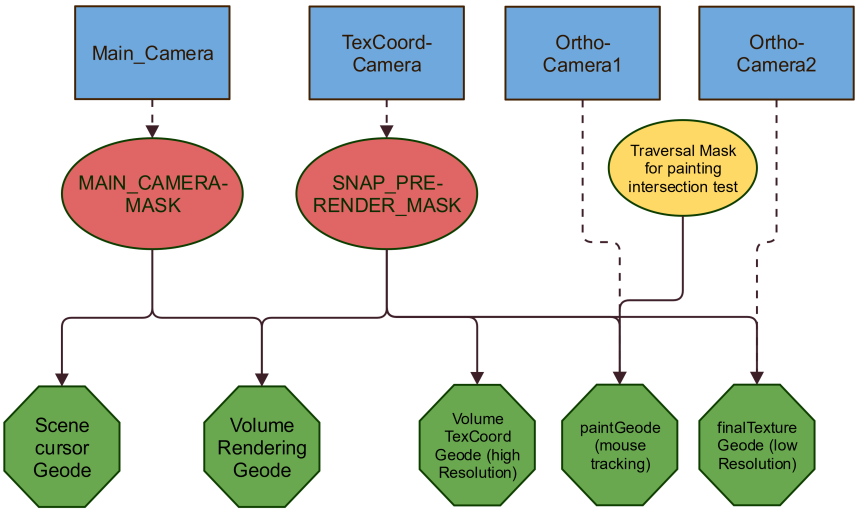

Abbildung 5.10: Die Hauptkamera rendert das Volumen und den 3D-Cursor; die Texturkamera rendert das Volumen in Texturkoordinaten, jedoch nicht sichtbar; außerdem nicht sichtbar: die painting Geode sowie die beiden Geodes für die FBOs

Es sind nun alle grundlegenden, für die entwickelte Methode notwendige Implementierungen aufgezeigt worden. Der komplette Ablaufplan der Methode ist noch einmal übersichtlich als Aktivitätsdiagramm dargestellt, im Anhang zu finden.

5.2.3 Softwarearchitektur

Nachdem die Implementierung der grundlegenden Funktionen soweit erklärt wurde, soll im folgenden Unterkapitel kurz der allgemeine Aufbau der Software, sowie das Zusammenspiel zwischen den bisherigen Klassen der Software und den neu entwickelten Klassen für die Methode, erläutert werden.

Die Software SeisViz3D ist in einzelne Module aufgeteilt (s. Abbildung 5.11), welche für unterschiedliche Aufgaben innerhalb der Software zuständig sind. Das *data management* ist verantwortlich für das Laden der seismischen Daten und beinhaltet ein Im- bzw. Exporter plugin. Das *network management* kümmert sich um die Kommunikation zwischen der Hauptanwendung und der *tablet application*, denn die Software lässt sich auch über ein Tablet steuern und zudem wurden auf dem Tablet auch einige Interakionsmöglichkeiten übernommen [53]. Das *geometry management* verwaltet die Geometrien, die innerhalb der Software erstellt werden. Der *volume renderer* ist für die Darstellung der Volumendaten verantwortlich und die *main application* ist das Hauptpaket und führt alle vorher aufgeführten Pakete zusammen. Außerdem befinden sich dort die Klassen für die unterschiedlichen Interaktionsmöglichkeiten der Software.

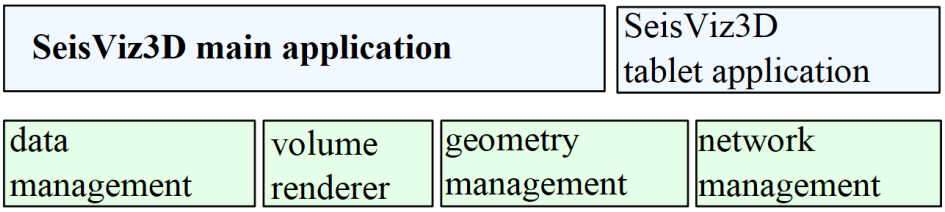

Abbildung 5.11: Grobarchitektur der Softwarepakete in SeisViz3D

Für die entwickelte Methode wurden hauptsächlich Änderungen in den Paketen: *volume renderer*, *geometry management* und natürlich in der *main application* durchgeführt. Es wurde dabei versucht, die Teile der Methode, welche auf Eingaben des Nutzers reagieren, in der *main application* zu implementieren, während die Funktionen, die sich um die Visualisierung kümmern und meist im Hintergrund ausgeführt werden, im *volume renderer* ihren Platz haben.

Dies ist wichtig für die Übersichtlichkeit der Software und die kontextuelle Zusammenhörigkeit einzelner Funktionen.

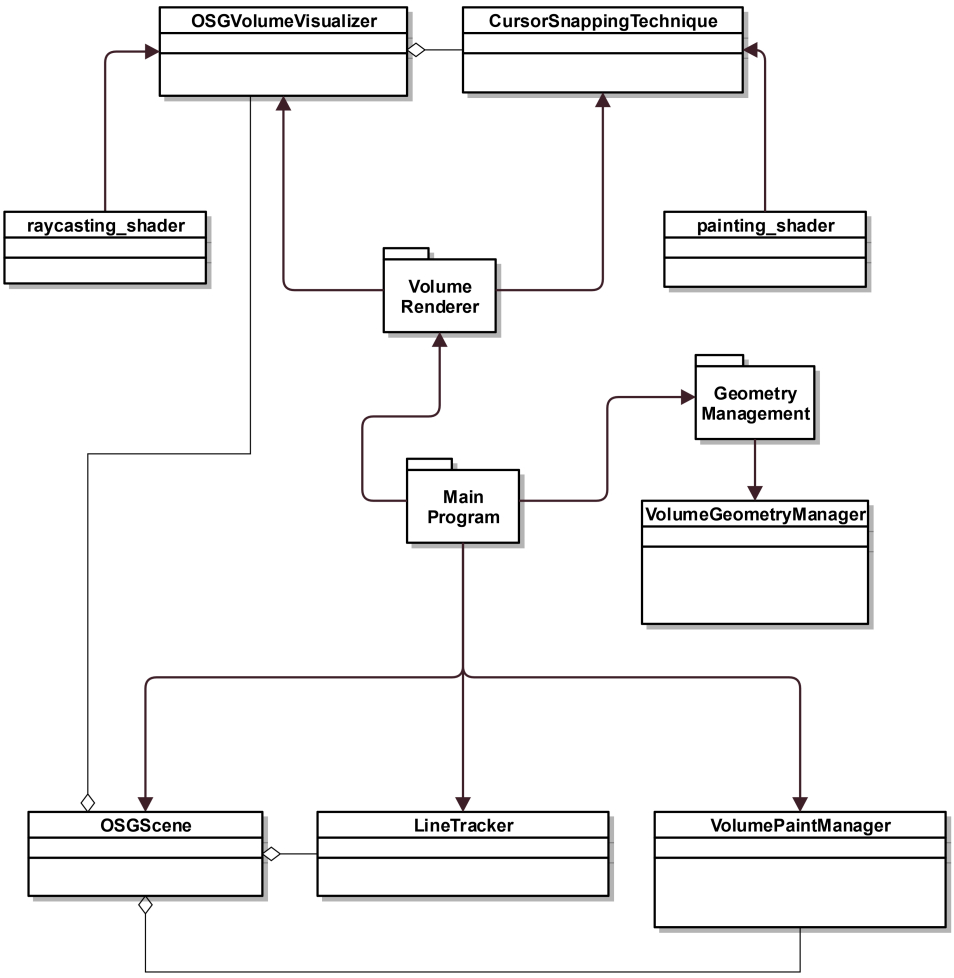

Abbildung 5.12: Klassendiagram für entwickelte Methode

In Abbildung 5.12 ist eine Diagramm zu sehen, das das Zusammenspiel der einzelnen für die entwickelte Methode notwendigen Klassen darstellt. In der *main application* sind zwei neue Klassen dazugekommen, die sich zum einen um die Neupositionierung des 3D-Cursors bzw. um die Verfolgung der Mausposition auf der Textur kümmert (*VolumeSnappingManager*) und zum anderen für die vereinfachte Erstellung der Störungsgeometrien verantwortlich ist (*LineTracker*)(vgl. nächstes Kapitel 6). Im *volume renderer* ist eine neue Klasse hinzugekommen (*CursorSnappingTechnique*), welche die zuvor beschriebene Technik des *cursor snappings*, implementiert. Diese Klasse ar-

beitet sehr eng mit bereits vorhandenen Klassen des Renderers zusammen, die für die Darstellung der Volumendaten verantwortlich sind und die Shader einbinden (*OSGVolumeVisualizer*).

Die einzelnen Ergebnisse der unterschiedlichen Methoden innerhalb der Klasse laufen letztendlich in einer Klasse der Hauptanwendung zusammen (*OSGScene*), welche dafür zuständig ist, den 3D-Cursor in der Szene zu positionieren. Eine detailliertere Version des Diagramms aus Abbildung 5.12 mitsamt Methoden ist nochmal im Anhang zu finden.

6 | Vereinfachte Erstellung von Störungsgeometrien

Im Folgenden wird das zweite Teilpaket der Implementierung vorgestellt; ein Verfahren, welches mit der zuvor implementierten Methode schnell und einfach Geometrien im Volumen zur Analyse und Interpretation seismischer Störungen erstellt.

6.1 Idee

Die grundlegenden Idee hinter diesem Verfahren ist es, dass durch die neu hinzugekommene Möglichkeit, den 3D-Cursor direkt automatisch auf den Daten zu platzieren, die sichtbaren Strukturen in den Daten nun einfach mit dem aktuellen Eingabegerät „nachgezeichnet“ (Stichwort: *painting*) werden können. Dadurch soll es für den Experten möglich sein, entlang der sichtbaren Strukturen der Volumendaten Linien zu zeichnen. Diese sollen anschließend automatisch interpoliert werden, wodurch in regelmäßigen Abständen Geometriepunkte auf den Linien erzeugt werden. Der Abstand der Punkte zueinander soll dabei variabel gehalten werden und vom Anwender selbst bestimmbar sein. Bei mindestens zwei solcher eingezeichneter Linien soll mittels Triangulation eine Fläche zwischen den Punkten aufgespannt werden, wodurch die Störungsgeometrie mit nur zwei *painting*-Interaktionen des Nutzers, sehr intuitiv erstellt werden kann.

6.2 Implementierung

Das Verfahren der intuitiven Erstellung von Störungsgeometrien wird im Folgenden auch auch als *line tracking* bezeichnet, da die Geometrie automatisch entlang der gezeichneten Linie erzeugt wird. Ein Großteil der Implementierung wurde bereits durch das *cursor snapping* Verfahren abgenommen, weil dadurch der 3D-Punkt auf den Volumendaten bekannt ist.

Die Funktion wird aktiviert, indem der Modus für die Störungsinterpretation ausgewählt wird (siehe nächstes Unterkapitel 6.3). Anschließend kann der Anwender bei gedrückter Taste und Bewegung des Mauszeigers, eine Linie entlang der Positionen, an denen sich der 3D-Cursor befindet, zeichnen.

Die Erstellung der Linie erfolgt dabei so, dass beim Drücken der Taste der initiale Vertex-Punkt der Linie erzeugt wird, beim Gedrückt-halten der Taste werden weitere Vertex-Punkte erzeugt und beim Loslassen der Taste werden alle Vertex-Punkte, die während eines einzigen Tastendrucks erzeugt wurden, mit einem *LineSegement* [37] verbunden. Beim nächsten Druck auf die Taste wird eine weitere Linie über dasselbe Verfahren erstellt.

Anschließend wird diese Linie über die Anzahl der Vertex-Punkte interpoliert und in regelmäßigen Abständen (je nach Anzahl der zu erstellenden Geometriepunkte für die Störungsgeometrie, welche vom Experten festgelegt wurde; niedrige Anzahl bedeutet große Abstände und umgekehrt) an den Vertex-Punkten Geometriepunkte erzeugt. Diese werden dann mit all ihren benachbarten Geometriepunkten mittels Triangulation [21] zu einer Fläche zusammengefasst.

Durch die Erstellung einer Linie und das Erzeugen einer Fläche mittels Triangulation, können Anforderungen 2 & 3 als erfüllt angesehen werden. Die Linie wird mit einer einzigen Interaktion des Nutzers (Gedrückt-halten der Taste) genau auf den Daten erstellt, die für ihn sichtbar sind. Die Punkte der Linie werden anschließend trianguliert, wodurch Dreiecksgeometrien im Volumendatensatz entstehen.

6.3 Integration in SeisViz3D Software

In der SeisViz3D-Software kann der Nutzer zwischen unterschiedlichen Modi auswählen, was die Interaktion mit den Daten betrifft (s. Abbildung 6.1). Die beiden linken Modi (*Navigation* & *Interpretation*) sind für die Exploration und Navigation in den seismischen Daten vorgesehen; sie verwenden weder das *cursor snapping* noch *line tracking*. Der Modus *Annotation* aktiviert das *cursor snapping*. In diesem Modus ist es möglich, Annotationen bzw. Markierungen in Form einer Fahnen-Geometrie in dem Datensatz zu erstellen (s. Abbildung 6.2). Mit deren Hilfe kann sich der Experte charakteristische oder markante Positionen, die ihm während der Interpretation der Daten aufgefallen sind, merken. Die entwickelte Methode unterstützt ihn dabei insofern, als dass er mit dem Cursor des Eingabegerätes nur noch auf die entsprechende Stelle zeigen muss und mit einem Tastendruck auf die *Enter*-Taste die Annotation automatisch direkt für ihn auf den Daten erstellt wird.

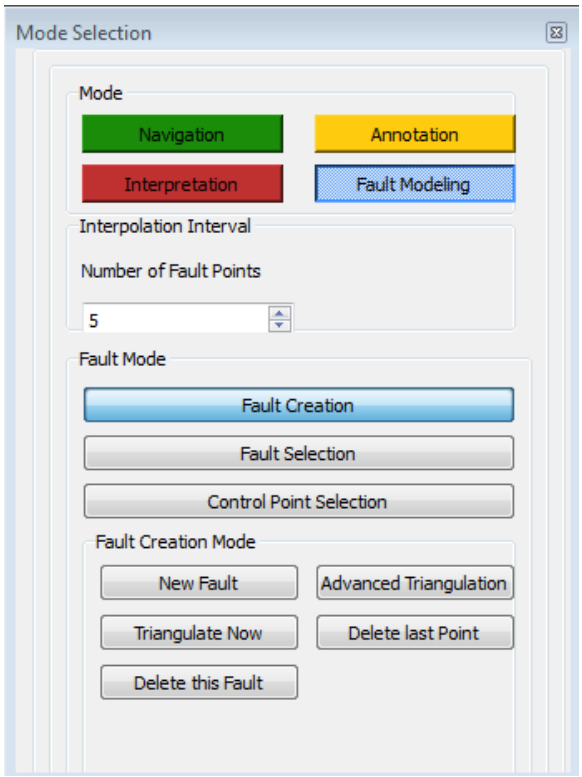

Abbildung 6.1: Menü der unterschiedlichen Interaktionsmethoden

Die letzte Option ist der *Fault Modeling* Modus. Ist dieser Modus aktiviert, können die Geometriepunkte für die Störungsgeometrien per Tastendruck an die aktuelle Stelle des 3D-Cursors gesetzt werden. Mit der Aktivierung dieses Modus werden ebenfalls das *cursor snapping* und *line tracking* aktiviert, was bedeutet, dass alle notwendigen Texturen, Images etc. initialisiert werden. Eine Ausnahme bildet dabei die Kamera für die Visualisierung des Volumens der Texturkoordinaten; diese muss bereits bei Start des Programms initialisiert werden, bleibt aber bis zum Aufruf dieses Modus deaktiviert. Es erscheint außerdem ein neues Feld im Menü, in welches der Anwender einen Wert eintragen kann, der die Anzahl der von der Methode zu erstellenden Geometriepunkte pro Linie angibt.

Mit einem Klick auf den Button *New Fault* können nun die Geometrien ohne weiteres Zutun in die Volumendaten eingezeichnet werden. Die Geometriepunkte werden bei der Erstellung automatisch trianguliert. Mit einem erneuten Klick auf den Button wird eine neue Geometrie in der Software angelegt, was bedeutet, dass neue Geometriepunkte nicht mehr mit den Punkten der letzten Störungsgeometrie trianguliert werden.

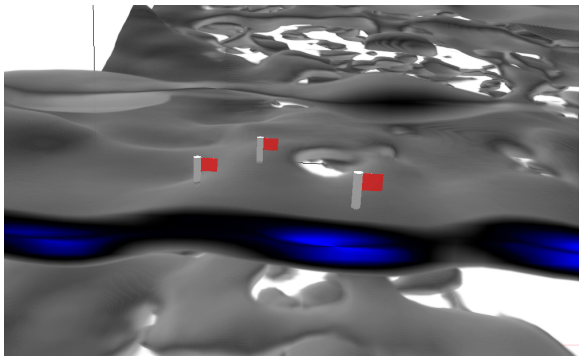

Abbildung 6.2: Annotationen, die mit *cursor snapping* Verfahren erstellt wurden

Bei einer Deaktivierung dieses Modus durch die Aktivierung eines anderen, werden über *Destruktoren* und die Abschaltung der Texturkamera alle angelegten Datentypen freigegeben bzw. deaktiviert, um Rechenleistung zu sparen.

Dadurch, dass das *cursor snapping* bzw. *line tracking* die Interaktionsmethoden *picking* bzw. *painting* verwenden, kann schließlich auch Anforderung 4 der Anforderungsanalyse aus Kapitel 3 als erfüllt betrachtet werden.

Einige Ansichten der fertig entwickelten Methode sind im nächsten Kapitel zu sehen.

7 | Ergebnisse

In diesem Kapitel soll in einigen Abbildungen kurz die finale Methode zur intuitiven Erstellung von Störungsgeometrien in seismischen Volumendatensätzen visuell dargestellt werden.

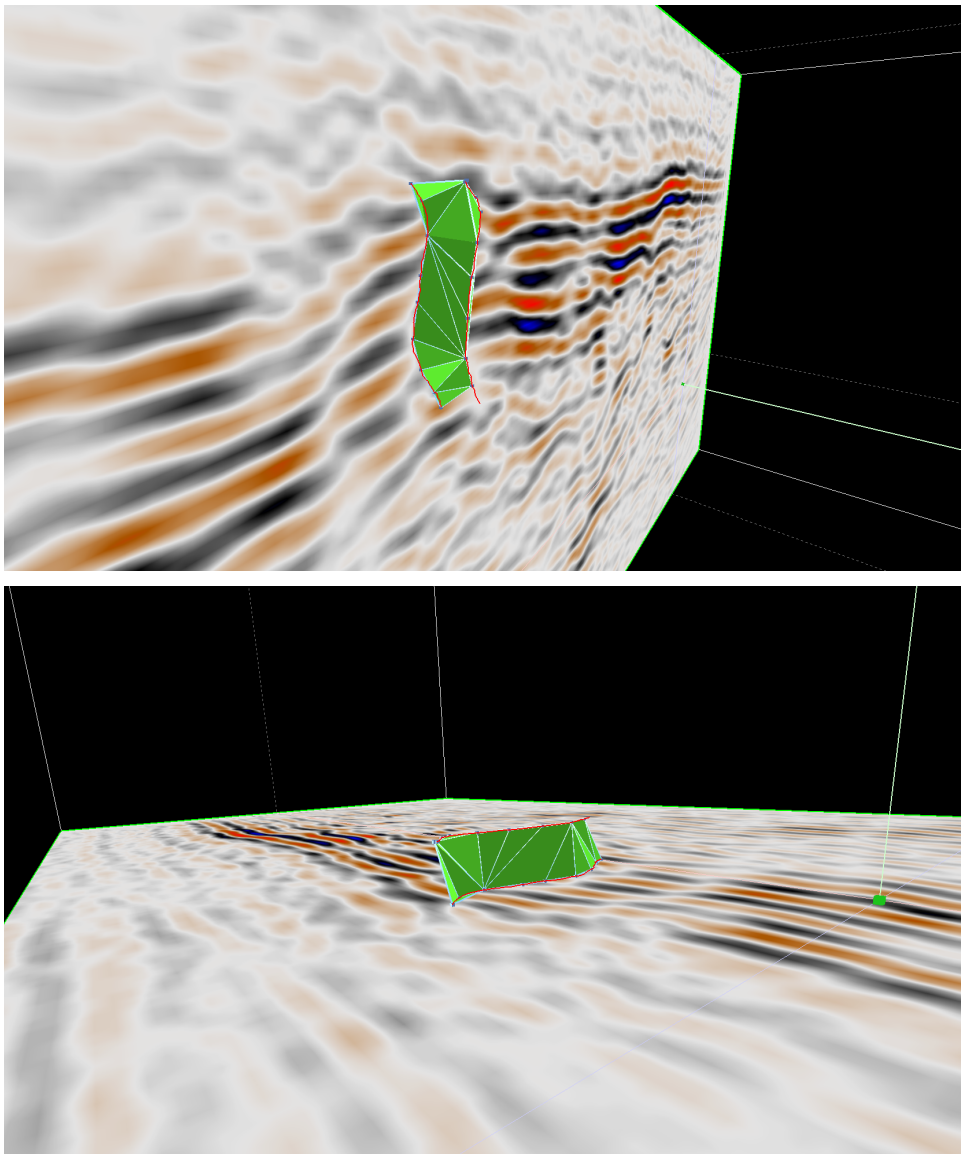

Abbildung 7.1: Erstellung von Störungsgeometrie auf 2D-Slice aus verschiedenen Kameraperspektiven

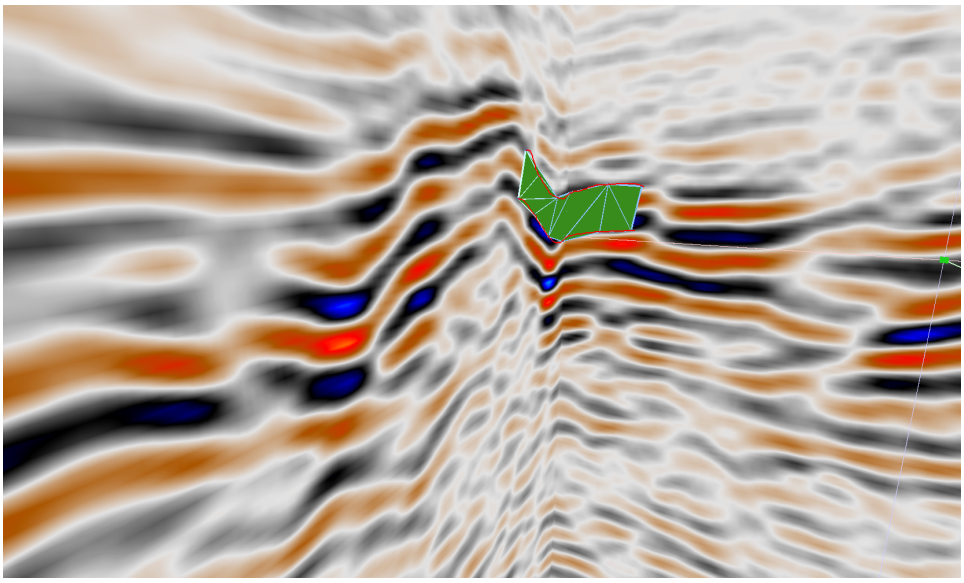

Abbildung 7.2: Erstellung von Störungsgeometrie zwischen Inline und Crossline 2D-Slice

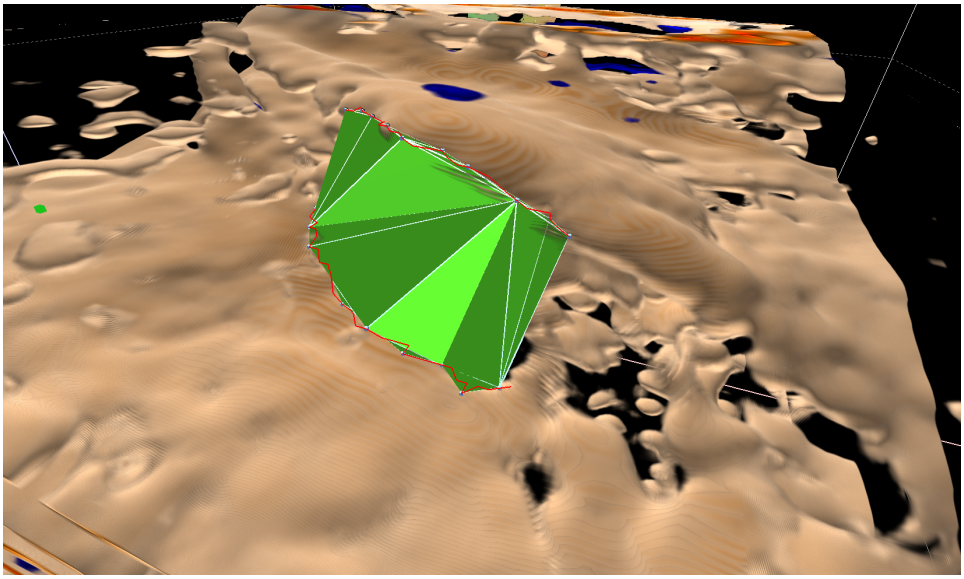

Abbildung 7.3: Erstellung von Störungsgeometrie im direkten Volumenrendering. Die Geometrie verbindet die beiden Bruchstellen miteinander.

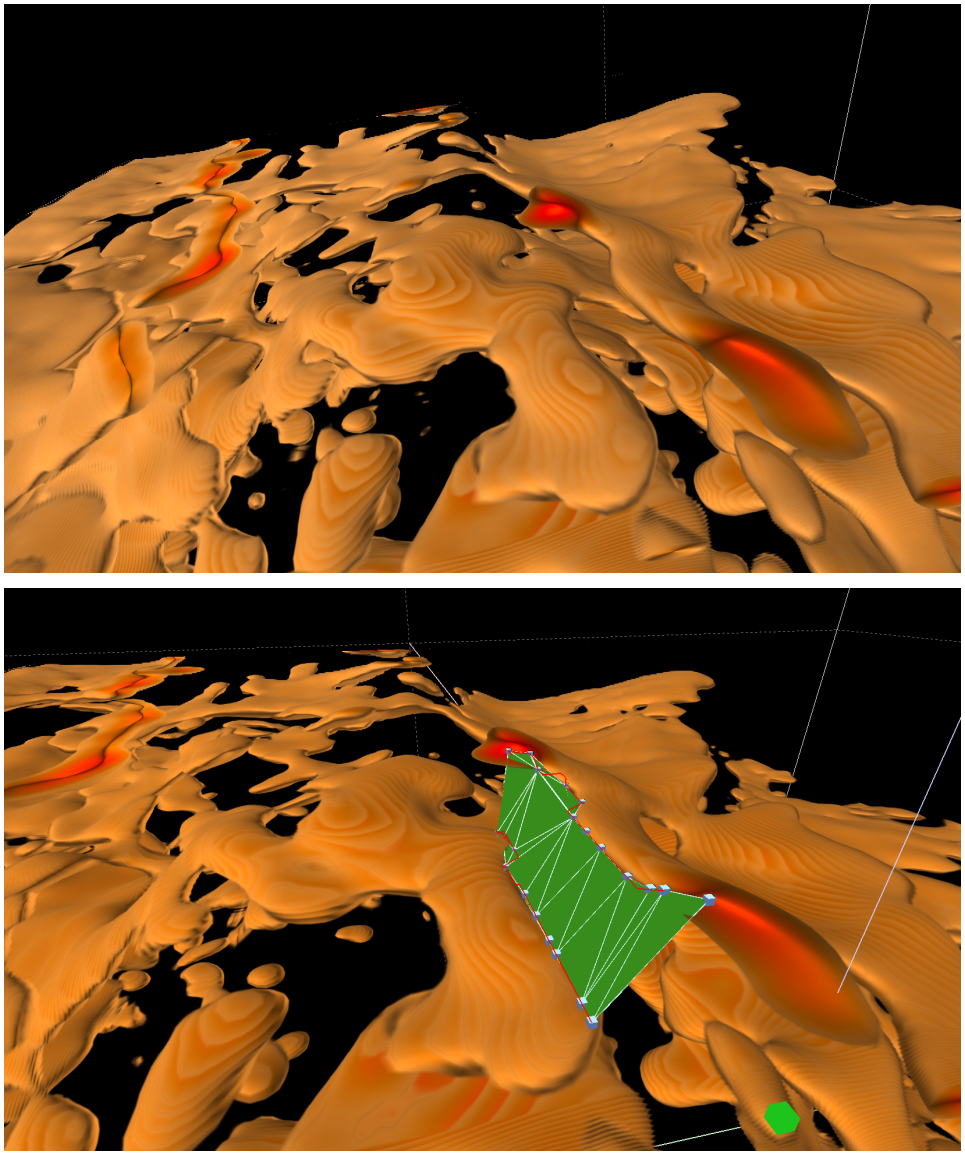

Abbildung 7.4: Erstellung von Störungsgeometrie im direkten Volumenrendering. Oben: Vor der Erstellung der Geometrie; Unten: Nach der Erstellung

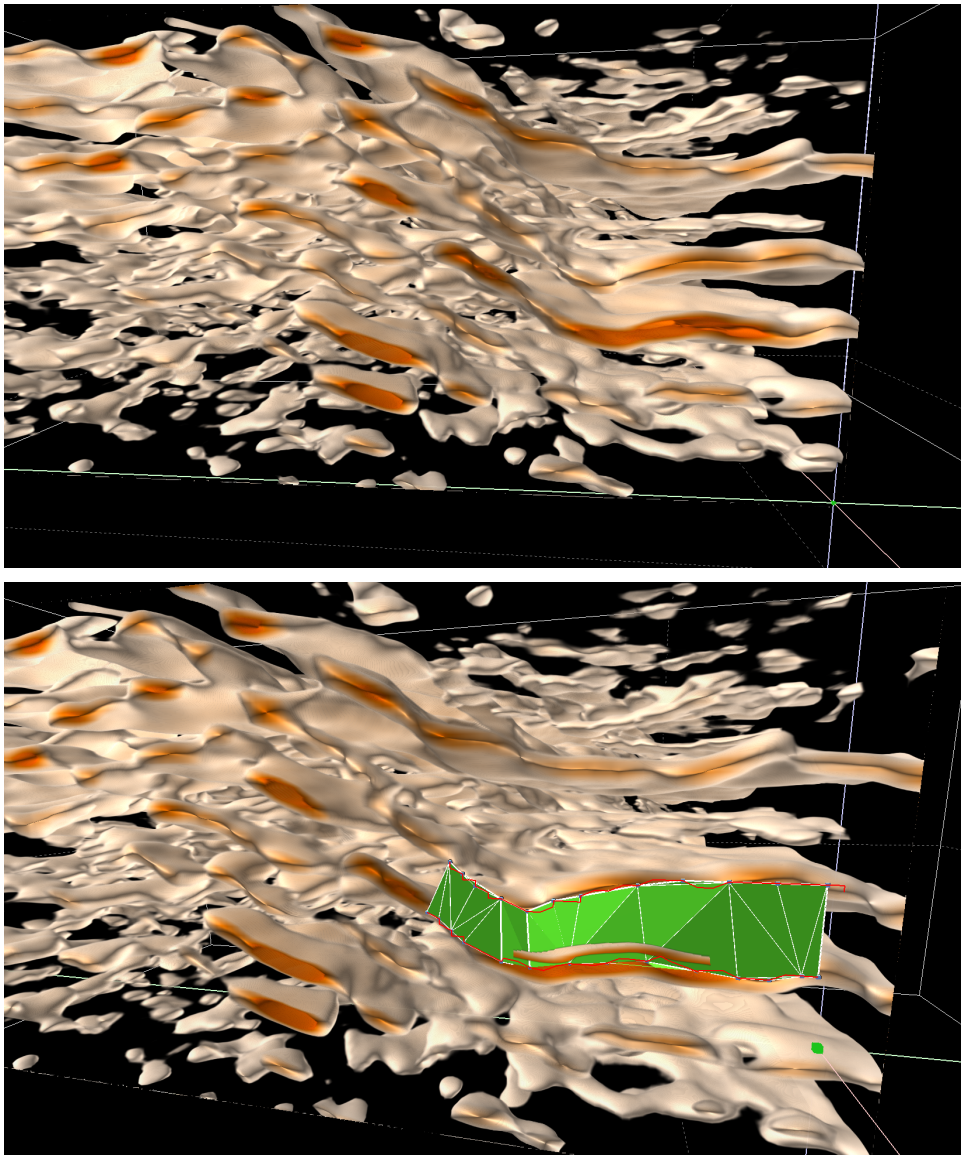

Abbildung 7.5: Erstellung von Störungsgeometrie im direkten Volumenrendering. Oben: Vor der Erstellung der Geometrie; Unten: Nach der Erstellung

8 | Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Methode für die intuitive Erstellung von Störungsgeometrien in seismischen Volumendatensätzen entwickelt. Es wurde dabei zunächst zusammen mit Experten auf diesem Feld eine Anforderungsanalyse erstellt (vgl. Kapitel 3). Die dort geforderten Funktionen sind alle in der entwickelten Methode implementiert und können eingesetzt werden. Es wurde außerdem der Stand der Technik im Bezug auf die Interaktion mit Volumendaten sowie die Funktionen in gängigen seismischen Interpretations-Softwarepaketen erläutert und es wurde gezeigt, dass es für die gewünschte Methode noch keine vorhanden Lösung gibt. Anschließend wurde eine Lösung vorgeschlagen und entwickelt, mit der die geforderte Methode umgesetzt werden konnte.

Natürlich ist das Verfahren noch nicht perfekt und kann verbessert werden. Der größte Kritikpunkt ist immer noch die Performance der Methode im direkten Volumenrendering. Der Zuwachs an Renderarbeit für die GPU hat der zusätzliche Renderpass, der für die Darstellung des Volumens in Texturkoordinaten notwendig ist, zu verantworten. Die CPU hingegen arbeitet jeden Frame an der neuen Erzeugung des *Images* für den Zugriff auf die benötigte Textur. Zwar wurde in dieser Arbeit bereits eine Methode vorgestellt, um das Verfahren etwas zu beschleunigen, dennoch gibt es hier noch weiteren Optimierungsbedarf.

Die Implementierung dieser Methode kann als grundlegende Vorarbeit für weitere Funktionalitäten, die in der Zukunft implementiert werden können, angesehen werden. Eine Funktionalität, welche die SeisViz3D-Software in Zukunft außerdem besitzen soll und auf die entwickelte Methode dieser Arbeit aufbaut, ist das *Advanced Horizon Tracking* [29]. Diese Funktion soll in einem vorher festgelegten Bereich einen Horizont im dreidimensionalen Raum verfolgen, indem entlang dieses Horizontes in alle Richtungen Geometriepunkte erstellt und diese zu einer Fläche zusammengefasst werden. Die Auswahl des Horizontes kann dabei durch die Zuhilfenahme der entwickelten Methode entweder in der Visualisierung mit 2D-Slices oder im dVR stattfinden.

Ein weiterer Ausblick für die Verwendung des *cursor snapping* Verfahrens, wäre es, eine Art „Radierer“ bzw. „Marker“ zu entwickeln, mit dem sich seismischen Daten im dVR ausblenden bzw. hervorheben lassen. Dies würde erreicht werden, indem an der aktuellen Stelle des 3D-Cursors der Amplitudenwert der seismischen Daten ausgelesen und die Transferfunktion an-

hand dieses Wertes automatisch angepasst wird. Die Transferfunktion würde dann entweder diesen Amplitudenwert ausblenden (*Radierer*) oder alle anderen Werte ausblenden, sodass nur noch die Daten mit diesem bestimmten Amplitudenwert angezeigt werden (*Marker*). Diese Technik lässt sich insbesondere im dVR einsetzen, da dort Amplitudenwerte zu sehen sind, die sich eventuell nicht über das gesamte Volumen erstrecken und dadurch in manchen Sektionen gar nicht vorhanden sind und so übersehen werden können. Eine Methode, welche dem zuvor erwähnten Radier-Verfahren sehr ähnlich ist, wurde bereits von Guo et al. [26] für medizinische Daten entwickelt (s. Abbildung 8.1). Allgemein bietet die Medizin sehr viele Bereiche, die

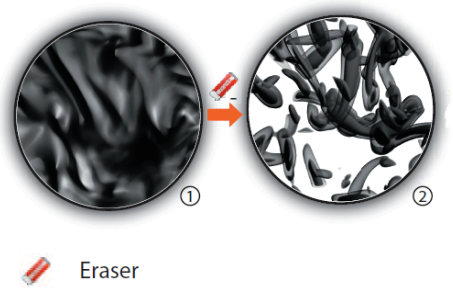

Abbildung 8.1: Links: Unübersichtliche medizinische Daten ohne angewandte Transferfunktion; Rechts: Intelligent angewandte Transferfunktion mittels automatischem Ausblenden eines bestimmten Datenbereiches zur Isolation von wichtigen Strukturen. Entnommen aus [26]

hier entwickelte Methode auch in der Analyse von medizinischen Daten zu verwenden. Hier könnte sich beispielsweise vorgestellt werden, mit der Technik der Modellierung von Geometrien in Volumendaten, Knochenplatten zur Verbindungen zweier Knochenfragmente direkt in der Volumenvisualisierung des Knochens zu modellieren. So hätte man die realistische Größe und kann kleinste Anpassungen in der virtuellen Umgebung vornehmen. In einem weiteren Schritt könnte diese Geometrie dann in ein 3D-Objekt (*Mesh*) umgewandelt und exportiert werden, um anschließend einen 3D-Druck davon zu erzeugen. Ob das für den Druck verwendete Material für das Innere des menschlichen Körpers geeignet ist, gilt es dabei herauszufinden.

Abbildungsverzeichnis

2.1 Ablauf der Ölmigration [57] 6
2.2 Aufnahme von seismischen Daten auf dem Wasser [6] 7
2.3 Aufnahme von seismischen Daten auf dem Land [6] 8
2.4 Aufnahme von seismischen Daten mit Hydrophon Array [33] . 9
2.5 Oben: Blattverschiebung, Mitte: Abschiebung, Unten: Aufschiebung [24] 10
2.6 Links: Störungslinie in einer 2D-Sektion; Rechts: Störungsgeometrie bestehend aus mehreren Störungslinien. Entnommen aus [31] 12
2.7 Darstellung von Volumendaten mittels 2D-Slices (inline und crossline). In der Mitte sind die Daten als Volumen mittels direktem Volumenrendering dargestellt [11] 13
2.8 Ein diskreter Räumlicher Datensatz nach Rasterisierung. Links ist der vollständige Datensatz dargestellt. Rechts ist zum besseren Verständnis der Datensatz nur zum Teil dargestellt um die Idee zu visualisieren, die einen Voxels als Würfel beschreibt, der eine Einheit im dreidimensionalen Raum repräsentiert. Entnommen aus [18] 14
2.9 Absorption des Punktes an der Stelle $t = d$. Entnommen aus [27] 15
2.10 Front-to-Back Compositing - Die finale Farbe wird von vorne nach hinten berechnet [2] 16
2.11 Raycasting-Verfahren. Entnommen aus [27] 17
2.12 Links: Editor zur Erstellung von eindimensionalen Transferfunktionen; Rechts: Adaptiertes Volumenrendering 19
2.13 Editor für mehrdimensionale Transferfunktionen mittels Paralleler Koordinaten. Entnommen aus [8] 20
2.14 Links: Die Umrisse eines Lungenknotens sind auf der 2D-Slice klar zu erkennen und lassen sich vom Rest abgrenzen. Rechts: Im direkten Volumenrendering lässt sich besser die Verwachsung desselben Knotens mit dem Bronchialsystem abschätzen. Entnommen aus [45] 21

4.1 Links: Bei zu niedrigem Schwellwert werden Daten ausgewählt, die nur ganz schwach sichtbar sind, statt klar erkennbarer Regionen; Rechts: Ein zu hoher Schwellwert ignoriert womöglich wichtige Strukturen in den Daten, die nicht komplett opak sind. Entnommen aus [55] 27

4.2 Auf der x-Achse ist das Sampling des picking-rays dargestellt und auf der y-Achse die Opazität. Der höchste Opazitätssprung findet im Intervall zwischen b und c statt. An dieser Stelle werden die Daten vermutet, die der Nutzer auswählen möchte. Entnommen aus [55] 28
4.3 Links: Medizinische Volumendaten eines Kopfes mit angewandter Transferfunktion; Mitte: Kontur des Gehirns wird nachgezeichnet; Rechts: Gehirn wird von umliegenden Daten (Knochen etc.) isoliert dargestellt. Entnommen aus [38] . . . 29
4.4 Software Petrel. Links: Horizontgeometrien im Volumen; Rechts: Markierte Horizonte auf der 2D-Slice mit sichtbarem Versatz [49] 30
4.5 Automatisch erstellte Störungen anhand getrackter Horizonte aus OpendTect [35] 31
4.6 Software GeoTeric. Links: Markierter Bereich auf 2D-Slice; Rechts: Verfolgung des Horizonts im dVR [23] 32
4.7 DecisionSpace Desktop: Störungsgeometrie wird anhand eingezeichneter Linien automatisch erstellt [28] 32
4.8 Links: 3D-Cursor vor den Daten (Falsch); Mitte: 3D-Cursor genau auf den Daten, da halb verdeckt (Richtig); Rechts: 3D-Cursor hinter den Daten (Falsch) 33

5.1 *Picking-ray* auf 2D-Slice 35
5.2 *Picking-ray* auf direktem Volumenrendering 37
5.3 Würfelgeometrie in Texturkoordinaten. Links: front faces; Mitte: back faces; Rechts: Ergebnis von Mitte - Links. Entnommen aus [27] 38
5.4 Volumendatensatz in Texturkoordinaten. Links: Ohne angewandte Transferfunktion; Rechts: Mit angewandter Transferfunktion (Volumen vor schwarzem Hintergrund) 39
5.5 Oben: Saubere Transferfunktion bei $T = 0.0$; Mitte: Unsaubere Transferfunktion bei $T = 0.0$; Unten: Unsaubere Transferfunktion bei $T = 0.05$ 42
5.6 Links: Maus-Cursor über Volumen in Texturkoordinaten; Mitte: *Picking ray* des Maus-Cursors; Rechts: Getroffener Farbwert in rotem Rechteck 44
5.7 Volumen in Texturkoordinaten. Links: Niedrige Auflösung; Rechts: Hohe Auflösung 45
5.8 Links: Ausschnitt von niedrig aufgelöster Textur an aktueller Mausposition; Rechts: Hochauflösende Textur des ganzen Volumens 46
5.9 Verfahren zur Beschleunigung der Methode 47

5.10 Die Hauptkamera rendert das Volumen und den 3D-Cursor; die Texturkamera rendert das Volumen in Texturkoordinaten, jedoch nicht sichtbar; außerdem nicht sichtbar: die painting Geode sowie die beiden Geodes für die FBOs 48
5.11 Grobarchitektur der Softwarepakete in SeisViz3D 49
5.12 Klassendiagram für entwickelte Methode 50

6.1 Menü der unterschiedlichen Interaktionsmethoden 54
6.2 Annotationen, die mit *cursor snapping* Verfahren erstellt wurden 55

7.1 Erstellung von Störungsgeometrie auf 2D-Slice aus verschiedenen Kameraperspektiven 57
7.2 Erstellung von Störungsgeometrie zwischen Inline und Crossline 2D-Slice 58
7.3 Erstellung von Störungsgeometrie im direkten Volumenrendering. Die Geometrie verbindet die beiden Bruchstellen miteinander. 58
7.4 Erstellung von Störungsgeometrie im direkten Volumenrendering. Oben: Vor der Erstellung der Geometrie; Unten: Nach der Erstellung 59
7.5 Erstellung von Störungsgeometrie im direkten Volumenrendering. Oben: Vor der Erstellung der Geometrie; Unten: Nach der Erstellung 60

8.1 Links: Unübersichtliche medizinische Daten ohne angewandte Transferfunktion; Rechts: Intelligent angewandte Transferfunktion mittels automatischem Ausblenden eines bestimmten Datenbereiches zur Isolation von wichtigen Strukturen. Entnommen aus [26] 62
2 Aktivitätsdiagramm für *cursor snapping* Methode 75
3 UML-Diagramm der entwickelten Methode 76

Literaturverzeichnis

[1] Big data analytics, fraunhofer gesellschaft. http://www.bigdata.fraunhofer.de/. Abgerufen am 29.05.2016.

[2] Front-to-back compositing. http://upload.wikimedia.org/wikipedia/commons/thumb/3/31/Compositing_Front_To_Back.svg/200px-Compositing_Front_To_Back.svg.png. Abgerufen am 10.06.2016.

[3] Opengl. https://www.opengl.org/. Abgerufen am 29.05.2016.

[4] Openscenegraph. http://www.openscenegraph.org/. Abgerufen am 08.06.2016.

[5] Basics of simd programming. https://www.kernel.org/pub/linux/kernel/people/geoff/cell/ps3-linux-docs/CellProgrammingTutorial/BasicsOfSIMDProgramming.html, 2013. Abgerufen am 15.05.2015.

[6] Deutsche Erdöl AG. Seismic exploration. http://www.dea-group.com/en/technology/exploration/seismik, 2016. Abgerufen am 03.06.2016.

[7] Harlen C. Batagelo and Wu Shin-Ting. What You See is What You Snap: Snapping to Geometry Deformed on the GPU. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 81–86, New York, NY, USA, 2005. ACM.

[8] Jan Christoph Beutgen. Interaktive Volumenvisualisierung mit multidimensionalen Transferfunktionen. Bachelor's thesis, 2015.

[9] Manfred Bogen. SeisViz3D. https://www.seisviz3d.de. Abgerufen am 08.06.2016.

[10] Manfred Bogen. Vrgeo - innovation for the oil and gas exploration. http://www.vrgeo.org/. Abgerufen am 08.06.2016.

[11] Manfred Bogen and Jan Christoph Beutgen. Vrgeo demo viewer application. http://www.vrgeo.org/, 2016. Abgerufen am 07.06.2016.

[12] Kate Brodie, Douglas Fettes, Ben Harte, and Rolf Schmid. Structural terms including fault rock terms. *Online at www. bgs. ac. uk/scmr. Last visited in January*, 8:2013, 2007.

[13] Microsoft Coorporation. Directx 9.0 programmer's reference. https://msdn.microsoft.com/en-us/library/windows/desktop/bb219839(v=vs.85).aspx, 2004. Abgerufen am 10.06.2016.

[14] Boris Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[15] OpenGL SDK Documentation. Vertex attributes. `https://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/attributes.php`. Abgerufen am 21.06.2016.

[16] Point Cloud Library Documentation. Fast triangulation of unordered point clouds. `http://pointclouds.org/documentation/tutorials/greedy_projection.php`. Abgerufen am 10.06.2016.

[17] Duden. Horizont. `http://www.duden.de/rechtschreibung/Horizont`, 2016. Abgerufen am 04.06.2016.

[18] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Aaron E. Lefohn, Christof Rezk Salama, and Daniel Weiskopf. *Real-time Volume Graphics*. SIGGRAPH '04. ACM, New York, NY, USA, 2004.

[19] GEO ExPro. Seismic amplitudes. `http://www.geoexpro.com/articles/2015/11/a-simple-guide-to-seismic-amplitudes-and-detuning`, 2015. Abgerufen am 07.06.2016.

[20] ExxonMobil. Energy lives here. `http://corporate.exxonmobil.com/`. Abgerufen am 16.06.2016.

[21] R Stuart Ferguson. *Practical algorithms for 3D computer graphics*. CRC Press, 2013.

[22] Leibniz-Institut für Angewandte Geophysik LIAG. `https://www.liag-hannover.de/home.html`. Abgerufen am 16.06.2016.

[23] GeoTeric. Video: Geoteric demo. `http://www.geoteric.com/`, 2016. Abgerufen am 15.06.2016.

[24] Earthquake Glossary. Fault. `http://earthquake.usgs.gov/learn/glossary/?term=fault`, 2016. Abgerufen am 04.06.2016.

[25] Enrico Gobbetti, Piero Pili, Antonio Zorcolo, and Massimiliano Tuveri. Interactive Virtual Angioscopy. In *Visualization'98. Proceedings*, pages 435–438. IEEE, 1998.

[26] Hanqi Guo, Ningyu Mao, and Xiaoru Yuan. WYSIWYG (What You See is What You Get) Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106–2114, December 2011.

[27] Markus Hadwiger, Patric Ljung, Christof Rezk Salama, and Timo Ropinski. Advanced Illumination Techniques for GPU Volume Raycasting.

In *ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08, pages 1:1–1:166, New York, NY, USA, 2008. ACM.

[28] Landmark Halliburton. Decisionspace desktop. Video on SEG in Las Vegas. Datasheet: `https://www.landmark.solutions/Portals/0/LMSDocs/Datasheets/DecisionSpace-GIS-data-sheet-A4.pdf`, 2012. Abgerufen am 15.06.2016.

[29] Geophysical Insights. Paradise. `http://www.geoinsights.com/paradise-1/#paradise-overview`. Abgerufen am 15.06.2016.

[30] Peter Kohlmann, Stefan Bruckner, Armin Kanitsar, and M Eduard Gröller. Contextual Picking of Volumetric Structures. In *Visualization Symposium, 2009. PacificVis' 09. IEEE Pacific*, pages 185–192. IEEE, 2009.

[31] Philipp Ladwig. Immersive dreidimensionale Modellierung von komplexen Objekten in der seismischen Interpretation. Bachelor's thesis, 2014.

[32] Andrew Lauritzen. Deferred Rendering for Current and Future Rendering Pipelines. *SIGGRAPH Course: Beyond Programmable Shading*, pages 1–34, 2010.

[33] Offshore Magazine. Seismic vessel count remains steady. `http://www.offshore-mag.com/articles/print/volume-72/issue-3/geology-geophysics/seismic-vessel-count-remains-steady.html`, 2012. Abgerufen am 03.06.2016.

[34] Ramakrishnan Mukundan. *Advanced methods in computer graphics: with examples in OpenGL*. Springer Science & Business Media, 2012.

[35] OpendTect. Faults & fractures. `http://opendtect.org/index.php/products/commercial-products/faults-fractures`. Abgerufen am 15.06.2016.

[36] OpendTect. User documentation - interpret horizons. `http://www.opendtect.org/relman/4.6.0/unpacked/4.6.0/doc/User/base/chapter3.3_interpret-horizons.htm`. Abgerufen am 16.06.2016.

[37] OSG. Documentation. `http://trac.openscenegraph.org/documentation/OpenSceneGraphReferenceDocs/`. Abgerufen am 11.06.2016.

[38] Shigeru Owada, Frank Nielsen, and Takeo Igarashi. Volume Catcher. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 111–116, New York, NY, USA, 2005. ACM.

[39] I. Hill P. Kearey, M. Brooks. *An Introduction to Geophysical Exploration Third Edition.* John Wiley & Sons, 2002.

[40] Paradigm. Fault extraction. `http://www.pdgm.com/products/seisearth/fault-extraction/`. Abgerufen am 15.06.2016.

[41] Herbert Paschen, Dagmar Oertel, and Reinhard Grünwald. Möglichkeiten geothermischer Stromerzeugung in Deutschland. 2003.

[42] Hanchuan Peng, Zongcai Ruan, Fuhui Long, Julie H Simpson, and Eugene W Myers. V3D enables real-time 3D Visualization and quantitative Analysis of large-scale biological image data sets. *Nature biotechnology*, 28(4):348–353, 2010.

[43] Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. The state of the art in flow visualisation: feature extraction and tracking. *Computer Graphics Forum*, 22(4), 2003. 0167-7055.

[44] Jörg-Stefan Praßni, Timo Ropinski, and Klaus Hinrichs. Efficient boundary detection and transfer function generation in direct volume rendering. In *14th International Fall Workshop on Vision, Modeling, and Visualization (VMV 2009), November 16-18, 2009, Braunschweig, Germany*, pages 285–294, 2009.

[45] Bernhard Preim and Charl P. Botha. *Visual Computing for Medicine: Theory, Algorithms, and Applications.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 2013.

[46] Christof Rezk-Salama and Andreas Kolb. Opacity Peeling for Direct Volume Rendering. *Computer Graphics Forum*, 25(3):597–606, 2006.

[47] Schlumberger. Oilfield glossary. `http://www.glossary.oilfield.slb.com/en/Terms/h/horizon.aspx`, 2016. Abgerufen am 03.06.2016.

[48] Dave Shreiner and The Khronos OpenGL ARB Working Group. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1.* Addison-Wesley Professional, 7th edition, 2009.

[49] Schlumberger Petrel E&P Software. Video: Quantitative fault seal analysis. `https://www.software.slb.com/products/petrel?tab=Videos`, 2015. Abgerufen am 15.06.2016.

[50] Statoil. `http://www.statoil.com/`. Abgerufen am 16.06.2016.

[51] Klaus D Toennies and Claus Derz. Volume Rendering for Interactive 3D Segmentation. In *Medical Imaging 1997*, pages 602–609. International Society for Optics and Photonics, 1997.

[52] Unity3D. Documentation - collider. `https://docs.unity3d.com/ScriptReference/`. Abgerufen am 11.06.2016.

[53] Hartwig von Hartmann, Manfred Bogen, Stefan Rilling, and Rüdiger Thomas. SeisViz3D: Stereoskopische Darstellung seismischer Daten - Interpretation und Immersion, 2015. Poster presented at DGG 2015 - 75. Jahrestagung der Deutschen Geophysikalischen Gesellschaft, Hannover, 23. - 26. März 2015.

[54] Rui Wang and Xuelei Qian. *OpenSceneGraph 3.0: Beginner's guide*. Packt Publishing Ltd, 2010.

[55] Alexander Wiebel, Frans Vos, David Foerster, and Hans-Christian Hege. WYSIWYP: What You See Is What You Pick. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2236 – 2244, 2012.

[56] Alexander Wiebel, Frans Vos, and Hans-Christian Hege. Perception-oriented Picking of Structures in Direct Volumetric Renderings. Technical report, Technical Report 11–45, ZIB, Berlin, Germany, 2011.

[57] Rick Wilkinson. *Speaking Oil & Gas*. BHP Billiton Petroleum, 2006.

[58] Shin-Ting Wu, José Elías Yauri Vidalón, and Lionis de Souza Watanabe. Snapping a Cursor on Volume Data. In *SIBGRAPI*, 2011.

Anhang

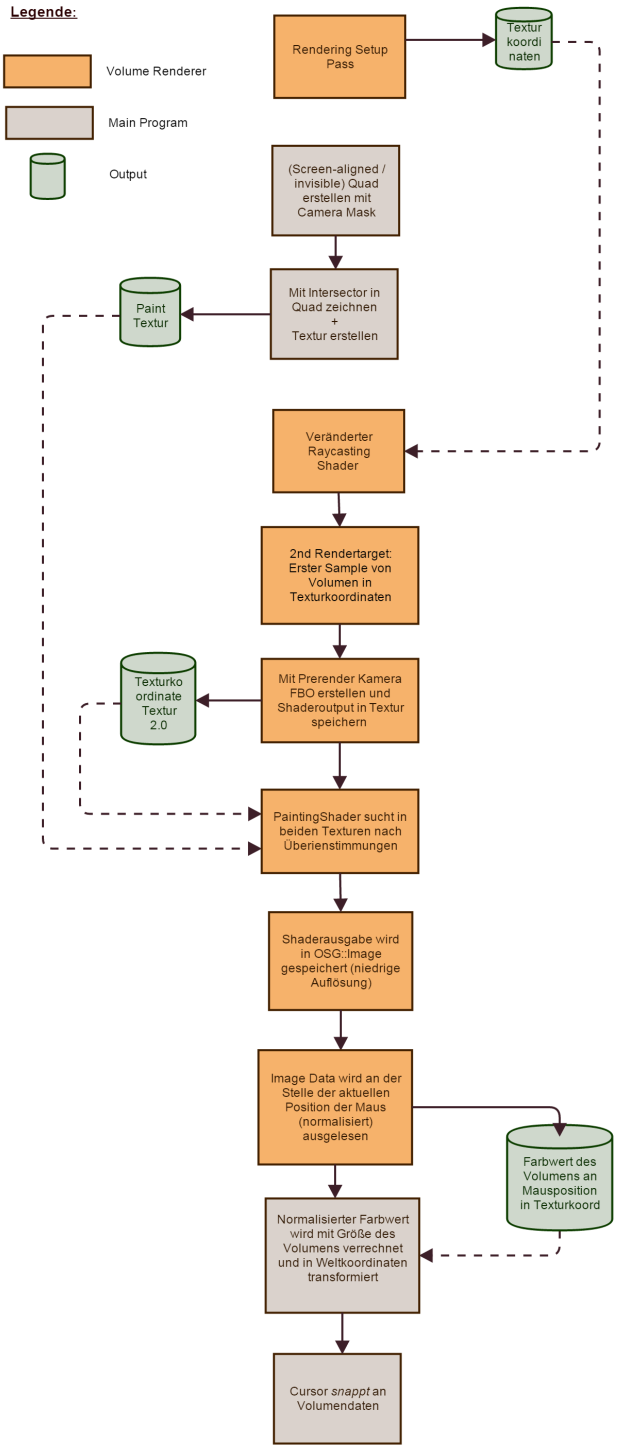

Abbildung 2: Aktivitätsdiagramm für *cursor snapping* Methode

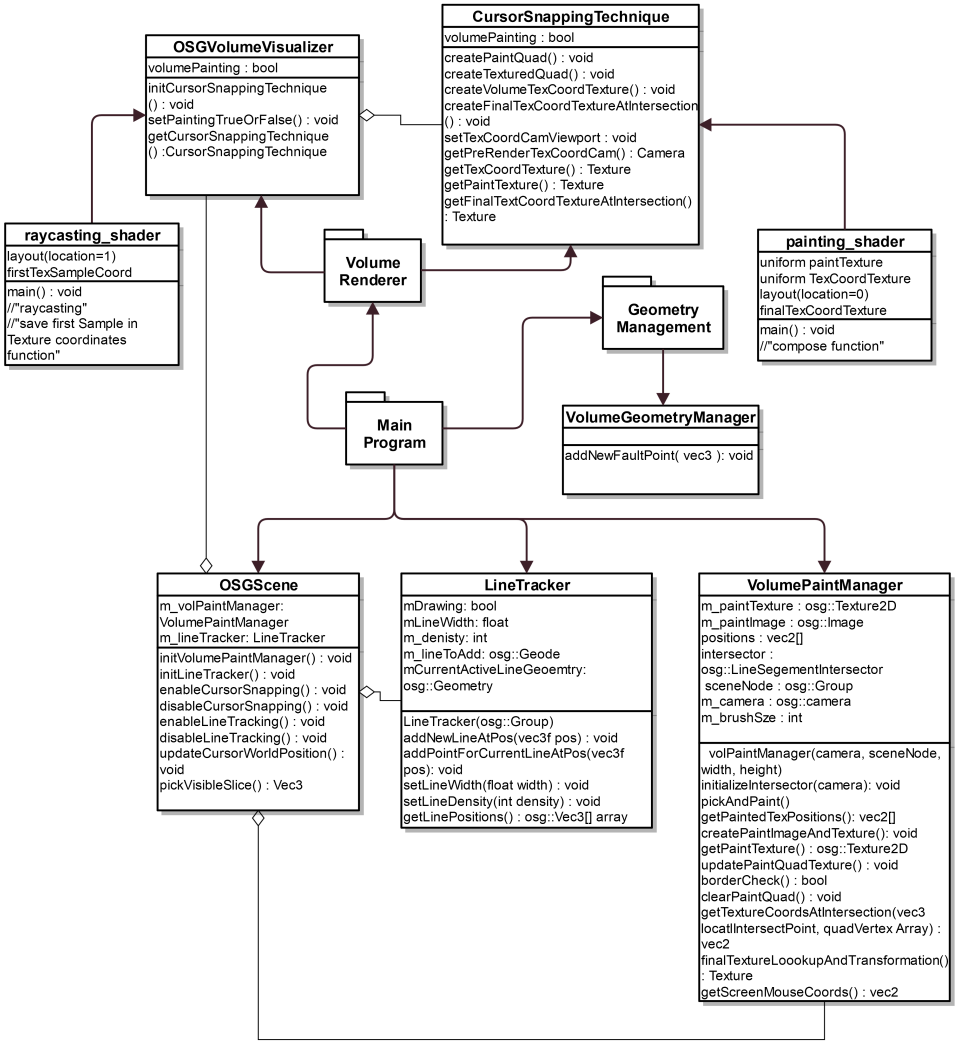

Abbildung 3: UML-Diagramm der entwickelten Methode

Köln, den 29. Juni 2016