Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis Nr. 31

# Identifikation und Visualisierung von Halos in kosmologischen Simulationen

Karsten Schatz

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer:** | Prof. Dr. Thomas Ertl |
| **Betreuer:** | Dipl.-Inf. Christoph Müller,<br>Dipl.-Inf. Michael Krone,<br>Dipl.-Inf. Jens Schneider (KAUST) |
| **Beginn am:** | 15. April 2015 |
| **Beendet am:** | 15. Oktober 2015 |
| **CR-Nummer:** | I.6.6, J.2 |

# Kurzfassung

Physikalische Simulationen des Universums generieren immer mehr Daten. Insbesondere Simulationen Dunkler Materie und der von dieser ausgebildeten Strukturen sind dabei sehr beliebt, da sie Aufschluss über die Historie des Weltalls geben. Zur Analyse dieser Simulationen ist es wichtig, über geeignete Mittel zu verfügen, die die schiere Datenmenge handhabbar machen. Visualisierungen der Simulationsresultate sind hierfür ein geeignetes Mittel.

Ziel dieser Arbeit ist es, zum einen so genannte Halos, also Bereiche, in der die Dunkle Materie dicht gepackt ist, zu erkennen und diese zum anderen zu visualisieren. Während für die reine Dunkle Materie relativ viele Verfahren existieren, diese darzustellen, sind die Anwendungen speziell für Halos weniger häufig. Diese Arbeit konzentriert sich daher vor allem auf die Visualisierung der Halos und deren Bewegungen im Verlauf der Simulationen. Hierfür werden verschiedenste Techniken verwendet, um Interaktivität zumindest bei der Betrachtung kleinerer Datensätze zu erreichen.

# Inhaltsverzeichnis


**1 Einleitung** **11**
1.1 Überblick über die Arbeit . . . . . 12

**2 Physikalische Grundlagen** **15**
2.1 Verwandte Arbeiten . . . . . 15
2.2 Wichtige Einheiten . . . . . 17
2.3 Dunkle Materie . . . . . 23
2.4 Halos . . . . . 25
2.5 Kosmologische Simulationen . . . . . 26

**3 Detektion von Halos** **29**
3.1 Eingabedaten . . . . . 29
3.2 Grundideen der Halo-Detektion . . . . . 30
3.3 Überblick über verschiedene Halo-Finder . . . . . 33
3.4 Rockstar-Algorithmus . . . . . 35
3.5 Friends-of-Friends-Algorithmus mit variabler Linking Length . . . . . 38
3.6 Implementierung . . . . . 41

**4 Visualisierung von Halos** **47**
4.1 Verwandte Arbeiten . . . . . 47
4.2 Eingabedaten . . . . . 49
4.3 Repräsentanten . . . . . 50
4.4 Visualisierung unterschiedlicher Parameter . . . . . 52
4.5 Merger-Bäume . . . . . 54
4.6 Sub-Halos . . . . . 57
4.7 Shading- und Beleuchtungstechniken . . . . . 58
4.8 Partikelzugehörigkeiten . . . . . 65

**5 Resultate** **67**
5.1 Visualisierung der Halos . . . . . 67
5.2 Vergleich der unterschiedlichen Halo-Finder . . . . . 72
5.3 Performance-Analyse . . . . . 74

**6 Zusammenfassung** **79**
6.1 Ausblick . . . . . 79

**Literaturverzeichnis** **81**

# Abbildungsverzeichnis


1.1 Die Galaxie M87, eine elliptische Galaxie. Man erkennt deutlich, wie die Leuchtkraft nach außen hin immer weiter abnimmt (Abbildung aus [Sch06a]). 11

2.1 Entfernungsbestimmung im Weltall mittels Triangulation. Unten in der Mitte befindet sich die Sonne, links und rechts davon die Erde auf ihrer Umlaufbahn zu zwei verschiedenen Zeitpunkten. Oben befindet sich das Objekt, für das die Distanz zu bestimmen ist, inklusive seiner zwei projizierten Positionen. Darüber liegen die fixen Referenzpunkte. (Abbildung aus [Sch06a]). . . . . 18

2.2 Rotationsgeschwindigkeiten um das Galaxiezentrum am Beispiel der Galaxie NGC 3198 in Abhängigkeit vom Radius um das Galaxiezentrum. Die beobachteten Geschwindigkeiten (obere Kurve) passen eindeutig nicht zu den Geschwindigkeiten, die auftreten müssten, wenn die Galaxie nur aus der beobachteten Materie bestünde (disk). (Abbildung aus [Sch06a] [ABBS85]). . 23

2.3 Simulierte Dichteverteilungen zweier Halos. Die obere stellt eine Dichteverteilung eines Galaxiehaufens dar, die untere die einer Galaxie (Längen nicht maßstabsgetreu). Die Substruktur ist sehr gut in Form von helleren Punkten sichtbar (Abbildung aus [Sch06a][MGG+99]). . . . . 26

3.1 Grundprinzip der Spherical-Overdensity-Verfahren. Rot bedeutet hohe Dichte, weiß geringe. Blaue Punkte bezeichnen lokale Maxima der Dichte, die Radii der Halos sind in schwarz eingezeichnet. . . . . 31

3.2 Grundprinzip der Friends-of-Friends-Verfahren. Schwarze Punkte sind die Partikelpositionen. Die größeren Kreise besitzen den Radius der Linking Length. Alle Partikel, die weniger Abstand als die Linking Length besitzen, also innerhalb der Kreisradii liegen, werden zu einem Halo zusammengenommen (rot). . . . . 32

3.3 Initialer Schritt des FOF-Algorithmus mit variabler Linking Length. Jede Box hat Diagonale $min_ll$. Jedes Partikel (rot) ist mit dem ihm im selben Halo assoziierten Partikeln verbunden. . . . . 39

3.4 Halo-Hierarchie für Halos mit unterschiedlicher Linking-Length. . . . . 40

3.5 Koordinatenkombinationen für einen Morton-Order Quadtree mit Tiefe 3 . . 42

3.6 Ablauf des Ellipsoid-Fittings. Links: Partikel (schwarz) inklusive deren Mittelpunkt in rot. Mitte: Durch PCA errechnete Haupt- (durchgängig) und Nebenachse (stoppt bei rotem Punkt). Rechts: Projektion aller Partikel auf die Hauptachse, resultierendes Ellipsoid. . . . . 45

4.1 Prozess der Erstellung von Ellipsoiden auf Grafikhardware. . . . . 51
4.2 Effekte unterschiedlicher Tessellierungs-Faktoren auf die Darstellung der repräsentierenden Ellipsoide / Kugeln. . . . . 52
4.3 Visualisierungen der Halo-Massen des letzten Zeitschritts des Datensatzes (a) und (b) besitzen genau denselben Kamera-Standpunkt. Die Halos sind mit Faktor 10 skaliert. . . . . 53
4.4 Blaue und rote Pfeile visualisieren die Bewegungsrichtungen und die Haupt-Drehachsen. . . . . 54
4.5 Unterschiedliche Darstellungen von Merger-Bäumen. Die Knoten entsprechen den Halos zu bestimmten Zeitpunkten. Bei (b) wird zusätzlich die Position der Knoten-Halos verwendet. . . . . 55
4.6 Unterschiedliche Ansichten von Merger-Bäumen, die durch die periodischen Randbedingen „auseinandergerissen“ wurden. . . . . 56
4.7 Sub-Halos werden innerhalb teilweise gerenderter Host-Halos gezeigt. $\epsilon = 0.05$ 58
4.8 Verwendete Texturen für das Deffered Shading inklusive des finalen Bildes. . 60
4.9 Schema für die Beleuchtung von Linie $\vec{T}$. $\vec{L}$ zeigt zur Lichtquelle, $\vec{V}$ zum Betrachter. $\vec{N}$ ist die gesuchte Normale (Abbildung nach [SZH97]). . . . . 61
4.10 Vergleich von unbeleuchteten und beleuchteten Linien. . . . . 62
4.11 Wie das Objekt $B$ im Vergleich zu Objekt $A$ von der Kamera entfernt ist, ist in der Abbildung nicht direkt erkennbar. . . . . 63
4.12 Vergleich von Depth Darkening zum Ursprungszustand. . . . . 64

5.1 Kompletter Screenshot der Visualisierung der Halos des letzten Zeitschritts, inklusive Partikel und Steuerelemente. . . . . 67
5.2 Typischer Prozess der Anwendung des Visualisierungsprogramms. . . . . 68
5.3 Detailansicht eines oder mehrerer Halos. . . . . 70
5.4 Der Effekt des Depth Darkening. . . . . 71
5.5 Problematische Stellen in der Visualisierung. . . . . 71
5.6 Vergleich der Resultate der beiden Algorithmen. . . . . 73
5.7 Laufzeitanalyse der einzelnen Schritte des implementierten Halo-Finders für Zeitschritt $a(t) = 0{,}98$. . . . . 75
5.8 Rechenzeiten pro Frame in der Visualisierung. Der Bereich, in dem die Berechnungsdauer schwankt, wird rot dargestellt. . . . . 77

# Tabellenverzeichnis

2.1 Übersicht wichtiger Werte . . . . . 22

3.1 Datensätze der DarkSky-Simulation . . . . . 29

4.1 Für alle Datensätze der DarkSky-Simulation vorhandene Halo-Parameter . . 49

# 1 Einleitung

**Abbildung 1.1:** Die Galaxie M87, eine elliptische Galaxie. Man erkennt deutlich, wie die Leuchtkraft nach außen hin immer weiter abnimmt (Abbildung aus [Sch06a]).

Seit jeher versucht der Mensch, Verständnis über die Welt und das Universum, in dem er lebt, zu erlangen oder es zu vergrößern. Seit viele Fragen bezüglich der Entstehung der eigenen Welt zumindest ansatzweise geklärt sind, gewinnt die Erforschung des Weltalls immer mehr an Bedeutung. Wer hat sich bei der Betrachtung eines Sternenhimmels noch nicht die Frage gestellt, wie all dies entstanden ist? Immer leistungsstärkere Teleskope werfen den Blick gen Himmel. Die größten Strukturen, die dabei auffallen, sind - neben den uns nahen Planeten und der Sonne - die Galaxien. Immer ausgefeiltere Modelle wurden entwickelt, die die Bewegung der beobachteten Objekte vorhersagen sollten.

Es fiel jedoch bald auf, dass die beobachteten Bewegungen nicht immer zu den Berechnungen passen, insbesondere im Außenbereich von Galaxien. Sterne im Außenbereich rotieren weitaus schneller um das Galaxiezentrum, als es errechnet wurde. Betrachtet man eine Galaxie, fällt sehr schnell auf, dass die Leuchtkraft vom Zentrum nach außen hin sehr rasch abnimmt (Vgl. Abb. 1.1). Daher liegt der Schluss nahe, dass auch die Massedichte auf dem Weg nach außen immer geringer wird, weil das Verhältnis von leuchtender zu nicht leuchtender Materie als nahezu konstant angesehen wurde [Sch06a]. Das alles lässt nur den Schluss zu, dass in den Außenbereichen der Galaxien mehr Materie existiert, als bisher beobachtet werden konnte. Da diese Materie nur indirekt durch ihre Auswirkungen auf die Umgebung nachgewiesen werden kann, nicht jedoch direkt, wird sie als Dunkle Materie bezeichnet.

Dunkle Materie spielt im aktuellen kosmologischen Standardmodell eine große Rolle. So wird angenommen, dass die meisten Galaxien einen so genannten Halo Dunkler Materie besitzen, welche sich viel weiter in den umgebenden Raum erstrecken, als die sichtbare Materie dies tut. Darüber hinaus wird angenommen, dass ca. $85\%$ aller Materie im Universum dunkle Materie ist [Pap07], was die Wichtigkeit neuer Erkenntnisse darüber noch unterstreicht.

Um diese Erkenntnisse zu erhalten, ist neben der Beobachtung des uns umgebenden Universums auch die Simulation desselben von Nutzen. So sind z. B. auch vergangene Zustände und mögliche zukünftige Entwicklungen betrachtbar. Daher wurden in der Vergangenheit mehrere große Simulationen, wie etwa der Millenium-Run ([SET+05]) oder die Bolshoi-Simulation ([KTGP11]) durchgeführt, um zum einen das aktuelle Modell des Universums mit den Beobachtungen abzugleichen und zum anderen, um Anhaltspunkte für weitere Beobachtungen zu gewinnen. Diese Simulationen generieren riesige Datenmengen, jedoch sind leider die menschlichen Fähigkeiten, Zusammenhänge in schier endlosen Zahlenkolonnen zu erkennen, nicht sehr weit entwickelt. Deshalb besteht eine großer Bedarf an Analyse- und Visualisierung-Werkzeugen, um der Daten Herr zu werden.

Somit ist es das Ziel dieser Arbeit, aus den Partikeldaten der DarkSky-Simulation ([SWT+14]) Halos Dunkler Materie zu detektieren und diese dann zu visualisieren. Diese simuliert die Verteilung Dunkler Materie im Universum von kurz nach dem Urknall bis heute. Eine der größeren Herausforderung hier ist die Handhabung der Datenmengen, da die gleichzeitige Haltung aller Daten im Arbeitsspeicher schwierig und im Grafikspeicher schier unmöglich ist. Dies kann mittels geeigneter Vorberechnung und weiterer Aufbereitung der Daten erreicht werden. Die Visualisierung soll zusätzliche Analysemöglichkeiten bieten, um die Daten nach spezifischen Parametern durchsuchen zu können.

## 1.1 Überblick über die Arbeit

In Abschnitt 2 werden zunächst die physikalischen Grundlagen genauer beleuchtet. Es werden wichtige Größen eingeführt und Dunkle Materie und die Definition von Halos

näher erklärt. In Kapitel 3 wird die Halo-Detektion betrachtet. Neben den Grundideen hierfür werden zwei Methoden, darunter auch die im Rahmen dieser Arbeit implementierte, genauer betrachtet. Abschnitt 4 behandelt die Visualisierung der Daten. Neben einem kurzen Überblick über die vorhandenen Daten werden dort hauptsächlich die implementierten Rendering- und Analyse-Techniken beschrieben. In Kapitel 5 werden die Resultate dieser Arbeit präsentiert und diskutiert. Zusätzlich zur qualitativen Bewertung wird auch die Performance der geschriebenen Software untersucht. Mit einer Zusammenfassung in Kapitel 6 schließt diese Arbeit dann ab.

# 2 Physikalische Grundlagen

Trotz der informatischen Ausrichtung dieser Arbeit ist es nötig, sich zunächst mit den physikalischen Aspekten des Themengebiets zu beschäftigen. Zum einen um zu verstehen, was genau visualisiert werden soll, und zum anderen, um Fehler zu vermeiden, die in der Interpretation der Eingabedaten dank zu geringem physikalischem Verständnis auftreten könnten.

Daher wird zunächst in Abschnitt 2.1 auf relevante andere Arbeiten im physikalischen Bereich verwiesen. Daraufhin wird ein Überblick über alle wichtigen physikalischen Einheiten gegeben, die in diesem Zusammenhang von Bedeutung sind. In Sektion 2.3 wird darauffolgend geklärt, was Dunkle Materie ist bzw. was Dunkle Materie nach aktuellem Stand der Wissenschaft sein könnte, da darüber momentan noch keine Klarheit herrscht. Daraufhin wird in Abschnitt 2.5 auf kosmologische Simulationen eingegangen. Anschließend ausgeführt, wie Halos definiert werden, damit sie im nächsten Kapitel dann detektiert werden können. Da fast die kompletten Eingabedaten für die Visualisierung Resultate einer solchen Simulation sind, ist es final noch nötig, sich mit deren Grundannahmen und deren Design zu beschäftigen.

## 2.1 Verwandte Arbeiten

Die relevante Literatur für diesen Abschnitt ist sehr weit gestreut, da es sich hierbei eher um eine Arbeit mit informatischen Anspruch handelt. Deshalb wird die meiste Literatur zu physikalischen Belangen eher zur Klärung von Grundlagenfragen herangezogen werden. Im Bereich der Kosmologie wird vor allem das umfangreiche Lehrbuch von Schneider [Sch06a] Anwendung finden. Bei diesem ist Vorsicht geboten, da sich die beiden verwendeten Varianten, zum einen die 1. deutsche Auflage [Sch06a], zum anderen die 2. englische Auflage [Sch06b], inhaltlich teils stark unterscheiden. Nichtsdestotrotz bietet das Werk eine Einführung in so gut wie alle für diese Arbeit relevanten Teilgebiete der Physik.

Ein Einführung zu Dunkler Materie und Dunkler Energie wurde von Papantonopoulos herausgegeben [Pap07]. Neben den wahrscheinlichsten möglichen Kandidaten für Dunkle Materie wird auch auf deren Effekt auf die Galaxieentstehung eingegangen. Des Weiteren werden noch alternative Hypothesen diskutiert, welche ohne Dunkle Materie zur Erklärung der beobachteten Phänomene auskommen sollen.

Zur Beobachtung des Universums um uns herum wurden in den vergangenen Jahrzehnten unterschiedlichste Vermessungen (Surveys) durchgeführt. Die Simulationsdaten, welche in dieser Arbeit verwendet werden, wurden größtenteils mit der Hilfe von Daten der Planck-Mission [AAA+14] erstellt. Diese wurden vom Planck-Satelliten der European Space Agency (ESA) aufgezeichnet, welcher sich von August 2009 bis Oktober 2013 im Lagrange-Punkt L2 (Punkt im All, an dem die Sonnenstrahlung von der Erde abgeschirmt wird) befand und mit Hilfe des verbauten Teleskops große Bereiche der Hemisphäre untersuchte. Hauptziel war es, sowohl die kosmische Mikrowellen-Hintergrundstrahlung als auch extragalaktische Objekte zu untersuchen. Neben Planck existierten in der Vergangenheit noch andere Surveys, welche das uns umgebende Weltall untersuchten. Darunter sticht vor allem die Sloan Digital Sky Survey (SDSS [YAAJ+00]) hervor, deren Ziel es war die hellsten Quasare und die hellsten Galaxien zu untersuchen. Nach Abschluss dieser Studie wurden zwei weitere Folgeuntersuchungen durchgeführt (SDSS-II [FBB+08], SDSS-III [EWA+11]). Während sich erstere vor allem auf Supernovae konzentrierte, maß letztere unter anderem im Rahmen der Baryon Osciallation Spectroscopics Survey (BOSS [DSA+13]) die Rotverschiebungen einer großen Anzahl von massiven Galaxien (zu Rotverschiebungen vgl. Abschnitt 2.2.4). Es wurde ebenfalls untersucht, wo sich in näherer Umgebung unseres Sonnensystems größere Mengen Dunkler Materie befinden könnten.

Da Dunkle Materie noch nirgends direkt beobachtet werden konnte, basieren viele Erkenntnisse über die Plausibilität dieser Theorie nur dank numerischer Simulation selbiger. Für gut aufgelöste Simulationen ist zumeist eine riesige Rechenleistung vonnöten. Die Simulation, deren Resultate hier zur Visualisierung herangezogen werden, ist die DarkSky-Simulation [SWT+14] aus dem Jahre 2014. Diese berechnete die möglichen Bewegungen dunkler Materie vom Urknall bis zum jetzigen Zeitpunkt. Dabei wurden mehrere Hundert Terabyte an Partikeldaten generiert, deren Analyse schon aufgrund der schieren Menge eine große Herausforderung darstellt. Dabei wurde, im Gegensatz zur Millenium-Simulation von 2005 [SET+05], ein gitterloses Verfahren verwendet, um die Kräfte zwischen den simulierten Partikeln Dunkler Materie zu berechnen. Während bei der Millenium-Simulation oder der Bolshoi-Simulation [KTGP11] aus dem Jahre 2011 „nur“ mehrere Milliarden Partikel verfolgt wurden, untersuchen die DarkSky-Simulationen zum Teil über eine Billionen Dunkle Materieteilchen.

Aus den Daten dieser Simulationen können Erkenntnisse gewonnen werden, wie sich die Dunkle Materie im Universum verteilt. Diese wird zumeist in Halos von Galaxien vermutet. Deshalb ist es von großer Wichtigkeit, diese Halos detektieren zu können (vgl. dazu Kapitel 3). Dazu ist es nötig, Vermutungen über die genauere Form dieser Halos anzustellen. Dies wurde unter anderem von Allgood et al. getan [AFP+06], welche unter anderem eine Methode präsentierten, Ellipsoide an die vermutete Form anzupassen. Speziell auf die Dichteverteilung innerhalb eines Halos konzentriert sich die Arbeit von Navarro, Frenk und White [NFW96]. Halos können auch recht spezielle Substrukturen ausbilden, welche von Moore et al. intensiv untersucht wurden [MGG+99].

Neben all den Arbeiten zu Dunkler Materie und deren Simulation und Eigenschaften sei noch erwähnt, dass alternative Hypothesen existieren, welche ohne Formen exotischer Materie auskommen. So formulieren die Modified Newtonian Dynamics von Milgrom (MOND, [Mil83]) die Gravitationsgleichungen um, um die Beobachtungen auch ohne neue Arten von Materie erklären zu können. Hinweise, dass die aktuelle Theorie der dunklen Materie fehlerhaft sein könnte, wurden unter anderem von Kroupa et al. gefunden [Kro12]. Anscheinend lassen sich Beobachtungen bestimmter Zwerggalaxien nicht mit dieser in Einklang bringen. Dennoch sei erwähnt, dass die so genannte kalte Dunkle Materie weiterhin zum Standardmodell der Kosmologie gezählt wird und zum jetzigen Zeitpunkt noch als aktueller wissenschaftlicher Stand wahrgenommen wird.

## 2.2 Wichtige Einheiten

Dieser Abschnitt führt einige für die Astrophysik wichtige Einheiten ein. Dabei wird eine Unterteilung in unterschiedliche Kategorien vorgenommen, die jedoch so nicht ganz zu halten sind. Die Übergänge zwischen den wichtigen Einheiten in der Astrophysik sind sehr fließend, daher ist es schwer, genaue Einordnungen zu treffen, wie insbesondere am Beispiel des kosmologischen Skalenfaktors klar werden wird.

### 2.2.1 Position und Distanzen

Durch unseren mehr oder weniger festgelegten Beobachterposten auf diesem Planeten gestaltet es sich als sehr schwer, exakte Positionen von Objekten im Weltall zu bestimmen, da sich sowohl unser Planet, als auch die Sonne, um die dieser kreist, als auch die Galaxie, in der sich diese befindet, weiterhin durch das Weltall bewegen. Zudem „sehen“ wir durch die sehr großen Entfernungen zu den beobachteten Objekten nur deren Position im zweidimensionalen Raum des Nachthimmels, nicht jedoch deren Distanz zu uns. Für die Information der Distanz wäre ein zweiter Beobachterposten nötig, um eine Triangulation zu ermöglichen. Unser Planet ist jedoch im Bezug auf die Distanzen, die im Weltall bestimmt werden sollen so klein, dass jeder mögliche zweite Beobachterposten auf der Erde zu nahe am ersten wäre, um genaue Bestimmungen zu ermöglichen. Aus diesen Gründen ist es nötig, sich entweder geometrischer Tricks neben der Triangulation zu bedienen oder aber weitere Parameter - wie die Leuchtkraft der beobachten Objekte - zu Hilfe zu nehmen [Sch06a].

Die grundlegendste der Methoden bedient sich eines kleinen Tricks, um die Distanz zwischen den beiden Beobachterpunkten für eine Triangulation zu erhöhen. Dazu sind jedoch sehr weit entfernte Objekte als Referenzpunkte nötig (viel weiter als das zu messende Objekt), welche durch sehr geringe Bewegung an der Himmelssphäre auffallen. Je näher sich ein Objekt am Beobachterposten befindet, desto mehr wird es sich für den Beobachter bewegen, wenn dieser seine Position ändert. Die Idee dabei ist, die Bewegung der Erde um die Sonne auszunutzen,

um so zwei möglichst weit entfernte Standpunkte zu bekommen. Der Distanzmessungsaufbau ist in Abbildung 2.1 gezeigt.

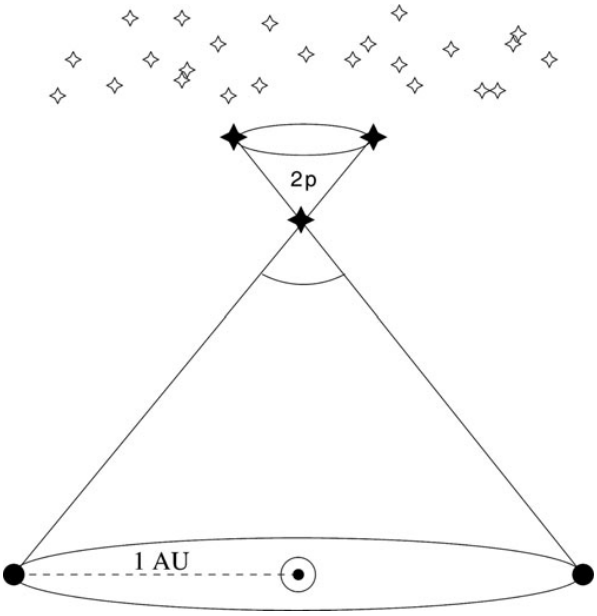



**Abbildung 2.1:** Entfernungsbestimmung im Weltall mittels Triangulation. Unten in der Mitte befindet sich die Sonne, links und rechts davon die Erde auf ihrer Umlaufbahn zu zwei verschiedenen Zeitpunkten. Oben befindet sich das Objekt, für das die Distanz zu bestimmen ist, inklusive seiner zwei projizierten Positionen. Darüber liegen die fixen Referenzpunkte. (Abbildung aus [Sch06a]).

Betrachtet man den Aufbau aus Abbildung 2.1, fällt auf, dass die Projektion des zu bestimmenden Punktes auf der Himmelssphäre eine Ellipse beschreibt. Von der Länge der größeren der beiden Halbachsen $p$ lässt sich mittels folgender Gleichung direkt auf die Distanz $D$ zum beobachteten Punkt schließen ([Sch06a]):

$$\frac{r}{D} = \tan p \approx p \tag{2.1}$$

Die Approximation des Tangens ist möglich, da $p$ so gut wie immer viel kleiner als $1$ sein wird. $p$ wird im Bogenmaß auf der Himmelssphäre bestimmt, bei $r$ handelt es sich um den Radius der Erdumlaufbahn. Nun ist alles Nötige zusammen, um eine der wichtigsten Distanzeinheiten der Astronomie zu definieren, des Parsecs (pc). Ein Parsec ist die Distanz eines hypothetischen Objekts, wenn die größere der beiden Halbachsen eine Länge von genau einer Bogensekunde besitzt. Daraus folgt (nach [Sch06a]):

$$1pc = 3{,}086 \cdot 10^{18} cm \tag{2.2}$$

Damit kann die Distanzgleichung auf

$$D = \left(\frac{p}{1''}\right)^{-1} pc \tag{2.3}$$

vereinfacht werden. Diese Form der Distanzbestimmung hat jedoch ihre Grenzen. Distanzbestimmungen benötigen mindestens ein halbes Jahr, idealerweise jedoch länger. Zudem ist diese Methode nur bei nahen Objekten innerhalb eines Radius von ca. $30pc$ praktikabel, da Verschmierungen durch die Erdatmosphäre die Messung von zu kleinen Winkeln unmöglich machen [Sch06a]. Für weitere Distanzen werden andere Methoden verwendet, die, mit Ausnahme der Distanzbestimmung per Rotverschiebung, hier jedoch nicht erörtert werden. Für diese sei jedoch auf Abschnitt 2.2.4 verwiesen, da bei dieser die Hubble-Konstante eine tragende Rolle spielt.

Es sei des Weiteren noch angemerkt, dass auf diese Weise nur Distanzen von der Erde zum jetzigen Zeitpunkt zu vergangenen Positionen von anderen Objekten berechnet werden, da insbesondere bei weiten Entfernungen die Reisezeit des Lichts zu uns eine starke Rolle spielt. Um die wirkliche aktuelle Distanz zu berechnen, müsste man zunächst die jetzige Position des Ziels extrapolieren.

Wie vielleicht schon aufgefallen sein sollte, wurden bis jetzt nur Positionen von Objekten relativ zu unserer Sonne bzw. der Erde vermessen. Das hat den Grund, dass eine absolute Positionsbestimmung eines absoluten Koordinatensystems bedarf, welches wir quasi nicht besitzen. Es wäre ein Ursprung für das Koordinatensystem nötig, in dem absolute Koordinaten bestimmt werden sollen. Bei Positionsbestimmungen auf der Erde können einfach alle Positionen relativ zum Erdmittelpunkt ermittelt werden, im Sonnensystem relativ zur Sonne. Es ist jedoch keine Position für ein „Zentrum“des Universums bekannt, was absolute Positionen sehr schwierig macht.

### 2.2.2 Zeit

Die Bestimmung von Zeitpunkten und Zeitdauern unterscheidet sich so gut wie gar nicht von der sonst verwandten Zeitmessung. Normalerweise wird das normale Erdenjahr (365 Tage) als Referenz gewählt, aufgrund der sehr langen Zeiträume jedoch meist als „Giga-Jahr“(Gyr) verwendet, was einer Milliarde Jahre entspricht.

Einen direkten Bezug zur Zeit hat der so genannte kosmologische Skalenfaktor $a(t)$. Dieser beschreibt, wie sich Koordinaten im Universum mit dessen Expansion ändern. Es gilt dabei, dass $a(t_0) = 1$, wobei $t_0$ heute ist. Die Expansion des Universums legt nahe, dass es einen Zeitpunkt $t_x$ mit $a(t_x) = 0$ gegeben haben muss, an dem alle beobachtbaren Positionen in einer Singularität lagen. Somit wäre $t_x$ der Zeitpunkt des Urknalls. Deshalb wird, in Annahme des Urknalls, $a(0) = 0$ gesetzt [Sch06a]. Der Verlauf von $a(t)$ ist jedoch nicht zwangsläufig linear, auch wenn es diesen Anschein hat. Der Skalenwert kann, wenn er bekannt ist,

zur Korrektur von vergangenen Objektpositionen verwendet werden, um dem Effekt des expandierenden Universums entgegenzuwirken und ein über die Zeit hinweg konsistentes Koordinatensystem zu erhalten. Zusätzlich kann von heutigen Positionen auf vergangene geschlossen werden. Dazu nimmt man ein Koordinatensystem zum jetzigen Zeitpunkt ($t_0$) an, dessen Zentrum das Zentrum der kugelförmigen Ausdehnung des Universums beschreibt. Damit können die Koordinaten $r(t)$ zu einem beliebigen Zeitpunkt $t \geq 0$ in Abhängigkeit der jetzigen Position $\vec{x}$ und des Skalenfaktors beschrieben werden [Sch06a]:

$$r(t) = a(t) \cdot \vec{x} \tag{2.4}$$

Solche korrigierten Koordinaten werden auch als mitbewegte Koordinaten (engl. comoving coordinates) bezeichnet.

### 2.2.3 Masse

Würde man bei Massenangaben im kosmologischen Maßstab auf altbekannte Einheiten wie die Tonne bzw. das Gramm zurückgreifen, wären die resultierenden Zahlenwerte exorbitant groß. Daher hat man sich auf eine andere Einheit für die Angabe größerer Massen geeinigt. Die Masse unserer Sonne $M_\odot$ dient hierbei als Bezug. Sie beträgt [Sch06a]:

$$M_\odot = 1{,}989 \cdot 10^{33} g \tag{2.5}$$

Eigentlich wäre hier noch eine Messungenauigkeit anzugeben, für unsere Zwecke reicht eine solche Festsetzung jedoch aus. Bei Massen weit unter einer Sonnenmasse wird selbstverständlich auf die normalen SI-Einheiten zurückgegriffen.

### 2.2.4 Die Hubble-Konstante

Beobachtet man die uns umgebenden Galaxien, fällt auf, dass sich alle bis auf die uns sehr nahen, von uns fortzubewegen scheinen. Dies gelingt durch die Beobachtung des Farbspektrums dieser Galaxien. Bei allen Galaxien, außer den uns nahen, ist das Spektrum deutlich ins rötliche verschoben, was auf eine Bewegung von uns fort hindeutet. Je weiter eine Galaxie von uns entfernt ist, desto rötlicher ist das Spektrum, d. h. je weiter entfernt, desto höher ist die Bewegungsgeschwindigkeit von uns Weg [Sch06a]. Dies wurde auch von Edwin Hubble entdeckt, welcher einen direkten Zusammenhang zwischen der Geschwindigkeit $v$ und der Distanz der Galaxien $D$ erkannte [Hub29]:

$$v = H_0 D \tag{2.6}$$

Der Wert $H_0$ wird daher auch als Hubble-Konstante bezeichnet. Der exakte Wert dieser Konstanten ist bis heute nicht genau festgelegt. Das ist der Fall, da, wie weiter oben schon

erwähnt, die absolute Entfernungsberechnung im galaktischen Maßstab eine sehr schwere Aufgabe ist. Der verwendete Wert kann in unterschiedlichen Publikationen stark variieren. In neueren Veröffentlichungen gilt aber fast durchgängig [Sch06a], dass

$$60\frac{km}{s \cdot Mpc} \lesssim H_0 \lesssim 80\frac{km}{s \cdot Mpc} \tag{2.7}$$

Für unsere Zwecke soll $H_0 = 68{,}8062\frac{km}{s \cdot Mpc}$ gelten, da dieser Wert in dem für diese Arbeit verwendeten Datensatz ([SWT+14]) verwendet wird. Aufgrund dieser Unbekanntheit der Hubble-Konstante ist man dazu übergegangen, statt $H_0$ den Wert $H$ zu verwenden [Sch06a][Cro13]:

$$H = h \cdot 100\frac{km}{s \cdot Mpc} \tag{2.8}$$

Ist dies der Fall, wird immer $h$ angegeben, in unserem Fall also $h = 0{,}688062$.

Wie schon erwähnt, ist die Beobachtung, die zur Hubble-Konstanten führt, ebenfalls nützlich, um Distanzen zu bestimmen. Dazu ist noch eine Definition der Rotverschiebung $z$ nötig [Sch06a]. Sie wird mit Hilfe der Spektralanalyse bestimmt. Es gilt:

$$z := \frac{\lambda_{obs} - \lambda_0}{\lambda_0} \tag{2.9}$$

Der Wert $\lambda_{obs}$ bezeichnet hierbei die beobachtete Wellenlänge, $\lambda_0$ ist die ursprüngliche. Für kleine Rotverschiebungen gilt $v \approx zc$, wobei $c$ die Lichtgeschwindigkeit im Vakuum ist [Sch06a]. Es existiert auch noch eine allgemeine Gleichung. Diese zu erörtern, würde jedoch den Rahmen der Arbeit sprengen.

Mit den bis jetzt entwickelten Formeln kann also direkt von der Rotverschiebung auf die Distanz zu uns geschlossen werden:

$$D \approx 3000zh^{-1}Mpc \tag{2.10}$$

Bei dieser Art von Distanzangabe hat jedoch immer der verwendete Wert von $h$ bekannt zu sein, da die Hubble-Konstante, wie bereits erwähnt, bisher nicht genau festgelegt wurde. Idealerweise werden die Distanzangaben noch nicht mit $h^{-1}$ verrechnet, um nachträgliche Korrekturen zu ermöglichen [Cro13].

Es sei noch erwähnt, dass die Rotverschiebung auch gerne verwendet wird, um Zeitpunkte zu beschreiben. Gemeint ist dann immer der Zeitpunkt, während dem das betrachtete Objekt die angegebene Rotverschiebung zu uns jetzt besitzt.

**Tabelle 2.1:** Übersicht wichtiger Werte

| | Wert | Einheit | entspricht(hier) | Notiz |
|---|---|---|---|---|
| Distanz | $D$ | $1pc$ | $3{,}086 \cdot 10^{18} cm$ | Parsec |
| Zeit | $t$ | $1Gyr$ | $10^9 yr$ | Eine Milliarde Jahre |
| Masse | $m$ | $1M_\odot$ | $1{,}989 \cdot 10^{33} g$ | Sonnenmasse |
| Hubble-Konstante | $H_0$ | $\frac{km}{s \cdot Mpc}$ | $68{,}8062 \frac{km}{s \cdot Mpc}$ | genauer Wert |
| Korrigierte Hubble-Konstante | $H$ | $\frac{km}{s \cdot Mpc}$ | $h \cdot 100 \frac{km}{s \cdot Mpc}$ | Korrigiert durch $h$ |
| Hubble-Korrekturparameter | $h$ | keine | $0{,}688062$ | |
| Kritische Dichte | $\rho_{cr}$ | $\frac{g}{cm^3}$ | $1{,}88 \cdot 10^{-29} h^2 \frac{g}{cm^3}$ | |
| Dichte | $\Omega_0$ | keine | $1$ | korrigiert mit $\rho_{cr}$ |
| Materiedichte | $\Omega_m$ | keine | $0{,}295038$ | korrigiert mit $\rho_{cr}$ |
| Strahlungsdichte | $\Omega_r$ | keine | $0$ | korrigiert mit $\rho_{cr}$ |
| Vakuumsdichte | $\Omega_\Lambda$ | keine | $0{,}704962$ | korrigiert mit $\rho_{cr}$ |

### 2.2.5 Weitere Werte

Zu den weiteren wichtigen Werte gehört unter anderem die Lichtgeschwindigkeit $c = 2{,}998 \cdot 10^8 \frac{m}{s}$. Für die spätere Definition von Halos sind die verschiedenen Dichteparameter des Universums von großer Bedeutung. Allen voran steht die kritische Dichte des Universums $\rho_{cr}$, für welche gilt [Sch06a]:

$$\rho_{cr} := \frac{3H_0^2}{8\pi G} = 1{,}88 \cdot 10^{-29} h^2 \frac{g}{cm^3} \tag{2.11}$$

Hierbei ist $G$ die Newtonsche Gravitationskonstante. Bei der kritischen Dichte handelt es sich genau um die Dichte, bei der das ausdehnende Universum wieder beginnen würde, sich zusammenzuziehen. Diese wird verwendet, um die anderen wichtigen Dichteparameter dimensionslos zu machen. Diese anderen wichtigen Dichten sind die Massedichte, die Strahlungsdichte und die Vakuumsdichte. Letztere hängt direkt von der von Einstein eingeführten kosmologischen Konstante $\Lambda$ ab, welche die Energiedichte des Vakuums beschreibt. Es gilt $\Omega_\Lambda = \frac{\Lambda}{3H_0^2}$. Hinzu kommt die allgemeine Dichte des Universums $\Omega_0 = \Omega_m + \Omega_r + \Omega_\Lambda$. Deren genau Bestimmung ist jedoch nicht Gegenstand dieser Arbeit. Die unterschiedlichen Dichteparameter sind, zusammen mit den anderen beschriebenen Parametern und den hier verwendeten Werten, in Tabelle 2.1 gezeigt. Speziell für die Dichten müssen die Zahlenwerte nicht unbedingt den real gültigen entsprechen. Sie sind die, die in [SWT+14] für die Simulation verwendet wurden.

## 2.3 Dunkle Materie

In der Vergangenheit wurden meist immer dann Anpassungen am physikalischen Weltbild vorgenommen, wenn die Beobachtungen nicht zu den berechneten Ergebnissen passten. Bei Dunkler Materie handelt es sich um eine dieser Anpassungen. Vor der Notwendigkeit der Einführung der Dunklen Materie ging man davon aus, dass das Newtonsche Gravitationsgesetz (evtl. inklusive kleiner relativistischer Anpassungen) für alle Objekte im Weltall Gültigkeit besitzt. Als sie sich jedoch auf die Außenbereiche von Galaxien fixierte, stellten die Forscher fest, dass die beobachteten Sternbewegungen nicht zu ihren berechneten Bahnen passten. Laut des Gravitationsgesetzes sind die Geschwindigkeiten dieser Sterne viel zu hoch, als dass diese sich kreisförmig um das Galaxiezentrum bewegen dürften. Vielmehr müssten sie langsam in den sie umgebenden Raum abdriften. Da sie das aber nicht taten, existierten zwei Möglichkeiten: Entweder waren die Gleichungen, aufgrund derer die Bahnen berechnet wurden, grundlegend falsch, oder es existierte eine zusätzliche noch unbekannte Gravitationsquelle in den Galaxien, welche die Sterne auf ihrer Bahn hielt.

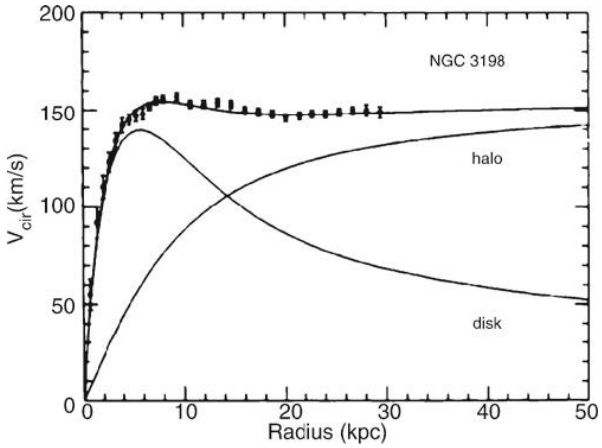



**Abbildung 2.2:** Rotationsgeschwindigkeiten um das Galaxiezentrum am Beispiel der Galaxie NGC 3198 in Abhängigkeit vom Radius um das Galaxiezentrum. Die beobachteten Geschwindigkeiten (obere Kurve) passen eindeutig nicht zu den Geschwindigkeiten, die auftreten müssten, wenn die Galaxie nur aus der beobachteten Materie bestünde (disk). (Abbildung aus [Sch06a] [ABBS85]).

Dies zeigt sich auch anhand des Beispiels der Galaxie NGC 3198 in Abbildung 2.2. Bestünde die Galaxie nur aus der beobachteten Materie (also größtenteils aus den Sternen), würde die Rotationskurve nach außen hin abflachen („disk“). Die beobachtete Rotationskurve tut dies jedoch nicht (obere Kurve). Es lag also der Schluss nahe, dass innerhalb der Galaxie noch mehr Masse existiert, die jedoch nicht beobachtet werden kann. Diese Masse müsste demzufolge mit der in der Kurve „halo“ gezeigten Geschwindigkeiten um das Zentrum rotieren, damit die Beobachtungen auch weiterhin mit den alten Gleichungen erklärt werden können [Sch06a]. Somit war die Idee der Dunklen Materie geboren.

Laut neueren Berechnungen hat die vermutete Dunkle Materie einen Anteil von mindestens $60\%$ an der Gesamt-Materie im Universum [OS03]. Bis heute konnte sie jedoch nicht direkt beobachtet werden, einzig ihr indirekter Beitrag in Form von Gravitation ist bekannt [Sch06a]. Neben der Dunklen Materie wird zur Erklärung aller Beobachtungen, insbesondere der sehr schnellen Expansion des Universums, noch die so genannte Dunkle Energie benötigt. Diese soll hier jedoch nicht weiter beachtet werden, da sie im Kontext dieser Arbeit keine Rolle spielt.

Offen bleibt weiterhin die Frage, aus was genau die Dunkle Materie besteht. Hier existieren mehrere Theorien. Naheliegend ist natürlich zunächst, dass es sich bei Dunkler Materie um baryonische Materie handelt, also um solche, die aus normalen Atomen aufgebaut ist. So könnte es sich dabei um sehr kalten Staub oder Gas handeln. Jedoch machen dies die anderen physikalischen Eigenschaften unmöglich, dass solch große Mengen unentdeckt blieben [Pap07]. Deshalb hat sich die Forschung größtenteils auf die nicht-baryonische Dunkle Materie konzentriert. Hier existieren zahlreiche Möglichkeiten und Theorien. Es wird dabei zwischen heißer und kalter Dunkler Materie unterschieden.

Bei heißer Dunkler Materie handelt es sich um sich sehr schnell bewegende Teilchen. Neutrinos zählen hier zu den bereits bekannten Kandidaten. Simuliert man jedoch die Auswirkungen von heißer Dunkler Materie auf die Strukturbildung im Universum, wird schnell klar, dass diese nicht die beobachtete Entstehungsgeschichte der Strukturen im Universum erklären können [Pap07].

Nicht-baryonische kalte Dunkle Materie stattdessen, die sich im Gegensatz zu heißer Dunkler Materie im Verhältnis zur Lichtgeschwindigkeit relativ langsam bewegt, würde die Beobachtungen weitaus besser erklären. Daher existieren relativ viele Kandidaten für kalte Dunkle Materie, von denen bis heute aber noch keiner experimentell nachgewiesen werden konnte. Zu diesen gehören die so genannten WIMPs (Weakly interacting massive particles), welche von den meisten Forschern als am aussichtsreichsten angesehen werden. Dabei sollte es sich um Teilchen mit einer Ruhemasse von über $\sim 100GeV$ handeln [Sch06a](Massen können in der Teilchenphysik dank der Relativitätstheorie auch in Energie-Einheiten angegeben werden). Für den Zweck dieser Arbeit soll davon ausgegangen werden, dass Dunkle Materie zumindest zum größten Teil aus WIMPs besteht, da diese auch von den meisten Simulationen als Modell verwendet werden.

Der Vollständigkeit halber sei noch angemerkt, dass neben der Theorie der Dunklen Materie noch weiter Hypothesen existieren, welche die beobachteten Phänomene erklären können. Die bekannteste davon wird MOND (Modified Newtonian Dynamics) genannt [Mil83]. Diese setzt auf eine Modifikation der Newtonschen Gleichungen ([New87]), um die Problematik zu lösen, was ohne die Einführung von irgendwelcher Materie geschieht. In der Tat können somit auch weitere Phänomene erklärt werden, wie es mit Dunkler Materie nicht möglich ist [Sch06a]. Jedoch schrecken die meisten Physiker vor einer Modifikation von als gesichert geltenden Gesetzen zurück. Daher wird wohl auch in Zukunft auch weiter von Dunkler Materie als ein Hauptbestandteil unserer Galaxis ausgegangen werden.

## 2.4 Halos

Da angenommen wird, dass es sich bei Dunkler Materie größtenteils um einzelne Elementarteilchen handeln muss, deren Effekt allein für sich genommen nicht besonders groß ausfällt, können nur größere Ansammlungen dieser Partikel relevante Effekte auslösen. So befinden sich um jedes Galaxiezentrum herum große Ansammlungen Dunkler Materie. Diese Ansammlungen um die Zentren herum werden Halos genannt. Die Halos können sich dabei erheblich weiter in den Weltraum erstrecken als die sichtbare Materie dies tut. Hier stellt sich nun die Frage, wie weit nach außen sich ein Halo erstreckt. Das ist sehr stark definitionsabhängig. Zudem ist eine Definition vonnöten, wie viel Dunkle Materie überhaupt vorhanden sein muss, um als Halo zu gelten.

Für diese Defintion wird gerne die kritische Dichte des Universums $\rho_{cr}$ (vgl. Abschnitt 2.2.5) als Referenz verwendet. Eine der beliebtesten Definitionen beruht auf der $200$-fachen kritischen Dichte. Der genau Zahlenwert ist jedoch von den gewählten Universums-Parametern abhängig. Also wird der Bereich dem Halo zugerechnet, in dem die mittlere Dichte der Dunklen Materie die $200$-fache kritische Dichte des Universums zu diesem Zeitpunkt überschreitet [Sch06a]. Der Radius, in dem dies der Fall ist, wird als Virialradius $R_{vir}$ oder als $R_{200}$ bezeichnet. Analog zum Virialradius wird die viriale Masse $M_{vir}$ definiert, welche die vorhandene Masse an Dunkler Materie innerhalb von $R_{vir}$ darstellt. Analog zu $R_{200}$ werden auch andere Radii $R_i$ definiert, die eben die $i$-fache kritische Dichte besitzen, um verschiedene Bereiche innerhalb der Halos gegeneinander abzugrenzen.

Die Dichte der Dunklen Materie um ein Galaxiezentrum herum ist natürlich nicht konstant. Sie fällt nach außen hin immer weiter ab. Eine Funktion, die diesen Dichteabfall beschreibt wurde von Navarro, Frenk u. White postuliert (NFW-Profil [NFW96]). Diese Funktion hat folgende Form [Sch06a]:

$$\rho(r) = \frac{\rho_s}{\frac{r}{r_s}\left(1+\frac{r}{r_s}\right)^2} \tag{2.12}$$

Hier bezeichnet $r$ den Radius um das Zentrum und $\rho_s = \rho_{cr} \cdot \delta_c$ ist eine Konstante, welche die Amplitude des Dichteprofils beschreibt. Da die beobachtete Dichte ab einem bestimmten Radius stärker zu fallen beginnt, bezeichnet $r_s$ genau jenen Radius. Bei Radii $r \ll r_s$ fällt die Kurve annähernd linear, bei $r \gg r_s$ fällt sie annähernd kubisch. Die Parameter $\rho_s$ und $r_s$ variieren von Halo zu Halo und sollten immer den Beobachtungen angepasst werden [Sch06a][NFW96].

Neben dem NFW-Profil existieren noch weitere Möglichkeiten, die Dichte zu beschreiben. So ist auch das Einasto-Profil ([Ein65]) eine gute Näherung an die beobachteten Werte. Dieses kann wie folgt beschrieben werden [Sch06b]:

$$\rho(r) = \rho_s \exp\left(\frac{-2}{\alpha}\left[\left(\frac{r}{r_s}\right)^\alpha - 1\right]\right) \tag{2.13}$$

Die Konstanten $\rho_s$ und $r_s$ sind hier analog zum NFW-Profil. Der Wert $\alpha$ kann einfach auf $0{,}17$ festgelegt werden, könnte aber bei ausgefalleneren Halo-Formen leicht variieren [Sch06b].

Neben Galaxien existieren auch größere Strukturen im Universum, Galaxiehaufen beispielsweise. Auch diese besitzen ihre eigenen Halos aus Dunkler Materie. Da aber eben Galaxien Bestandteil von Galaxienhaufen sind, passt der stetige Abfall der Dichte-Modelle nach außen hin nicht immer perfekt zu den Beobachtungen. Innerhalb des größeren Halos können mehrere kleine existieren, die Sub-Halos genannt werden. Diese werden besonders gut in den Ergebnissen von kosmologischen Simulationen sichtbar. Hierzu sei auf Abbildung 2.3 verwiesen. Dort werden simulierte Dichteverteilungen dargestellt. Im oberen Galaxiehaufen werden die Sub-Halos in Form von einzelnen Galaxien deutlich sichtbar. Auch die untere Dichteverteilung, die zu einer Galaxie gehört, weist deutliche Substrukturen auf. Bis jetzt wurden solche Substrukturen im Falle von Galaxien nicht in dieser Häufigkeit beobachtet. Das deutet entweder auf zu schwache Detektionsverfahren oder einen Fehler in den aktuellen Theorien hin [Sch06a].

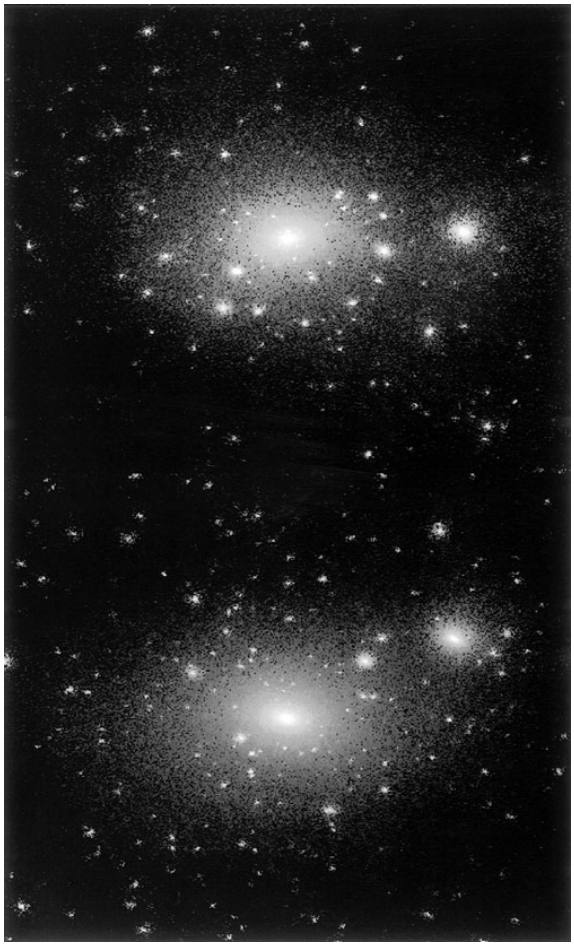


**Abbildung 2.3:** Simulierte Dichteverteilungen zweier Halos. Die obere stellt eine Dichteverteilung eines Galaxiehaufens dar, die untere die einer Galaxie (Längen nicht maßstabsgetreu). Die Substruktur ist sehr gut in Form von helleren Punkten sichtbar (Abbildung aus [Sch06a][MGG+99]).

Wie genau Halos in Verteilungen Dunkler Materie detektiert werden können, wird näher in Kapitel 3 beschrieben. Auch auf die Detektion von Sub-Halos wird dort eingegangen.

## 2.5 Kosmologische Simulationen

Wie unter anderem in Abschnitt 2.2.1 bereits erörtert, sind unsere Möglichkeiten, das uns umgebende Weltall zu beobachten, sehr beschränkt. Auch unser Verständnis des uns umgebenden Alls weist noch sehr große Lücken auf. Hierbei können numerische Simulationen eine große Hilfe sein. Zum einen können sie aufzeigen, was in Bereichen des Alls vor sich

gehen könnte, die wir nicht direkt einsehen können. Zum anderen bieten Simulationen die Möglichkeit, Hypothesen über bestimmte physikalische Begebenheiten zu überprüfen und mit der Realität abzugleichen. So konnte zum Beispiel ausgeschlossen werden, dass der Großteil der Dunklen Materie aus heißer Dunkler Materie besteht, da sich sonst die galaktischen Strukturen komplett anders entwickelt hätten [Sch06a].

Zum Vergleich mit den Resultaten der Simulationen werden die Daten unterschiedlicher Surveys herangezogen, die mit verschiedensten Weltraumteleskopen und anderen Gerätschaften ein möglichst lückenloses Bild des Weltalls zu kreieren versuchen. Zu diesen Surveys gehört unter anderem die Planck-Mission ([AAA+14]), welche mit Hilfe des Planck-Satelliten unter anderem die Mikrowellen- Hintergrundstrahlung und die optischen Eigenschaften extragalaktischer Strukturen in hoher Auflösung untersucht und aufzeichnet. Die etwas weiter in der Vergangenheit durchgeführte Sloan Digital Sky Survey (SDSS, [YAAJ+00]) hatte insbesondere zum Ziel, besonders helle Galaxien und Quasare zu kartografieren. Von derselben Forschungsgruppe wurden in den vergangenen Jahren weitere Surveys mit ähnlichen Zielen durchgeführt (u. a. SDSS-III [EWA+11] inklusive der Baryon Oscillation Spectroscopic Survey BOSS [DSA+13]).

Beim Abgleich der Simulationen und der Resultate der Surveys in den vergangenen Jahrzehnten wurde bald klar, dass insbesondere das $\Lambda$CDM-Modell (Lambda Cold Dark Matter, vgl. u. a. [Sch06b]) gute Vorhersagen für die Struktur des Universums liefert. Das $\Lambda$CDM-Modell vermag dies mit der Hilfe von nur sechs grundlegenden Parametern. Aufgrund der Einfachheit und der recht guten Vorhersagen wird es deshalb auch als das Standardmodell der Kosmologie bezeichnet. Es geht davon aus, dass die Materie zu großen Teilen sowohl kalt als auch dunkel ist. Viele Simulationen gehen sogar so weit, nur kalte Dunkle Materie zur Simulation heranzuziehen [Sch06b], was in Anbetracht der Häufigkeit selbiger nur gerechtfertigt ist. Dies hat zudem auch praktische Gründe, da die Berechnung von nur einer Materie-Art weitaus weniger Leistung erfordert, als wenn man alle in Betracht ziehen würde.

Eine solche Simulation wird wie folgt aufgebaut: Die initiale Masse der Dunklen Materie wird in ausdehnungslose einzelne Massepunkte gleicher Masse aufgeteilt. Diese Massepunkte können durchaus das Gewicht mehrerer Tausend Sonnenmassen besitzen. Je mehr Rechenleistung bzw. Speicher zur Verfügung steht, desto mehr und somit auch umso leichtere Partikel können in Kauf genommen werden. Simuliert wird ein Ausschnitt des Universums, der möglichst repräsentativ zu sein hat und groß genug sein muss, um die größten bekannten Strukturen im Universum in sich aufzunehmen. Hierbei werden im Normalfall mitbewegte Koordinaten (vgl. Abschnitt 2.2.2) innerhalb eines würfelförmigen Gebiets gewählt, da dies fortlaufende Korrekturen der Partikelpositionen durch die Ausdehnung des Weltalls überflüssig macht. Die initiale Verteilung der Massepunkte hat dann dem linearen Leistungspektrum $P(k,z)$ (vgl. [Sch06a]) zu entsprechen, welches, sehr simpel ausgedrückt, die Verhältnisse zwischen großen und kleinen Strukturen beschreibt.

Viele Modelle nehmen als einzig wirkende Kraft während der Simulation die Gravitation zwischen den einzelnen Partikeln an [Sch06a]. So könnte dann für jedes Partikel die auf

es wirkenden Kräfte, die von den anderen Massepunkten stammen, summiert werden. Mit Hilfe dieser resultierenden Kraft kann dann die Beschleunigung des Partikels und folglich auch die Bewegungsrichtung und neue Position bestimmt werden. Meist sind solche direkten Berechnungen aber nicht praktikabel, da allein schon die Anzahl der Partikel riesig ist und die Berechnung aller Kräfte naiv quadratische Laufzeit besitzt. Daher wird von den meisten Simulationen ein Gitter eingeführt, auf dessen Gitterpunkte alle Partikel in der Umgebung projiziert werden. Dann reicht es aus, die Berechnungen nur für die Gitterpunkte durchzuführen. Die Simulationen des Millenium-Runs (vgl. [SET+05]) verwenden ein solches Gitter, dies ist jedoch in der hier verwendeten Simulation (vgl. [SWT+14]) nicht der Fall. In den DarkSky-Simulationen wird die so genannte 2HOT-Methode verwendet (vgl. [War13]). Dabei handelt es sich um hoch optimierten Octree-Code, welcher eine sehr effiziente Berechnung der Kräfte erlaubt.

Wichtig für die Kraftberechnung sind auch die Randbedingungen, von denen ausgegangen wird. In kosmologischen Simulationen werden sogut wie immer periodische Randbedingungen gewählt, da sich Dinge, die ausserhalb des Simulationsgebiets angenommen werden, physikalisch korrekt verhalten müssen. Periodische Randbedingungen leisten dies, denn alles, was man außerhalb des simulierten Würfels annimmt, hat man in dessen Inneren ja schon korrekt simuliert. Vorsicht ist auch bei der Wahl der Zeitschritte geboten, da weite Bewegungssprünge von Partikeln innerhalb sehr dichter Regionen sehr schnell zu numerischen Instabilitäten führen.

Resultat der Simulation ist dann eine lange Liste von Partikelpositionen, -geschwindigkeiten und -beschleunigungen über die Zeit hinweg. Mit deren Hilfe können dann weitere Berechnungen durchgeführt werden. So können etwa Dichteverteilungen bestimmt oder auch bestimmte Strukturen ausgemacht werden. Zu diesen Strukturen gehören auch die Halos, mit deren Detektion wir uns nun im nächsten Abschnitt näher auseinandersetzen wollen.

# 3 Detektion von Halos

Mit vorliegenden Simulationsresultaten kann nun begonnen werden, aus den Positionen und anderen bekannten Werten der Dunklen Materie Halos zu detektieren. Zunächst wird in Abschnitt 3.1 ein Überblick über die vorhandenen Daten gegeben, die zur Halo-Detektion benutzt werden können. In Abschnitt 3.2 werden dann die Grundideen selbiger diskutiert, woraufhin in 3.3 die Ansätze verschiedener bekannter Halo-Finder diskutiert werden. Daraufhin wird in Sektion 3.4 speziell auf den Rockstar-Algorithmus [BWW13] eingegangen, welcher von den Autoren der DarkSky-Simulation [SWT+14] verwendet wurde und dessen Resultate den Simulationsresultaten beiliegen. Anschließend wird der Algorithmus von Widanagamaachchi et al. [WBS+14] vorgestellt, der im Rahmen dieser Arbeit implementiert wurde, um selbst Halos detektieren zu können. Abschließend wird in 3.6 auf die Implementierung desselbigen eingegangen.

## 3.1 Eingabedaten

Wie bereits erwähnt, stammen die Daten, die hier benutzt werden, aus der DarkSky-Simulation, welche im Jahre 2014 durchgeführt wurde. Zusätzlich zu den direkt veröffentlichten größeren Datensätzen wurden im Rahmen des SciVis-Contests 2015 der IEEE [HGTS15] weitere Daten herausgegeben, welche eine Nachverfolgung der Partikel über knapp $100$ Zeitschritte erlauben. Die Datensätze sind in Tabelle 3.1 aufgelistet. Natürlich wurden, je nach Datensatz, mehrere Zeitschritte (ca. $500 - 3500$, je nachdem) berechnet, nur eigentümlicherweise wurde pro durchgeführter Simulation nur ein Zeitschritt veröffentlicht.

**Tabelle 3.1:** Datensätze der DarkSky-Simulation

| Datensatz | $\sqrt[3]{N}$ | Würfellänge $L$ $[h^{-1}Mpc]$ | Partikelmasse $M_p$ $[h^{-1}M_\odot]$ |
|---|---|---|---|
| ds14_a | $10240$ | $8000$ | $3{,}9 \cdot 10^{10}$ |
| ds14_g_1600_4096 | $4096$ | $1600$ | $4{,}9 \cdot 10^{9}$ |
| ds14_g_800_4096 | $4096$ | $800$ | $6{,}1 \cdot 10^{8}$ |
| ds14_g_200_2048 | $2048$ | $200$ | $7{,}6 \cdot 10^{7}$ |
| ds14_g_100_2048 | $2048$ | $100$ | $9{,}5 \cdot 10^{6}$ |
| ds14_scivis_0128 | $128$ | $62{,}5$ | $1{,}38 \cdot 10^{10}$ |

Die einzige Ausnahme bilden die Contest-Daten (ds14_scivis_0128). Von diesem Datensatz sind Partikeldaten aus $100$ Zeitschritten verfügbar. Die wichtigsten globalen Parameter aller Datensätze wie der kritischen Dichte $\rho_{cr}$ und der Hubble-Konstante $H_0$ wurden bereits in Abschnitt 2.2.5 angegeben. Pro simulierten Partikel sind pro Zeitschritt folgende Werte vorhanden:

- Position $\vec{x}$ in mitbewegten Koordinaten.
- Geschwindigkeit $\vec{v}$ relativ zu den mitbewegten Koordinaten.
- Eindeutige 64-Bit ID.

Im Contest-Datensatz kommen zu diesen drei Werten noch die Beschleunigung $\vec{a}$ und das Potenzial $\phi$ hinzu, welche hier jedoch nicht weiter beachtet werden. Zur Detektion von Halos sind zumeist auch keine weiteren Daten vonnöten, die meisten Verfahren kommen allein mit Position und Geschwindigkeit aus, manche benötigen sogar nur die Position (abgesehen von den globalen Werten wie der Partikelmasse natürlich).

Die Partikeldaten wurden im SDF-Format veröffentlicht (Self-Describing File). Dieses besteht normalerweise aus einem ASCII-Header, welcher alle nötigen globalen Parameter enthält, gefolgt von den Binärdaten für alle Partikel. Wie der Name des Dateityps schon andeutet, ist der Zweck dieses Formats grundlegend darin begründet, dass die Beschreibung der abgespeicherten Binärdaten in der Datei selber vorgenommen wird. Dies erspart bei Änderung der Struktur der Rohdaten längeren Programmieraufwand für den Datei-Parser. Zum Lesen der so abgespeicherten Daten kann die SDF-Library [SW14] verwendet werden, jedoch wurde, nicht zuletzt weil sich die Struktur der Rohdaten nicht mehr ändert, im Rahmen dieser Arbeit ein eigener Parser implementiert. Dieser ignoriert alle für uns irrelevanten Werte.

Neben den Partikeldatensätzen wurden auch Halo-Kataloge veröffentlicht, welche mit Hilfe des Rockstar-Algorithmus (vgl. Abschnitt 3.4) erstellt wurden. Hierbei ist die Beschreibung der Halos im Falle des Contest-Datensatzes im Gegensatz zum Rest weitaus ausführlicher. Für eine nähere Beschreibung der Halo-Kataloge sei auf Abschnitt 4.2 verwiesen.

## 3.2 Grundideen der Halo-Detektion

Bei der Detektion von Halos Dunkler Materie haben sich in den letzten Jahrzehnten zwei grundlegende Arten von Verfahren herauskristallisiert. Während sich die so genannten Spherical-Overdensity-Verfahren (SO, basierend auf [PS74]) vom Prinzip her stark an der in Kapitel 2.4 erläuterten Definition von Halos orientieren, ist der Bezug der zweiten Art, der Friends-of-Friends-Verfahren (FOF, basierend auf [DEFW85]) zu dieser Definition weniger offensichtlich. Daher wollen wir zunächst die SO-Verfahren erläutern.

### 3.2.1 Spherical Overdensity

Die Definition der Halos, die hier vorgenommen wurde, führt einen quasi direkt zum Prinzip der Spherical Overdensity [PS74]. Die Idee ist nach lokalen Maxima im Dichtefeld zu suchen. Diese sind dann, bei ausreichender Dichte, Zentren von Halos. Um diese Zentren werden, immer größer werdend, Kugeln aufgespannt. Das geschieht so lange bis die durchschnittliche Dichte innerhalb der Kugel einen gewählten Wert unterschreitet (hier z. B. $200\rho_{cr}$). Der Kugelradius, bei dem diese Dichte gerade nicht unterschritten wird, beschreibt dann den Radius des Halos. Das Prinzip wird auch in Abbildung 3.1 zweidimensional veranschaulicht.

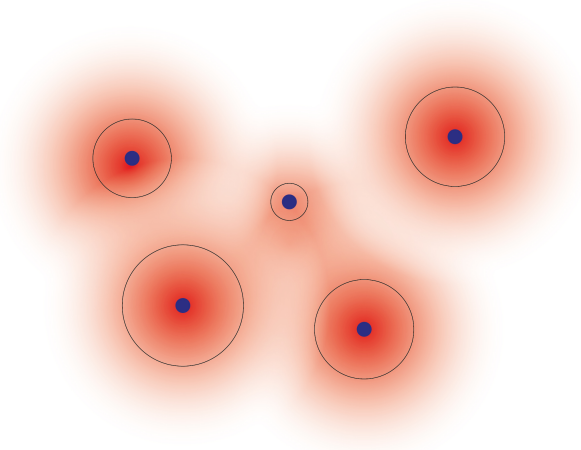


**Abbildung 3.1:** Grundprinzip der Spherical-Overdensity-Verfahren. Rot bedeutet hohe Dichte, weiß geringe. Blaue Punkte bezeichnen lokale Maxima der Dichte, die Radii der Halos sind in schwarz eingezeichnet.

Um die lokalen Maxima (blau) spannen sich die Halos auf bis ihre durchschnittliche Dichte (wird in rot dargestellt) auf einen bestimmten Wert absinkt. Es ist offensichtlich, dass an den Stellen der lokalen Maxima eine gewisse Grunddichte, also hier $200\rho_{cr}$ herrschen muss, damit überhaupt ein Halo detektiert werden kann. Da in den von uns verwendeten Simulationen die Dunkle Materie der einzige Faktor ist, der zur Dichte beiträgt, muss ein solches Maximum auf eine Partikel-Position fallen. Auch die Detektion von Sub-Halos ist so kein Problem. In diesem Fall befindet sich ein lokales Maximum, welches einen Halo aufspannt, innerhalb des Virialradius eines anderen, größeren Halos.

Das Dichtefeld kann auf viele verschiedene Arten bestimmt werden. Zum einen ist es möglich, den Raum in einem Gitter zu diskretisieren und die Dichte an jedem Gitterpunkt zu bestimmen. Dies geschieht durch Summierung gewichteter Abstände zu diesen Gitterpunkten. Die Maxima können dann z. B. mit Hilfe finiter Differenzen bestimmt werden. Eine weitere Möglichkeit wäre die Verwendung von Smoothed Particle Hydrodynamics (SPH, vgl.

[Luc77][GM77]). Damit können die Dichten an allen Partikelpositionen bestimmt werden. Das Prinzip ist hier ebenfalls gewichtete Addition von den Partikelmassen der Umgebung. Da sich die Maxima an Partikelpositionen befinden müssen, kann dies unter Umständen ausreichen. Werte an anderen Positionen könnten dann z. B. mittels Interpolation bestimmt werden.

Einer der Nachteile dieser Methode ist die so gut wie immer kugelförmige Struktur der Halos. Dies geschieht selbst an Stellen, wo eher längliche Strukturen, wie etwa lang gezogene Ellipsoide, besser zur Masseverteilung passen würden. Dies muss bei der Wahl dieser Methode bedacht werden.

### 3.2.2 Friends-of-Friends

Die Friends-of-Friends-Methode ([DEFW85]) basiert auf der Idee, dass eine hohe Dichte durch viele, sich nah aneinander befindende Massepunkte ausgelöst wird. Darum wird einfach eine Länge $l$ gewählt, welche Linking Length genannt wird. Besitzen zwei Partikel einen kürzeren Abstand als $l$, werden sie demselben Halo zugerechnet. Praktischerweise würde man solche Halos nur ab einer bestimmten Partikelanzahl bzw. einer bestimmten Mindestmasse akzeptieren. Dies wird in Abbildung 3.2 gezeigt.

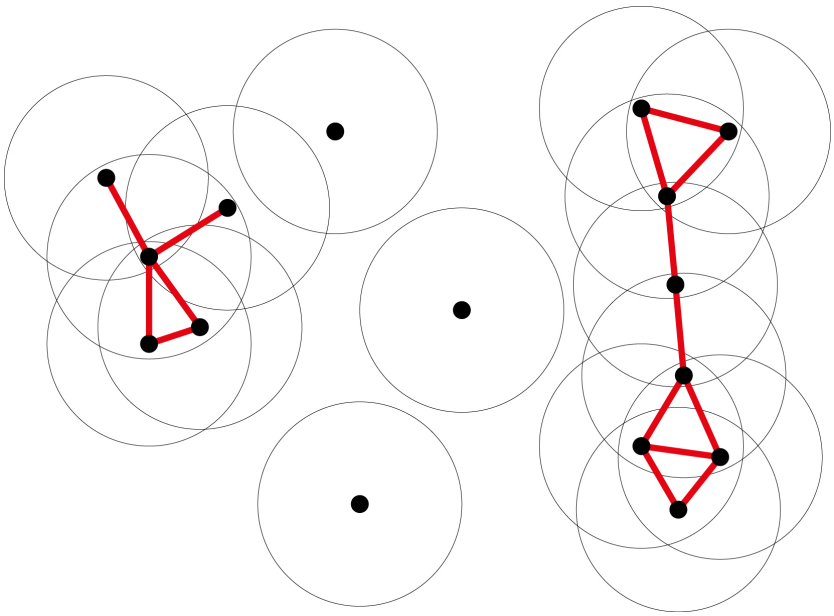


**Abbildung 3.2:** Grundprinzip der Friends-of-Friends-Verfahren. Schwarze Punkte sind die Partikelpositionen. Die größeren Kreise besitzen den Radius der Linking Length. Alle Partikel, die weniger Abstand als die Linking Length besitzen, also innerhalb der Kreisradii liegen, werden zu einem Halo zusammengenommen (rot).

Alle Partikel (schwarze Punkte), die sich innerhalb der Linking Length um ein anderes Partikel befinden, werden demselben Halo zugerechnet (rote Linien). Existiert also ein Pfad über solche Halo-Verbindungen von einem Partikel zu einem anderen, gehören die betroffenen Partikel zum selben Halo.

Die kritische Frage hier ist, wie lang die Linking Length $l$ sein soll. Eine Idee ist dies mit der Dichte zu verknüpfen. Diese Dichte soll von maximal zwei Partikeln verursacht werden, damit auch Halos aus nur zwei Partikeln existieren können. Eine Möglichkeit ist die Summers et al. [SDE95]. Bei dieser wird einfach in einer Kugel mit Radius der Linking Length dafür gesorgt, dass in dieser die Dichte einem bestimmten Wert entspricht. Dies führt zu der Gleichung [SDE95]:

$$\rho = \frac{2 \cdot M}{\frac{4}{3}\pi l^3} \tag{3.1}$$

$M$ bezeichnet hier die Masse eines Partikels. Setzt man z. B. $\rho = 200\rho_{cr}$ und formt die Gleichung nach $l$ um, erhält man die Linking Length die benötigt wird, um dem Halo-Kriterium zu genügen. Man kann den Wert von $\rho$ auch so ändern, dass mehrere Partikel notwendig sind, um einen Halo zu bilden.

Neben der Abhängigkeit von der Dichte kann auch der durchschnittliche Partikelabstand als ein Maß für eine gute Linking Length dienen. Viele Arbeiten nehmen einen Bruchteil $b$ des durchschnittlichen Partikelabstands als $l$ an [BWW13]. Häufige Werte für $b$ sind $b = 0{,}15$ oder $b = 0{,}2$ [MKDG11][BWW13].

Die so detektierten Halos besitzen fast die gegensätzlichen Eigenschaften zu denen der Spherical-Overdensity-Methode. Existieren große Materie-Ansammlungen, werden diese zwar ähnlich gut erkannt, nur tendiert FOF dazu, längliche Strukturen zu finden oder mehrere Halos zu einem zu verbinden, weil sich einige wenige Partikel zufällig dazwischen befinden. Bei ausgefalleneren Strukturen der Halos ist FOF daher eventuell besser geeignet, da das Verfahren keine sphärische Form forciert. Sub-Halos können auch erkannt werden, indem man die Linking Length innerhalb von detektierten Halos anpasst und innerhalb dieser eine neue Detektion durchführt.

Die hier verwendeten Verfahren, der Rockstar-Algorithmus und die Methode von Widanagamaachchi et al., basieren beide auf dem FOF-Verfahren.

## 3.3 Überblick über verschiedene Halo-Finder

Es existiert eine Vielzahl unterschiedlichster Halo-Finder. Die Arbeiten von Knebe et al. [KKM+11] und Onions et al. [OAB+13] bieten hierbei einen guten Überblick. Während erstere einen Vergleich mehrerer Halo-Finder durchführt, konzentriert sich die zweite darauf,

wie gut die untersuchten Halo-Finder Substrukturen in den Halos erkennen können. Die untersuchten Halo-Finder sind bei beiden Arbeiten größtenteils dieselben.

Im Folgenden sollen einige Halo-Finder vorgestellt werden.

### 3.3.1 SO-Finder

Die zumeist einfachste Möglichkeit, einen bereits existierenden Algorithmus schneller zu machen, ist Parallelisierung. Dies taten Sutter und Ricker mit dem Spherical-Overdensity-Algorithmus. Ihre Variante, der Parallel Spherical Overdensity Algorithmus (PSO [SR10]), orientiert sich stark an der Grundmethode. Er ist aber in der Lage, bestimmte Halo-Eigenschaften ohne Post-Processing zu berechnen. Das ist möglich, weil sich dieser das Gitter, mit dem die meisten Simulationen arbeiten, zunutze macht und direkt auf diesem arbeitet.

Der Methode, die Dichten nur an den Partikelpositionen zu bestimmen, bedienen sich Klypin und Holtzmann mit ihrem Verfahren. Der Bound Density Maxima Algorithmus (BDM [KH97]) bestimmt die Dichte durch Gewichtung der Massen der $N$ nächsten Partikel, wobei $N$ frei wählbar ist. Daraufhin wird um jedes Dichtemaximum eine Kugel erzeugt, welche eine bestimmte Mindestmasse an Partikeln enthalten soll. Diese Kugel beschreibt dann einen Halo.

Um Sub-Halos detektieren zu können, sind oft weitere Überlegungen nötig. Der Amiga Halo Finder (AHF [KK09]) von Knollman und Knebe bedient sich dazu eines rekursiv gröber werdenden Gitters, auf dem das Dichtefeld berechnet wird. Damit ist es möglich, neben einfachen Sub-Halos auch Sub-Sub-Halos und so weiter zu erkennen.

### 3.3.2 FOF-Finder

Parallelisierung ist auch bei den FOF-Halo-Findern eine gute Möglichkeit, mehr Geschwindigkeit herauszuholen. So unterteilen Rasera et al. mit dem Parallel FOF (PFOF [RAC+10]) den Raum in gleich große Würfel und berechnen die Halos in diesen dann parallel. Bei Halos, die über die Würfelgrenzen herausragen könnten, werden von den benachbarten Regionen die Partikelpositionen abgefragt.

Ein Möglichkeit, die Qualität der FOF-Halos zu verbessern, ist es, mehr Parameter als nur die Positionen zu betrachten. Sowohl der 6DFOF-Algorithmus von Zemp et al. [ZMS+08] als auch der Rockstar-Algorithmus von der Gruppe um Behroozi ([BWW13], vgl. Abschnitt 3.4) bedienen sich dieser Möglichkeit. 6DFOF verwendet zusätzlich zur Position noch die Geschwindigkeit der Partikel, um die Nähe zu bestimmen. Zusätzlich zur normalen räumlichen Linking Length wird dazu eine Linking Length im Geschwindigkeits-Raum eingeführt. Partikel müssen dann sowohl im Ortsraum als auch im Geschwindigkeitsraum nahe beieinander liegen, um zu einem Halo zusammengefügt zu werden. Diese Methode verhindert, dass

Partikel, die sich nur zufällig am selben Ort befinden, sich jedoch innerhalb der nächsten Zeitschritte sehr weit entfernen werden, Halos zugerechnet werden. Insbesondere wenn Halos über mehrere Zeitschritte hinweg detektiert werden sollen, ist dies wichtig, da sonst einmal zugewiesene Partikel schnell wieder aus ihren Halos herausgerissen werden können, weil sie sich wegbewegt haben. Diese Klasse von Halo-Findern wird Phasenraum-Halo-Finder genannt.

Es existieren viele Möglichkeiten, um einen FOF-basierten Algorithmus zu beschleunigen oder zu realisieren. So verwenden Habib et al. einen kD-Baum, um die Suche nach benachbarten Partikeln zu beschleunigen [HPL+09]. Gottlöber und Yepes konstruieren einen minimalen Spannbaum aller Partikelpositionen, um mit deren Hilfe die Halos zu bestimmen. Sogar Sub-Halo-Detektion ist mit dieser Methode möglich. Neyrinck et al. verwenden mit VOBOZ (VOronoi BOund Zones, [NGH05]) Voronoi-Diagramme, um nahe Partikel zu finden.

Mit SKID (Spline Kernel Interpolative Denmax) von Stadel [Sta01] existiert auch eine Methode, welche zunächst die Partikel mit der höchsten Dichte identifiziert und daraufhin die Halo-Zugehörigkeiten mit einer simplen SPH-Simulation (vgl. [Luc77][GM77]) bestimmt. Dabei werden alle Partikel, die kein lokales Maximum darstellen, von den lokalen Maxima angezogen. Nach mehreren Zeitschritten haben sich dann Gruppen von Partikeln gebildet, welche die Halo-Zugehörigkeiten widerspiegeln.

Im Folgenden wollen wir uns auf zwei Algorithmen konzentrieren, welche beide der FOF-Kategorie zuzuordnen sind. Der Rockstar-Algorithmus wurde von den Machern der DarkSky-Simulation verwendet, um ihre Halo-Kataloge zu erstellen. Der FOF-Algorithmus mit variabler Linking Length [WBS+14] ermöglicht es Halos für verschiedene Werte von $l$ gleichzeitig zu bestimmen und wurde im Rahmen dieser Arbeit implementiert. Diese beiden Halo-Finder sollen in den folgenden zwei Abschnitten näher erläutert werden.

## 3.4 Rockstar-Algorithmus

Beim Rockstar-Algorithmus von Behroozi et al. ([BWW13]) handelt es sich um einen Friends-of-Friends Halo-Finder, der im Phasenraum operiert. Rockstar steht hierbei für Robust Overdensity Calculation using K-Space Topologically Adaptive Refinement. Zusätzlich zu den sechs Phasenraum-Dimensionen wird auch noch die Zeit als siebte Dimension berücksichtigt, um zuverlässiges Tracking von Halos über die Zeit hinweg zu gewährleisten. Dabei ist der Algorithmus so parallel gehalten, dass er auch sehr gut mit größeren Datensätzen skaliert.

Der Algorithmus besteht aus mehreren Phasen, die nacheinander durchlaufen werden.

1. **Unterteilung** Um die Partikel in sinnvoll zu parallelisierende Einheiten zu unterteilen, wird initial ein normaler 3D-FOF-Algorithmus mit relativ großer Linking Length durchgeführt. Dieser Prozess ist recht ähnlich zum PFOF-Algorithmus (vgl. Abschnitt

3.3.2) und läuft ebenfalls parallel ab. Als Linking Length wird typischerweise das $0{,}28$-fache des mittleren Partikelabstands gewählt. Resultat dieses Abschnitt sind große Gruppen von Partikeln, die sich räumlich nah beieinander befinden. Diese werden dann für den nächsten Schritt an alle verfügbaren Prozessoren verteilt. Der nächste Schritt wird dann für jede dieser Gruppen durchgeführt.

**2. Normalisierung** Für jede Gruppe wird die durchschnittliche Partikelposition und die Geschwindigkeitsdispersion berechnet. Jede Partikelposition wird durch die durchschnittliche Partikelposition normalisiert. Dasselbe geschieht mit den Geschwindigkeiten und der Geschwindigkeitsdispersion. Dies gewährleistet, dass in allen Gruppen ein ähnliches Koordinatensystem verwendet wird.

**3. Phasenraum-FOF** Für jede der normalisierten Gruppen wird nun ein Phasenraum-FOF durchgeführt. Die Linking Length wird dabei adaptiv bestimmt. Dies geschieht, indem für jedes Partikel $p_1$ die Distanz zu seinem nächsten Nachbarn $p_2$ berechnet wird. Dabei wird folgende Distanzmetrik verwendet [BWW13]:

$$d(p_1, p_2) = \sqrt{\frac{|\vec{x_1} - \vec{x_2}|^2}{\sigma_x^2} + \frac{|\vec{v_1} - \vec{v_2}|^2}{\sigma_v^2}} \tag{3.2}$$

Die Linking Length wird so gewählt, dass ca. $70\%$ der Partikel mit mindestens einem anderen Partikel eine Untergruppe bilden. Ein größerer Anteil würde zu zu vielen detektierten Sub-Halos führen, ein kleinerer würde zu viele wichtige Sub-Halos auslassen.

**4. Rekursion** Die Schritte $2$ und $3$ werden nun so lange rekursiv für jede Untergruppe durchgeführt, bis eine Gruppengröße von unter $10$ Partikeln erreicht ist. Dies gewährleistet die Detektion von Sub-Halos, Sub-Sub-Halos und so weiter.

**5. Halo-Zuordnung** Die untersten Rekursionsstufen werden als initiale Halos angesehen. Es wird nun von diesen untersten Stufen in der Hierarchie wieder nach oben gewandert. Während dieses Prozesses wird jedes Partikel dem ihm im Phasenraum nächsten Halo zugeordnet. Dies geschieht mit folgender Distanzmetrik für Halo $h$ und Partikel $p$ [BWW13]:

$$d(h,p) = \sqrt{\frac{|\vec{x_h} - \vec{x_p}|^2}{r_{dyn,vir}^2} + \frac{|\vec{v_h} - \vec{v_p}|^2}{\sigma_v^2}} \tag{3.3}$$

$$r_{dyn,vir} = v_{max} \cdot t_{dyn,vir} = \frac{v_{max}}{\sqrt{\frac{4}{3}\pi G \rho_{vir}}} \tag{3.4}$$

Hierbei ist ist $G$ die Gravitationskonstante, $\rho_{vir}$ entspricht der $360$-fachen Hintergrunddichte des Universums und $v_{max}$ ist die maximale Radialgeschwindigkeit eines

Partikels, die im Halo auftritt. Mit anderen Worten ist ein Partikel dann nah an einem Halo, wenn er eine ähnliche Bewegungsrichtung und -geschwindigkeit besitzt, räumlich nahe bei ihm ist und sich möglichst in seinem gravitativen Einfluss befindet (ausgedrückt durch $r_{dyn,vir}$). Sehr nah beieinander liegende Halos können auf dem Weg nach oben in der Hierarchie zusammengeführt werden, wenn sie, statistisch gesehen, nicht mehr sicher voneinander unterschieden werden können.

Existieren auf derselben Rekursionsebene mehrere Halos, die nicht zusammengeführt werden können, wird angenommen, dass die kleineren Halos Sub-Halos des größten Halos auf derselben Ebene sind. Halos werden dann immer demjenigen als Sub-Halos zugewiesen, der eine größere Masse besitzt und sich unter Verwendung von Gleichung 3.3 am nächsten befindet.

**Postprocessing** Ist die Halo-Zuordnung geschehen, werden zunächst ungebundene Partikel aus den Halos entfernt. Das können Partikel sein, die sich zu schnell oder in die falsche Richtung bewegen. Alle Halos mit einer Partikelzahl $< N$ werden nicht weiter beachtet, wobei $N = 20$ meist ein guter Wert ist. Ist das geschehen, werden die finalen Massen, Geschwindigkeiten und weitere Parameter der Halos berechnet. Es werden für jeden Halo Ellipsoide berechnet, welche die darin enthaltenen Partikel möglichst umschließen. Für eine Möglichkeit, wie dies geschehen kann, siehe Abschnitt 3.6.4.

Der Prozess des Findens von Halos ist nach Berechnung aller wichtigen Parameter abgeschlossen, jedoch wurde die Zeit noch nicht als Dimension berücksichtigt. Um Halo-Hierarchien über die Zeit hinweg zu erstellen, wird die obige Prozedur für jeden Zeitschritt durchgeführt. Ist das geschehen, wird für jeden Halo in einem Zeitschritt derjenige Halo gesucht, welcher mit dem aktuellen die meisten Partikel gemeinsam hat. Dieser ist dann der Vorgänger in der Zeit. Diese Prozedur kreiert somit Merger-Bäume für die Halos. Bei diesem Prozess werden auch Inkonsistenzen erkannt und entfernt, z. B. Halos, die verschwinden und plötzlich wieder auftauchen.

Rockstar liefert im Allgemeinen bessere Ergebnisse als die anderen vorgestellten FOF-Halo-Finder (vgl. [KKM+11], [OAB+13]). Insbesondere bei der Detektion von Sub-Halos werden weniger Fehler gemacht. Der Rockstar-Algorithmus hat jedoch wie alle anderen FOF-Halo-Finder dasselbe Problem: Will man eine andere Linking Length ausprobieren, muss der komplette Algorithmus erneut durchlaufen werden. Daher wird im nächsten Abschnitt eine Methode vorgestellt, wie dies nicht zwangsläufig der Fall sein muss.

## 3.5 Friends-of-Friends-Algorithmus mit variabler Linking Length

Die Linking Length ist ein essentieller Parameter für jeden Friends-of-Friends-Algorithmus. Wird sie zu groß gewählt, könnte nur ein einziger Halo detektiert werden, der aus allen Partikeln besteht. Bei zu kleiner Wahl werden meist nur sehr kleine Halos gefunden, die sehr wenige Partikel enthalten. Beides kann nicht den beobachteten Effekten im Weltall entsprechen. Wenn die bestmögliche Wahl für die Linking Length nicht initial bekannt ist, kann es recht schwer sein, den korrekten Wert zu finden, der nicht zu einem der beiden Extreme führt. Leider kann, abgesehen von theoretischen Vorüberlegungen, jeder Wert letztendlich nur ausprobiert werden. Dafür müsste man aber für jeden Wert den kompletten Prozess des Halo-Findens durchlaufen, was sehr lange dauern kann. Daher entwickelten Widanagamaachchi et al. eine Variante des Friends-of-Friends-Algorithmus, welcher komplette Linking Length-Bereiche auf einmal berechnen kann [WBS+14]. Dieser soll im Folgenden vorgestellt werden.

Der Algorithmus berechnet eine komplette Halo-Hierarchie, die, wenn eine bestimmte Linking Length angefragt wird, die Halos für diese Länge zurückgeben kann. Er benötigt zwei Längen als Eingabe. Eine minimale Linking Length $min_ll$ und eine maximale $max_ll$. Alle angefragten Längen dürfen nur zwischen diesen beiden liegen. Um das zu erreichen, werden alle Distanzen zwischen Partikeln berechnet, die maximal $max_ll$ lang sind. Diese werden dann zur Konstruktion der Halo-Hierarchie verwendet.

### 3.5.1 Berechnung der initialen Halos

Der erste Schritt besteht in der Berechnung von Halos mit Linking Length $min_ll$. Diese werden dann später mit größeren Linking Lengths erweitert. Dazu wird eine spatiale Datenstruktur verwendet, welche den Definitionsraum der Partikel in gleich große Würfel unterteilt. Diese Würfel müssen eine Kantenlänge von $\frac{min_ll}{\sqrt{3}}$ besitzen (im Dreidimensionalen, versteht sich), haben also eine Diagonale, die genau $min_ll$ lang ist. Ist $\frac{min_ll}{\sqrt{3}}$ kein ganzzahliger Teiler der Kantenlänge des Definitionsbereiches, so wird dieser gleichmäßig nach außen erweitert, bis dies der Fall ist.

Ist die Datenstruktur aufgebaut, so ist die Festlegung der initialen Halos gewissermaßen schon geschehen. Da jeder der Würfel der Datenstruktur eine Diagonale der Länge $min_ll$ besitzt, ist gewährleistet, dass alle innerhalb eines Würfels auftretenden Längen $\leq min_ll$ sind. Somit bilden alle Partikel innerhalb eines Würfels einen Halo mit der Linking Length $min_ll$. Das Resultat dieses Prozesses wird in Abbildung 3.3 gezeigt. Es ist offensichtlich, dass diese Halos nicht zwangsweise vollständig sein müssen, denn es können für jeden von ihnen Partikel in benachbarten Würfeln existieren, welche zu einem der Halo-Partikel Abstand $\leq min_ll$ besitzen. Das ist z. B. in den Boxen im rechten oberen Eck der Abbildung der Fall.

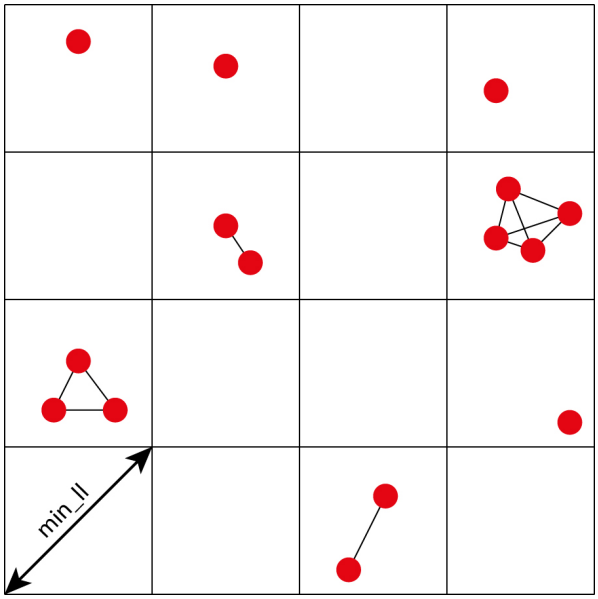



**Abbildung 3.3:** Initialer Schritt des FOF-Algorithmus mit variabler Linking Length. Jede Box hat Diagonale $min_ll$. Jedes Partikel (rot) ist mit dem ihm im selben Halo assoziierten Partikeln verbunden.

Das einzelne Partikel besitzt einen kleineren Abstand als $min_ll$ zu dem $4$-er Halo in der der Box darunter. Dies wird im nächsten Schritt, dem Aufbau der kompletten Hierarchie, korrigiert.

### 3.5.2 Hierarchieaufbau

Um eine Hierarchie von Halos aufzubauen, muss für jedes Partikel die jeweilige Nachbarschaft abgesucht werden. Es müssen nun alle Partikeldistanzen in Betracht gezogen werden, welche $\leq max_ll$ sind und nicht nur die, die $\leq min_ll$ sind. Der Radius, in dem um ein Partikel gesucht werden müsste, ist genau $max_ll$, jedoch würde ein Überprüfen aller Partikel zu einer garantiert quadratischen Laufzeit führen. Deshalb macht man sich die im letzten Schritt aufgebaute Datenstruktur zu Nutze und beschränkt sich bei dieser Suche nur auf Würfel in der Umgebung. Der Radius, der in diesem Fall abgesucht werden muss, beträgt dann $\left\lceil \frac{max_ll \cdot \sqrt{3}}{min_ll} \right\rceil$ Würfel. Das stellt sicher, dass mit mindestens $max_ll$ Radius um jede Zelle / jedes Partikel gesucht wird.

Die Idee ist nun, die initialen Halos, die im ersten Schritt generiert wurden, mit Hilfe der neu gefundenen Partikelrelationen zu erweitern. Ordnet man die Relationen nach deren Länge, kann so eine baumartige Struktur aufgebaut werden. Diese ist in Abbildung 3.4 links gezeigt. Die farbigen Knoten oben stellen die initialen Halos dar, die ihrerseits die Partikel repräsentieren, aus denen sie bestehen. Berücksichtigt man dann die gefundenen neuen

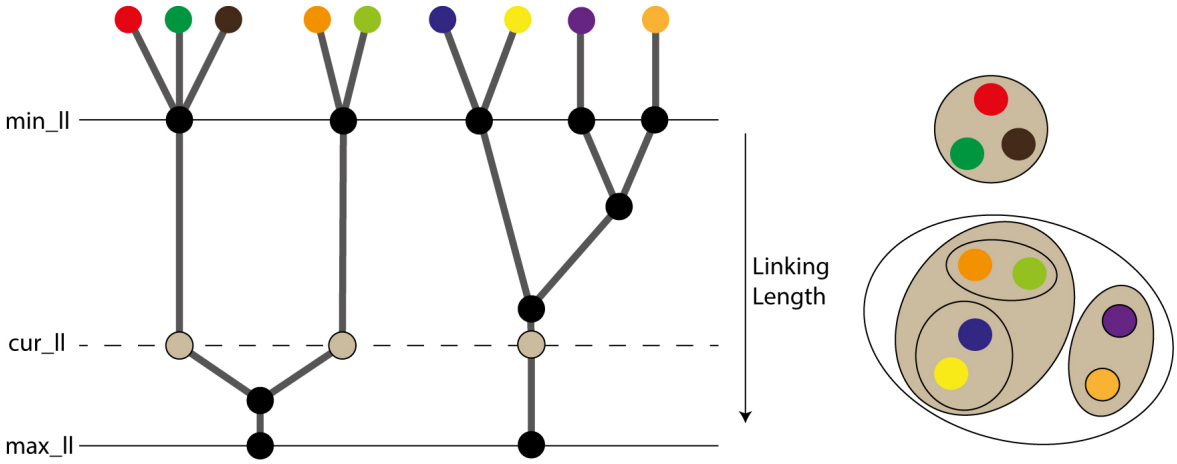



**Abbildung 3.4:** Halo-Hierarchie für Halos mit unterschiedlicher Linking-Length.

Partikelrelationen mit einer Länge von $\leq min_ll$, erhält man Kombinationen aus diesen initialen Halos, welche wirklich die gesuchten Halos für die Linking Length $min_ll$ darstellen. Mit steigender Linking Length können immer mehr Relationen berücksichtigt werden und immer mehr Halos kombinieren sich. Dies geschieht, bis $max_ll$ erreicht ist. Um diese Idee praktisch durchzuführen, ist es einfacher, den Bereich zwischen den beiden gegebenen Linking Lengths in diskreten Schritten abzuarbeiten. Das heißt, man berücksichtigt zunächst alle Relationen mit der Länge $\leq min_ll$. Im nächsten Kombinationsschritt werden alle Relationslängen $l$ mit $min_ll < l \leq min_ll + 0{,}001$ verwendet. Darauf folgend dann mit $min_ll + 0{,}001 < l < min_ll + 0{,}002$ usw., bis $max_ll$ erreicht ist. Die genau Schrittweite hängt natürlich von den gewählten Linking Lengths und den Präferenzen des Benutzers ab.

Wurden all diese Schritte durchgeführt erhält man eine Hierarchie wie in Abbildung 3.4. An diese kann man nun Anfragen mit beliebiger Linking Length $cur_ll$ stellen, wobei $min_ll \leq cur_ll \leq max_ll$ gilt. Dadurch werden nur die Partikelrelationen mit einer Länge von $\leq cur_ll$ berücksichtigt, was einem normalen FOF-Algorithmus mit Linking Length $cur_ll$ entspricht. Eine solche Anfrage wird in Abbildung 3.4 ebenfalls gezeigt (gestrichelte Linie). Diese resultiert in einer Ausgabe dreier Halos, welche im Baum durch die drei sandfarbenen Knoten bzw. die schwarzen Knoten direkt darüber repräsentiert werden. Die sandfarbenen Knoten sind eigentlich nicht Teil der Hierarchie, sie wurden hier nur der besseren Verständlichkeit halber eingefügt. Im Diagramm rechts werden die gefundenen Halo-Kombinationen von den drei sandfarbenden Flächen repräsentiert. Die schwarzen Ellipsen zeigen die gefundenen Halo-Kombinationen für alle berücksichtigten Linking Lengths auf, wobei jede Ellipse einem schwarzen Knoten in der Hierarchie links entspricht.

Steht die Halo-Hierarchie, ist der Algorithmus abgeschlossen. Es können natürlich noch weitere Parameter wie Halo-Massen, Massezentren, Geschwindigkeiten oder Rotationen im Nachhinein berechnet werden. Diese Werte gelten aber dann nur für die Linking Length, für

die sie berechnet wurden. Aus diesem Grund kann es praktikabler sein, diese Berechnungen erst durchzuführen, wenn eine Anfrage mit einer bestimmten Linking Length an die Hierarchie gestellt wurde. Alternativ kann diese Berechnung aber auch für jeden Knoten der Hierarchie getätigt werden und die Resultate können ebenda gespeichert werden. Da jeder Knoten einen Halo repräsentiert, hätte so jeder Halo alle relevanten Werte bei sich gespeichert. Hier muss man also wissen was man opfern will - nachträgliche Rechenzeit bei jeder Anfrage oder Speicherplatz.

## 3.6 Implementierung

Der Algorithmus welcher im letzten Abschnitt vorgestellt wurde, wurd im Rahmen dieser Arbeit implementiert. Hier soll ein kurzer Überblick über die Implementierung und die verwendeten Datenstrukturen gegeben werden. Die Software wurde dabei in C++ unter Windows geschrieben, sollte aber problemlos auch für andere Betriebssysteme kompiliert werden können.

### 3.6.1 Morton-Order-Octrees

Ein essenzieller Bestandteil des Friends-of-Friends-Algorithmus mit variabler Linking Length ist die spatiale Datenstruktur, mit deren Hilfe die initialen Halos bestimmt und die weiteren Suchen durchgeführt werden. Eine Datenstruktur, die der in [WBS+14] beschriebenen sehr nahe kommt ist der Octree (bzw. im zweidimensionalen der Quadtree). Dieser unterteilt das zu repräsentierende Gebiet in mit fortschreitender Baumtiefe immer kleiner werdende Würfel. Alle Würfel auf einer Ebene besitzen dabei dieselben Maße. Daher war die Idee die vorgestellte hypothetische Datenstruktur einfach durch einen Octree zu repräsentieren naheliegend.

Bei einer naiven Implementierung einer Baumstruktur würde zusätzlich zu dem benötigten Speicherplatz für die Daten auch Speicherplatz für den Baum selbst vonnöten sein. Das ist bei Octrees nicht zwangsläufig nötig. Diese können auch durch eine spezielle Ordnung der Datenpunkte (hier: Partikel) kodiert werden. Diese Ordnung wird auch als Z-Ordnung oder Morton-Ordnung (nach [Mor66]) bezeichnet. Somit werden solche speziellen Octrees auch Z-Order- oder Morton-Order-Octrees genannt (vgl. dazu u. a. [BET99] od. [CK10]). Diese arbeiten im Normalfall mit ganzzahligen Koordinaten. Jede Koordinate eines Datenpunktes wird dann nach einem bestimmten Prinzip in eine Bit-Signatur umgewandelt, welche zugleich dessen Position im Raum und die im Baum repräsentiert. Diese Signatur muss dabei nicht einmal gespeichert werden, da sie für jede Integer-Koordinate eindeutig ist und sehr schnell bestimmt werden kann. Diese erhält man, indem man die Koordinate für jede Dimension zunächst binär ausdrückt und diese Binär- Repräsentationen kombiniert. Das ist in Abbildung 3.5 für einen Quadtree, also für Dimension $d = 2$, gezeigt.

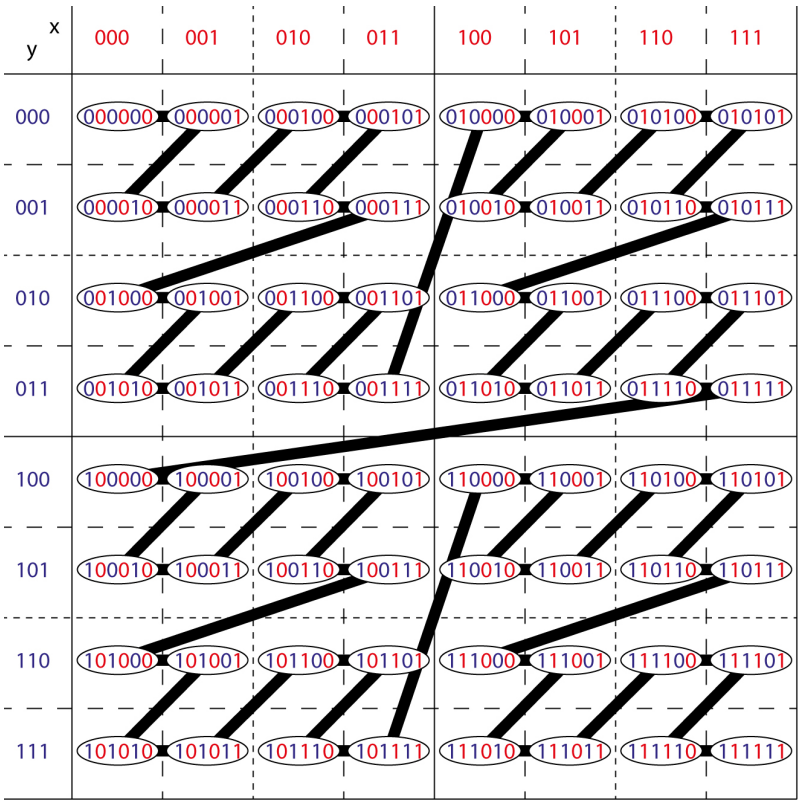



**Abbildung 3.5:** Koordinatenkombinationen für einen Morton-Order Quadtree mit Tiefe 3

Für jedes Paar aus x-Koordinate $x_n \dots x_2 x_1 x_0$ und y-Koordinate $y_n \dots y_2 y_1 y_0$ (beide in Binärdarstellung) wird die neue Koordinate $y_n x_n \dots y_2 x_2 y_1 x_1 y_0 x_0$ generiert. Man kann diese neuen Koordinaten dann aufsteigend ordnen, was in der Abbildung durch die schwarze Linie ausgedrückt wird. Der Z-artige Verlauf der Kurve ist auch für die Namensgebung verantwortlich. Die ersten $d$ Bit der erzeugten Koordinate geben die Position im ersten Level des Baums wider, die zweiten $d$ Bit die im zweiten, usw. So beginnen alle Koordinaten in der oberen Hälfte beispielsweise mit einer $0$, während die in der unteren Hälfte mit einer $1$ beginnen. Ordnet man alle Datenpunkte gemäß ihrer so generierten Koordinate, kann die Baumstruktur einfach in dieser Ordnung kodiert sein. Für drei Dimensionen funktioniert die Konstruktion der Koordinaten analog.

So wie hier formuliert funktioniert das Konzept leider nur für ganzzahlige Koordinaten. Die Partikeldaten der DarkSky-Simulation besitzen jedoch Koordinaten in Gleitkomma-Repräsentation. Damit die Methodik auch für diese Daten Anwendung finden kann, wurden die Koordinaten auf einen Bereich zwischen $0$ und $4294967295$ (der größten durch eine vorzeichenlose 32-bit Ganzzahl darstellbaren Zahl) skaliert. Dann wurden die Werte einfach auf den nächsten ganzzahligen Wert gerundet und die so resultierende Zahl als Koordinate weiterverwendet. Das funktioniert nur, weil wir den Octree nur bis zu einer bestimmten Tiefe benötigen und alles darunter ignorieren können und weil die Beträge der Koordinaten

im Vergleich zu $4294967295$ sehr klein sind. Je tiefer man in der Baumstruktur wandert, desto gravierender wird der Effekt von den so gemachten Rundungsfehlern. Wir benötigen den Baum nur bis zu der Tiefe, in der die Diagonale der repräsentierten Raumeinheit der gewählten minimalen Linking Length (vgl. Abschnitt 3.5) entspricht. Für alle sinnvollen Wahlen der Linking Length wird die benötigte Tiefe wohl nie $20$ übersteigen. Das lässt uns noch $12$ weitere Tiefenstufen für Rundungsfehler, was mehr als ausreichend ist.

Um nun eine Anfrage an die Datenstruktur zu stellen, muss man nur wissen, was für eine Bit-Signatur der gesuchte Teilraum besitzt. Für Anfragen auf Level $l$ des Octrees beschränkt man sich dafür nur auf die ersten $l \cdot d$ Bits. Da die Daten nach der Bit-Signatur geordnet vorliegen muss nun nur noch eine binäre Suche durchgeführt werden. Das Resultat dieser Suche ist dann entweder ein Datenpunkt, dessen Bit-Signatur-Präfix mit dem gewollten übereinstimmt oder es wird nichts gefunden. Falls sich mehrere Datenpunkte im selben Raumelement wie der gefundene befinden, sind diese in der Liste genau davor oder genau dahinter.

### 3.6.2 Parallelisierung

Da viele der Schritte des Friends-of-Friends-Algorithmus mit variabler Linking Length wie z. B. die Nachbarschaftssuche sehr häufig durchgeführt werden müssen, liegt die Idee nahe, dies mittels Parallelisierung zu beschleunigen. Schritte wie das Einlesen der Daten und der Aufbau des Octrees sind dabei schwer zu parallelisieren oder geschehen so schnell, dass sich es nicht lohnt. Die beiden Schritte, die den größten Einfluss auf die Performance haben, sind die Nachbarschaftssuche und der anschließende Hierarchieaufbau. Dabei benötigt die Nachbarschaftssuche für jedes Partikel, insbesondere bei ungünstiger Wahl von $min_ll$ und $max_ll$, am meisten Zeit. Der Hierarchieaufbau muss leider schrittweise geschehen, da dort jeweils die Daten vom letzten Schritt benötigt werden.

Die Parallelisierung der Nachbarschaftssuche gestaltet sich glücklicherweise einfach. Diese muss für jedes Partikel unabhängig durchgeführt werden und kann somit ohne Rücksicht auf Synchronisation oder andere Dinge geschehen. Dazu wird die Menge aller Partikel (beim Contest-Datensatz ca. 2 Millionen, vgl. Abschnitt 3.1) in $N$ gleich große Mengen aufgeteilt. Das $N$ steht dabei für die Anzahl der verwendeten Prozesse, auf die die Arbeit aufgeteilt werden soll. Jeder Prozess kümmert sich dann um die Nachbarschaftssuche für eine dieser Mengen. Die Suche findet jedoch auf dem gesamten Datensatz statt und beschränkt sich nicht auf die zugewiesenen Teilmengen. Haben alle Prozesse die Arbeit beendet, können einfach alle gefundenen Relationen in einer Liste zusammengeführt und nach Länge sortiert werden.

Die Parallelisierung wurde mit Hilfe von C++11-Threads durchgeführt. Eine hybride Parallelisierung, die sowohl die GPU als auch die CPU belastet, so wie sie in der Originalarbeit

[WBS+14] verwendet wird, wurde nicht erstellt. Die Performancegewinne sind nichtsdestotrotz signifikant, was in Abschnitt 5.3 gezeigt wird.

### 3.6.3 Post-Processing

Nach dem Aufbau der Halo-Hierarchie und ggf. nach der Extraktion der Halos einer bestimmten Linking Length ist man zumeist an den Eigenschaften der gefundenen Halos interessiert. Um diese herauszufinden, ist ein Post-Processing der Daten notwendig. Bevor relevante Werte berechnet werden, werden alle Halos aus dem Resultat entfernt, welche aus weniger als 20 Partikeln bestehen. Diese sind zu klein, um statistisch von Bedeutung zu sein bzw. könnten evtl. durch Fehler in den Daten entstanden sein. Diese Mindestgröße ist bei vielen Halo-Findern üblich und wird auch vom Rockstar-Algorithmus [BWW13] verwendet.

Mit die wichtigste Eigenschaft eines Halos ist seine Position. Diese wurde hier einfach als Durchschnitt der Partikelpositionen angenommen.

$$\vec{x}_{halo} = \frac{1}{n} \sum_{i=0}^{n} \vec{x}_i \tag{3.5}$$

Hier bezeichnen die $\vec{x}_i$ die Positionen der Partikel, $n$ ist deren Anzahl im Halo. Die durchschnittliche Geschwindigkeiten bzw. die Bewegungsrichtungen der Halos werden analog berechnet. Die Halo-Massen werden schlicht als Summe der Partikelmassen angesehen.

### 3.6.4 Ellipsoid-Fitting

Zusätzlich zu den anderen berechneten Parametern wurde noch an die Partikel jedes Halos ein Ellipsoid angepasst, welches diese nach Möglichkeit umschließen soll. Dafür bestehen mehrere Möglichkeiten. Es existieren einige Algorithmen für umschließende Ellipsoide mit minimalem Volumen. Solche wie der von Kumar und Yildirim [KY05] oder der von Mostagh [Mos05] sind leider entweder zu langsam für unsere Zwecke oder sie sind in der Implementierung sehr komplex. Daher wurde eine ähnliche Methode verwendet, wie sie von Allgood et al. schon explizit für das Fitting an Halos vorgeschlagen wurde [AFP+06]. Es sei dabei aber angemerkt, dass dabei nicht zwangsläufig ein umschließendes Ellipsoid mit minimalem Volumen berechnet wird, jedoch ist die Annäherung daran recht brauchbar.

Ein Ellipsoid kann grundsätzlich mit Hilfe von zwei Vektoren und zwei skalaren Werten beschrieben werden. Dabei handelt es sich um die Position des Mittelpunktes $\vec{x}$, die Richtung und Länge der Hauptachse $\vec{a}$ und Längenverhältnisse der Nebenachsen zur Hauptachse $b$ und $c$. Der Mittelpunkt ist recht einfach zu erhalten, da er dem Mittelpunkt der Halo-Partikel entspricht. Die Hauptachse des Halos wird mittels Hauptkomponentenanalyse bestimmt (PCA, [Pea01]). Der Prozess ist in Abbildung 3.6 gezeigt. Als erster Schritt wird die Hauptachse

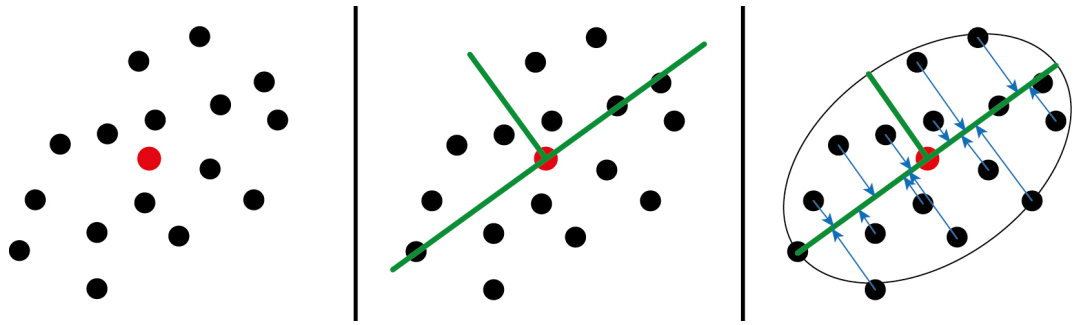


**Abbildung 3.6:** Ablauf des Ellipsoid-Fittings. Links: Partikel (schwarz) inklusive deren Mittelpunkt in rot. Mitte: Durch PCA errechnete Haupt- (durchgängig) und Nebenachse (stoppt bei rotem Punkt). Rechts: Projektion aller Partikel auf die Hauptachse, resultierendes Ellipsoid.

für alle Partikel des Halos bestimmt (Mitte). Ist das geschehen, werden alle Partikel auf diese Hauptachse projiziert (rechts). Das Partikel, dessen projizierter Punkt am weitesten vom Mittelpunkt entfernt ist, bestimmt die Länge der Hauptachse. In der Abbildung ist das das Partikel links unten, welches sich sehr nah an der Hauptachse befindet. Dasselbe wird auch für die aus der PCA resultierende Nebenachse durchgeführt. Im Dreidimensionalen existieren im Gegensatz zur Abbildung natürlich zwei Nebenachsen. Da diese immer orthogonal auf der Hauptachse und zueinander stehen, benötigt man nur die Richtung der Hauptachse, um die Richtungen der Nebenachsen zu kennen. Anhand der Abbildung ist auch zu erkennen, dass das resultierende Ellipsoid nicht notwendigerweise alle Partikel einschließt (unterstes Partikel), jedoch ist die Näherung recht gut. Aus der Länge der Nebenachsen kann dann anschließend ihre Länge im Verhältnis zur Länge der Hauptachse berechnet werden und man besitzt alle Werte, die zur Beschreibung des Ellipsoids notwendig sind.

Die Hauptkomponentenanalyse wurde größtenteils mit Hilfe der Matrix-Bibliothek Eigen [Gue11] implementiert. Insbesondere die dort vorhandene Singulärwertzerlegung war dabei von Nutzen.

# 4 Visualisierung von Halos

Im folgenden soll näher auf die Visualisierung der aus der DarkSky-Simulation resultierenden Halos eingegangen werden. Der Fokus liegt dabei eher auf vielfältigen interaktiven Analysemöglichkeiten für die vorliegenden Daten als auf Skalierbarkeit für sehr große Datensätze. Es wird jedoch nicht ausgeschlossen, dass das Resultat mit entsprechenden Modifikationen auch gut für diese verwendet werden kann. Interaktiv soll in diesem Zusammenhang mindestens ca. 25 generierte Bilder pro Sekunde bedeuten. Das garantiert, dass Benutzereingaben quasi ohne Wartezeit in Bilder umgewandelt werden und keine als ruckelnd erscheinenden Bildfolgen auftreten.

Die Anwendung wurde in C++ mit OpenGL als Grafik-API programmiert. Es findet vor allem modernes OpenGL (Version 4.3 aufwärts) Anwendung. Der Benutzer hat die Möglichkeit, eine im digitalen Raum platzierte Kamera mittels Tastatur- und Mauseingaben frei zu bewegen und zu rotieren. Dies ermöglicht eine freie Exploration der gezeigten Daten. Um dem Benutzer weitere Möglichkeiten zur Kontrolle der Visualisierung zu geben, wurde die AntTweakBar-Bibliothek verwendet [Dec05], welche ein Menü zur Verfügung stellt, in dem beliebige Parameter modifiziert werden können.

Es wird zunächst in Abschnitt 4.1 auf Arbeiten eingegangen, die ähnliche Methoden untersucht und implementiert haben. Daraufhin wird in 4.2 ein Überblick über die Eingabedaten gegeben, welche von der DarkSky-Simulation stammen. In Sektion 4.3 wird darauf eingegangen, wie die Halos optisch repräsentiert werden. Die darauf folgenden Abschnitte beschäftigen sich dann mit verschiedenen Visualisierungstechniken, deren Angebrachtheit bzw. Notwendigkeit sich jedoch erst aus den Eingabedaten erschließt.

## 4.1 Verwandte Arbeiten

Bezüglich der Visualisierung der Resultate von Simulationen Dunkler Materie existiert eine Vielzahl von Arbeiten. Einige konzentrieren sich dabei hauptsächlich auf die Verwaltung der riesigen Datenmengen, die durch die Simulationen generiert werden. Szalay et al. [SSL08] verwenden hierzu einen Octree, welcher auf einem externen Server gehalten wird, um nicht übermäßige Anforderungen an die Anwenderhardware zu stellen. Zur Visualisierung werden dann die relevanten Daten komprimiert von diesem Server abgefragt und direkt mittels Point-Splatting visualisiert. In letzter Zeit wurden die Datenmenge pro Zeitschritt jedoch so

groß, dass selbst die Daten eines einzelnen Zeitschritts für interaktive Visualisierungen Out-of-Core gerendert werden müssen. Dieses Problems nehmen sich Fraedrich et al. an, indem sie zum einen die vorhandenen Daten per Vektor-Quantisierung komprimieren und sich zum anderen eines stark angepassten zu einem so genannten Page-Tree umorganisierten Octree bedienen [FSW09]. Dieser organisiert die Daten so, dass Partikel, die sich nahe beieinander befinden, sich auch im Datenspeicher nah beieinander sind, was Speicherlatenzen reduziert. Zudem werden möglicherweise in der Zukunft benötigte Bereiche im Voraus geladen, um potentiell zur Verfügung stehende Speicherbandbreite auszunutzen.

Während die sich auf Speichermanagement konzentrierenden Arbeiten zumeist Punkt-Splatting als Haupt-Renderingtechnik verwenden, setzen andere auf die unterschiedlichsten Techniken. Dolag et al. setzen beispielsweise auf volumetrisches Raycasting [DRGI08]. Im Gegensatz dazu berechnen Navrátil et al. zunächst Isoflächen, die anschließend gerendert werden [NJB07]. Kaehler et al. konzentrieren sich auf die filamentartigen Strukturen, die Dunkle Materie in großen Skalenbereichen ausbildet [KHA12]. Sie präsentieren drei unterschiedliche Techniken, wobei sie durch Verfolgen der Partikel Dunkler Materie über die Zeit hinweg die Bildqualität gegenüber anderen Techniken beträchtlich steigern können.

Eher allgemeine Rendering-Methoden werden von Turk et al. verwendet, um aus möglichst beliebigen Partikel-Datensätzen brauchbare Bilder zu generieren. Mit $yt$ entwickelten sie ein Toolkit, welches auf die Daten möglichst vieler kosmologischer Simulationen zugeschnitten ist und zumeist nicht-interaktiv unterschiedlichste Visualisierungen und Diagramme generiert [TSO+11]. Eher auf das Aufzeigen von Formationen mehrerer Galaxien ausgerichtet ist die Methode von Miller et al. [MQF06]. Hierbei wird ein starker Fokus auf die Analysemöglichkeiten gesetzt. So können zum Beispiel Galaxien markiert werden, die sich um so genannte Void-Zones, also anscheinend leere Stellen im Weltall, anordnen.

Die Visualisierung Dunkler Materie wird von bestehenden Arbeiten recht gut abgedeckt, jedoch existieren kaum solche, die sich auf das Aufzeigen von Halos konzentrieren. Die Visualisierungen hierzu verfolgen demnach recht unterschiedliche Zwecke. Takle et al. konzentrieren sich beispielsweise in [TSH12] auf das Nachverfolgen und Visualisieren von Satelliten-Halos (also Halos, welche an größere Halos gebunden sind. In [TSKH13] präsentiert weitesgehend dieselbe Gruppe von Wissenschaftlern Glyphen, die zur Repräsentation von Halos dienen können. Eine sphärische Glyphe kodiert dabei zugleich die Masse, die Position, die Drehgeschwindigkeit und viele weitere Parameter. Darunter befinden sich auch solche, die nicht direkt mit der Dunklen Materie zu tun haben, sondern auch mit der Galaxie aus baryonischer Materie zusammenhängen, an die die Dunkle Materie gravitativ gebunden ist.

Sehr stark in Richtung des Ansatzes, der hier verfolgt wird, gehen die Arbeiten von Ahrens et al. und von Shan et al. Erstere konzentriert sich dabei stark auf vergleichende Visualisierung von unterschiedlichen Simulationen, deren Techniken im Rahmen der Arbeit in das quelloffene Visualisierungs-Tool ParaView integriert wurden [AHH+06]. Letztere verwendet sowohl normale Halo-Daten als auch zur Verfügung stehende Merger-Bäume, um die

Pfade der Halos nachzuverfolgen [SXG+14]. Die zu den Halos gehörenden Partikel Dunkler Materie werden ebenfalls visualisiert. Die Halos selbst werden meist nur durch Färbung der Partikel oder durch farbige Flächen repräsentiert. Hierbei ist es möglich spezielle Bereiche zu selektieren und näher zu betrachten. Im Nachhinein betrachtet besitzen die Ideen von Shan et al. sehr viele Überschneidungen mit den hier präsentierten Ansätzen. Der Unterschied besteht unter anderem in der Repräsentation der Halos selbst, der hier vorgenommenen erweiterten Visualisierung der Halo-Parameter und einigen zusätzlichen Ideen im Bereich der Merger-Bäume. Es sei jedoch angemerkt, dass diese Arbeit dem Autor erst bekannt wurde, nachdem ein Großteil der Visualisierung bereits implementiert war.

## 4.2 Eingabedaten

Neben den in Abschnitt 3.1 präsentierten Partikeldaten, die von der DarkSky-Simulation entweder eigenständig [SWT+14] oder im Rahmen des SciVis-Contests 2015 [HGTS15] veröffentlicht wurden, existieren noch Halo-Kataloge, welche mit Hilfe des Rockstar-Algorithmus berechnet wurden (vgl. Abschnitt 3.4). Je nach Datensatz besitzen diese jedoch stark unterschiedlichen Umfang. Während der große ds14_a-Datensatz und die kleineren ds14_g_*-Datensätze recht eingeschränkte Daten für jeden Halo bieten, ist der Contest-Datensatz weitaus umfangreicher. Die gemeinsamen Daten sind in Tabelle 4.1 gezeigt.

**Tabelle 4.1:** Für alle Datensätze der DarkSky-Simulation vorhandene Halo-Parameter

| Parameter | Einheit | Bedeutung |
|---|---|---|
| $\vec{x}$ | $\frac{Mpc}{h}$ | Position des Halo-Zentrums |
| $\vec{v}$ | $\frac{km}{s}$ | Geschwindigkeitsvektor des Halos |
| $M_{vir}$ $M_{200b}$ $M_{200c}$ $M_{500c}$ $M_{2500c}$ | $\frac{Mpc}{h}$ | Halo-Massen innerhalb unterschiedlicher Radii |
| $v_{max}$ | $\frac{km}{s}$ | Maximale Rotationsgeschwindigkeit |
| $spin$ | keine | u. a. errechnet aus Drehimpuls |
| $kin_to_pot$ | keine | Verhältnis von kinetischer zu potentieller Energie |
| $ID$ | keine | Eindeutige 64-Bit ID. |
| $pID$ | keine | Bei Sub-Halos: ID des Vater-Halos. Sonst: pID = ID. |

Der Contest-Datensatz ds14_scivis_0128 besitzt zusätzlich zu den in der Tabelle dargestellten Parametern noch einmal grob $40$ weitere Werte. Das ist nicht zuletzt dem Fakt geschuldet, dass hierfür mehrere Zeitschritte zur Verfügung stehen und somit auch der zeitliche Ablauf eine Rolle spielt. Dafür bietet der Contest-Datensatz neben den Daten für die einzelnen Zeitschritte eine Merger-Baum-Datei, welche die komplette Halo-Historie widerspiegelt. Darin ist z. B. enthalten, welche kleineren Halos von welchem größeren so stark gravitativ angezogen wird, dass sie im nächsten Zeitschritt voll im größeren Halo aufgehen. Zudem

wurden dort Halos, die Ungenauigkeiten des Detektionsalgorithmus geschuldet sind, z. B. solche, die nur einen Zeitschritt existieren, entfernt. Es ist also für jeden Halo vollständig ermittelbar, aus welchen kleineren Halos der Vergangenheit er entstanden ist. Zusätzlich sind weitere interessante Parameter wie der Drehimpuls oder die Geschwindigkeitsdispersion vorhanden, welche beispielsweise verwendet werden kann, um den Typus von Galaxien zu bestimmen.

Für die Visualisierung besonders interessant ist der zur Verfügung gestellte Virialradius und die Parameter für umschließende Ellipsoide. Diese wurden für die Partikel eines jeden Halos berechnet und werden hier verwendet, um die Halos in der Visualisierung optisch darzustellen. Dies soll genauer im nächsten Abschnitt erläutert werden.

Aufgrund der offensichtlich umfangreicheren Daten wurde hier die Entscheidung gefällt, die Visualisierung hauptsächlich für die Merger-Baum-Datei des Contest-Datensatzes zu optimieren. Dies vereinfacht zudem Vieles, da es sich (vieleicht abgesehen vom ds14_g_100_2048-Datensatz) um den einzigen Datensatz handelt, der problemlos sowohl in aktuelle Arbeits- als auch Grafikspeicher passt.

## 4.3 Repräsentanten

Eine der wichtigsten Fragen bei einer Visualisierung ist es, wie das am häufigsten auftretende zu zeigende Element dargestellt werden soll. Hier handelt es sich dabei um Halos Dunkler Materie. Streng genommen handelt es sich dabei nur um Bereiche, in denen die Dichte Dunkler Materie höher ist als an anderen Stellen. Eine alleinige Visualisierung der Dichteverteilungen wäre jedoch nicht genug, um beispielsweise Halos, die sich nahe beieinander befinden optisch voneinander abzugrenzen. Daher sind eindeutigere Repräsentanten für die Halos vonnöten. Bei der Definition von Halos spielt der Virialradius $R_{vir}$ eine große Rolle. Im Dreidimensionalen implizieren einzelne Radii zumeist ein kugelförmiges Objekt, was uns zu Kugeln als mögliche Repräsentanten führt. Wie weiter oben bereits erläutert, stellen die Daten ebenfalls Werte für die Partikel umschließende Ellipsoide zur Verfügung. Glücklicherweise handelt es sich bei Kugeln mathematisch gesehen nur um Spezialfälle von Ellipsoiden, d. h. wenn man es schafft, diese Ellipsoide zu zeigen, sollte es ein Einfaches sein, die Visualisierung so zu ändern, das auch das Zeigen der Radii als Kugeln möglich ist. Somit wurde die Entscheidung getroffen, die Halos als Ellipsoide im dreidimensionalen Raum darzustellen.

Für das Rendern von Kugeln existieren viele verschiedene Möglichkeiten. So können z. B. zur Kamera ausgerichtete Rechtecke als eigentliche Primitive dienen, um darauf die Kugeln zu zeichnen. Moderne Grafik-APIs besitzen im Normalfall keine Kugeln als Grundprimitive. Jedoch können beliebige Objekte, auch Kugeln und Ellipsoide, durch ein Dreiecksnetz angenähert werden. Bei Ellipsoiden kann erstere Methode weitaus komplizierter sein, denn

Ellipsoide besitzen, je nach Orientierung, unterschiedliche Projektionsflächen auf Bildschirmen; im Gegensatz zu Kugeln. Daher werden hier die Ellipsoide als Dreiecksnetz gerendert. Diese können beliebig rotiert und bewegt werden, ohne dass Zeit an die Berechnung der Projektionsfläche verschwendet werden muss. Jedoch ist dadurch die Anzahl der gerenderten Primitive weitaus höher.

Wie bereits erwähnt besitzen die meisten Grafik-APIs keine Kugel-Primitive. Daher wird zunächst ein nur aus Dreiecken bestehendes reguläres Ikosaeder (20-Eck) gerendert. Mittels Tessellation werden dann die Oberflächendreiecke weiter unterteilt und die neu entstandenen Punkte so angeordnet, dass alle den selben Abstand zum Ursprung besitzen. Das Resultat ist eine aus vielen Dreiecken bestehende Annäherung einer Einheitskugel. Um aus dieser Kugel dann ein Ellipsoid zu machen, sollten wir uns zunächst in Erinnerung rufen, wie diese in den Daten vorliegen. Ein Ellipsoid im Dreidimensionalen kann grundsätzlich durch drei Radii ausgedrückt werden. Gegeben ist die Hauptachse $\vec{a}$ sowie die Längenverhältnisse der beiden Seitenachsen zur Hauptachse $b$ und $c$. Der erste Radius ist die Länge der Hauptachse, also $|\vec{a}|$. Da $b$ und $c$ Längenverhältnisse beschreiben, sind die beiden anderen gesuchten Radii $b \cdot |\vec{a}|$ und $c \cdot |\vec{a}|$. Die Idee ist jetzt, die bestehende Kugel so zu deformieren, dass sie in den drei Hauptrichtungen diese drei Radii besitzt. Dabei wurde festgelegt, dass die Hauptachse auf die x-Achse abgebildet werden soll. Eine solche benötigte Deformierung kann mittels einer anisotropen Skalierung erreicht werden. Diese Skalierung kann, in homogenen Koordinaten, mit folgender Matrix ausgedrückt werden:

$$S = \begin{pmatrix} |\vec{a}| & 0 & 0 & 0 \\ 0 & b \cdot |\vec{a}| & 0 & 0 \\ 0 & 0 & c \cdot |\vec{a}| & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

Multipliziert man die homogene Koordinate jedes Punktes mit dieser Matrix erhält man das gewollte Ellipsoid. Dieses ist jedoch noch nicht korrekt rotiert und befindet sich noch im Ursprung des Koordinatensystems. Der Prozess der Erstellung der Ellipsoide ist noch einmal in Abbildung 4.1 gezeigt. Will man statt der Ellipsoide nur Kugeln mit $R_{vir}$ als Radius rendern, verwendet man stattdessen eine Matrix mit $R_{vir}$ an den ersten drei Werten der Hauptdiagonalen der Matrix. Der Effekt der Tessellierung ist in Abbildung 4.2 gezeigt. Es hat sich herausgestellt, dass ein Faktor von $5$ mit den besten Kompromiss zwischen Performance und Annäherung des gewollten Primitivs darstellt.

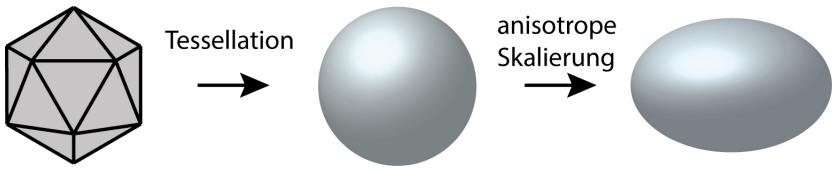
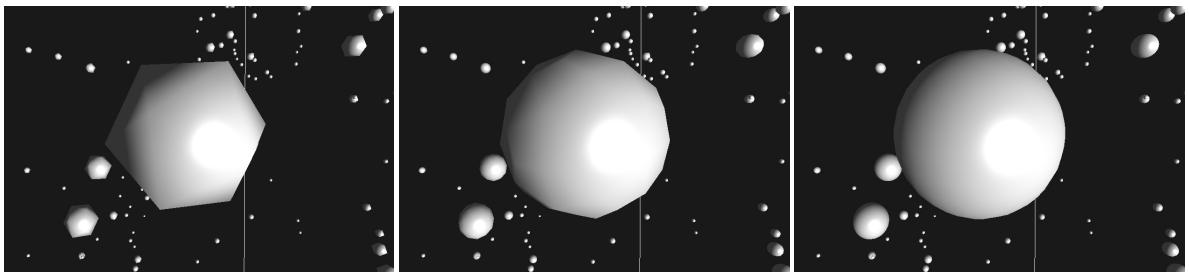



**Abbildung 4.1:** Prozess der Erstellung von Ellipsoiden auf Grafikhardware.



(a) Tess-Faktor = 1 (b) Tess-Faktor = 2 (c) Tess-Faktor = 5

**Abbildung 4.2:** Effekte unterschiedlicher Tessellierungs-Faktoren auf die Darstellung der repräsentierenden Ellipsoide / Kugeln.

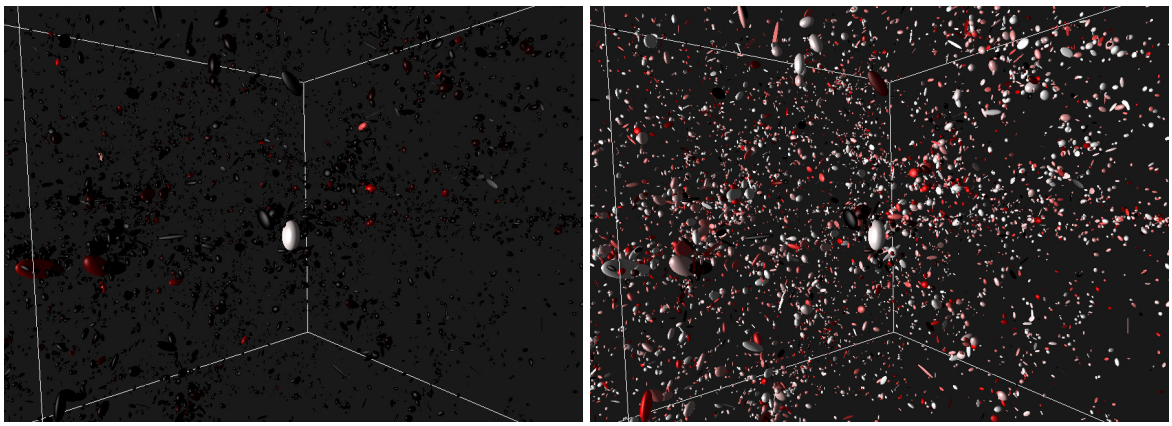
Die Form und Größe des Ellipsoids ist nach all diesen Schritten nun korrekt, es muss nur noch richtig rotiert und an die korrekte Stelle geschoben werden. Die benötigte Rotationsmatrix $R$ enspricht einer Matrix, die die x-Achse auf den Vektor $\frac{\vec{a}}{|\vec{a}|}$ rotiert. Die Translationsmatrix $T$ kann einfach aus der Position des Halos, die in den Daten enthalten ist, erstellt werden.

Der komplette Prozess resultiert dann in einer Model-Matrix $M = T \cdot R \cdot S$. Multipliziert man die Positionen der Oberflächenpunkte der Einheitskugel mit dieser Matrix, erhält man die korrekte Position des Punktes im Raum auf der Ellipsoidoberfläche. Diese Operation muss für jeden Vertex von jedem dargestelltem Ellipsoid pro Frame mindestens einmal durchgeführt werden. Zu diesem Zweck wird sowohl die Matrix $M$ als auch ihre Inverse $M^{-1}$ vor Start der Visualisierung für jeden Halo vorberechnet.

Es ist dem Benutzer möglich, einen Skalierungsfaktor anzugeben, um die gerenderten Primitive größer darstellen zu lassen. Das ist vor allem für die Ellipsoide ratsam, da diese im Vergleich zum Virialradius sehr klein sind und dem Betrachter so manche Details entgehen können. Für korrekte Abgrenzung der Halos zueinander sollte dieser Faktor aber wieder auf 1 zurückgestellt werden.

## 4.4 Visualisierung unterschiedlicher Parameter

Jeder Halo im Datensatz besitzt unterschiedliche Eigenschaften, welche zumeist durch genau einen Zahlenwert ausgedrückt werden können. Darunter befinden sich beispielsweise die Massen oder die Geschwindigkeitsdispersion. Wie bereits erwähnt, ist die Zahl der Werte pro Halo relativ groß, daher ist die gleichzeitige Darstellung in einer Glyphe so gut wie ausgeschlossen. Aufgrund dessen werden diese Werte einzeln als Farbe kodiert dargestellt. Zu diesem Zweck werden die maximalen, minimalen und durchschnittlichen Werte für jede der Variablen im Voraus berechnet, was eine einfachere Interpolation der Farben ermöglicht. Diese Statistiken werden global, pro Zeitschritt und pro Baum in der Merger-Baum-Hierarchie



(a) Global (b) Pro Baum

**Abbildung 4.3:** Visualisierungen der Halo-Massen des letzten Zeitschritts des Datensatzes (a) und (b) besitzen genau denselben Kamera-Standpunkt. Die Halos sind mit Faktor 10 skaliert.

erhoben, was differenziertere Analysen ermöglicht. Die Einfärbung wird in Abbildung 4.3 gezeigt. Sowohl das linke als auch das rechte Bild zeigen den letzten Zeitschritt des Contest-Datensatzes. Links wurde nach der globalen Statistik eingefärbt, rechts nach Statistik des Baumes, dem der Halo zugehörig ist. Werte nahe des Minimums werden schwarz dargestellt, Werte am Mittelwert rot und Werte nahe des Maximums weiß. Dazwischen wird linear interpoliert. Es ist unschwer zu erkennen, dass links vor allem größere Halos viel Masse besitzen. Das ist rechts nicht zwangsläufig zu erkennen. Da es sich um eine Darstellung des letzten Zeitschritts handelt, ist die vornehmlich hellere bis weißliche Färbung nur logisch, da Halos über die Zeit hinweg meist an Masse zunehmen und sie somit zumeist die massivsten Halos ihres Baumes darstellen. Neben der Einfärbung in drei Farben ist auch die direkte Interpolation zwischen Minimum und Maximum möglich, ohne dass der Mittelwert eine spezifische Farbe besitzt.

Neben den skalaren Werten, die einfach durch eine Farbkodierung darstellbar sind, existieren noch zwei vektorielle Größen in den Daten. Namentlich sind dies die Geschwindigkeit und der Drehimpuls. Für diese sind vor allem die Richtungen, in die diese Vektoren zeigen, interessant. Im Falle der Geschwindigkeit zeigt das die Bewegungsrichtung auf, im Falle des Drehimpulses zeigt es die Achse, um die sich der Halo selbst dreht. Eine naheliegende Lösung ist es, diese Richtungen einfach als Pfeile zu rendern, was auch hier getan wurde. Das ist in Abbildung 4.4 dargestellt. Diese bestehen aus einer einfachen Linie und einem Tetraeder als Spitze. Da die Vektoren absolut gesehen eigentlich sehr lang sein und weit aus dem Definitionsbereich herausragen müssten, wurden die Längen der Geschwindigkeiten einfach logarithmisch, die Längen der Drehimpulse sogar doppelt logarithmisch skaliert. Um nicht zu viel visuelles Durcheinander zu verursachen, ist es dem Benutzer möglich, die

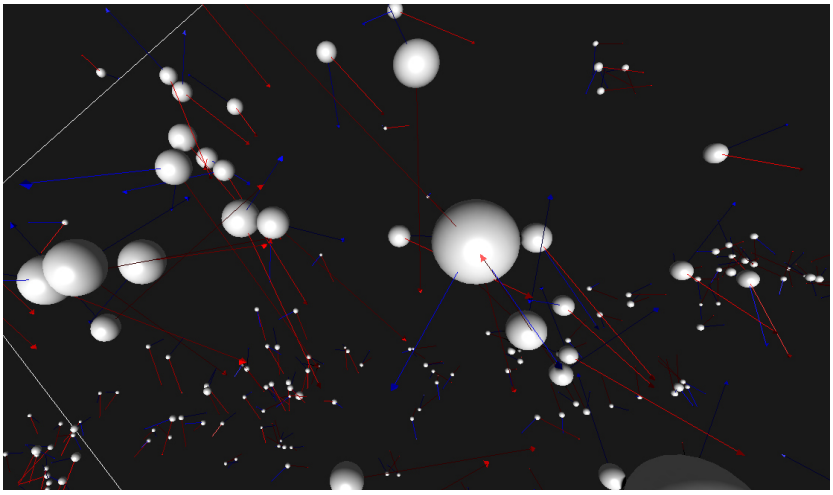


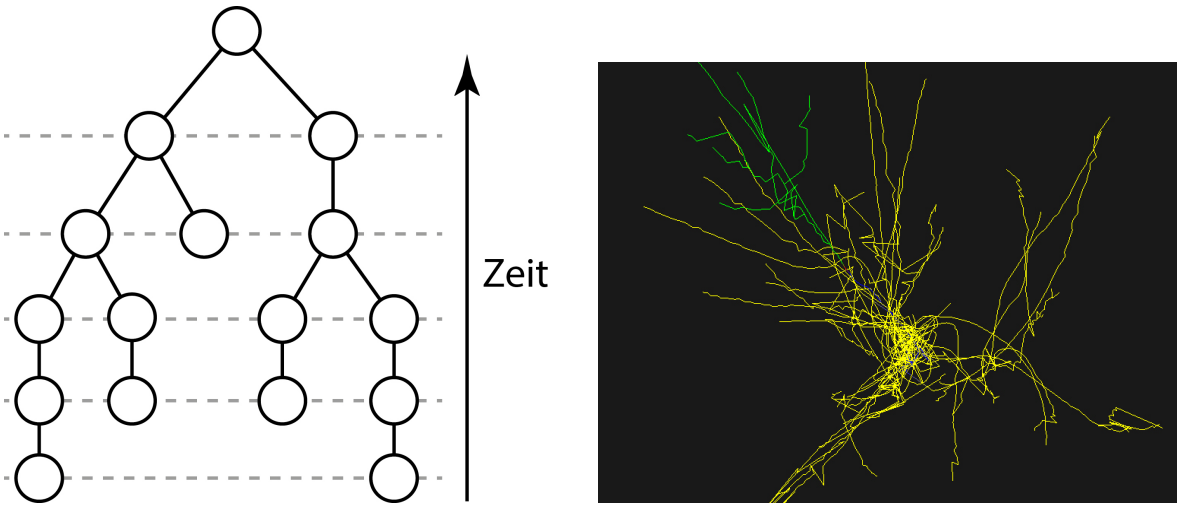
**Abbildung 4.4:** Blaue und rote Pfeile visualisieren die Bewegungsrichtungen und die Haupt-Drehachsen.

Darstellung der Pfeile abzuschalten. Es sei zusätzlich noch angemerkt, dass die Pfeile nicht beim Mittelpunkt der Ellipsoide starten, sondern erst an deren Oberfläche beginnen, um das Verdecken von zu kurzen Pfeilen zu vermeiden.

## 4.5 Merger-Bäume

Der Contest-Datensatz bietet vielfältige Möglichkeiten, die Historie eines jeden Halos zu untersuchen. Diese werden in Form von Merger-Bäumen zur Verfügung gestellt. Diese beschreiben, wie Halos sich über die Zeit hinweg zusammenfügen. Ein solcher Baum ist in Abbildung 4.5(a) gezeigt. Die Wurzel des Baumes repräsentiert einen Halo am für diesen Baum spätesten Punkt in der Zeit. Der Wurzel-Halo kann ein Resultat vieler Fusionierungsprozesse kleinerer Halos sein. Immer wenn sich zwei Halos zusammenfügen, besitzen sie einen gemeinsamen Vaterknoten. Im Datensatz kann es auch vorkommen, dass ein Halo über die Zeit hinweg Masse anhäuft, ohne dass zwangsläufig dabei kleinere andere Halos durch die Gravitation „aufgesogen“ werden. Diese Massezunahme kann durch ungebundene Dunkle Materie zustande kommen. Aus diesem Grund wird für jeden neuen Zeitschritt ein weiterer Knoten benötigt, um die Änderungen der unterschiedlichsten Halo-Parameter Rechnung zu tragen.

In der Visualisierung ist es dem Benutzer möglich, selbst durch Mausklick ein sichtbares Halo-Ellipsoid auszuwählen und für dieses den Merger-Baum anzeigen zu lassen. Dabei werden sowohl alle Vorgänger als auch Nachfolger in der Zeit angezeigt. Der Baum wird dabei anders angeordnet als in Abbildung 4.5(a). Die dargestellten Halo-Knoten werden zusätzlich



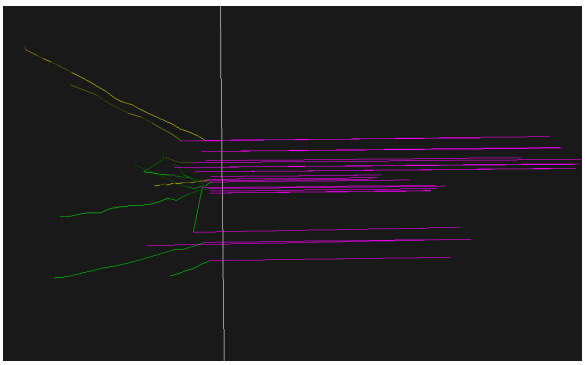

(a) Merger-Baum (b) Merger-Baum in der resultierenden Anwendung

**Abbildung 4.5:** Unterschiedliche Darstellungen von Merger-Bäumen. Die Knoten entsprechen den Halos zu bestimmten Zeitpunkten. Bei (b) wird zusätzlich die Position der Knoten-Halos verwendet.

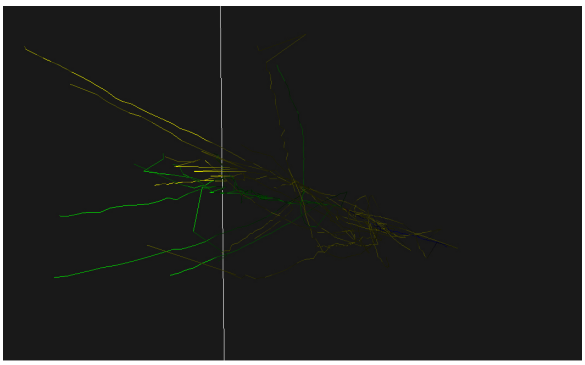
noch auf ihre korrekte Position im dreidimensionalen Raum verschoben. Das ermöglicht eine genaue Nachverfolgung, wie und aus welchen Richtungen die hereinkommenden Halos sich genähert haben. Das ist in Abbildung 4.5(b) zu sehen. Dort wird eine Farbkodierung verwendet. Direkte zeitliche Vorgänger des ausgewählten Halos sind grün dargestellt, alle Halos, die sich später mit dem ausgewählten vereinigen werden, gelb. Direkte Nachfolger in der Zeit sind blau gefärbt, der ausgewählte selbst rot, was in der Abbildung jedoch kaum zu erkennen ist, weil diese von sich nähernden gelben Halos umkreist werden.

### 4.5.1 Randverhalten

Betrachtet man in der Visualisierung verschiedene Merger-Bäume fällt schnell auf, dass einige von ihnen „auseinandergerissen“ erscheinen. Ein Teil des Baumes befindet sich auf der einen Seite des Definitionsbereichs, der andere befindet sich auf der anderen, was auch in Abbildung 4.6(c) gezeigt wird. Rechts und links befinden sich die beiden Teile des Baumes, die weißen Linien begrenzen das hintere Ende des Definitionsbereiches. Hier werden statt der Kanten, wie in Abbildung 4.5(b) gesehen, die Knoten gerendert. Würde man hier die Kanten zeigen, würde es mindestens eine geben, die durch den kompletten Definitionsbereich auf die andere Seite wandert. Dies entspräche in der Realität einer Bewegung mit Überlichtgeschwindigkeit, was nach aktuellem physikalischem Stand nicht möglich ist. Es wurde wahrscheinlich auch von der Simulation her sichergestellt, dass solches Verhalten nicht auftreten kann.



(a) Über die Grenze hinausgehende Kanten werden violett dargestellt



(b) Wie der Baum auf der anderen Seite des Bereichs weitergehen würde wird desaturiert dargestellt.

(c) Merger-Bäume befinden sich manchmal an beiden Enden des Definitionsbereichs gleichzeitig

**Abbildung 4.6:** Unterschiedliche Ansichten von Merger-Bäumen, die durch die periodischen Randbedingen „auseinandergerissen“ wurden.

Der Grund für diese Anomalie ist in den Randbedingungen der Simulation zu suchen. Wie bereits in Abschnitt 2.5 erläutert, verwenden die meisten kosmologischen Simulationen periodische Randbedingungen, um in jedem Fall außerhalb des Randes korrektes physisches Verhalten zu bekommen. Das impliziert jedoch auch, dass alle simulierten Partikel Dunkler Materie, welche sich über den Rand hinaus bewegen, auf der anderen Seite wieder in den Bereich eintreten. Es ist also keine Bewegung mit Überlichtgeschwindigkeit passiert, sondern nur eine kurze Bewegung über den Rand des Definitionsbereichs hinaus, was von den Randbedingungen verhindert wird.

Um diese Fälle in der Visualisierung speziell kenntlich machen zu können, ist es nötig, ein Maß dafür zu haben, wann eine Merger-Baum-Kante über die Grenzen geraten sein muss. Fakt ist, dass diese Kanten in der Visualisierung sehr lang werden würden und alle anderen „normalen“ beobachteten Kanten im Vergleich dazu recht kurz sind. Daher werden jene Kanten als über die Grenzen hinausgehend betrachtet, die länger als die Hälfte des Durchmessers des Definitionsbereichs sind.

Zum Zwecke der schnelleren Darstellung wird mit dem soeben vorgestellten Maß für jeden Baum im Voraus berechnet, ob dieser eine solche Kante enthält. Wenn ja, wird zugleich mitgespeichert, in welcher der drei Hauptrichtungen diese über den Definitionsbereich hinausgetreten ist. In den finalen Bildern kann dies dann auf zwei unterschiedliche Weisen dargestellt werden. Diese beiden Arten sind in den Abbildungen 4.6(a) und 4.6(b) gezeigt. Bei ersterer werden die relevanten Kanten aus dem Definitionsbereich hinausgeführt und violett eingefärbt, um ein Austreten aus diesem kenntlich zu machen. Die zweite Methode rendert den Teil des Baumes, welcher sich komplett auf der anderen Seite des Defitionsbereichs befinde,t noch einmal desaturiert außerhalb desselben, so als ob man mehrere Versionen des Bereiches nebeneinander zeigen würde. Das vermeidet, dass der Betrachter zur genauen Ansicht des Baumes zwischen beiden Seiten hin und her springen muss.

## 4.6 Sub-Halos

Neben den Halos, die vollständig von anderen geschluckt werden, existieren auch noch solche, die sich ständig im Gravitationsbereich anderer befinden, aber nie ganz ihre eigenständige Bewegung verlieren. Diese Sub-Halos befinden sich räumlich zumeist sehr nahe an ihren so genannten Host-Halos, welche oft größer sind. Dies kollidiert leider mit der undurchsichtigen Darstellung der Halos in der Visualisierung. Dort sind die Ellipsoide, welche die Host-Halos darstellen, zumeist so groß, dass sie die Repräsentanten der Sub-Halos vollständig umschließen und somit für den Betrachter unsichtbar machen. Um die Sub-Halos wieder sichtbar zu machen, sind mehrere Methoden möglich.

Die erste mögliche Methode wäre es, die Host-Halos transparent zu rendern. Das würde eine Tiefenordnung aller Primitive erfordern, die für jedes Bild neu berechnet werden müsste, was sehr teuer und somit langsam werden kann. Die zweite Möglichkeit bestünde in einer Wireframe-Anzeige der Host-Halos, also nur dem Rendern der Umrisse der Dreiecke, aus denen die Ellipsoid-Primitive bestehen. Die hier präsentierte Lösung ist eine Verbesserung dieses zweiten Ansatzes.

In der Visualisierung kann die Farbe eines jeden Halos mitunter eine Bedeutung haben. So kann sie etwa für die Masse stehen. Zeigt man den Host-Halo nur im konventionellen Wireframe-Modus an, wird diese Farbe in so dünnen Linien dargestellt, dass sie rein optisch nur schwer mit den Farben der Sub-Halos zu vergleichen sind. Daher ist es ratsam, diese Linien so dick wie möglich anzuzeigen, jedoch nicht zu dick, damit die Sub-Halos nicht zu sehr verdeckt werden. Die Lösung, die hier gewählt wurde, ist in Abbildung 4.7 gezeigt. Diese Art, die Ellipsoide (im Fall der Abbildung Kugeln mit $R_{vir}$ als Radius) zu rendern, ist nur möglich, weil diese mittels Tessellation erzeugt wurden.

Der Tessellator der Grafikkarte liefert jede Koordinate der Dreiecksunterteilungen in baryzentrischen Koordinaten. Innerhalb der Dreiecke, auf welche diese sich beziehen, ist jede der

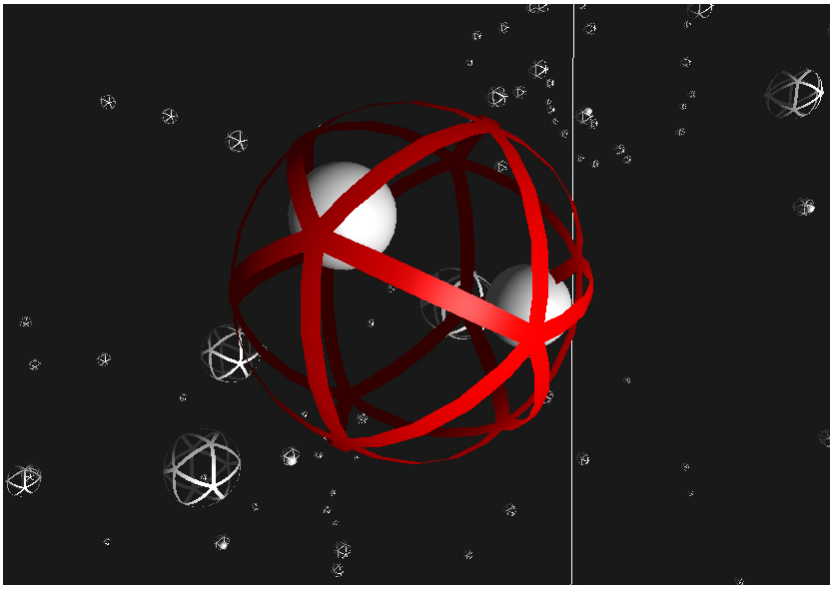


**Abbildung 4.7:** Sub-Halos werden innerhalb teilweise gerenderter Host-Halos gezeigt. $\epsilon = 0.05$

drei Koordinaten $\leq 1$ und $\geq 0$. Nahe am Rand der Dreiecke ist immer mindestens eine der drei Koordinaten sehr nahe an $0$, auf dem Rand genau $0$. Somit ist es möglich, einfach alle Fragmente eines Dreiecks nicht zu rendern, bei denen alle drei Koordinaten eine bestimmte Mindestgröße $\epsilon$ besitzen. Variiert man dieses $\epsilon$, kann die Dicke der Linien des Host-Halos verändert werden. Wie auch an der Abbildung unschwer zu erkennen ist, ermöglicht dies einen einfachen Farbvergleich zwischen Host-Halo und Sub-Halos.

## 4.7 Shading- und Beleuchtungstechniken

Bei Halos handelt es sich um dreidimensionale Objekte im dreidimensionalen Raum. Deshalb ist es der Visualisierung nur zuträglich, dass das Abbild jener Halos ebenfalls einen dreidimensionalen Eindruck erweckt, wozu eine korrekt erscheinende Beleuchtung notwendig ist. Für die Beleuchtungsberechnung sind vor allem drei Dinge wichtig. Zu diesen gehören die Eigenschaften der Lichtquelle, die Eigenschaften der beleuchteten Oberflächen und die Position des Betrachters. Für eine korrekt erscheinende Beleuchtung sind von diesen dreien aber zumeist die verwendeten Oberflächeneigenschaften ausschlaggebend.

Zu den Oberflächeneigenschaften gehören unter anderem das Material der Oberfläche und die Normalen selbiger. Das Material kann, wie schon weiter oben gezeigt wurde, durch bestimmte Anforderungen an das Programm vorherbestimmt sein. So ist z. B. eine bestimmte Farbe nötig, um anzuzeigen, dass der dargestellte Halo eine bestimmte Masse besitzt. Was

dann als einzig essentieller zu bestimmender Faktor übrig bleibt, sind die Normalen der Objektoberflächen.

Die Normalen der Ellipsoide werden mittels Phong-Shading [Pho75] bestimmt, d. h. die Normalen werden zunächst pro Vertex berechnet und dann für alle weiteren Pixel, die nicht direkt einen Vertex abbilden, interpoliert. Diese Berechnung geschieht im Modell-Raum, in welchem die Ellipsoide, wie in Abschnitt 4.3 beschrieben, als Einheitkugeln dargestellt sind. Da die Normalen einer Kugel immer der Richtung vom Mittelpunkt zum Oberflächenpunkt entsprechen, kommt der Normalenvektor eines Vertex auf der Oberfläche einer Einheitskugel im Modellraum einfach dessen Positionsvektor gleich. Für die Verwendung in anderen Räumen muss dieser Vektor nur entsprechend transformiert werden.

Für die anschließende Beleuchtung der Ellipsoide wird das Blinn-Phong-Modell [Bli77] verwendet. Bei diesem kann der ambiente, der diffuse und der spekulare Anteil des einfallenden Lichtes moduliert werden. Da in der Visualisierung nur eine einzige Punktlichtquelle verwendet wird und die Kamera frei beweglich ist, kann es passieren, dass nur hauptsächlich die von der Lichtquelle abgewandten Seiten gezeigt werden. Daher ist ein ambienter Faktor $\neq 0$ erforderlich, um die Rückseiten nicht komplett schwarz erscheinen zu lassen. Alle anderen Werte sollten aber frei wählbar sein, je nach Vorliebe des Benutzers. Das Modell sieht für einen einzelnen Farbkanal wie folgt aus:

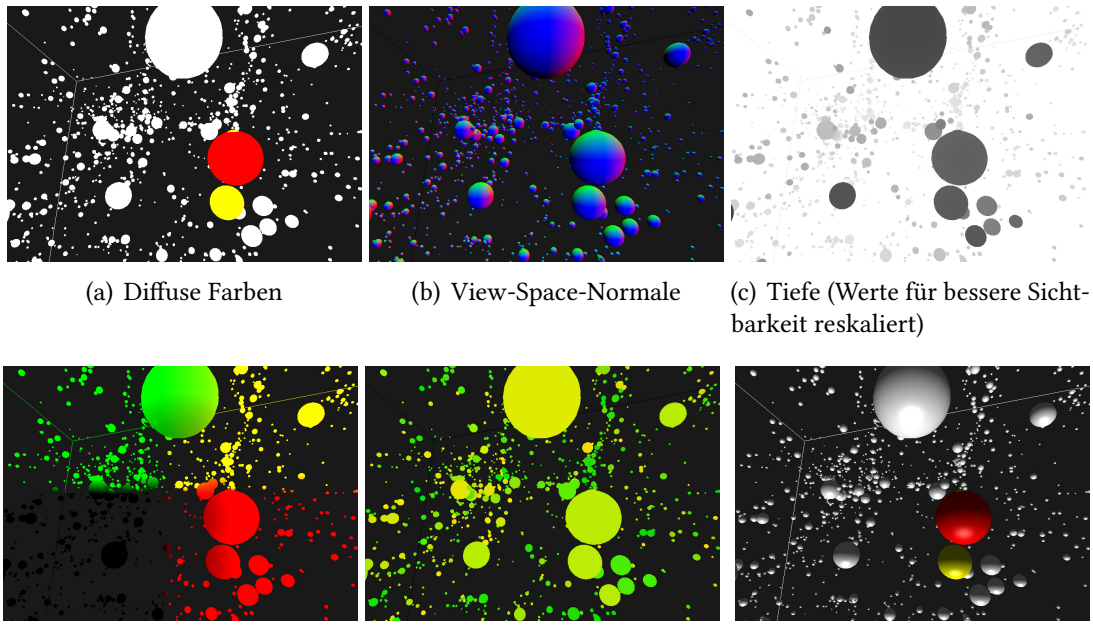
$$I = I_a + I_d + I_s = k_a + k_d \vec{L} \cdot \vec{N} + k_s (\vec{H} \cdot \vec{N})^\alpha \tag{4.2}$$

Dabei bezeichnen $I_a$, $I_d$ und $I_s$ den ambienten, diffusen und spekularen Intensitätsanteil des Lichts. $L$ ist der Vektor, der vom beleuchteten Punkt zur Lichtquelle zeigt, $N$ der Normalenvektor und $H$ ist der Halfway-Vektor, also die Winkelhalbierende zwischen $L$ und $V$, dem Vektor zum Betrachter. Die $k_i$ sind die einfallende Lichtintensität multipliziert mit jeweils einem benutzerbestimmten Faktor. Auch der Exponent $\alpha$ kann frei gewählt werden. Will man reale Materialen simulieren, können diese Faktoren experimentell bestimmt werden.

Neben den Ellipsoiden existieren in der Visualisierung noch andere Objekte, die beleuchtet werden müssen. Dazu gehören die Merger-Bäume. Wie diese beleuchtet werden, wird in Abschnitt 4.7.2 diskutiert. Zusätzlich zur konventionellen Beleuchtung werden noch zwei weitere Verfahren eingesetzt. Zum einen ist es das Deferred Shading, welches die Lichtberechnung von der Geometrieberechnung trennt (vgl. Abschnitt 4.7.3), zum anderen ist es das so genannte Depth Darkening (vgl. 4.7.3), das zusätzliche Schatten in die Szene einfügt, um Tiefenunterschiede deutlich zu machen.

### 4.7.1 Deferred Shading

Deferred Shading (etwa: „verzögerte Schattierung“,vgl. [DWS+88] u. [ST90]) bezeichnet ein Verfahren, dass es ermöglicht, die Geometrieverarbeitung von der Beleuchtung zu trennen. In der Praxis wird dies meist verwendet, um die Kosten der Beleuchtungsberechnung bei vielen



(a) Diffuse Farben (b) View-Space-Normale (c) Tiefe (Werte für bessere Sichtbarkeit reskaliert)

(d) View-Space-Positionen (e) Objekt-IDs (als Farbe kodiert) (f) Finales Bild

**Abbildung 4.8:** Verwendete Texturen für das Deffered Shading inklusive des finalen Bildes.

Lichtquellen zu senken. Hier jedoch wird es verwendet, um das weiter unten beschriebene Depth Darkening implementieren zu können. Zusätzlich dazu vereinfacht es das Picking, also die Möglichkeit für den Benutzer, bestimmte Objekte auf dem Bildschirm per Mausklick markieren zu können.

Für das Deferred Shading werden zunächst alle die Geometrie betreffende Berechnungen abgeschlossen und dann in mehreren Texturen abgespeichert. In einem zweiten Schritt werden dann diese Texturen zur Beleuchtungsberechnung herangezogen und das finale Bild auf dem Bildschirm ausgegeben. Zu den Texturen gehören die Tiefentextur, die View-Space-Normalen und die diffusen Objektfarben. Alle hier verwendeten Texturen sind in Abbildung 4.8 gezeigt. Besonders sei auf das Bild 4.8(e) hingewiesen, welches die ID-Textur zeigt. Diese wird für das Picking verwendet. Klickt der Benutzer auf eine bestimmte Stelle des Bildes, kann so einfach an der entsprechenden Stelle in der Textur nachgeschlagen werden, welches Objekt angeklickt wurde. Konventionell wäre es vonnöten gewesen, die Szene erneut zu rendern, nur um dieselbe Information zu erhalten.

Die Tiefentextur wird später noch für das Depth Darkening (vgl. Abschnitt 4.7.3) verwendet werden. Zusätzlich zu den in der Abbildung dargestellten Texturen wird dann noch eine weitere hinzukommen, deren Zweck weiter unten noch weiter erläutert wird.

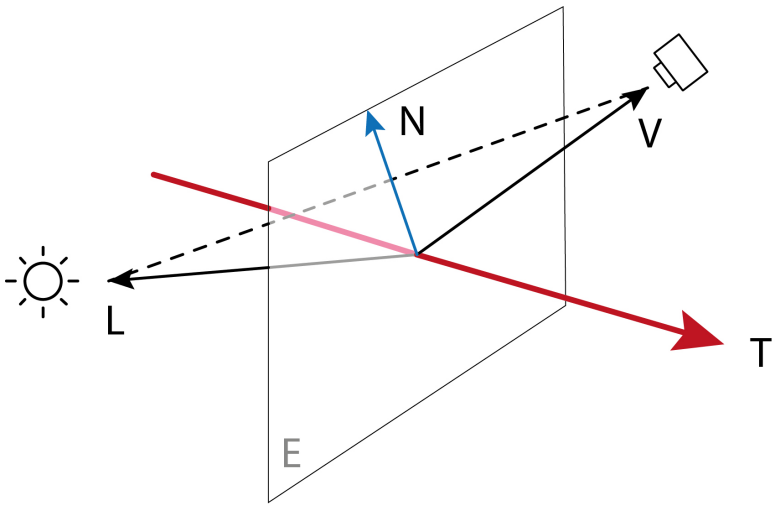



**Abbildung 4.9:** Schema für die Beleuchtung von Linie $\vec{T}$. $\vec{L}$ zeigt zur Lichtquelle, $\vec{V}$ zum Betrachter. $\vec{N}$ ist die gesuchte Normale (Abbildung nach [SZH97]).

### 4.7.2 Line Lighting

Die neben den die Halos darstellenden Ellipsoiden vornehmlich dargestellten Primitive sind die Linien der Merger-Bäume. Da die Linien keine räumliche Ausdehnung bzw. kein Volumen besitzen, existiert keine offensichtliche Wahl für die Oberflächennormalen. Das macht die Beleuchtung schwierig, weil diese ohne brauchbare Normale nicht machbar ist. Ohne Beleuchtung ist die genaue Richtung der Linie im dreidimensionalen Raum leider nicht auszumachen, was eine genaue Interpretation der Bilder erschwert.

Für die Beleuchtung der Linien wird eine Arbeit von Mallo et al. [MPSS05] herangezogen. Diese bauen ihrerseits auf der Arbeit von Banks [Ban94] und der von Stalling et al. [SZH97] auf. Diese verwenden statt dem Blinn-Phong-Beleuchtungsmodell das Phong-Modell [Pho75], welches sich von diesem aber nur im spekularen Term $I_s$ unterscheidet. Der lautet wie folgt:

$$I_s = k_s(\vec{V} \cdot \vec{R})^\alpha \tag{4.3}$$

Der einzige hier noch nicht bekannte Wert ist $\vec{R}$, der dem am Normalenvektor $\vec{N}$ gespiegelten Lichtvektor $\vec{L}$ entspricht.

Die einzige Bedingung, die für die gesuchte Normale existiert, ist die, dass sie orthogonal auf der assoziierten Oberfläche zu stehen hat. Die gesuchte Normale $\vec{N}$ hier muss also im rechten Winkel zur Richtung der zu beleuchteten Linie $\vec{T}$ stehen. Das schränkt den Raum, in dem $\vec{N}$ zu suchen ist, auf eine Ebene $\vec{E}$ ein, deren eigene Normale wiederum $\vec{T}$ ist. Die beschriebene Situation ist in Abbildung 4.9 gezeigt.

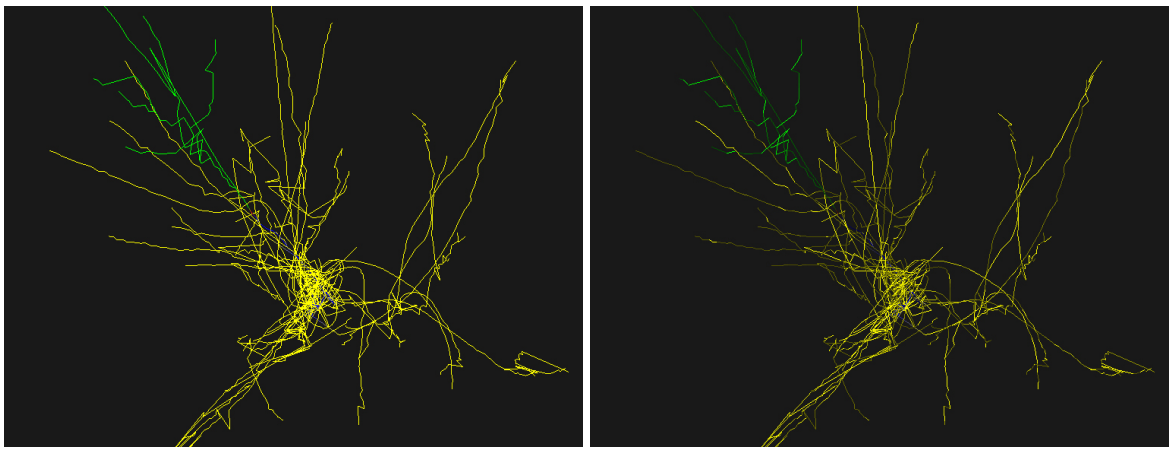


**Abbildung 4.10:** Vergleich von unbeleuchteten und beleuchteten Linien.

Um die Berechnung durchführen zu können, wird noch die Bitangente $B$ benötigt, welche orthogonal auf $T$ und $V$ steht. In dem Prozess wird $\vec{N}$ selbst nicht direkt berechnet. Die gezeigte Vorstellung von $\vec{N}$ hilft jedoch, korrekte Gleichungen für $I_a$, $I_d$ und $I_s$ zu finden. Der Lichtvektor kann dann wie folgt ausgedrückt werden [MPSS05]:

$$\vec{L} = \begin{pmatrix} L_T \\ L_N \\ L_B \end{pmatrix} = \begin{pmatrix} L_T \\ \sqrt{1 - L_T^2} \cos \beta \\ \sqrt{1 - L_T^2} \sin \beta \end{pmatrix} \tag{4.4}$$

Die $L_i$ sind dabei die auf den Vektor $i$ projizierten Komponenten von $\vec{L}$, $\beta$ der Winkel zwischen $\vec{N}$ und der Projektion von $\vec{L}$ auf die Ebene, die von $\vec{N}$ und $\vec{B}$ aufgespannt wird. Laut Phong-Modell hängt die diffuse Reflexion vom Skalarprodukt von $\vec{L}$ und $\vec{N}$ ab. $\vec{N}$ kann in derselben Basis ausgedrückt werden, wie das mit $\vec{L}$ geschehen ist. Tut man das und maximiert man das Resultat, ist die resultierende Gleichung für den diffusen Term [MPSS05]:

$$I_d = k_d \sqrt{1 - L_T^2} \tag{4.5}$$

Dasselbe kann nun für den spekularen Term durchgeführt werden. Dabei werden dann jedoch die Vektoren $R$ und $V$ verwendet, was in folgender Gleichung resultiert [MPSS05]:

$$I_s = k_s(-V_T L_T + \sqrt{1 - V_T^2}\sqrt{1 - L_T^2})^\alpha \tag{4.6}$$

Der ambiente Term bleibt gleich, da dieser nicht von der zu berechnenden Normalen abhängt. So sind nun alle nötigen Terme zur Anwendung des Phong-Modells zur Beleuchtung bekannt. Die Beleuchtung kann durchgeführt werden.

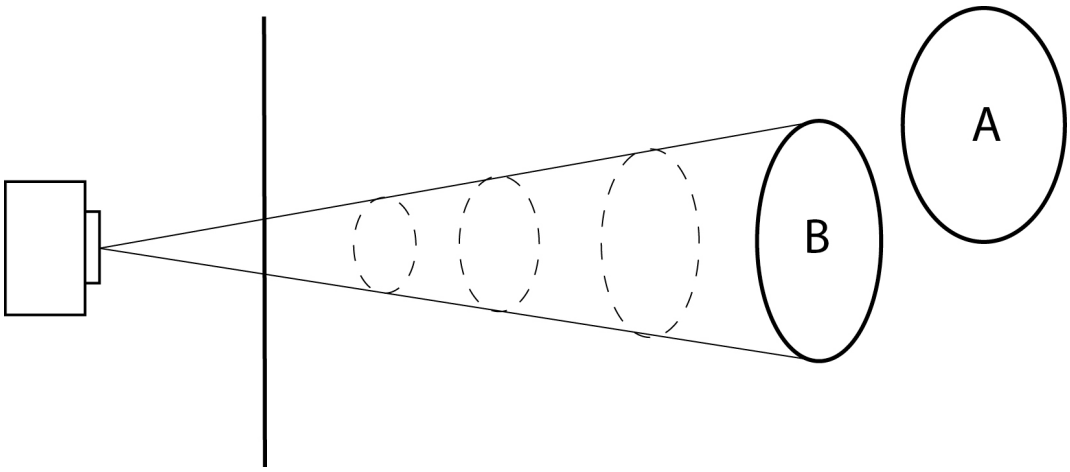


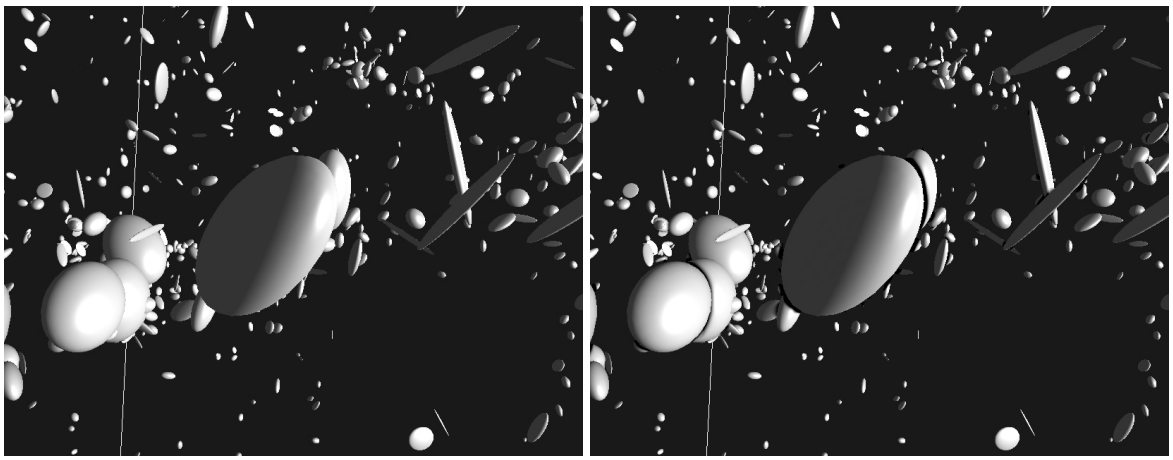

**Abbildung 4.11:** Wie das Objekt $B$ im Vergleich zu Objekt $A$ von der Kamera entfernt ist, ist in der Abbildung nicht direkt erkennbar.

Die Effekte der Beleuchtung sind in Abbildung 4.10 gezeigt. Im rechten beleuchteten Teil weisen Linien, die links als gerade erschienen, deutliche Beleuchtungsänderungen auf. Das heißt, dass das, was als Gerade erschien, dort plötzlich grundlegend die Richtung geändert hat. Der Benutzer hat, wenn so eine Stelle entdeckt wird, die Möglichkeit, dies genauer durch Bewegung der Kamera zu untersuchen. Dabei kann dann die wirkliche Ausrichtung der Linie erkannt werden.

### 4.7.3 Depth Darkening

Durch die Projektion vom dreidimensionalen Raum auf den zweidimensionalen Bildschirm geht die Tiefeninformation verloren. Insbesondere bei abstrakten Objekten kann dies zu Fehleinschätzungen führen, wenn keinerlei Bezugsrahmen für die Größe von Objekten existiert. Sieht man beispielsweise auf einem Bild zwei Stühle, der eine groß abgebildet, der andere klein, geht man als Betrachter in den meisten Fällen davon aus, dass sich der größere näher an der Kamera befunden haben muss. Das rührt daher, dass die meisten Stühle in dieser Welt eine ähnliche Größe besitzen und der Betrachter somit einen Bezugsrahmen hat, in den er die Abbildung setzen kann. Hier werden jedoch Objekte abgebildet, die keine festgelegte Größe besitzen, was auch in Abbildung 4.11 gezeigt ist. Jedes der gestrichelten Ellipsoide könnte die wahre Position von $B$ sein, die Projektion auf die Bildebene liefert darüber jedoch keine Information mehr.

Da es sich bei der hier um eine interaktive Visualisierung handelt, hat der Benutzer natürlich die Möglichkeit, die Kameraposition zu verändern, um die eigentlichen Tiefenverhältnisse genau zu untersuchen. Bietet sich jedoch kein Anlass, um diese Untersuchung vorzunehmen, wird eventuell von falschen Tatsachen ausgegangen. Um dem entgegenzuwirken, wurde



(a) ohne Depth Darkening (b) mit Depth Darkening

**Abbildung 4.12:** Vergleich von Depth Darkening zum Ursprungszustand.

hier die Methode des Depth Darkening implementiert, welche auf der Arbeit von Luft et al. basiert [LCD06].

Die Idee beim Depth Darkening ist es, weiter vorne liegende Objekte auf weiter hinter liegende einen Schatten werfen zu lassen, um diese visuell voneinander zu trennen. Diese Schatten sind nicht zwangsläufig physikalisch korrekt und dienen nur der optischen Trennung. Die Schattierung wird durch eine Modulierung des finalen Bildes mit dem Tiefenpuffer der Rendering-Pipeline erreicht. Das geschieht wie folgt.

Der Tiefenpuffer $D$ wird zunächst mittels eines Gauss-Filters $G$ gefiltert und die Differenz $\Delta D$ zum alten Tiefenpuffer wird berechnet [LCD06]:

$$\Delta D = G * D - D \tag{4.7}$$

Das Differenzbild $\Delta D$ ist vor allem um die Bereiche $\neq 0$, an denen in $D$ große Änderungen in der Tiefe stattfinden. Das entspricht schon den Bereichen, die schattiert werden sollen. Jedoch wollen wir nicht, dass das vordere Objekt auch schattiert wird. Daher wird zur Filterung des Originalbildes nicht $\Delta D$ sondern $\Delta D^-$ verwendet, welches nur die negativen Anteile von $\Delta D$ enthält und sonst $0$ ist. In der Original-Arbeit wird empfohlen, einen Gauss-Kernel mit einer Breite von ca. $2\% - 5\%$ der Bilddiagonalen zu verwenden, daher wurde hier ein $41 \times 41$-Kernel benutzt, in Annahme einer Auflösung von $1920 \times 1080$. Für eine schnellere Berechnung wurde dieser separiert.

Das resultierende Bild $I'$ wird dann letztendlich wie folgt berechnet, wobei $I$ das Ursprungsbild ist [LCD06]:

$$I' = I + \lambda \cdot \Delta D^- \tag{4.8}$$

Bei $\lambda$ handelt es sich um einen positiven Faktor, mit dessen Hilfe die Stärke des Effekts des Depth Darkenings geregelt werden kann. Ein Resultat dieses Prozesses ist in Abbildung 4.12 gezeigt. Besonders gut sichtbar ist der Effekt am rechten Rand des zentral abgebildeten großen Ellipsoids. Im linken Bild kann an dieser Stelle nicht festgestellt werden, wie weit die beiden Halos in der Tiefe voneinander entfernt sind. Der schwarze Rand in der rechten Abbildung löst diese Problematik. Wenn bei einer solchen Überdeckung ein schwarzer Schatten wahrgenommen werden kann, ist sicher, dass die beiden Objekte nicht direkt beieinander liegen. Je weiter sie dabei auseinander liegen, desto stärker wird der resultierende Schatten.

## 4.8 Partikelzugehörigkeiten

Jeder in der Visualisierung dargestellte Halo bestand in der vorangegangen Simulation aus einzelnen Partikeln. Auch wenn jetzt schon viele Parameter der Halos visualisiert werden können, wäre es vorteilhaft, wenn für jeden Halo seine zugehörigen Partikel angezeigt werden könnten. Für die resultierenden Halos des im Rahmen dieser Arbeit implementierten Algorithmus von Widanagamaachchi et al. ([WBS+14], vgl. auch Abschnitt 3.5) ist das kein Problem, da die Halo-Zugehörigkeiten für jedes Partikel einfach herausgeschrieben werden können. Bei den Resultaten des Rockstar-Algorithmus ([BWW13], vgl. auch Abschnitt 3.4) ist das jedoch nicht möglich, da hier kein Einfluss auf die bereitgestellten Daten genommen werden kann. Im Falle des Rockstar-Algorithmus müssen wir uns hier also anders behelfen.

### 4.8.1 Postprocessing der Rockstar-Daten

Da bei den Resultaten des Rockstar-Algorithmus die von der SciVis-Simulation stammen keine Informationen über die Partikelzugehörigkeiten beiliegen, ist es vonnöten diese aus den zur Verfügung stehenden Informationen abzuleiten. Hierzu werden primär der Virialradius $R_{vir}$, die Positionen $\vec{x}$ der Halos und die Ellipsoid- Daten, also Hauptachse $\vec{a}$ und die Relationen zu den Nebenachsen $b$ und $c$, verwendet. Die Grundidee ist es, für jedes Partikel zu überprüfen, ob es zum einen innerhalb des Virialradius eines Halos liegt und zum anderen, ob es in dessen angegebenem Ellipsoid liegt. Natürlich hat diese Methode keine Garantie auf Korrektheit, aber leider stehen nicht genug Informationen zur Verfügung, um die Resultate des Rockstar-Algorithmus genau zu emulieren.

Der erste Schritt besteht darin, für jedes Halo zu überprüfen, welche Partikel sich innerhalb seines Virialradius befinden. Naiv durchgeführt wäre dieser Schritt relativ teuer, deshalb wird hier die Approximate Nearest Neighbor Bibliothek [MA10] verwendet, welche geeignete Datenstrukturen und Operationen bereitstellt, um schnelle Nachbarschaftssuchen zu betreiben. Zusätzlich macht sich der folgende Algorithmus den Fakt zunutze, dass die angegebenen

Ellipsoide immer von der von $R_{vir}$ beschriebenen Kugel umschlossen werden. Der Prozess der Zugehörigkeitssuche funktioniert dann wie folgt:

1. Alle Partikel werden in den von der Bibliothek bereitgestellten kD-Baum (vgl. [Ben75]) eingefügt.
2. Pro Halo werden von der Datenstruktur alle Partikel erfragt, welche sich innerhalb seines $R_{vir}$ befinden.
3. Pro Partikel werden die Halo-Zugehörigkeiten für $R_{vir}$ zugewiesen. Dabei kann es passieren, dass sich ein Partikel innerhalb mehrerer Halos befindet. Sollte es sich dabei um Paare aus Halo und Sub-Halo halten, so wird es dem Sub-Halo zugewiesen. Falls das nicht der Fall ist, wird der schwerere Halo gewählt. Bei komplizierteren Konstellationen wird immer zunächst nach Halo-Masse entschieden und dann werden erst die Sub-Halos betrachtet.
4. Pro Halo werden alle ihm zugewiesenen Partikel nach Zugehörigkeit zum Ellipsoid des Halos überprüft. Das geschieht, indem zunächst pro Halo die inverse Model-Matrix für sein Ellipsoid berechnet wird. Daraufhin wird die berechnete Inverse mit der homogenen Partikelposition multipliziert, um einfacher im Modell-Raum arbeiten zu können. Dort kann einfach überprüft werden, ob sich ein Punkt näher als $1$ am Ursprung befindet. Wenn ja, ist er innerhalb des Ellipsoids, wenn nein, dann nicht.
5. Die Halo-Zugehörigkeiten für die Ellipsoide werden pro Partikel zugewiesen. Die Partikel können hier entweder dem Halo zugewiesen werden, dem sie schon für $R_{vir}$ zugewiesen wurden oder keinem.
6. Die Resultate werden auf die Festplatte geschrieben.

Für diesen Prozess wurde eine eigene Anwendung geschrieben, die Merger-Baum-Datei einliest, für jeden Zeitschritt einzeln die Partikel-Datei liest, die Zugehörigkeiten überprüft und dann das Resultat wieder auf die Festplatte ausgibt. Die resultierenden Dateien können dann von der Visualisierungsanwendung eingelesen und zur Zuordnung verwendet werden.

# 5 Resultate

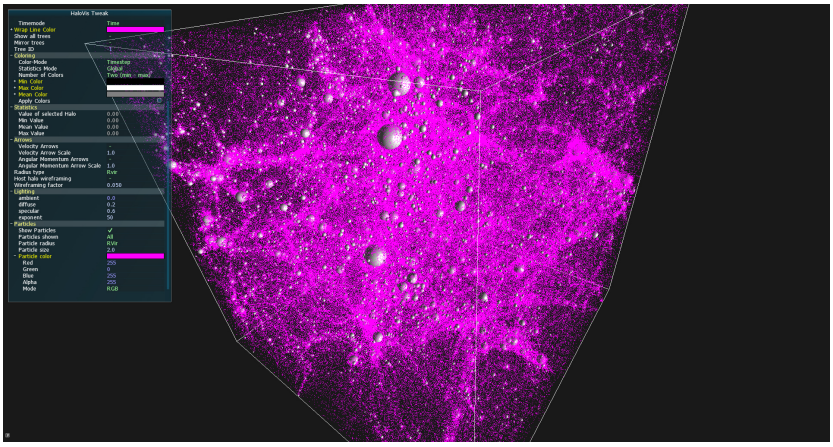
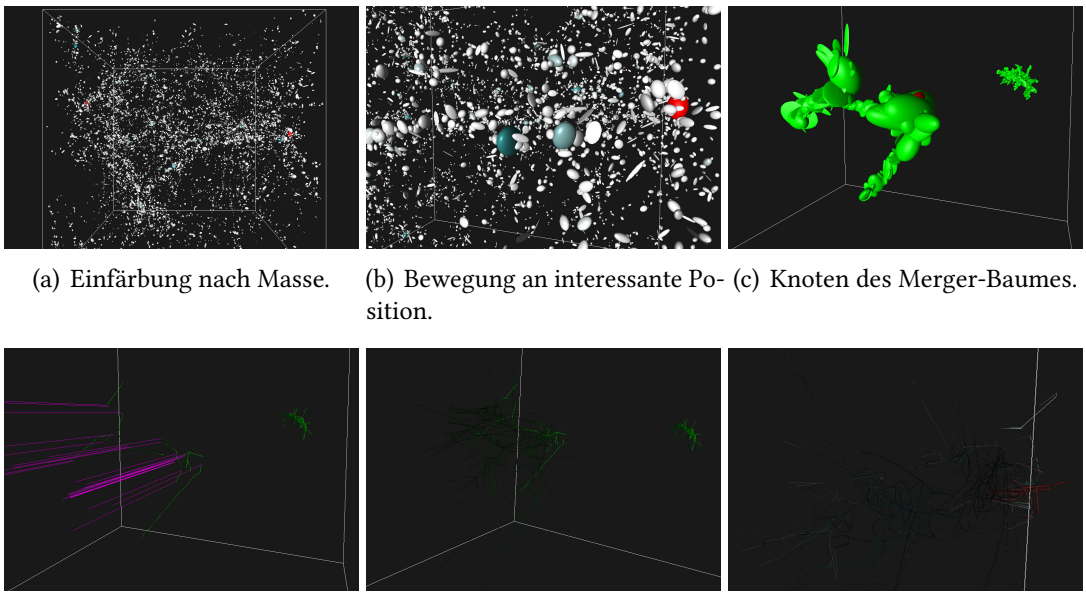



**Abbildung 5.1:** Kompletter Screenshot der Visualisierung der Halos des letzten Zeitschritts, inklusive Partikel und Steuerelemente.

Im Laufe dieser Arbeit wurde mit Hilfe der weiter oben beschriebenen Methoden zum einen ein Halo-Finder und zum anderen eine Halo-Visualisierung implementiert. Die Resultate sollen im Folgenden präsentiert und evaluiert werden. In Abschnitt 5.1 werden dazu zunächst die Resultate der Visualisierung diskutiert. Daraufhin werden in Sektion 5.2 der selbst implementierte Halo-Finder mit dem in der DarkSky-Simulation verwendeten Rockstar-Algorithmus verglichen. Abschließend wird dann noch auf die Performance-Aspekte von sowohl der Visialisierung als auch dem Halo-Finder eingegangen.

## 5.1 Visualisierung der Halos

Die resultierende Visualisierungsanwendung ist in der Lage sowohl die Resultate des Rockstar-Algorithmus des SciVis-Contest- Datensatzes als auch des selbst implementierten Halo-Finders in Bildmaterial umzuwandeln. Zusätzlich können parallel dazu die Partikeldatensätze geladen und angezeigt werden, was auch Abbildung 5.1 der Fall ist. Diese zeigt einen vollständigen Screenshot der Anwendung. Links hat der Benutzer die Möglichkeit, verschiedenste Einstellungen vorzunehmen, wie die Farbe des Hintergrundes, das Anzeigen der Bounding



(a) Einfärbung nach Masse. (b) Bewegung an interessante Position. (c) Knoten des Merger-Baumes.

(d) Kanten des Merger-Baumes. (e) Spiegeln des Baumes. (f) Gespiegelter Merger-Baum nach Zeit eingefärbt.

**Abbildung 5.2:** Typischer Prozess der Anwendung des Visualisierungsprogramms.

Box und die spezifischeren Einstellungen für die weiter oben erläuterten Visualisierungstechniken. Zusätzlich kann die Kamera im dreidimensionalen Modell frei mittels Maus und Tastatur oder nur der Tastatur bewegt und rotiert werden.

### 5.1.1 Technische Details

Die Anwendung verwendet OpenGL 4.3 zur Visualisierung und wurde in C++ geschrieben. Zu den verwendeten Bibliotheken zählen die SDF-Bibliothek [SW14] zum Laden der Partikeldaten, GLM für die OpenGL-spezifische Mathematik [Ric05] und TheLib [MRG12], welche größtenteils für das Parsing der Eingabedaten verwendet wurde. Für die auch in der Abbildung zu sehende GUI wurde die AntTweakBar [Dec05] als Komponente hinzugefügt.

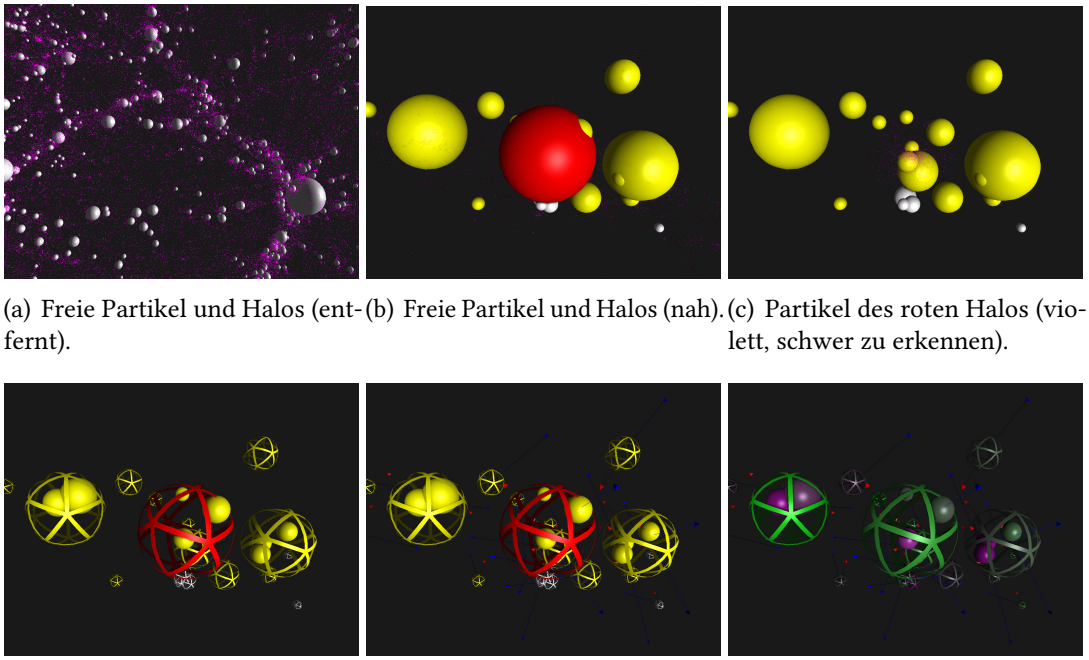
Für die Abbildung der Halo-Ellipsoide wurde, wie in Abschnitt 4.3 beschrieben, Tessellation verwendet. Das Spiegeln der Merger-Bäume (vgl. Abschnitt 4.5) geschieht mit Hilfe eines Geometry-Shaders. Dafür wird pro Baum vorberechnet, ob er überhaupt gespiegelt werden muss. Wenn das der Fall ist, wird pro Baum die Richtung, in die er gespiegelt werden muss, gespeichert. Neben den Spiegelungen der Merger-Bäume werden zudem noch die Statistiken pro Baum, Zeitschritt und global vorberechnet. Zudem werden noch verschiedene Sortierungen des kompletten Datensatzes erstellt, beispielsweise nach ID, nach Zeit oder nach Baum.

Zur Visualisierung der Linien werden normale GL_LINES verwendet, die – wie weiter oben beschrieben – beleuchtet werden. Die Visualisierung der einzelnen Partikel Dunkler Materie geschieht mittels simpler GL_POINTS, was auch angesichts deren purer Menge erforderlich ist, um eine flüssige Darstellung zu gewährleisten. Da das Laden der Partikel eines jeden Zeitschritts auf einmal nicht möglich ist, werden diese immer beim Wechsel eines Zeitschritts nachgeladen.

### 5.1.2 Typischer Anwendungsfall

Der typische Prozess zur Analyse der Daten soll im Folgenden beschrieben werden. Nach dem Laden und dem Preprocessing der Daten werden die Halos des kompletten Datensatzes angezeigt. Der Benutzer hat dann die Möglichkeit, diese – je nach Präferenz – unterschiedlich einzufärben. In Abbildung 5.2(a) ist eine solche eingefärbte Totale gezeigt. Die Färbung geschieht dort nach der Virialmasse von Weiß über Türkis nach Rot. Da sehr viele leichte Halos existieren, sind wahrscheinlich die türkisen und roten Halos die interessanten. Der Rest des Prozesses ist auch chronologisch in Abbildung 5.2 gezeigt. Die Kamera kann, wenn die interessanten Stellen ausgemacht sind, in eine bessere Position manövriert werden. Ein interessanter Halo kann ausgewählt und dessen Historie untersucht werden. In Abbildung 5.2(b) fiel die Entscheidung auf den helleren der beiden zentral angezeigten türkisen Halos. Dessen Merger-Baum wird dann in Abbildung 5.2(c) und 5.3(d) gezeigt. Schon in der Knotenansicht (c) ist zu erkennen, dass Vorfahren des ausgewählten Halos (Vorfahren in Grün, der Halo selbst in Rot) sich auf der anderen Seite des Simulationsraums befinden. Dies geschah, wie weiter oben bereits erläutert, durch die periodischen Randbedingungen. Ist das in der Knoten-Ansicht nicht zu erkennen, erlangt man in der Kanten-Ansicht (d) die Gewissheit, da dort alle Kanten, welche über den Rand hinausragen und somit auf der anderen Seite wieder eintreten würden, violett dargestellt werden. Ist so eine Stelle gefunden, kann, wie in (e) gezeigt, der Rest des Baumes von der anderen Seite auf dieser Seite desaturiert gezeigt werden. Hier fällt auf, dass sich der Großteil eigentlich auf der anderen Seite befindet. Zusätzlich dazu können auch die Merger-Bäume, genau so wie auch die Halo-Ellipsoide, nach bestimmten Werten eingefärbt werden. In (f) geschieht dies nach Zeit, mit demselben Farbschlüssel wie bei der Einfärbung nach Masse vorhin. Hier fällt auf, dass der Großteil der Historie des Halos auf der anderen Seite des Simulationsbereichs stattfand und erst recht spät den Rand überschritten hat.

Neben dem Analyseprozess für die Historie eines Halos kann auch die genauere Struktur in der Gegenwart des aktuellen Zeitschritts untersucht werden. Dazu werden in Abbildung 5.3(a) zusätzlich die Partikel in violett angezeigt. Dort fällt auf, dass die Dunkle Materie faserige Strukturen bildet, auf denen sich die Halos größtenteils befinden. Massereiche, große Halos sind meist an Knotenpunkten dieser Strukturen. Ist ein interessanter Halo ausgemacht, kann dieser, wie oben auch, selektiert werden. In Abbildung 5.3(b) ist das der rote Halo, die gelben werden sich in späteren Zeitschritten mit diesem vereinen. In 5.3(c) werden statt



(a) Freie Partikel und Halos (entfernt).

(b) Freie Partikel und Halos (nah).

(c) Partikel des roten Halos (violett, schwer zu erkennen).

(d) Sub-Halos.

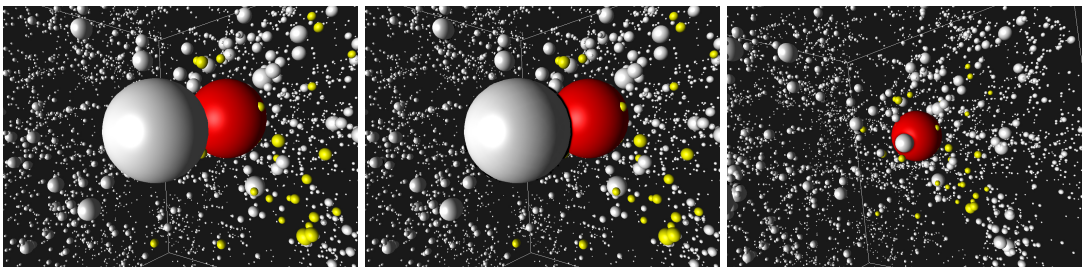
(e) Geschwindigkeitspfeile.

(f) Einfärbung nach Geschwindigkeitsbetrag.

**Abbildung 5.3:** Detailansicht eines oder mehrerer Halos.

des Ellipsoids die Partikel angezeigt, die sich innerhalb des Virialradius des roten Halos befinden. Die Partikel sind aufgrund deren geringer Größe in der Abbildung leider nur schwer zu erkennen. Bei der Ansicht der Partikel fällt auf, dass sich innerhalb des Ellipsoids noch andere, kleinere Halos befinden. Diese können mit Hilfe des speziellen Wireframe-Modus genauer untersucht werden (vgl. 5.3(d)). Nun könnte etwa noch interessant sein, in welche Richtung sich die gezeigten Halos bewegen und um welche Achse diese rotieren. Dazu werden in 5.3(e) die Pfeile zugeschaltet, welche genau diese Dinge anzeigen. Die Geschwindigkeitsvektoren (rot) wurden dabei logarithmisch skaliert, die Vektoren für das Drehmoment (blau) sogar doppelt logarithmisch. Die auftretenden Werte im Datensatz sind für eine direkte Visualisierung leider zu groß. Da durch die Projektion auf die Bildfläche die genaue Richtung und somit auch der genaue Betrag eines Vektors nicht mehr eingeschätzt werden kann, kann nun auch jeder Halo nach dem Betrag seiner Geschwindigkeit eingefärbt werden. Das lässt den Benutzer einfacher jene Halos finden, deren Geschwindigkeitsvektor eine bestimmte Länge besitzt, auch wenn das aus der Abbildung des Vektors nicht deutlich wird.

Durch die Projektion auf die Bildebene gehen neben den genauen Vektor-Beträgen leider auch die Tiefeninformationen der Halos verloren. Diesem wird, wie in Abschnitt 4.7.3 beschrieben, durch Depth Darkening entgegengewirkt. Ein Fall davon ist in Abbildung 5.4 gezeigt. Links



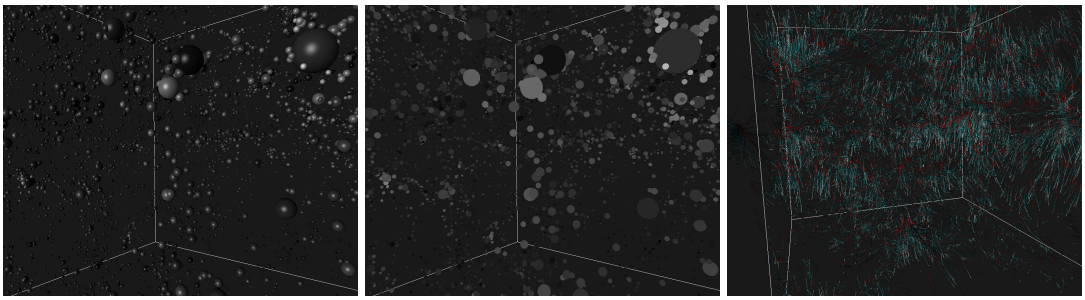
(a) Ohne Depth Darkening. (b) Mit Depth Darkening. (c) Herausgezoomt.

**Abbildung 5.4:** Der Effekt des Depth Darkening.

ist die Ansicht ohne Depth Darkening zu sehen. Der groß dargestellte weiße Halo befindet sich leicht vor dem roten. Er könnte sich nun sehr nahe beim Roten befinden und somit sehr klein sein. Die zweite Möglichkeit ist, dass es sich um einen relativ kleinen Halo handelt, der sich nur zufällig nahe an der Kamera befindet. Schaltet man an dieser Stelle, wie in der mittleren Abbildung gezeigt, das Depth-Darkening ein, wirft der weiße Halo einen deutlichen Schatten auf den roten. Das bedeutet, dass diese beiden relativ weit auseinander sein müssen und der weiße Halo folglich im Vergleich zum roten eher klein sein muss. Ein Herausfahren der Kamera, wie rechts gezeigt, bestätigt diesen Eindruck.

### 5.1.3 Problematiken

Einige der soeben erläuterten Analysemöglichkeiten werfen leider diverse Probleme auf. So ist das menschliche Auge kaum dazu in der Lage, die Farben von weiter auseinanderliegenden Flächen zu vergleichen. Zusätzlich erschwert die Beleuchtung der Halo-Ellipsoide einen



(a) Schwer zu vergleichende Farben (b) Reines Anzeigen der diffusen Farben. (c) Alle Merger-Bäume gleichzeitig.

**Abbildung 5.5:** Problematische Stellen in der Visualisierung.

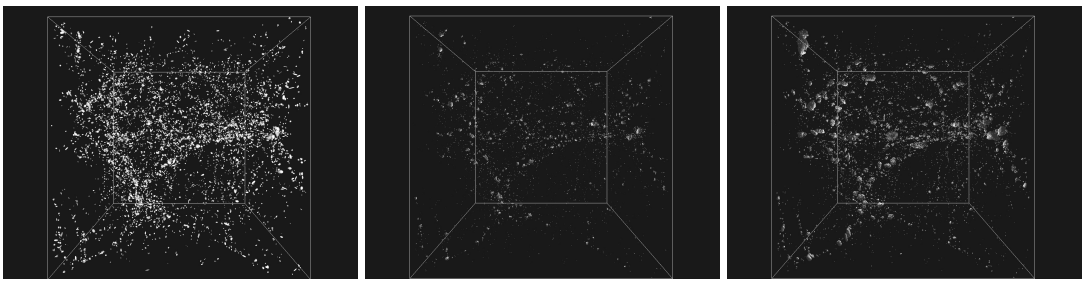
Vergleich der Farben, was in Abbildung 5.5(a) deutlich wird. Durch die Wahl von Grautönen als Farbe wird dort der Effekt noch verstärkt, weil eine Beleuchtung mit einer weißen Lichtquelle die wahrgenommenen Grauwerte verändert. Das Resultat sind ebenfalls Grauwerte, welche sich jedoch über die Ellipsoide hinweg ändern, was die wahre Farbe verschleiert. Dem kann zum Glück mit Hilfe des Deferred Shading entgegengewirkt werden, indem man temporär nur die Farbtextur für die diffuse Farbe anzeigt, was in 5.5(b) dargestellt wird. Eine Notlösung wie diese sollte aber im Normalfall nicht angewendet werden, vor allem weil sie das Problem des Vergleichs von weit auseinander liegenden Halos ebenfalls nicht löst. Abhilfe würde eventuell eine Legende schaffen, welche alle verfügbaren Farben auf einmal anzeigt und mit deren Hilfe ein genauer Wert für jeden Halo festgelegt werden kann.

Das Erkennen der Position eines Halos ist zum Glück in der Kombination mit einer frei beweglichen Kamera ein Leichtes, insbesondere wenn eine Bounding Box als Referenz angezeigt wird. Bei den Merger-Bäumen ist das Erkennen der genauen Position leider ein größeres Problem, was besonders an der Visualisierung aller Bäume gleichzeitig sichtbar wird (vgl. Abbildung 5.5(c)). Die Bäume sind dort, wie auch in Abbildung 5.2(f) nach Zeit eingefärbt. Kleinere Halos neigen dazu, sich größeren in kreisförmigen Bahnen zu nähern. Das führt zwangsläufig dazu, dass diese Bahnen sich kreuzen und ähnliche Wege beschreiben. Wird so etwas abgebildet, ist es schwer, einem einzelnen Strang mit dem Auge zu folgen, da ständig andere mit dem im Fokus verwechselt werden können.

Die Verwechslung von Linien kann auch bei der Anzeige der Geschwindigkeits- oder Drehmomentvektoren zum Problem werden. Halos, die nahe beieinander liegen, besitzen nicht zwangsläufig ähnliche Geschwindigkeiten, deshalb weisen auch die Linien dort in viele unterschiedliche Richtungen. Vor allem die Zuordnung eines Vektors zum zugehörigen Halo bereitet hier Schwierigkeiten. Die Existenz von Sub-Halos verschlimmert diese Situation zusätzlich, da man sich nicht sicher sein kann, dass ein aus einer Oberfläche tretender Vektor auch zu diesem Halo gehört und nicht von einem Sub-Halo stammt. Außerdem können die Vektoren eines Halos in einem anderen enden, was die genau Länge desselben verbirgt. Die Wireframe-Ansicht wirkt hierbei vielen dieser Probleme entgegen, jedoch sollte hier eine bessere Lösung möglich sein.

## 5.2 Vergleich der unterschiedlichen Halo-Finder

Zusätzlich zur Visualisierung wurde der Halo-Finder mit variabler Linking Length von Widanagamaachchi et al. [WBS+14] implementiert. Dieser wurde ebenfalls in C++ geschrieben. Die Ausgabe sind hier Hierarchien von Halos desselben Zeitschritts, wobei höher in der Hierarchie stehende Halos durch eine größere Linking Length entstanden sind. Die Visualisierunganwendung kann diese Resultate ebenfalls wiedergeben, auch wenn dort die Analysemöglichkeiten im Vergleich zu den Resultaten des Rockstar-Algorithmus stark eingeschränkt sind. Da keinerlei Relationen zwischen den Zeitschritten erkannt werden, können



(a) Rockstar-Algorithmus. (b) Widanagamaachchi et al. mit $l = 100\text{kpc}\cdot h$. (c) Widanagamaachchi et al. mit $l = 200\text{kpc}\cdot h$.

**Abbildung 5.6:** Vergleich der Resultate der beiden Algorithmen.

beispielsweise keine Merger-Bäume angezeigt werden. Auch die Detektion von Sub-Halos funktioniert dabei nicht.

Da die genauen, beim Rockstar-Algorithmus verwendeten Werte zwar vorhanden sind, aber nicht genau zugeordnet werden können, kann eine passende Linking Length für den hier implementierten Algorithmus nur geschätzt werden. Laut Behroozi et al. [BWW13] ist dafür in etwa der $0{,}2$-fache mittlere Partikelabstand eine passende Wahl. Bei den vorliegenden Datensätzen entspricht das einer Linking Length von etwa $200\text{kpc}\cdot h$. Da vom Rockstar-Algorithmus aus zeitlichen Konsistenzgründen diverse Halos entfernt werden, wird hier auch eine niedrigere Linking Length von $100\text{kpc}\cdot h$ in Betracht gezogen. Die Visualisierungen dieser beiden Fälle und das Resultat des Rockstar-Algorithmus sind für den letzten Zeitschritt, für den Ergebnisse existieren ($a(t) = 0{,}98$), in Abbildung 5.6 gezeigt. Die grobe Struktur ist bei allen drei Varianten dieselbe. Besonders der von der rechten Mitte nach links unten laufende Bogen ist gut erkennbar.

Rein visuell hat es den Anschein, als hätte der Rockstar-Algorithmus mehr Halos hervorgebracht. Das ist jedoch nicht der Fall. Während im linken Bild nur etwas über $7000$ Halos gezeigt werden, sind es rechts über $30000$ und in der Mitte sogar knapp $50000$. Diese sind jedoch im Vergleich mit denen des Rockstar-Algorithmus sehr klein, was zum dunkleren Gesamteindruck der Bilder führt. Da der Rockstar-Algorithmus nur Halos mit einer Gesamtgröße von mindestens $10$ Partikeln zulässt (vgl. [BWW13]) und hier eine Mindestgröße von $20$ gewählt wurde, müssten eigentlich weniger Halos erkannt worden sein. Da von Behroozi et al. auch erwähnt wurde, dass die Eigenschaften solcher kleinen Halos jedoch meist über die Zeit hinweg nicht konsistent sind, könnten diese beim Erstellen der Merger-Bäume herausgefallen sein. Im unteren mittigen Bereich von Abbildung 5.6(b) existieren dutzende, wenn nicht hunderte, winzigster Halos, die jedoch aufgrund ihrer geringen Größe hier kaum visuell wiedergegeben werden können.

Vergleicht man Abbildung 5.6(b) mit 5.6(c), fällt auf, dass vor allem die größeren Halos mit zunehmender Linking Length an Volumen zugenommen haben. Das wird besonders

rechts in der Mitte der Abbildung oder links oben deutlich. Diese großen haben auch in den Ergebnissen des Rockstar-Algorithmus zumeist ein Pendant, auch wenn es hier passieren kann, dass einer dieser großen Rockstar-Halos durch mehrere dicht nebeneinanderliegende kleine dargestellt wird. Ob es sich dabei um einen Fehler im Algorithmus handelt oder ob schlicht die Linking Length zu klein gewählt wurde, kann nur vermutet werden. Dass aber einige der großen Halos recht akkurat wiedergegeben werden, lässt leider auf einen kleinen Fehler im Algorithmus schließen, außer es handelt sich dabei um sehr dichte, gut nach außen abgegrenzte Halos. Dies kann leider nur schwer überprüft werden.

Alles in allem jedoch sind die Resultate des Halo-Finders recht zufriedenstellend. Sowohl die großen als auch die kleineren Strukturen passen gut zu den bereits vorhandenen Daten.

## 5.3 Performance-Analyse

Die resultierenden Anwendungen wurden auf mehreren Maschinen auf Funktionalität hin überprüft. Sowohl auf Notebooks als auch auf im Vergleich dazu leistungsstärkeren Tower-PCs konnte diese gewährleistet werden. Der Rechner, der nun hier für die Performance-Analyse verwendet wurde, ist mit einem Intel i7-5820K 6-Kern-Prozessor mit einer Takfrequenz von $3{,}3$GHz ausgestattet. Für die Grafikberechnungen ist eine Nvidia GTX $980$ Ti Grafikkarte verbaut, der Arbeitsspeicher beläuft sich auf $16$GB. Während für die Halo-Detektion vor allem der Prozessor ausgelastet wird, ist es bei der Visualisierung natürlicherweise die Grafikkarte, die stärker beansprucht wird.

### 5.3.1 Halo-Finder

Der Halo-Finder wird komplett prozessorseitig ausgeführt, die Grafikkarte spielt keinerlei Rolle in der Berechnung der Halos. Laufzeiten für verschiedene Szenarien sind in Abbildung 5.7 aufgeführt. Zum einen wurde eine relativ kleine Linking Length von $100$kpc$\cdot h$ in Betracht gezogen. Anhand der plausibelsten Länge von $200$kpc$\cdot h$ wurde dann der Effekt der Vergrößerung der Nachbarschaft überprüft. Schlussendlich wurde auch eine viel zu große Linking Length von $2000$kpc$\cdot h$ in Betracht gezogen. Zusätzlich dazu wurde für jede dieser Varianten eine Schrittweite von $1$ und eine von $5$ verwendet, um diese Änderung ebenfalls zu berücksichtigen.

Der Algorithmus startet mit dem Einlesen der Daten und der Konstruktion des Octrees. Die Laufzeit dort ist über alle Varianten hinweg relativ konstant, da überall gleich viele Partikel (ca. 2 Millionen) eingelesen und sortiert werden müssen. Die Berechnung der initialen Halos, also jener Zellen auf dem untersten Level des Octrees, welche belegt sind, nimmt ebenfalls recht wenig Zeit in Anspruch. Dabei handelt es sich um eine simple Anfrage an die Datenstruktur.

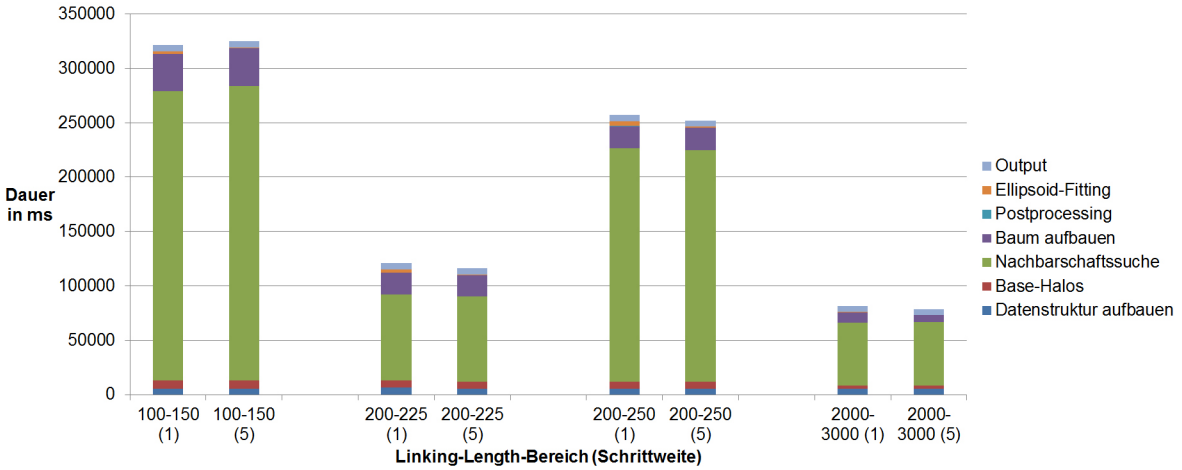



**Abbildung 5.7:** Laufzeitanalyse der einzelnen Schritte des implementierten Halo-Finders für Zeitschritt $a(t) = 0{,}98$.

Es folgt der Schritt, welcher am meisten Zeit benötigt: Die Nachbarschaftssuche. Während bei allen anderen Paaren von Durchläufen die Nachbarschaft einen Radius von $3$ hat, hat sie beim zweiten Balkenpaar der Abbildung nur einen Radius von $2$. Das führt im Vergleich zum dritten Paar zu einer starken Verringerung der Laufzeit. Der Grund dieses starken Abfalls ist recht offensichtlich, da Größe der Nachbarschaft kubisch vom Radius selbiger abhängt. Natürlich spielt auch die Anzahl der Zellen, für die eine Nachbarschaftssuche durchgeführt werden muss, eine Rolle. Während bei einer Linking Length von $100\text{kpc}\cdot h$ mit einer Anzahl von über $1{,}7$ Millionen sehr viele solcher Zellen existieren sind es bei $2000\text{kpc}\cdot h$ nur ca. $275000$. Da der Raum, in dem alles berechnet werden muss, sich nicht ändert, die benötigte Größe des untersten Levels des Octrees aber schon, ist dieser Werteunterschied leicht zu erklären. Die dadurch verursachte Laufzeit besitzt jedoch eine klare Untergrenze in Form der Anzahl der Partikel. Es können nicht mehr Zellen von Partikeln besetzt sein, als solche vorhanden sind, außer es passieren Rundungsfehler. Diese Maximalzahl ist mit einer Linking Length von $100\text{kpc}\cdot h$ bereits zu $85\%$ erreicht.

Die bisher durchgeführten Schritte werden von der gewählten Schrittweite nicht beeinflusst. Diese wird erst beim Aufbau der eigentlichen Baumstruktur wichtig. Je größer sie gewählt wird, desto geringer wird die Berechnungsdauer. Es handelt sich dabei zwar um den zweitteuersten Schritt des Gesamtprozesses, jedoch ist die Berechnungsdauer im Vergleich zur Nachbarschaftssuche immer noch vernachlässigbar. Sind diese Schritte abgeschlossen, folgt das Postprocessing, welches zu kleine Halos löscht und neue Indizes zuweist. Die Dauer dieses Prozesses ist komplett vernachlässigbar. Auch im gezeigten Diagramm ist das Postprocessing kaum zu erkennen. Es folgt das Fitting der Ellipsoide, welches stark von der Anzahl der resultierenden Halos abhängt. Da bei einer kleineren Schrittweite natürlicherweise mehr Halos erkannt werden bzw. weniger übersprungen werden, nimmt mit dieser die Dauer

dieses Prozesses zu. Die Zeitdauer der Ausgabe in eine Datei ist schlussendlich größtenteils konstant, da der Großteil der Zeit vom Ausgeben der Partikelzuordnungen beansprucht wird, deren Anzahl konstant ist. Die im Vergleich dazu wenigen Halos fallen dabei kaum ins Gewicht.

Eben weil die Nachbarschaftssuche die meiste Zeit in Anspruch nimmt, wurde auch nur diese parallelisiert. In Abbildung 5.7 wird diese parallelisierte Version gezeigt. Dafür werden alle Zellen, für die eine Suche durchgeführt werden soll, in $n$ gleich große Mengen aufgeteilt, welche an die $n$ Prozessoren verteilt werden. Bei der Berechnung wird zumeist eine Prozessorauslastung von annähernd $100\%$ erreicht. Ohne diese Parallelsierung würde allein die letzte in der Abbildung dargestellte Versuchsreihe über $10$ Minuten benötigen. All das gilt natürlich nicht für die restlichen Schritte, weil für diese nur ein Prozessorkern verwendet wird. Nichtsdestotrotz wären Schritte wie das Ellipsoid-Fitting leicht parallelisierbar.

Ein Vergleich der hier gezeigten Werte mit den Ergebnissen der Originalarbeit von Widanagamaachchi et al. [WBS+14] ist hier nicht anzuraten, da dort zum einen ein weit leistungsstärkerer Rechner verwendet und zum anderen ein anderer Datensatz benutzt wurde. Es hat jedoch den Anschein, dass dort die Nachbarschaftssuche weit schneller vonstatten gegangen ist als hier, was wohl an der speziellen räumlichen Datenstruktur liegt, die evtl. einen geringeren Overhead besitzt als der hier verwendete Octree.

### 5.3.2 Visualisierung

Die Zielsetzung bei der Performance war es, für die Visualisierung Interaktivität zu erreichen. Das wird hier so interpretiert, dass ein Mensch bei Bewegung der Kamera eine flüssige Bildfolge wahrnimmt und nicht abgehackte Einzelbilder. Das entspricht einer Mindest-Framerate von $25$ FPS oder einer Berechnungsdauer von maximal $40$ms pro Bild. Diese eigene Zielsetzung wurde erreicht, wie in Abbildung 5.8 zu erkennen ist. Für diese Erhebung wurde der Zeitschritt mit den meisten zu visualisierenden Halos verwendet. Es wird auch eine Unterscheidung nach Tessellation-Faktoren getroffen: Zum einen wurde für die Halos ein Faktor von 5, zum anderen einer von 10 verwendet. Rein visuell betrachtet ist der Unterschied zwischen diesen beiden Faktoren kaum zu erkennen, jedoch kann sich eine Erhöhung stark auf die Performance auswirken.

In der Abbildung ist der Bereich, in dem die Bildrate schwankt, rot dargestellt. Reines Rendern der Ellipsoide für die Halos ist dabei sehr schnell. Es wird eine Bildrate von ca. 250 Bildern pro Sekunde für einen Tessellation-Faktor von 5 erreicht. Eine Erhöhung auf einen Faktor von 10 verdoppelt die nötige Berechnungsdauer pro Bild annähernd. Fügt man der Visualisierung noch die Geschwindigkeits- und Drehmomentvektoren hinzu, wirkt sich das kaum auf die Performance aus. Diese wird dabei maximal um $0{,}5$ms erhöht. Auch das Darstellen eines einzelnen Merger-Baumes ist mit etwa $2{,}38$ms oder ca. $400$ FPS sehr schnell, es werden ja auch nur einige wenige GL_LINES gerendert. Einzig das Darstellen aller Merger-Bäume

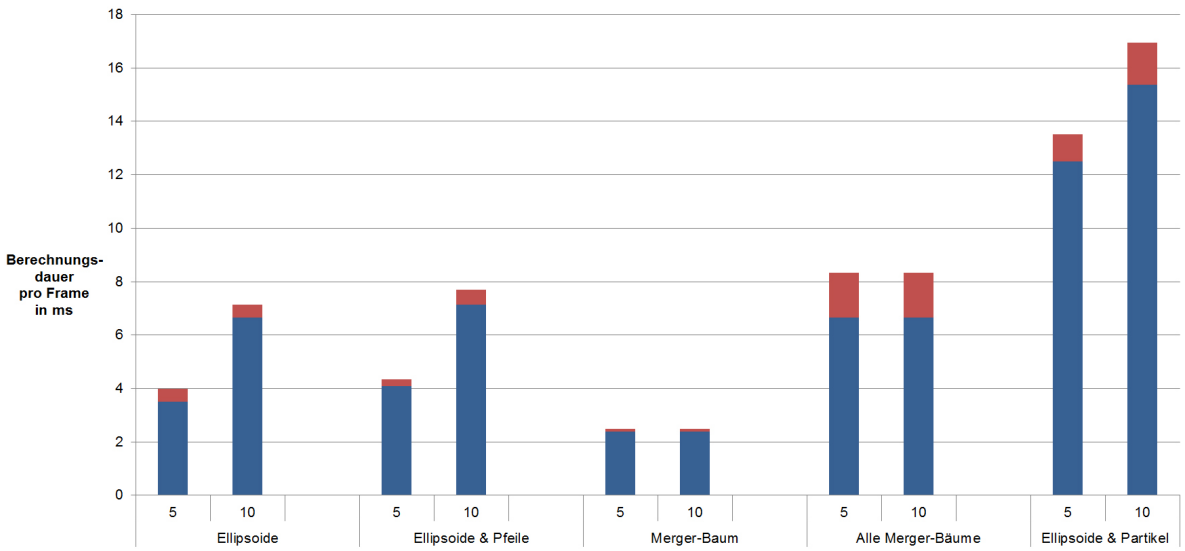



**Abbildung 5.8:** Rechenzeiten pro Frame in der Visualisierung. Der Bereich, in dem die Berechnungsdauer schwankt, wird rot dargestellt.

gleichzeitig erreicht in etwa dieselbe Berechnungsdauer pro Bild wie ein einzelner Zeitschritt visualisierter Ellipsoide. Jedoch schwankt hierbei die Framerate sehr stark. Woran das liegt, kann nur spekuliert werden.

Der Teil des Rendering, welcher sich am stärksten auf die Performance auswirkt, ist die Darstellung der einzelnen Partikel Dunkler Materie. Werden gleichzeitig noch die Halos desselben Zeitschritts dargestellt, kann die Framerate auf knapp unter 60 FPS fallen. Der Grund dafür ist wohl die schiere Anzahl der zu rendernden Partikel. Bei Visualisierungen dieser Art ist zumeist die Anzahl der anzuzeigenden Objekte das größte Problem. Sollte beispielsweise der nächstgrößere verfügbare Datensatz angezeigt werden, müssten ca. 16 Millionen Ellipsoide auf den Bildschirm gebracht werden, was mehrere Größenordnungen über der hier erreichten Maximalzahl von ca. 50000 liegt. Ohne Streaming-Methoden ist das schlichtweg unmöglich, da allein diese Daten in keinen der verfügbaren Grafikspeicher passen würden. Die Begrenztheit des Grafikspeichers ist auch der Grund, warum die Partikeldaten für jeden Zeitschritt nachgeladen werden müssen. Das nimmt pro Zeitschrittwechsel jeweils ca. 1s in Anspruch, kann jedoch durch ein deaktivieren der Partikelvisualisierung umgangen werden.

Zusätzlich zu der Performance im laufenden Betrieb ist noch die Zeit wichtig, welche bis zum Anzeigen des ersten Bildes nötig ist. Beim Start der Anwendung wird zum einen der Halo-Datensatz geladen und zum anderen werden für diesen diverse Statistiken vorberechnet. Bei dem Halo-Datensatz handelt es sich um eine reine Textdatei, die zunächst geparst werden muss, was ca. 30 Sekunden in Anspruch nimmt. Diese Zeit konnte durch ein Umwandeln

des Datensatzes in eine Binärdatei auf ca. 3s gedrückt werden. In dieser Zeit ist bereits das Vorberechnen der diversen Statistiken inbegriffen.

Alles in allem kann gesagt werden, dass für den gegebenen Datensatz die eigenen Performance-Ziele erreicht wurden. Diese hohe Performanz wird aber wohl bei größeren Datensätzen schwerlich ohne größere Modifikationen gehalten werden können.

# 6 Zusammenfassung

Im Verlauf dieser Arbeit wurde sowohl ein Halo-Finder als auch eine Halo-Visualisierung implementiert. Der Fokus lag dabei auf der Visualisierung. Die verwendeten Simulationsdaten stammen von der DarkSky-Simulation [SWT+14] und wurden im Rahmen des SciVis-Contests 2015 der IEEE [HGTS15] veröffentlicht.

Bei dem Halo-Finder für Halos Dunkler Materie handelte es sich um den von Widanagamaachchi et al. erstmals vorgestellten Friends-of-Friends-Algorithmus mit variabler Linking Length [WBS+14]. Dieser kreiert eine komplette Hierarchie von Halos, aus der die gewollte Linking Length einfach extrahiert werden kann. Für die dafür benötigte räumliche Nachbarschaftssuche wurde eine angepasste Octree-Datenstruktur verwendet. Es wurde gezeigt, dass die resultierenden Halos größtenteils mit denen des Contest-Datensatzes übereinstimmen.

Die entwickelte interaktive Visualisierungslösung ist in der Lage sowohl die Halo-Finder-Ergebnisse des Contest-Datensatzes als auch die eigenen wiederzugeben. Da der bereits gegebene Datensatz über sehr viele vorberechnete Werte verfügt, wurde die Visualisierung größtenteils diesem angepasst. Neben der Darstellung der Halos ist die präsentierte Lösung in der Lage, sowohl die zeitliche Historie der Halos als auch die Partikel darzustellen, aus denen die Halos bestehen. Hierfür wurden verschiedenste Visualisierungs- und Beleuchtungstechniken implementiert, wozu unter anderem Depth Darkening und Deferred Shading gehören. Besonders erwähnenswert ist auch die spezielle Berücksichtigung der gängigen periodischen Randbedingung in den Datensätzen. Es konnte gezeigt werden, dass die Anwendung bei der Visualisierung der verwendeten Datensätze konstant interaktive Bildraten beibehält.

## 6.1 Ausblick

Zusätzlich zu den implementierten Verfahren sind selbstverständlich noch weitere Erweiterungen möglich. So kann beispielsweise der Halo-Finder so modifiziert werden, dass auch Halos über die Zeit hinweg gefunden und nachverfolgt werden können. Zusätzlich dazu könnten noch weitere Werte für die gefundenen Halos berechnet werden, wie beispielsweise das Drehmoment.

Die Erweiterungsmöglichkeiten für die Visualisierung sind sogar noch weitaus zahlreicher. So könnte sie, mit Hilfe geeigneter Methoden, auch für größere Datensätze verwendet werden. Speziell die Visualisierung von Merger-Bäumen könnte ebenfalls verbessert werden.

So könnte man die Positionen der Halos zwischen den Zeitschritten statt – wie hier geschehen – linear beispielsweise mit Hilfe von B-Splines (vgl. u. a. [PBP02]) interpolieren. Das würde zu glatter erscheinenden Merger-Bäumen führen. Für die Visualisierung könnte auch die im Skalenfaktor $a(t)$ kodierte Positionsänderung berücksichtigt werden, um das Auseinanderdriften der universalen Strukturen zu zeigen.

Alles in Allem kann festgehalten werden, dass die Nachfrage nach geeigneten Visualisierungs-Anwendungen für die Ergebnisse großer Simulationen, wie sie in der Astrophysik durchgeführt werden, sicher nicht sinken wird. Die immer weiter steigenden Mengen an generierten Daten erfordern eher noch weitaus mehr Analysemöglichkeiten als zur Zeit zur Verfügung stehen. Da selbst für den hier verwendeten relativ kleinen Datensatz diese Aufgabe schon nicht als trivial angesehen werden kann, wird in Zukunft die Entwicklung geeigneter Methoden nicht still stehen dürfen.

# Literaturverzeichnis


[AAA+14] P. Ade, N. Aghanim, M. Alves, C. Armitage-Caplan, M. Arnaud, M. Ashdown, F. Atrio-Barandela, J. Aumont, H. Aussel, C. Baccigalupi, et al. Planck 2013 results. I. Overview of products and scientific results. *Astronomy & Astrophysics*, 571:A1, 2014. (Zitiert auf den Seiten 16 und 27)

[ABBS85] T. van Albada, J. N. Bahcall, K. Begeman, R. Sancisi. Distribution of dark matter in the spiral galaxy NGC 3198. *The Astrophysical Journal*, 295:305–313, 1985. (Zitiert auf den Seiten 7 und 23)

[AFP+06] B. Allgood, R. A. Flores, J. R. Primack, A. V. Kravtsov, R. H. Wechsler, A. Faltenbacher, J. S. Bullock. The shape of dark matter haloes: dependence on mass, redshift, radius and formation. *Monthly Notices of the Royal Astronomical Society*, 367(4):1781–1796, 2006. (Zitiert auf den Seiten 16 und 44)

[AHH+06] J. Ahrens, K. Heitmann, S. Habib, L. Ankeny, P. McCormick, J. Inman, R. Armstrong, K.-L. Ma. Quantitative and comparative visualization applied to cosmological simulations. In *Journal of Physics: Conference Series*, Band 46, S. 526. IOP Publishing, 2006. (Zitiert auf Seite 48)

[Ban94] D. C. Banks. Illumination in diverse codimensions. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, S. 327–334. ACM, 1994. (Zitiert auf Seite 61)

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. (Zitiert auf Seite 66)

[BET99] M. Bern, D. Eppstein, S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *International Journal of Computational Geometry & Applications*, 9(06):517–532, 1999. (Zitiert auf Seite 41)

[Bli77] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH Computer Graphics*, Band 11, S. 192–198. ACM, 1977. (Zitiert auf Seite 59)

[BWW13] P. S. Behroozi, R. H. Wechsler, H.-Y. Wu. The ROCKSTAR phase-space temporal halo finder and the velocity offsets of cluster cores. *The Astrophysical Journal*, 762(2):109, 2013. (Zitiert auf den Seiten 29, 33, 34, 35, 36, 44, 65 und 73)

[CK10] M. Connor, P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):599–608, 2010. (Zitiert auf Seite 41)

[Cro13] D. J. Croton. Damn You, Little h! (or, Real-World Applications Of The Hubble Constant Using Observed And Simulated Data). *Publications of the Astronomical Society of Australia*, 30:e052, 2013. (Zitiert auf Seite 21)

[Dec05] P. Decaudin. AntTweakBar Library, 2005. URL `http://anttweakbar.sourceforge.net/doc/`. (Zitiert auf den Seiten 47 und 68)

[DEFW85] M. Davis, G. Efstathiou, C. S. Frenk, S. D. White. The evolution of large-scale structure in a universe dominated by cold dark matter. *The Astrophysical Journal*, 292:371–394, 1985. (Zitiert auf den Seiten 30 und 32)

[DRGI08] K. Dolag, M. Reinecke, C. Gheller, S. Imboden. Splotch: visualizing cosmological simulations. *New Journal of Physics*, 10(12):125006, 2008. (Zitiert auf Seite 48)

[DSA+13] K. S. Dawson, D. J. Schlegel, C. P. Ahn, S. F. Anderson, É. Aubourg, S. Bailey, R. H. Barkhouser, J. E. Bautista, A. Beifiori, A. A. Berlind, et al. The baryon oscillation spectroscopic survey of SDSS-III. *The Astronomical Journal*, 145(1):10, 2013. (Zitiert auf den Seiten 16 und 27)

[DWS+88] M. Deering, S. Winner, B. Schediwy, C. Duffy, N. Hunt. The triangle processor and normal vector shader: a VLSI system for high performance graphics. In *ACM SIGGRAPH Computer Graphics*, Band 22, S. 21–30. ACM, 1988. (Zitiert auf Seite 59)

[Ein65] J. Einasto. Kinematics and dynamics of stellar systems. *Trudy Inst. Astroz. Alma-Ata*, 51:87, 1965. (Zitiert auf Seite 25)

[EWA+11] D. J. Eisenstein, D. H. Weinberg, E. Agol, H. Aihara, C. A. Prieto, S. F. Anderson, J. A. Arns, É. Aubourg, S. Bailey, E. Balbinot, et al. SDSS-III: Massive spectroscopic surveys of the distant universe, the Milky Way, and extra-solar planetary systems. *The Astronomical Journal*, 142(3):72, 2011. (Zitiert auf den Seiten 16 und 27)

[FBB+08] J. A. Frieman, B. Bassett, A. Becker, C. Choi, D. Cinabro, F. DeJongh, D. L. Depoy, B. Dilday, M. Doi, P. M. Garnavich, et al. The sloan digital sky survey-II supernova survey: technical summary. *The Astronomical Journal*, 135(1):338, 2008. (Zitiert auf Seite 16)

[FSW09] R. Fraedrich, J. Schneider, R. Westermann. Exploring the Millennium Run - Scalable Rendering of Large-Scale Cosmological Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1251–1258, 2009. (Zitiert auf Seite 48)

[GM77] R. A. Gingold, J. J. Monaghan. Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977. (Zitiert auf den Seiten 32 und 35)

[Gue11] G. Guennebaud. Eigen Library, 2011. URL http://eigen.tuxfamily.org. (Zitiert auf Seite 45)

[HGTS15] B. Hentschel, B. Geveci, M. Turk, S. Skillman. SciVis-Contest 2015: Visualize the Universe, 2015. URL http://darksky.slac.stanford.edu/scivis2015/. (Zitiert auf den Seiten 29, 49 und 79)

[HPL+09] S. Habib, A. Pope, Z. Lukić, D. Daniel, P. Fasel, N. Desai, K. Heitmann, C.-H. Hsu, L. Ankeny, G. Mark, et al. Hybrid petacomputing meets cosmology: The Roadrunner Universe project. In *Journal of Physics: Conference Series*, Band 180, S. 012019. IOP Publishing, 2009. (Zitiert auf Seite 35)

[Hub29] E. Hubble. A relation between distance and radial velocity among extra-galactic nebulae. *Proceedings of the National Academy of Sciences*, 15(3):168–173, 1929. (Zitiert auf Seite 20)

[KH97] A. Klypin, J. Holtzman. Particle-Mesh code for cosmological simulations. *preprint (arXiv e-prints)*, 1997. (Zitiert auf Seite 34)

[KHA12] R. Kaehler, O. Hahn, T. Abel. A novel approach to visualizing dark matter simulations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2078–2087, 2012. (Zitiert auf Seite 48)

[KK09] S. R. Knollmann, A. Knebe. AHF: Amiga's halo finder. *The Astrophysical Journal Supplement Series*, 182(2):608, 2009. (Zitiert auf Seite 34)

[KKM+11] A. Knebe, S. R. Knollmann, S. I. Muldrew, F. R. Pearce, M. A. Aragon-Calvo, Y. Ascasibar, P. S. Behroozi, D. Ceverino, S. Colombi, J. Diemand, et al. Haloes gone mad: The halo-finder comparison project. *Monthly Notices of the Royal Astronomical Society*, 415(3):2293–2318, 2011. (Zitiert auf den Seiten 33 und 37)

[Kro12] P. Kroupa. The dark matter crisis: falsification of the current standard model of cosmology. *Publications of the Astronomical Society of Australia*, 29(04):395–433, 2012. (Zitiert auf Seite 17)

[KTGP11] A. A. Klypin, S. Trujillo-Gomez, J. Primack. Dark Matter Halos in the Standard Cosmological Model: Results from the Bolshoi Simulation. *The Astrophysical Journal*, 740(102):17ff, 2011. (Zitiert auf den Seiten 12 und 16)

[KY05] P. Kumar, E. A. Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1):1–21, 2005. (Zitiert auf Seite 44)

[LCD06] T. Luft, C. Colditz, O. Deussen. Image enhancement by unsharp masking the depth buffer. In *ACM Transactions on Graphics*. 2006. (Zitiert auf Seite 64)

[Luc77] L. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977. (Zitiert auf den Seiten 32 und 35)

[MA10] D. M. Mount, S. Arya. Approximate Nearest Neighbor Library, 2010. URL https://www.cs.umd.edu/~mount/ANN/. (Zitiert auf Seite 65)

[MGG+99] B. Moore, S. Ghigna, F. Governato, G. Lake, T. Quinn, J. Stadel, P. Tozzi. Dark matter substructure within galactic halos. *The Astrophysical Journal Letters*, 524(1):L19, 1999. (Zitiert auf den Seiten 7, 16 und 26)

[Mil83] M. Milgrom. A modification of the Newtonian dynamics as a possible alternative to the hidden mass hypothesis. *The Astrophysical Journal*, 270:365–370, 1983. (Zitiert auf den Seiten 17 und 24)

[MKDG11] S. More, A. V. Kravtsov, N. Dalal, S. Gottlöber. The overdensity and masses of the friends-of-friends Halos and universality of Halo mass function. *The Astrophysical Journal Supplement Series*, 195(1):4, 2011. (Zitiert auf Seite 33)

[Mor66] G. M. Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company, 1966. (Zitiert auf Seite 41)

[Mos05] N. Moshtagh. Minimum volume enclosing ellipsoid. *Convex Optimization*, 111:112, 2005. (Zitiert auf Seite 44)

[MPSS05] O. Mallo, R. Peikert, C. Sigg, F. Sadlo. Illuminated lines revisited. In *Visualization, 2005. VIS 05. IEEE*, S. 19–26. IEEE, 2005. (Zitiert auf den Seiten 61 und 62)

[MQF06] J. Miller, C. W. Quammen, M. C. Fleenor. Interactive visualization of intercluster galaxy structures in the horologium-reticulum supercluster. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1149–1156, 2006. (Zitiert auf Seite 48)

[MRG12] C. Müller, G. Reina, S. Grottel. TheLib, 2012. URL http://thelib.sourceforge.net/index.php/TheLib. (Zitiert auf Seite 68)

[New87] I. Newton. *Philosophiae Naturalis Principia Mathematica*, Band 1. Halley, E., 1687. (Zitiert auf Seite 24)

[NFW96] J. F. Navarro, C. S. Frenk, S. D. White. The Structure of Cold Dark Matter Halos. *The Astrophysical Journal*, 462:563, 1996. (Zitiert auf den Seiten 16 und 25)

[NGH05] M. C. Neyrinck, N. Y. Gnedin, A. J. Hamilton. VOBOZ: an almost-parameter-free halo-finding algorithm. *Monthly Notices of the Royal Astronomical Society*, 356(4):1222–1232, 2005. (Zitiert auf Seite 35)

[NJB07] P. Navrátil, J. Johnson, V. Bromm. Visualization of cosmological particle-based datasets. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1712–1718, 2007. (Zitiert auf Seite 48)

[OAB+13] J. Onions, Y. Ascasibar, P. Behroozi, J. Casado, P. Elahi, J. Han, A. Knebe, H. Lux, M. E. Merchán, S. I. Muldrew, et al. Subhaloes gone Notts: spin across subhaloes and finders. *Monthly Notices of the Royal Astronomical Society*, 429(3):2739–2747, 2013. (Zitiert auf den Seiten 33 und 37)

[OS03] J. P. Ostriker, P. Steinhardt. New light on dark matter. *Science*, 300(5627):1909–1913, 2003. (Zitiert auf Seite 24)

[Pap07] L. Papantonopoulos, Herausgeber. *The Invisible Universe: Dark Matter and Dark Energy*. Springer, 2007. (Zitiert auf den Seiten 12, 15 und 24)

[PBP02] H. Prautzsch, W. Boehm, M. Paluszny. *Bézier and B-spline techniques*. Springer, 2002. (Zitiert auf Seite 80)

[Pea01] K. Pearson. On lines and planes of closest fit to a system of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559–572, 1901. (Zitiert auf Seite 44)

[Pho75] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975. (Zitiert auf den Seiten 59 und 61)

[PS74] W. H. Press, P. Schechter. Formation of galaxies and clusters of galaxies by self-similar gravitational condensation. *The Astrophysical Journal*, 187:425–438, 1974. (Zitiert auf den Seiten 30 und 31)

[RAC+10] Y. Rasera, J.-M. Alimi, J. Courtin, F. Roy, P.-S. Corasaniti, A. Füzfa, V. Boucher. Introducing the Dark Energy Universe Simulation Series (DEUSS). In *American Institute of Physics Conference Series*, Band 1241, S. 1134–1139. 2010. (Zitiert auf Seite 34)

[Ric05] C. Riccio. OpenGL Mathematics Library (GLM), 2005. URL `http://glm.g-truc.net/0.9.7/index.html`. (Zitiert auf Seite 68)

[Sch06a] P. Schneider. *Einführung in die Extragalaktische Astronomie und Kosmologie*. Springer, 2006. (Zitiert auf den Seiten 7, 11, 12, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 und 27)

[Sch06b] P. Schneider. *Extragalactic Astronomy and Cosmology – An Introduction*. Springer, 2. Auflage, 2006. (Zitiert auf den Seiten 15, 25, 26 und 27)

[SDE95] F. Summers, M. Davis, A. E. Evrard. Galaxy Tracers and Velocity Bias. *The Astrophysical Journal*, 454:1, 1995. (Zitiert auf Seite 33)

[SET+05] V. Springel, A. Evrard, P. Thomas, J. Colberg, J. Peacock, D. Croton, S. Cole, A. Jenkins, S. White, J. Navarro, et al. Simulating the joint evolution of quasars, galaxies and their large-scale distribution. *Nature*, 435(7042):629–636, 2005. (Zitiert auf den Seiten 12, 16 und 28)

[SR10] P. Sutter, P. Ricker. Examining subgrid models of supermassive black holes in cosmological simulation. *The Astrophysical Journal*, 723(2):1308, 2010. (Zitiert auf Seite 34)

[SSL08] T. Szalay, V. Springel, G. Lemson. Gpu-based interactive visualization of billion point cosmological simulations. *CoRR*, 2008. (Zitiert auf Seite 47)

[ST90] T. Saito, T. Takahashi. Comprehensible rendering of 3-D shapes. In *ACM SIGGRAPH Computer Graphics*, Band 24, S. 197–206. ACM, 1990. (Zitiert auf Seite 59)

[Sta01] J. G. Stadel. *Cosmological N-body Simulations and their Analysis*. Dissertation, University of Washington, 2001. (Zitiert auf Seite 35)

[SW14] J. K. Salmon, M. S. Warren. libSDF, 2014. URL https://bitbucket.org/JohnSalmon/sdf. (Zitiert auf den Seiten 30 und 68)

[SWT+14] S. W. Skillman, M. S. Warren, M. J. Turk, R. H. Wechsler, D. E. Holz, P. M. Sutter. Dark Sky Simulations: Early Data Release, 2014. (Zitiert auf den Seiten 12, 16, 21, 22, 28, 29, 49 und 79)

[SXG+14] G. Shan, M. Xie, Y. Gao, X. Chi, et al. Interactive visual exploration of halos in large-scale cosmology simulation. *Journal of Visualization*, 17(3):145–156, 2014. (Zitiert auf Seite 49)

[SZH97] D. Stalling, M. Zöckler, H.-C. Hege. Fast display of illuminated field lines. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):118–128, 1997. (Zitiert auf den Seiten 8 und 61)

[TSH12] J. Takle, D. Silver, K. Heitmann. A case study: Tracking and visualizing the evolution of dark matter halos and groups of satellite halos in cosmology simulations. In *Conference on Visual Analytics Science and Technology (VAST)*, S. 243–244. IEEE, 2012. (Zitiert auf Seite 48)

[TSKH13] J. Takle, D. Silver, E. Kovacs, K. Heitmann. Visualization of multivariate dark matter halos in cosmology simulations. In *Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, S. 131–132. IEEE, 2013. (Zitiert auf Seite 48)

[TSO+11] M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, M. L. Norman. yt: A multi-code analysis toolkit for astrophysical simulation data. *The Astrophysical Journal Supplement Series*, 192(1):9, 2011. (Zitiert auf Seite 48)

[War13] M. S. Warren. 2HOT: An improved parallel hashed oct-tree N-body algorithm for cosmological simulation. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, S. 72. ACM, 2013. (Zitiert auf Seite 28)

[WBS+14] W. Widanagamaachchi, P.-T. Bremer, C. Sewell, L.-T. Lo, J. Ahrens, V. Pascuccik. Data-parallel halo finding with variable linking lengths. In *4th Symposium on Large Data Analysis and Visualization (LDAV)*, S. 27–34. IEEE, 2014. (Zitiert auf den Seiten 29, 35, 38, 41, 44, 65, 72, 76 und 79)

[YAAJ+00] D. G. York, J. Adelman, J. E. Anderson Jr, S. F. Anderson, J. Annis, N. A. Bahcall, J. Bakken, R. Barkhouser, S. Bastian, E. Berman, et al. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120(3):1579, 2000. (Zitiert auf den Seiten 16 und 27)

[ZMS+08] M. Zemp, B. Moore, J. Stadel, C. M. Carollo, P. Madau. Multimass spherical structure models for N-body simulations. *Monthly Notices of the Royal Astronomical Society*, 386(3):1543–1556, 2008. (Zitiert auf Seite 34)

Alle URLs wurden zuletzt am 14. 10. 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.

---

Stuttgart, 15.10.2015