Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 193

Visualisierung von Wellenstrukturen in Molekular-Dynamik-Systemen mittels Frequenzanalyse

Maximilian Schmidt

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Thomas Ertl
Betreuer/in:	Dipl.-Inf. Katrin Scharnowski, Dr. Guido Reina
Beginn am:	15. Dezember 2014
Beendet am:	16. Juni 2015
CR-Nummer:	I.3.5, I.3.8, I.6.6, J.2

Kurzfassung

Durch den im Laufe dieser Bachelorarbeit entwickelten Prototypen wird das Problem angegangen, dass sich die Grenzschicht von Molekulardynamik-Simulationen, insbesondere bei Abdampfungsvorgängen, ohne weitere Hilfsmittel kaum sinnvoll analysieren lässt. Daher wird nach einer Einarbeitung in die Thematik ein Prototyp vorgestellt, welcher aus der Punktewolke einer Molekulardynamik-Simulation eine Oberfläche rekonstruiert. Dies geschieht mit Hilfe von radialen Basisfunktionen, durch welche eine implizite Oberfläche entsteht. Anschließend wird basierend auf dieser ein Höhenfeld extrahiert, welches die Grenzschicht möglichst gut wiedergibt. Als letzter Schritt wird darauf wiederum eine Fourier-Transformation durchgeführt, um dieses Höhenfeld in den Frequenzbereich zu transformieren. Mehrere dieser Frequenzverteilungen werden über die Zeit gestackt und als Space-Time Cube gerendert. Ergebnisse sowie Vergleiche zwischen diesen zeigen den Prototypen in Aktion und ein abschließender Ausblick stellt Anknüpfungspunkte vor.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

1. Einleitung

Nachfolgend werden Motivation und Zielsetzung sowie die Gliederung dieser Arbeit erläutert, um einen besseren Überblick zu geben.

1.1. Motivation

Molekulardynamik-Simulationen beschreiben das Verhalten von Molekülen, deren Bewegung nach einem mathematischen Modell berechnet wurden [Sch]. Während des Abdampfungsvorgangs in solchen Simulationen, in welchen bei den in dieser Arbeit genutzten Daten Flüssigkeitsfilme simuliert werden, bilden sich an der Grenzschicht des Materials wellenförmige Muster aus. Um die Analyse dieser Muster zu unterstützen und die Vorgänge an der Grenzschicht besser verstehen zu können, möchte man diese in einer geeigneten Visualisierung näher betrachten (Abb. 1.1).

1.2. Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines Prototypen auf der Basis von C/C++ und OpenGL, welcher die Visualisierung und Analyse bestimmter thermodynamischer Vorgänge, wie zum Beispiel von Abdampfungsvorgängen, unterstützen soll. Dafür muss auf Basis der gegebenen Daten für jeden Zeitschritt die Oberfläche rekonstruiert und daraus ein Höhenfeld extrahiert werden, auf welchem wiederum eine Transformation in den Frequenzraum mittels *Fourier-Transformation* durchgeführt werden soll. Dies liefert ein Ergebnis, welches über die Zeit gestackt und als Volumen gerendert wird.

1.3. Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Theorie erläutert Konzepte aus der Literatur sowie den mathematischen Hintergrund zur Oberflächenrekonstruktion und zur Fourier-Analyse.

Kapitel 3 – Konzept stellt das genutzte Konzept vom Filtern der Daten bis zum Volumenrendering der gestackten Frequenzbilder vor.

Kapitel 4 – Implementierung geht auf die Erstellung eines Plug-Ins in MegaMol [Meg, GKM+15], die Implementierung und die verwendete Technik ein.

Kapitel 5 – Ergebnisse zeigt und vergleicht Ergebnisse unter Nutzung des implementierten Tools.

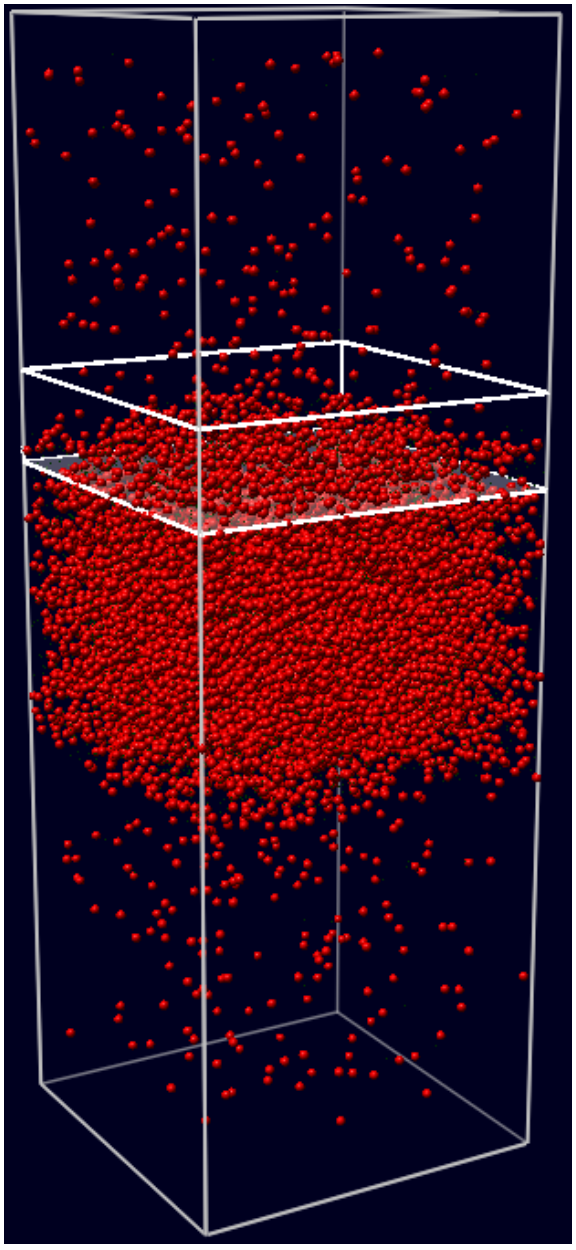


Abbildung 1.1.: Ohne weitergehende Analyse der Grenzschicht lassen sich die Simulationsdaten nur mäßig gut analysieren.

Kapitel 6 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und zeigt Ausblicke auf.

2. Theorie

Um die Grenzschicht (Abb. 2.1) in Molekulardynamik-Simulationen, welche bei dieser Arbeit die Bewegung von unstrukturierten Partikeln beschreiben, analysieren zu können, muss erst ein Höhenfeld extrahiert werden. Dies erfordert wiederum eine Konkretisierung der Grenzschicht. Diese Definition lässt sich auf das Problem der Oberflächenrekonstruktion verallgemeinern:

2.1. Oberflächenrekonstruktion

Aus einer Punktwolke soll durch Approximation [DG06, BLN+13], optional mit Normalen oder aus einem Tiefen-/Höhenbild [BLN+13], die gesuchte Oberfläche rekonstruiert werden. Anders ausgedrückt versteht man unter Oberflächenrekonstruktion eine explizitere Darstellung einer Punktwolke eines gegebenen Objekts in der Form eines Dreiecksmeshes, einer Isofläche (implizit) bzw. die Umwandlung in eine solche oder als eine Menge von parametrisierten Primitiva. Weitergehend fallen darunter auch Prozesse, welche die Grenzen der Oberfläche identifizieren und interpretieren können [HAAT95]. Abb. 2.2 zeigt eine solche Oberflächenrekonstruktion.

Grundsätzlich gibt es verschiedene Szenarien der Oberflächenrekonstruktion: Daten

- von Laser Range Scannern: liefern ein rechteckiges Gitter deren Knoten die Distanzen vom Sensor bis zur gescannten Oberfläche enthalten [HDD+92, ABK98]
- von (digitalisierten) Konturen: treten oft im medizinischen Bereich auf und bestehen aus 2D Bildern mit den Konturen der betrachteten Objekte, aus denen man ein 3D Bild gewinnen möchte [HDD+92, ABK98]

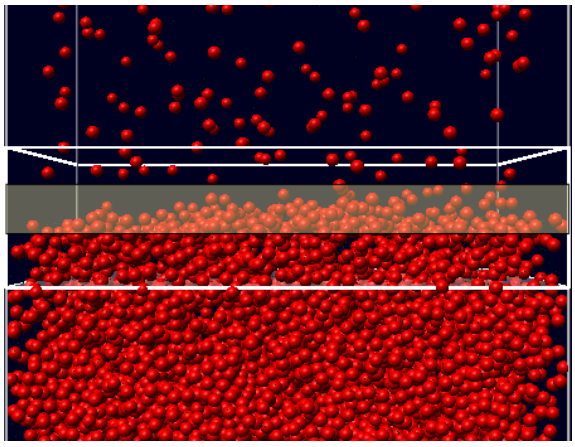
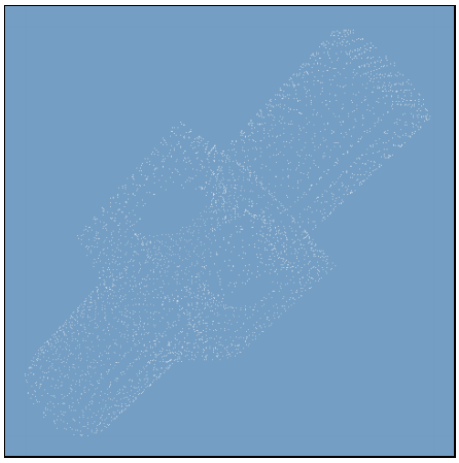
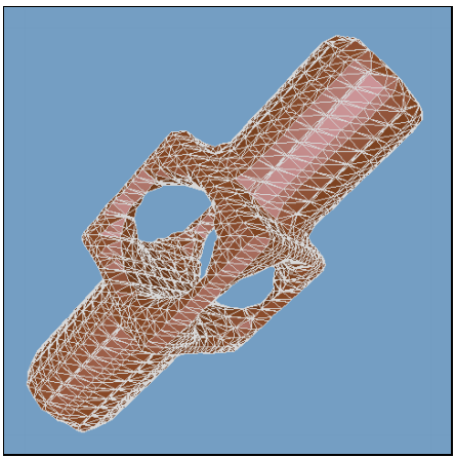


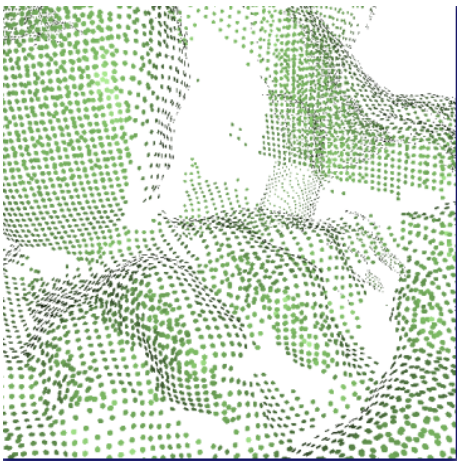
Abbildung 2.1.: Simulationsdaten mit der Grenzschicht, eingefärbt in hellem Gelb.



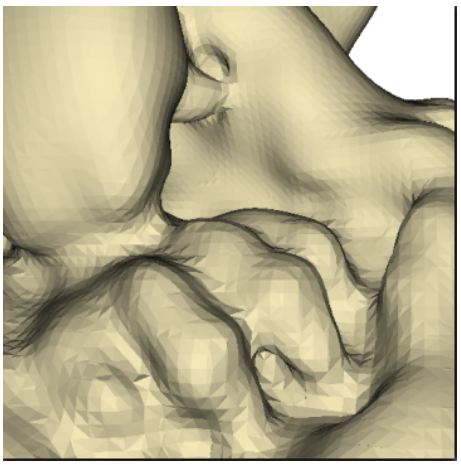
(a) Oberflächenrekonstruktion: Eine Punktwolke, aus der die Oberfläche rekonstruiert werden soll.



(b) Rekonstruierte Oberfläche, berechnet durch einen modifizierten Marching Cubes Algorithmus. [HDD+92]



(c) Eine weitere Punktwolke eines Objekts.



(d) Die rekonstruierte Oberfläche, durch radial basis functions approximiert. [BLN+13]

Abbildung 2.2.: Oberflächenrekonstruktion.

- von interaktiven Eingaben: modelliert durch Maus oder Stylus Pen [HDD+92]
- durch mathematische Modelle definiert [ABK98]

Durch die Ein- und Ausgabedaten sowie durch den Zweck bzw. die Weiterverarbeitung der rekonstruierten Oberfläche werden diverse Anforderungen an Algorithmen gestellt, welche diese Oberfläche berechnen. Diese sind unter anderem, dass die Algorithmen mit komplexer Geometrie und Topologie sowie Rauschen umgehen können sollten, um eine gute Approximation der Oberfläche zu erhalten [ZOF01].

Falls die Daten, genauer die Punktwolke, eine Struktur - beispielsweise durch Dichtewerte - aufweisen, kann diese bei der Rekonstruktion ausgenutzt werden, was allerdings in drei oder mehr Dimensionen nicht trivial ist [ZOF01]; ansonsten sind Algorithmen, welche auf unstrukturierten bzw. unorganisierten Punkten arbeiten, die einzige Möglichkeit und damit nicht für organisierte

Punktwolken geeignet, da sie für den Rekonstruktionsprozess nützliche Informationen wie z. B. Normalen nicht berücksichtigen [CL96]. Letztere sind außerdem auch nicht trivial, da keine eindeutige Lösung existiert [ZOF01] und, wie Curless et al. [CL96] beschreiben, nicht immer in den Bereichen robust sind, in denen die Oberfläche Krümmungen aufweist.

Oberflächenrekonstruktion im Allgemeinen ist nicht zuletzt zunehmend populär aufgrund des breiten Anwendungsfeldes [DG06] und durch erschwingliche und genaue Scanner, welche eine Punktwolke liefern, dicht genug um daraus eine Oberfläche zu rekonstruieren [ABCO$^+$01] als auch durch den Fakt, dass Punkte durch die benötigte hohe Anzahl an Primitiven zur Darstellung hochauflösender Oberflächen eine effektive Möglichkeit darstellen, diese zu repräsentieren.

Für den Prozess der Oberflächenrekonstruktion ergeben sich mitunter einige Schwierigkeiten wie eine oft fehlende Gleichmäßigkeit, mit der die Punkte gesampelt wurden sowie Rauschen, welches durch eventuelle Ungenauigkeiten beim Sampling auftreten kann [KBH06].

Interpolation im Kontext der Oberflächenrekonstruktion bedeutet, dass die Oberfläche so rekonstruiert wird, dass unbekannte Werte durch alle [DTS02, BLN$^+$13] oder einen Teil [BLN$^+$13] der bekannten Punkte interpoliert und damit bestimmt werden und diese Oberfläche genau in den bekannten Punkten liegt [DTS02, SOS04, HAAT95]. Die Interpolation empfiehlt sich, falls die Daten präzise sind und somit weder Rauschen noch Ausreißer auftauchen, welche das Ergebnis beeinflussen könnten [DTS02].

Bei der Approximation wird die Oberfläche so konstruiert, dass diese nah an den Eingabedaten liegt (aber nicht genau in diesen) [DTS02, BLN$^+$13, HAAT95] und glättet somit Geometrie und Topologie [SOS04] und ist von Vorteil, wenn Fehler in den Eingabedaten auftauchen [DTS02]. Dazu gibt es viele verschiedene Ansätze mit angemessener Oberfläche als Ergebnis, welche bei hoher Punktedichte recht eindeutig ist. Diese werden in Abschnitt 2.1.1 und 2.1.2 näher erläutert. Abhängigkeiten bestehen unter anderem von den Eingabedaten bzw. deren Eigenschaften, der gewünschten Ausgabe und der verfügbaren Software [GP07].

Nach Gross et al. [GP07] und Berger et al. [BLN$^+$13] sollte die Wahl des Algorithmus zu einem wesentlichen Teil von den Eingabedaten abhängen, da Punktwolken mit viel Rauschen in der Hinsicht verarbeitet werden sollten, dass nicht alle Informationen gewünscht sind und herausgefiltert werden müssen. Für die im Zuge dieser Bachelorarbeit verwendeten Daten ist diese Abhängigkeit von den Daten nicht trivial, da die Punktwolke nicht durch Sampling aus einer Oberfläche entstanden ist, sondern vielmehr eine sinnvolle gefunden werden muss. Daher lassen sich übliche Kriterien, wie z. B. Rauschen, schlecht bestimmen. Zwischen diesen Extrema liegen Eingabedaten mit wenig Rauschen an bestimmten, anfälligen Stellen [GP07].

Anstatt ein Fehlermodell miteinzubeziehen, haben sich Wood et al. [WHDS04] damit beschäftigt, *topologisches Rauschen* in der rekonstruierten Oberfläche zu beseitigen. Dabei bezieht sich topologisches Rauschen auf Fehler in der Struktur von Objekten, beispielsweise Tunnel, die bei der Erfassung des Modells aufgrund von Rauschen entstanden sind [GW01].

Des Weiteren hängt die Wahl eines geeigneten Algorithmus in manchen Fällen auch von der gewünschten Ausgabe ab [GP07, BLN$^+$13]: Wird die angestrebte Oberfläche zum Beispiel zur Parametrisierung oder für eine *Finite-Elemente-Analyse* eingesetzt, so sollte diese wasserdicht sein, also keine Löcher aufweisen [GP07].

Einschränkungen bezüglich des Aussehens der rekonstruierte Oberfläche erlauben robustere Algorithmen, da dadurch weniger Fehler möglich sind. Dennoch muss die Methode auf die Eingabedaten eingehen: Wenn wie im vorhergehenden Beispiel eine wasserdichte Oberfläche von

einem Objekt mit vielen Löchern erstellt werden soll, so muss diese Methode, welche vorzugsweise Oberflächen mit Rändern erzeugt, große Teile interpolieren, was wiederum zu Uneindeutigkeiten in der Darstellung führen kann [GP07].

Zusätzlich zu den genannten Punkten, der Ein- und Ausgabe, hängt die Wahl auch von der theoretischen Analyse des Algorithmus ab, wobei insbesondere betrachtet wird, wie gut ein Algorithmus die gesuchte Oberfläche approximiert und wann dieser somit einwandfrei arbeitet. Dafür wird eine Punktwolke aus einer Oberfläche gesampelt, für die angenommen wird, dass sie dicht genug ist, um daraus die Oberfläche rekonstruieren zu können und anschließend gezeigt, dass die Topologie korrekt ist und die Geometrie gut approximiert wird [GP07].

Des Weiteren sind bereits Algorithmen vorhanden, welche theoretische Garantien geben, dass das Ergebnis bzw. die Oberfläche wie erwähnt korrekt ist, sogar bei einem Vorhandensein von Rauschen [DG06], was allerdings nicht unbedingt bedeutet, dass diese Algorithmen in der Praxis nützlich sind, da Oberflächen üblicherweise nicht überall glatt erscheinen sowie Daten nicht dicht genug gesampelt sind und Rauschen enthalten, was diese Algorithmen außen vor lassen [ABK98].

Diese genannten Garantien basieren auf einer vorausgesetzten Dichte relativ zur *local feature size*, welche oft allerdings durch Rauschen, welches aufgrund der Natur der Datenakquisition bei Scannern zu Stande kommt, nicht gegeben ist [DG06].

Bisher wurde oft erwähnt, dass eine Punktwolke dicht genug gesampelt sein muss, ohne eine konkrete Definition abzugeben. Amenta et al. [ABK98] haben eine solche aufgestellt. Sie beschreiben ein „gutes“ Sample als solches, welches dicht in lokal detaillierten Bereichen und eventuell weniger dicht an Stellen ohne Details gesampelt ist [ABK98].

Auch wenngleich für Algorithmen theoretische Garantien existieren, hat man sich wenig mit deren Evaluierung beschäftigt. Berger et al. [BLN+13] haben in einem ersten Ansatz verschiedene Algorithmen zur Oberflächenrekonstruktion miteinander in einem Benchmark verglichen.

Einen einheitlichen und bekannten Ansatz, welcher auf unorganisierten Punkten arbeitet, d. h. unabhängig von der Struktur der Eingabedaten ist, stellt der Algorithmus von Hoppe [HDD+92] dar: Finde die k-nächsten Nachbarn von Punkt p, $N_k(p)$ und nimm die Normale der orthogonalen Regression durch $N_k(p)$ als Oberflächennormale an p. Ein Problem tritt dabei auf, falls die Punkte nicht gleichmäßig verteilt sind, denn dann muss die durch die Nachbarschaft $N_k(p)$ bestimmte Normale nicht die gesuchte für p sein, sondern ist im schlimmsten Fall senkrecht dazu. Auch wenn man alle Punkte um einen festen Radius r um p für die Berechnung dessen Normalen heranzieht, so führt dies zu denselben Problemen [GP07]. Mitra et al. [MN03] haben dieses Problem weiter verfolgt und wählen die Nachbarschaft adaptiv, basierend auf p.

Eine weitere Idee ist, die Komplexität der Oberfläche zu reduzieren, indem die Darstellung der Oberfläche in verschiedenen Auflösungen erfolgt, um das *level of detail* (LOD) zu kontrollieren [Sze10]. Diese Technik findet in der gestellten Aufgabe allerdings keine Anwendung und wird daher auch nicht ausführlicher beschrieben, da unterschiedliche LOD nicht genutzt werden und die Auflösung fest sein sollte, um ein eindeutiges und reproduzierbares Ergebnis aus jeder Perspektive zu erhalten.

Nach [DTS02, TO02, BLN+13, CL96, ZOF01, HDD+92] können verschiedene Methoden zur Rekonstruktion in explizite und implizite Algorithmen kategorisiert werden.

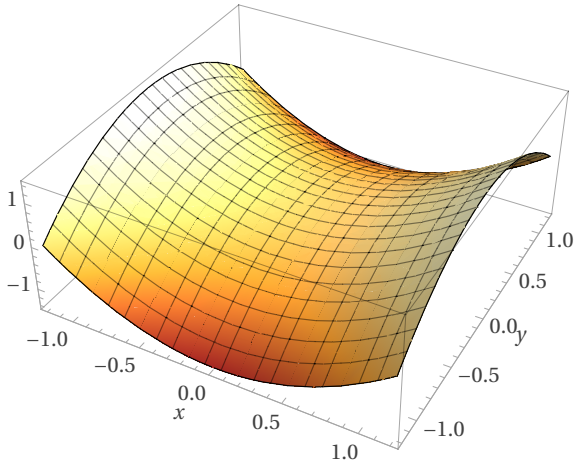


Abbildung 2.3.: Eine beispielhafte parametrische Oberfläche mit $f(x, y) = x^2 - y^2$ [Wol].

2.1.1. Explizite Algorithmen

Explizite oder auch parametrische Algorithmen schreiben die präzise Lage von Oberflächen im Raum vor, beispielsweise durch Parameter oder diskrete Dreiecksmeshes, welche allerdings eine gute Parametrisierung benötigen, sodass die aus den einzelnen Punkten oder Partikeln rekonstruierte Oberfläche ein Graph im Parameterraum ist, was durchaus eine Herausforderung im drei- oder höherdimensionalen Raum sein kann. Des Weiteren lässt sich eventuelles Rauschen in den Daten durch Parameter schwer beschreiben [ZOF01]. Nichtsdestotrotz werden besonders explizite Methoden am meisten für die Repräsentation genutzt [DTS02], welche oft auch die Struktur der Daten mit einbeziehen [CL96]. Abb. 2.3 zeigt beispielhaft eine parametrische Oberfläche.

Eine Möglichkeit, eine Oberfläche explizit zu beschreiben, ist, die *Delaunay Tetraedisierung* [Kaz05, KSO04] oder *Voronoi Diagramme* [Kaz05] zu berechnen und deren Datenstruktur (welche Zellen entsprechen) auszunutzen, um die topologische Verbindung zwischen Punkten der Punktwolke zu bestimmen. Delaunay Tetraedisierung bestimmt dabei für jeden Tetraeder, ob dieser innerhalb oder außerhalb des Objekts liegt, wodurch man eine wasserdichte Oberfläche ohne großen Aufwand erhält und sich die Dreiecksdichte an der Punktedichte orientiert (falls sich die Dichte der Punktwolke lokal beachtlich unterscheidet) [KSO04].

Nachteile im Fall einer diskreten Darstellung (z. B. als Dreiecksmesh) sind, dass glatte Oberflächen nur approximiert werden können, und dies wiederum auch nur mit einer fixen Auflösung [DTS02]. Dagegen müssen parametrische Oberflächen mehrere „Fetzen“ kombinieren, um eine geschlossene Oberfläche darzustellen [DTS02].

Allerdings können parametrische Oberflächen, soweit es keine weiteren Einschränkungen gibt, in beliebigen Auflösungen dargestellt werden [DTS02].

2.1.2. Implizite Algorithmen

Nachdem Ansätze für explizite Algorithmen sowie deren Anwendung besprochen wurden, soll nun noch auf implizite Methoden zur Rekonstruktion von Oberflächen eingegangen werden. Implizite oder auch volumetrische Algorithmen sind eine weit verbreitete Möglichkeit, um eine

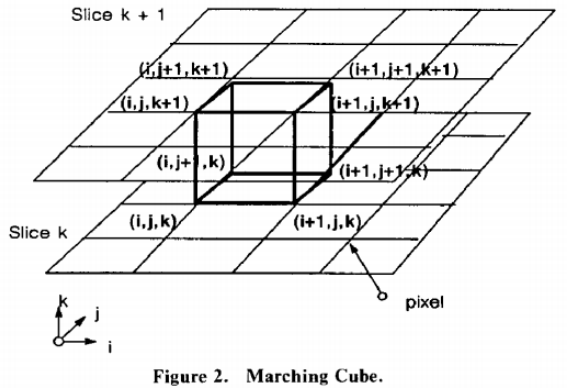


Abbildung 2.4.: Der Marching Cubes Algorithmus durchläuft ein Voxelgitter und bestimmt die Schnittpunkte des Kubus mit der Isofläche [LC87].

Oberfläche als Isofläche einer Skalarfunktion darzustellen [ZOF01, CBC+01]. Außerdem müssen die Daten nicht auf einem regulären Gitter liegen, um solche Isoflächen zu bestimmen [CFB97].

Eine Möglichkeit für diese Darstellung ist, eine glatte (implizite) Funktion $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ zu suchen, sodass $x_1, ..., x_n$ auf der Nullmenge $Z(f)$ liegt [HDD+92], wie es bei den Algorithmen von Hoppe sowie denen basierend auf *radial basis functions* und *level sets* zum Einsatz kommt. Die Nullmenge entspricht dabei der Isofläche, welche die rekonstruierte Oberfläche als implizite wiedergibt [ZOF01].

Soll die implizit beschriebene Oberfläche aus der Isofläche rekonstruiert werden - beispielsweise zum Rendern oder um ein Höhenfeld zu extrahieren - muss die Oberfläche aus dieser extrahiert werden [Kaz05], wobei der Isowert zum Beispiel so gewählt wird, dass die rekonstruierte Oberfläche die Eingabedaten gut approximiert [KBH06]. Anschließend wird daraus ein Dreiecksmesh bzw. Polygonmodell [DTS02] oder eine der Anwendung entsprechende Repräsentation durch Raytracing [MYR+05] erstellt.

Ein Algorithmus für die Oberflächenextraktion aus Isoflächen ist der *Marching Cubes Algorithmus*, welcher Polygonmodelle aus Oberflächen konstanter Dichte erstellt, Ausgabe ist demnach ein Dreiecksmesh. Für diese Berechnung benötigt der Algorithmus allerdings ein Voxelgitter, siehe Abb. 2.4 [LC87].

Des Weiteren gibt es wie von Kazhdan et al. [KBH06] beschrieben noch weitere Möglichkeiten für die Isoflächenberechnung, eine basiert auf den bereits genannten radial basis functions. Radial basis functions approximieren eine Oberfläche $f(x)$ durch mehrere Daten d_i an Position x_i:

$$f(x) = \frac{\sum_i \omega_i(x) d_i}{\sum_i \omega_i(x)} \tag{2.1}$$

mit den Gewichten $\omega_i(x) = K(||x - x_i||)$ und einer radialen Basisfunktion $K(r)$, welche glatt, monoton fallend, überall positiv definit und sphärensymmetrisch ist, wie z. B. die Gauß- bzw. Normalverteilung [Sze10, MYR+05, ABCO+01]. Anders ausgedrückt ist $f(x)$ eine Summe von gewichteten radial basis functions [DTS02], welche an den Punkten der Punktwolke zentriert werden [TO02, MYR+05, DTS02, TO99], nur von der Distanz des Arguments abhängen [KP06] und somit eine Funktion mit n Punkten durch n radial basis functions mit jeweils n Summanden interpoliert wird, was bei naiver Implementierung zu einer Laufzeitkomplexität von $O(n^2)$ führt.

Abb. 2.2c und 2.2d zeigen beispielhaft eine implizit durch radial basis functions rekonstruierte Oberfläche.

Diese impliziten Methoden, welche Basisfunktionen gewichten und aufsummieren, sind sehr teuer bezogen auf den benötigten Berechnungsaufwand, da sich die Änderung eines Punktes auf die gesamte Isofläche auswirkt [ZOF01]. Des Weiteren kann es nötig sein, ein Polynom ersten Grades zu addieren, um die Linearität von f zu gewährleisten sowie zu einer positiv definiten Lösung beizutragen [MYR+05].

Eine weitere Möglichkeit zur Oberflächenrekonstruktion basiert auf einem Poissonansatz, welche eine sehr glatte Oberfläche aus rauschenbehafteten Daten approximieren [KBH06].

Diverse Vorteile messen den impliziten Methoden einen hohen Stellenwert bei: Sie rekonstruieren „wasserdichte“ Oberflächen, d. h. es entstehen weder Löcher noch eine Einschränkung der extrahierten Oberfläche bzgl. dessen Topologie und Komplexität, wodurch die Repräsentation als implizite Oberfläche für viele (3D) Modelle genutzt werden kann [Kaz05]. Falls außerdem nicht ausreichend Daten verfügbar sind, die Punktwolke also wie beschrieben nicht dicht genug ist oder Rauschen auftaucht, kann die Oberfläche dennoch gut approximiert werden [DTS02]. Lokales Detailreichtum bei gleichzeitiger Glätte (Regelmäßigkeit) der gesamten Oberfläche lassen dieses Modell für viele Operationen, wie etwa Kollisionserkennung bzw. -berechnung, topologische Operationen und Blending (Überschneiden bzw. Kombinieren von Körpern, in diesem Fall von Oberflächen), eine gute Wahl werden [DTS02].

Wie bereits erwähnt zieht eine Änderung eines Punktes eine Veränderung der gesamten Oberfläche nach sich, wodurch ein hoher Berechnungsaufwand entsteht [MYR+05, FN80, LM02] und die Komplexität der Rekonstruktion von der Auflösung des Voxelgitters abhängig ist [Kaz05]. Um diesen hohen Berechnungsaufwand zu kompensieren, können die Funktionen lokal eingegrenzt werden, was wiederum den Berechnungsaufwand reduziert [Kaz05, WKE99, WVG92]. Außerdem ermöglichen schnellere, bessere und günstigere Prozessoren neue und komplexere Modelle, u. a. basierend auf Variationsansätzen und verbesserten, *compactly supported* radial basis functions, welche durch das ebenso endliche Ausmaß der Funktionen das Problem angehen, dass radial basis functions globalen Einfluss haben. Für *konstruktive Festkörpergeometrie* (CSG) gibt es solche Ansätze allerdings noch nicht, da es mehrere mögliche Nullmengen und damit verschiedene in Frage kommende Isoflächen gibt [MYR+05].

Die hohe Anzahl an vorhandenen Algorithmen und deren Funktionsweisen machen deutlich, dass für die gestellte Aufgabe ein passender Ansatz zur Oberflächenrekonstruktion, basierend auf den Eigenschaften der Daten, gewählt werden muss [ABK98].

Da es für die Analyse von Wellenstrukturen in Grenzschichten Sinn macht, wenn Partikel von einem Frame auf den anderen nicht plötzlich verschwinden oder erscheinen, wird die Oberflächenrekonstruktion im Prototypen durch die implizite Methode basierend auf radial basis functions durchgeführt. Außerdem wurden bereits in anderen Arbeiten auf gleiche Verfahren bei ähnlichen Daten zurückgegriffen [KSES12].

2.2. Fourier-Analyse

Nachdem die Rekonstruktion von Oberflächen eingehend behandelt wurde soll nun noch die notwendige Theorie zur ebenfalls in dieser Arbeit genutzten Fourier-Analyse behandelt werden.

Auf diese wird jedoch nur kurz eingegangen, da nur die Transformation von Bildern in den Frequenzbereich für die Implementierung genutzt wird.

Die Idee der Fourier-Transformation ist, dass jede Funktion $f(x)$ als Summe einer Reihe von Sinus- und Kosinusfunktionen mit unterschiedlichen Frequenzen ausgedrückt werden können [Yoo01]. Wie Funktionen über die Zeit können auch solche über den Raum, wie z. B. Bilder, durch die *Fourier-Reihe* in den Frequenzbereich transformiert werden.

Im Frequenzraum können Filteroperationen, wie etwa Konvolution, schneller berechnet werden, Rauschen kann erkannt und eliminiert sowie Messungen, Rekonstruktion und Bildkompression ermöglicht werden [Yoo01].

Diese unendliche Summe wird beschrieben durch die kontinuierliche Fourier-Transformation [Bra00, Osg09]

$$\mathcal{F}(p) = \int_{-\infty}^{\infty} f(m) e^{-2\pi i p m} \mathrm{d}m \tag{2.2}$$

mit der Inversen

$$f(m) = \int_{-\infty}^{\infty} \mathcal{F}(p) e^{2\pi i p m} \mathrm{d}p. \tag{2.3}$$

Die (eindimensionale) Diskrete Fourier-Transformation (DFT) eines N-Tupels $f = (f[0], \ldots, f[N-1])$ ergibt sich damit aus

$$F[p] = \sum_{m=0}^{M-1} f[m] e^{-2\pi i p m / M} \tag{2.4}$$

mit $p = 0, \ldots, M-1$

bzw. die inverse DFT durch

$$f[m] = \sum_{p=0}^{N-1} F[p] e^{2\pi i p m / M} \tag{2.5}$$

mit $m = 0, \ldots, M-1$.

Die im Bereich der Bildverarbeitung häufig genutzte diskrete 2D Fourier-Transformation erhält man dank Separierbarkeit durch zeilen- und spaltenweises Anwenden der 1D Fourier-Transformation:

$$F[p][q] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m][n] e^{-2\pi i p m / M} e^{-2\pi i q n / N} \tag{2.6}$$

mit $p = 0, \ldots, M-1$; $q = 0, \ldots, N-1$.

Die inverse 2D DFT ergibt sich damit analog durch

$$F[m][n] = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} f[p][q] e^{2\pi i p m / M} e^{2\pi i q n / N} \tag{2.7}$$

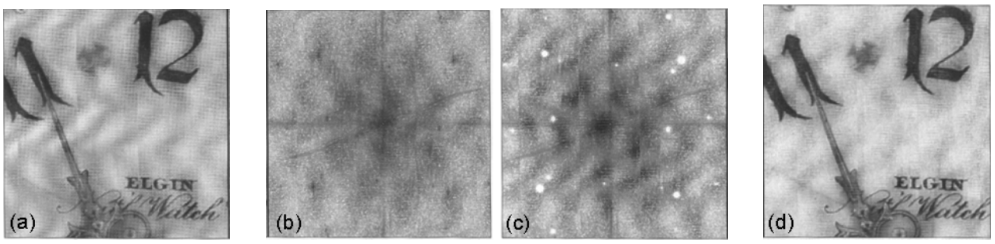


Abbildung 2.5.: Beispiel einer Frequenzanalyse in der Bildverarbeitung. Aus dem Originalbild (a) entsteht durch die Diskrete Fourier-Transformation das Frequenzspektrum (b). Das im Originalbild periodisch auftauchende Rausch erscheint dort als einzelne Punkte, welche in (c) entfernt wurden. Schließlich wird die inverse DFT auf das modifizierte Frequenzbild angewendet und man erhält das rekonstruierte Bild (d), welches weniger Rauschen als das Originalbild enthält [Yoo01]

wobei $m = 0, \ldots, M-1; n = 0, \ldots, N-1$.

Wendet man diese auf ein Bild an, so erhält man wie in Abb. 2.5 dargestellt das Frequenzspektrum, bei welchem hohe Frequenzen durch Rauschen auffallen. Bei niedrigeren Frequenzen werden teils grobe Strukturen deutlich.

Die DFT hat eine Zeitkomplexität von $O(n^2)$, wobei man durch eine geschickte Zerlegung von Gleichung (2.4) die *Fast Fourier Transform* (FFT) mit einer Laufzeit von $O(n \log_2 n)$ erhält [CLW69, Bra00, Osg09].

Die Fourier-Transformation wird oft auf Bilder der Astronomie, Mikrobiologie sowie für chemische Strukturen oder physikalische Prozesse angewandt, da es diese ermöglicht, periodische Strukturen bzw. Vorgänge und Komponenten zu identifizieren. Damit kann unter anderem regelmäßiges Rauschen in Bildern entfernt oder ein Anti-Aliasing-Effekt erzielt werden [Yoo01].

3. Konzept

Da nun bereits ein Einblick in die Oberflächenrekonstruktion sowie Fourier-Analyse in der Literatur erfolgt ist, soll hier das Konzept erläutert werden, welches im Zuge dieser Arbeit verfolgt wird, um das gewünschte Ergebnis, ein Volumen aus gestackten Frequenzbildern zu rendern, zu erhalten.

Mit MegaMol[Meg, GKM+15] besteht bereits ein am Visualisierungsinstitut der Universität Stuttgart entwickeltes *Framework*, um punktbasierte Daten zu visualisieren. Die Funktionalität wird durch *Module* und *Calls* bereitgestellt und weitere können durch *Plug-Ins* hinzugefügt werden. Aus diesen Modulen und Calls wird ein Graph gebaut und erst zur Laufzeit zu Visualisierungs-Methoden zusammengesetzt. Diese Graphen werden als Projekt-Datei (*.mmprj oder *.xml) gespeichert und können unter anderem als Views instanziiert werden. Die Idee ist, die in Abschnitt 1.2 genannten Schritte (siehe auch Abb. 3.1) durch ein Plug-In zu implementieren und dadurch MegaMol mit der genannten Funktionalität zu erweitern.

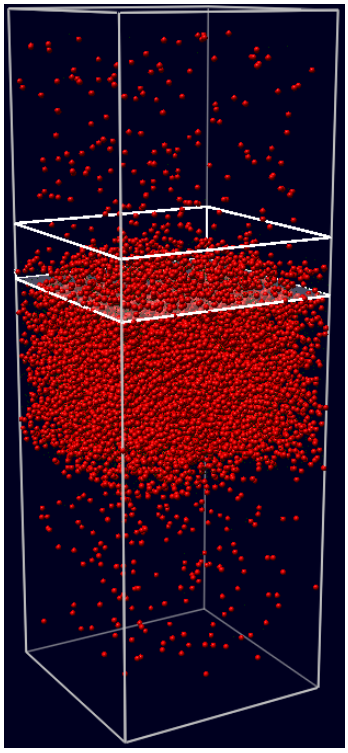
3.1. Bestimmung des Höhenfeldes

Die der Arbeit zugrundeliegenden Daten sind Ergebnisse von Molekulardynamik-Simulationen und liegen für jeden Zeitschritt wie in [ABK98, HDD+92] als unstrukturierte Partikel vor, wobei die zeitliche Auflösung von den Daten bzw. der zugrundeliegenden Simulation abhängt. Auf diesen Partikeln muss durch den Benutzer die Grenzfläche eingegrenzt werden, um sowohl die Oberflächennormale und damit die Ausrichtung der Oberfläche zu bestimmen als auch die zur Berechnung der Oberfläche in Frage kommenden Partikel einzugrenzen. Der letztgenannte Schritt wird in einem eigenen Modul realisiert, welches die Partikel auf Basis der Einschränkungen vorfiltert (Abb. 3.1a).

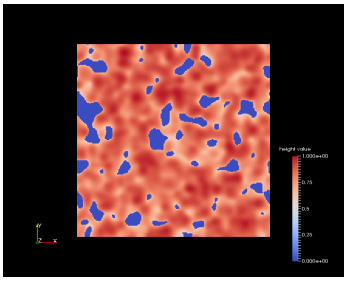
Durch den modularen Aufbau des Plug-Ins lässt sich z. B. die Bestimmung der Grenzfläche um einen Algorithmus erweitern, sodass diese automatisch durch ein Dichtesampling gefunden wird [VKFH06].

Anhand dieser Beschränkungen wird die Oberfläche durch die in Abschnitt 2.1.2 eingeführten radial basis functions approximiert. Diese eignen sich hierfür gut, da die zugrundeliegenden Daten Ergebnisse von Simulationen sind und somit zumindest kein bei der Datenakquisition entstandenes Rauschen auftritt, womit einhergehend bessere Ergebnisse als im Fall von rauschenbehafteten Daten erzielt werden können [BLN+13]. Außerdem wurden auf radial basis functions basierende implizite Oberflächen erfolgreich auf verschiedene Probleme der Computergraphik angewendet, darunter auch auf die Oberflächenrekonstruktion [TO99, DTS02].

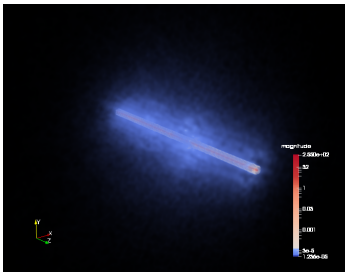
Dennoch haben implizite Oberflächen, denen radial basis functions bei der Berechnung zugrunde liegen, wie bereits erläutert eine Zeitkomplexität von $O(n^2)$ und damit eine zu hohe Laufzeit für Echtzeitanwendungen. Da die Laufzeit des Algorithmus bereits hoch ist und die Anzahl an Partikeln sehr groß sein kann, muss auf lokale Methoden zur Rekonstruktion zurückgegriffen



(a) Vorfilterung der Partikel.



(b) Extraktion des Höhenfeldes, $f = 0$, $\sigma^2 = 10$, $r = 0.5$, $\Delta = 20$, $\epsilon = 7$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.5$, $nx = ny = 256$.



(c) Gestackte Frequenzverteilungen als Volumenrendering mit den Parametern $f = 0 - 99$, $\sigma^2 = 2.5$, $r = 0.5$, $\Delta = 10$, $\epsilon = 4$, $\delta = 12$, $\vec{n} = (0, 1, 0)^T$, $\rho = 1$, $\phi = 0.1$, $nx = ny = 256$.

Abbildung 3.1.: Übersicht über die einzelnen Schritte des im Prototyp implementierten Rekonstruktionsprozesses.

werden [FN80, LM02]. Die Idee ist, als radial basis function eine Gauß- bzw. Normalverteilung einzusetzen und diese nur im Intervall $\pm 3\sigma$ zu berechnen, da bekannt ist, dass bei der Normalverteilung $99,73\%$ aller Werte in diesem Intervall liegen [Naj05]. Gleichung (3.1) zeigt die verwendete Gauß-Dichtefunktion:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.1}$$

Zur weiteren Verwendung muss aus der berechneten Isofläche ein Höhenfeld extrahiert werden, da es nicht das Ziel ist, die Oberfläche zu rendern, sondern das Höhenfeld weiterzuverarbeiten. Daher wird der Ansatz gewählt, aus der implizit berechneten Isofläche durch Raytracing senkrecht auf die vom Nutzer festgelegte Nullebene die nötigen Höhenwerte zu bestimmen. Dafür wird die Distanz zur Nullebene genommen, dessen Gitterpunkt als erster einen vom Nutzer durch einen Parameter festgelegten minimalen Wert überschreitet. Dieser muss basierend auf den Daten, den vorgenommenen Einschränkungen und der gewünschten Ausgabe angepasst werden. Diese Vorgehensweise wird in Abb. 3.2 illustriert.

Damit ist ein Höhenfeld festgelegt, basierend auf den Daten und den Einstellungen des Nutzers.

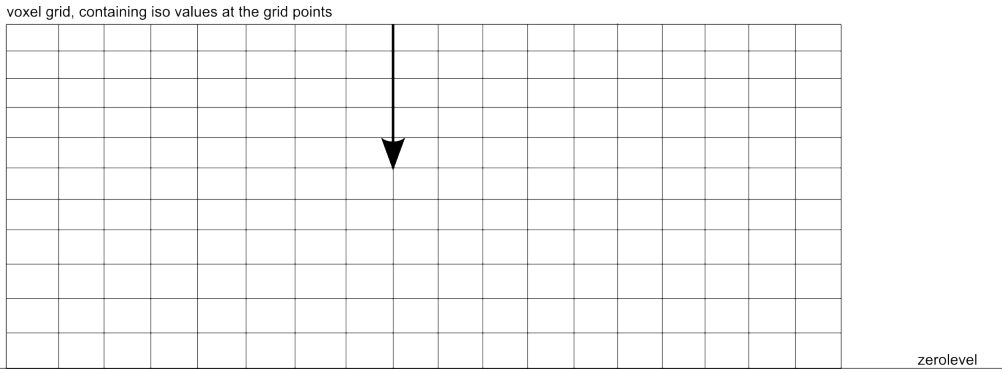


Abbildung 3.2.: Die Illustration zeigt das zur Extraktion des Höhenfeldes aus den berechneten Isoflächen verwendete Raycasting-Verfahren, wobei die dritte Dimension zum einfacheren Verständnis vernachlässigt wurde. Ausgehend von den Gitterpunkten am oberen Ende des Gitters wird dieses nach unten durchlaufen, bis man auf einen Punkt trifft, dessen Isowert den benutzerdefinierten minimalen Isowert überschreitet und dessen Höhe als Wert im Höhenfeld gesetzt wird. Sollte man auf der untersten Ebene ankommen, d. h. hatte keiner der darüberliegenden Partikel einen Isowert größer als der definierte, hat das Höhenfeld an dieser Stelle den Wert 0.

Den Algorithmus zur Oberflächenrekonstruktion und Höhenfeldextraktion zeigt Algorithmus 3.1 als Pseudocode, wobei for-Schleifen unter Verwendung von OpenMP für eine schnellere Berechnung parallelisiert worden sind.

3.2. Fourier-Analyse

Um das Grenzgebiet genauer untersuchen zu können, soll auf dem extrahierten Höhenfeld jedes Zeitschrittes der Simulation eine 2D Fourier-Analyse durchgeführt und diese über die Zeit gesammelt als gestackte Frequenzverteilungen ausgegeben werden.

Hierbei kommt die in Abschnitt 2.2 beschriebene 2D Fourier-Transformation in den Frequenzraum zum Einsatz, indem diese auf ein in jedem Zeitschritt extrahiertes Höhenfeld nach dem letzten Abschnitt angewendet wird. Als Ergebnis erhält man einen Space-Time Cube der gestackten Frequenzverteilungen, durch welchen sich Daten mit temporaler Abhängigkeit sinnvoll visualisieren lassen [BDA+14]. Dieser Space-Time Cube wird als strukturierte Punktemenge in eine *vtk*-Datei exportiert, welche wiederum mit gängigen Visualisierungsanwendungen, wie z. B. ParaView [Par], visualisiert werden kann, um das Ergebnis zu betrachten.

Dabei sollte allerdings darauf geachtet werden, eine gute Transferfunktion bei der Darstellung zu finden, um so interessante Strukturen hervorzuheben, da Fourier-Analysen unter Umständen viel Rauschen enthalten.

Gesteuert werden die Parameter sowie der Datenexport über die bereits durch MegaMol eingebundene *AntTweakbar* [Ant].

Algorithmus 3.1 Algorithmus für Oberflächenrekonstruktion

```
1:  procedure SurfaceReconstruction
2:      for all particles p do
3:          kernel ← estimateGridPoints(p)
4:          for all k ∈ kernel do do
5:              distance ← Distance(k, p)
6:              if distance ≤ 3σ then
7:                  voxelgrid[k_x][k_y][k_z] ← voxelgrid[k_x][k_y][k_z] + Gauss(distance)
8:              end if
9:          end for
10:     end for
11:     for i = 0; i < HEIGHT; i++ do
12:         for j = 0; j < WIDTH; j++ do
13:             for h = 0; h < GRIDHEIGHT; h++ do
14:                 if voxelgrid[i][j][h] ≥ MINIMUMISOVALUE then
15:                     heightmap[i][j] ← h
16:                 end if
17:             end for
18:         end for
19:     end for
20: end procedure
```

4. Implementierung

Nachdem nun das Konzept beschrieben wurde, nach welchem die Aufgabenstellung umgesetzt wird, soll hier auf die konkrete Implementierung eingegangen und die Klassen samt Funktionalität erläutert werden.

4.1. Allgemeines

Wie bereits erwähnt wird ein Plug-In für MegaMol [Meg] entwickelt, welches durch *Module* und *Calls*, die wiederum jeweils als Klasse implementiert werden, die nötige Funktionalität bereitstellt. Dieses Plug-In, namentlich *Evaporation*, wird auf Basis von C/C++ sowie OpenGL entwickelt. Abb. 4.1 zeigt ein mögliches Szenario und dessen Kontrollfluss, erstellt mit dem MegaMol - Configurator[1], sowie gleichzeitig eine Übersicht, mit welchen Modulen und Calls das Plug-In MegaMol erweitert.

Die genannten Calls verbinden Module und sorgen für den Datentransfer zwischen diesen, indem Module - in diesem Fall als *Caller* - Calls setzen und anschließend andere Module als *Callee* aufrufen können.

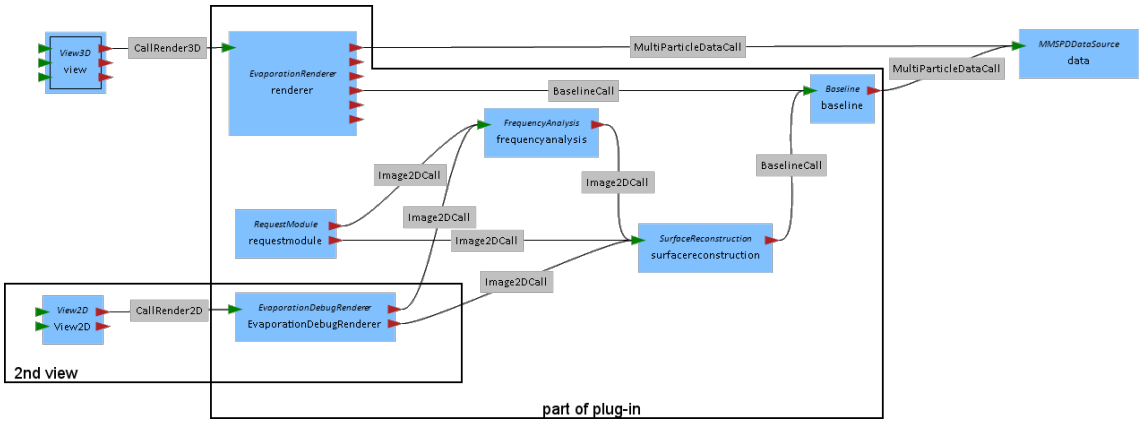
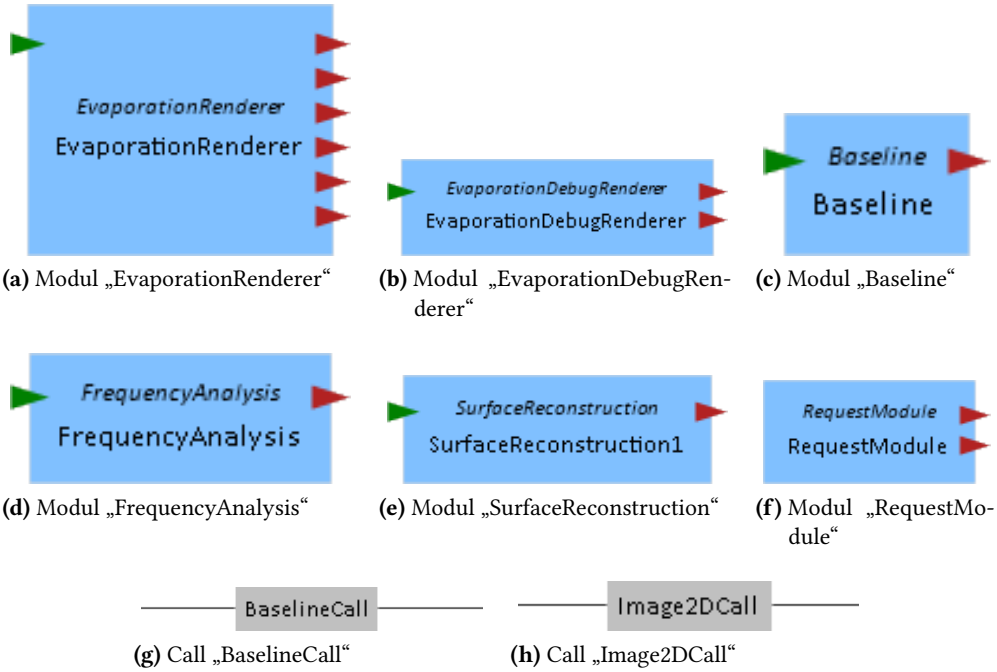


Abbildung 4.1.: Mögliches Szenario für eine Visualisierung mit MegaMol sowie Übersicht des Plug-Ins, erstellt mit dem MegaMol - Configurator



(a) Modul „EvaporationRenderer“ **(b)** Modul „EvaporationDebugRenderer“ **(c)** Modul „Baseline“

(d) Modul „FrequencyAnalysis“ **(e)** Modul „SurfaceReconstruction“ **(f)** Modul „RequestModule“

(g) Call „BaselineCall“ **(h)** Call „Image2DCall“

Abbildung 4.2.: Übersicht über die implementierten Module und Calls.

4.2. Einzelne Module und Calls

Die in diesem Plug-In bereitgestellten Module heißen namentlich *EvaporationRenderer*, *EvaporationDebugRenderer*, *FrequencyAnalysis*, *SurfaceReconstruction*, *Baseline* und *RequestModule* sowie die Calls *Image2DCall* und *BaselineCall*, siehe auch Abb. 4.2. Diese sollen im Folgenden einzeln ausführlicher behandelt werden.

Über *Slots* erhalten Module Zugang zu der Ein- und Ausgabe von Daten. Slots, über die Module aufgerufen werden können, werden als *Callee Slots* und jene, mit denen Module andere aufrufen können, als *Caller Slots* bezeichnet.

4.2.1. EvaporationRenderer

Das Modul EvaporationRenderer (Abb. 4.2a) dient zum Rendern der Partikel für den aktuellen Zeitschritt sowie der Nullebene, welches über die Calls *MultiParticleDataCall* und *BaselineCall* von dem Modul *MMSPDDataSource* als Quelle bzw. von *Baseline* die Daten erhält. Der Teil dieses Moduls, der für das Rendern zuständig ist, wurde größtenteils vom Modul *SimpleSphereRenderer* aus dem MegaMol-Core übernommen, weshalb dieser auch einen Callee Slot „rendering“ besitzt, durch welchen das Modul z. B. von *View3D* zum Rendern aufgerufen werden kann. Die Caller Slots sind „getdata“ zum Erhalten von Daten, „gettransferfunction“ um optional eine Transferfunktion zum Rendern der Partikel zu erhalten, „getclipplane“ um ebenso optional mit einem Modul zu verbinden, welches ein *clipping plane* bereitstellt, „getbaseline“ für die Daten der Nullebene und

[1] https://svn.vis.uni-stuttgart.de/projects/megamol/supplement/MegaMolConf/trunk/

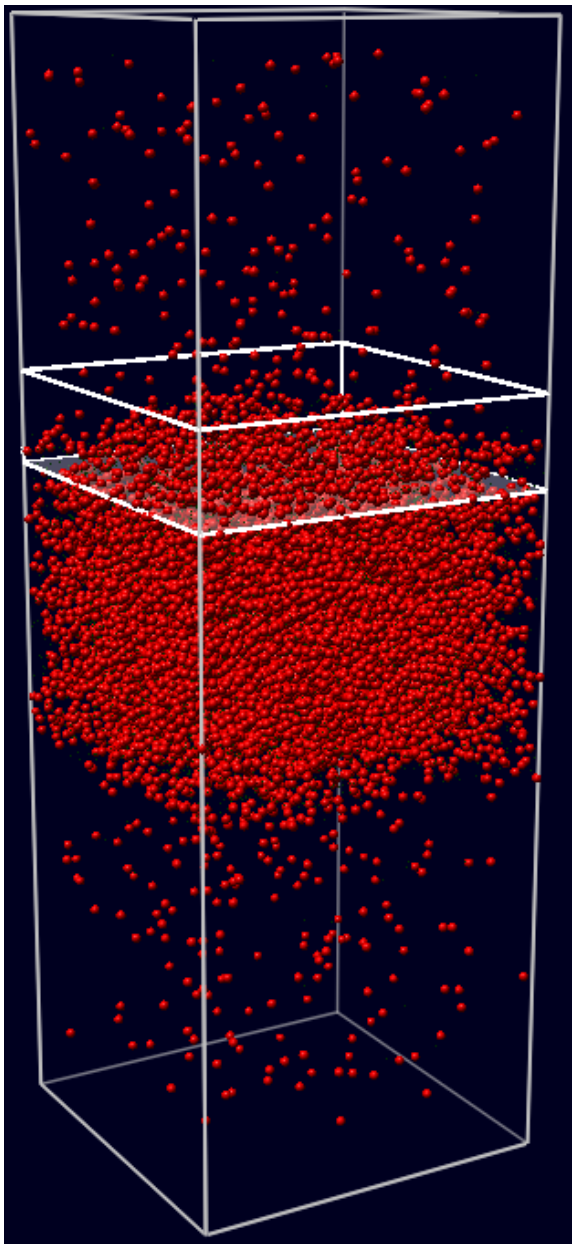


Abbildung 4.3.: EvaporationRenderer in einem instanziierten View.

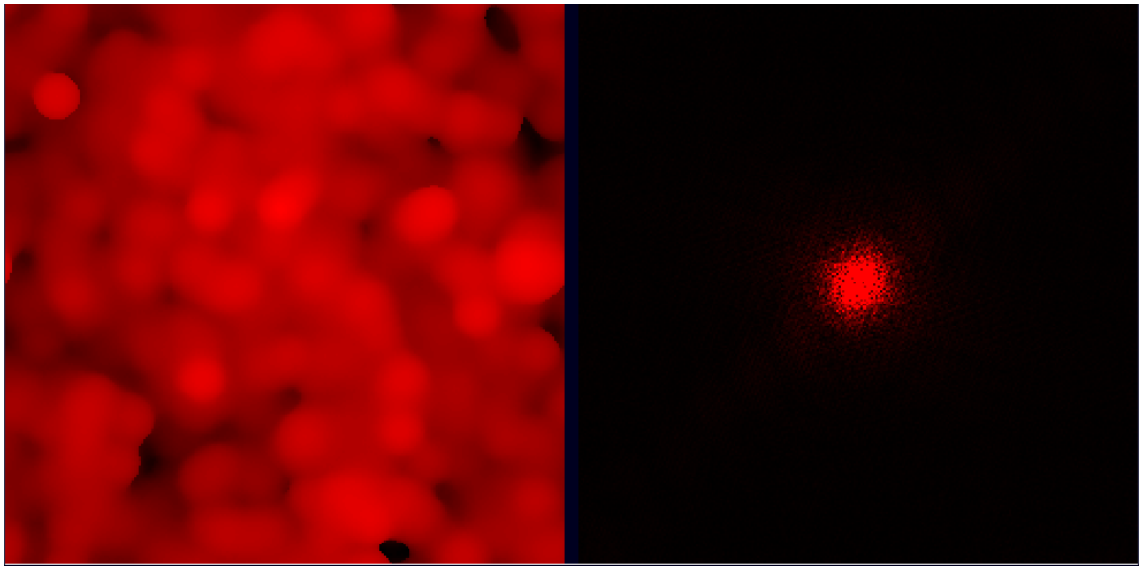
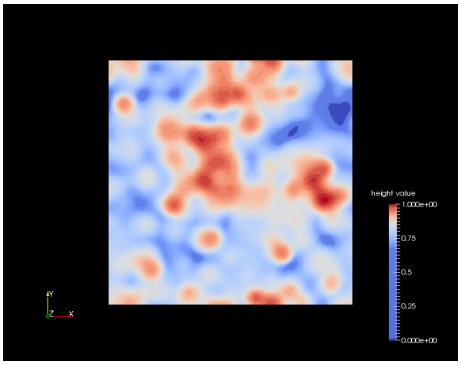


Abbildung 4.4.: EvaporationDebugRenderer in einem instanziierten View, welcher das Höhenfeld sowie das daraus resultierende Frequenzspektrum aus dem Datensatz *leichtrot.mmspd* zeigt; $f = 0$, $\sigma^2 = 1$, $r = 0.5$, $\Delta = 13$, $\epsilon = 2$, $\delta = 9$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 256$.

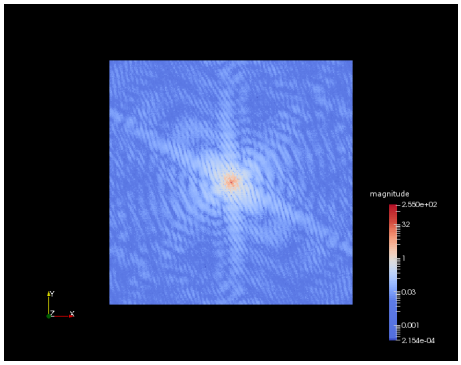
die vorgefilterten Partikel sowie „getfrequency“ und „getheightmap“ um die Frequenz bzw. das Höhenfeld jeweils als zweidimensionales Bild zu erhalten. Neben dem Rendern der Nullebene, welche als Bezug für das Höhenfeld genommen wird, rendert dieses Modul ebenfalls die Filterbox (siehe Abb. 4.3), welche beide durch den Nutzer über Parameter (siehe Abb. 4.6) in der genannten AntTweakBar definiert werden. Als Filterbox wird hierbei der Bereich bezeichnet, in dem die vorgefilterten Partikel liegen. Zusätzlich zur Normalen, der Entfernung zum Ursprung des Weltkoordinatensystems und der Tiefe der Filterbox kann der Nutzer auch zwischen verschiedenen Modi - manuell und automatisch - wählen. Bei letzterem Modus werden die Schnittpunkte der Nullebene mit den Kanten der *Object Bounding Box* automatisch berechnet, während der manuelle Modus eine feinere Lokalisierung der Nullebene samt Filterbox erlaubt. Weitere Modi können einfach hinzugefügt werden, z. B. um die Nullebene an die Daten anzupassen oder nicht nur auf Rechtecke zu beschränken.

4.2.2. EvaporationDebugRenderer

Das Modul EvaporationDebugRenderer (4.2b) ist hauptsächlich entstanden, um die einzelnen Schritte des Plug-Ins besser nachverfolgen zu können. Das Modul kann wie EvaporationRenderer und in Abb. 4.4 dargestellt in einem weiteren View instanziiert werden, um das Höhenfeld sowie die Frequenzanalyse auf jeweils einem Quad als Textur zu rendern. Dazu erfolgt ein Renderaufruf eines Moduls, z. B. View2D, über den Callee Slot „rendering“. Die dafür notwendigen Daten werden über die Caller Slots „getheightmap“ und „getfrequency“ von den Modulen SurfaceReconstruction bzw. FrequencyAnalysis über einen Image2DCall zur Verfügung gestellt. Das Höhenfeld zeigt die Höhe der ersten Isowerte von der Höhe der Filterbox senkrecht auf die Nullebene, welche größer oder gleich einem festgelegten Parameter sind, während das Frequenzbild eine FFT zeigt, trans-



(a) Höhenfeld als Eingabedaten…



(b) und die zugehörige Fouriertransformation, mit dem Durschnitt der Farbwerte des Höhenfeldes als höchsten Wert in der Mitte

Abbildung 4.5.: Höhenfeld und daraus transformiertes Frequenzbild; $f = 25$, $\sigma^2 = 1.5$, $r = 0.5$, $\Delta = 10$, $\epsilon = 3$, $\delta = 12$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 256$.

formiert aus dem Höhenfeld. Die beiden Module, welche das Höhenfeld bzw. die Frequenzanalyse bereitstellen, werden nun genauer erläutert.

4.2.3. FrequencyAnalysis

FrequencyAnalysis (4.2d), ein weiteres Modul, erstellt ein Frequenzbild aus einem über den Caller Slot „getheightmap“ erhaltenen Höhenfeld, indem die Fourier-Transformation auf dieses angewendet wird. Die Ausgabe kann über den Callee Slot „getfrequency“ an andere Module über einen Image2DCall weitergegeben werden. Die in Abschnitt 2.2 vorgestellte DFT bzw. die performantere Variante als FFT sind in dieser Klasse umgesetzt, dabei kann entweder die selbst implementierte 2D DFT zur Berechnung genutzt oder auf die von Paul Bourke [Bou] umgesetzte FFT zurückgegriffen werden. Außerdem wird anschließend das Frequenzbild logarithmisch skaliert, um die vielen Pixel mit unterschiedlichem Höhenwert sichtbar zu machen, die sonst schwarz wären. Abb. 4.5 zeigt ein Frequenzbild, entstanden aus einem Höhenbild. Die Höhe und Breite des Höhenbildes, auf dem diese Transformation in den Frequenzbereich durchgeführt wird, und der Ausgabe werden durch globale Parameter bestimmt, welche Grundlage zur Darstellung in vielen Modulen sind.

4.2.4. SurfaceReconstruction

Da inzwischen mehrere Parameter eingeführt wurden und weitere hinzukommen, gibt Tabelle 4.1 eine Übersicht über die in dieser Arbeit verwendeten Parameter.

Um das Höhenfeld bestimmen zu können, muss, wie bereits erklärt, vorhergehend die Oberfläche rekonstruiert werden. Die implizite Rekonstruktion unter Verwendung von radial basis functions sowie die anschließende Extraktion des Höhenfeldes wird in dem Modul SurfaceReconstruction (4.2e) implementiert. Dazu wird für jeden von dem Modul Baseline vorgefilterten und über den Caller Slot „getbaseline“ eingehenden Partikel dessen Einflussgebiet unter Zuhilfenahme einer Gauß-/Normalverteilung als Kubus approximiert, wobei die Varianz σ^2 abhängig vom Radius der Partikel ist und zusätzlich beliebig skaliert werden kann. Konkret werden dabei die Gitterpunkte

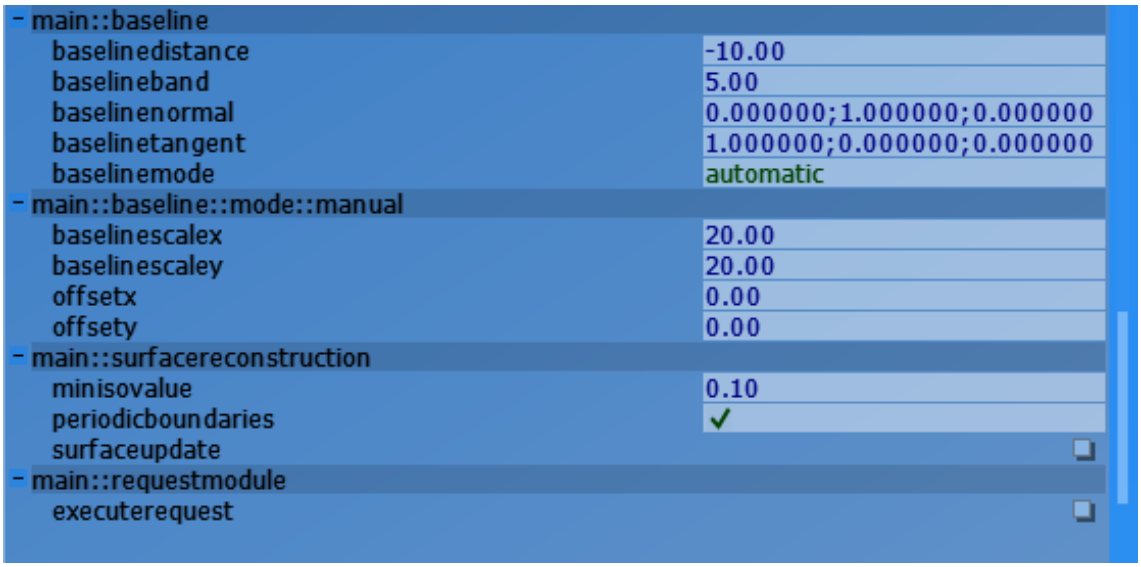
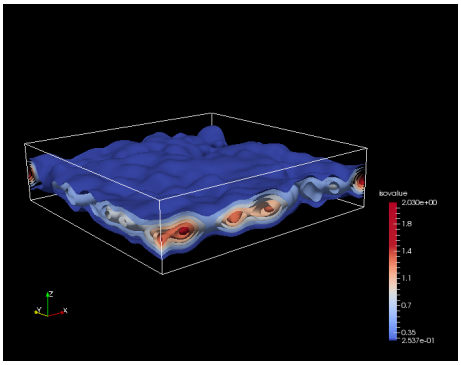


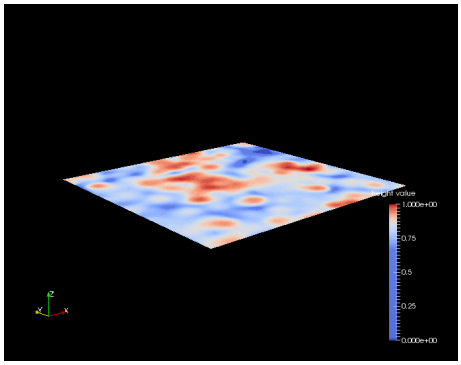
Abbildung 4.6.: Parameter der verschiedenen Module des Plug-Ins in der AntTweakBar [Ant]

Parameter	Name	Beschreibung
f	Frames	Frames der Simulation, für welche die Ausgabe berechnet wurde
σ^2	Varianz	Varianz der Gaußverteilung, bestimmt Breite der Dichtefunktion und damit den Einflussbereich der Partikel
r	Radius	Radius der Partikel
Δ	Volumenhöhe	Höhe der Filterbox und des Volumengitters, welches bei der Oberflächenrekonstruktion eingesetzt wird
ϵ	Cutoff	Grenze der Filterbox, innerhalb derer Partikel nicht berücksichtigt werden
δ	Abstand der Nullebene	Abstand der Nullebene vom Mittelpunkt der Object g Box
$\vec{n}$	Normale	Normale der Nullebene
ρ	Auflösung in Normalenrichtung	Beschreibt die Auflösung des Voxelgitters in der Richtung der Normalen, d. h. wie groß der Abstand zweier Gitterpunkte in diese Richtung ist
ϕ	Minimum Isowert	Isowert, ab dem ein Gitterpunkt in das Höhenfeld einfließt
nx	Breite der Bilder	Die Breite des Voxelgitters sowie des Höhen- und des Frequenzbildes
ny	Höhe der Bilder	Die Höhe des Voxelgitters sowie des Höhen- und des Frequenzbildes

Tabelle 4.1.: Übersicht der Parameter



(a) Volumenrendering aus Oberflächenrekonstruktion mit Isoflächen. Gut zu sehen ist, dass Partikel oben und unten in der Filterbox nicht berücksichtigt werden, da diese eine Schnittfläche mit der Box bilden würden…



(b) und das durch Raycasting extrahierte Höhenfeld, welches die Oberfläche des Volumens widergibt.

Abbildung 4.7.: Isoflächen und extrahiertes Höhenfeld; $f = 25$, $\sigma^2 = 1.5$, $r = 0.5$, $\Delta = 10$, $\epsilon = 3$, $\delta = 12$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 256$.

der Filterbox berücksichtigt, welche in das Intervall $\pm 3\sigma$ in allen drei Dimensionen um den Partikel fallen. Außerdem werden nur solche Punkte einbezogen, welche innerhalb der Filterbox liegen, da diese nicht unbedingt alle vorgefilterten Partikel enthalten muss. Die genauen Gitterpunkte können über Skalarprodukte und die *pixel spacings* berechnet werden. Das pixel spacing ergibt sich aus der Länge des Vektors der Nullebene zwischen zwei Kanten der Object Bounding Box, geteilt durch die entsprechende Auflösung. Über diese Kernelapproximation wird iteriert und die euklidische Distanz der Gitterpunkte in Weltkoordinaten zum Partikel berechnet. Abhängig davon wird der Wert der Dichtefunktion auf den jeweiligen Gitterpunkt addiert. Das Ergebnis sind Isoflächen innerhalb der Filterbox, mit den gewählten Auflösungen. Bei der richtigen Wahl der Parameter (hauptsächlich der Varianz σ^2 und der Lage der Filterbox) sind die Isoflächen glatt und überall vorhanden. Um keine Frequenzen durch abgeschnittene Partikelvolumina an der Filterbox zu erzeugen, sollte der Cutoff ϵ entsprechend gewählt werden. Da die Simulationen, aus denen die zugrundeliegenden Daten hervorgegangen sind, periodisch in x- und y-Richtung fortgeführt werden, können in den Parametern (4.6) periodische Randbedingungen aktiviert werden. Dies bietet sich an, wenn die Nullebene orthogonal zur Object Bounding Box bzw. zur Simulation ausgerichtet ist und schlägt sich nieder im berechneten Volumen samt Isoflächen und damit auch im Höhenfeld sowie in der vom Modul FrequencyAnalysis durchgeführten Fourier-Transformation.

Aus dem Voxelgitter der Filterbox, d. h. den Isoflächen, muss nun noch ein zweidimensionales Höhenfeld extrahiert werden. Dies geschieht in diesem Fall durch Raycasting von den höchsten Punkten der Filterbox senkrecht auf die Nullebene, wobei für das Höhenfeld der Wert genommen wird, welcher zuerst einen durch einen Parameter definierten minimalen Isowert überschreitet. Wie in Kapitel 5 gezeigt, kann die Grenzschicht bzw. daraus ein Höhenfeld gut mit diesem Algorithmus extrahiert werden. Abb. 4.7a und 4.7b zeigen das Zwischenergebnis und das beschriebene Höhenfeld.

Das Ergebnis, ein zweidimensionales Höhenfeld, wird skaliert auf den Wertebereich $[0, 1]$ über den Callee Slot „getheightmap“ anderen Modulen bereitgestellt.

4.2.5. Baseline

Viele der bereits genannten Module nutzen für die jeweiligen Berechnungen die Nullebene oder die oft erwähnten vorgefilterten Partikel. Die Nullebene samt Filterbox wird in dem Modul Baseline (4.2c) sowohl über Parameter definiert (siehe Abb. 4.6) als auch berechnet sowie die Partikel danach gefiltert. Mit dem Call MultiParticleDataCall aus dem MegaMol-Core werden Partikeldaten übergeben, weshalb BaselineCall von dieser Klasse erbt und somit auch Partikeldaten transportieren kann und über den Callee Slot „getbaselinedata“ die gefilterten Partikel für andere Module zugänglich macht. Die Daten zum Filtern kommen über einen MultiParticleDataCall am Caller Slot „getdate“ von einem anderen Modul, z. B. dem MMSPDDataLoader als Quelle. Die zu den Partikeln gehörenden Attribute, wie Farbwerte, Radius, etc. werden - soweit vorhanden - entsprechend übernommen. Um nicht unnötig Rechenzeit zu verschwenden wird in diesem wie auch in den anderen Modulen nur bei einem Update der eingehenden Daten bzw. dahinter stehender Module die Berechnung, in diesem Fall die Filterung, auf Basis der neuen Daten wiederholt.

Weiterhin wird in dem Modul Baseline neben der Filterung auch die Nullebene sowie die Filterbox, bzw. genauer die Eckpunkte davon, berechnet. So setzt dieses Modul über den Callee Slot „getbaselinedata“ diverse Variablen des Calls BaselineCall, wie z. B. die Normale der Nullebene für dessen Ausrichtung oder die Eckpunkte derselben. Die Tangente und die Bitangente der Oberfläche werden abhängig von der Normalen berechnet, sofern für die Tangente kein möglicher Wert gesetzt wird, d. h. $< Tangente, Bitangente >\neq 0$ ist. Des Weiteren wurde in Abschnitt 4.2.1 die Möglichkeit erwähnt, zwischen verschiedenen Modi zu wählen, nach welchen die Nullebene und die Filterbox gerendert werden sollen. Für die vorhandenen Modi, automatisch und manuell, werden in diesem Modul auch die benötigten Eckpunkte automatisch berechnet oder wie vom Nutzer festgelegt ermittelt. Dabei werden die in einer Liste gehaltenen Eckpunkte gleichzeitig sortiert, damit es beim Rendern und Berechnen von Richtungen nicht zu Problemen kommt.

Im Modus 'automatisch' werden die Schnittpunkte mit der durch - über Parameter definierten - Normale und Abstand zum Ursprung festgelegten Ebene berechnet, welche die Eckpunkte der Nullebene als Bezug für die Isoflächenberechnung und damit für das Höhenfeld bestimmen. Bei 'manuell' wird ein Offset in Richtung der Tangente und der Bitangente der Nullebene sowie eine Skalierung in beide Richtungen durch den Nutzer angegeben.

Um die Berechnung zu beschleunigen wurde bei der Rekonstruktion der Oberfläche (d. h. bei den for-Schleifen) auf OpenMP[2] gesetzt, da diese API unabhängig von spezieller Hardware läuft (abgesehen von Single-Core Prozessoren) und das Programm auch noch lauffähig ist, wenn OpenMP auf dem System nicht verfügbar ist bzw. ohne OpenMP kompiliert wird.

4.2.6. RequestModule

Das letzte Modul, RequestModule (4.2f), ist entstanden, um Anfragen an andere Module zu stellen und damit Ergebnisse und vor allem Zwischenergebnisse abfragen und exportieren zu können. Wie die anderen Module erbt es von der im MegaMol-Core bereitgestellten Klasse *Module*. Für diese Arbeit wurden daher über die Caller Slots „getheightmap“ und „getfrequency“ Daten angefordert, sobald der Request über einen Parameter (Abb. 4.6) getriggert wurde.

[2] http://openmp.org/wp/

Der Datenexport wird durch die Klasse *VtkExport* gesteuert und in eine .vtk-Datei geschrieben, welche wiederum mit ParaView [Par] eingelesen und gerendert werden kann. Weiterhin werden diverse Hilfsfunktionen durch die Klasse *Util* implementiert, welche z. B. das Schreiben in und Auslesen von dreidimensionalen Arrays, die in einem Block allokiert wurden, erleichtern.

4.2.7. Image2DCall

Um Daten zwischen Modulen auszutauschen wurden bereits Calls erwähnt. Einer dieser Calls, welche durch dieses Plug-In implementiert werden, heißt Image2DCall (4.2h). Dieser transportiert Daten in Form von Float-Arrays, welche für zweidimensionale Bilder genutzt werden.

4.2.8. BaselineCall

Der andere Call, BaselineCall, erbt wie bereits erwähnt von der Klasse MultiParticleDataCall, ebenfalls aus dem Core. Somit kann dieser Call neben den Details zur Nullebene und der Filterbox auch Partikeldaten, und damit auch gefilterte Partikel, weitergeben. Die Calls bestehen hauptsächlich aus *Gettern* und *Settern*.

Das Plug-In wird im SVN Repository[3] des Visualisierungsinstituts der Universität Stuttgart bereitgestellt.

[3] https://svn.vis.uni-stuttgart.de/projects/megamol/plugins/evaporation/

5. Ergebnisse

Auf die Implementierung folgend sollen im weiteren Verlauf der vorliegenden Arbeit die Ergebnisse zunächst vorgestellt und anschließend vergleichend gegenüber gestellt werden. Die in Listing A.1 gezeigte Projekt-Datei wurde - teils mit leichten Abänderungen der Parameter - als Konfiguration genutzt, um die in dieser Arbeit dargestellten Ergebnisse zu erhalten.

Zuerst werden im Folgenden Zwischenergebnisse und Endresultate des Prototypen gezeigt, bevor in diesem Kapitel darauf folgend verschiedene Datensätze miteinander verglichen werden.

Da die Grenzfläche untersucht werden soll, wird die Nullebene orthogonal zur Filterbox ausgerichtet, weshalb für alle hier vorgestellten Ergebnisse periodische Randbedingungen aktiv sind, während die Parameter sowie der Datensatz variieren.

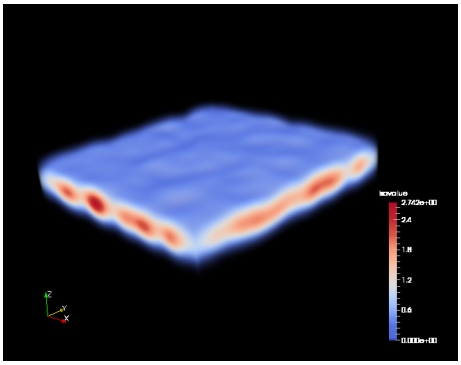
5.1. Ergebnisse

Wie in Kapitel 4 beschrieben, können die diversen Berechnungen angepasst werden, sei es die Höhe der Filterbox oder die Breite der Gauß-Dichtefunktion bei der Oberflächenrekonstruktion. Bei den Ergebnissen werden diese Parameter - mit Bezug auf Tabelle 4.1 - angegeben.

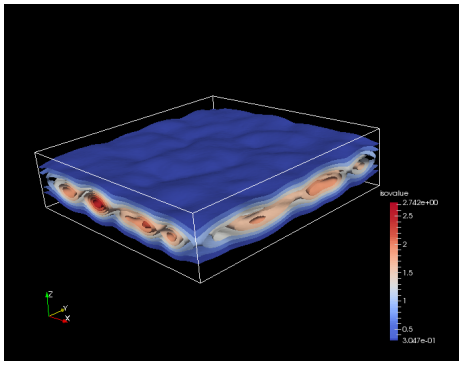
Das für die Extraktion des Höhenfeldes benötigte und bei der Oberflächenrekonstruktion entstehende Volumen der Isowerte zeigt beispielhaft Abb. 5.1a. Deutlich erkennbar sind die Gaußglocken über den Partikeln, welche nach außen stark abflachen bzw. an Wert verlieren. Aus diesem Volumen ist Abb. 5.1b entstanden, indem bestimmte Isowerte mit derselben Farbe eingefärbt wurden. Reduziert man die Isoflächen auf die äußerste bzw. auf die mit dem kleinsten Isowert, so erhält man in diesem Beispiel eine gute Repräsentation der Oberfläche, wie sie in Abb. 5.1c dargestellt wird. Das extrahierte Höhenfeld (Abb. 5.1d) spiegelt direkt die Höhenwerte wider, wie in dem Vergleich gut zu sehen ist. Direkt aus diesem Höhenfeld ist Abb. 5.1e entstanden, die Frequenzverteilung. Diese wurde nach der Fourier-Transformation logarithmisch skaliert, um auch die kleineren Werte sichtbar zu machen. Ein Beispiel, wie diese Frequenzverteilung für mehrere Zeitschritte über die Zeit gestackt aussieht, zeigt Abb. 5.2a, während Abb. 5.2b diese als Volumen gerenderte Frequenzverteilung von der Seite zeigt. Gut sichtbar sind die über Zeit (von rechts nach links) kleiner werdenden Frequenzen.

5.2. Vergleich der Ergebnisse

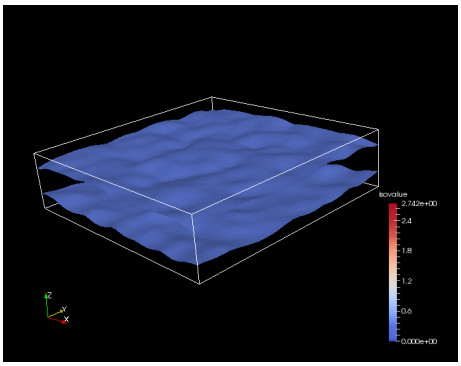
Wie bereits erwähnt, werden für sämtliche Ergebnisse die Parameter aus Tabelle 4.1 aufgeführt, sodass diese reproduzierbar sind. Darunter fallen die Frames, für welche das Ergebnis berechnet wurde sowie die Varianz σ^2 innerhalb der Gauß-/Normalverteilung, wovon die Breite der Dichtefunktion abhängig ist. Der Radius r ist bei den Partikeln in allen Datensätzen 0.5, während die Volumenhöhe Δ, der Abstand der Nullebene vom Mittelpunkt der Object Bounding Box δ sowie



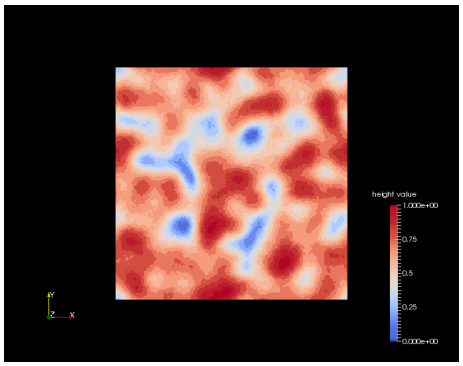
(a) Berechnetes Volumen mit dargestellten Isowerten.



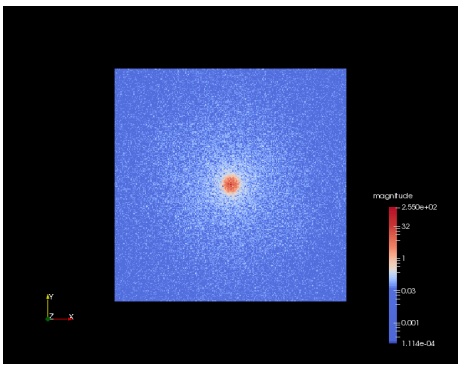
(b) Isoflächendarstellung des Volumens.



(c) Potenzielle rekonstruierte Oberfläche aus Isoflächen.

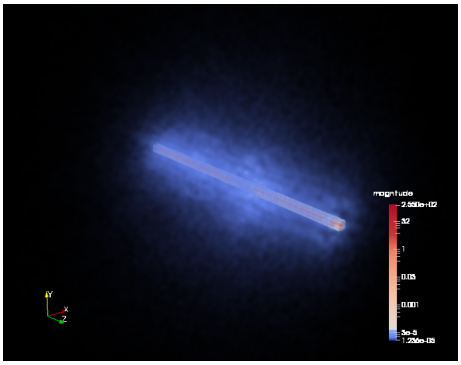


(d) Höhenfeld der Oberfläche.

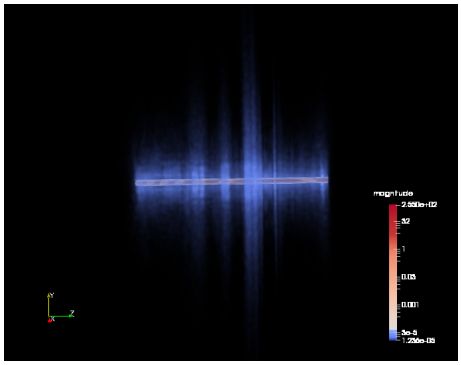


(e) Frequenz-Analyse des Höhenfeldes.

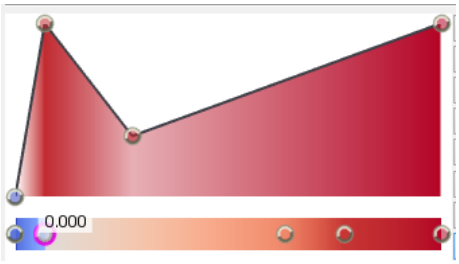
Abbildung 5.1.: Zwischenschritte und Ergebnisse des Prototypen aus dem Datensatz *leichtrot.mmspd*. $f = 0, \sigma^2 = 2.5, r = 0.5, \Delta = 10, \epsilon = 4, \delta = -5, \vec{n} = (0, 1, 0)^T, \rho = 0.1, \phi = 0.1, nx = ny = 256$.



(a) Gestackte Frequenzverteilungen, das verrauschte Bild deutet auf viele Frequenzen hin.



(b) Gestackte Frequenzverteilungen in der Seitenansicht, wobei manche Frame deutlich mehr und höhere Frequenzen aufweisen.



(c) Die für das Volumenrendering der Frequenzverteilungen genutzte Transferfunktion.

Abbildung 5.2.: Weitere Zwischenschritte und Ergebnisse des Prototypen aus demselben Datensatz. $f = 0 - 99$, $\sigma^2 = 2.5$, $r = 0.5$, $\Delta = 10$, $\epsilon = 4$, $\delta = 12$, $\vec{n} = (0, 1, 0)^T$, $\rho = 1$, $\phi = 0.1$, $nx = ny = 256$.

der Cutoff ϵ vom Nutzer festgelegt werden können. Weiter bestimmt dieser die Normale $\vec{n}$ sowie die Auflösung des Voxelgitters in Richtung dieser Normalen ρ. Über den minimalen Isowert ϕ wird gesteuert, welche Gitterpunkte in das Höhenfeld beim Raycasting miteinfließen. nx bzw. ny beschreiben die Breite und Höhe der Bilder.

Zum Testen des Prototypen und um Ergebnisse zu erhalten wurden zwei Datensätze mit 25GB und 22,7GB bereitgestellt, wobei 1 Frame 50 Zeitschritten in der Simulation entspricht. Einer davon, *t0-80_surf.mmspd*, zeigt eine Simulation von 1 Million Teilchen bei einer Temperatur von 0.8, wobei sich diese wie alle weiteren aufgeführten Temperaturen auf eine einheitliche Größe beziehen, wie in [VKFH06] beschrieben. Wie dieser besteht der andere Datensatz *t0-95_surf.mmspd* ebenso aus demselben, physikalisch modellierten Reinstoff mit 1 Million Teilchen, allerdings bei einer Temperatur von 0.95. Dieser Reinstoff ist von ca. 0.6 bis 1.08 flüssig, weshalb der Datensatz mit der höheren Temperatur auch höhere Frequenzen an der Grenzschicht aufweist, da dieser dort ähnliche Eigenschaften wie in Gasform besitzt.

Abschnitt 5.2 zeigt den Datensatz, welcher den Reinstoff mit der niedrigeren Temperatur enthält, wobei aus Platzgründen in der Mitte Partikel ausgeschnitten wurden, welche für die Analyse der Grenzschicht unwichtig sind.

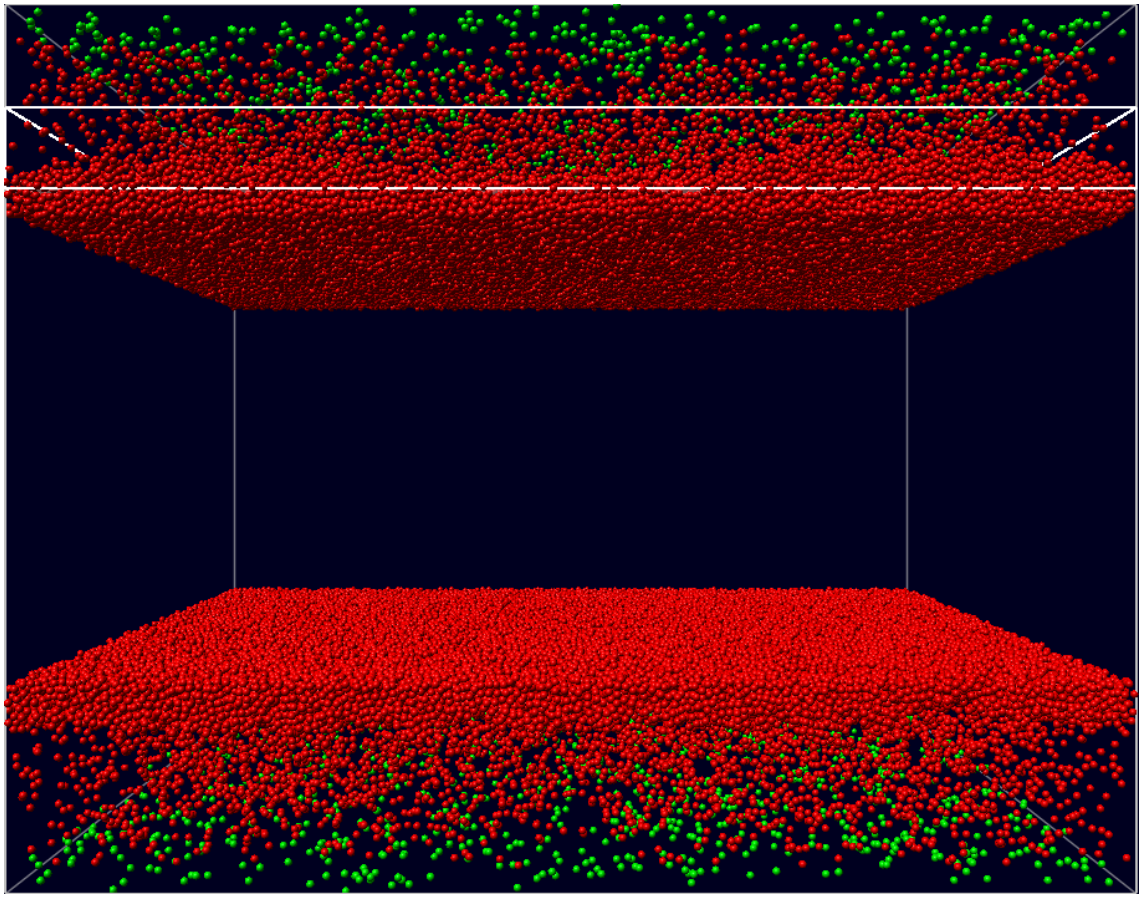


Abbildung 5.3.: Der Datensatz, welcher 1 Million Teilchen des Reinstoffs bei niedriger Temperatur zeigt, die unrelevanten Partikel in der Mitte wurden weggelassen.

Bei diesem Datensatz wird deutlich, welchen Einfluss die Parameter σ^2 und ϕ nehmen, wenn man die Frequenzspektren in Abschnitt 5.2 vergleicht. Mit zunehmender Varianz σ^2 und damit größerem Gauß-Kernel wird das Frequenzspektrum rauschfreier, da die Oberfläche glatter wird. Bei kleinem minimalen Isowert ϕ werden von der Nullebene weiter entfernte Gitterpunkte bei der Extraktion des Höhenfeldes berücksichtigt, wodurch herausstehende Kanten und damit zusätzliche, künstliche niedrige Frequenzen entstehen, welche durch größere Wellen im Frequenzspektrum in Erscheinung treten. Daher verschwinden diese niedrigeren Frequenzen mit zunehmendem ϕ, sodass hohe Frequenzen das Spektrum prägen, was wiederum als Rauschen wahrgenommen wird.

Ähnlich lassen sich die Höhenfelder wie in Abschnitt 5.2 vergleichen. Je größer ϕ, umso tiefere Gitterpunkte werden bei der Extraktion des Höhenfeldes berücksichtigt, da darüber nur Isoflächen mit niedrigen Werten liegen. Dies führt zu weiteren Frequenzen, welche die eigentliche Grenzschicht und damit die Analyse verfälschen. Dies zeigt noch einmal deutlich, wie wichtig eine gute Abstimmung der Parameter ist, da auch die Gauß-Dichtefunktion mit zunehmender Varianz σ^2 flacher wird und dementsprechend ϕ angepasst werden muss.

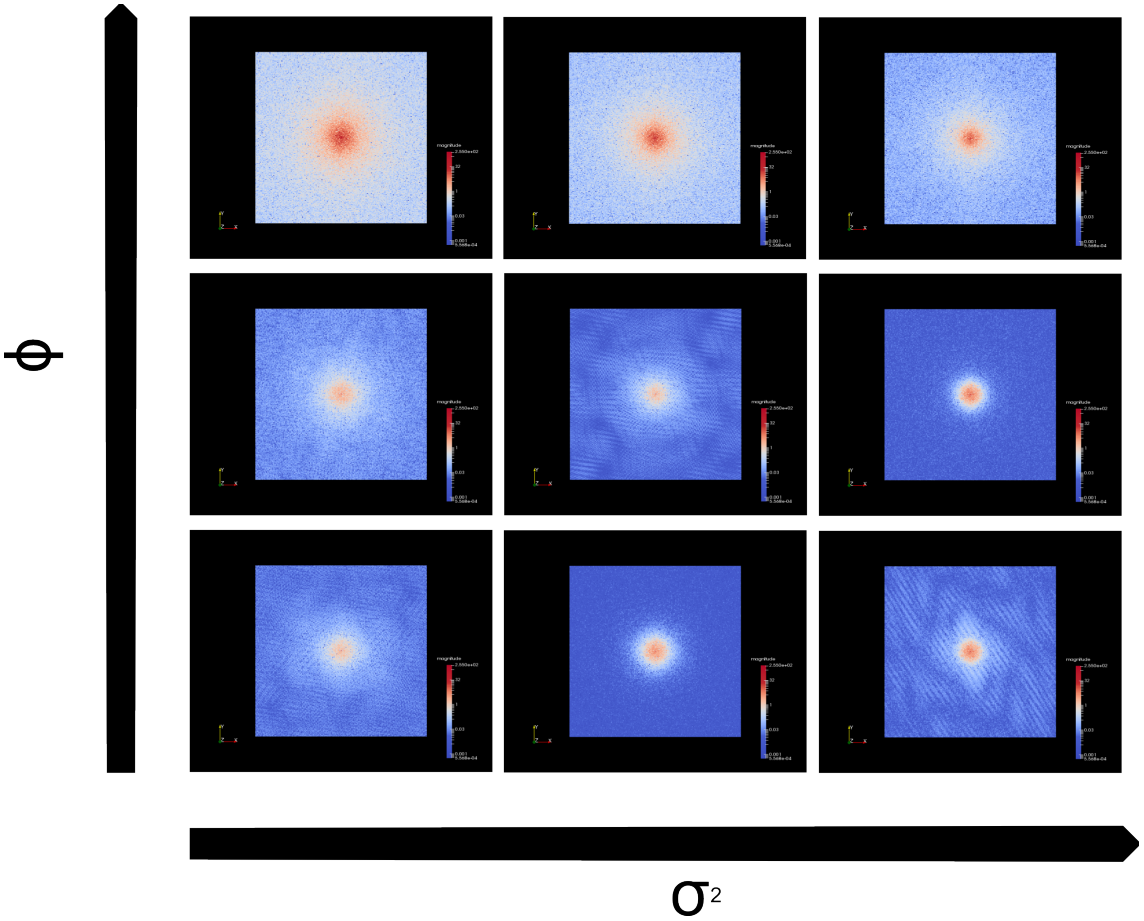


Abbildung 5.4.: 7, 10 und 15 für σ^2 bzw. 0.05, 0.1 und 0.5 für ϕ liefern unterschiedliche Ergebnisse mit eindeutigen Richtungen, die restlichen Parameter sind: $f = 0$, $r = 0.5$, $\Delta = 20$, $\epsilon = 7$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $nx = ny = 256$.

Obwohl die Transferfunktionen teilweise bei den Höhenfeldern mit einem höheren Wert für σ^2 den Eindruck machen, kleinere Volumen zu erzeugen, so nehmen sie stark zu, wie die obere Reihe deutlich macht.

Um beide Datensätze direkt gegenüber zu stellen, sollen die Oberflächen und die Frequenzanalysen miteinander verglichen werden. Abb. 5.6 zeigt die noch deutlich vorhandene Struktur bei Frame 1304 der Teilchensimulation bei einer Temperatur von 0.8.

Noch deutlicher wird dies, wenn man nur einzelne Isoflächen rendert, wie in Abb. 5.7 dargestellt. Von der äußersten Isofläche sollte man sich nicht beirren lassen, da es bei der Extraktion des Höhenfeldes darauf ankommt, wie der minimale Isowert ϕ gewählt wird, wie Abb. 5.8 verdeutlicht. So weist das Höhenfeld des Datensatzes bei höherer Temperatur bei geeigneter Wahl von ϕ deutlich größere Differenzen auf, was in höheren Frequenzen im Frequenzspektrum (Abb. 5.9) resultiert. Dagegen sind in der Frequenzanalyse des Reinstoffs bei niedrigerer Temperatur weniger Rauschen und deutlichere Strukturen zu sehen, was wiederum auf niedrigere Frequenzen hinweist.

Die Frequenzverteilungen gestackt und als Volumen gerendert zeigt Abb. 5.10 für alle Frames von dem Datensatz mit der niedrigeren Temperatur. Dabei sind einzelne Frames mit hohen

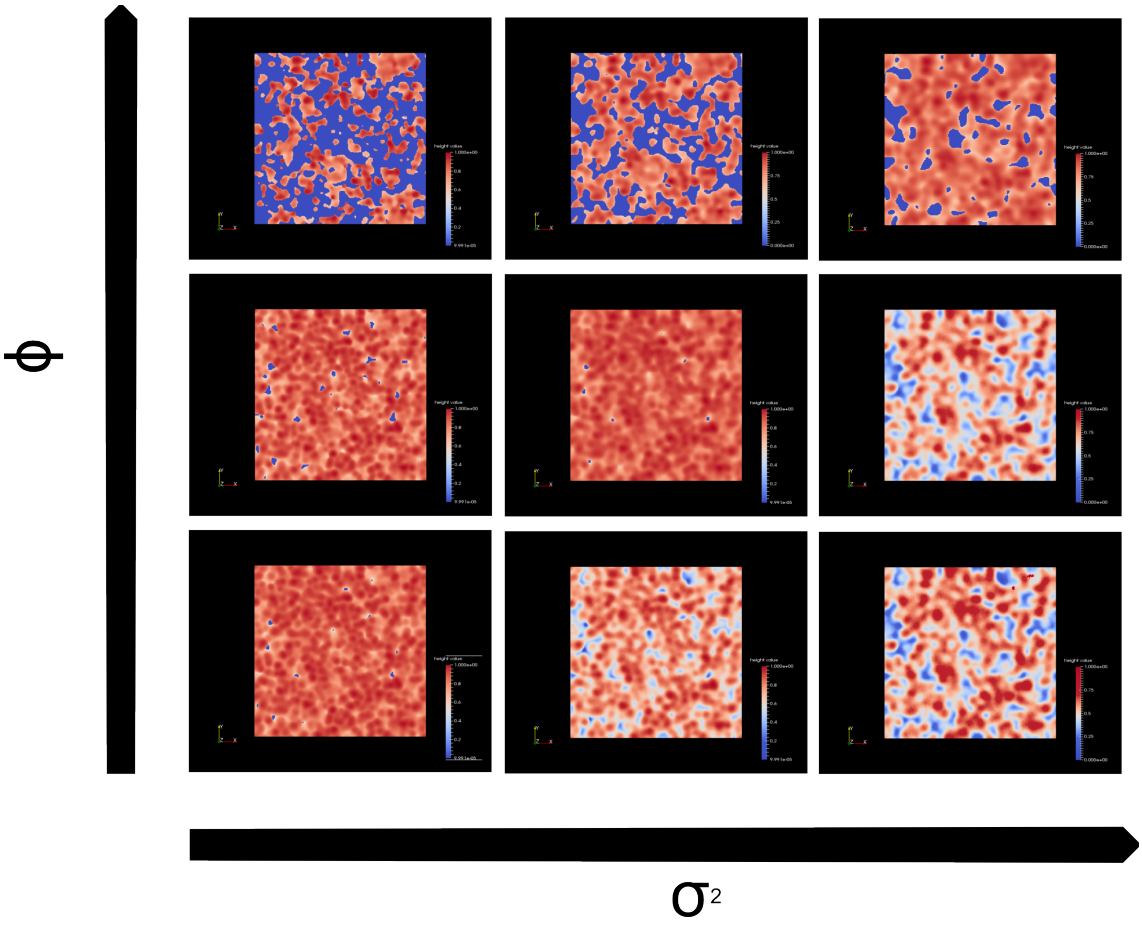
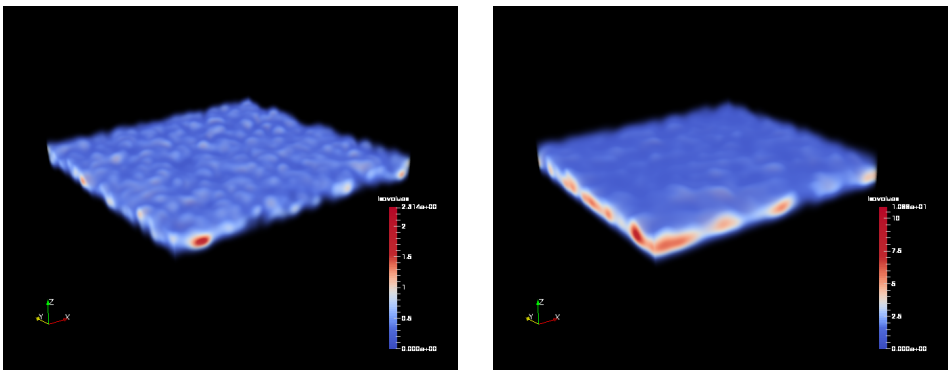


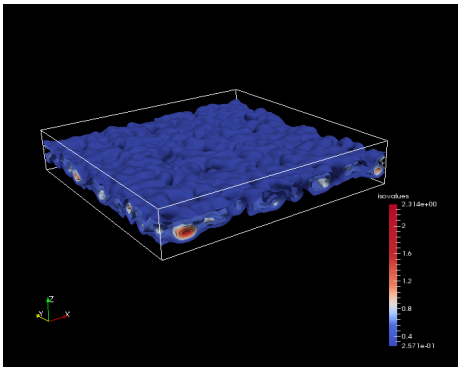
Abbildung 5.5.: 7, 10 und 15 für σ^2 bzw. 0.05, 0.1 und 0.5 für ϕ liefern unterschiedliche Ergebnisse, wobei Muster zu erkennen sind.



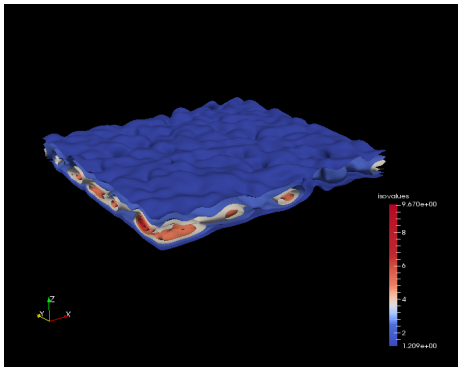
(a) Berechnetes Isovolumen der Partikel aus Datensatz 1 durch eine Gauß-Verteilung, bei Frame 1304 sind sie deutlich sichtbar…

(b) während sie in Datensatz 2 eine glattere Oberfläche bilden.

Abbildung 5.6.: Vergleich von Volumen der Partikel mit Isowerten aus unterschiedlichen Datensätzen, $f = 1304$, $\sigma^2 = 5$, $r = 0.5$, $\Delta = 25$, $\epsilon = 8$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 512$ und $\delta = 29$ für 5.6a bzw. $\delta = 69$ für 5.6b.

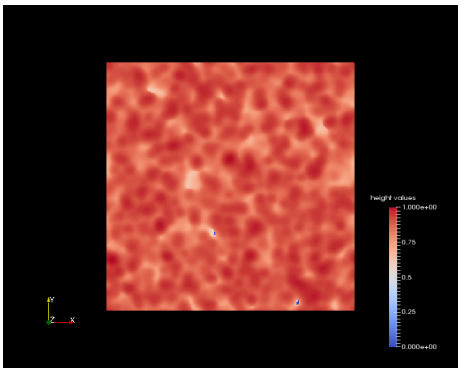


(a) Gerenderte Isoflächen geben die Struktur der betrachteten Partikel wieder.

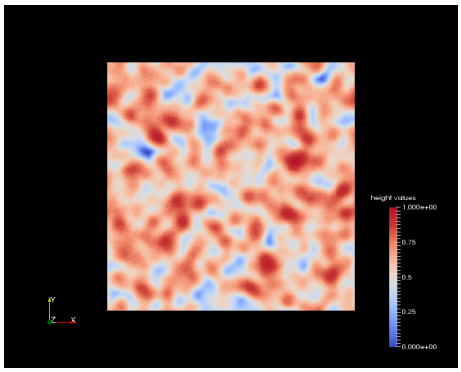


(b) Isoflächen aus dem zweiten Datensatz.

Abbildung 5.7.: Vergleich Isoflächen verschiedener Datensätze, $f = 1304$, $\sigma^2 = 5$, $r = 0.5$, $\Delta = 25$, $\epsilon = 8$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 512$ und $\delta = 29$ für 5.7a bzw. $\delta = 69$ für 5.7b.

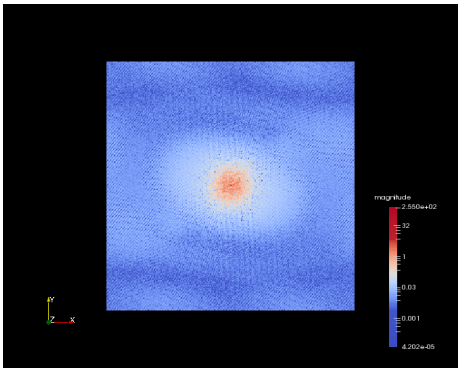


(a) Aus den durch implizite Oberflächenrekonstruktion erhaltenen Isoflächen extrahierte Höhenfelder. Beim Vergleich der beiden Höhenfelder fällt auf, dass…

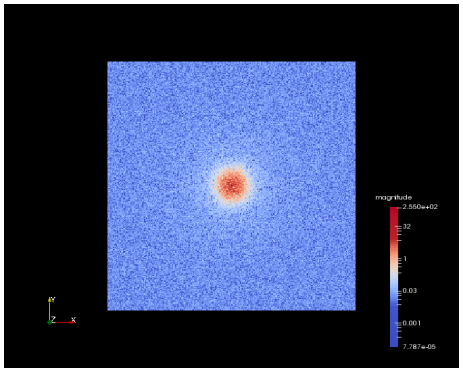


(b) bei höherer Temperatur deutlich größere Höhenunterschiede vorhanden sind.

Abbildung 5.8.: Vergleich von extrahierten Höhenfeldern, mit den Parametern $f = 1304, \sigma^2 = 5$, $r = 0.5$, $\Delta = 25$, $\epsilon = 8$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 512$ und $\delta = 29$ für 5.8a bzw. $\delta = 69$ für 5.8b.



(a) Frequenztransformation eines Höhenfeldes aus dem Datensatz mit den Teilchen bei niedriger Temperatur…



(b) und eines Höhenfeldes aus dem anderen Datensatz, mit deutlich höheren Frequenzen, erkennbar am Rauschen.

Abbildung 5.9.: Vergleich von Frequenzanalysen zweier Datensätze mit $f = 1304$, $\sigma^2 = 5$, $r = 0.5$, $\Delta = 25$, $\epsilon = 8$, $\vec{n} = (0, 1, 0)^T$, $\rho = 0.1$, $\phi = 0.1$, $nx = ny = 512$ und $\delta = 29$ für 5.9a bzw. $\delta = 69$ für 5.9b.

Frequenzen gut sichtbar genau wie eine Abnahme zur Mitte hin, d. h. dort ist die Oberfläche glatter. Die Frequenzen am Ende (links im Bild) weisen ein Rauschen auf, da hier die manuell eingegrenzte Grenzfläche zwar noch in der Filterbox liegt, diese aber nach unten gesunken ist, sodass weniger und eher lose Partikel mit in die Grenzfläche einfließen. Lose Partikel erzeugen größere Höhenunterschied, wodurch das Höhenfeld dahingehend beeinflusst wird, dass weitere Frequenzen hinzukommen.

Vergleicht man diese mit den als Space-Time Cube gerenderten Frequenzverteilungen aller Zeitschritte aus dem Datensatz *t0-95_surf.mmspd* (Abb. 5.11), so erkennt man generell deutlich mehr Rauschen. Dieses entsteht zum Teil auch durch die ungenaue Eingrenzung der Oberfläche. Ein weiterer Grund dafür ist, dass die Teilchen bei höherer Temperatur stärker bzw. schneller Verdampfen. Interessant sind zusätzlich die hellblauen Strukturen mit kleinem Radius, welche auf Grund der Veränderung des Höhenfeldes fluktuieren.

Für einen Frame aus dem Datensatz *t0-95_surf.mmspd* betrug die Berechnungszeit der Oberflächenrekonstruktion und der Extraktion eines Höhenfeldes auf ein 256x256 Bild 3,938s bei $\sigma^2 = 10$, $\phi = 0.1$ und $\rho = 1$. Bei einer Verdopplung der Auflösung auf 512x512 bei den sonst selben Parametern betrug die Dauer der Berechnung bereits 15,548s und bei 1024x124 62,594s, während bei einer noch passablen Auflösung von 128x128 gerade mal 1,074s benötigt wurden. Vergleicht man den Einfluss der Änderung der Auflösung in die Normalenrichtung ρ beispielsweise auf ein Zehntel, so ergibt sich mit 38,443s eine Verzehnfachung. Während für die Oberflächenrekonstruktion 54,516s bei $\sigma^2 = 2.5$, $\rho = 0.1$, einer Auflösung von 512x512 und $\phi = 0.1$ benötigt wurden, ergab derselbe Frame für $\phi = 0.01$ mit 54,063s kaum eine auf diese Parameteränderung zurückführbare Veränderung des Berechnungsaufwands. Als Frame wurde in allen in Tabelle 5.1 aufgeführten Szenarien der erste des Datensatzes *t0-95_surf.mmspd* genutzt, wobei die Partikel mit einer Anzahl von 55 751 immer dieselben waren, sodass ein guter Vergleich ermöglicht wird. Aus diesem Grund wurde der Cutoff ϵ auch immer mit 7 gewählt, da dieser lediglich die Anzahl der berücksichtigten Partikel beeinflusst. Alle Berechnungen wurden auf einer aktuellen Quad-Core CPU durchgeführt.

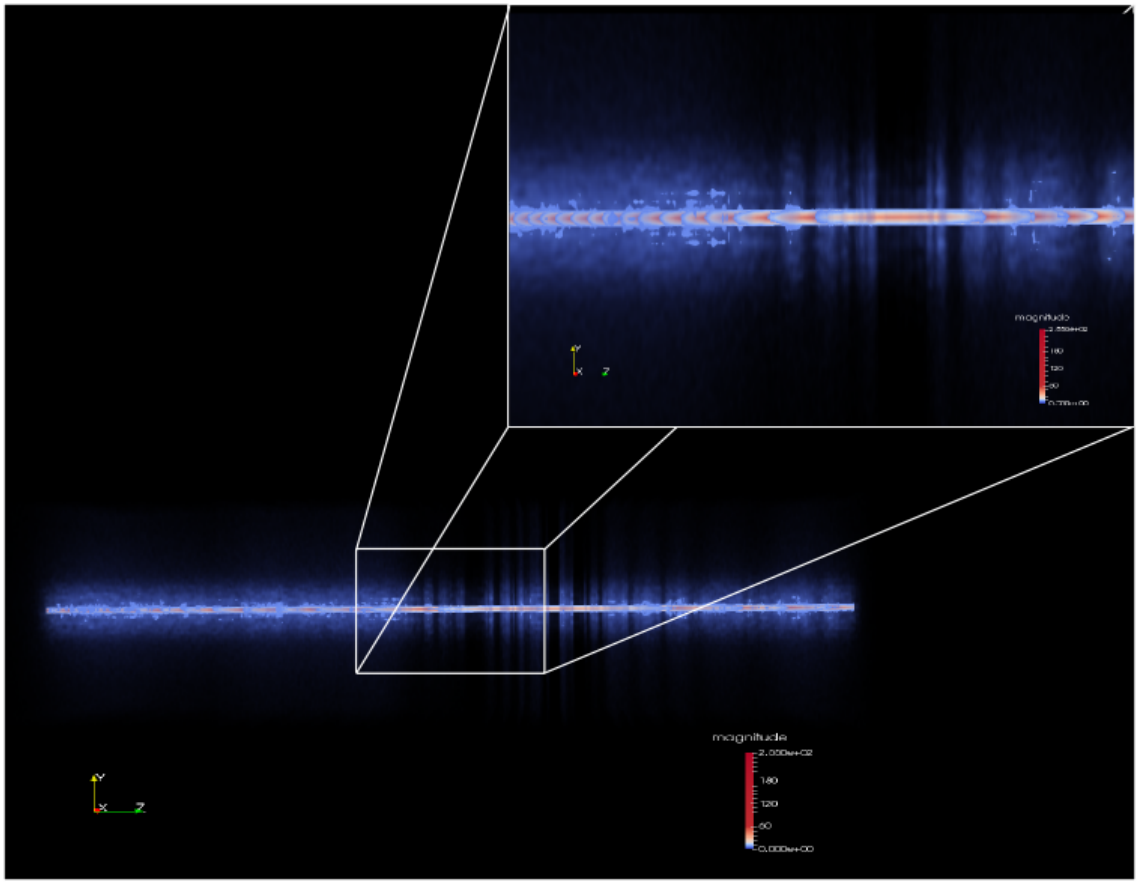


Abbildung 5.10.: Gestackte Frequenzverteilungen aller Zeitschritte des Datensatzes *t0-80_surf.mmspd*. Die Parameter sind $f = 0 - 4000$, $\sigma^2 = 10$, $r = 0.5$, $\Delta = 25$, $\epsilon = 10$, $\delta = 20$, $\vec{n} = (0, 1, 0)^T$, $\rho = 1$, $\phi = 0.1$ und $nx = ny = 128$.

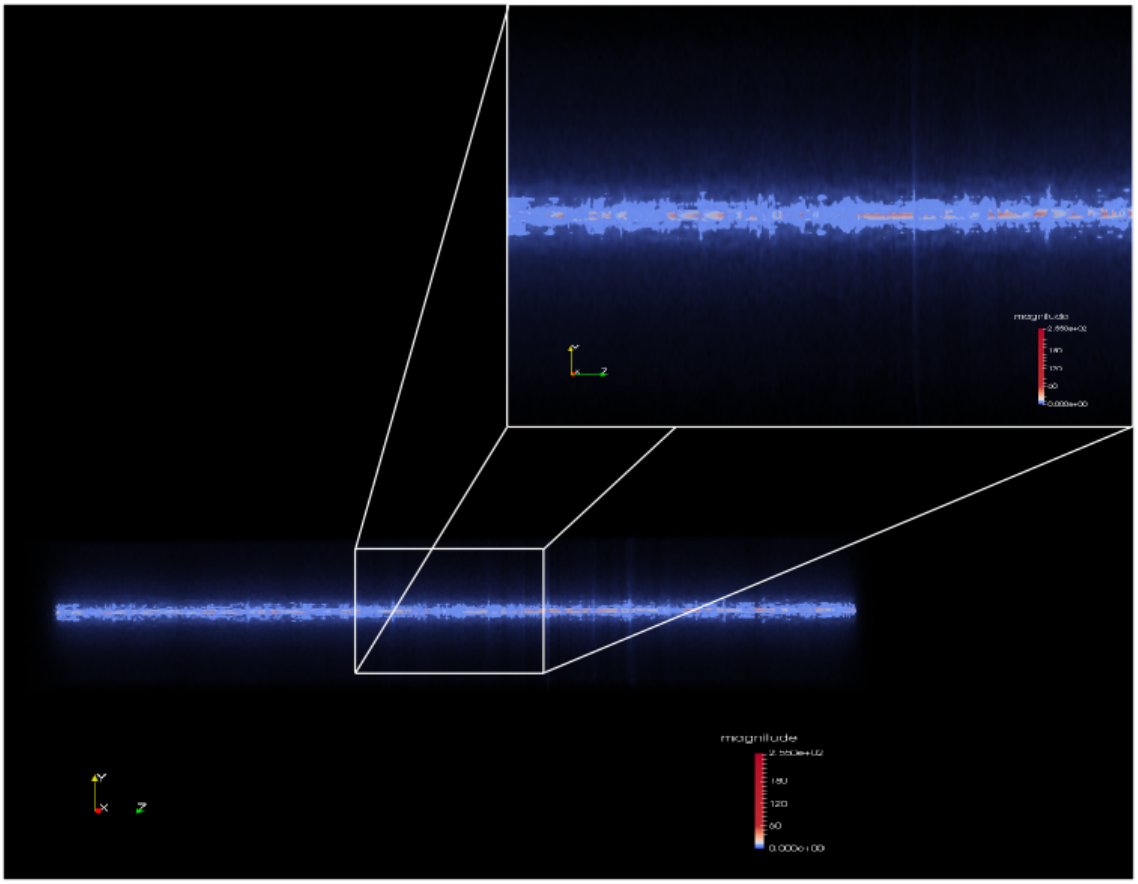


Abbildung 5.11.: Gestackte Frequenzverteilungen aller Zeitschritte des Datensatzes *t0-95_surf.mmspd*. Die Parameter sind $f = 0 - 4000$, $\sigma^2 = 10$, $r = 0.5$, $\Delta = 25$, $\epsilon = 10$, $\delta = 67$, $\vec{n} = (0, 1, 0)^T$, $\rho = 1$, $\phi = 0.1$ und $nx = ny = 128$.

Zeit in s	Anzahl der Partikel	σ^2	ϕ	ρ	nx, ny	ϵ
62,594	55 751	10	0,1	1	1024	7
15,548	55 751	10	0,1	1	512	7
53,86	55 751	2,5	1	0,1	512	7
54,063	55 751	2,5	0,01	0,1	512	7
54,516	55 751	2,5	0,1	0,1	512	7
15,509	55 751	5	1	1	512	7
15,67	55 751	5	0,1	1	512	7
155,051	55 751	5	0,1	0,1	512	7
13,769	55 751	2,5	0,1	0,1	256	7
39,32	55 751	5	0,1	0,1	256	7
7,591	55 751	10	0,1	0,5	256	7
38,443	55 751	10	0,1	0,1	256	7
37,487	55 751	10	0,5	0,1	256	7
3,938	55 751	10	0,1	1	256	7
7,567	55 751	10	0,5	0,5	256	7
1,074	55 751	10	0,1	1	128	7

Tabelle 5.1.: Berechnungszeiten für Frame $f = 0$ aus dem Datensatz *t0-95_surf.mmspd* mit $r = 0.5$, $\Delta = 20$, $\delta = 69$, $\vec{n} = (0, 1, 0 =^T)$ und .

6. Zusammenfassung und Ausblick

In der vorliegenden Bachelorarbeit wurde ein Prototyp zur Analyse von Wellenstrukturen der Grenzschicht in Molekulardynamik-Simulationen vorgestellt und implementiert. Dazu wurde sowohl auf bekannte und bewährte Verfahren aus der Literatur aufgebaut als auch eigene Konzepte, wie beispielsweise die Eingrenzung der Grenzfläche durch den Nutzer, umgesetzt. Die im einzelnen beschriebenen Module als Teil des Prototyps filtern die Partikelteilchen der Eingabedaten vor, aus welchen anschließend unter Verwendung einer Gauß-/Normalverteilung als radiale Basisfunktion das Volumen in Form eines Voxelgitters berechnet wird, sodass Isoflächen entstehen. Durch einen Parameter wird beim Raycasting anschließend bestimmt, welche Gitterpunkte in das Höhenfeld einfließen. Um im Anschluss daran bei der Fourier-Transformation aus dem Höhenfeld ein gutes Bild zu bekommen, welches nicht zu viel Rauschen enthält und damit relevante Frequenzen gut wiedergibt, sollte das Höhenfeld und die für dessen Extraktion wichtigen Parameter mit Bedacht gewählt werden. Außerdem wurden Grundlagen und Theorie zu der in Kapitel 4 implementierten Diskreten Fourier Transformation in Kapitel 2 beschrieben. Diese Frequenzanalyse kann für beliebig viele Zeitschritte berechnet und gestackt als Space-Time Cube ausgegeben werden, sodass eine Analyse der Grenzschicht über Zeit ermöglicht wird.

Mit Hilfe des Prototypen lassen sich außerdem die Zwischenschritte, wie z. B. das Voxelgitter bzw. die Isofläche bei der Oberflächenrekonstruktion und das in den Frequenzbereich transformierte Höhenfeld ausgeben und betrachten. Anschließend wurden Ergebnisse vorgestellt und besprochen, um die Möglichkeiten des Prototypen zu zeigen.

Zur schnelleren Berechnung wurde bei der Oberflächenrekonstruktion auf Parallelisierung unter Nutzung der API OpenMP gesetzt.

Ausblick

Nachdem der Prototyp vorgestellt und seine Möglichkeiten aufgezeigt wurden, soll darauf aufbauend gezeigt werden, wie an diese Arbeit angeknüpft werden kann.

Zum einen kann bei der Oberflächenrekonstruktion auf komplexere radial basis functions zurückgegriffen werden, um bessere Ergebnisse im Sinne von glatteren Funktionen zu erhalten [DTS02], was wiederum in weniger Frequenzen resultiert und damit einhergehend die Unterschiede zwischen den Zeitschritten bei der Frequenzanalyse besser auffallen. Solche komplexeren Methoden wären z. B. welche mit mehreren Skalen, um effizient auf radial basis functions basierende Oberflächen von vielen 100 000 Punkten berechnen zu können [CBC$^+$01]. Des Weiteren könnte man die Basisfunktionen auch iterativ addieren, bis eine ausreichende Auflösung erwünscht ist oder die verfügbare Zeit die Berechnung einschränkt [CBC$^+$01].

Bei der Isoflächenberechnung kann man außerdem noch die Höhe, in der Partikel innerhalb der Filterbox berücksichtigt werden - in dieser Arbeit als Cutoff beschrieben - abhängig von dem

minimalen Isowert und der Varianz berechnen, sodass abgeschnittene Partikel im Höhenbild keine Frequenzen im Frequenzspektrum erzeugen.

Weitergehend wäre es bei der Oberflächenrekonstruktion denkbar, die Grenzschicht z. B. über ein Dichtesampling automatisch wie in Abschnitt 3.1 beschrieben zu ermitteln, anstatt durch den Nutzer festlegen zu lassen. Dies hätte den Vorteil, dass man nicht in jedem Zeitschritt überprüfen muss, ob die Grenzschicht noch in der Filterbox liegt, da bei den Abdampfungsvorgängen der Simulationen über Zeit Moleküle verdampfen und die Grenzschicht nicht konstant bleibt.

Generell kann der Prototyp noch erweitert werden, um die Wechselwirkung von binären Stoffen herauszustellen. So wäre denkbar, die Isoflächen in diesem Fall getrennt zu berechnen und geeignete Transferfunktionen zu finden, sodass z. B. die wechselseitigen Vorgänge, wie bspw. dass Teilchen den Platz von dem anderen Stoff einnehmen, in der Grenzschicht der verschiedenen Stoffe sichtbar werden.

Auch wenn bereits auf Parallelisierung gesetzt wurde, so kann diese auch auf andere Module als nur auf die Oberflächenrekonstruktion ausgeweitet oder komplett auf die GPU übertragen werden (General Purpose Computation on Graphics Processing Unit - GPGPU), z. B. unter Verwendung von Nvidia's CUDA [RRB+08].

Des Weiteren werden die Ergebnisse derzeit in eine vtk-Datei exportiert und zur Darstellung in ein Drittanbieterprogramm geladen. Denkbar sind daher weitere Formate, die beispielsweise die Daten effizienter speichern oder den Zugriff auf diese erleichtern. Möglich wäre auch, das Volumen direkt in MegaMol zu rendern und unter Verwendung von Shadern eine passende Transferfunktion zu finden, die eventuell vorhandene Strukturen betont.

Außerdem wurden in Kapitel 4 verschiedene Modi für die Darstellung der Nullebene beschrieben. Der Prototyp erlaubt es, diese Stelle einfach um weitere Modi zu erweitern, um noch genauer auf die Bedürfnisse des Nutzers oder die Daten einzugehen.

A. Anhang

A.1. Projekt-Datei

Listing A.1 Projekt-Datei

```
<?xml version="1.0" encoding="utf-8"?>
<MegaMol type="project" version="1.0">
<!-- generated by ™MegaMol Configurator 1.0.1.0 -->

<!--

Use this command line arguments to start ™MegaMol
in Cmd:
  -p D:\ba\megamol\demo\mmspdview.xml -i mmspdview inst
in PowerShell:
  -p D:\ba\megamol\demo\mmspdview.xml -i mmspdview inst

-->
      <view name="mmspdview" viewmod="view">
            <module class="View3D" name="view" confpos="{X=20,Y=177}" />
            <module class="MMSPDDataSource" name="data" confpos="{X=1107,Y=174}">
                  <param name="filename" value="leicht-rot.mmspd" />
            </module>
            <module class="FrequencyAnalysis" name="frequencyanalysis"
                confpos="{X=510,Y=262}" />
            <module class="Baseline" name="baseline" confpos="{X=874,Y=224}">
                  <param name="baselineband" value="15" />
                  <param name="baselinenormal" value="0.000000;1.000000;0.000000" />
                  <param name="baselinedistance" value="10" />
            </module>
            <module class="SurfaceReconstruction" name="surfacereconstruction"
                confpos="{X=682,Y=391}" />
            <module class="EvaporationRenderer" name="renderer" confpos="{X=225,Y=176}" />
            <module class="RequestModule" name="requestmodule" confpos="{X=232,Y=375}" />
            <call class="MultiParticleDataCall" from="baseline::getdata" to="data::getdata" />
            <call class="Image2DCall" from="frequencyanalysis::getheightmap"
                to="surfacereconstruction::getheightmap" />
            <call class="MultiParticleDataCall" from="renderer::getdata" to="data::getdata" />
            <call class="CallRender3D" from="view::rendering" to="renderer::rendering" />
            <call class="BaselineCall" from="renderer::getbaseline"
                to="baseline::getbaselinedata" />
            <call class="BaselineCall" from="surfacereconstruction::getbaseline"
                to="baseline::getbaselinedata" />
            <call class="Image2DCall" from="requestmodule::getfrequency"
                to="frequencyanalysis::getfrequency" />
            <call class="Image2DCall" from="requestmodule::getheightmap"
                to="surfacereconstruction::getheightmap" />
      </view>
      <view name="debugview" viewmod="view2D">
            <module class="View2D" name="view2D" confpos="{X=35,Y=205}" />
            <module class="EvaporationDebugRenderer" name="renderer" confpos="{X=274,Y=202}"
                />
            <!--<call class="CallTimeControl" from="view2D::timecontrolslave"
                to="::main::view::timecontrolmaster" />-->
            <call class="CallRender2D" from="view2D::rendering" to="renderer::rendering" />
            <call class="Image2DCall" from="renderer::getheightmap"
                to="::main::surfacereconstruction::getheightmap" />
            <call class="Image2DCall" from="renderer::getfrequency"
                to="::main::frequencyanalysis::getfrequency" />
      </view>
</MegaMol>
```

Literaturverzeichnis

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva. Point Set Surfaces. In *Proceedings of the Conference on Visualization '01*, VIS '01, S. 21–28. IEEE Computer Society, Washington, DC, USA, 2001. URL http://dl.acm.org/citation.cfm?id=601671.601673. (Zitiert auf den Seiten 15 und 18)

[ABK98] N. Amenta, M. Bern, M. Kamvysselis. A New Voronoi-based Surface Reconstruction Algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, S. 415–421. ACM, New York, NY, USA, 1998. doi:10.1145/280814.280947. URL http://doi.acm.org/10.1145/280814.280947. (Zitiert auf den Seiten 13, 14, 16, 19 und 23)

[Ant] AntTweakBar. http://anttweakbar.sourceforge.net/doc/. (Zitiert auf den Seiten 25 und 32)

[BDA+14] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, C. Sheelagh. A Review of Temporal Data Visualizations Based on Space-Time Cube Operations. In *Eurographics Conference on Visualization*. Eurographics, 2014. (Zitiert auf Seite 25)

[BLN+13] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, C. T. Silva. A Benchmark for Surface Reconstruction. *ACM Trans. Graph.*, 32(2):20:1–20:17, 2013. doi:10.1145/2451236.2451246. URL http://doi.acm.org/10.1145/2451236.2451246. (Zitiert auf den Seiten 13, 14, 15, 16 und 23)

[Bou] P. Bourke. DFT FFT. http://paulbourke.net/miscellaneous/dft/. (Zitiert auf Seite 31)

[Bra00] R. N. Bracewell. *The Fourier transform and its applications*. Boston : McGraw Hill, 3rd ed Auflage, 2000. Previous ed.: 1978. (Zitiert auf den Seiten 20 und 21)

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, S. 67–76. ACM, New York, NY, USA, 2001. doi:10.1145/383259.383266. URL http://doi.acm.org/10.1145/383259.383266. (Zitiert auf den Seiten 18 und 49)

[CFB97] J. Carr, W. Fright, R. Beatson. Surface interpolation with radial basis functions for medical imaging. *Medical Imaging, IEEE Transactions on*, 16(1):96–107, 1997. doi:10.1109/42.552059. (Zitiert auf Seite 18)

[CL96] B. Curless, M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, S. 303–312. ACM, New York, NY, USA, 1996. doi:10.1145/237170.237269. URL http://doi.acm.org/10.1145/237170.237269. (Zitiert auf den Seiten 15, 16 und 17)

[CLW69] J. W. Cooley, P. A. Lewis, P. D. Welch. The fast Fourier transform and its applications. *Education, IEEE Transactions on*, 12(1):27–34, 1969. (Zitiert auf Seite 21)

[DG06] T. K. Dey, S. Goswami. Provable Surface Reconstruction from Noisy Samples. *Comput. Geom. Theory Appl.*, 35(1):124–141, 2006. doi:10.1016/j.comgeo.2005.10.006. URL http://dx.doi.org/10.1016/j.comgeo.2005.10.006. (Zitiert auf den Seiten 13, 15 und 16)

[DTS02] H. Q. Dinh, G. Turk, G. Slabaugh. Reconstructing Surfaces by Volumetric Regularization Using Radial Basis Functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(10):1358–1371, 2002. doi:10.1109/TPAMI.2002.1039207. URL http://dx.doi.org/10.1109/TPAMI.2002.1039207. (Zitiert auf den Seiten 15, 16, 17, 18, 19, 23 und 49)

[FN80] R. Franke, G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, 15:1691 – 1704, 1980. URL http://onlinelibrary.wiley.com/doi/10.1002/nme.1620151110/pdf. (Zitiert auf den Seiten 19 und 24)

[GKM+15] S. Grottel, M. Krone, C. Muller, G. Reina, T. Ertl. MegaMol - A Prototyping Framework for Particle-Based Visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 21(2):201–214, 2015. doi:10.1109/TVCG.2014.2350479. (Zitiert auf den Seiten 9 und 23)

[GP07] M. Gross, H. Pfister. *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007. (Zitiert auf den Seiten 15 und 16)

[GW01] I. Guskov, Z. J. Wood. Topological Noise Removal. In *Proceedings of Graphics Interface 2001*, GI '01, S. 19–26. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2001. URL http://dl.acm.org/citation.cfm?id=780986.780990. (Zitiert auf Seite 15)

[HAAT95] E. C. Hildreth, H. Ando, R. A. Andersen, S. Treues. Recovering three-dimensional structure from motion with surface reconstruction. *Vision Research*, 35(1):117 – 137, 1995. doi:http://dx.doi.org/10.1016/0042-6989(94)E0068-V. URL http://www.sciencedirect.com/science/article/pii/0042698994E0068V. (Zitiert auf den Seiten 13 und 15)

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. Surface Reconstruction from Unorganized Points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, S. 71–78. ACM, New York, NY, USA, 1992. doi:10.1145/133994.134011. URL http://doi.acm.org/10.1145/133994.134011. (Zitiert auf den Seiten 13, 14, 16, 18 und 23)

[Kaz05] M. Kazhdan. Reconstruction of Solid Models from Oriented Point Sets. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2005. URL http://dl.acm.org/citation.cfm?id=1281920.1281931. (Zitiert auf den Seiten 17, 18 und 19)

[KBH06] M. Kazhdan, M. Bolitho, H. Hoppe. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, S. 61–70. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006. URL http:

//dl.acm.org/citation.cfm?id=1281957.1281965. (Zitiert auf den Seiten 15, 18 und 19)

[KP06] J. Keiner, J. Prestin. A fast algorithm for spherical basis approximation. *PURE AND APPLIED MATHEMATICS-MARCEL DEKKER INCORPORATED-*, 282:259, 2006. (Zitiert auf Seite 18)

[KSES12] M. Krone, J. E. Stone, T. Ertl, K. Schulten. Fast visualization of gaussian density surfaces for molecular dynamics and particle system trajectories. *EuroVis 2012 Short Papers*, 1:67–71, 2012. (Zitiert auf Seite 19)

[KSO04] R. Kolluri, J. R. Shewchuk, J. F. O'Brien. Spectral Surface Reconstruction from Noisy Point Clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, S. 11–21. ACM, New York, NY, USA, 2004. doi: 10.1145/1057432.1057434. URL http://doi.acm.org/10.1145/1057432.1057434. (Zitiert auf Seite 17)

[LC87] W. E. Lorensen, H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, S. 163–169. ACM, New York, NY, USA, 1987. doi:10.1145/37401.37422. URL http://doi.acm.org/10.1145/37401.37422. (Zitiert auf Seite 18)

[LM02] D. Lazzaro, L. B. Montefusco. Radial basis functions for the multivariate interpolation of large scattered data sets. *Journal of Computational and Applied Mathematics*, 140(1 - 2):521 – 536, 2002. doi:http://dx.doi.org/10.1016/S0377-0427(01)00485-X. URL http://www.sciencedirect.com/science/article/pii/S037704270100485X. Int. Congress on Computational and Applied Mathematics 2000. (Zitiert auf den Seiten 19 und 24)

[Meg] MegaMol. http://go.visus.uni-stuttgart.de/megamol/. (Zitiert auf den Seiten 9, 23 und 27)

[MN03] N. J. Mitra, A. Nguyen. Estimating Surface Normals in Noisy Point Cloud Data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, S. 322–328. ACM, New York, NY, USA, 2003. doi:10.1145/777792.777840. URL http://doi.acm.org/10.1145/777792.777840. (Zitiert auf Seite 16)

[MYR+05] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, K. R. Subramanian. Interpolating Implicit Surfaces from Scattered Surface Data Using Compactly Supported Radial Basis Functions. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05. ACM, New York, NY, USA, 2005. doi:10.1145/1198555.1198645. URL http://doi.acm.org/10.1145/1198555.1198645. (Zitiert auf den Seiten 18 und 19)

[Naj05] F. N. Najm. On the Need for Statistical Timing Analysis. In *Proceedings of the 42Nd Annual Design Automation Conference*, DAC '05, S. 764–765. ACM, New York, NY, USA, 2005. doi:10.1145/1065579.1065781. URL http://doi.acm.org/10.1145/1065579.1065781. (Zitiert auf Seite 24)

[Osg09] B. Osgood. The Fourier transform and its applications. *Lecture Notes for EE*, 261, 2009. (Zitiert auf den Seiten 20 und 21)

[Par] ParaView. http://www.paraview.org/. (Zitiert auf den Seiten 25 und 35)

[RRB+08] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, W.-m. W. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, S. 73–82. ACM, 2008. (Zitiert auf Seite 50)

[Sch] C. Schwermann. Molekulardynamiksimulation. (Zitiert auf Seite 9)

[SOS04] C. Shen, J. F. O'Brien, J. R. Shewchuk. Interpolating and Approximating Implicit Surfaces from Polygon Soup. In *Proceedings of ACM SIGGRAPH 2004*, S. 896–904. ACM Press, 2004. URL http://graphics.cs.berkeley.edu/papers/Shen-IAI-2004-08/. (Zitiert auf Seite 15)

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st Auflage, 2010. (Zitiert auf den Seiten 16 und 18)

[TO99] G. Turk, J. F. O'Brien. Shape Transformation Using Variational Implicit Functions. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, S. 335–342. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999. doi:10.1145/311535.311580. URL http://dx.doi.org/10.1145/311535.311580. (Zitiert auf den Seiten 18 und 23)

[TO02] G. Turk, J. F. O'Brien. Modelling with Implicit Surfaces That Interpolate. *ACM Trans. Graph.*, 21(4):855–873, 2002. doi:10.1145/571647.571650. URL http://doi.acm.org/10.1145/571647.571650. (Zitiert auf den Seiten 16 und 18)

[VKFH06] J. Vrabec, G. K. Kedia, G. Fuchs, H. Hasse. Comprehensive study of the vapour–liquid coexistence of the truncated and shifted Lennard–Jones fluid including planar and spherical interface properties. *Molecular physics*, 104(09):1509–1527, 2006. (Zitiert auf den Seiten 23 und 39)

[WHDS04] Z. Wood, H. Hoppe, M. Desbrun, P. Schröder. Removing Excess Topology from Isosurfaces. *ACM Trans. Graph.*, 23(2):190–208, 2004. doi:10.1145/990002.990007. URL http://doi.acm.org/10.1145/990002.990007. (Zitiert auf Seite 15)

[WKE99] R. Westermann, L. Kobbelt, T. Ertl. Real-time Exploration of Regular Volume Data by Adaptive Reconstruction of Iso-Surfaces. *The Visual Computer*, 15:100–111, 1999. (Zitiert auf Seite 19)

[Wol] Wolfram Alpha. http://www.wolframalpha.com. (Zitiert auf Seite 17)

[WVG92] J. Wilhelms, A. Van Gelder. Octrees for Faster Isosurface Generation. *ACM Trans. Graph.*, 11(3):201–227, 1992. doi:10.1145/130881.130882. URL http://doi.acm.org/10.1145/130881.130882. (Zitiert auf Seite 19)

[Yoo01] Y. Yoo. Tutorial on Fourier theory. *Retrieved June*, 17:2007, 2001. (Zitiert auf den Seiten 20 und 21)

[ZOF01] H.-K. Zhao, S. Osher, R. Fedkiw. Fast Surface Reconstruction Using the Level Set Method. In *Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01)*, VLSM '01, S. 194–. IEEE Computer Society, Washington, DC, USA, 2001. URL http://dl.acm.org/citation.cfm?id=832286.835639. (Zitiert auf den Seiten 14, 15, 16, 17, 18 und 19)

Alle URLs wurden zuletzt am 15. 06. 2015 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift