

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik



Bachelorarbeit

Latenzverringerung bei der interaktiven Visualisierung von Simulationsdaten aus der Astronomie

Autor:

Kilian Gärtner

4. Mai 2015

Betreuer:

Prof. Dr.-Ing. Bernhard Preim
Dr.-Ing. Steffen Oeltze-Jafra
Institut für Simulation und Graphik

Dipl.-Phys. Bastian Sander, M.Sc.

Fraunhofer Institut für Fabrikbetrieb und -automatisierung IFF

Gärtner, Kilian:

Latenzverringerung bei der interaktiven Visualisierung von Simulationsdaten aus der Astronomie

Bachelorarbeit, Otto-von-Guericke-Universität Magdeburg, 2015.

Inhaltsangabe

Im Rahmen dieser Arbeit werden Möglichkeiten evaluiert, die Latenzzeit bei der interaktiven Visualisierung von wissenschaftlichen Daten zu verringern. Ausgangspunkt ist die Annahme, dass eine erhöhte Latenz zwischen Ein- und Ausgabe zu einem geringeren Informationsgewinn führt. Für die Untersuchung wurden Daten aus numerischen Simulationen im Bereich der Astrophysik mit verschiedenen Downsample Algorithmen reduziert. Mit Hilfe von bereits existierenden Tools wurden sowohl die reduzierten als auch die originalen Daten visualisiert und dabei visuell ihre Bildgüte unter zu Hilfenahme von Differenzbildern verglichen. Eine Zeitmessung für das Preprocessing und für die Interaktion mit den Visualisierungen war die Grundlage für die zentrale Frage, ob sich der Aufwand rentiert.

Danksagung

Ich bedanke mich bei Dipl.-Phys. Bastian Sander, M.Sc. für die intensive Betreuung, das Interesse an meiner Arbeit und stellvertretend bei die Universität Wien für die Bereitstellung der Daten ihrer Forschung, um meine Arbeit überhaupt durchführen zu können.

Ich bedanke mich auch bei Dr.-Ing. Steffen Oeltze-Jafra für die offene und herzliche Hilfe bei der Themenfindung und die Betreuung im Ganzen. Auch bedanke ich mich bei Prof. Dr.-Ing. Bernhard Preim für die Bereitschaft, sich meinem Praktikum und meiner Bachelorarbeit anzunehmen.

Ebenso bedanke ich mich bei den Mitarbeitern des Fraunhofer IFF B.Sc. Andreas Wiedemann und M.Sc Fabian Hecklau für die Unterstützung bei der Suche nach einem Thema für meine Bachelorarbeit.

Abschließend bedanke ich mich bei Gregor Grohmann dafür, dass er mir bei der Einhaltung der Deutschen Sprache half und meine Arbeit gegengelesen und korrigiert hat.

Teile dieser Arbeit wurden von dem Österreichischen Fonds zur Förderung der wissenschaftlichen Forschung, FWF, unterstützt, Projekt Nummer AP2109721. Weiterhin wurden Teile dieser Arbeit während eines wissenschaftlichen Aufenthaltes von Bastian Sander an der Jacobs Universität Bremen (Deutschland) mit Hilfe eines Zuschusses im Rahmen des KWA Programmes bei der Universität Wien(Nummer 000386) erarbeitet.

Die verwendete Software in dieser Arbeit wurde teilweise von DOE NNSA-ASC OASCR Flash Center an der Universität Chicago entwickelt. Die computergestützten, präsentierten Ergebnisse wurden teilweise mit dem Vienna Scientific Cluster 1 (VSC-1) und teilweise dem Astro-Cluster im Institut für Astrophysik an der Universität Wien berechnet.

Die teilweise in dieser Arbeit verwendete Software PARAMESH wurde am NASA Goddard Space Flight Center und Drexel University in den Projekten NASA's HP-CC und ESTO/CT und unter Zuhilfenahme von Zuschüssen NNG04GP79G des NASA/AISR Projektes entwickelt.

Inhaltsverzeichnis

Abbildungsverzeichnis	x
Quelltextverzeichnis	xi
Abkürzungsverzeichnis	xiii
1 Einführung	1
1.1 Motivation	2
1.2 Zielstellung der Arbeit	3
1.3 Gliederung der Arbeit	3
2 Verwandte Arbeiten	5
3 Grundlagen	7
3.1 Hochgeschwindigkeitswolken	7
3.2 FLASH	8
3.3 HDF5	9
3.4 Gitterstruktur	9
4 Experiment	11
4.1 Daten	11
4.2 Spezifikationen des Testsystems	13
4.3 Verwendete Tools	13
4.3.1 VisIt	14
4.3.2 NeonGenesisTool	14
4.4 Umgesetzte Verfahren	15
4.4.1 Downsampling mit statischen Mitteln	15
4.4.1.1 Median	15
4.4.1.2 Arithmetisches Mittel - Durchschnitt	16
4.4.2 Downsampling auf Grund physikalischer Eigenschaften	16
4.4.2.1 Mischtemperatur - Gewichtetes Mittel	16
4.5 Szenarien	17
4.5.1 1. Szenario - Vier-Fenster	17
4.5.2 2. Szenario - Detail-View	19
4.6 Ergebnisse	20
4.6.1 Zeiten für Downsampling	20
4.6.2 Zeiten für Szenarien	23
4.6.2.1 1. Szenario	23

4.6.2.2	2. Szenario	25
4.6.3	Visualisierungen	26
4.6.3.1	Kontur-Plot	27
4.6.3.2	Vektor-Plot	31
4.6.3.3	Histogramm	33
4.6.3.4	Scatter-Plot	35
4.6.3.5	Volumen-Plot	38
5	Zusammenfassung	41
5.1	Bildgüte	41
5.2	Downsampling Zeit	41
5.3	Datengröße	42
5.4	Zeit für Visualisierung	42
5.5	Resultat der Arbeit	42
5.6	VisIt im Allgemeinen	44
6	Zukünftige Arbeiten	47
6.1	Tools	47
6.2	Verfahren	48
6.3	Daten	48
A	Anhang	49
A.1	NeonGenesisTool Kommandozeilenreferenz	49
A.1.1	Anwendungsbeispiele	49
A.2	NeonGenesisTool Struktur	50
A.3	VisIt Python Scripts	53
A.4	yt-project Script	58
Literaturverzeichnis		59

Abbildungsverzeichnis

1.1	Beispiel interaktive Visualisierung - Links: Auswahlmenü der Daten und Art der Visualisierung. Rechts: Verschiedene Views und Visualisierungen der Daten, jeder in Echtzeit drehbar/verschiebbar und zoombar.	2
3.1	HVCs am Himmel der Milchstraße	8
3.2	Octree Visualisierung	9
4.1	Komplexität und Dateigröße der Ausgangsdaten	12
4.2	Entwicklung der HVC - Volumen-Plot der Dichte	12
4.3	Downsampling - Schematisch	15
4.4	1. Szenario - Vier Fenster	17
4.5	Unterschied Volumen-Plot (links) und Kontur-Plot	18
4.6	2. Szenario - Detail-View	19
4.7	Unterschied Splatting (links) und Raycasting	19
4.8	Zeit für Downsamling des Beispieldatensatzes	20
4.9	Zeit für Downsamling je Datei des Beispieldatensatzes	20
4.10	Vier Fenster - Detaillierte Rendering Zeiten	23
4.11	Vier Fenster - Kumulierte Rendering Zeiten	23
4.12	Vier Fenster - Kumulierte Zeiten für Interaktionen	24
4.13	Detail-View - Detaillierte Rendering Zeiten	25
4.14	Detail-View - Kumulierte Zeiten für Interaktion	25
4.15	Beispiel für Differenzbild (Kontur-Plot)	26
4.16	Kontur-Plot - Durchschnitt	27
4.17	Kontur-Plot - Median	28
4.18	Kontur-Plot - Mischtemperatur	29

4.20 Vektor-Plot Durchschnitt	31
4.21 Vektor-Plot Median	32
4.22 Histogramm - Durchschnitt	33
4.23 Histogramm - Median	34
4.24 Scatter-Plot - Durchschnitt	35
4.25 Scatter-Plot - Median	36
4.26 Scatter-Plot - Mischtemperatur	37
4.27 Volumen-Plot - Durchschnitt	38
4.28 Volumen-Plot - Unterschied im Detail	39
4.29 Volumen-Plot - Median	40
5.1 Vektor Plot - Fehler bei Darstellung	44
5.2 Problematische Bedienung bei der Erstellung einer Schnittebene	45
A.1 UML Diagramm von NeonGenesisTool	50

Quelltextverzeichnis

4.1	Extraktion der Werte für Mischtemperatur(Vereinfacht)	21
4.2	Extraktion der Werte für andere Variable(Vereinfacht)n	21
4.3	Berechnung der Mischtemperatur(Vereinfacht)	22
4.4	Berechnung des Durchschnittes(Vereinfacht)	22
A.1	Anwendungsbeispiel 1	49
A.2	Anwendungsbeispiel 2	50
A.3	Anwendungsbeispiel 3	50
A.4	Python Script zum Starten von VisIt mit Scripts	53
A.5	Visit Script - 4 Plots in VisIt	53
A.6	VisIt Script - Volume Renderung mit Raycasting	57
A.7	yt-project Script - Erstellung Slice Plots	58

Abkürzungsverzeichnis

AMR	Adaptive Mesh Refinement (dt. adaptive Gitterverfeinerung)
APU	Accelerated-Processing-Unit
CGS	Centimetre-gram-second system, auch Gauss-System
HVC	High-Velocity Cloud (dt. Hochgeschwindigkeitswolke)
JVM	Java-Virtual-Machine
kpc	Kiloparsec = $3.09 \cdot 10^{19}$ m
Refinementlevel	Refinementlevel(dt. Verfeinerungsgrad)
RGB	Red-Green-Blue (dt. Rot-Grün-Blau)
SciVis	Scientific Visualization

1. Einführung

Durch die steigende Leistungsfähigkeit von Computern und den erhöhten Einsatz massiver Parallelrechner in Form von Clustern und Supercomputer können Phänomene in den Naturwissenschaften einerseits detailliert und andererseits auch schneller simuliert werden. Dabei werden die Simulationsmodelle von Forschern modelliert und auf dedizierten, leistungsfähigen Systemen wie dem **VSC-1**¹ berechnet.

Dieser Anstieg von auszuwertenden Daten, sowohl durch ihre Auflösung als auch durch die Quantität an zu untersuchenden Problemen, führt zu sehr großen Datensätzen, die im wissenschaftlichen Umfeld sehr schnell mehrere Terabyte umfassen können. In der Astrophysik wurden Probleme untersucht, deren Ergebnisse bereits Maßstab von Petabyte² vorliegen [HF11]. Diese Datensätze sind zu umfangreich, um sie manuell zu untersuchen, weswegen die Datenauswertung ebenfalls Computer gestützt erfolgt. Dazu können Algorithmen aus dem Themengebiet des **Data-Mining** angewendet werden, die interessante Zusammenhänge eigenständig in den Daten suchen. Alternativ, und in dieser Arbeit weiter beleuchtet, kann die Exploration der Daten auch mit Hilfe von **interaktiven Visualisierungen** geschehen. In diesem Bereich kann der Nutzer aktiv am Visualisierungsprozess mitwirken und bestimmt, wenn möglich, in Echtzeit die Art der Visualisierung als auch Details in Form von Farben oder ausgewählten Datenbereichen.

¹Vienna Scientific Cluster 1 - <http://vsc.ac.at/systems/vsc-1/>

²Es würden ca. 20 000 Blue-ray Discs à 50 Gigabyte benötigt werden.

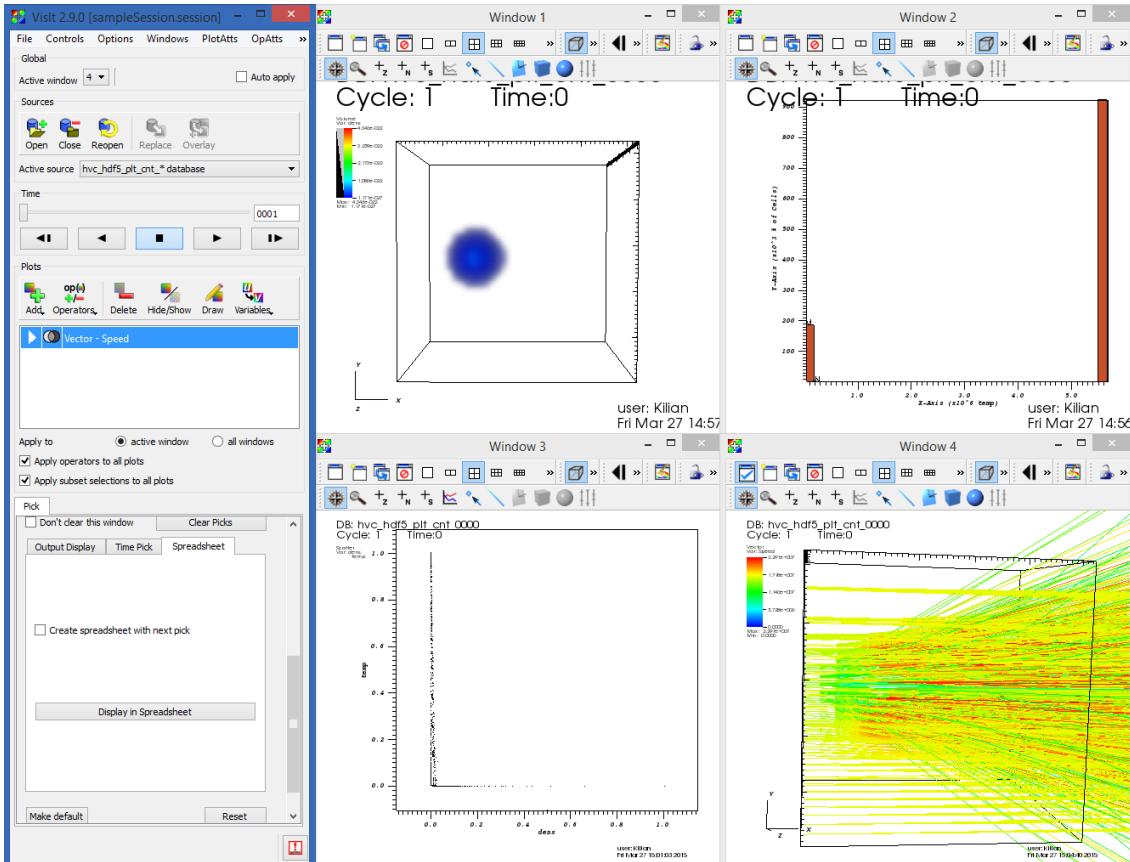


Abbildung 1.1: Beispiel interaktive Visualisierung - Links: Auswahlmenü der Daten und Art der Visualisierung. Rechts: Verschiedene Views und Visualisierungen der Daten, jeder in Echtzeit drehbar/verschiebbar und zoombar.

1.1 Motivation

Wichtig ist hierbei, dass jede Aktion vom Nutzer unmittelbar umgesetzt wird und so ein fließender Prozess zwischen Eingabe und Ausgabe entsteht. Eine erhöhte Latenz kann zu einem Informationsverlust in der Exploration durch mangelnde Motivation des Nutzers führen, wie Liu et al.[LH14] zeigte. So zeigte sich dort, dass bereits eine Latenz von 500 Millisekunden zu Unterschieden im Informationsgewinn durch den Nutzer führte. Deshalb ist es wichtig, nach Verfahren zu suchen, die trotz der angesprochenen Datenmenge die Latenz reduzieren.

Eine Möglichkeit dazu ist der Einsatz von schnellerer, jedoch gleichzeitig meist teurer Hardware als Grundlage. Es ist bei dieser Möglichkeit der finanzielle Mehraufwand und der heutzutage wichtige Aspekt der Energieeffizienz zu betrachten. Ein normaler PC kostet um die \$ 400[Gas] und verbraucht je nach Konfiguration 80 bis 250 Watt[Uni]. Im Gegensatz dazu kostet ein Super Computer wie der erwähnte **VSC-1** in der Anschaffung €2 Millionen und hat eine Anschlussleistung von 187 Kilowatt mit zusätzlich 56 Kilowatt Verbrauch für die Kühlung(Quelle: Private Kommunikation mit Dr. Markus Stöhr). Der **VSC-1** ist dabei noch ein Supercomputer der unteren Leistungskategorie. Führende Systeme wie der **Tianhe-2**³, verbrauchen mit

³Tianhe-2 (dt. Milchstraße-2) in China - <http://www.top500.org/system/177999>

17 Megawatt deutlich mehr und kosten in der Anschaffung ein Vielfaches. Deswegen ist es praktikabel, so viele Arbeiten wie möglich auf den günstigeren Desktoprechnern durchzuführen.

Ein weiteres Problem bei einfacherer Hardware ist neben der beschränkten Leistung die überhaupt zur Verfügung stehende Speichermenge. Heutige Festplatten auf Basis von Magnetscheiben fassen Speichermengen im unteren Terabyte-Bereich. Der viel schnellere Arbeitsspeicher jedoch ist um die Größenordnung 1000 kleiner und heutige Computer besitzen acht bis 16 Gigabyte. Diese Größenordnungen sind ein vielfaches kleiner als die der zu untersuchenden Daten. Eine mögliche Lösung dazu ist die Aufteilung der Daten in kleinere Gruppen, die einzeln untersucht werden können. So können Simulationsdaten zu bestimmten Zeiten abgebildet und jeder Zeitpunkt einzeln exploriert werden. Unpraktikabel wird es, wenn ganze Zeitverläufe analysiert werden oder man eine Entwicklung des Prozesses im Ganzen betrachten möchte. Als Alternative Lösung erscheint eine Reduzierung der Datenmenge durch Umrechnung der Auflösung.

1.2 Zielstellung der Arbeit

Ziel dieser Arbeit ist es, ein Verfahren zur Reduzierung der Datenmenge zu finden, mit denen auf normalen Desktop Computern wie dem Testsystem in [Abschnitt 4.2](#) die Datenmengen interaktiv exploriert werden können. Dabei ist ein Kompromiss zwischen Wiedergabegüte und Verarbeitungsgeschwindigkeit der Visualisierung zu finden. Die Darstellung der reduzierten Daten darf dabei zu keinen fehlerhaften Aussagen führen, sondern die Gesamtaussage weiterhin tragen. Einzelne Unterschiede wie unscharfe Darstellungen sind tolerierbar, wenn auch zu vermeiden. Ebenso muss als Randbedingung die Dauer für die Vorverarbeitung realistisch gehalten werden und muss in einem akzeptablem Zusammenhang zur Latenzreduzierung stehen. Des Weiteren ist es ein Ziel, dabei bestehende Systeme und Tools zu nutzen und nur zu adaptieren, sofern kein dringend benötigter Bedarf existiert.

1.3 Gliederung der Arbeit

Diese Arbeit ist in sechs Kapitel unterteilt, wovon [Kapitel 1](#) als Einleitung in die Thematik bereits besprochen wurde. Das folgende [Kapitel 2](#) zeigt verwandte Arbeiten auf, die sich mit thematisch ähnlichen Problem befassen und die z.B. in dieser Arbeit verwendet worden sind. [Kapitel 3](#) erläutert benötigte Begrifflichkeiten, die im Laufe von [Kapitel 4](#), dem Experiment, verwendet werden. Im vorletzten [Kapitel 5](#) erfolgt die zusammenfassende Diskussion der Ergebnisse des Experiments und als Ausblick für weiterführende Arbeiten und nicht behandelte, dennoch interessante, Probleme dient [Kapitel 6](#).

2. Verwandte Arbeiten

Die Thematik um Visualisierungen vor allem im Astrophysik Bereich ist kein neues Thema, sondern fing schon in den 1980er Jahren an. So konzipierte McCormick [McC87], wie **Scientific Visualization (SciVis)** auszusehen hat und welche Schwierigkeiten dies mit sich bringt. Ebenso Frenkel[Fre88] 1988, der sich auf McCormicks Arbeit beruft und auch 1989 DeFanti et al.[DBM89]. Diese drei lieferten die Grundkonzepte für **SciVis** und kristallisierten bereits die Anwendungsmöglichkeit für die Astrophysik heraus.

2005 demonstrieren Borkin et al.[BRGH05] mit dem Tool **3D Slicer**¹ die Anwendung auch für Daten aus der Astronomie, obwohl das Tool eigentlich für medizinische Daten entwickelt worden war. Sie adaptierten das Tool, um die ihnen vorliegenden Daten einzulesen, da es das einzige Tool zu dem Zeitpunkt war für interaktive 3D-Visualisierungen mit der Möglichkeit für Slice-Plots. Heutige Tools, wie das in dieser Arbeit verwendete **VisIt**, existierten damals nicht oder waren zu eingeschränkt.

2011 reduzierten Wang et al.[WWLM11] die Datenmenge, in dem sie das zugrunde liegende Gitter deformierten. Dazu identifizierten und markierten sie wichtige Regionen im Gitter und vergrößerten dieses, während unwichtige Regionen verkleinert worden sind. Diese Verfahren ähnelt einem **Adaptive Mesh Refinement (dt. adaptive Gitterverfeinerung) (AMR)**, wird jedoch im Gegensatz nach der Simulation für die Auswertung angewandt.

Andere Wege, um die Bildqualität zu erhalten und dennoch die Datensätze zu reduzieren, wurden ebenfalls unternommen. So beschäftigten sich Younesy et al.[YMC06] sich mit verbesserten Visualisierungen bei dem Volumen-Rendering von medizinischen Daten. Diese verwendeten zusätzliche statistische Informationen in Form von Histogrammen, um Artefaktbildungen wie in **Abschnitt 4.6.3.5** zu sehen zu minimieren. Aufbauend auf dieser Arbeit versuchten Sicat et al.[SKMH14] verfeinerten sie die Methode durch Einführung von Sparse probability density functions, pdfs, um die Speichergröße zu reduzieren. Ähnliche Ansätze untersuchten auch Thompson et. all[TLB⁺11], die ebenfalls als zusätzliche Datenstruktur ein Histogramm je Datenpunkt einführen. All diese Verfahren führen zu einer besseren Darstellung bei

¹Website: <http://am.iic.harvard.edu/index.cgi/UsingSlicer>

der Reduzierung, haben jedoch einen höheren Aufwand und führen eine zusätzliche Datenstruktur ein. Im Rahmen dieser Arbeit wurden jedoch nur verfügbare Tools und deren Möglichkeiten genutzt, ohne diese zu modifizieren und die erwähnten Verbesserungen nutzen zu können.

3. Grundlagen

3.1 Hochgeschwindigkeitswolken

Die sogenannten [High-Velocity Clouds](#) (dt. Hochgeschwindigkeitswolken) (HVCs) wurden das erste Mal Anfang der 1960er Jahre entdeckt [MOR63, 1963] und wurden seitdem auch bei anderen Forschungen entdeckt[vW04]. Sie sind charakterisiert durch eine Radialgeschwindigkeit, die im Widerspruch zu dem Modell einer einfachen, differenziellen Rotation der Galaxie steht. Ihr erhöhter Drehimpuls lässt darauf schließen, dass sie ihren Ursprung außerhalb der Milchstraße, unserer Heimatgalaxie, besitzen. Die Entfernungn betragen zwischen 10 und 150 [Kiloparsec](#) von der Sonne [vW04]. HVCs sind ein relativ häufiges Phänomen und bedecken ca. 20-40% des Himmels um die Milchstraße (siehe [Abbildung 3.1](#)), wurden aber auch in anderen Galaxien nachgewiesen [Wak01]. Diese bestehen zu einem großen Anteil aus Wasserstoff, einen kleinerem Teil Helium und Metallen¹. Trotz ihrer intensiven Erforschung innerhalb der letzten 50 Jahre geben sie noch immer Rätsel auf. So gibt es noch kein einheitliches Modell für ihren Ursprung, welches alle Beobachtungsdetails hinreichend beschreibt. Auch ihre Entfernungen sind mangels Vordergrund- und Hintergrundquellen entweder gar nicht oder nur sehr ungenau bekannt. Ein weiteres Problem ist das der Sternentstehung. Obwohl [HVCs](#) sämtliche physikalische Randbedingungen für Sternentstehung bieten (ausreichend Masse, Existenz gravitativer Instabilitäten, viele chemische Elemente, ausreichend lange Lebensdauer), wurden noch nie assoziierte Sterne nachgewiesen.[TRH98, TWM⁺02]

¹Bezeichnet in der Astrophysik alle Elemente die schwerer als Helium sind und weicht von der chemischen Definition für Metalle ab

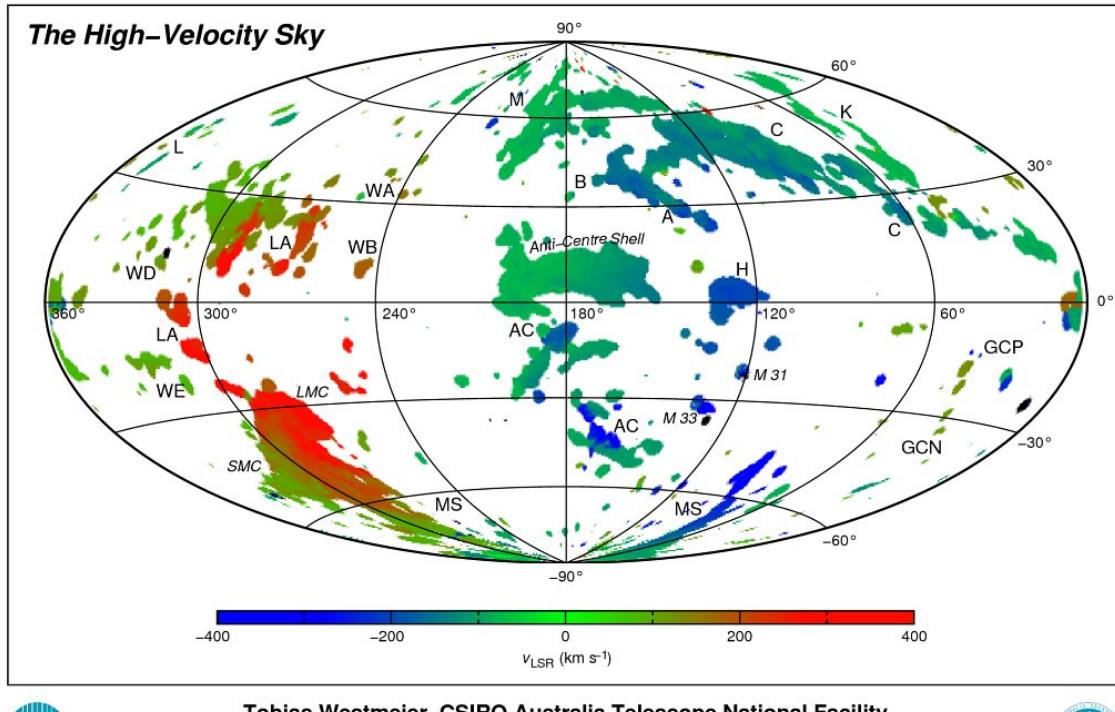


Abbildung 3.1: HVCs am Himmel der Milchstraße

Quelle: http://www.atnf.csiro.au/people/Tobias.Westmeier/research_hvcsky.php (2015 abgerufen)

Neben den Beobachtungen mit sowohl erdgebundenen als auch Satellitenteleskopen und analytischen Erklärungsansätzen werden seit Ende der 1980er Jahre auch numerische Rechnungen benutzt, um die Entwicklung von HVCs zu untersuchen [TBRF87, FTB⁺88]. In ihren Simulationen untersuchten Sander & Hensler [SG15] die Entwicklung von HVCs in drei Dimensionen mithilfe des adaptiven Gittercodes FLASH (siehe Abschnitt 3.2). Dabei wurden sowohl hydrodynamische, thermische, gravitative und chemische Prozesse berücksichtigt.

3.2 FLASH

FLASH² [DAG⁰⁹] ist ein öffentlich verfügbarer Code in Fortran95 für numerische Simulationen, dessen aktive Entwicklung vom Flash Center for Computational Science in Chicago koordiniert wird, an der sich aber Forscher auf der ganzen Welt maßgeblich beteiligen. Für die vorliegenden Simulationen wurde Version 3.2 benutzt. Die aktuelleren Versionen (Aktuell 4.2.2) haben deutlich an Funktionalität gewonnen. Zum Beispiel physikalische Prozesse bei hohen Energiedichten, diese wurden für die hier simulierten Sachverhalte nicht benötigt. FLASH wurde für den Einsatz auf großen Rechenclustern entworfen, was entweder durch OpenMP- oder MPI-Bibliotheken [For94] gewährleistet ist, so dass die Simulation für sehr viele Prozessoren linear skaliert.

Die Ausgabe erfolgt in FLASH in sog. Plotfiles oder Checkpointfiles. Während erstere

²Webseite: <http://flash.uchicago.edu/site/flashcode/>

eine vom Benutzer festgelegte Anzahl an Variablen enthalten, spiegeln letztere den kompletten Systemzustand wider und können als Ausgangspunkt für einen Neustart der Simulation benutzt werden. Sie belegen daher deutlich mehr Speicherplatz (in den vorliegenden Simulationen bis zu zehnmal mehr). Für die Analyse der Ergebnisse reichen die Plotfiles in den meisten Fällen jedoch aus und wurde für diese Arbeit auch nur verwendet.

3.3 HDF5

Die in dieser Arbeit verwendeten plotfiles von FLASH liegen im HDF5³[TheNN] Format vor. Dieses wurde speziell für sehr große Datenmengen wie sie in der wissenschaftlichen Anwendung existieren, entwickelt. Es ist ein Binärformat, welches Metadaten beinhaltet, um das Format der Daten zu speichern. Im Gegensatz zu relationaler Datenspeicherung wie in heutigen Datenbanksystemen sind die Daten in einer Hierarchie angeordnet. Als Ergebnis dieser Entscheidung werden statt Tabellen n-dimensionale Matrizen verwendet, in denen die Daten abgelegt werden. Beispielsweise liegen die Temperaturwerte für einen Zeitpunkt in einer 4D Matrix vor, wovon die erste Dimension die Blocknummer ist und die darauffolgenden Dimensionen jeweils für die X-, Y- und Z-Ebene stehen. Die Daten in einer Matrix müssen den gleichen Datentyp besitzen. Dieser ist jedoch frei wählbar und kann auch ein zusammengesetzter Datentyp sein. Zeichenketten können eine fest definierte oder eine variable, durch ein Terminierungszeichen gekennzeichnet, Länge haben. Die für diese Arbeit wichtigen Werte sind als 32 Bit Gleitkomma Zahl im IEEE 754-Standard[iee08] gespeichert.

3.4 Gitterstruktur

Um Speicherplatz und Simulationszeit einzusparen, wurde bei der Simulation der Daten ein AMR verwendet. Im Gegensatz zu einem uniformen Gitter können Blöcke in diesem unterschiedliche Größen besitzen. So werden für relativ leere Bereiche Blöcke in einer niedrigen Refinementlevel(dt. Verfeinerungsgrad) benutzt, während in Bereichen mit vielen Prozessen ein höheres Refinementlevel verwendet wird. Die Relation zwischen den verschiedenen Refinementlevels wird durch einen Octree⁴ beschrieben.

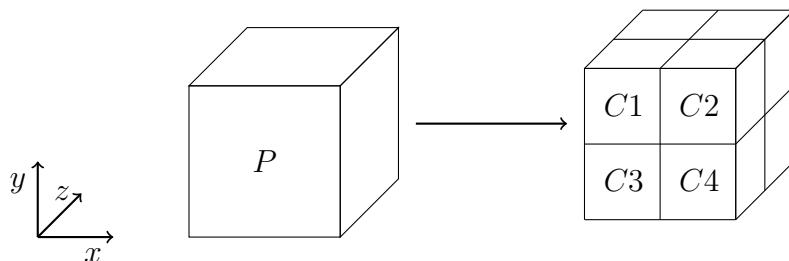


Abbildung 3.2: Octree Visualisierung

³Website: <http://www.hdfgroup.org/>

⁴Ein Baum, dessen Knoten jeweils acht oder kein Kindknoten besitzt

Jedes Kind C repräsentiert ein Achtel vom Raume des Eltern Knotens P . Dabei kann jedes Kind wiederum in weitere Blöcke unterteilt werden. Die Höhe jedes Knotens im Octree entspricht seinem Refinementlevel. Die Wurzel des Octree ist dabei der komplette Simulationsraum. Jeder Block ist nochmal in mehrere Zellen aufgeteilt, welche die Werte für die Skalare beinhalten. Dabei besitzt jeder Block, egal in welcher Position der Hierarchie er sich befindet, die selbe Anzahl an Zellen.

In der verwendeten Simulation und ihren Daten beträgt das maximale Refinementlevel 6. Der Simulationsraum selber hat eine Kantenlänge von 340pc und der feinst aufgelöste Block $340\text{pc} \div 2^6 = 5.3125\text{pc}$. Jeder Block besitzt dabei in der x-, y- und z-Dimension jeweils 8 Zellen und somit insgesamt 512 Zellen. Die räumliche Auflösung für eine Zelle eines Blockes mit dem Refinementlevel 6 beträgt dann $5.3125\text{pc} \div 8 = 0.6640625^5$.

⁵Würde man unser Sonnensystem abbilden wollen, würde dieses selbst in so einer Zelle nicht dargestellt werden können

4. Experiment

In diesem Kapitel wird das Experiment beschrieben, das zur Untersuchung der Aufgabenstellung aufgestellt und durchgeführt wurde.

4.1 Daten

Im folgendem eine Auflistung der berechneten Variablen, beistehend immer der Name der Matrix und deren Einheit nach dem Centimetre-gram-second system, auch Gauss-System (CGS):

- **dens** Dichte in $\frac{\text{g}}{\text{cm}^3}$
- **eint** Innere Energie in $\frac{\text{erg}}{\text{g}}$ (Energien in FLASH (Abschnitt 3.2) immer pro Masse)
- **ener** Totale Energie in $\frac{\text{erg}}{\text{g}}$
- **gpot** Gravitationspotential in $\frac{\text{cm}^2}{\text{s}^2\text{g}}$
- **metl** Verhältnis der Metallizität zur solaren Metallizität
- **ms_h** Prozentualer Anteil des Zellenmaterial von dem Wolkenmaterial. Indikator für Vermischung der Materie. $\text{ms_h} = 1 - \text{ms_i}$
- **ms_i** Prozentualer Anteil des Zellenmaterial von dem umliegendem Raum. Indikator für Vermischung der Materie. $\text{ms_i} = 1 - \text{ms_h}$
- **pres** Druck in $\frac{\text{erg}}{\text{cm}^3}$
- **temp** Temperatur in K

- **velx** Geschwindigkeit in X-Richtung in $\frac{\text{cm}}{\text{s}}$
- **vely** Geschwindigkeit in Y-Richtung in $\frac{\text{cm}}{\text{s}}$
- **velz** Geschwindigkeit in Z-Richtung in $\frac{\text{cm}}{\text{s}}$

Der Simulationszeitraum betrug 10^8 Jahre¹ und jedes Megajahr wurde ein *Checkpointfile* angelegt, in dem der Zustand der komplett Simulation zu diesem Zeitpunkt festgehalten wurde. Aus diesen Dateien wurden die *Plotfiles* erstellt, mit denen in dieser Arbeit gearbeitet wurde.

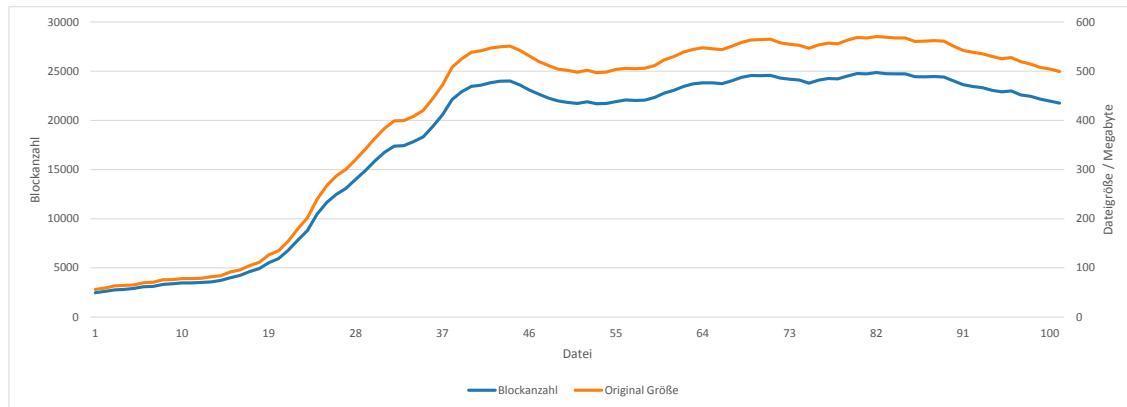
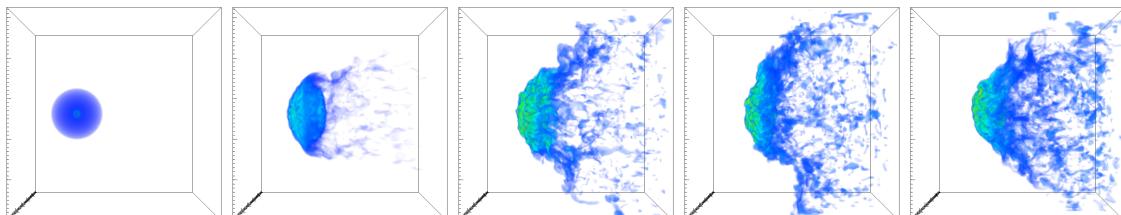


Abbildung 4.1: Komplexität und Dateigröße der Ausgangsdaten

In dem verwendeten Simulationsmodell fliegt die Wolke mit einer bestimmten Geschwindigkeit durch den interstellaren Raum und es lösen sich Teile der Wolke von dem Hauptkörper ab. Wenn dies geschieht, werden für diese Bereiche höhere **Refinementlevel** zur Erfassung feiner Strukturen verwendet. Deswegen erhöht sich im Laufe der Simulation die Blockanzahl, siehe Abbildung 4.1. Proportional dazu erhöht sich auch die Dateigröße, da für jeden zusätzlichen Block weitere 4 Byte(32-Bit Float) je Zelle benötigt werden.



(a) Initial (b) $25 \cdot 10^6$ Jahre (c) $50 \cdot 10^6$ Jahre (d) $75 \cdot 10^6$ Jahre (e) $100 \cdot 10^6$ Jahre

Abbildung 4.2: Entwicklung der HVC - Volumen-Plot der Dichte

¹Entspricht ca. 2% des Alters der Erde mit $4,54 \cdot 10^9$ Jahren [Dal01, MADH80, Dal94]

4.2 Spezifikationen des Testsystems

Die Berechnungen werden zur Zeitmessung auf dem selben System in einem möglichst konstanten Zustand durchgeführt. Die Spezifikationen dafür sind die folgenden:

Betriebssystem

Windows 8.1 Professional 64-Bit

Prozessor

Intel Core i5-4750, vier Kerne mit bis zu 3.20 GHz

Arbeitsspeicher

12 Gigabyte DDR-3 1070 MHz auf 4 Riegel, Dual Channel

Mainboard

ASRock H87 Pro4

Grafikkarte

NVIDIA GeForce GT 560 TI, 1 Gigabyte GDDR5, Treiber 347.52

Festplatte - Lesen der Daten

Samsung HD501LJ 500 Gigabyte HDD, SATA3

Festplatte - Schreiben der Daten

Samsung HD103UI 1000 Gigabyte HDD, SATA3

Festplatte - System und Programme

Samsung 840 EVO 120 Gigabyte SSD, SATA3

Auflösung

Dual Monitor 3360x1050 - Single 1680x1050

4.3 Verwendete Tools

Es existieren bereits frei verfügbare Tools zur Darstellung von wissenschaftlichen Daten. Deswegen bestand für diese Arbeit keine Notwendigkeit, ein eigenes Tool zu entwickeln. Bedingung für das zu nutzende Tool war wie bereits angesprochen die freie Verfügbarkeit und Nutzung des Tools und die Möglichkeit Ergebnisse von FLASH zu verarbeiten. Hassan & Fluke[HF11, p. 14f, 2011] evaluierten bereits mehrere Tools und gaben einen Überblick über deren Funktionalitäten. Diese Liste diente zur Orientierung von möglichen Tools.

Bisher verwendeten Sander & Hensel die Softwarelösung IDL². Diese ist jedoch proprietär und wurde deswegen hier nicht weiter betrachtet. Als weitere, viel versprechende Lösung wurde Paraview³[Aya15] in der Version 4.3.1 untersucht. Leider war es nicht möglich, dort die Ergebnisse der Simulationen zu laden. Es existiert ein FLASH Interpreter, jedoch konnte nicht herausgefunden werden, warum die Daten nicht geladen werden können.

²Website: <http://www.exelisvis.com/ProductsServices/IDL.aspx>

³Website: <http://www.paraview.org/>

Das Projekt `yt-project`⁴ [TSO⁺¹¹] auf Basis von Python ist ein Toolkit für die Visualisierung und scheint ein sehr aktiv geführtes, modernes Projekt zur Visualisierung von wissenschaftlichen Daten zu sein. Es unterstützt auch ohne Probleme `FLASH`, hat durch ein mitgeliefertes Installationsskript für die verschiedenen Betriebssysteme und kann mit wenigen Befehle komplexe Plots anfertigen (siehe Beispiel für Schnittplot [Quelltext A.7](#)). Dieses Projekt hat jedoch keine Möglichkeit zur interaktiven Visualisierung und wird deswegen hier nur als potentielle Alternative für umfangreiche Visualisierungen aufgeführt.

4.3.1 VisIt

Die Daten wurden mit dem Tool `VisIt`[CBW⁺¹²] in der Version 2.9⁵ visualisiert. Dieses Tool unterstützt die benötigten Eigenschaften für eine interaktive Visualisierung und kann u.a. die `HDF5` Dateien von `FLASH` einlesen. `VisIt` ist ein sehr komplexes, wenn leider auch mangelhaft dokumentiertes Tool (siehe [Abschnitt 5.6](#)), weswegen in dieser Arbeit nicht bis ins Detail alle Funktionen vorgestellt oder genutzt worden konnten. Es ist u.a. darauf ausgelegt, dass die Berechnungen auf einem externen, viel leistungsstärkerem Server ausgeführt werden als auf dem Client, der nur die Nutzereingabe aufnimmt, an den Server leitet und dann die erzeugten Bilder anzeigt. Diese Vorgehensweise widerspricht jedoch den gesetzten Zielen dieser Arbeit und wird im weiterem Verlauf nicht weiter verwendet. Dagegen wurde die Nutzung von Macros via Python Scripts zur Erstellung von Visualisierungen im Batch Betrieb verwendet. Diese Fähigkeit wurde im Laufe des Experiments genutzt, um die Zeiten für die Visualisierungen transparent und automatisiert zu messen.

Für die Nutzung und dem weiteren Umfang sind die Website zu Nutzen oder die Auflistung der dezentralen Dokumentation in [Abschnitt 5.6](#).

4.3.2 NeonGenesisTool

Um die Daten zur reduzieren, wurde im Rahmen der Bachelorarbeit ein Tool mit dem Namen `NeonGenesisTool`⁶ in Java 8 entwickelt. Dieses liest die `HDF` Dateien ein und reduziert sie nach einem der in [Abschnitt 4.4](#) aufgelisteten Verfahren. Dazu bleibt die Datei weiterhin `FLASH` kompatibel und wird von den Interpretern als vollwertige Datei anerkannt. Um die Dateien zu lesen und zu schreiben, wurde das `HDF5-API`⁷ verwendet. Die Bedienung des Tools erfolgt über die Kommandozeile, deren Referenz unter [Abschnitt A.1](#) nachzulesen ist. Eine grafische Oberfläche wurde wegen Zeitgründen nicht umgesetzt, kann jedoch durchgeführt werden. Die allgemeine Funktionsweise wird im Anhang unter [Abschnitt A.2](#) umrissen. Wegen des Umfangs des Quellcodes ist dieser, sofern er der Arbeit nicht beiliegt, auf der Website zu sichten. Er unterliegt der MIT Lizenz⁸.

⁴Website: yt-project.org

⁵Website: <https://wci.llnl.gov/simulation/computer-codes/visit>

⁶GitHub Repository: <https://github.com/Meldanor/NeonGenesisTool>

⁷HDF5 API für Java: <http://www.hdfgroup.org/products/java/>

⁸MIT Lizenz(englisch): <http://opensource.org/licenses/MIT>

4.4 Umgesetzte Verfahren

Um die Daten zu reduzieren, wurden mit dem Tool `NeonGenesisTool` mehrere Arten des Downsampling implementiert. Dazu werden die Daten mit einem Informationsverlust reduziert, der im Gegensatz zur Kompression nicht vermieden werden kann. Da die Daten auf einem strukturiertem Gitter liegen, wurden die Zellen der Blöcke von $8^3 = 512$ zu $4^3 = 64$ reduziert.

Dazu wurden immer $2 \times 2 \times 2$ Zellen zu einer Zelle (siehe Abbildung 4.3) zusammengefasst. Die verschiedenen Arten der Zusammenfassung werden im folgenden einzeln erläutert. Die Auswertung der benötigten Zeit und der Unterschiede in den Visualisierungen erfolgen in Abschnitt 4.6.

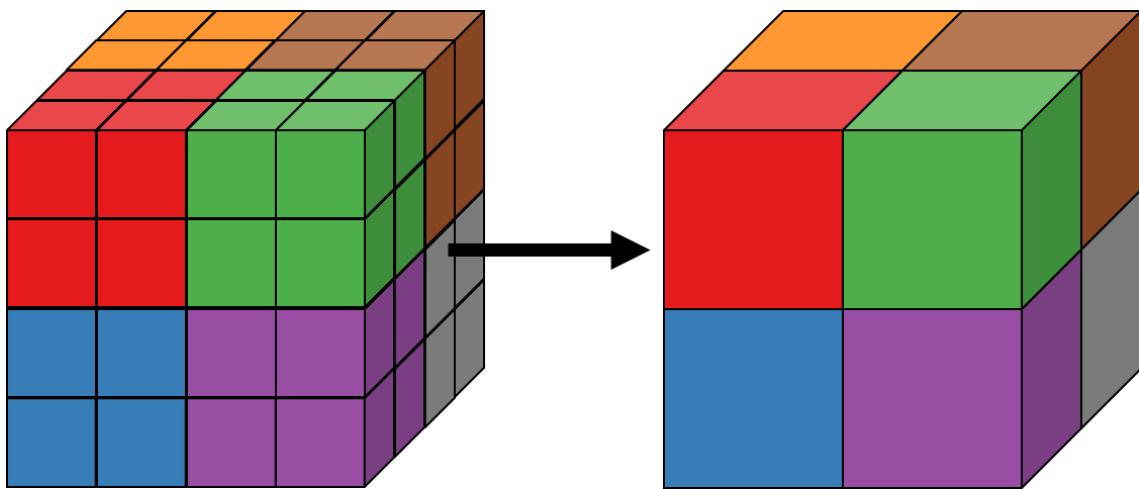


Abbildung 4.3: Downsampling - Schematisch

4.4.1 Downsampling mit statischen Mitteln

Ohne genaueres Wissen über die Daten zu besitzen, können diese mit statischen Verfahren reduziert werden.

4.4.1.1 Median

In einer sortierten Menge ist der Median das mittlere Element. Ist die Anzahl der Elemente in der Menge gerade, so existiert kein mittleres Element und der Median ergibt sich aus den beiden mittleren Elementen und dessen Durchschnitt. Der Aufwand dieser Operation ist eine Sortierung der Daten, die mindestens $\mathcal{O}(n \log n)$ (Quicksort [SW11, p. 342]) beträgt, je nach gewählten Sortieralgorithmen.

Sedgewick & Wayne zeigten jedoch auch die Möglichkeit zur Bestimmung des Medians in nahezu linearem Zeitaufwand [SW11, p. 346]. Dieser wurde jedoch wegen der geringen Größe der Elemente hier nicht implementiert, kann aber für größere n interessant sein.

4.4.1.2 Arithmetisches Mittel - Durchschnitt

Das arithmetische Mittel ist wie folgt definiert:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Der Aufwand dieser Operation ist $\mathcal{O}(n)$, da der Datensatz nur einmal iteriert und aufsummiert wird. Um ein Überlauf der Variable für sehr große oder kleine Werte ($+/-1,4\text{E}-45$, $+/-3,4\text{E}+38$) im Programm möglichst zu vermeiden, wird folgende Formel verwendet:

$$\bar{x} = \sum_{i=1}^n \frac{x_i}{n}$$

Dennoch ist ein Überlauf möglich, jedoch nicht so häufig wie bei der ersten Formel. Die zweite Formel führt jedoch zu einer erhöhten Ungenauigkeit durch häufigeres Dividieren, was bei einer 32-Bit Gleitkommazahl bei vielen Divisionen schnell passieren kann. Bei den Berechnungen werden jedoch nur maximal 8 Zahlen aufaddiert, wo eine Ungenauigkeit durch die Divisionen kaum ins Gewicht fällt, bei der Datenmenge jedoch ein Überlauf von positiven zu negativen Zahlen gravierende Auswirkungen haben kann.

4.4.2 Downsampling auf Grund physikalischer Eigenschaften

Statt die Daten per Durchschnitt oder Median zusammenzufassen, werden auch Alternativen untersucht. So kann mit Wissen über die Herkunft und die Zusammenhänge der Daten eine Reduzierung erfolgen, deren Unterschied zum Original eventuell geringer als bei statistischen Verfahren ist. In diesem Abschnitt werden physikalische Gegebenheiten aufgelistet, die für eine Reduzierung verwendet werden können.

Variablen, für die keine explizite Formel angewandt wird, werden weiterhin mit statistischen Verfahren reduziert. Der Modus für die gemischte Anwendung wird in der laufenden Arbeit als *Physik+Statistisches Verfahren* bezeichnet.

4.4.2.1 Mischtemperatur - Gewichtetes Mittel

Diese beschreibt die Temperatur, wenn zwei Körper miteinander vermischt werden. Dabei ist die Masse der Körper die Gewichtung, da ihre Energie in Form der Temperatur je nach Masse verschieden stark eingebracht werden. Betrachtet man in den vorliegenden Daten die Zellen als einzelne Körper, kann man die Temperatur beim Downsamplen über die Mischtemperatur T_M berechnen. Dazu wird neben der Temperatur T_i auch noch die Dichte ϱ_i des Körpers K_i benötigt.

$$T_M = \frac{\sum_{i=1}^n \varrho_i T_i}{\sum_{i=1}^n \varrho_i}$$

4.5 Szenarien

Um die Zeiten für die interaktive Visualisierung zu messen, wurden für **VisIt** zwei Szenarien erstellt. Diese werden mit Hilfe von Python Scripts abgebildet, um diese automatisiert visualisieren und dabei die Zeiten für das Rendering und einfache Interaktionen messen zu können. In jedem Szenario wurden für die Zeitmessungen sowohl die Originaldaten als auch ein reduzierter Datensatz verwendet. Obwohl verschiedene Verfahren für die Reduzierung angewandt worden sind, so unterscheiden sich diese nicht in der erzeugten Datengröße und Komplexität und somit auch nicht in der benötigten Zeit. Für die Visualisierung hingegen wurde jedes Szenario visualisiert und deren Bildgüte miteinander verglichen.

4.5.1 1. Szenario - Vier-Fenster

In diesem werden in vier verschiedenen Fenster jeweils ein Plot dargestellt. Im ersten wird ein Kontur-Plot⁹ der Temperatur als 3D-Visualisierung der **HVC** verwendet, im zweiten ein Histogramm der Dichte, im dritten ein Vektor-Plot der Geschwindigkeit und im vierten ein Scatter-Plot(dt. Streudiagramm), der das Verhältnis der Dichte zur Temperatur widerspiegelt. Bei dem Scatter-Plot werden die Werte logarithmisch zur besseren Übersicht skaliert. Durch die verschiedenen Plots soll das Explorieren der Daten simuliert werden, in dem mit verschiedenen Ansichten gearbeitet wird. Im Folgenden eine Darstellung des Aufbaus des ersten Szenarios:

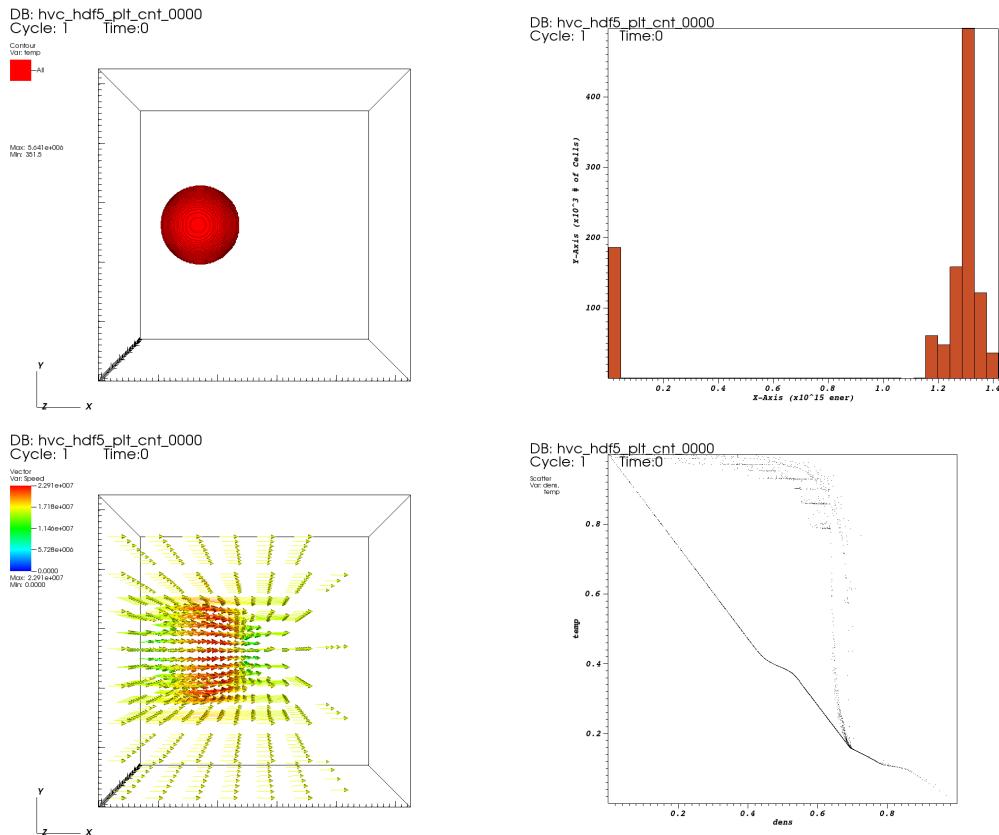


Abbildung 4.4: 1. Szenario - Vier Fenster

⁹Durch den 3D Bezug ein Isoflächen-Plot(engl. *isosurface*), in **VisIt** jedoch als *Contour-Plot* bezeichnet.

Dieses Szenario wurde mit Quelltext A.5 abgebildet. Die 3D Visualisierung der Wolke ist nicht so hoch auflösend wie im zweiten Szenario, da es nur zur räumlichen Orientierung dienen soll. Es werden auch nur die äußereren Strukturen abgebildet, jedoch das Innere der HVC wird dabei ausgeblendet. Eine Alternative wäre die Wahl eines Volumen-Plot mit dem Splatting-Verfahren¹⁰[Wes91, CXZ11], dieses bietet jedoch keine so detaillierte Darstellung der Wolke. Für höhere Auflösung siehe Szenario 2.

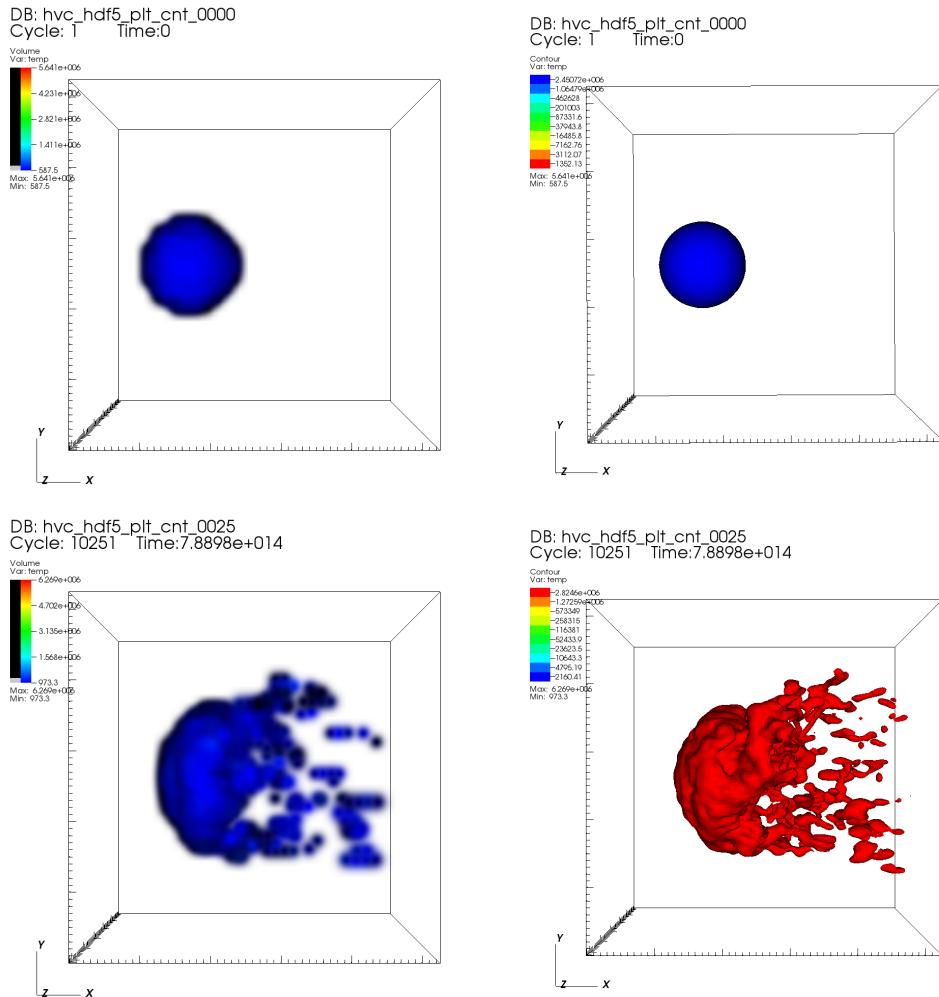


Abbildung 4.5: Unterschied Volumen-Plot (links) und Kontur-Plot

Für die Abbildung der Interaktivität mit den Plots wird für jeden Plot ein Zoom durchgeführt und anschließend eine Rotation bei den 3D und ein Panning bei den 2D Visualisierungen. Da diese beiden Operationen bei der Exploration von neuen Datensätzen sehr häufig angewendet werden, um ein Überblick über die Daten zu erlangen, ist diese einfache Simulation von Interaktion genügend.

¹⁰von engl. *to splatter*: “spritzen”

4.5.2 2. Szenario - Detail-View

In diesem wird nur ein Fenster verwendet, das ein hochauflösenden Volumen-Plot der Dichte anzeigt, das mit dem Raycasting Verfahren[SK94] erzeugt wird. Dieses Szenario stellt eine Untersuchung der Form und strukturellen Beschaffenheit der Daten ab. Dabei werden in dem Szenario ein Überblick, eine Rotation nach Rechts um ca. 45° und abschließen ein Zoom durchgeführt und als einzelne Bilder angezeigt. Im Folgenden eine Darstellung des zweiten Szenarios:

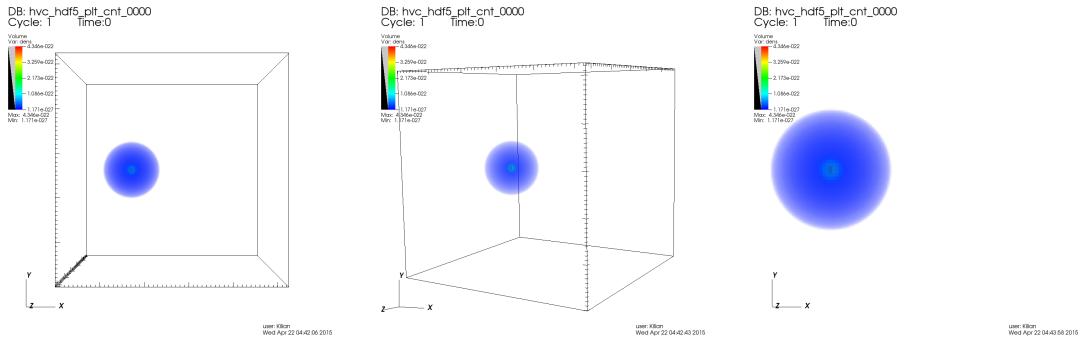


Abbildung 4.6: 2. Szenario - Detail-View

Im Gegensatz zu einem Volumen-Plot im Splatting-Verfahren hat Raycasting einen viel höheren Aufwand, wobei das Resultat jedoch feinere Strukturen der HVC darstellt. Um eine Echtzeitdarstellung zu ermöglichen, ist ein hoch parallelisierbares System wie z.B. ein Cluster via MPI[For94] nötig, in dieser Arbeit jedoch, wie bereits in der Einführung erwähnt, nicht zur Verfügung steht.

Dieses Szenario wurde mit Quelltext A.6 abgebildet.

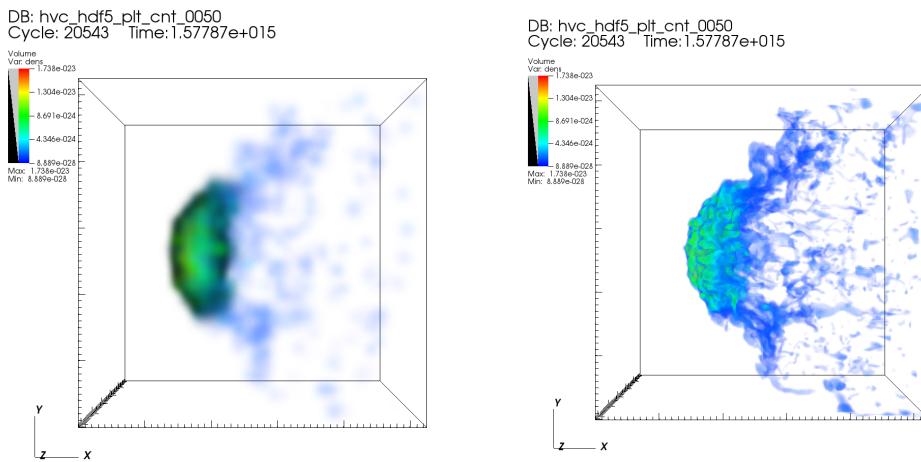


Abbildung 4.7: Unterschied Splatting (links) und Raycasting

Für jedes Szenario wurden die Zeitpunkte 0, 25, 50, 75 und 100 in der Simulation verwendet, um eine Varianz über die Komplexität der Wolke zu erfassen (siehe Abbildung 4.2) und visuell abzubilden. Für die Zeitmessungen wurden für jedes Szenario und jeden Zeitpunkt fünf Durchläufe gemacht und die Zeiten gemittelt.

4.6 Ergebnisse

In diesem Abschnitt werden die Ergebnisse des beschriebenen Experiments aufgeführt.

4.6.1 Zeiten für Downsampling

Mit dem NeonGenesisTool wurde der Beispieldatensatz (Abbildung 4.1) nach den verschiedenen Verfahren (Abschnitt 4.4) reduziert und die davon produzierten Log ausgewertet. Dazu wurde Java in der Version 8u40 64-Bit verwendet, die JVM mit den Standardparametern gestartet. Das Reduzieren einer Datei umfasst das Einlesen der Datei, das Aufbauen des Octree für die Struktur, das Downsampling der einzelnen Variablen und das Schreiben der neuen Datei. Für möglichst authentische Zeiten wurde jedes Verfahren fünfmal ausgeführt und die benötigten Zeiten gemittelt. Da die Abweichungen in den Laufzeiten sehr gering sind ($\sim 1\%$), wurde auf eine Anzeige der Abweichung mit Fehlerindikatoren verzichtet.

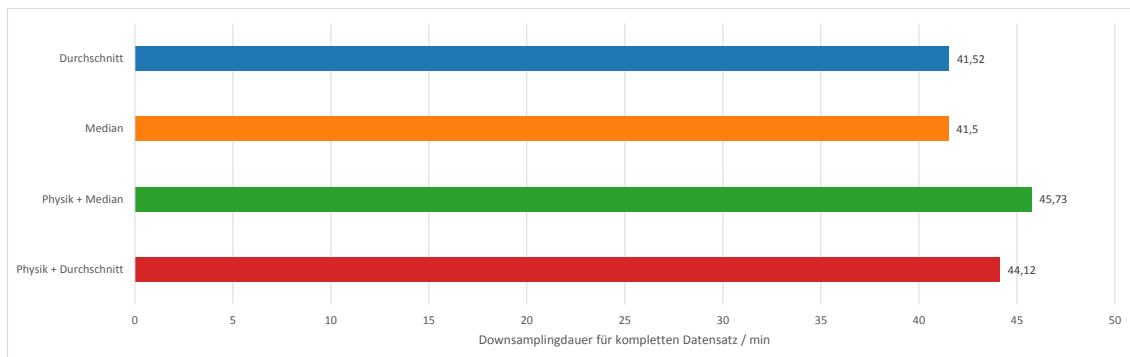


Abbildung 4.8: Zeit für Downsampling des Beispieldatensatzes

Für eine detaillierte Betrachtung des Prozesses wurden ebenfalls die Zeiten für das Reduzieren einer einzelnen Datei erfasst.

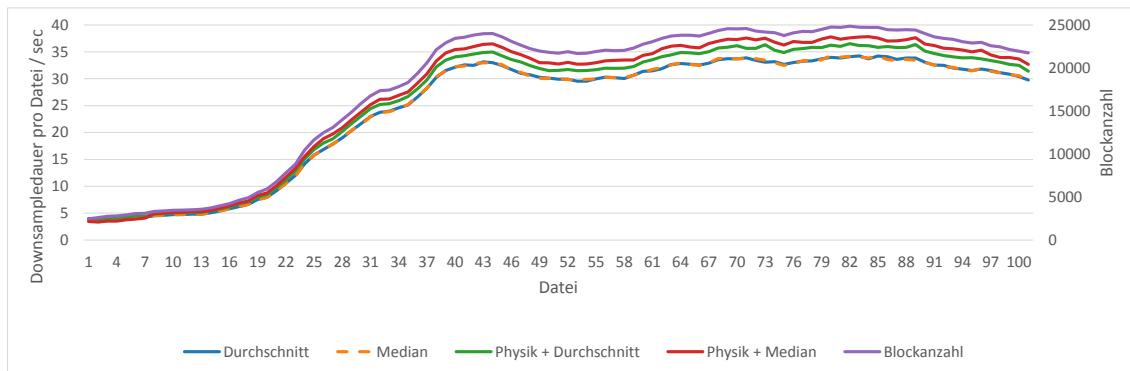


Abbildung 4.9: Zeit für Downsampling je Datei des Beispieldatensatzes

Dabei ist zu erkennen, dass die Wahl des Verfahrens eine Auswirkung auf die Zeit hat. Dabei sind die rein statischen Verfahren ca. 10% schneller als jene, die physikalische Formeln verwenden. Grund dafür ist der höhere Leseaufwand der HDF Datei, wenn die Mischtemperatur benötigt wird. Statt wie bei anderen Variablen nur die Matrix auszulesen, werden bei der Mischtemperatur zwei Matrizen geladen. Der Unterschied im Programm ist folgender:

```

1 float[] temp = new float[8];
2 float[] density = new float[8];
3 for (int i = 0, x = 0; x < xDim; x += 2) {
4     for (int y = 0; y < yDim; y += 2) {
5         for (int z = 0; z < zDim; z += 2, i++) {
6             // The temperature
7             temp[0] = flatOriginalTemperature[index(x, y, yDim, z, zDim)];
8             temp[1] = flatOriginalTemperature[index(x+1, y, yDim, z, zDim)];
9             temp[2] = flatOriginalTemperature[index(x, y+1, yDim, z, zDim)];
10            temp[3] = flatOriginalTemperature[index(x+1, y+1, yDim, z, zDim)];
11            temp[4] = flatOriginalTemperature[index(x, y, yDim, z+1, zDim)];
12            temp[5] = flatOriginalTemperature[index(x+1, y, yDim, z+1, zDim)];
13            temp[6] = flatOriginalTemperature[index(x, y+1, yDim, z+1, zDim)];
14            temp[7] = flatOriginalTemperature[index(x+1, y+1, yDim, z+1, zDim)];
15             // The density
16            density[0] = flatOriginalDensity[index(x, y, yDim, z, zDim)];
17            density[1] = flatOriginalDensity[index(x+1, y, yDim, z, zDim)];
18            density[2] = flatOriginalDensity[index(x, y+1, yDim, z, zDim)];
19            density[3] = flatOriginalDensity[index(x+1, y+1, yDim, z, zDim)];
20            density[4] = flatOriginalDensity[index(x, y, yDim, z+1, zDim)];
21            density[5] = flatOriginalDensity[index(x+1, y, yDim, z+1, zDim)];
22            density[6] = flatOriginalDensity[index(x, y+1, yDim, z+1, zDim)];
23            density[7] = flatOriginalDensity[index(x+1, y+1, yDim, z+1, zDim)];
24             // Calculate the mix temperature
25            result[i] = calculateMixTemperature(temperature, density);
26        }
27    }
28}

```

Quelltext 4.1: Extraktion der Werte für Mischtemperatur(Vereinfacht)

```

1 float[] array = new float[8];
2 for (int i = 0, x = 0; x < xDim; x += 2) {
3     for (int y = 0; y < yDim; y += 2) {
4         for (int z = 0; z < zDim; z += 2, i++) {
5             // The neighbors of the cell
6             array[0] = flatOriginal[index(x, y, yDim, z, zDim)];
7             array[1] = flatOriginal[index(x + 1, y, yDim, z, zDim)];
8             array[2] = flatOriginal[index(x, y + 1, yDim, z, zDim)];
9             array[3] = flatOriginal[index(x + 1, y + 1, yDim, z, zDim)];
10            array[4] = flatOriginal[index(x, y, yDim, z + 1, zDim)];
11            array[5] = flatOriginal[index(x + 1, y, yDim, z + 1, zDim)];
12            array[6] = flatOriginal[index(x, y + 1, yDim, z + 1, zDim)];
13            array[7] = flatOriginal[index(x + 1, y + 1, yDim, z + 1, zDim)];
14            result[i] = reducer.calculateMean(array);
15        }
16    }
17}

```

Quelltext 4.2: Extraktion der Werte für andere Variable(Vereinfacht)n

Ebenfalls ist die Berechnung der Mischtemperatur aufwendiger als die des Durchschnitts oder des Medians durch die Bildung von zwei Summen.

```

1 float calculateMixTemperature(float[] temps, float[] density) {
2     float res = 0.0F;
3     float sumDensity = 0.0F;
4     for (float dens : density) {
5         sumDensity += dens;
6     }
7     for (int i = 0; i < temps.length; i++) {
8         float temp = temps[i];
9         float dens = density[i];
10        res += ((temp * dens) / sumDensity);
11    }
12    return res;
13 }
```

Quelltext 4.3: Berechnung der Mischtemperatur(Vereinfacht)

```

1 float calculateMean(float[] array) {
2     double mean = 0.0F;
3     double length = (double) array.length;
4     for (float i : array) {
5         mean += (i / length);
6     }
7     return (float) mean;
8 }
```

Quelltext 4.4: Berechnung des Durchschnittes(Vereinfacht)

4.6.2 Zeiten für Szenarien

4.6.2.1 1. Szenario

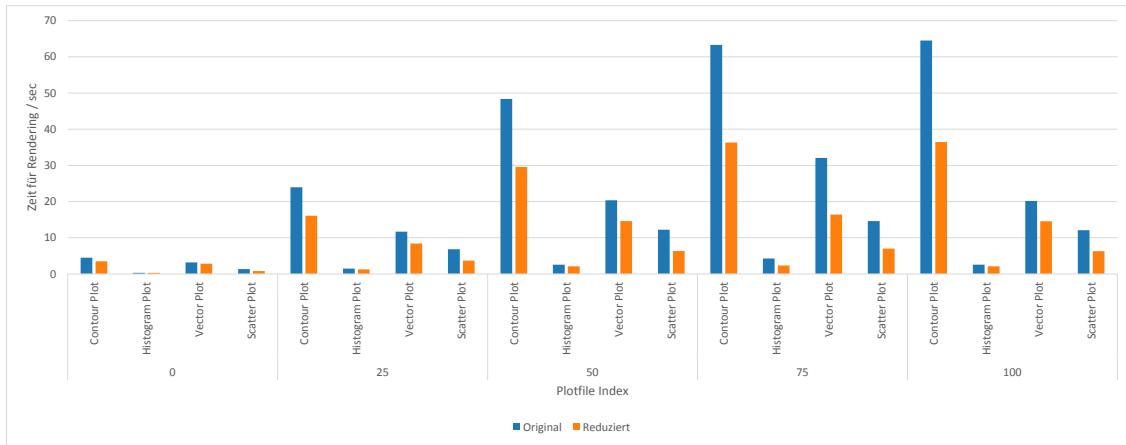


Abbildung 4.10: Vier Fenster - Detaillierte Rendering Zeiten

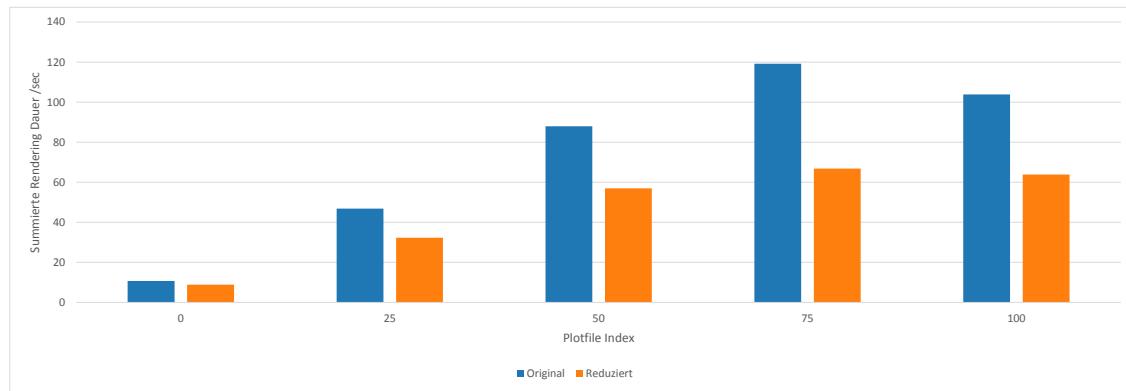


Abbildung 4.11: Vier Fenster - Kumulierte Rendering Zeiten

Bei diesem Szenario ist ein deutlicher Geschwindigkeitsgewinn beim Erstellen der Plots durch die Reduzierung zu erkennen. Ein Großteil der Zeit wird bei dem Kontur-Plot benötigt, wodurch die Reduzierung bis zu 40% der Zeit eingespart werden können. Auch ist deutlich zu erkennen, dass, je mehr Blöcke und je komplexer der Simulationsraum ist, desto größer sind die Zeitgewinne.

Zu vernachlässigen sind jedoch die Zeitgewinne bei dem Histogramm, da die totale Differenz auch bei komplexeren Raumstrukturen nur wenige Sekunden betragen.

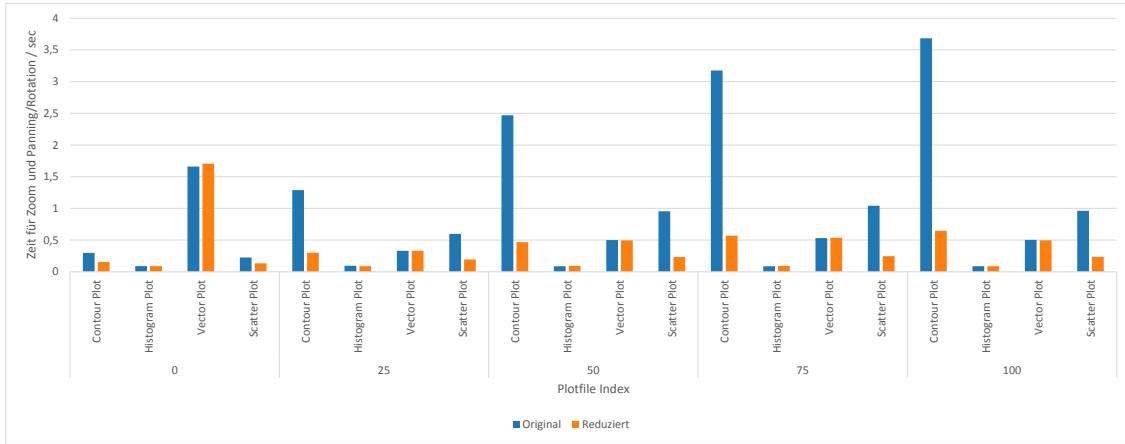


Abbildung 4.12: Vier Fenster - Kumulierte Zeiten für Interaktionen

Betrachtet man die Interaktion mit den Plots nach dem Rendering, sind Geschwindigkeitsvorteile in der Interaktion mit diesen zu erkennen. Während bei dem anspruchsvollsten Kontur-Plot die Latenzen bei der Interaktion mit den Originaldaten bei mehreren Sekunden liegen, so benötigt der Nutzer bei dem reduzierten Datensatz nur ein Bruchteil der Zeit. Als statistischer Ausreißer ist der Vector Plot zu bezeichnen, da trotz geringer Komplexität der Aufwand bei dem Anfangszustand sehr hoch und bei den anderen Zeiten nahezu konstant bleibt. Der Grund dafür ist mir unbekannt und kann an einer fehlerhaften Implementierung in VisIt liegen (siehe Abschnitt 5.6). Die Zeitgewinne bei dem Histogramm und beim Vektor-Plot sind durch die Reduzierung kaum messbar und betragen nur wenige Mikrosekunden. Dies ist für den Menschen nicht unterscheidbar und für den Nutzer entsteht dort kein Zeitgewinn. Bei dem Scatter-Plot wird hingegen ein spürbarer Zeitgewinn erreicht. Statt bis zu einer Sekunde warten zu müssen, werden die Befehle in bis zu einem Viertel der Zeit umgesetzt.

4.6.2.2 2. Szenario

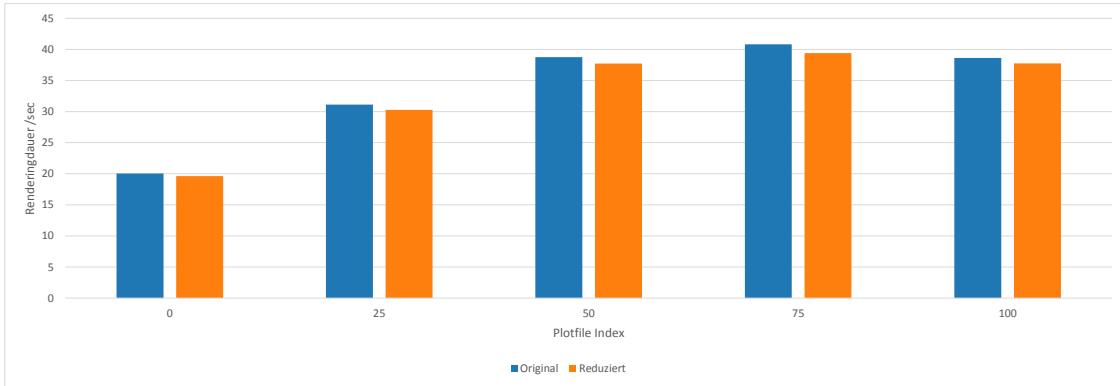


Abbildung 4.13: Detail-View - Detaillierte Rendering Zeiten

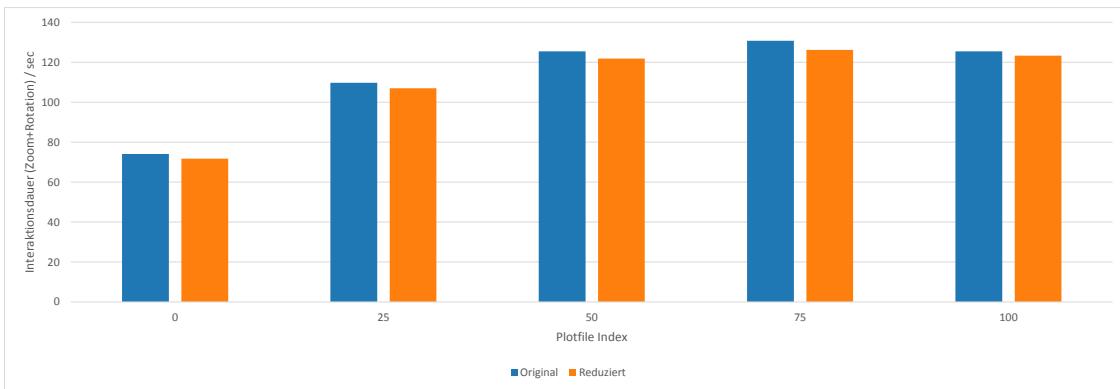


Abbildung 4.14: Detail-View - Kumulierte Zeiten für Interaktion

Bei diesem detaillierten Rendering hat die Reduzierung keinen Vorteil bezüglich der Geschwindigkeit ergeben. Es wurden zwar Zeittersparnisse bis zu 5% erreicht wie bei Index 75, allerdings ist die benötigte Zeit mit bis zu eineinhalb Minuten sehr groß und weit entfernt von einer interaktiven Visualisierung. Der Aufwand für das Reduzieren der Daten ist hier größer als der Zeitgewinn. Es ist auch entscheidend, dass jede Veränderung des Blickwinkels, sei es durch Zoom, Panning oder Rotation eine komplette Neuberechnung erfordert. Es ist daher eher zu empfehlen, in einem zweiten Fenster eine Darstellung via Splatting des Volumen zu haben und den Blickwinkel auf die Daten einzustellen. So wird unnötiges Zoomen, Rotieren oder Bewegen und damit eine Neuberechnung vermieden. Optimal für eine hochauflösende Exploration der Struktur ist jedoch auch diese Vorgehensweise nicht.

4.6.3 Visualisierungen

In diesem Abschnitt werden die Visualisierungen der Daten, die mit den verschiedenen Verfahren reduziert worden sind, miteinander verglichen. Dabei ist wichtig, dass ein Informationsverlust und somit ein Bildunterschied immer existiert. Es ist hierbei die Frage, ob das Bild der reduzierten Daten den Gesamteindruck verändert oder ob die Unterschiede kosmetischer Natur sind. Um zu verhindern, dass zu viele ähnliche Bilder erscheinen, werden pro Verfahren nur die Zeitschritte 0, 25 und 75 abgebildet. Dies entspricht drei verschiedenen komplexen Zuständen der Wolke und der Unterschied zu den anderen beiden 50 und 100 ist wie in Abbildung 4.1 zu entnehmen gering.

Da der Mensch Veränderungen in komplexen Visualisierungen nicht erkennen kann [NHT01, MXW⁺13], wurden Differenzbilder erstellt. Dazu wurde die folgende Formel verwendet, wobei D das Differenzbild, O das Bild mit den original Daten, R das Bild mit den reduzierten Daten ist und $D(x,y)$, $O(x,y)$, $R(x,y)$ den jeweiligen **RGB** Farbwert an der Stelle x und y entspricht:

$$D(x,y) = |O(x,y) - R(x,y)|$$

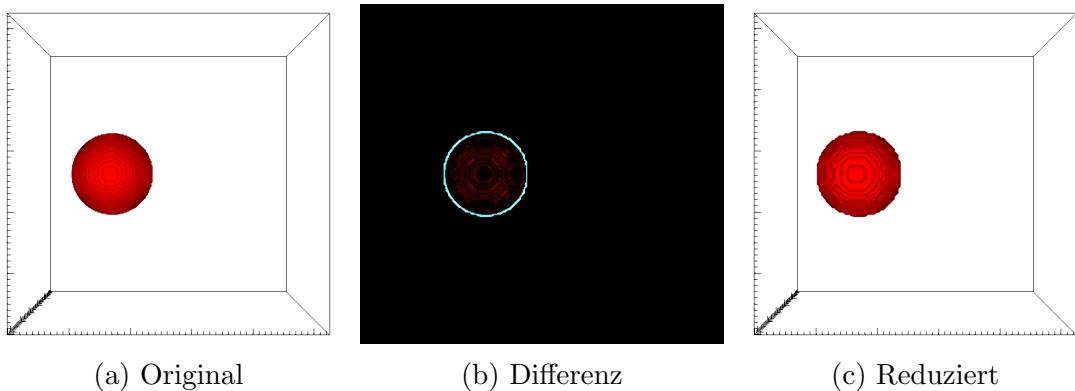


Abbildung 4.15: Beispiel für Differenzbild (Kontur-Plot)

Je geringer die Entfernung der Farbwerte in einem Pixel ist, desto dunkler wird das Resultat, da der Wert gegen 0 geht und dies Schwarz im **RGB**-Farbraum entspricht. Das bedeutet, dass je ähnlicher die Bilder sind, desto dunkler erscheint das Differenzbild und bei identischen Bildern ist das Bild komplett schwarz. Starke Abweichungen erscheinen in helleren Farben und werden so hervorgehoben.

4.6.3.1 Kontur-Plot

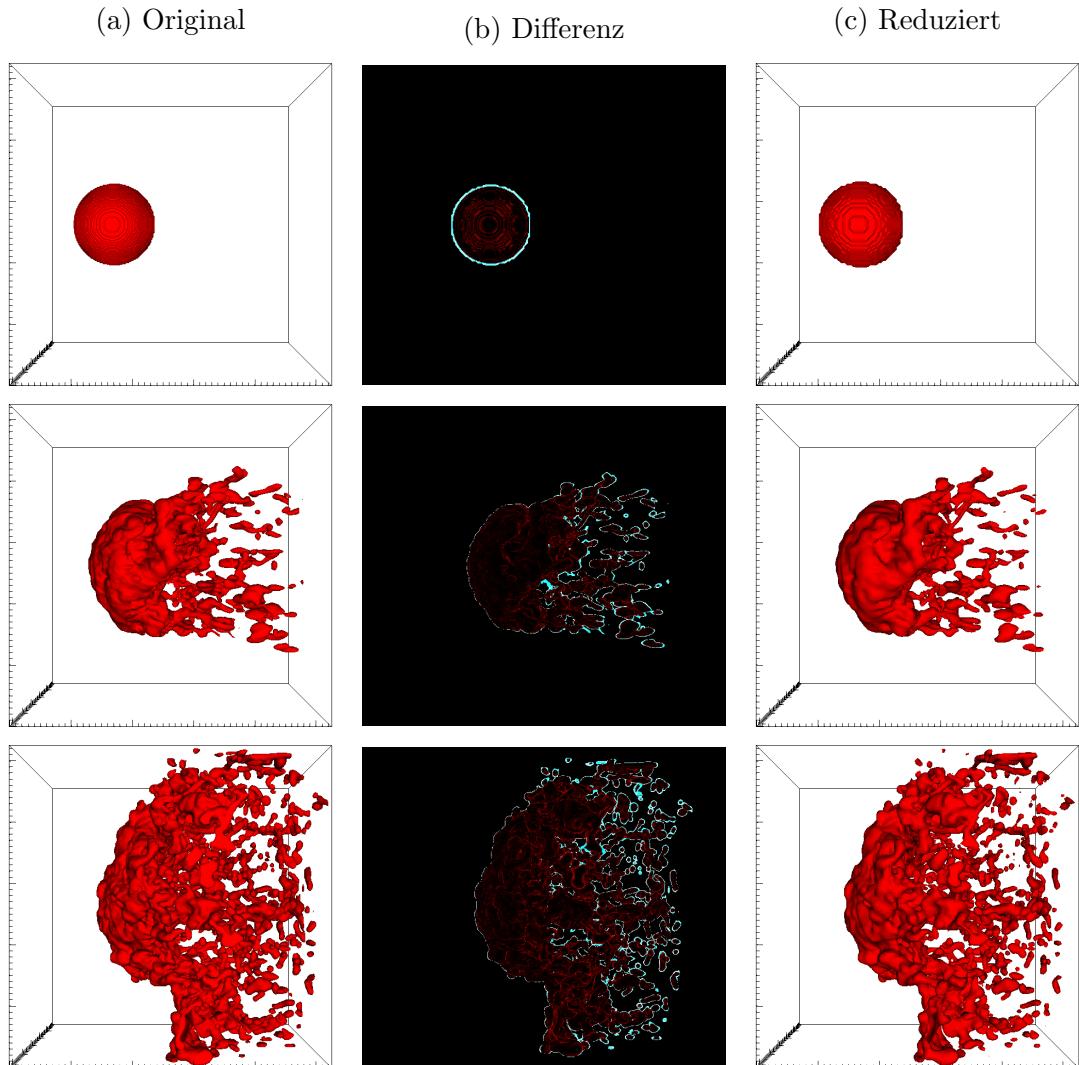


Abbildung 4.16: Kontur-Plot - Durchschnitt

Bei dem Durchschnitt ist auf den ersten Blick kein Unterschied zu erkennen. Die räumliche Struktur bleibt weitestgehend erhalten. Bei der runden Kugel am Anfang ist vor allem ein Unterschied an den Rändern der Kugel zu betrachten, da dort sowohl hohe als auch niedrige Werte nah bei einander liegen und somit der Durchschnitt sich weit von den eigentlichen Werten entfernt.

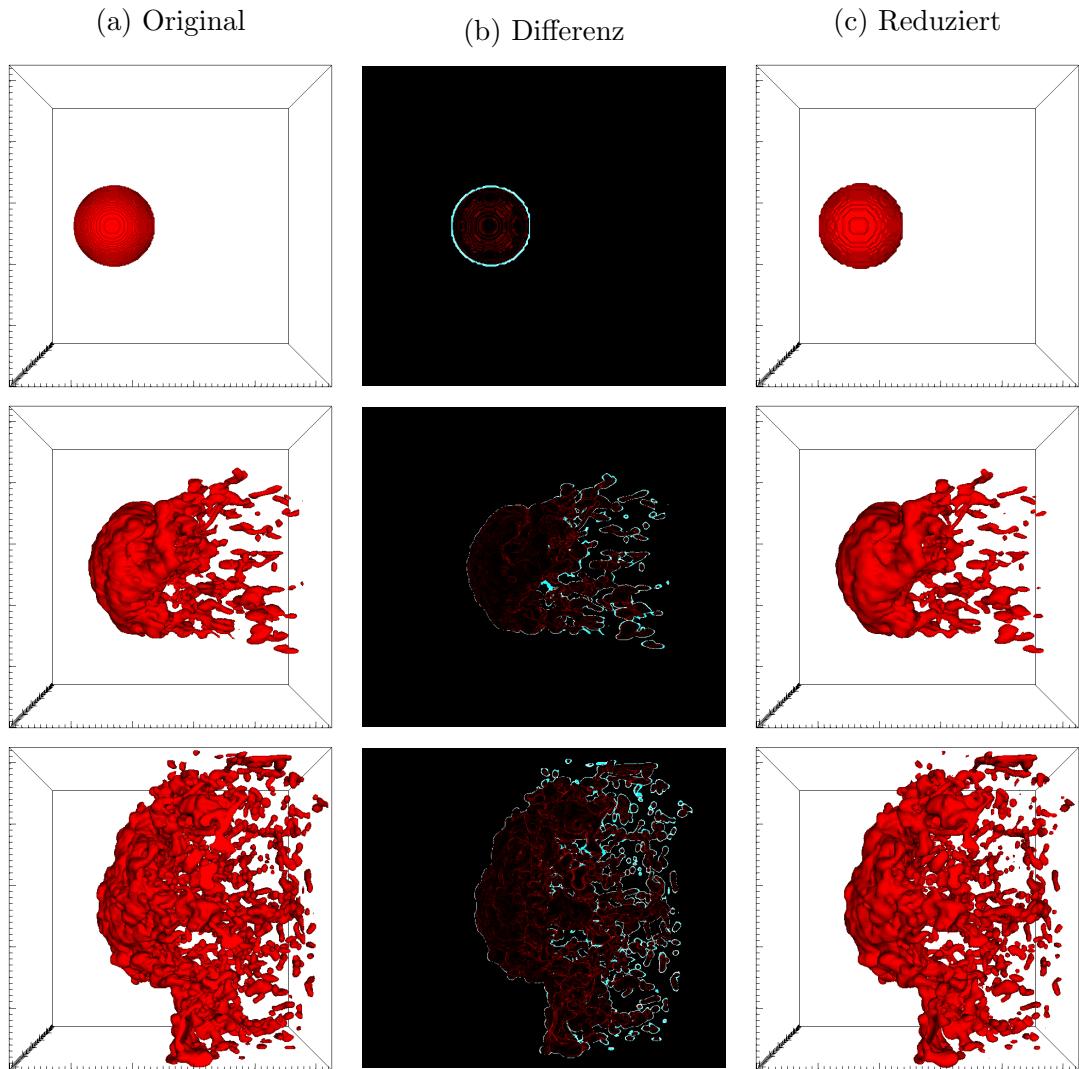


Abbildung 4.17: Kontur-Plot - Median

Bei dem Median findet sich ein gleiches Bild wie bei dem Durchschnitt. Es existiert zwischen den beiden Verfahren keinerlei erkennbarer Unterschied.

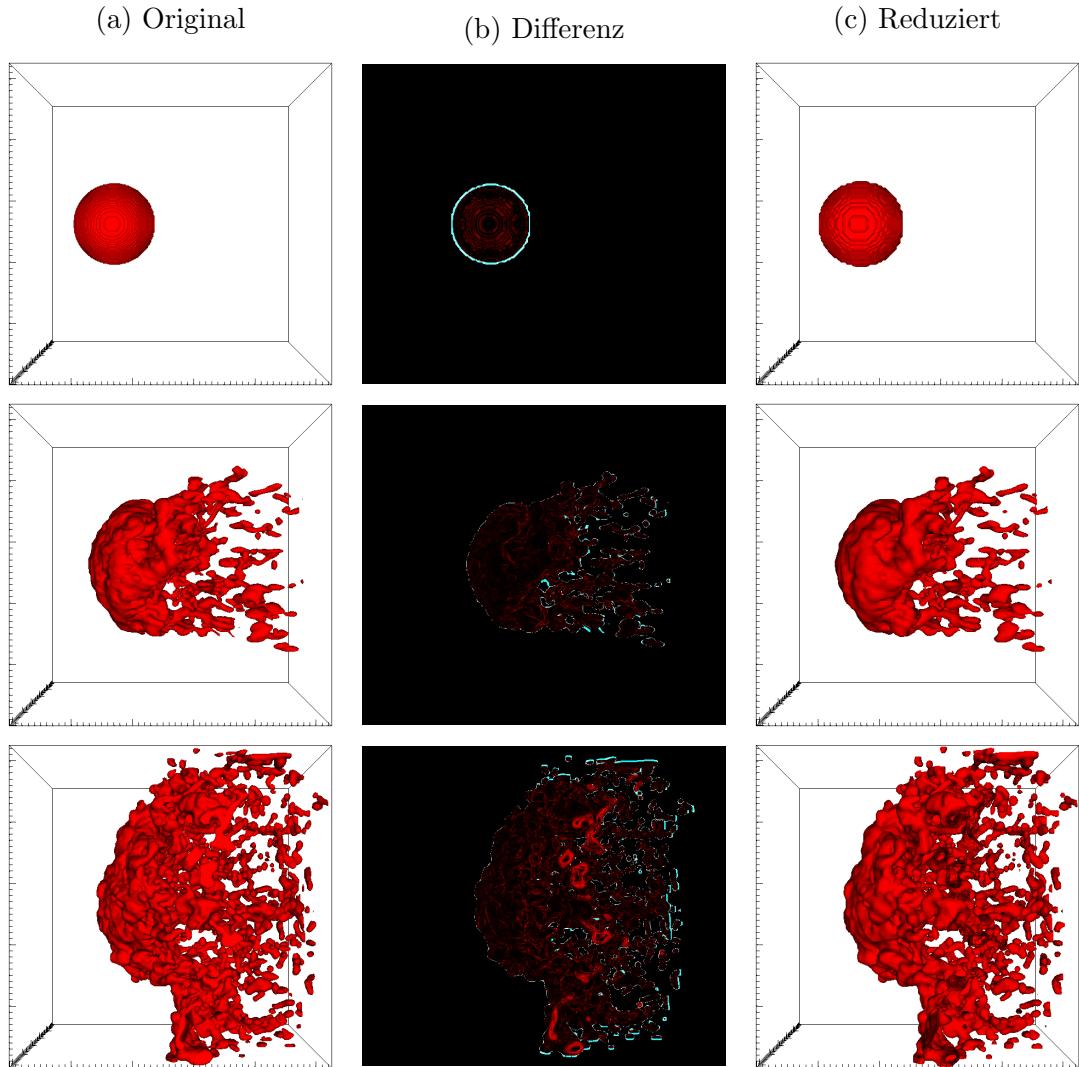
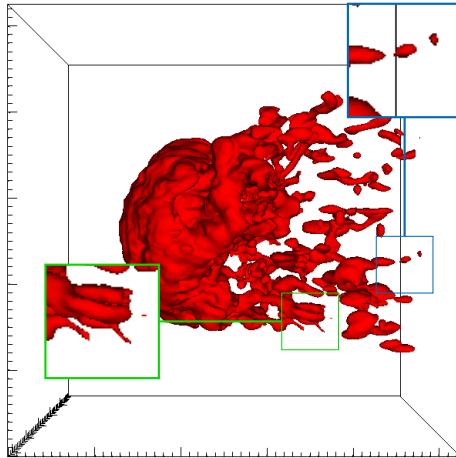


Abbildung 4.18: Kontur-Plot - Mischtemperatur

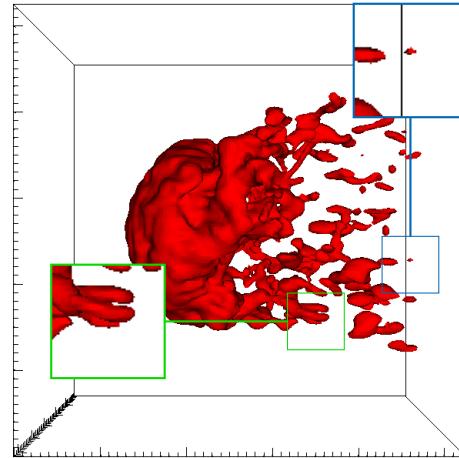
Bei der Mischtemperatur hingegen zeichnet sich ein anders Bild ab. Vor allem bei komplexeren Strukturen erweist sich das Differenzbild als dunkler und somit ähnlicher. An den Rändern sind die Unterschiede geringer als bei dem Durchschnitt oder dem Median. Auffällig sind jedoch die Unterschiede an den Grenzen des Simulationsraumes.

Betrachtet man die Bilder im Detail, wird offensichtlich, dass kleinere Partikel bei dem Durchschnitt und dem Median nicht mehr sichtbar sind.

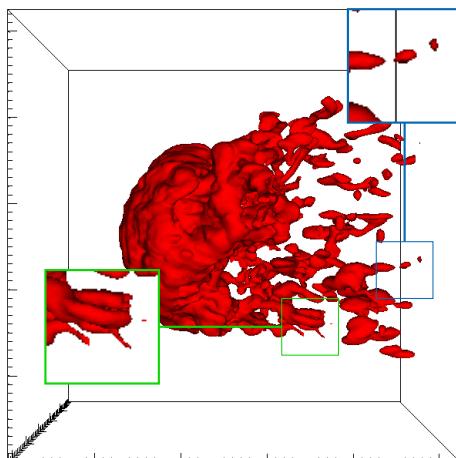
Original



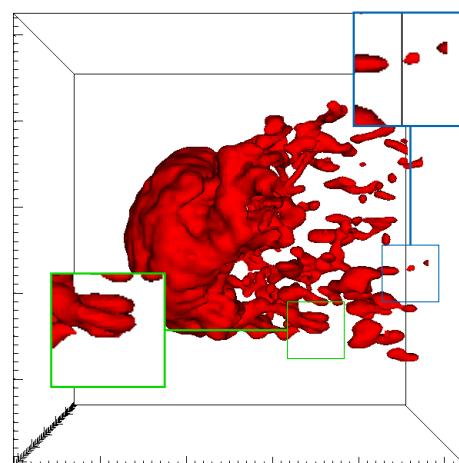
Reduziert



(a) Durchschnitt



(b) Median



(c) Mischtemperatur

Insgesamt ist zu sagen, dass die Mischtemperatur bei einem Kontur-Plot nach diesen Ergebnissen das originalgetreuere Bild liefert.

4.6.3.2 Vektor-Plot

Hier wird nur die Verfahren Durchschnitt und Median mit dem Original verglichen, da die Mischtemperatur auf den Vektor-Plot der Geschwindigkeit keine Auswirkung hat.

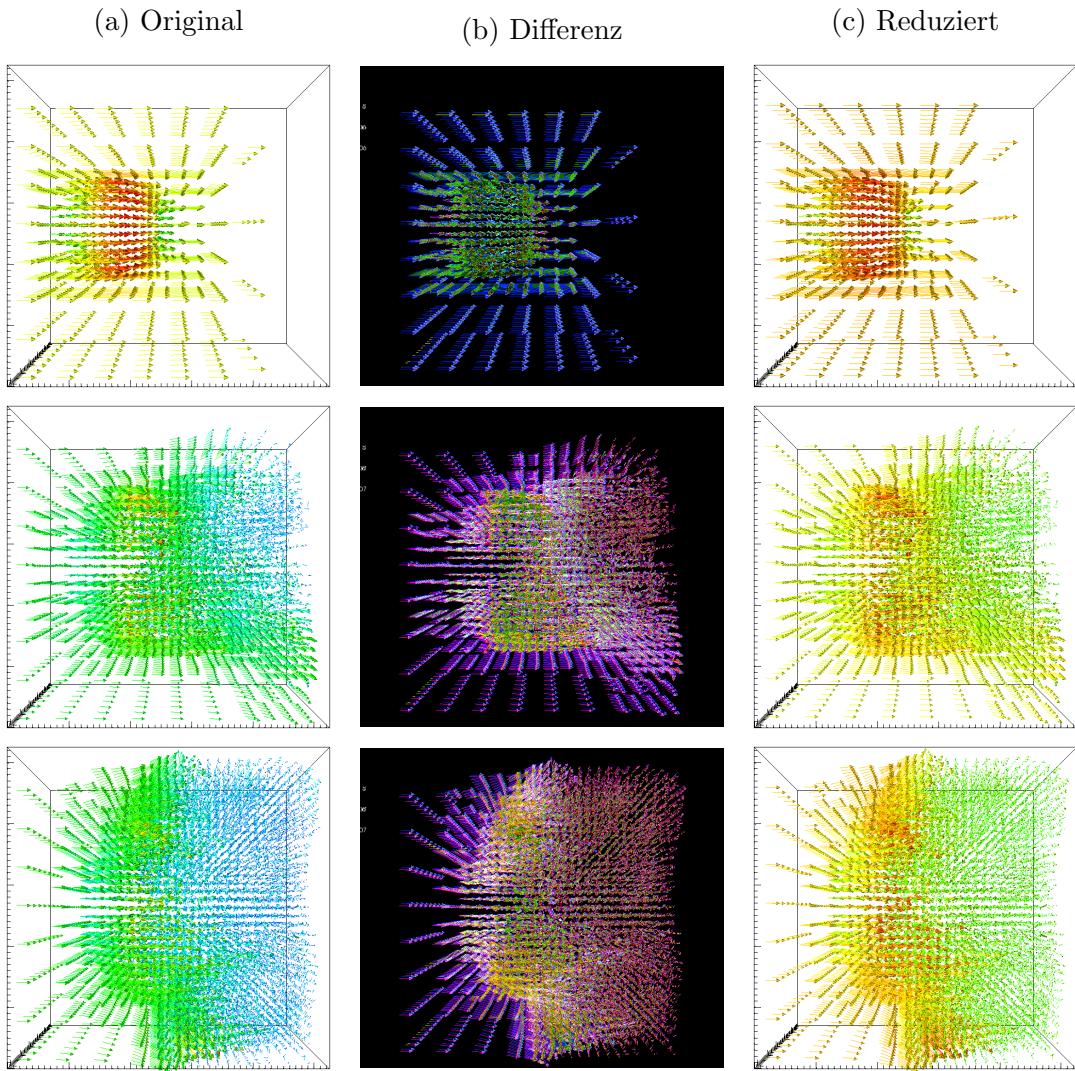


Abbildung 4.20: Vektor-Plot Durchschnitt

Auf den ersten Blick fällt nur die unterschiedliche Färbung der Vektoren auf, die Richtung jedoch erhalten bleibt. Da für das reduzierte und das originale Bild die selbe Skala verwendet wurde, treten diese Farbunterschiede auf. Bei einer Anpassung der Skala ergibt sich ein Bild, mit dem gearbeitet werden kann.

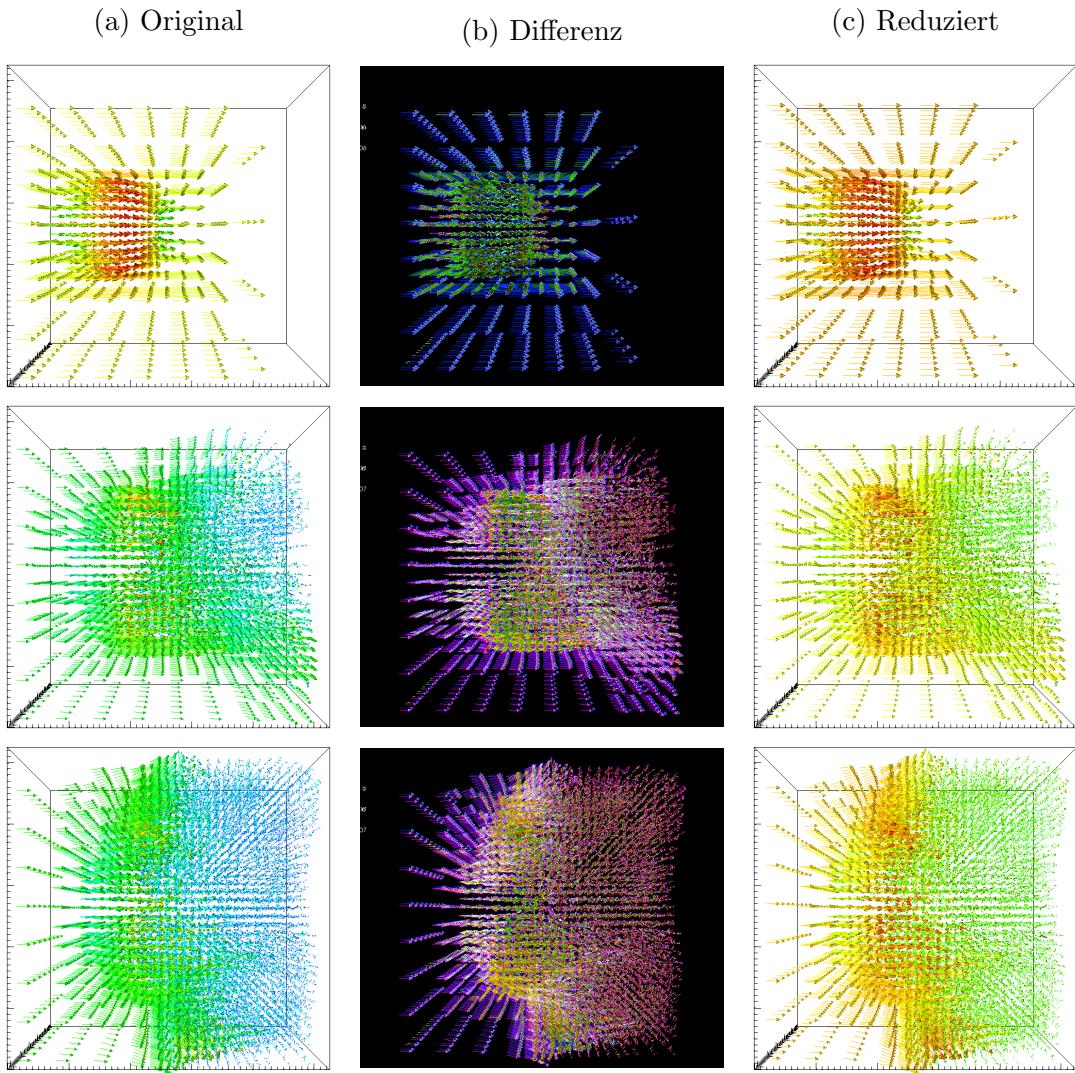


Abbildung 4.21: Vektor-Plot Median

Bei dem Median ergibt sich das gleiche Bild wie bei dem Durchschnitt. Es treten keine Unterschiede in der Struktur auf.

Beide Verfahren bieten Bilder, die verwendet werden können. Jedoch sollten für Analysen der Werten die Originaldaten verwendet werden. Für einen Überblick der Verläufe der Geschwindigkeit ist die reduzierte Version jedoch vollkommen ausreichend.

4.6.3.3 Histogramm

Wie auch bei dem Vektor-Plot in Abschnitt 4.6.3.2 werden nur der Durchschnitt und der Median bei dem Histogramm untersucht, da die Misctemperatur keine Auswirkung auf dieses hat.

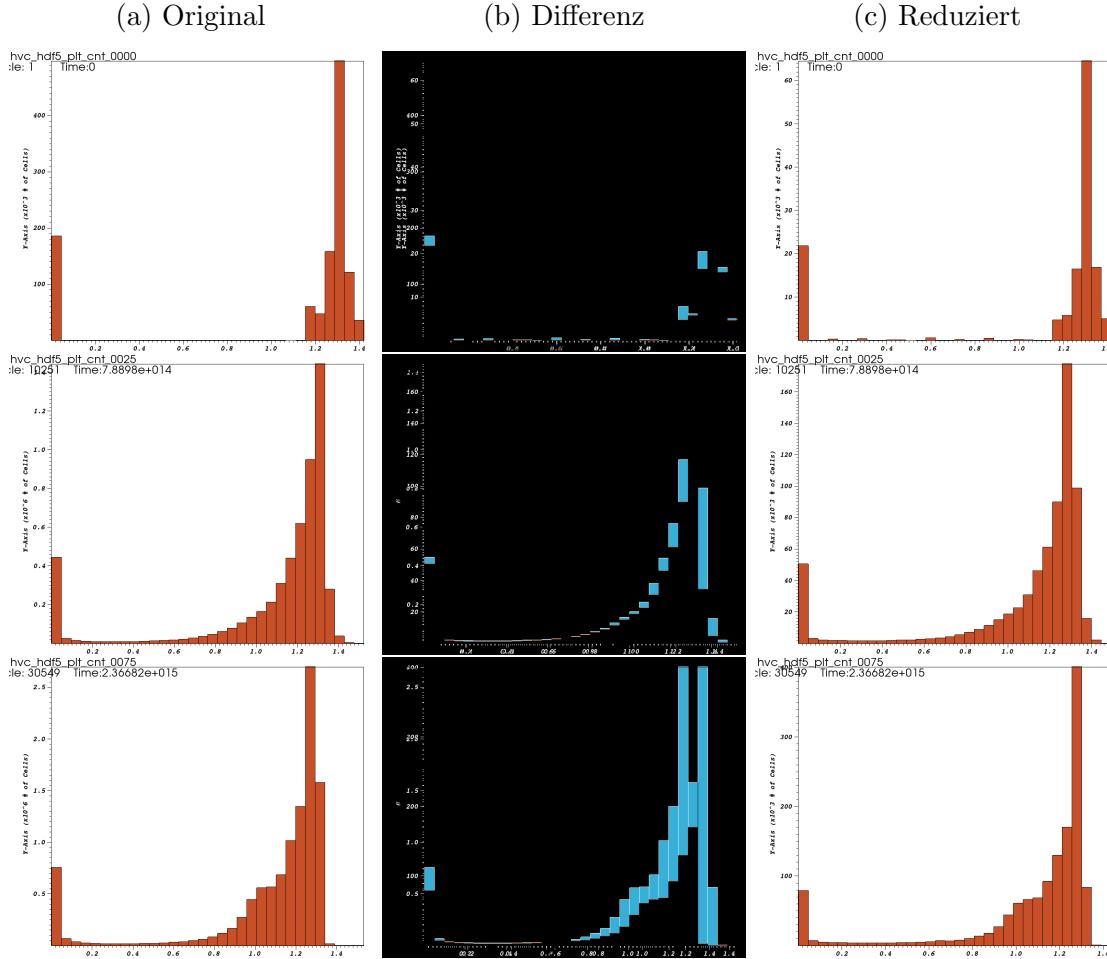


Abbildung 4.22: Histogramm - Durchschnitt

Im Gegensatz zu den vorherigen Plots fallen hier die Unterschiede gravierend aus. Es verändert sich nicht nur die Struktur in der Anordnung der Balken, es verändert sich auch massiv die Skala. Bei dem ersten Bildtripel ist im Original das Maximum bei ca. 500×10^3 Zellen und es existieren nur Zellen mit wenig bis keiner Energie und welche mit einem hohen Anteil. Dazwischen existieren keine. Bei dem Durchschnitt sind jedoch solche Zellen vorhanden und die Skalierung verändert sich stark, diese hat nun ein Maximum an Zellen von ca. 65×10^3 Zellen. Auch in späteren Zeitpunkten sind die Verläufe verschoben und weichen stark vom Original ab.

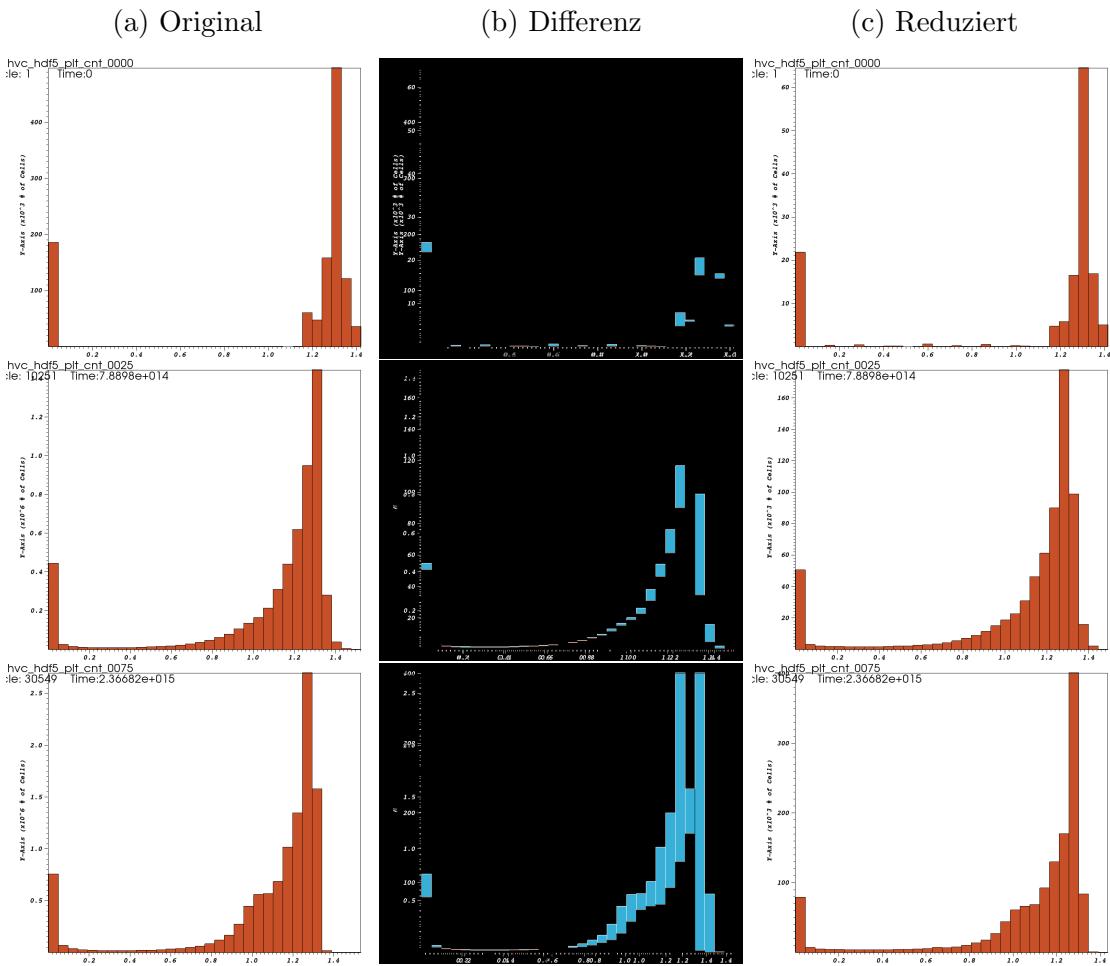


Abbildung 4.23: Histogramm - Median

Wie auch bei den anderen Verfahren zeigt sich ein ähnliches Bild bei dem Median wie bei dem Durchschnitt.

Weder Median noch Durchschnitt sind für diese Art von Plot geeignet, da andere Werte und Verteilungen suggeriert werden. Hier ist mit den Originaldaten zu arbeiten.

4.6.3.4 Scatter-Plot

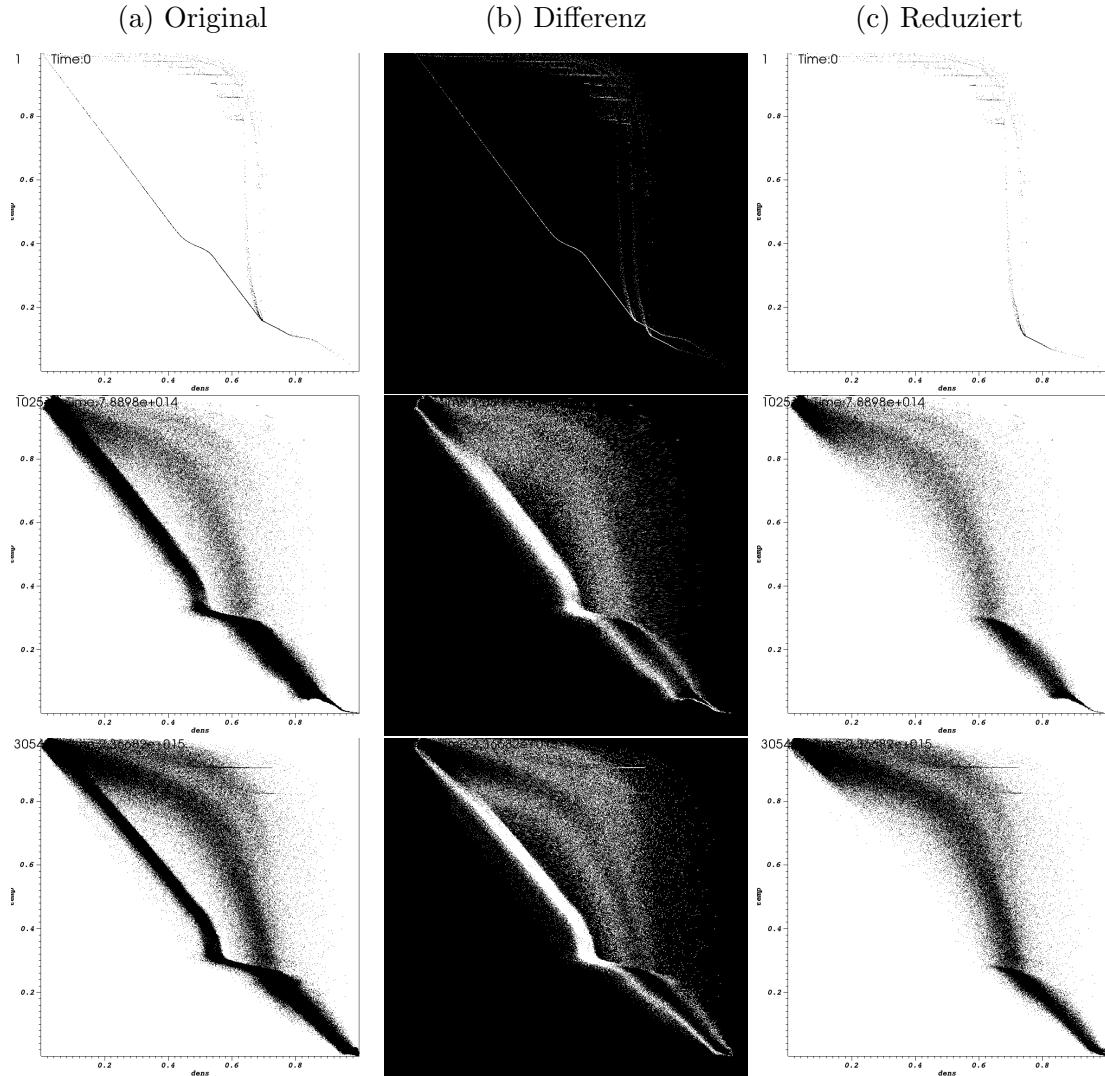


Abbildung 4.24: Scatter-Plot - Durchschnitt

Ähnlich dem Verhalten bei dem Histogramms sind hier die Unterschiede sehr deutlich. So existiert eine negative Korrelation zwischen der Temperatur und der Dichte. Je wärmer das Gas, desto stärker dehnt es sich aus und dadurch sinkt die Dichte. Diese Verhalten kann bei dem Original gut erkannt werden. Jedoch existiert die nahezu gerade Linie bei dem reduzierten Bild gar nicht. Es wird nur die Streuung oberhalb der nicht mehr vorhandenen Linie abgebildet.

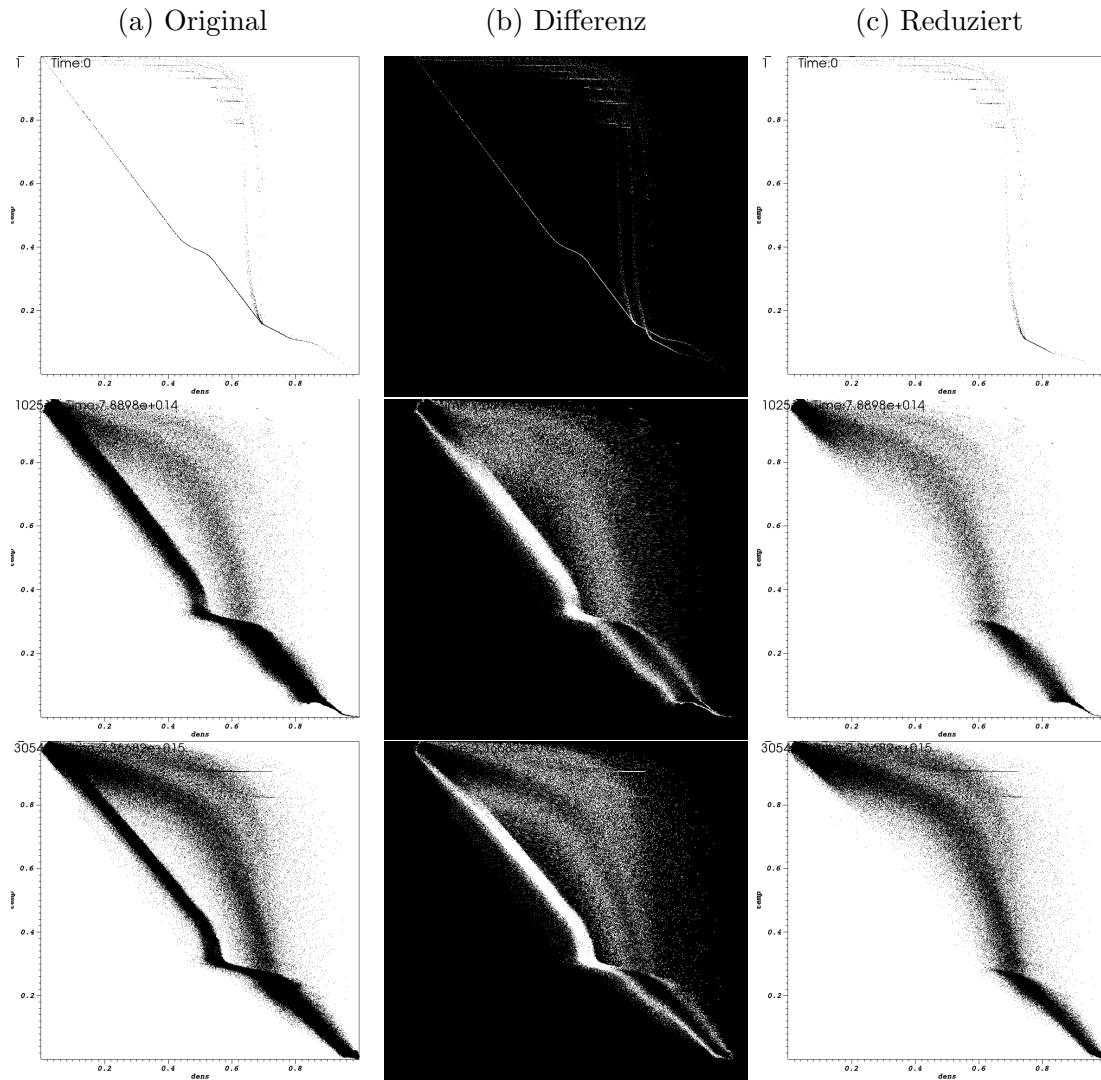


Abbildung 4.25: Scatter-Plot - Median

Der Median zeigt ein ähnliches Bild wie bei dem Durchschnitt. Es gehen auch hierbei wichtige Informationen verloren und der Gesamteindruck wird verfälscht.

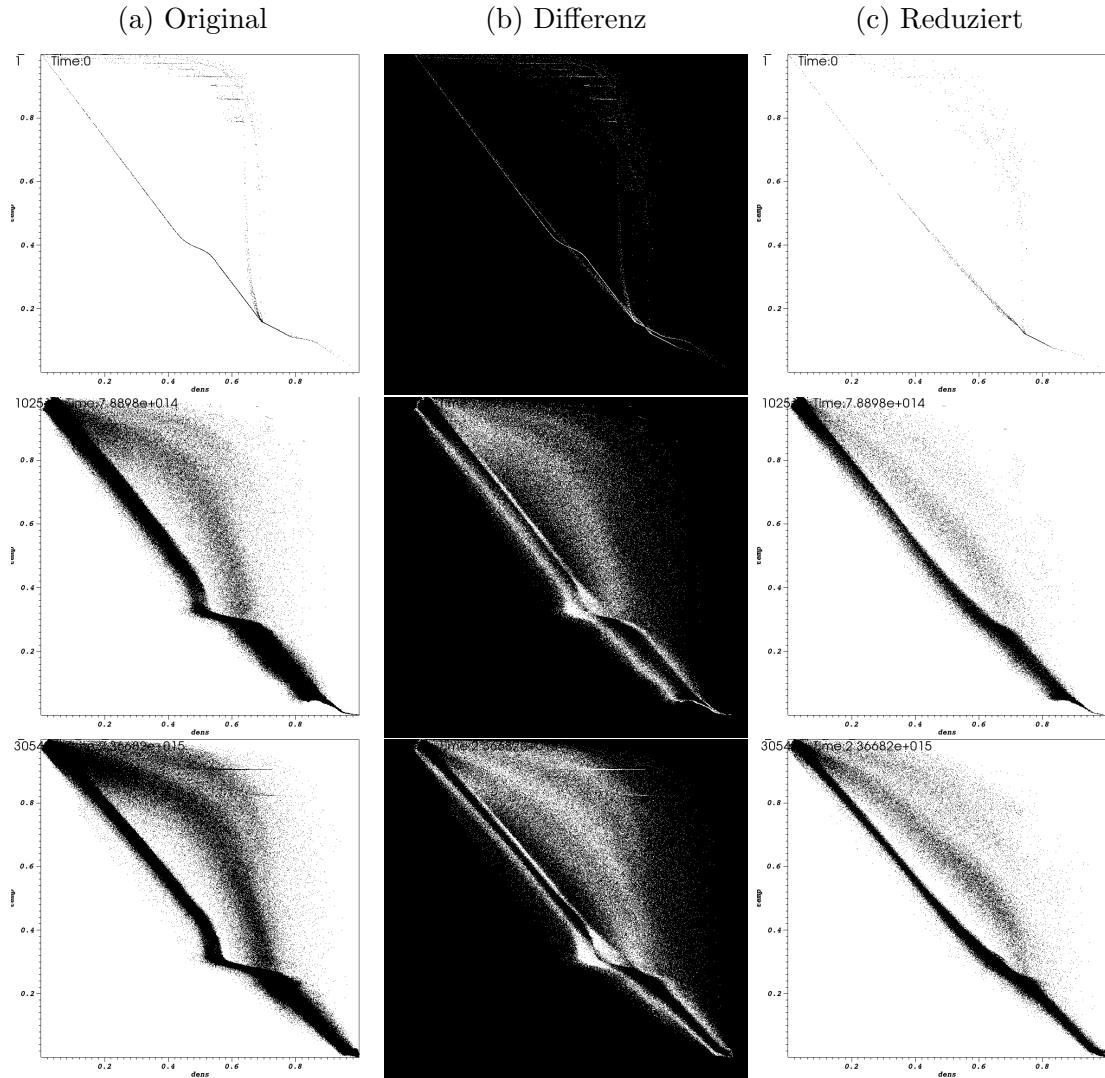


Abbildung 4.26: Scatter-Plot - Mischtemperatur

Die Mischtemperatur bildet hinsichtlich der negativen Korrelation ein dem Original ähnliches Bild ab. So wird im Gegensatz zu den anderen Verfahren die Linie ähnlich abgebildet, jedoch die Ausreißer oberhalb werden gemindert. So bildet die Mischtemperatur ein besseres Bild, wobei jedoch wichtige Details wie der Ausreißer im unteren Bildausschnitt und die Streuung nach rechts im oberen Teil verloren gehen.

Hier sollte deswegen eher mit dem Originaldaten gearbeitet werden, wenn es die Geschwindigkeit zulässt.

4.6.3.5 Volumen-Plot

Da die Dichte nicht von der Mischtemperatur betroffen ist, werden hier auch wie in Abschnitt 4.6.3.2 und Abschnitt 4.6.3.3 nur Median und Durchschnitt betrachtet.

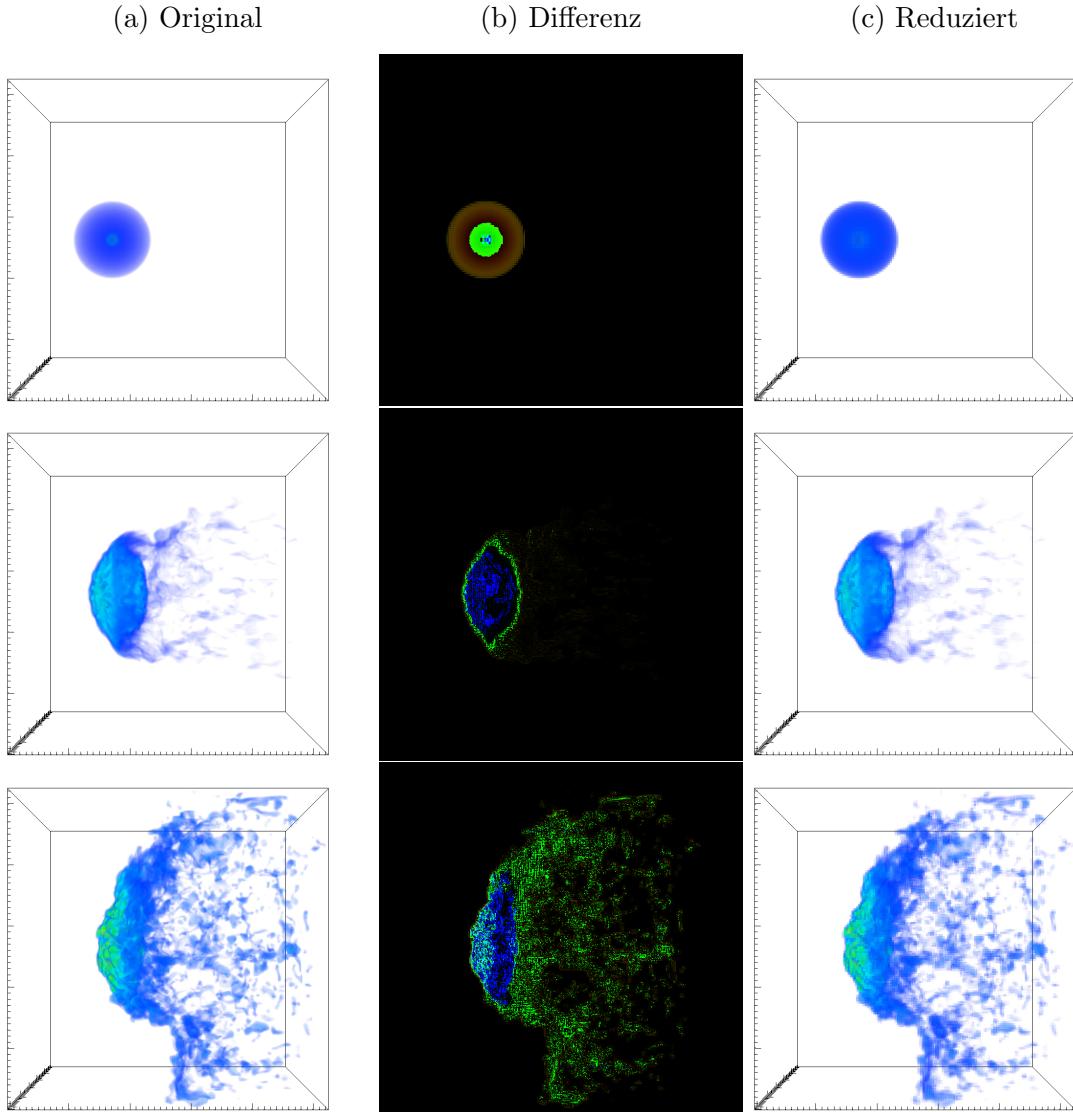
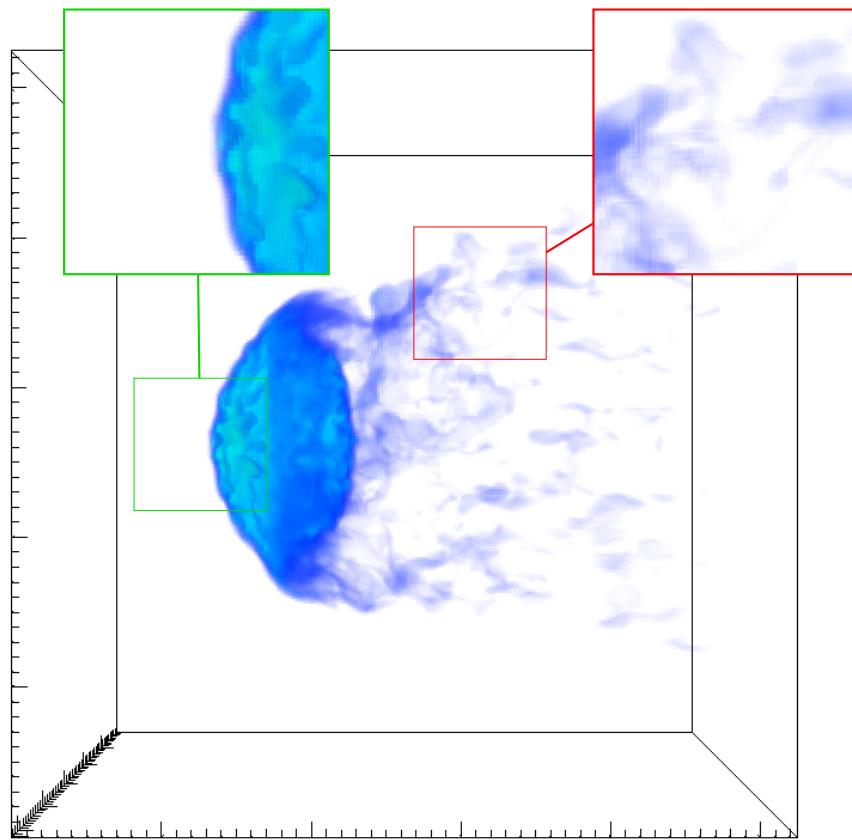
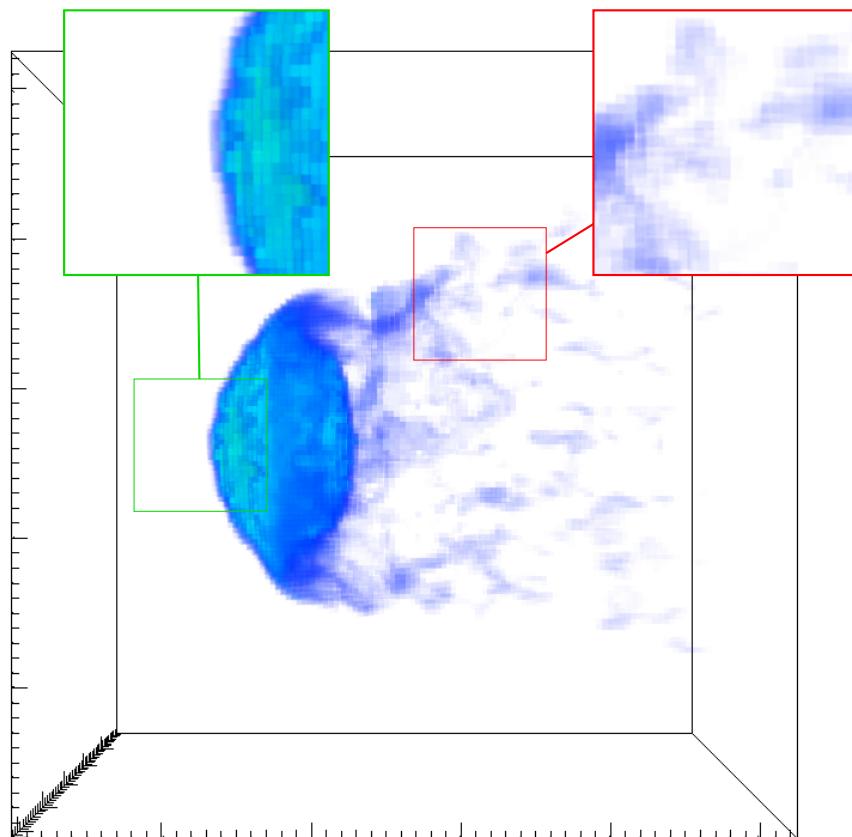


Abbildung 4.27: Volumen-Plot - Durchschnitt

Auf den ersten Blick unterscheiden sich die reduzierten Bilder nicht von dem Original. Die Form und scheinbar einzelne Partikel werden erhaltend wiedergegeben. Nur über das Differenzbild sind die Unterschiede sichtbarer.



Original



Reduziert

Abbildung 4.28: Volumen-Plot - Unterschied im Detail

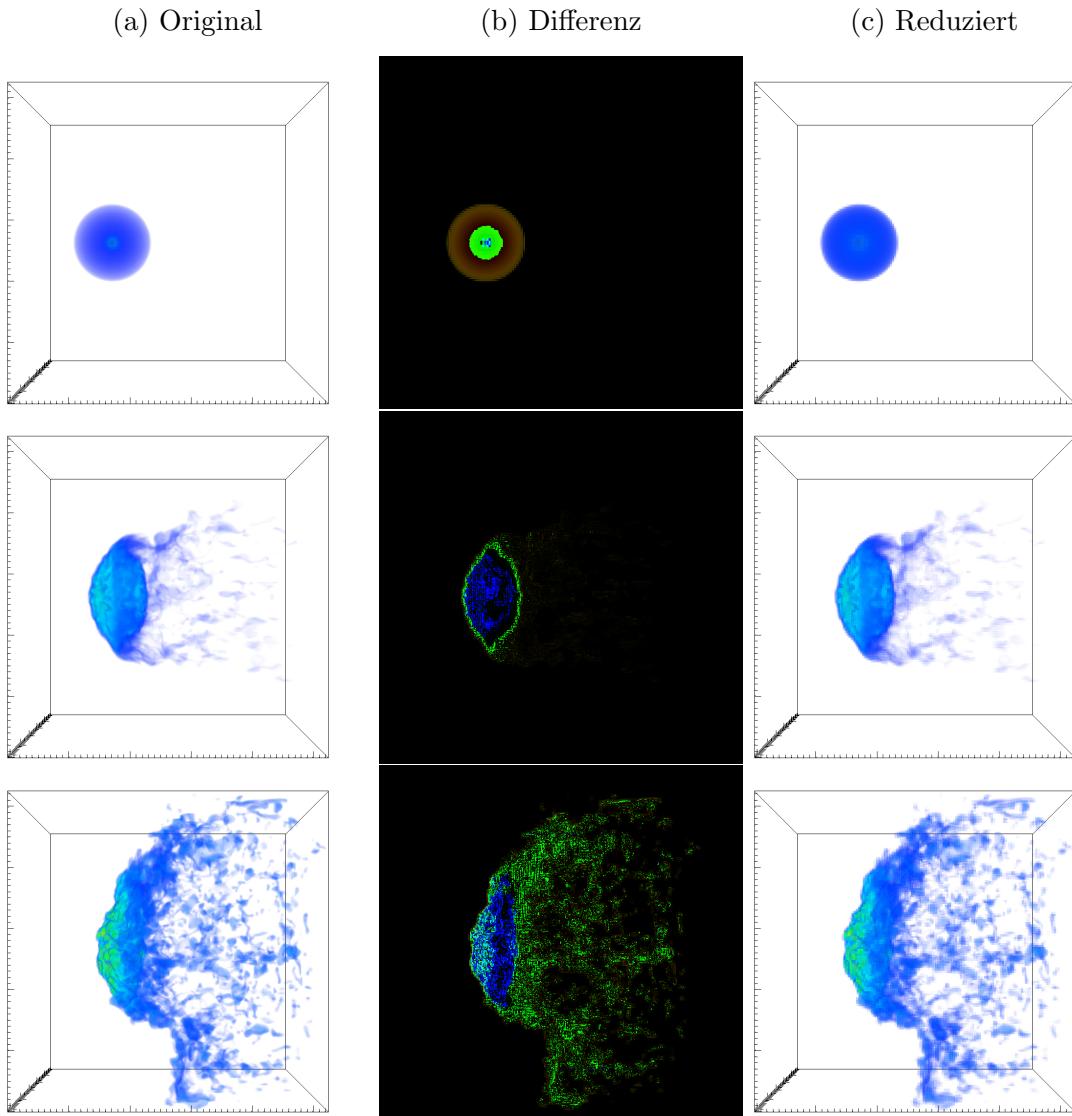


Abbildung 4.29: Volumen-Plot - Median

Wie auch in den anderen Verfahren zu erkennen war, unterscheiden sich der Median und der Durchschnitt auch in diesem Plot nicht sichtbar voneinander.

Bei beiden Verfahren erzeugt die Reduzierung keine direkt erkennbaren Unterschiede, das Bild wirkt nur ein wenig unschärfer, die Struktur bleibt weiterhin ersichtlich.

5. Zusammenfassung

5.1 Bildgüte

Der Unterschied zwischen dem Median und dem Durchschnitt als Mittel zum Down-sampling ist so gering, dass das menschliche Auge diese nicht wahrnehmen kann. So-mit ist auch der Aufwand für die unterschiedlichen Implementierungen einzusparen. Auch das Differenzbild hat bei dem Vergleich keine erkennbaren Ergebnisse gebracht. Deswegen ist die Entscheidung, ob Median oder Durchschnitt für ein generelles Down-sampling zu nehmen, in den gezeigten Plots und den verwendeten Daten trivial. Bei dem Vektor-Plot kann mit minimalen Einschränkungen der reduzierte Datensatz verwendet werden, solange für eine genaue Analyse auf Datenbasis die Originaldaten verwendet werden. Bei dem Histogramm dagegen wird der Gesamteindruck derart verfälscht, dass eine Nutzung von reduzierten Daten dafür vermieden werden sollte. Die Wahl der Mischtemperatur als gewichtetes Mittel erzeugt jedoch ein unterschiedliches Bild als die des Medians und des Durchschnittes. So zeigte sich bei dem Kontur-Plot im Detail eine original treuere Abbildung als bei den anderen Verfahren. Kleinere Partikel blieben sichtbar, wenn auch nicht alle. Bei dem Scatter-Plot blieb die Hauptaussage der negativen Korrelation eher erhalten als bei dem Median und dem Durchschnitt.

Bei dem Volumen-Plot ist ein deutlicher Unterschied zu erkennen, jedoch bleibt die Gesamtaussage über die Struktur erhalten. Die Analyse einzelner, feiner Strukturen wird jedoch erschwert.

5.2 DownSampling Zeit

Das DownSampling als Preprocessing Schritt wird, da extern, einmalig durchgeführt. Deswegen ist die Zeit von ca. einer Stunde als einmaliger Aufwand akzeptabel. Mit weiteren Verfeinerung wie parallelisierten Verarbeiten kann diese Zeit definitiv noch verringert werden, weil jede Datei für sich steht und einzeln berechnet werden kann. Eine weitere Möglichkeit zur Steigerung der Performance ist die Verwendung eines rein Java basierten API. Die von der HDFGroup gelieferte Java-API für HDF5 Da-teien bietet keine Möglichkeit zum Schreiben von Teilen eines Datensatzes, es muss

die komplette Matrix in einem Stück geschrieben werden. So wird jeder Block und dessen Daten einzeln gelesen, am Ende jedoch unnötigerweise zu einem sehr großen Array zusammengefasst. Dieser Schritt des mehrmaligen Kopierens von Speichers kann sehr viel Zeit beanspruchen.

Auch besteht die Gefahr, dass die zu schreibende Matrix größer ist als die maximale Anzahl an Elementen in einem Array in Java. Da Java Arrays über 32-Bit signed Integer indiziert werden, ist die maximale Größe für ein Array ca. 2^{31} . Bei Reduzierungen von sehr komplexen Daten mit hohen Auflösungen wie 64^3 zu 32^3 Zellen je Block überschreitet dies den Wert sehr schnell.

Als Lösung für die Performance und für die Limitierung sollte deswegen ein anderes API verwendet werden, welches gepufferte Schreiboperationen zulässt.

5.3 Datengröße

In dem verwendeten Verfahren wurde die Datenmenge um ein Achtel(Overhead vernachlässigt) reduziert. Dies geschah durch die Transformation von 512 4-Byte zu 64 4-Byte Werten je Block. Dadurch werden die Beispieldaten von 41,7 Gigabyte auf 5,40 Gigabyte reduziert. Zur Relation der Daten: Der Originaldatensatz konnte nicht komplett in den RAM des Testsystems geladen werden, während der reduzierte Datensatz komplett im RAM verarbeitet werden könnte.

Die verwendeten Daten könnten ein weiteres mal reduziert werden auf 8 4-Byte Werte pro Block. Davon ist jedoch abzuraten, weil so ein noch größerer Informationsverlust entsteht und die strukturellen Darstellungen noch ungenauer werden könnten.

5.4 Zeit für Visualisierung

Insgesamt konnte in jedem Bereich der Szenarien eine Reduzierung erreicht werden. Bei dem ersten Szenario fiel der Unterschied noch sehr deutlich aus, während bei dem zweiten Szenario der Unterschied zwar messbar, für den Menschen jedoch schwer wahrnehmbar war.

Im ersten Szenario hingegen ist die Entscheidung nicht so einfach. Für den Kontur-Plot ist die eingesparte Zeit sehr hoch und die Qualität wird mit der Mischtemperatur gut erhalten. Ebenso bei dem Vektor-Plot. Bei den 2D Plots wie dem Scatter und dem Histogramm hingegen sollten die Originaldaten verwendet werden, die eingesparte Zeit ist bei diesen zu gering und der visuelle Informationsverlust in keinem Verhältnis.

Im zweiten Szenario ist der Zeitgewinn zu gering, um eine geringere Auflösung zu akzeptieren. Auch ist die Zeit für eine Interaktion wie einer Drehung oder ein Zoom viel zu hoch, um für eine interaktive Visualisierung genutzt werden zu können.

5.5 Resultat der Arbeit

Die Beschäftigung um eine Reduzierung der Daten kann sich in einigen Fällen lohnen, ist jedoch keine Universallösung für die Problematik. Arbeitet man mit den reduzierten Daten, sollten die Erkenntnisse immer noch mit den Originaldaten abgeglichen werden. Für eine interaktive Exploration der Daten in der Hinsicht auf

ihre Struktur lohnt sich der Aufwand.

Mit Hilfe der hier gezeigten Verfahren konnte die Latenz reduziert werden und für die verwendeten Beispiele eine Exploration auf normalen Systemen erfolgen.

5.6 VisIt im Allgemeinen

VisIt war das einzige frei verfügbare Tool, was eine interaktive Visualisierung von FLASH Daten ermöglichte. Leider war die Benutzung mit Schwierigkeiten durchzogen. Die fragmentierte Dokumentation in Form eines veralteten Handbuchs für Version 1.5¹, eines schlecht strukturierten Wikis² und eines Forums³, für dieses man in spezialisierten Bereichen eine Registrierung(die immer noch nicht innerhalb von zwei Wochen freigeschaltet wurde) benötigte, erschwerten die Bedienung unnötig. Zusätzlich traten Bugs bei der Nutzung auf, der Vektor-Plot ist bei der Verwendung unbrauchbar:

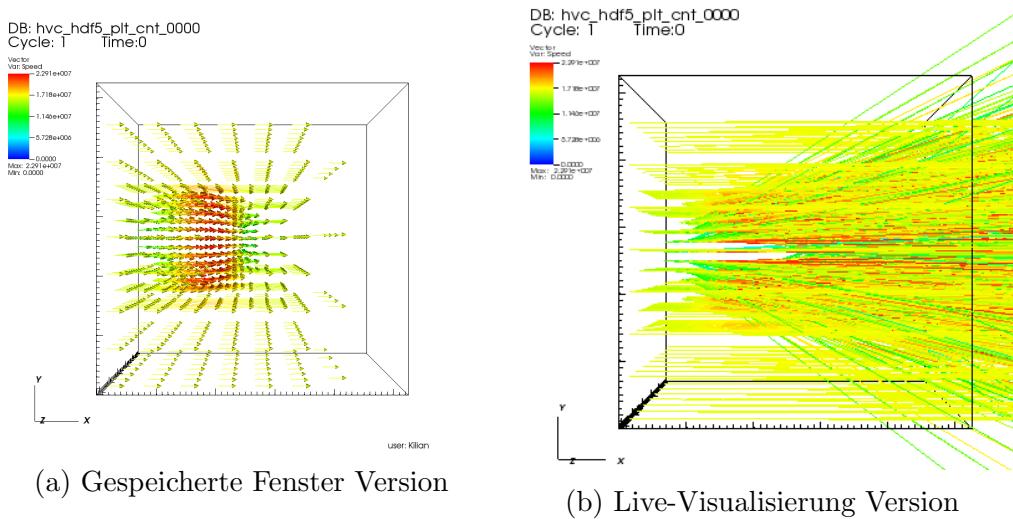


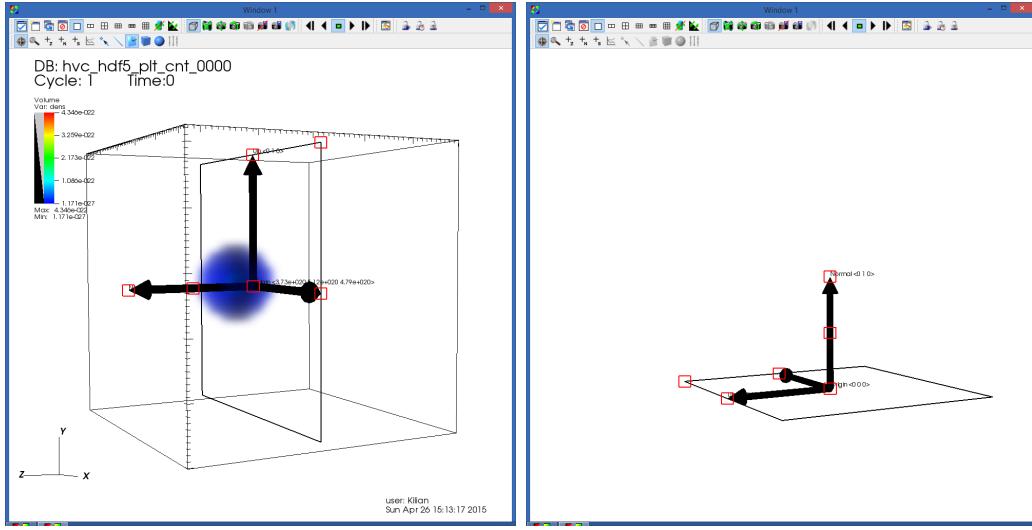
Abbildung 5.1: Vektor Plot - Fehler bei Darstellung

Dieser Anzeigefehler trat jedoch nur auf, wenn der Vektor-Plot in einem Fenster angezeigt wurde und nicht, wenn dieses als Bild abgespeichert werden soll. Ebenso ist die Bedienung zum Setzen einer Schnittebene fehlerhaft. Dabei existiert die durchaus intuitive Platzierung der Ebene im Raum, während der Plot sichtbar ist. Möchte man nun einen Slice-Plot an dieser Stelle erzeugen, muss man den Operator **Slice** hinzufügen. Dabei wird jedoch nicht nur die Ebene auf den Koordinatenursprung zurückgesetzt und die vorherige Interaktion rückgängig gemacht, es wird zusätzlich auch noch der Volumen-Plot zur Orientierung ausgeblendet. Der Vorteil einer interaktiven Visualisierung geht so verloren.

¹URL:<https://wci.llnl.gov/content/assets/docs/simulation/computer-codes/visit/>
VisItUsersManual1.5.pdf

²URL: <http://www.visitusers.org/>

³URL: <http://www.visitusers.org/forum/forum.pl>



(a) Vor Auswahl Slice-Operator

(b) Nach Auswahl Sclice-Operator

Abbildung 5.2: Problematische Bedienung bei der Erstellung einer Schnittebene

Nichts desto trotz erleichtert die interaktive Erstellung von Visualisierung die Exploration der Daten deutlich und war für die Arbeit eine große Hilfe.

6. Zukünftige Arbeiten

Da diese Arbeit unmöglich in dem gesetzten Zeitraum alle möglichen Fragen dieses umfassenden Themengebietes abdecken kann, werden in diesem Kapitel Vorschläge für zukünftige Arbeiten und noch offene Fragen gesammelt.

6.1 Tools

Das Tool `VisIt` wurde in der Version 2.9.0 unter Windows 8.1 64-Bit verwendet. Dabei traten einerseits Fehler in der Darstellung ([Abbildung 5.1](#)) und in der Bedienung ([Abbildung 5.2](#)) auf. Es wäre noch zu untersuchen, ob die Zeiten auf andere Betriebssystemen wie Linux-Distributionen oder OS sich verändern. Für den PreProcessing-Schritt mit `NeonGenesisTool` muss jedoch dann die jeweilige `HDF5` API verwendet werden, da diese vom Betriebssystem abhängig ist.

Ebenso kann mit Sicherheit die Zeit für den PreProcessing-Schritt gesenkt werden. Auch wenn das Tool nach besten Wissen und Gewissen optimiert wurde, so ist es doch stark beschränkt. Eine Reduzierung erfolgt immer auf Basis von einer Halbierung der Zellenanzahl und funktioniert nur für `FLASH 3.2`. Auch kann untersucht werden, ob ein Gewinn der Zeit durch eine Nutzung von mehreren Festplatten verbessert werden kann. Das Programm kann selber mehrere Dateien auf einmal reduzieren, wird jedoch durch die Lese- und Schreibgeschwindigkeit der Festplatten eingeschränkt. Zusätzlich erlaubt die `Java-Virtual-Machine (JVM)` eine große Anzahl an Konfigurationen, die vielleicht zu einer kleineren Zeitspanne führen kann.

Auch ist der Zugriff auf die Dateien via dem `HDF5-API` für Java nicht besonders performant. Das Lesen einzelner Bereiche in einer Matrix aus einem Datensatz kann bestimmt werden, so dass die Blöcke einzeln gelesen und reduziert werden können. Das Schreiben eines Datensatzes geht jedoch nur als ganze Matrix und nicht als einzelne Blöcke, die Stück für Stück in die Matrix eingefügt werden. Dies führt zu einem unnötigen Engpass und verhindert eine höhere Parallelisierung. Durch die Entwicklung einer reinen Java-API, die auch unabhängig von den Betriebssystem ist, könnten diese Probleme beseitigt werden. Ob sich dieser Aufwand lohnt, ist eine andere Frage.

Alternativ kann auch dieser Prozess der Reduzierung direkt in die jeweiligen Tools

zur Visualisierung eingebaut werden. So wird dem Nutzer die Bedienung erleichtert. Auch die Untersuchung anderer Tools und die Überprüfung der Geschwindigkeit anhand der hier aufgestellten Szenarien ist ein Punkt, der in dieser Arbeit nicht weiter verfolgt wurde. So ist die Nutzung von VisIt nicht optimal und kann verbessert werden.

6.2 Verfahren

Die in Kapitel 2 dargestellten Verfahren zur Verbesserung der Qualität bei Volumen-Plots wurde hier wegen Erhaltung des Datenformates nicht weiter betrachtet. Um diese Verfahren, die zusätzliche Datenstrukturen benötigen, umzusetzen, müssen dazu die FLASH-Interpreter in den Tools und gegebenenfalls auch der Renderer angepasst werden.

Da die Mischtemperatur Vorteile in der Bildgüte ergeben kann, sollte ebenso nach anderen physikalischen Gegebenheiten gesucht werden, die angewandt werden können. Dabei ist zu beachten, dass dies Speziallösungen sind und nicht auf jeden Fall zutreffen.

6.3 Daten

Der Datenaufbau der FLASH Dateien ist nicht optimal. Wie bereits erwähnt, werden die Blöcke in einem Octree abgespeichert. Dabei haben jedoch alle Blöcke, ungenachtert ihrer Position in der Hierarchie, eine fixe Anzahl von Zellen. Selbst die Wurzel des Baumes, die den kompletten Simulationsraum abbildet, besitzt Werte und Zellen. Diese werden jedoch, da es feiner aufgelöste Daten gibt, nicht im Visualisierungsprozess verwendet. Diese Blöcke werden als unsicher markiert und dennoch abgespeichert. Diese unnötige Speicherung bedeutete größere Dateien und längere Ladezeiten und somit im schlimmsten Fall auch höhere Latenzen. Ursache dafür ist die mangelnde Fähigkeit von HDF5 Matrizen mit variablen Dimensionen abzuspeichern.

A. Anhang

A.1 NeonGenesisTool Kommandozeilenreferenz

- id, –inputDirectory** (Erforderlich) Pfad zum Ordner mit den Originaldateien.
Der Pfad muss existieren und darf nicht leer sein. Es werden nur Dateien und keine Ordner betrachtet.
- od, –outputDirectory** (Erforderlich) Pfad zum Ordner, in dem die reduzierten Dateien gespeichert werden sollen. Der Order wird angelegt, sofern er nicht existiert.
- ds, –datasets** Auflistung der Variablen, die reduziert werden sollen. Angabe als Liste mit Leerzeichen getrennt. Sind keine Variablennamen angegeben, werden alle reduziert.
- rt, –reduceType** Auswahl, welches Verfahren beim reduzieren verwendet werden soll. Mögliche Werte sind *mean*, *median*, *physicalMean*, *physicalMedian*. Standardwert ist *mean*.
- v, –verbose** Zeigt extra Information beim Reduzierungsprozess an. Standardwert ist *false*.
- h, –help** Zeigt die Hilfe an.

A.1.1 Anwendungsbeispiele

Quelltext A.1: Anwendungsbeispiel 1

```
java -jar NeonGenesisTool.jar -id plotfiles_original -od \
plotfiles_downsampled -v
```

Downsampling aller Datensätze im Ordner *plotfiles_original*. Ergebnisse werden in *plotfiles_downsampled* gespeichert. Dabei werden Zeiten und Größen der Dateien auf der Konsole ausgegeben. Es verwendet für das Downsampling den Mittelwert.

Quelltext A.2: Anwendungsbeispiel 2

```
java -jar NeonGenesisTool.jar -id plotfiles_original -od \
plotfiles_downsampled_median --reduceType median
```

Downsampling aller Datensätze im Ordner *plotfiles_original*. Ergebnisse werden in *plotfiles_downsampled_median* gespeichert. Es verwendet für das Downsampling den Median statt den Mittelwert.

Quelltext A.3: Anwendungsbeispiel 3

```
java -jar NeonGenesisTool.jar -id plotfiles_original -od \
plotfiles_downsampled_physicalMean -ds temp pres -\
reduceType physicalMean
```

Downsampling der Datensätze *temp* (Temperatur) und *pres* (Druck) im Ordner *plotfiles_original*. Ergebnisse werden in *plotfiles_downsampled_physicalMean* gespeichert. Es verwendet für alle möglichen Datensätze physikalische Eigenschaften zum Reduzieren, z.B. bei der Temperatur die Mischtemperatur ([Abschnitt 4.4.2.1](#)).

A.2 NeonGenesisTool Struktur

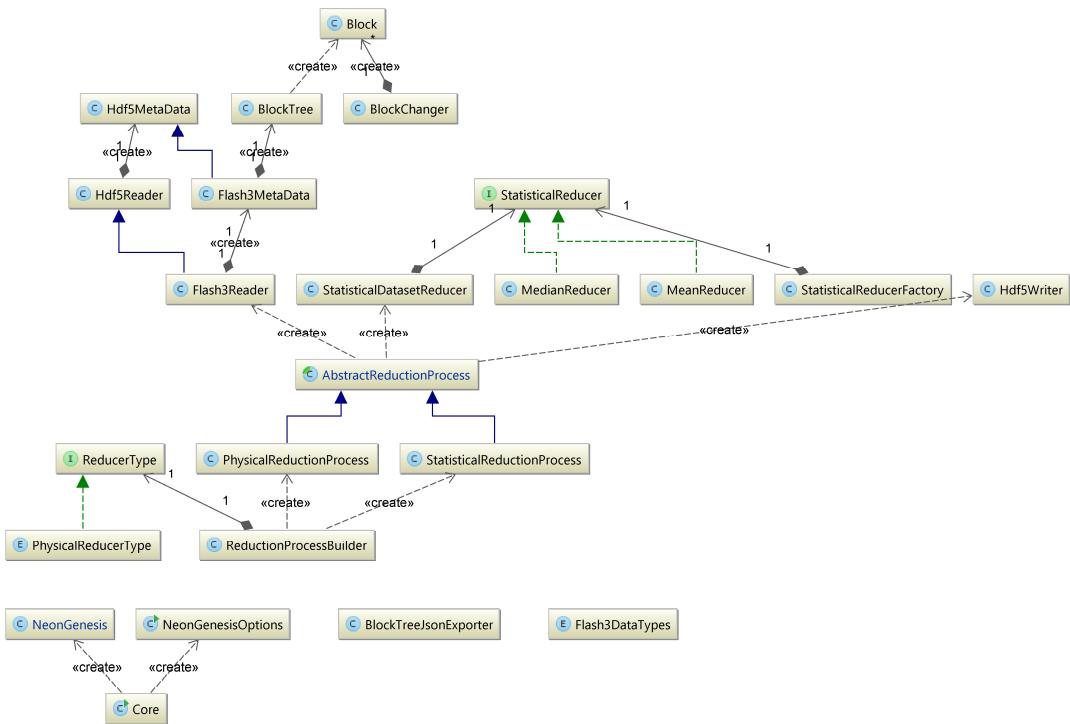


Abbildung A.1: UML Diagramm von NeonGenesisTool

Eine kurze Erläuterung der Klasse des Tools, um ein Grundverständnis für den Aufbau des Programmes zu erhalten. Für genauere Erklärungen und Methodenbeschreibungen ist der beigelegte Quellcode zu sichten, alle für diese Arbeit relevanten Methoden wurden im Hauptteil besprochen.

Core Einstiegspunkt des Programmes. Liest die Kommandozeilenargumente aus und ruft **NeonGenesis** auf.

NeonGenesis Erstellt aus den Startparametern den Downsampleprozess und startet ihn.

NeonGesesisOptions Datenklasse, welche die möglichen Startparameter des Programmes beinhaltet und mit Hilfe der Bibliothek **JCommander**¹ die Kommandozeilenargumente auf die Attribute abbildet.

Block Bildet die Metadaten eines Blockes im adaptiven Gitter ab. Beinhaltet u.a. die Größe, die Position und die Nachbarn. Die Daten der Zellen werden dort nicht gespeichert, sondern nach Bedarf aus der HDF5 Datei ausgelesen.

BlockTree Bildet den Octree des adaptiven Gitter aus und stellt Methoden zum Zugriff auf einzelne **Blöcke** via ID bereit. Wird pro Datei einmal aufgebaut und wiederverwertet.

BlockTreeJsonExporter Hilfsklasse, um einen **BlockTree** als JSON-String zu exportieren. Dieser wurde verwendet, um ihn mit Hilfe von D3 zu visualisieren und die Korrektheit des Baumes einfacher zu überprüfen.

BlockChanger Hilfsklasse um **Blöcke** zu verändern.

Hdf5Reader Klasse, die mit Hilfe des HDF5-API eine HDF5-Datei ausliest. Pro Datei existiert ein Reader.

Hdf5MetaData Basisklasse zum Zugriff auf einzelne Datensätze in einer HDF5 Datei. Wird von einem **Hdf5Reader** einmal pro Datei erstellt.

Flash3MetaReader Spezialisierung von **Hdf5Reader** um FLASH3 Dateien auf Basis von HDF5 Formaten einfacher auszulesen. Ermöglicht den Zugriff auf Daten eines Blockes via ID.

Flash3MetaData Spezialisierung von **Hdf5MetaData** um FLASH3 spezifische Datensätze aus der HDF5 Datei abzubilden. Erstellt u.a. den **BlockTree**.

Hdf5Writer Klasse, um eine neue HDF5 Datei anzulegen und mit Datensätzen zu beschreiben. Ermöglicht auch das kopieren ganzer Datensätze aus der Originaldatei. Wird z.B. bei der Reduzierung angewendet, um Simulationsparameter wie *integer_runtime_scalar* zu kopieren, da diese nicht verändert werden. Problem hier: Datensätze werden als ganze Matrix auf einmal geschrieben, es gibt kein Anhängen einzelner Daten an einen Datensatz. Hier liegt ein Performanceproblem vor, da so eine 25000x4x4x4 32-bit Float Matrix auf einmal geschrieben werden muss, statt diese in Stück für Stück zu schreiben.

Flash3DataType Da HDF5-Dateien variable Datentypendefinitionen haben können, sind in diesem **enum** die für Flash3 gängigen Datentypen wie Gleitkommazahlen und ganze Zahlen als auch Zeichenketten definiert.

¹Website: <http://jcommander.org/>

AbstractReductionProzess Abstrakte Basis Klasse, die den Downsampleprozess abbildet und Methoden zur Reduktion einzelner Datensätze anbietet.

PhysicalReductionProzess Spezialisierung von **AbstractReductionProzess**, welche physikalische Verfahren zur Reduktion von Datensätzen verwendet. Ist für einen Datensatztyp kein Verfahren definiert, greift es auf statistische Verfahren zur Reduktion zurück.

StatisticalReductionProzess Spezialisierung von **AbstractReductionProzess**, welche nur statistische Verfahren zur Reduktion von Datensätzen anwendet.

ReductionProzessBuilder Stellt ein Builder-Pattern für einen **AbstractReductionProzess** bereit, um Parameter wie zu reduzierende Datensätze oder Verfahren auszuwählen.

ReducerType Interface um ein Verfahren zur Reduktion darzustellen. Wird von **ReductionProzessBuilder** verwendet, um die jeweilige Klasse für ein Verfahren zu instanzieren.

PhysicalReducerType Enum, welches das Interface **ReducerType** implementiert und signalisiert, dass der Downsampleprozess ein **PhysicalReductionProzess** verwenden soll.

StatisticalReducer Analog zu **PhysicalReducerType**, nur dass es rein statistische Verfahren verwenden soll. Gibt an, welche Methode ausgeführt wird.

StatisticalReducerFactory Erzeugt einer Konstanten das jeweilige **StatisticalReducer**.

MedianReducer Implementierung des Median.

Mean Implementierung des arithmetischen Durchschnittes.

StatisticalDatasetReducer Klasse, die einen einzelnen Datensatz mit Hilfe von statistischen Verfahren reduziert.

A.3 VisIt Python Scripts

Aufgelistet sind die Python Scripts, die für die Messung der Zeiten in VisIt verwendet worden sind. Pfade zu den Dateien wurden verallgemeinert und sind bei Nutzung der Scripts anzupassen. Zur Ausführung wird empfohlen, nur ein einziges **LaunchProfile** in VisIt zu besitzen. Existieren mehrere, wird der Nutzer bei der Ausführung des Scriptes bei jedem Start von VisIt nach dem Nutzerprofil gefragt, was mühselig ist. Um dies zu erreichen, half es in dem VisIt Unterordner *hosts* die .xml Datei anzupassen nur ein Object mit dem Namen **LaunchProfile** zu besitzen. VisIt 2.9.0 speicherte das Entfernen des Profiles in der GUI nicht ab.

Quelltext A.4: Python Script zum Starten von VisIt mit Scripts

```

1 import subprocess
3 visitPath = "/path/to/visit.exe"
scripts = ['FourViews.py', 'DetailView.py']
plotTypes = ['original', 'mean', 'median', 'physicalMean', 'physicalMedian']
6 for script in scripts:
    for plotType in plotTypes:
        for timeSlot in range(0, 101, 25):
9            for run in range(0, 5):
                subprocess.call([visitPath, "-cli", "-s", script, str(timeSlot), plotType])

```

Quelltext A.5: Visit Script - 4 Plots in VisIt

```

1 from visit import *
3 databaseLocation = "\\\path\\\to\\\plotfilesDir"
try:
    Launch()
    OpenComputeEngine('localhost')
except:
    pass
9 databaseString = "localhost:" + databaseLocation + "\\\hvc_hdf5_plt_cnt_* database"
OpenDatabase(databaseString)
SetWindowLayout(4)
12
# Contour Plot
SetActiveWindow(1)
15 AddPlot("Contour", "temp", 1, 1)
ContourAtts = ContourAttributes()
ContourAtts.defaultPalette.smoothing = ContourAtts.defaultPalette.None # None, Linear,
    CubicSpline
18 ContourAtts.defaultPalette.equalSpacingFlag = 1
ContourAtts.defaultPalette.discreteFlag = 1
ContourAtts.defaultPalette.categoryName = "Standard"
21 ContourAtts.changedColors = ()
ContourAtts.colorType = ContourAtts.ColorBySingleColor # ColorBySingleColor,
    ColorByMultipleColors, ColorByColorTable
ContourAtts.colorTableName = "Default"
24 ContourAtts.invertColorTable = 0
ContourAtts.legendFlag = 1
ContourAtts.lineStyle = ContourAtts.SOLID # SOLID, DASH, DOT, DOTDASH
27 ContourAtts.lineWidth = 0
ContourAtts.singleColor = (255, 0, 0, 255)
ContourAtts.contourNLevels = 10
30 ContourAtts.contourValue = ()
ContourAtts.contourPercent = ()
ContourAtts.contourMethod = ContourAtts.Level # Level, Value, Percent
33 ContourAtts.minFlag = 0
ContourAtts.maxFlag = 0
ContourAtts.scaling = ContourAtts.Log # Linear, Log
36 ContourAtts.wireframe = 0
SetPlotOptions(ContourAtts)
DrawPlots()
39 # Zoom Contour Plot

```

```

View3DAtts = View3DAttributes()
View3DAtts.viewNormal = (0, 0, 1)
42 View3DAtts.focus = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.viewUp = (0, 1, 0)
View3DAtts.viewAngle = 30
45 View3DAtts.parallelScale = 9.08574e+020
View3DAtts.nearPlane = -1.81715e+021
View3DAtts.farPlane = 1.81715e+021
48 View3DAtts.imagePan = (0, 0)
View3DAtts.imageZoom = 1.4641
View3DAtts.perspective = 1
51 View3DAtts.eyeAngle = 2
View3DAtts.centerOfRotationSet = 0
View3DAtts.centerOfRotation = (5.24565e+020, 5.24565e+020, 5.24565e+020)
54 View3DAtts.axis3DScaleFlag = 0
View3DAtts.axis3DScales = (1, 1, 1)
View3DAtts.shear = (0, 0, 1)
57 View3DAtts.windowValid = 1
SetView3D(View3DAtts)
DrawPlots()
60 # Rotate Contour Plot
View3DAtts = View3DAttributes()
View3DAtts.viewNormal = (0.438119, 0.0702442, 0.896168)
63 View3DAtts.focus = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.viewUp = (-0.056096, 0.997136, -0.0507341)
View3DAtts.viewAngle = 30
66 View3DAtts.parallelScale = 9.08574e+020
View3DAtts.nearPlane = -1.81715e+021
View3DAtts.farPlane = 1.81715e+021
69 View3DAtts.imagePan = (0, 0)
View3DAtts.imageZoom = 1
View3DAtts.perspective = 1
72 View3DAtts.eyeAngle = 2
View3DAtts.centerOfRotationSet = 0
View3DAtts.centerOfRotation = (5.24565e+020, 5.24565e+020, 5.24565e+020)
75 View3DAtts.axis3DScaleFlag = 0
View3DAtts.axis3DScales = (1, 1, 1)
View3DAtts.shear = (0, 0, 1)
78 View3DAtts.windowValid = 1
SetView3D(View3DAtts)
DrawPlots()
81 # Histogram
SetActiveWindow(2)
84 AddPlot("Histogram", "ener", 1, 1)
HistogramAtts = HistogramAttributes()
HistogramAtts.basedOn = HistogramAtts.ManyZonesForSingleVar # ManyVarsForSingleZone,
ManyZonesForSingleVar
87 HistogramAtts.histogramType = HistogramAtts.Frequency # Frequency, Weighted, Variable
HistogramAtts.weightVariable = "default"
HistogramAtts.limitsMode = HistogramAtts.CurrentPlot # OriginalData, CurrentPlot
90 HistogramAtts.minFlag = 0
HistogramAtts.maxFlag = 0
HistogramAtts.numBins = 32
93 HistogramAtts.domain = 0
HistogramAtts.zone = 0
HistogramAtts.useBinWidths = 1
96 HistogramAtts.outputType = HistogramAtts.Block # Curve, Block
HistogramAtts.lineStyle = HistogramAtts.SOLID # SOLID, DASH, DOT, DOTDASH
HistogramAtts.lineWidth = 0
99 HistogramAtts.color = (200, 80, 40, 255)
HistogramAtts.dataScale = HistogramAtts.Linear # Linear, Log, SquareRoot
HistogramAtts.binScale = HistogramAtts.Linear # Linear, Log, SquareRoot
102 HistogramAtts.normalizeHistogram = 0
HistogramAtts.computeAsCDF = 0
SetPlotOptions(HistogramAtts)
105 DrawPlots()

# Histogram zoom
108 View2DAtts = View2DAttributes()
View2DAtts.windowCoords = (0.0565941, 0.943406, 0.0867769, 0.913223)
View2DAtts.viewportCoords = (0.2, 0.95, 0.15, 0.95)

```

```
111 View2DAtts.fullFrameActivationMode = View2DAtts.Auto # On, Off, Auto
View2DAtts.fullFrameAutoThreshold = 100
View2DAtts.xScale = View2DAtts.LINEAR # LINEAR, LOG
114 View2DAtts.yScale = View2DAtts.LINEAR # LINEAR, LOG
View2DAtts.windowValid = 1
SetView2D(View2DAtts)
117 DrawPlots()

# Histogram Panning
120 View2DAtts = View2DAttributes()
View2DAtts.windowCoords = (0.0331325, 1.03313, 0.0873494, 1.08735)
View2DAtts.viewportCoords = (0.2, 0.95, 0.15, 0.95)
123 View2DAtts.fullFrameActivationMode = View2DAtts.Auto # On, Off, Auto
View2DAtts.fullFrameAutoThreshold = 100
View2DAtts.xScale = View2DAtts.LINEAR # LINEAR, LOG
126 View2DAtts.yScale = View2DAtts.LINEAR # LINEAR, LOG
View2DAtts.windowValid = 1
SetView2D(View2DAtts)
129 DrawPlots()

# Vector Plot
132 SetActiveWindow(3)
DefineVectorExpression("Speed", "{velx, vely, velz}")
AddPlot("Vector", "Speed", 1, 1)
135 DrawPlots()

# Zoom Vector Plot
138 View3DAtts = View3DAttributes()
View3DAtts.viewNormal = (0, 0, 1)
View3DAtts.focus = (5.24565e+020, 5.24565e+020, 5.24565e+020)
141 View3DAtts.viewUp = (0, 1, 0)
View3DAtts.viewAngle = 30
View3DAtts.parallelScale = 9.08574e+020
144 View3DAtts.nearPlane = -1.81715e+021
View3DAtts.farPlane = 1.81715e+021
View3DAtts.imagePan = (0, 0)
147 View3DAtts.imageZoom = 1.4641
View3DAtts.perspective = 1
View3DAtts.eyeAngle = 2
150 View3DAtts.centerOfRotationSet = 0
View3DAtts.centerOfRotation = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.axis3DScaleFlag = 0
153 View3DAtts.axis3DScales = (1, 1, 1)
View3DAtts.shear = (0, 0, 1)
View3DAtts.windowValid = 1
156 SetView3D(View3DAtts)
DrawPlots()

159 # Rotate Vector Plot
View3DAtts = View3DAttributes()
View3DAtts.viewNormal = (0.438119, 0.0702442, 0.896168)
162 View3DAtts.focus = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.viewUp = (-0.056096, 0.997136, -0.0507341)
View3DAtts.viewAngle = 30
165 View3DAtts.parallelScale = 9.08574e+020
View3DAtts.nearPlane = -1.81715e+021
View3DAtts.farPlane = 1.81715e+021
168 View3DAtts.imagePan = (0, 0)
View3DAtts.imageZoom = 1
View3DAtts.perspective = 1
171 View3DAtts.eyeAngle = 2
View3DAtts.centerOfRotationSet = 0
View3DAtts.centerOfRotation = (5.24565e+020, 5.24565e+020, 5.24565e+020)
174 View3DAtts.axis3DScaleFlag = 0
View3DAtts.axis3DScales = (1, 1, 1)
View3DAtts.shear = (0, 0, 1)
177 View3DAtts.windowValid = 1
SetView3D(View3DAtts)
DrawPlots()

180 # Scatter Plot
SetActiveWindow(4)
```

```

183 AddPlot("Scatter", "dens", 1, 1)
ScatterAtts = ScatterAttributes()
ScatterAtts.var1 = "dens"
186 ScatterAtts.var1Role = ScatterAtts.Coordinate0
ScatterAtts.var1MinFlag = 0
ScatterAtts.var1MaxFlag = 0
189 ScatterAtts.var1Min = 0
ScatterAtts.var1Max = 1
ScatterAtts.var1Scaling = ScatterAtts.Log
192 ScatterAtts.var1SkewFactor = 1
ScatterAtts.var2Role = ScatterAtts.Coordinate1
ScatterAtts.var2 = "temp"
195 ScatterAtts.var2MinFlag = 0
ScatterAtts.var2MaxFlag = 0
ScatterAtts.var2Min = 0
198 ScatterAtts.var2Max = 1
ScatterAtts.var2Scaling = ScatterAtts.Log
ScatterAtts.var2SkewFactor = 1
201 ScatterAtts.var3Role = ScatterAtts.None
ScatterAtts.var3 = "default"
ScatterAtts.var3MinFlag = 0
204 ScatterAtts.var3MaxFlag = 0
ScatterAtts.var3Min = 0
ScatterAtts.var3Max = 1
207 ScatterAtts.var3Scaling = ScatterAtts.Linear
ScatterAtts.var3SkewFactor = 1
ScatterAtts.var4Role = ScatterAtts.None
210 ScatterAtts.var4 = "default"
ScatterAtts.var4MinFlag = 0
ScatterAtts.var4MaxFlag = 0
213 ScatterAtts.var4Min = 0
ScatterAtts.var4Max = 1
ScatterAtts.var4Scaling = ScatterAtts.Linear
216 ScatterAtts.var4SkewFactor = 1
ScatterAtts.pointSize = 0.05
ScatterAtts.pointSizePixels = 1
219 ScatterAtts.pointType = ScatterAtts.Point
ScatterAtts.scaleCube = 1
ScatterAtts.colorType = ScatterAtts.ColorByForegroundColor
222 ScatterAtts.singleColor = (255, 0, 0, 255)
ScatterAtts.colorTableName = "Default"
ScatterAtts.invertColorTable = 0
225 ScatterAtts.legendFlag = 1
SetPlotOptions(ScatterAtts)
DrawPlots()
228
# Zoom Scatter Plot
View2DAtts = View2DAttributes()
231 View2DAtts.windowCoords = (0.0565941, 0.943406, 0.0867769, 0.913223)
View2DAtts.viewportCoords = (0.2, 0.95, 0.15, 0.95)
View2DAtts.fullFrameActivationMode = View2DAtts.Auto # On, Off, Auto
234 View2DAtts.fullFrameAutoThreshold = 100
View2DAtts.xScale = View2DAtts.LINEAR # LINEAR, LOG
View2DAtts.yScale = View2DAtts.LINEAR # LINEAR, LOG
237 View2DAtts.windowValid = 1
SetView2D(View2DAtts)
DrawPlots()
240
# Panning Scatter Plot
View2DAtts = View2DAttributes()
243 View2DAtts.windowCoords = (0.0331325, 1.03313, 0.0873494, 1.08735)
View2DAtts.viewportCoords = (0.2, 0.95, 0.15, 0.95)
View2DAtts.fullFrameActivationMode = View2DAtts.Auto # On, Off, Auto
246 View2DAtts.fullFrameAutoThreshold = 100
View2DAtts.xScale = View2DAtts.LINEAR # LINEAR, LOG
View2DAtts.yScale = View2DAtts.LINEAR # LINEAR, LOG
249 View2DAtts.windowValid = 1
SetView2D(View2DAtts)
DrawPlots()

```

Quelltext A.6: VisIt Script - Volume Renderung mit Raycasting

```

1 from visit import *
3 databaseLocation = "\\\path\\to\\\\plotfilesDir"
try:
    Launch()
    OpenComputeEngine('localhost')
except:
    pass
9 databaseString = "localhost:" + databaseLocation + "\\hvc_hdf5_plt_cnt_* database"
OpenDatabase(databaseString)

12 # Volumne Plot
AddPlot("Volume", "dens", 1, 1)
VolumeAtts = VolumeAttributes()
15 VolumeAtts.legendFlag = 1
VolumeAtts.lightingFlag = 0
VolumeAtts.colorControlPoints.GetControlPoints(0).colors = (0, 0, 255, 255)
18 VolumeAtts.colorControlPoints.GetControlPoints(0).position = 0
VolumeAtts.colorControlPoints.GetControlPoints(1).colors = (0, 255, 255, 255)
VolumeAtts.colorControlPoints.GetControlPoints(1).position = 0.25
21 VolumeAtts.colorControlPoints.GetControlPoints(2).colors = (0, 255, 0, 255)
VolumeAtts.colorControlPoints.GetControlPoints(2).position = 0.5
VolumeAtts.colorControlPoints.GetControlPoints(3).colors = (255, 255, 0, 255)
24 VolumeAtts.colorControlPoints.GetControlPoints(3).position = 0.75
VolumeAtts.colorControlPoints.GetControlPoints(4).colors = (255, 0, 0, 255)
VolumeAtts.colorControlPoints.GetControlPoints(4).position = 1
27 VolumeAtts.colorControlPoints.smoothing = VolumeAtts.colorControlPoints.Linear
VolumeAtts.colorControlPoints.equalSpacingFlag = 0
VolumeAtts.colorControlPoints.discreteFlag = 0
30 VolumeAtts.colorControlPoints.categoryName = ""
VolumeAtts.opacityAttenuation = 1
VolumeAtts.opacityMode = VolumeAtts.FreeformMode
33 VolumeAtts.resampleFlag = 1
VolumeAtts.resampleTarget = 50000
VolumeAtts.opacityVariable = "default"
36 VolumeAtts.compactVariable = "default"
VolumeAtts.freeformOpacity = tuple(range(0, 256))
VolumeAtts.useColorVarMin = 0
39 VolumeAtts.colorVarMin = 0
VolumeAtts.useColorVarMax = 0
VolumeAtts.colorVarMax = 0
42 VolumeAtts.useOpacityVarMin = 0
VolumeAtts.opacityVarMin = 0
VolumeAtts.useOpacityVarMax = 0
45 VolumeAtts.opacityVarMax = 0
VolumeAtts.smoothData = 0
VolumeAtts.samplesPerRay = 500
48 VolumeAtts.rendererType = VolumeAtts.RayCasting
VolumeAtts.gradientType = VolumeAtts.SobelOperator
VolumeAtts.num3DSlices = 200
51 VolumeAtts.scaling = VolumeAtts.Linear
VolumeAtts.skewFactor = 1
VolumeAtts.limitsMode = VolumeAtts.OriginalData
54 VolumeAtts.sampling = VolumeAtts.Rasterization
VolumeAtts.rendererSamples = 3
VolumeAtts.transferFunctionDim = 1
57 VolumeAtts.lowGradientLightingReduction = VolumeAtts.Lower
VolumeAtts.lowGradientLightingClampFlag = 0
VolumeAtts.lowGradientLightingClampValue = 1
60 VolumeAtts.materialProperties = (0.4, 0.75, 0, 15)
SetPlotOptions(VolumeAtts)
DrawPlots()

63 # Rotate Volumne Plot
View3DAtts = View3DAttributes()
66 View3DAtts.viewNormal = (0.520274, 0.118673, 0.845714)
View3DAtts.focus = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.viewUp = (-0.0480846, 0.992798, -0.109731)
69 View3DAtts.viewAngle = 30
View3DAtts.parallelScale = 9.08574e+020

```

```

72 View3DAtts.nearPlane = -1.81715e+021
View3DAtts.farPlane = 1.81715e+021
View3DAtts.imagePan = (0, 0)
View3DAtts.imageZoom = 1
75 View3DAtts.perspective = 1
View3DAtts.eyeAngle = 2
View3DAtts.centerOfRotationSet = 0
78 View3DAtts.centerOfRotation = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.axis3DScaleFlag = 0
View3DAtts.axis3DScales = (1, 1, 1)
81 View3DAtts.shear = (0, 0, 1)
View3DAtts.windowValid = 1
SetView3D(View3DAtts)
84
# Zoom
View3DAtts = View3DAttributes()
87 View3DAtts.viewNormal = (0, 0, 1)
View3DAtts.focus = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.viewUp = (0, 1, 0)
90 View3DAtts.viewAngle = 30
View3DAtts.parallelScale = 9.08574e+020
View3DAtts.nearPlane = -1.81715e+021
93 View3DAtts.farPlane = 1.81715e+021
View3DAtts.imagePan = (0, 0)
View3DAtts.imageZoom = 2.14359
96 View3DAtts.perspective = 1
View3DAtts.eyeAngle = 2
View3DAtts.centerOfRotationSet = 0
99 View3DAtts.centerOfRotation = (5.24565e+020, 5.24565e+020, 5.24565e+020)
View3DAtts.axis3DScaleFlag = 0
View3DAtts.axis3DScales = (1, 1, 1)
102 View3DAtts.shear = (0, 0, 1)
View3DAtts.windowValid = 1
SetView3D(View3DAtts)

```

A.4 yt-project Script

Quelltext A.7: yt-project Script - Erstellung Slice Plots

```

1 import yt
2 attributes = ['dens', 'temp', 'gpot']
3 plotfile = '/path/to/plotfile'

4 ds = yt.load(plotfile)
5 # Create slices of density, temperature and gravity potential
6 # in all three axes and save the plots as PNG images
7 yt.SlicePlot(ds, 'x', attributes).save('slicePlotX')
8 yt.SlicePlot(ds, 'y', attributes).save('slicePlotY')
9 yt.SlicePlot(ds, 'z', attributes).save('slicePlotZ')

```

Literaturverzeichnis

- [Aya15] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Incorporated, paraview 4.3 edition, 1 2015. (zitiert auf Seite 13)
- [BRGH05] Michelle A. Borkin, Naomi A. Ridge, Alyssa A. Goodman, and Michael Halle. Demonstration of the applicability of 3D slicer to astronomical data using CO-13 and C18O observations of IC 348. 2005. (zitiert auf Seite 5)
- [CBW⁺12] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012. (zitiert auf Seite 14)
- [CXZ11] Roger Crawfis, Daqing Xue, and Caixia Zhang. Desktop delivery: Access to large datasets. In Charles D. Hansen and Chris R. Johnson, editors, *Visualization Handbook*. Academic Press, Amsterdam, Boston, 2011. (zitiert auf Seite 18)
- [DAG⁺09] Anshu Dubey, Katie Antypas, Murali K. Ganapathy, Lynn B. Reid, Katherine Riley, Dan Sheeler, Andrew Siegel, and Klaus Weide. Extensible component-based architecture for flash, a massively parallel, multiphysics simulation code. *Parallel Computing*, 35(10–11):512 – 522, 2009. (zitiert auf Seite 8)
- [Dal94] G. Brent Dalrymple. *The Age of the Earth* -. Stanford University Press, Stanford, 1994. (zitiert auf Seite 12)
- [Dal01] G. B. Dalrymple. The age of the Earth in the twentieth century: a problem (mostly) solved. *Geological Society of London Special Publications*, 190:205–221, January 2001. (zitiert auf Seite 12)
- [DBM89] Thomas A. DeFanti, Maxine D. Brown, and Bruce H. McCormick. Visualization: Expanding scientific and engineering research opportunities. *Computer*, 22(8):12–25, August 1989. (zitiert auf Seite 5)

- [For94] Message P Forum. Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994. (zitiert auf Seite 8 und 19)
- [Fre88] Karen A. Frenkel. The art and science of visualizing data. *Commun. ACM*, 31(2):111–121, February 1988. (zitiert auf Seite 5)
- [FTB⁺88] J. Franco, G. Tenorio-Tagle, P. Bodenheimer, M. Rozyczka, and I. F. Mirabel. On the origin of the Orion and Monoceros molecular cloud complexes. , 333:826–839, October 1988. (zitiert auf Seite 8)
- [Gas] Alban Gashi. IT Hardware. <https://thenextpractices.files.wordpress.com/2011/03/cs-it-hardware.pdf>. Zugriff: 29.04.2015. (zitiert auf Seite 2)
- [HF11] A. Hassan and C. J. Fluke. Scientific Visualization in Astronomy: Towards the Petascale Astronomy Era. , 28:150–170, June 2011. (zitiert auf Seite 1 und 13)
- [iee08] IEEE Standard for Floating-Point Arithmetic. Technical report, Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY 10016-5997, USA, August 2008. (zitiert auf Seite 9)
- [LH14] Zhicheng Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2122–2131, Dec 2014. (zitiert auf Seite 2)
- [MADH80] G. Manhes, C. J. Allègre, B. Dupré, and B. Hamelin. Lead isotope study of basic-ultrabasic layered complexes: Speculations about the age of the earth and primitive mantle characteristics. *Earth and Planetary Science Letters*, 47:370–382, May 1980. (zitiert auf Seite 12)
- [McC87] B. McCormick. Visualization in scientific computing. *Computer Graphics*, 21(6), 1987. (zitiert auf Seite 5)
- [MOR63] C. A. Muller, J. H. Oort, and E. Raimond. Hydrogène neutre dans la couronne galactique? *Academie des Sciences Paris Comptes Rendus*, 257:1661–1662, 1963. (zitiert auf Seite 7)
- [MXW⁺13] Li-Qian Ma, Kun Xu, Tien-Tsin Wong, Bi-Ye Jiang, and Shi-Min Hu. Change blindness images. *Visualization and Computer Graphics, IEEE Transactions on*, 19(11):1808–1819, Nov 2013. (zitiert auf Seite 26)
- [NHT01] L. Nowell, E. Hetzler, and T. Tanasse. Change blindness in information visualization: a case study. In *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pages 15–22, 2001. (zitiert auf Seite 26)
- [SG15] B. Sander and G. Hensler. Saturated heat conduction in high-velocity clouds. *Astronomy and Astrophysics*, 2015. (zitiert auf Seite 8)

- [SK94] Lisa M. Sobierajski and Arie E. Kaufman. Volumetric ray tracing. In *Proceedings of the 1994 Symposium on Volume Visualization*, VVS '94, pages 11–18, New York, NY, USA, 1994. ACM. (zitiert auf Seite 19)
- [SKMH14] R. Sicat, J. Kruger, T. Moller, and M. Hadwiger. Sparse pdf volumes for consistent multi-resolution volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2417–2426, Dec 2014. (zitiert auf Seite 5)
- [SW11] Robert Sedgewick and Kevin Wayne. *Algorithms, 4th Edition*. Addison-Wesley, 2011. (zitiert auf Seite 15)
- [TBRF87] G. Tenorio Tagle, P. Bodenheimer, M. Rozyczka, and J. Franco. Collisions of High-Velocity Clouds with a Galactic Disk. , 14:241, May 1987. (zitiert auf Seite 8)
- [TheNN] The HDF Group. Hierarchical Data Format, version 5, 1997-NNNN. <http://www.hdfgroup.org/HDF5/>. (zitiert auf Seite 9)
- [TLB⁺11] D. Thompson, J.A. Levine, J.C. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P.P. Pebay. Analysis of large-scale scalar data using hixels. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 23–30, Oct 2011. (zitiert auf Seite 5)
- [TRH98] S. L. Tufte, R. J. Reynolds, and L. M. Haffner. WHAM Observations of H α Emission from High-Velocity Clouds in the M, A, and C Complexes. , 504:773–784, September 1998. (zitiert auf Seite 7)
- [TSO⁺11] M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, and M. L. Norman. yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. , 192:9–+, January 2011. (zitiert auf Seite 14)
- [TWM⁺02] S. L. Tufte, J. D. Wilson, G. J. Madsen, L. M. Haffner, and R. J. Reynolds. WHAM Observations of H α from High-Velocity Clouds: Are They Galactic or Extragalactic? , 572:L153–L156, June 2002. (zitiert auf Seite 7)
- [Uni] Griffith University. Average computer energy usage. <http://www.griffith.edu.au/sustainability/sustainable-campuses/sustainable-initiatives/energy/average-computer-energy-usage>. Zugriff: 29.04.2015. (zitiert auf Seite 2)
- [vW04] H. van Woerden and B. P. Wakker. Distances and Metallicities of HVCS. In H. van Woerden, B. P. Wakker, U. J. Schwarz, and K. S. de Boer, editors, *High Velocity Clouds*, volume 312 of *Astrophysics and Space Science Library*, page 195, January 2004. (zitiert auf Seite 7)
- [Wak01] B. P. Wakker. Distances and metallicities of high- and intermediate-velocity clouds. *The Astrophysical Journal Supplement Series*, 136(2):463, 2001. (zitiert auf Seite 7)

- [Wes91] Lee Alan Westover. *Splatting: A Parallel, Feed-forward Volume Rendering Algorithm*. PhD thesis, Chapel Hill, NC, USA, 1991. UMI Order No. GAX92-08005. (zitiert auf Seite 18)
- [WWLM11] Yu-Shuen Wang, Chaoli Wang, Tong-Yee Lee, and Kwan-Liu Ma. Feature-preserving volume data reduction and focus+context visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(2):171–181, Feb 2011. (zitiert auf Seite 5)
- [YMC06] Hamid Younesy, Torsten Möller, and Hamish Carr. Improving the quality of multi-resolution volume rendering. In Beatriz Sousa Santos, Thomas Ertl, and Kenneth I. Joy, editors, *EuroVis*, pages 251–258. Eurographics Association, 2006. (zitiert auf Seite 5)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 4. Mai 2015