

Arbeitsname Bachelorarbeit

VISUALISIERUNG VON STRUKTURVERÄNDERUNGEN IN MOLEKULARDYNA MIKDATEN

Richard Hähne

Geboren am: 10. Februar 1982 in Dresden

Matrikelnummer: 2873574

Immatrikulationsjahr: 2011

BACHELOR WIP

zur Erlangung des akademischen Grades

BACHELOR OF SCIENCE (B.SC.)

Betreuer

Sebastian Grottel

Ludwig Schmutzler

Betreuer Hochschullehrer

Prof. Dr. Stefan Gumhold

Eingereicht am: 31. Dezember 2945

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelor WIP mit dem Titel *Visualisierung von Strukturveränderungen in Molekulardynamikdaten* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in der Arbeit angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Es waren keine weiteren Personen an der geistigen Herstellung der vorliegenden Arbeit beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Dresden, 31. Dezember 2945

Richard Hähne

INHALTSVERZEICHNIS

1. Aufgabenstellung	5
2. Einleitung	6
3. Verwandte Arbeiten	8
3.1. Skelettextraktion	8
3.2. Zusammenhangskomponenten und Konturbäume	9
3.3. Visualisierung großer Datensätze	11
4. Überblick	12
5. Erkennung der Cluster und Strukturereignisse	13
5.1. Datengrundlage	13
5.2. Speicherung der Partikel- und Clusterdaten	13
5.3. Bestimmung der Cluster ohne Nachbarschaftskenntnisse	14
5.4. Bestimmung der nächsten Nachbarn	14
5.4.1. kD-Baum	14
5.5. Fast-Depth-Algorithmus zur Bestimmung der Cluster	15
5.6. Reduzierung der kleinen Cluster	15
5.7. Bestimmung der Clusterzusammenhänge	16
5.7.1. Nachbarschaftsgraph	16
5.7.2. Frameübergreifender Vergleich der Partikelzugehörigkeit	16
5.8. Erkennung von Ereignissen	17
5.9. Speicherung der Daten	18
5.10. Programmaufbau für die Berechnung	18
6. Visualisierung	19
6.1. Grundlagen	19
6.1.1. Glyphdesign	19
6.1.2. Gestaltgesetze	19
6.1.3. Präattentive Wirkung visueller Variablen	20
6.1.4. Formgebung	20
6.2. Visualisierungsentwurf	21
6.2.1. Taxonomie der Visualisierung	21
6.2.2. Parameter-Attributszuweisung	24
6.2.3. Mockups	25
6.2.4. Datensatz	25
6.2.5. Unterscheidbarkeit von Polyedern	27

6.3.	Entwurfsauswertung	28
6.3.1.	Glyphen	28
6.4.	Vorgehen	28
6.4.1.	Gestaltung der Glyphen	28
6.4.2.	Renderer und Shader	28
6.4.3.	Visualisierung der Clusterberechnung	28
6.4.4.	Weiterleitung der Daten	28
6.4.5.	Programmaufbau für die Visualisierung	29
7.	Evaluation	30
7.1.	Ermittlung der Ereignisse	30
7.2.	Einfluss der Parameter der Berechnungsschritte	30
7.2.1.	Einfluss der Nachbarschaftssuche	30
7.3.	Auswertung der Visualisierung	31
7.3.1.	Glyphdesign	31
7.3.2.	Glyphdarstellung im Partikelraum	31
7.4.	Ressourcenverbrauch	32
7.4.1.	Optimierung durch Parallelisierung	32
8.	Zusammenfassung	34
9.	Ausblick	35
A.	Anhang	37
A.1.	Berechnungen Mockups	37
A.2.	Werkzeuge und Bibliotheken	39
Literatur		41

1. AUFGABENSTELLUNG

Die visuelle Analyse spielt für komplexe, partikelbasierte Daten, wie sie beispielsweise in der Molekulardynamik entstehen, eine immer wichtiger werdende Rolle. Da die Datensätze immer größer werden, gewinnen abstrahierte Visualisierungen zur Vermittlung eines Überblicks zunehmend an Bedeutung. Kritisch ist hierbei vor allem die Darstellung zeitlicher Entwicklungen. Üblicherweise werden diese als Animation oder in rein abstrakter Form dargestellt. Diese Visualisierungen bieten nur einen schlechten zeitlichen Überblick oder stellen kaum Bezug zum ursprünglich simulierten Ortsraum dar.

Ziel dieser Arbeit ist es daher an einem konkreten Beispiel eine Visualisierung zu entwickeln, die im geometrischen Kontext der Originaldaten die zeitliche Entwicklung der Struktur des Datensatzes darstellt. Der zu untersuchende Datensatz (`exp2mill.mmpld`, 2 Mio. Partikel, 150 Zeitschritte) zeigt einen Flüssigkeitsfilm im Vakuum, der aufgrund seines übergroßen Innendrucks explodiert. Hierbei bilden sich Molekülcluster, -Tröpfchen und -Filamentstrukturen, welche sich über die Zeit hinweg verändern (aufspalten und verschmelzen).

Zunächst soll der Datensatz in mehreren Stufen analysiert werden. Ein erster Schritt bestimmt die Interface-Grenzen zwischen der flüssigen Phase und dem Vakuum, bzw. in späteren Zeitschritten der Gasphase. In einem zweiten Schritt wird auf dieser Basis eine vorzeichenbehaftete Distanzfunktion aufgestellt, die für alle Partikel die Entfernung zum Rand bestimmt. Für diese beiden Arbeitsschritte kann auf bestehenden Quellcode zurückgegriffen werden. Anschließend wird, auf Basis eines vom Benutzer vorgegebenen Schwellwerts die Struktur der Daten, in Form von Clustern/Knoten und Verbindungen, vergleichbar zu einem Contour-Tree extrahiert. Besondere zeitliche Ereignisse sind gekennzeichnet durch Änderungen in dieser Struktur (Split, Merge, Birth, Death). Diese Ereignisse werden, beispielsweise über Glyphen, in einer zeitunabhängigen Ansicht in Ortsraumkoordinaten des Originaldatensatzes visualisiert.

Die entstandene Lösung soll in zweierlei Hinsicht evaluiert werden. Zum einen betrifft dies die Berechnungsgeschwindigkeit. Hierbei ist jedoch eine umfassende Optimierung nicht so wichtig wie die grundsätzliche Komplexität und Skalierbarkeit der eingesetzten Algorithmen. Als zweites soll die entstandene Visualisierung in einer informellen Diskussion mit Experten (auch per E-Mail) bewertet werden.

2. EINLEITUNG

Atome und Moleküle bestimmten Orten in festen, flüssigen, gasförmigen oder plasmatischen Phasen zuzuordnen, ist eine wesentliche Grundlage zahlreicher Anwendungen der Materialwissenschaft, der Physik oder der Chemie. Durch Diffusion oder Konvektion kann sich die Zugehörigkeit dieser Teilchen zu bestimmten Agglomerationen über die Zeit ändern. Agglomerationen können Molekülcluster, -tröpfchen und -filamentstrukturen sein. Da in den Agglomerationen die Teilchen eine räumliche Nähe zueinander aufweisen und auch durch chemische Bindungen zusammenhängen können, ist eine Betrachtung als Zusammenhangskomponente möglich.

Molekulardynamische Berechnungen ermöglichen die Vorhersage von Materialverhalten. Sie werden aufgrund des hohen Berechnungsaufwandes verursacht durch die komplexen Struktur-Eigenschafts-Beziehungen von Atomen und Molekülen meist im nanoskopischen Bereich mit begrenzter Anzahl von betrachteten Teilchen durchgeführt. Dieser Arbeit liegt ein Datensatz mit zwei Millionen Partikeln in einer Flüssigkeitsphase zugrunde, die sich im Zentrum eines evakuiertem Quaders befindet. Durch den hohen Druckunterschied zwischen der Flüssigkeit und dem des sie umgebenden Vakuums expandiert sie explosionsartig. Dabei entstehen Filamentstrukturen und Tröpfchen. Auch gehen Partikel von der Flüssigkeits- in die Gasphase über.

Die Partikel liegen als diskrete Topologie, das heißt mit voneinander disjunkten Positionen vor. Bis auf die räumliche Nähe weisen sie keine Zusammenhänge auf. Jeder Partikel besitzt einen *Tiefenwert*, der unter anderem markiert, ob er sich in der flüssigen oder gasförmigen Phase befindet.

Um die Bewegung der Partikel verfolgen zu können, werden sie in Gruppen eingeteilt, ähnlich der Zusammenhangskomponenten. Da jedoch auch Strukturereignisse innerhalb dieser Komponenten erkannt werden sollen, erfolgt eine wesentlich feinere Zuordnung in sogenannte Cluster. Diese Cluster können eine gesamte Zusammenhangskomponente beinhalten oder auch nur ein Teil deren sein.

Daher liegt ein Schwerpunkt der Arbeit auf der Clustererzeugung, wozu der *Fast-Depth-Algorithmus* entwickelt wurde. Er ermöglicht eine reproduzierbare Einteilung der Partikel innerhalb von Zusammenhangskomponenten, erzeugt über mehrere Zeitschritte jedoch ein Rauschen, da Änderungen in der Partikelstruktur eine andere Clusterbildung zur Folge haben.

Der zweite und der dritte Schwerpunkt und auch das Ziel dieser Arbeit sind die Erkennung der Ereignisse sowie deren Visualisierung im Raum der Partikel. Die Erkennung erfolgt auf Basis eines Vergleichs der gebildeten Cluster über zwei Zeitschritte und nutzt ein heuristisches Vorgehen. Die Analyse der daraus entstehenden Daten ermöglicht die Auswahl bestimmter Parameter, die zur Steuerung der Anzahl und der Qualität der erkannten Ereignisse dienen. Da die Partikel als dreidimensionale Kugeln dargestellt werden, sollte die Ereignisdarstellung ebenfalls mit dreidimensionalen Glyphen durchgeführt werden, um die Konsistenz in der Anzeige zu erhalten. Die Erstellung eines Prototypen hat jedoch gezeigt, dass Formen abseits von

Kugeln, wie die im Prototyp verwendeten Polygone, eine Erkennung erschweren. So werden im Ergebnis 3D Sprites (im Folgenden als *Billboards* bezeichnet) verwendet, die sich stark von den Partikeln abheben und damit gut sichtbar sind. Weiterhin wird eine Taxonomie für eine Verknüpfung bestimmter visueller Attribute mit den Ereignisparametern vorgeschlagen, die zur Gestaltung und Positionierung der Glyphen dient. Ein im Rahmen dieser Arbeit erstellter Renderer ermöglicht eine Filterung und Skalierung der Glyphen, so dass der Anwender die Darstellung seinen Bedürfnissen nach anpassen kann.

Eine Herausforderung stellt der Ansatz für die Clusterbildung dar. Dies kann über den *Tiefenwert* der Partikel kombiniert mit Abstandsmessungen geschehen oder über die Bestimmung ihrer Nachbarschaftsbeziehungen. Der Aufwand für beide Berechnungen skaliert mit $\mathcal{O}(n^2)$ mit n als Partikelanzahl. Unklar ist zu Beginn, wie die Aufteilung der Zusammenhangskomponenten in einzelne Cluster erfolgen kann. Weiterhin musste geprüft werden, ob Ereignisse durch die Untersuchung einzelner Partikel, durch topologische Veränderungen der durch die Partikel beschriebene Grenze (keine Grenzfläche, da diskrete und keine kontinuierlichen Daten vorliegen), durch die Nachbarschaften von Clustern oder über Mengenvergleiche erkennbar sind.

Die gesamte Berechnung soll testfähig sein. Das bedeutet, dass die Berechnungen der Cluster für einen Zeitpunkt des zwei Millionen Partikel großen Datensatzes innerhalb von Minuten und nicht von Stunden abgeschlossen sein muss. Erkenntnisse aus der Skelettextraktion, der Behandlung von Laserscandaten sowie die Nutzung von Konturbäumen helfen bei der Entwicklung.

3. VERWANDTE ARBEITEN

Für die Zuordnung von Teilchen zu Agglomerationen bzw. Zusammenhangskomponenten ist das Wissen über deren räumliche Begrenzung Voraussetzung, das Wissen über ihre Topologie. Dazu ist es notwendig, den Oberflächenverlauf oder Positionsinformationen über das Volumen zu kennen. Die Volumen- und Oberflächenbestimmung von großen Datensätzen ist bei Skelettextraktionen untersucht, die zum Beispiel die durch Laserscanverfahren ermittelte Daten auswerten. Weiterhin wurden Datenstrukturen entwickelt, um räumliche Datensätze beliebiger Dimensionen anhand ihrer Funktionswerte in Zusammenhangskomponenten einzuteilen. Sie werden als Konturbäume bezeichnet.

Anschließend wird auf die Visualisierung großer Datenmengen und Zeitverläufe eingegangen.

3.1. SKELETTEXTRAKTION

Skelettextraktionsverfahren nutzen Oberflächen oder Volumen dazu, eine (volumenlose) 1D Skelettkurve zu erstellen, die die Oberfläche oder das Volumen eines 3D Objektes durch ihren Bogenverlauf abbildet und zusätzliche Eigenschaften als Metadaten enthält wie z.B. die Volumendicke [Au+08]. Palágyi vergleicht die Kurve mit einem brennenden Gegenstand, der von allen Seiten gleichmäßig niederbrennt. Die Stellen, an denen sich das Feuer auslöscht, bilden das Skelett des Gegenstandes [Pal08]. Blum nennt sie Medialachsenskelette [Blu67]. Daneben gibt es, abhängig vom Skelettermittlungsverfahren weitere Skelettypen wie z.B. Kurvenskelette [DS06] [CSM07] oder die durch down-sampling davon abgeleiteten Knochenskelette, die vor allem in der Charakteranimation [WPP07] und Netzdeformation [Web+07] eingesetzt werden. Um Skelette aus Oberflächendaten zu berechnen, kommen zum Beispiel Voronoidiagramme zum Einsatz, mit denen der Raum in Regionen aufgeteilt wird und die mediale Oberflächen aus den inneren Kanten und Flächen des Diagramms extrahieren [DS06]. Eine Alternative ist die Nutzung des Reebgraphen, dessen Knoten die kritischen Punkte einer auf die Oberfläche angewandten reellwertigen Funktion sind, welche Topologieänderungen der Oberfläche entsprechen [Pas+07]. Der Graph liegt etwa der Laplaceglättung zugrunde, bei der mehrere Eckpunkte eines Netzes zu einem Vertex zusammengefasst werden und so das Objekt verkleinert wird [Au+08]. Weitere Methoden sind das Ausdünnen beim Vorliegen von binären Daten (Daten mit nur zwei Intensitätszuständen, wie z.B. schwarz-weiß Bilder) durch schrittweises Entfernen der äußeren Datenpunkte [Pal08], die Berechnung mittels mittleren Krümmungsflusses, wo die Bewegungsgeschwindigkeit der Normalen eines Punktes auf der Oberfläche vom Krümmungsradius der Oberfläche an diesem Punkt abhängt [Tag+12] oder Netzzerlegungen unter Beachtung der geodätischen Abstände sowie von konvexen Verläufen und Verlinken der Komponenten [KT03]. Die Berechnung von Skeletten aus Volumendaten erfolgt ebenfalls durch Ausdünnung, etwa von Voxeln durch schrittweises Entfernen der äußeren Voxel [MWL02], durch vorherige

Verkleinerung der Volumenmodelle [WL08] oder durch Distanzfeldmethoden, bei denen ein Distanzfeld für jeden inneren Punkt aufgebaut wird, das die Entfernung zum Rand des Objektes enthält und daraus bestimmte Voxel ausgewählt werden, die bei einer Verbindung untereinander eine mediale Oberfläche erzeugen [HF05]. Andere Feldmethoden nutzen einen Potentialwert für innere Punkte durch Einbeziehung mehrerer Randvoxel und sind somit weniger anfällig für Rauschen, jedoch aufwändiger in der Berechnung [Cor+05]. Diese Liste ist nur ein Auszug von üblichen Methoden. Für Kurvenskelette gibt [CSM07] eine umfassende Methodenübersicht.

Den Methoden ist gemeinsam, dass sie die vorhandenen Oberflächen- oder Volumendaten diskretisieren müssen, sollten sie als kontinuierliche Funktionen vorliegen, und die Daten durch eine Bewegung von außen nach innen reduzieren, wodurch eine ursprünglich große Datenmenge durch wenige Knoten repräsentiert werden kann und somit ein Einsatz vor allem in Echtzeitanwendungen findet. Die Verfahren haben dabei mit Problemen wie fehlender Wiedergabe von Oberflächenbeschaffenheiten, insbesondere bei Ausdünnungsverfahren, mit Rauschen oder mit numerischen Instabilitäten bei schlechter Datendiskretisierung zu kämpfen und benötigen teilweise umfangreiche Pre- oder Postprocessingverfahren etwa zum Entfernen von Artefakten oder zum nachträglichen Zentrieren des Skeletts [CSM07].

Für die Bestimmung der Zugehörigkeit von Partikeln des dieser Arbeit vorliegenden Partikeldatensatzes zu Agglomerationen können diese in der Skeletteextraktion verwendeten Methoden nicht übernommen werden. Zum Einen basieren viele auf Netzdaten, bei denen die Punkte des Datensatzes im Gegensatz zum Datensatz dieser Arbeit miteinander in Beziehung stehen. Zum Anderen sind auch solche Verfahren ungeeignet, die als Datengrundlage vollständige Punktmengen [WL08] [Sha+07] oder bei Laserscanverfahren unvollständig ermittelte Punktmengen [TZC09] verwenden. Denn die Daten werden derart reduziert, dass die für die Ereigniserkennung notwendige Wanderung von Partikeln nicht verfolgt werden kann. Hingegen ist die Idee des Reebgraphen [Pas+07], die eine auf kritische Punkte beschränkte Struktur darstellt, ein guter Ansatz für eine mögliche Einteilung der Strukturen in Cluster.

3.2. ZUSAMMENHANGSKOMPONENTEN UND KONTURBÄUME

Um die Bewegung der Partikel verfolgen zu können, ist die Bildung von Zusammenhangskomponenten sinnvoll. Dabei werden Partikel zu Gruppen zusammengefasst, so dass die Wanderung von Partikeln im Verlauf der Zeit von einer zur nächsten Gruppe sichtbar wird. Diese Gruppen können mit den eingangs erwähnten Agglomerationen gleichgesetzt werden.

Zusammenhangskomponenten können bei Daten, die in einem Gitter angeordnet sind, durch Einteilung der Daten in Segmente parallel verarbeitet und damit schneller gefunden werden [Tre+13]. Die Erzeugung von Zusammenhangskomponenten aus Punktdaten ist über die Herstellung von Nachbarschaftsinformationen dieser Punkte möglich. Klasing et al. nutzen dabei einen definierten Maximalabstand, in dem Punkte noch als Nachbarn gelten und beschreiben eine effektive Implementation ohne die Nutzung von Baumstrukturen [KWB08]. Dies kann hier aufgrund der Erfordernis einer vollständigen Datenstruktur nicht angewandt werden. Jedoch wird in dieser Arbeit die radienbasierte Nachbarschaftssuche als Grundlage für die Clusterbildung verwendet (siehe [Abschnitt 5.4](#)).

Konturbäume bieten die Möglichkeit, Datenelemente anhand ihres Funktionswertes zu gruppieren und für verschiedene Funktionswerte Zusammenhangskomponenten erkennen zu können [Kre+97] [BPS97]. Diese Flexibilität ist ihr Vorteil. Weiterhin kann mit ihnen festgestellt werden, wann die Zusammenhangskomponenten entstehen, wann sie verschwinden, sich verbinden und sterben, analog zu den mit dieser Arbeit zu erkennenden Strukturereignissen. Sie sind für beliebige Dimensionen geeignet, setzen jedoch wie die Skeletteextraktionsverfahren Daten in einer Netz- oder Gitterstruktur voraus, denen ein Skalarfeld zugeordnet ist. Dieses weist jedem Punkt einen Funktionswert zu.

Der *Konturbaum* ist ein Graph, der die Konturen von Niveaumengen im Verbinden und Ent-

stehen oder Trennen und Sterben verfolgt. Eine Niveaumenge ist die Menge aller Punkte einer Funktion bzw. eines Skalarfeldes, denen ein gleicher Wert zugeordnet ist. Dieser Wert wird nach [CSP10, S. 1] als *Isowert* bezeichnet. Im Zweidimensionalen werden damit *Isolinien* erzeugt, im Dreidimensionalen sind es *Isooberflächen*. Sie finden zum Beispiel bei der Visualisierung von Höhendaten auf topografischen [Hur10] [All+15], von Temperatur-, Wind- und Luftdruckdaten auf meteorologischen Landschaftskarten [Hop96] oder von Gewebestrukturen auf Computertomographieaufnahmen [Tan+14] Verwendung. Der dieser Arbeit zugrundeliegende Datensatz befindet sich in einem dreidimensionalen Raum, so dass sich auf *Isooberflächen* beschränkt wird. Diese Oberflächen schließen ein Volumen ein. Die Partikel innerhalb des Volumens bilden eine oder mehrere Zusammenhangskomponenten, die als *Konturen* bezeichnet werden [CSA01a, S. 2].

KRITISCHE FUNKTIONSWERTE ZUR ERKENNUNG VON KONTURVERÄNDERUNGEN

Um die Entwicklung dieser *Konturen* zu verfolgen, ist die Erkennung sogenannter kritischer Funktionswerte notwendig, die bei einem derartigen *Isowert* auftreten, bei dem sich die Topologie oder Anzahl der *Konturen* ändert. In der Morsetheorie ist ein Punkt dann ein kritischer Punkt, wenn kein Gradient vorhanden ist [Mil63] [SKK91]. Damit setzen die Theoreme voraus, dass sich an diesen Punkten die Topologie der Niveaumenge ändert. Bei einer abschnittsweisen linearen Funktion heißt das, wenn der *Isowert* durch den Funktionswert verläuft und sich dabei die Topologie der Niveaumenge ändert, dann ist es ein kritischer Funktionswert. Alle anderen sind reguläre Funktionswerte [CSP10] [Chi+05].

ABTASTALGORITHMUS

Konturbäume enthalten Funktionswerte in Form von Minima, Maxima und Sattelpunkten. Nicht jeder dieser Werte ist ein kritischer Funktionswert. Der Konturbaum wird mit einem Algorithmus in vier Schritten gebildet.

Da die Daten in einem Gitter oder Netz vorliegen und ihnen ein Skalarfeld zugeordnet ist, werden sie als erstes nach ihrem des im Skalarfeld enthaltenem Funktionswert sortiert. Anschließend werden in jeweils einem Schritt zwei Bäume erstellt, indem die Punkte einmal vom kleinsten zum größten Funktionswert und ein zweites Mal vom größten zum kleinsten Wert durchlaufen werden. Die Knoten des Baumes bilden *Isowerte*, bei denen Niveaumengen entstehen, teilen, verbinden oder verschwinden. Geschlechtsänderungen werden nicht beachtet. Die Kanten bilden äquivalente *Konturen*. Dabei entstehen ein Verbindungs- und ein Teilungsbau [CSA01b]. Schließlich werden im vierten Schritt beide Bäume zusammengefügt, wobei alle Knoten mit dem Grad zwei entfernt werden. Dieser resultierende Baum wird Konturbaum genannt und die Methodik als Abtastalgorithmus bezeichnet [CSA01a] [Chi+05, S. 176].

Da die Konturbäume nur kritische Funktionswerte beinhalten, sind sie eine spezielle Version der auch bei Skeletteextraktionsverfahren verwendeten Reebgraphen [Pas+07] für reellwertige Skalarfelder [CSP10, S. 44]. Diese speziellen Reebgraphen finden auch Anwendung bei Varianten des Konturbaums. Beispielsweise bietet das Multi-Resolution-Verfahren umfangreichere Filtermöglichkeiten durch Erweiterung des Konturbaumes, ist jedoch auch langsamer [PCS04]. Hingegen ist ein schnelleres und speichersparenderes Verfahren der Monotonpfadalgorithmus, der nur eine für die aktuelle Betrachtung interessante Teilmenge der kritischen Werte im Baum speichert [Chi+05].

Dem Abtastalgorithmus des Konturbaums, der davon abgeleiteten Varianten sowie einiger Skeletteextraktionsverfahren ist gemeinsam, dass sie kritische Werte des über den Daten liegenden Skalarfeldes nutzen, um räumliche Strukturen zu analysieren und zu speichern. Leider können diese Methoden auf den dieser Arbeit zugrundeliegenden Datensatz nicht angewandt werden. Zwar enthält der Partikeldatensatz ebenfalls ein Skalarfeld, das jedem Datenpunkt bzw. Partikel einen Funktionswert zuweist, jedoch sind keine Zusammenhänge zwischen den Partikeln

bekannt. Dagegen sind die vorgestellten Methoden für in Gittern oder Netzen vorliegende Daten ausgelegt bzw. sie werden aus kontinuierlichen Flächen in beispielsweise simpliziale Netze umgewandelt. Carr fasst dies in einer Arbeit zusammen:

[The contour tree] has been used for isosurface extraction, for abstract representation of the field, to index individual contours and their geometric properties, to guide simplification of input meshes, and to compare scalar fields. Although recent algorithms for computing the contour tree are efficient, they were originally defined only for simplicial meshes, then extended to trilinear meshes and digital images. [CS09, S. 1]

Dennoch kann ein Teil des Abtastalgorithmus als Ideengeber verwendet werden. Die Bildung des Verbindungsbaumes lässt sich auf das Ablaufen der Partikel bei der Clustersuche übertragen, indem der ihnen zugewiesene Funktionswert genutzt wird und indem zuvor Nachbarschaftsverhältnisse gebildet werden (siehe [Abschnitt 5.5](#)).

TEMPORALE BETRACHTUNGEN BEI KONTURBÄUMEN

Der vorliegende Partikeldatensatz enthält einen Zeitverlauf über 150 Zeitschritte, der zur Ermittlung der Strukturereignisse verwendet wird. Für Konturbäume gibt es eine Beschreibung von Laney et al über ein temporales Verfolgen von räumlichen, in einer regelmäßigen Gitterstruktur angeordneten Daten mit einem geometrischen Ansatz, der die kritischen Punkte von aufeinanderfolgenden *Isooberflächen* miteinander vergleicht [Lan+06, S. 1056].

Sohn und Bajaj hingegen berechnen Ähnlichkeiten von Konturbäumen bei aufeinanderfolgenden Zeitschritten durch einen Volumenvergleich [SW97]. Dabei wird das von *Isooberflächen* eingeschlossene Volumen von aufeinanderfolgenden Zeitschritten verglichen, wodurch für einen festen *Isowert* Änderungen der zugehörigen *Isooberfläche* verfolgt werden können, die in einem sogenannten topologischen Änderungsgraph gespeichert werden [SB06]. Diese Idee kann auf den Vergleich der Cluster und die damit verbundene Ereigniserkennung angewandt werden (siehe [Abschnitt 5.7](#)).

Die Arbeiten von Edelsbrunner et al [Ede+04] und Szymczak [Szy05] beinhalten topologische Analysen, um Zeitdaten mit Hilfe der Evolution von zeitveränderlichen Reebgraphen bzw. Konturbäumen zu analysieren. Sie nutzen ebenfalls kontinuierliche Daten bzw. ein zeitabhängiges Skalarfeld über einem Gitter, so dass dieselben Einschränkungen für die Übertragung auf den vorliegenden Partikeldatensatz wie bei den bereits erwähnten Arbeiten über die Skelettextraktion und Konturbäume gelten.

3.3. VISUALISIERUNG GROSSER DATENSÄTZE

Molekularvisualisierungen

Zeitgraphen

Juxtaposition: Hier verwendet

GLYPHDESIGN

siehe Quellen oder auch Grundlagen aus Kapitel zwei

4. ÜBERBLICK

Diese Arbeit stellt eine auf den Ideen der Skelettextraktion und Konturbäumen aufbauende Möglichkeit vor, um aus Punktdaten Strukturen zu erkennen. Diese Strukturen werden genutzt, um daraus Strukturereignisse abzuleiten.

Weiterhin werden

Die Schwerpunkte werden der Reihenfolge nach behandelt, wobei die Erkennung der Cluster- und Strukturereignisse zusammengefasst werden. Anschließend wird die Visualisierung der Ereignisse betrachtet.

Erklärung der Abschnitte, Module, Visualisierung.

Dateigröße von Datensätzen, Echtzeitdarstellung

5. ERKENNUNG DER CLUSTER UND STRUKTUREREIGNISSE

5.1. DATENGRUNDLAGE

Es liegen diskrete Eingangsdaten mit Partikeln mit definierten und disjunkten Positionen vor. Bis auf die räumliche Nähe weisen sie keine Zusammenhänge auf.

Weiterhin existiert ein Funktionswert für jeden Partikel in Form einer vorzeichenbehafteten Entfernungsgabe. Dieser wurde mit Hilfe der Grenzen zwischen der flüssigen Phase und dem Vakuum bzw. der Gasphase bestimmt. Anschließend wird auf dieser Basis eine vorzeichenbehaftete Distanzfunktion aufgestellt, die für alle Partikel die Entfernung zum Rand bestimmt. Somit gibt der Funktionswert die Distanz zum nächsten Partikel an, welcher auf dem Rand eines Clusters liegt. Im Folgenden wird diese Größe als *Tiefenwert* eines Partikels bezeichnet.

Partikel innerhalb von Clustern besitzen einen positiven *Tiefenwert*, Partikel auf dem Rand weisen den Abstand null auf, Partikel in der Gasphase, d.h. außerhalb von Clustern, haben eine negative *Tiefe*.

5.2. SPEICHERUNG DER PARTIKEL- UND CLUSTERDATEN

Die Partikeldaten werden als dichtgepacktes Structure gespeichert. So werden nur Datentypen mit vier und acht Byte verwendet, so dass der Compiler die Struktur wegen Ausrichtungsbeschränkungen im Speicher nicht auffüllt, wie es etwa bei ein Byte großen Datentypen passieren kann [Ray94].

Um in der folgenden Nachbarschaftsbestimmung die Suche zu beschleunigen, wird die Liste nicht neu sortiert. So kann ein Partikel mit einem Pointer auf das Listenelement gefunden werden. Dies ist sehr viel schneller im Gegensatz zur langwierigen Durchsuchung jedes Partikelements durch Vergleich der jeweiligen Partikel ID, was je nach Partikelposition bis zu einer halben Sekunde pro Partikel dauert.

Auch werden die Nachbarn pro Partikel nicht als Objekte, sondern lediglich als Referenz gespeichert. Die Speicherung als Objekt ist eine enorme Speicherbelastung und erfordert auf dem Entwicklungssystem ständiges Auslagern auf die Festplatte. Der Nachteil dabei ist, dass keine Sortierung der Liste erfolgen darf, da sonst die Referenzen ungültig werden.

Die Verlinkung der Partikeln zu Cluster kann nicht über Referenzen verfolgen, da nicht sichergestellt werden kann, dass die Clusterliste im Speicher verschoben wird aufgrund der dynamischen und nicht vorhersagbaren Größe. So zeigen Tests, dass sich während der Clustererstellung die Position der Cluster im Speicher verändert, was dadurch bedingt ist, dass die gesamte Liste verschoben wird. Das kann man daran erkennen, dass die zurückgegebenen int einen Wert von

-572662307, was ein Hinweis auf memory allocator putting a value into the deallocated memory ist*. Stattdessen: - ausreichend Speicherplatz für den vector reservieren (schwer zu bestimmen)
- id's des vectors statt pointer nutzen (- clusterList darf immer noch nicht umsortiert werden,
+ clusterList darf im Speicher reallokiert werden) - id's im struct Cluster nutzen (+ Sortierung erlaubt, - im Vergleich Zugriff saulangsam)

*<http://stackoverflow.com/a/11370042/4566599>

5.3. BESTIMMUNG DER CLUSTER OHNE NACHBARSCHAFTSKENNTNISSE

Der Abstand eines Partikels zum Rand seines Clusters wird mit einem vorzeichenbehafteten *Tiefenwert* gekennzeichnet. Innerhalb von Clustern weisen Partikel eine positive *Tiefe* Tiefenwert auf. Partikel auf dem Rand besitzen den Abstand null, Partikel in der Gasphase, d.h. außerhalb von Clustern, haben einen negativen *Tiefenwert*.

Partikel eines Clusters werden über eine radialbasierte Nachbarschaftsbeziehung bestimmt, wobei vom Partikel mit der größten Tiefe, dem sogenannten *Saatpartikel* begonnen wird.

Um diesen Prozess zu beschleunigen, werden beim Auslesen der MMPLD Daten alle Partikel nach ihrem *Tiefenwert* in einer Liste sortiert. Eine speichersplatzsparendere Methode, die auf das Kopieren aller Partikel im Speicher verzichtet, wäre das Nutzen von Zeigern auf den vorhandenen Datensatz, allerdings wäre zum Einen eine Änderung der Daten aufwändig (Lockingmechanismen) und der nächste *Tiefenwert* müsste bei jeder Partikelbehandlung aus dem ursprünglichen MPDC erneut herausgesucht werden.

Ausgehend vom *Saatpartikel* werden die Partikel in der Liste abgelaufen und in eine neue Liste gespeichert. Sollte der nächste Partikel, der den gleichen oder einen etwas geringeren *Tiefenwert* aufweist, in einem bestimmten Radius zum vorhergehenden Partikel liegen, so wird dieser in dieselbe Liste geschrieben, ansonsten in eine andere. Der Radius, mit dem geprüft wird, hängt von der Differenz des *Tiefenwertes* zwischen betrachteten Partikel und Minimalpartikel der aktuellen Liste ab. Sollte der Partikel außerhalb dieses Radius' liegen, wird eine neue Liste erstellt.

Eine Ausnahme bilden Partikel mit negativem *Tiefenwert*. Sie werden in eine separate Liste geschrieben und es erfolgt keine Positionsprüfung.

Jede Liste enthält einen Offset. Dort befindet sich der Listenidentifikator. Alternativ eine zusätzliche Liste, die die anderen Listen enthält?

Dieser Algorithmus ist wegen der Entfernungsbestimmung über den Radius sehr langsam. Daher wird diese Methode nicht weiter verfolgt und stattdessen werden andere Methoden zur Nachbarschaftsbestimmung untersucht und ausgehend von diesen eine Clusterbestimmung vorgenommen.

5.4. BESTIMMUNG DER NÄCHSTEN NACHBARN

Beschleunigungsvarianten: - eine Liste (wie ohne Beschleunigung): Einschränkung der Entfernungsbestimmung bei den Partikeln auf solche mit ähnlichen Tiefenwerten $O(n^2 - k|k = n_{SignedDistance} > interval|)$ - mehrere Listen: Einteilung des Raumes in ein Gitter, nur die Nachbarzellen (jede Zelle entspricht einer Liste) durchsuchen: voraussichtlich wesentlich weniger Partikel als mit Entfernungsbestimmung - k-d-Baum, da statt $O(n^2)$ nur noch $O(n \log n)$. Bibliotheken: ANN, FLANN (<http://www.cs.ubc.ca/research/flann/>): evtl auch so schnell wie grid

5.4.1. KD-BAUM

Speicherplatzreservierung der particleListe mit Einbeziehung der maximalen Anzahl möglicher Nachbarn.

Erstellung des kD-Baums.

Periodische Randbedingungen werden beachtet, können aber durch den Nutzer für Datensätze ohne solche Randbedingungen deaktiviert werden.

Suchparameter: Radius, maxNeighbours: Setzt die Größe des Containers für die IDs der festgestellten Nachbarn.

Der Suchradius ist durch den Nutzer einstellbar, da mit der Reichweite der Nachbarschaftssuche die Anzahl der Cluster beeinflusst werden kann, siehe [Unterabschnitt 7.2.1](#).

Der Wert für maxNeighbours wird abhängig von Radius automatisch gesetzt. Die passenden Werte wurden experimentell bestimmt und erfüllen die beiden Voraussetzungen, dass alle im durch den Radius definierten Volumen befindlichen Artikel in die Nachbarschaftsliste aufgenommen werden können und andererseits möglichst wenig zusätzlicher, unnötiger Speicherplatz für den bei der Suche verwendeten Container für das Aufnehmen der Nachbarn reserviert wird.

/// Minimal maxNeighbours for radius so no particle in radius gets excluded (determined by experiments): /// radiusModifier, maxNeighbours /// 4, 35 /// 5, 60 /// 6, 100 /// 7, 155 /// 10, 425 /// 20, 3270

5.5. FAST-DEPTH-ALGORITHMUS ZUR BESTIMMUNG DER CLUSTER

Da auch Verschmelzungen und Teilungen innerhalb von Zusammenhangskomponenten erkannt werden sollen, genügt es nicht, wenn die Clusterbildung alle benachbarten Partikel einer Gruppe zuordnet. Vielmehr sollen auch Zusammenhangskomponenten selbst unterteilt werden. Dabei wird die Idee des Reebgraphen sowie die Bildung des Verbindungsbaumes beim Abtastalgorithmus der Konturbäume (siehe [Abschnitt 3.2](#)) aufgegriffen und es werden kritische Punkte im Skalarfeld aller Partikel gesucht. Im Gegensatz zu den beiden genannten Ansätzen wird die Suche auf lokale Maxima beschränkt. Dies erfüllt die Anforderung, Partikel einer Zusammenhangskomponente in Gruppen zu unterteilen und ermöglicht zudem eine hohe Geschwindigkeit.

Gaspartikel werden übersprungen, Flüssigkeitspartikel ohne Nachbarn werden übersprungen

Keine Sortierung wie beim vorgestellten Konturbaum. Sie ist teuer und würde den *Fast-Depth-Algorithmus* nicht beschleunigen. Stattdessen werden auf dem Pfad getroffene Partikel zum Cluster mit hinzugefügt.

Sobald die umliegenden Partikel nur noch den gleichen oder einen höheren Tiefenwert aufweisen, gilt die Suche als abgeschlossen und der aktuelle Partikel ist gefunden.

Zur Beschleunigung werden alle Partikel, die beim Suchen entlang des Pfades getroffen wurden, ebenfalls zum Cluster hinzugefügt.

Der Begriff dieses Algorithmus wurde ausgehend von der Nutzung des *Tiefenwertes* gewählt, angelehnt an die Depth-First Suche in Baumstrukturen.

5.6. REDUZIERUNG DER KLEINEN CLUSTER

Sehr kleine Cluster in den Zusammenhangskomponenten können unerwünschtes Rauschen in der Ereigniserkennung verursachen, so dass deren Partikel benachbarten, größeren Clustern zugewiesen werden. Ausgenommen sind diejenigen Cluster, die alleinig eine Zusammenhangskomponente bilden, auch wenn sie die Mindestgröße unterschreiten.

Variante 1: Clusterbeziehungen 1) Zwei Mengen, eine mit den kleinen Clustern und eine mit den Übrigbleibenden. 1) mit jedem kleinen Cluster nach den naheliegensten Übrigbleibenden suchen, z.B. in dem die Rootpartikel der Übrigbleibenden in einen kd-Baum angeordnet werden. 3) Bestimmen, der kleine Cluster mit den gefundenen Nachbarcluster zusammenhängen. Schritt 3) ist das Problem: Zusammenhangskomponente zwischen root aktueller Cluster und root benachbarter Cluster müsste bestimmt werden: Sehr aufwändig und nur sinnvoll, wenn bereits clusterunabhängige Zusammenhangskomponenten gebildet worden wären.

Variante 2: Bestehende Nachbarschaftsbeziehung der Partikel nutzen. 1) Zwei Mengen, eine mit den kleinen Clustern und eine mit den Übrigbleibenden; in der Implementation nicht nötig (Clustergröße wird direkt getestet). 2) Die Nachbarn jedes Partikel der minCluster durchsuchen: gleicher Cluster? Mit den Nachbarpartikeln weitersuchen, bis Partikel mit Übrigbleibenden gefunden. a) Keiner dabei? Cluster bestehen lassen und alles abbrechen. b) Nur einer? Hinzufügen c) Mehrere? Anhand des Winkels entscheiden: $1+x$ Vektoren: 1: Richtung root -> betrachteter Partikel x: Richtung Partikel -> Wurzel benachbarter Cluster 1, ...

Bei welchem x Vektor der Winkel zum 1. Vektor am Kleinsten ist: Partikel zu diesem Cluster hinzufügen.

MinClusterSize von 10.

Begrenzung: Sucht maximal den drittentferntesten Nachbarn nach Clustern

`/// No deletion of clusters here to not destroy /// referencing by vector indices. Instead zero /// particle clusters should be ignored by the /// compareCluster function.`

Da die Partikel voneinander unabhängig sind, können sie parallel durchlaufen werden. Weiterhin wird die Winkelberechnung zu jedem benachbarten Partikel mit OpenMP parallelisiert.

5.7. BESTIMMUNG DER CLUSTERZUSAMMENHÄNGE

Die bestehenden Nachbarschaftsbeziehungen der Partikel können zum Vergleich der Clusterbeziehungen zwischen zwei Zeitschritten genutzt werden, so dass keine neuen Datenstrukturen zur Bestimmung erstellt werden müssen.

[Structure Events Cluster Compare \(SECC\)](#)

Einfärbung der Cluster parallelisiert.

5.7.1. NACHBARSCHAFTSGRAPH

Eine Variante ist die Erzeugung eines Graphen, der den Zusammenhang zwischen den Clustern zeigt. Dazu kann die Clusterzugehörigkeit der Nachbarn jedes Partikel untersucht werden. Sollte ein Nachbar einem anderen Cluster angehören, als der aktuelle Partikel, so sind der Cluster des aktuellen und der Cluster des benachbarten Partikels zusammengehörig.

Problem: Clusteridentifizierung schwierig, da sie über den root Partikel läuft und dieser sich mit großer Wahrscheinlichkeit im nächsten Frame ändert. Es müsste ein neues System zur Clusterbestimmung eingeführt werden (z.B. Durchschnittswert der enthaltenen Partikel IDs mit einer Toleranzgrenze, die experimentell bestimmt werden müsste).

5.7.2. FRAMEÜBERGREIFENDER VERGLEICH DER PARTIKELZUGEHÖRIGKEIT

Ein direkteres Vorgehen, das unabhängig der Nachbarschaftsbeziehungen von Clustern ist, bietet der Vergleich der Partikelmengen der Cluster. Diese Methode greift die Idee des Volumenvergleichs für *Isooberflächen* von Konturbäumen auf [Abschnitt 3.2](#). Die Cluster nehmen dabei die Position der *Konturen* ein und die Partikel die Position des Gitters, da sie über ihren Identifikator eindeutig zugeordnet werden können. Weil hier keine kontinuierliche Fläche existiert, wird nicht von eingeschlossenem Volumen, sondern von Mengen gesprochen.

Zu jedem Cluster ist eine disjunkte Menge an Partikeln zugeordnet. Um diese Mengen über zwei Zeitschritte hinweg miteinander zu vergleichen, wird eine Matrix aufgebaut, deren Zeilen die Cluster des neuen Zeitschrittes enthalten und deren Spalten die Cluster des alten Zeitschrittes. Ein Partikel wird in diese *Clustervergleichsmatrix* eingetragen, indem die Clusterzugehörigkeit des aktuellen Zeitschrittes sowie die Clusterzugehörigkeit des vorherigen Zeitschrittes ermittelt wird.

Anschließend werden zwei Schritte übernommen, um die für die Auswertung und schließlich die für die Eventerkennung notwendigen Werte zu ermitteln. In jedem Schritt werden alle Cluster eines Zeitschrittes betrachtet. Für jeden dieser Cluster werden Cluster des anderen

Zeitschrittes gesucht, die mit dem betrachteten Cluster gemeinsame Partikel aufweisen. Diese werden als *Partnercluster* bezeichnet. Im ersten Schritt werden alle derzeitigen Cluster mit jedem des vorherigen Zeitschrittes verglichen (vorwärtsgerichteter Vergleich), im zweiten Schritt alle vorherigen mit jedem Cluster des aktuellen Zeitschrittes (rückwärtsgerichteter Vergleich).

Beide Schritte geben die Anzahl der gemeinsamen Partikel für jeden vorhergehenden Cluster und jeden aktuellen aus, woraus sich die weiteren für die Analyse notwendigen Werte ermitteln lassen.

5.8. ERKENNUNG VON EREIGNISSEN

Es wird jeweils ein Cluster und seine *Partnercluster* betrachtet.

Heuristisches Vorgehen:

Tests haben gezeigt, dass die Partikel selten in Clustern mit derselben Clusternummer bleiben. Das lässt sich damit erklären, dass die Clusterzuweisung stark von der Position der Partikel und ihrer Nachbarschaftsverhältnisse abhängen und diese sich durch die sich bewegenden Partikel von Zeitschritt zu Zeitschritt stark ändern.

Somit muss ein Vergleich der Anteile von Partikeln in den Clustern durchgeführt werden, um zu erkennen, ob sich der Zusammenhang zwischen den Partikeln verändert hat. Die für die Betrachtung relevante Clustereigenschaft ist die Anzahl der Partikel eines Clusters und wird als *Clustergröße* bezeichnet. Die

Betrachtete Faktoren: - Relevanz: clusterSize (nicht unbedingt aussagekräftig, daher Prozentwerte)

- selber Cluster: getAveragePartnerCommonPercentage groß, getLocalMaxTotalPercentage groß, getNumberOfPartners klein

- Rauschen: getAveragePartnerCommonPercentage klein, getSmallPartnerAmount groß, LocalMaxTotal klein oder getBigPartnerAmount(25) = 0

- Split: getBigPartnerAmount(25) > 1 (forward), evtl zusätzlich Partners sehr klein und getAveragePartnerCommonPercentage mittel (backwards) bei großen Clustern size und getCommonPercentage hoch

- Merge: getBigPartnerAmount(25) > 1, 25 für = 2 zu niedrig: 40, 35, 30, testen (backwards), bei großen Clustern size und getCommonPercentage hoch

- Birth: getNumberOfPartners = 0 (backwards) oder getTotalCommonPercentage < 10.

- Death: getNumberOfPartners = 0 (forward) oder getTotalCommonPercentage < 10.

Die für die Spliterkennung verwendete Methode erfasst nicht die Abspaltung von Clustern, die ??% kleiner sind als der Ausgangscluster. Bei Merge ist es synonym.

Werte relativ zum Anteil der gemeinsamen Partikel, um kleine und große Cluster gleichermaßen zu behandeln.

CODING TODO vielleicht: Beachtet werden muss ein Rauschen, welches durch ständiges Hinzufügen und Lösen von Randpartikeln eines Clusters besteht. Da dieses Rauschen mit zunehmender Clustergröße durch die höhere Anzahl an Randpartikeln verstärkt wird, werden die Betrachtungen mit prozentualen Werten durchgeführt. Andererseits können so große Cluster verstärkt in das Rauschen eingeteilt werden, so dass auch große *Partnercluster* mit Werten kleiner als der festgelegte Grenzwert für Split und Merge Ereignisse in Betracht gezogen werden. Dabei wird getCommonPercentage auf hohe Werte untersucht.

Erste Vorgehensweise: 1) Vergleich eines vorhergehenden Clusters mit allen Folgenden (Spalte), Referenzclustergröße: alter Cluster a) 0 gemeinsame (keine Partikel in einem Cluster): Death b) 0 im Selben, viele/alle in (einem oder mehreren) anderen: Merge c) Sehr viele/alle im Selben, wenige in anderen: Nichts passiert oder Merge (damit nicht feststellbar). d) Hälfte im Selben, Hälfte in einem anderen oder verteilt: Split Birth so nicht feststellbar.

2) Vergleich des aktuellen Clusters mit allen Vorhergehenden (Zeile), Referenzclustergröße: neuer Cluster a) 0 gemeinsame (keine Partikel in vorhergehenden Clustern): Birth b) Schritt (c)

Abbildung 5.1. Datentypen

von (a) noch machen. Death so nicht feststellbar.

5.9. SPEICHERUNG DER DATEN

Umwandlung von framebehafteten MMPLD in frameloses MMSE.

Das [MegaMol Structure Event \(MMSE\)](#) enthält die berechneten Ereignisse in einem Bytecode fest. Der Vorteil im Vergleich zu JSON/XML oder anderen textbasierten Speicherverfahren ist die höhere Performance beim Lesen und Schreiben sowie die kleinere Dateigröße. Die Ereignisdaten sind dabei sequentiell abgelegt

5.10. PROGRAMMAUFBAU FÜR DIE BERECHNUNG

Speicherung der Clustervisualisierung durch MMPDC.

Auslösung der Berechnung für die unterschiedlichen Frames erfolgt durch den verbundenen SimpleSphereRenderer, da der aus der Calculation ausgehende MPDC für den Wechsel der Frames verantwortlich ist.

Call zur intermodularen Kommunikation. Sämtliche Listenelemente, auf die mit Zeigern zugegriffen wird - wie Partikel, Cluster und Strukturereignisse - sind dichtgepackte Strukturen mit Datentypen der Länge vier oder acht Byte, damit der Compiler kein zusätzliches Padding einfügt und so die Strides und damit die Pointer ungültig macht.

6. VISUALISIERUNG

6.1. GRUNDLAGEN

6.1.1. GLYPHDESIGN

6.1.2. GESTALTGESETZE

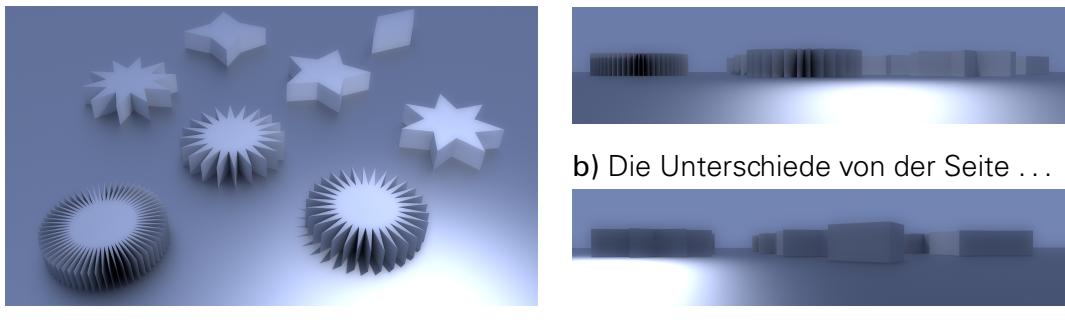
Die Gestaltgesetze, formuliert von Wertheimer und erweitert von Stephen Palmer (QUELLEN), sind eine Möglichkeit der Einteilung für die Fähigkeit der menschlichen Wahrnehmung, in Sinneseindrücken Strukturen und Ordnungen zu erkennen. Für diese Arbeit spielt einerseits die Erkennung von Zusammengehörigkeit zwischen Partikeln und Ereignissen als auch die Unterscheidungsmöglichkeit der Ereignisarten voneinander eine Rolle. Beides kann durch die Berücksichtigung der Gestaltgesetze gezielter erreicht werden.

Die Erkennung der Zusammengehörigkeit ist bereits durch die Aufgabenstellung gegeben, da die Ereignisse im Raum der Partikel dargestellt werden sollen. Dabei wird das Gesetz der Nähe genutzt, das besagt, dass Elemente mit geringen Abständen zueinander als zusammengehörig wahrgenommen werden. Auch kann die Erkennung von Clustern in der Ausgangsanimation, die keine Hilfsmittel wie Einfärbung der Cluster oder Markierung durch Glyphen verwendet, darauf zurückgeführt werden.

Die Erkennung und Unterscheidung der Ereignisarten wird durch die Glyphgestaltung beeinflusst. Diese wird vom Gesetz der Geschlossenheit, vom Gesetz der Ähnlichkeit sowie vom Gesetz der Prägnanz maßgeblich bestimmt. Das erste beschreibt, dass eine Fläche, die von Linien umschlossen wird, leichter als eine Einheit aufgefasst werden kann als eine nicht umschlossene Fläche. Das zweite sagt aus, dass einander ähnliche Elemente eher als zusammengehörig erlebt werden als einander unähnliche. Letzteres spricht von einer Prägnanztendenz, durch die besonders solche Gestalten wahrgenommen werden, die sich durch ein bestimmtes Merkmal von anderen abheben, als auch von der „Guten Gestalt“, die die Reduktion von Figuren auf einfache Strukturen bei der Wahrnehmung ausdrückt.

Die Partikel in der Animation werden auch durch das Gesetz der gemeinsamen Bewegung als zusammengehörig empfunden; sich gleichzeitig und in dieselbe Richtung bewegende Elemente werden als eine Gestalt wahrgenommen. Das spielt aufgrund der statischen Position der Ereignisglyphen für diese Arbeit keine Rolle.

Die zum Testen der Clustererkennung verwendete Einfärbung der Partikel nutzt das Gesetz der Kontinuität. Es besagt, dass Reize, die eine Fortsetzung vorangehender Reize zu sein scheinen, als zusammengehörig angesehen werden.

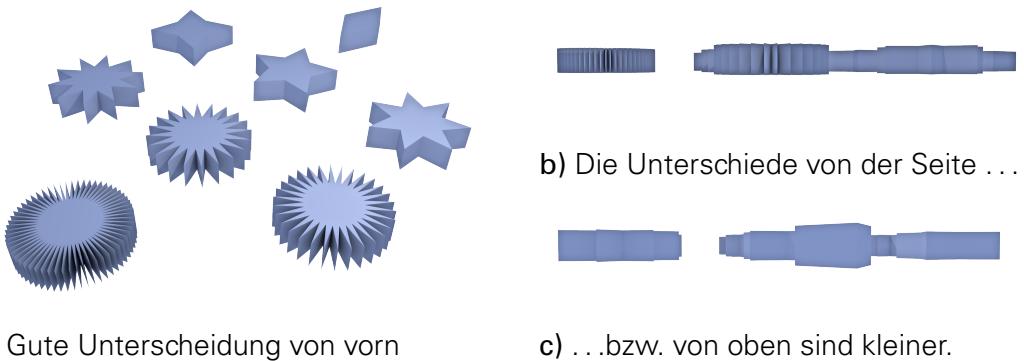


a) Gute Unterscheidung von vorn

b) Die Unterschiede von der Seite ...

c) ... bzw. von oben sind kleiner.

Abbildung 6.1. Formgebung mit sternenförmigem, nichtkonvexen Prisma.
ERKENNBARKEIT SCHLECHTER ALS BEI TRANS UNTEN?



a) Gute Unterscheidung von vorn

b) Die Unterschiede von der Seite ...

c) ... bzw. von oben sind kleiner.

Abbildung 6.2. Formgebung mit sternenförmigem, nichtkonvexen Prisma.
TRANSPARENT

6.1.3. PRÄATTENTIVE WIRKUNG VISUELLER VARIABLEN

- Präattentive Wirkung visueller Variablen (für Priorisierungsreihenfolge visueller Attribute in [Abschnitt 6.2.1](#) sowie die Nutzung für Agglomerate). Räumliche Verknüpfungen sind oftmals präattentiv, z.B. Bewegung und (Farbe | Form| 3D-Disparität). Die meisten Verknüpfungen sind nicht präattentiv. => Nutzung möglichst weniger Variablen fördert präattentive Informationsaufnahme => Mehrfachzuweisung (Farbe und Form für Zeit z.B.) vermeiden, auch wenn sie scheinbar zur verstärkung beitragen.
- falls visuelle Relationen vorkommen: Beispiele von Netzwerken
- falls Zoom eine Rolle spielt: Ansichten, siehe KP InfVis

6.1.4. FORMGEBUNG

Eine einfache Möglichkeit, unterschiedliche Formen aus einer runden Grundform zu erzeugen, ist ein Zylinder mit nach außen führenden Spitzen wie bei einem Stern, was einem nichtkonvexen Prisma entspricht [CLM54]. Wie in [Abbildung 6.1](#) zu sehen, sind die Sternformen von vorn und hinten gut zu unterscheiden. Von der Seite und von oben fällt die Abgrenzung voneinander jedoch schwerer. Weiterhin haben die Elemente in der Seitenansicht eine andere Erscheinung als in der Frontalansicht, so dass dadurch die Verknüpfung zwischen beiden Ansichten mit zusätzlichem kognitiven Aufwand verbunden ist. Deswegen ist diese Art der Formgebung für eine 3D Ansicht nicht geeignet.

Die Anforderung, eine Formgebung zu wählen, die einen ähnlichen Umkugelradius und damit eine einheitliche Größe erhält, einen aus allen Winkeln ähnlichen Durchmesser sowie eine ähnliche Form aufweist, führt zu bestimmten Gruppen von Polyedern. Polyeder sind

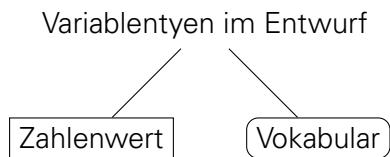


Abbildung 6.3. Taxonomie der Variablenarten

dreidimensionale Polytope und im engeren Sinne eine Teilmenge des \mathbb{R}^3 , welche ausschließlich durch gerade Flächen begrenzt wird. Darunter zählen unter anderem die fünf *platonischen* [Bog15], die 13 *archimedischen*, die 13 *catalanischen* sowie die 92 *Johnson-Körper* [Joh66]. Eine Charakteristik dieser Polyeder ist ihre konvexe Form, was gleichzeitig die Unterscheidung aus der Distanz erschwert. Die prismatischen Polyeder fallen ebenfalls unter diese Kategorie, werden jedoch aus denselben Gründen ausgeschlossen wie die sternförmigen Prismen.

Andererseits erfüllen die sternförmigen *Kepler-Poinsot-Körper* („Sternpolyeder“) die Anforderungen. Sie besitzen im Gegensatz zu den vorgenannten Polyedern auch konkave Flächen [Wei15] und können daher, wie in der interaktiven Visualisierung von [Gra15] zu sehen, leicht von den konvexen Polyedern unterschieden werden.

Ebenfalls geeignet ist der Vorschlag, eine Sternform durch sogenanntes „Poken“ zu erzeugen: „By “poke” we mean a function to create a pyramid on each face of a given object. (This is different from Kepler’s stellation operation, which extends the face planes.)“ [Har08, S. 4]

Somit gibt es mit der auf nichtprismatischen Polyedern basierenden Formgebung eine hohe Anzahl verschiedener Formen, die alle die Anforderung an eine einheitliche Größe, an einen aus allen Blickwinkeln ähnlichen Durchmesser und an die Wiedererkennbarkeit gewährleisten. Lediglich die Unterscheidung innerhalb der konvexen Polyeder kann schwerfallen. Am Deutlichsten unterscheiden sich die *platonischen Körper* untereinander sowie die konvexen von den konkaven, sternförmigen Polyedern.

6.2. VISUALISIERUNGSENTWURF

6.2.1. TAXONOMIE DER VISUALISIERUNG

Die Berechnung des Clusterverhaltens ergibt eine Datenbank, in der Ereignisse abgelegt sind. Ein Ereignis stellt einen Datensatz dieser Datenbank dar, die Parameter des Ereignisses sind die Datensatzfelder. Die Aufgabe der Visualisierung ist es, den Parametern eines Ereignisses visuelle Attribute zuzuweisen. Diese Zuweisung kann in einer weiteren Datenbank als Relation zwischen Parameter und Attribut festgehalten werden.

Zum leichteren Verständnis erfolgt eine Einteilung der Parameter, der Attribute sowie der Zuweisungen in die beiden in Abbildung 6.3 gezeigten Variablenarten. Der Typ *Zahlenwert* steht dabei für die Nutzung eines kontinuierlichen Zahlenraumes, während der Variabtentyp *Vokabular* eine diskrete, beschränkte Zuordnung mit voneinander disjunkten Elementen beschreibt.

PARAMETER EINESEREIGNISSES

Ein Ereignis besteht aus drei Parametern, die jeweils zwischen ein bis drei Freiheitsgraden aufweisen.

- Zeitpunkt: ein Wert (t)
- Ort: drei Werte (x, y, z)
- Art: ein Vokabular [Split, Merge, Death, Birth]

VERGLEICHSBILD MIT WECHSELNDER SÄTTIGUNG UND HELLIGKEIT

Abbildung 6.4. Farbtafel mit unterschiedlicher Sättigung und Helligkeit

VISUELLE ATTRIBUTE EINES EREIGNISELEMENTS

Die visuellen Attribute können sowohl als fließender *Zahlenwert* als auch als festes *Vokabular* umgesetzt werden. Beispielsweise bedeutet der Vokabulartyp für das *Positionsattribut* die Festlegung bestimmter Koordinaten für jeden Begriff des Vokabulars.

Jedes Attribut besitzt eine individuelle Anzahl an Freiheitsgraden, die zur Visualisierung genutzt werden können.

- Position: 3 (x, y, z)
- Farbe: 2 (Farbwert/Sättigung, Helligkeit)
- Form: 1, eine Wertzuweisung ist möglich durch die Anzahl der Ecken, Kanten und Flächen. Anderes wie z.B. der Abstand der Ecken vom Körperzentrum kollidiert mit dem Attribut Größe.
- Größe: 1
- Opazität: 1

Da Farbwerte bei geringer Sättigung nicht mehr voneinander unterscheiden werden können, werden diese beiden Farbeigenschaften zusammengefasst. Dadurch kann die Helligkeit leicht als separates Attribut vom Betrachter erkannt werden, wie in [Abbildung 6.4](#) zu sehen.

Bei der *Form* sind mehrere Freiheitsgrade möglich, etwa eine Kombination aus Anzahl der Ecken mit einem variablen Abstand dieser vom Körperzentrum. Der variable Abstand kollidiert jedoch mit dem Attribut *Größe*. Daher wird das Attribut *Form* auf einen Freiheitsgrad beschränkt und es werden die in [Unterabschnitt 6.1.4](#) beschriebenen Polyeder verwendet. Da die Anzahl an Polyedern begrenzt ist und die Unterscheidungsmöglichkeit mit zunehmender Flächenanzahl sinkt, ist das Attribut *Form* nur eingeschränkt als Zahlenwert verwendbar.

Bei der Bestimmung der Attribute wurde darauf geachtet, ein umfassendes Spektrum der Gestaltgesetze nutzen zu können. Alle visuellen Attribute fallen unter den Einfluss des *Gesetzes der Prägnanz*. Mit der *Position* kann das *Gesetz der Nähe* sowie das *Gesetz der gemeinsamen Region* genutzt werden, eine Kombination aus der *Position* mit den anderen Attributen resultiert in der Verwendbarkeit des *Gesetzes der Kontinuität*. Bis auf die *Position* kann mit allen visuellen Attributen das *Gesetz der Ähnlichkeit* genutzt werden. Das *Gesetz der Geschlossenheit* kann mit dem *Positionsattributs* verarbeitet werden, falls durch entsprechende Zuweisung die Elemente nah beieinander liegen und sie eine geschlossene Form bilden. Mit den hier definierten Attributen kann somit von sechs der zehn in [Unterabschnitt 6.1.2](#) aufgezählten Gestaltgesetze Gebrauch gemacht werden.

Texturen als ein eigenständiges visuelles Attribut werden ausgeklammert, da sie sich mit dem Attribut *Farbe* und bei der Nutzung als Oberflächenstruktur mit dem Attribut *Form* überschneiden. Ein Einsatz als Bump Map oder Displacement Map zur Unterstützung des Attributs *Form* wäre denkbar. Aufgrund dessen, dass bereit Polyeder für das *Formatattribut* genutzt werden, wird von einer Verwendung von Texturen allerdings abgesehen.

Animationen bieten weitere visuelle Attribute wie Bewegungsrichtung, Bewegungsbahn, Beschleunigung und Geschwindigkeit als auch die Veränderung der anderen Attribute und eröffnen damit die Nutzung des *Gesetzes der Gleichzeitigkeit* sowie des *Gesetzes der gemeinsamen Bewegung*. Sie werden im Rahmen dieser Arbeit jedoch nicht behandelt.

Die visuellen Attribute lassen sich durch den Betrachter unterschiedlich gut unterscheiden. In der folgenden Sortierung nach ihrer Unterscheidungsgüte stehen die am Leichtesten zu unterscheidenden Attribute ganz oben. (QUELLE)

1. Position

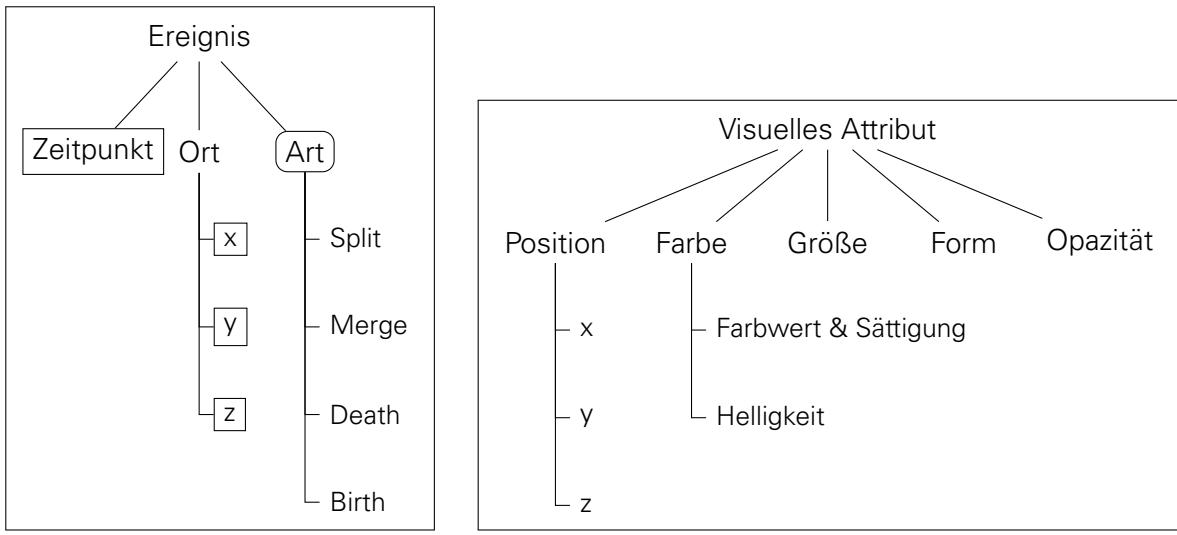


Abbildung 6.5. Taxonomie der Daten und visuellen Attribute

2. Farbwert
3. Größe
4. Form
5. Helligkeit
6. Opazität

Die Attribute mit höherer Unterscheidungsgüte sollten bevorzugt gewählt werden. Form, Helligkeit und Opazität sind durch die geringere Güte besser als Vokabular- denn als Wertattribut geeignet, da bei den Vokabularen die Anzahl der Begriffe im Vergleich zur Menge der Zahlenwerte gering ist. Dadurch können die Attributseigenschaften weiter auseinanderliegen, was eine leichtere Unterscheidung ermöglicht.

Die

In der *MegaMol* Molekularsimulation wird ausschließlich das visuelle Attribut *Position* zur Darstellung genutzt.

PROBLEM DER LOKALEN UND TEMPORALEN AGGLOMERATION

Die Anzahl der Datensätze, die denselben Ort oder denselben Zeitpunkt aufweisen können, ist unbekannt. Schon eine geringe Anzahl identischer Orte oder Zeitpunkte kann je nach Zuordnung der visuellen Attribute zu einer überlagerten, nicht mehr unterscheidbaren Darstellung führen, wodurch die Erkennung solcher Agglomerate nicht gegeben ist. Daher wird ein visuelles Attribut reserviert, um eine solche Anhäufung von Ereignissen anzuzeigen. Die *Häufung* wird bei der Zuweisung zu den visuellen Attributen wie ein Ereignisparameter behandelt. Abhängig von dieser Zuweisung in [Unterabschnitt 6.2.2](#) können zwei Werte für die lokale und temporale Häufigkeit notwendig sein. Dieser wird jedem Ereignis zugewiesen und wird beim Auffinden von identischen Ortsparametern bzw. Zeitparametern in den Datensätzen inkrementiert.

Die Begriffe *Agglomeration* und *Häufung* sind hier synonym.

Bei der Visualisierung von Agglomeraten kann die natürliche Wirkung (QUELLE) der visuellen Attribute genutzt werden. Groß, opaque und dunkel stehen für eine starke Agglomeration. Klein, transparent und hell für eine niedrige.

Tabelle 6.1. Zuweisungsmöglichkeiten der Parameter zu den visuellen Attributen bei Kopplung des *Ortes* an die *Position*. Zur Anzeige der Häufung H wird die *Opazität* reserviert. Die Zeit wird in einigen Varianten nicht kodiert.

Attribu- te	Position x	Position y	Position z	Farb- wert & Sätti- gung	Hellig- keit	Ecken- anzahl	Größe	Opazi- tät
Ort x	Ort y	Ort z	Art	Zeit	Zeit	Zeit	Zeit	H
	Ort y	Ort z	Art					H
	Ort y	Ort z	Art					H
	Ort y	Ort z	Art					H
Ort x	Ort y	Ort z	Zeit	Art	Zeit	Zeit	Zeit	H
	Ort y	Ort z	Zeit	Art				H
	Ort y	Ort z	Zeit	Art				H
	Ort y	Ort z	Zeit	Art				H
Ort x	Ort y	Ort z	Zeit	Zeit	Art	Art	Art	H
	Ort y	Ort z	Zeit	Zeit	Art			H
	Ort y	Ort z	Zeit	Zeit	Art			H
	Ort y	Ort z	Zeit	Zeit	Art			H
Ort x	Ort y	Ort z	Zeit	Zeit	Zeit	Art	Art	H
	Ort y	Ort z	Zeit	Zeit	Zeit	Art		H
	Ort y	Ort z	Zeit	Zeit	Zeit	Art		H
	Ort y	Ort z	Zeit	Zeit	Zeit	Art		H

VERBUNDENHEIT

Wenn sich ein Cluster teilt (Split) und sich die resultierenden Cluster wiederum teilen oder mit anderen Clustern verschmelzen (Merge), so kann man diesen Ereignissen Vorgänger (Eltern) sowie Nachfolger (Kinder) zuordnen. IST DAS RICHTIG?

Gesetz der Geschlossenheit Gesetz der fortgesetzt durchgehenden Linie Gesetz der verbundenen Elemente

6.2.2. PARAMETER-ATTRIBUTSUWEISUNG

In der *MegaMol* Simulation wird der Ort des Ereignisses mithilfe der Teilchenposition dynamisch visualisiert. Insofern kann angenommen werden, dass für eine nahtlose Benutzbarkeit zwischen der *MegaMol* Animation und der statischen Visualisierung die Zuweisung des Elementattributes *Position* an den Parameter *Ort* empfehlenswert ist. Weiterhin ist die Reservierung der *Opazität* für die *Häufung* die einfachste Möglichkeit, denn dies hat zwei Vorteile. Zum Einen ist keine nachträgliche Modifizierung der visuellen Attribute der einzelnen Ereignisse notwendig, denn die Opazitätseigenschaft sorgt durch Überlagerung für eine von der Anzahl direkt abhängige Hintergrundverdeckung. Dadurch werden zum Anderen auch keine Informationen über die Häufigkeit im Datensatz benötigt. Hingegen muss zum Beispiel eine Größenveränderung nachträglich für jedes Ereignis im Agglomerat auf der Basis eines temporalen oder lokalen *Häufungswert* durchgeführt werden.

Darüberhinaus wird jedem Ereignisparameter nur eine visuelle Attribut zugeordnet, denn Mehrfachzuweisungen sollten wegen einer Verminderung der präattentiven Wahrnehmung (VERWEIS AUF GRUNDLAGEN) vermieden werden.

Daraus ergeben sich die in [Tabelle 6.1](#) aufgeführten Visualisierungsmöglichkeiten.

Tabelle 6.2. Ausgewählte Zuweisungsmöglichkeiten der Parameter zu den visuellen Attributen bei Loslösung des *Ortes* von der *Position* sowie Trennung der *Häufung H* von der *Opazität*. Nicht jeder Parameter wird kodiert. UNVOLLSTÄNDIG

Attribut	Position x	Position y	Position z	Farbwert & Sättigung	Helligkeit	Eckenanzahl	Größe	Opazität
Zeit	H			Art				
Zeit	Art			H	Ort x			
Ort x	Art						H	
Ort x	Zeit					H		

Derzeit ist nicht bekannt, wie groß die örtliche Konzentration von Ereignissen ist. Sollte sie eine kritische Anzahl überschreiten und das der *Häufung* zugewiesene Attribut keine sinnvollen Ergebnisse mehr liefern können, so kann der Ort des Ereignisses einem anderen Attribut zugewiesen werden. Weiterhin kann das Loslösen des *Ortes* von der Position sinnvoll sein, wenn für eine Auswertung der Ort des Ereignisses nur eine untergeordnete Rolle spielt. Vorschläge für entsprechende Zuweisungsmöglichkeiten sind in [Tabelle 6.2](#) zu finden.

Da jedoch in der Molekularsimulation der Ort eines Ereignisses der visuellen Position zugewiesen ist und die Visualisierung in der vorliegenden, ersten Iteration mit der Simulation harmonieren soll, erfolgt die Zuweisung des Ortes ausschließlich an die *Position*, so dass die weitere Betrachtung auf die in [Tabelle 6.1](#) gezeigten Zuweisungsmöglichkeiten beschränkt wird.

6.2.3. MOCKUPS

WERKZEUG

Zur Überprüfung einer sinnvollen Zuweisung der Parameter zu den Attributen dienen Mockups. Aufgrund der Vielzahl der in [Unterabschnitt 6.2.2](#) genannten Kombinationsmöglichkeiten wird eine programmatische Erstellung der Mockups gewählt. Da eine schnelle Umsetzbarkeit im Vordergrund steht, wird *Unity3D* als Entwicklungsumgebung gewählt. Ein Vorteil dieser Engine ist die Möglichkeit, rasch Prototypen erstellen zu können sowie die einfache Umsetzbarkeit für verschiedenste Plattformen, ohne auf Containermanager wie *Docker* angewiesen zu sein. Letzterer Umstand ist für den möglichen Einsatz der Mockups als Evaluationswerkzeug von Bedeutung. Die Performance spielt beim Mockup eine untergeordnete Rolle.

Beim Deploy der *Unity3D* WebGL Anwendung kam es auf bestimmten Apachekonfigurationen zu Problemen. Dies war durch eine [manuelle Umbenennung von Dateien und Verweisen](#) in der durch *Unity3D* generierten Anwendung zu lösen. Von offizieller Seite wurde dies als [Bug markiert](#). SOLCHE SÄTZE HABEN IN DER BA VERMUTLICH NIX ZU SUCHEN?

6.2.4. DATENSATZ

Um eine Vergleichbarkeit der verschiedenen Mockups zu gewährleisten, liegt ein fester Datensatz in [Tabelle A.1](#) zugrunde. Sämtliche Werte sind einheitenlos und wurden für die Parameter *Ort* und *Zeit* so gewählt, dass eine Überlappung der Ereigniselemente von nebeneinanderliegenden Ereignissen bei einer Zuweisung an das visuelle Attribut *Position* nicht auftritt, da andernfalls durch das *Gesetz der verbundenen Elemente* ein falsches Abbild der zugrundeliegenden Daten übertragen werden würde. Der Zahlenbereich für die Zeit ist auf das Intervall [1, 22] begrenzt. Da sich in der Simulation die meisten Elemente mit zunehmender Zeit vom Zentrum entfernen,

wird die x-Koordinate des Ortes mit [Gleichung 6.1](#) direkt an die Zeit t gekoppelt.

$$x = \left\lfloor \frac{4}{3} \cdot t + 0,5 \right\rfloor \quad (6.1)$$

Aufgrund des zylinderförmigen Raumes in der Simulation wird sowohl für die y-Koordinate als auch für die z-Koordinate des Ortes ein identisches und kleineres Intervall von [1, 10] gewählt.

Die Zuweisung der *Art* des Ereignisses erfolgt manuell. Da die simulierten Partikel in der *MegaMol* zu Beginn der Simulation einen einzigen Cluster in flüssiger Phase bilden, kommen bei kleinen Zeitwerten vor allem Splits vor.

Zur Darstellung der in [Abschnitt 6.2.1](#) beschriebenen Häufung, werden die Ereignisse 6 – 9, 35/36 sowie 46 – 48 jeweils gruppiert und händisch an denselben Ort und dieselbe Zeit gesetzt.

ZUWEISUNG DER ZAHLENWERTE

Zur Gewährleistung der in [Unterabschnitt 6.2.4](#) beschriebenen Vermeidung unnötiger Überlappungen der Ereigniselemente wird der Standardwert des Attributs *Größe*, d.h. wenn das Attribut keinem Parameter zugewiesen ist, entsprechend skaliert. Um nah bei der Visualisierung der Molekularsimulation selbst zu bleiben, wird in den Mockups, die nicht das Attribut *Form* beinhalten, eine Kugel verwendet.

Die Attribute in den Mockups weisen folgende Zahlenbereiche auf:

- x-, y-, z-Position: $[-\infty, \infty]$, beim Parameter *Ort* erfolgt eine direkte Zuweisung
- Farbwert: $[0, 340]^\circ$ bei 100% Sättigung ([Hue-Saturation-Brightness \(HSB\)](#) Modell)
- Helligkeit $[40, 100]\%$, Standardwert 100% ([HSB](#)-Modell)
- Form: besitzt keinen Zahlenbereich, sondern beinhaltet verschiedene Formen von Polyedern sowie die Kugel als Standardwert
- Skalierung: $[0.25, 2]$, Standardwert 1
- Opazität: $[40, 100]\%$, Standardwert 100%

Die Mindesthelligkeit b_{\min} von 40% dient dazu, um noch eine Farbwertunterscheidung treffen zu können, genauso wie die Mindestopazität von ebenfalls 40%. Der Maximalfarbwert h_{\max} von 340° wird festgelegt, um eine Unterscheidung zu den niedrigen Farbwerten zu erhalten.

Da die Grenzwerte der visuellen Attribute a_{\min} und a_{\max} aus [Unterabschnitt 6.2.3](#) bekannt sind und sich für die Parameter p aus [Tabelle A.1](#) ein minimaler und maximaler Wert p_{\min} und p_{\max} bestimmen lässt, wird für die Berechnung der Attributwerte die Linearfunktion in [Gleichung 6.2](#) verwendet.

$$\begin{aligned}
 a &= m \cdot p + n \\
 a_{\max} &= m \cdot p_{\max} + n \\
 a_{\min} &= m \cdot p_{\min} + n \\
 m &= \boxed{\frac{a_{\max} - n}{p_{\max}}} \\
 n &= a_{\min} - m \cdot p_{\min} = a_{\min} - \frac{a_{\max} - n}{p_{\max}} \cdot p_{\min} \\
 &= a_{\min} - a_{\max} \cdot \frac{p_{\min}}{p_{\max}} + n \cdot \frac{p_{\min}}{p_{\max}} \\
 n \cdot p_{\max} &= a_{\min} \cdot p_{\max} - a_{\max} \cdot p_{\min} + n \cdot p_{\min} \\
 n \cdot (p_{\max} - p_{\min}) &= a_{\min} \cdot p_{\max} - a_{\max} \cdot p_{\min} \\
 n &= \boxed{\frac{a_{\min} \cdot p_{\max} - a_{\max} \cdot p_{\min}}{p_{\max} - p_{\min}}}
 \end{aligned} \quad (6.2)$$

Tabelle 6.3. Festlegung der Attributeigenschaften für den Parameter *Art* in den Mockups.

Art	Farbwert [°]	Helligkeit [%]	Position x	Form	Größe
Death	0	40	1	T	0,25
Merge	55	60	11	H	0,83
Split	245	80	21	O	1,42
Birth	145	100	31	D	2

Anschließend erfolgt eine Rundung zur handlichen Verwendung der Ergebnisse.

$$a = \lfloor m \cdot p + n + 0,5 \rfloor \quad (6.3)$$

Die Berechnungsergebnisse sind in [Abschnitt A.1](#) zu finden.

ZUWEISUNG DES VOKABULARS

Da die Mächtigkeit M des Vokabulars von *Art* bekannt ist, kann das Intervall v des Attributs a in [Gleichung 6.4](#) zur rechnerischen Ermittlung der Attributwerte a_i genutzt werden.

$$\begin{aligned} M &= 4 \\ \Rightarrow p_{0...3} &= \left(0, \frac{1}{3}, \frac{2}{3}, 1\right) \\ v &= a_{\max} - a_{\min} \end{aligned} \quad (6.4)$$

$$a_i = \{p_i \cdot v + a_{\min} \mid 0 \leq i \leq M - 1\}$$

Ausnahmen bilden die Attribute *Farbe* und *Form*. Wegen der kleinen Mächtigkeit des Vokabulars und zur Sicherung einer optimale Aufteilung des Farbwerts, wird dieser in [Tabelle 6.3](#) manuell definiert. Bei dem Attribut *Form* werden zur optimalen Unterscheidung vier Varianten aus der Menge der *platonischen Körper* ausgewählt.

AUSNAHME: KONSTANTER ATTRIBUTSWERT BEI OPAZITÄT

Aus den in [Unterabschnitt 6.2.2](#) aufgeführten Gründen ist es sinnvoll, die *Opazität* der Häufung zuzuweisen und sie mit einem konstanten Variablentyp auszustatten, so dass keine zusätzliche Ereigniswerte für die *Agglomeration* benötigt werden. Dieser konstante Wert wird auf 50% Opazität festgesetzt.

6.2.5. UNTERSCHIEDBARKEIT VON POLYEDERN

Da die Unterscheidung von Polyedern mit zunehmender Flächenanzahl schwieriger wird, findet im Mockup eine Begrenzung auf die fünf *platonischen Körper* mit einer geringen Ecken- und Flächenanzahl statt.

- T Tetraeder mit 4 Ecken und 4 Flächen
- O Oktaeder mit 6 Ecken und 8 Flächen
- H Hexaeder mit 8 Ecken und 6 Flächen
- I Icosaeder mit 12 Ecken und 20 Flächen
- D Dodekaeder mit 20 Ecken und 12 Flächen

Die Abkürzungen folgen dem Vorschlag von Baierl [[Bai04](#), S. 42]. Bei der Modellierung in *Blender* wurde für die Meshes die maximale Dimension von 1 (einheitenlos) entsprechend der Dimension der kugelförmigen Standardelemente in *Unity3D* gewählt.

Abbildung 6.6. Mockups mit freier Positionszuweisung.

6.3. ENTWURFSAUSWERTUNG

In den Abbildungen ?? sind Beispiele dafür zu sehen, wenn die *Häufung* anderen Attributen außer der *Opazität* zugewiesen wird.

6.3.1. GLYPHEN

- Polygone nicht so gut geeignet, da Unterscheidung auf Entfernung schwer möglich und auch hier noch Abhängigkeit vom Blickwinkel - daher zweidimensionale, vordefinierte Bilder an Eventtypen gekoppelt, immer zur Kamera ausgerichtet vgl. Sprites

Hier viel aus den Grundlagen Entwurfsabschnitten mit reinnehmen inkl. 3D-Prototyp!

6.4. VORGEHEN

6.4.1. GESTALTUNG DER GLYPHEN

Dicke Ränder zur Vermeidung von Aliasing und Flimmern.

6.4.2. RENDERER UND SHADER

Shader: common.btf für Standardlicht billboard.btf für Billboards in OGL 2.1 Nutzung von OpenGL 2.1 zur Wahrung der Kompatibilität mit den MM Modulen. Laden der Bilder mit lodepng

Nutzung der glm Bibliothek, da dadurch die Datentypen in C++ und in GLSL konsistent angelegt werden können.

- Begrenzung auf Zeitpunkte. - Begrenzen auf Eventarten.

Die Texturen haben einen festgelegten Pfad im Ordner der ausführenden Datei, da der Nutzer sie nicht ändern können soll. Dies liegt darin begründet, dass der Shader zur Darstellung des Zeitpunktes eines Ereignisses auf bestimmte Farben in der Textur angewiesen ist. So wird die Farbe lila (rgb 255, 0, 255) durch

Der Renderer wird bei einer hohen Anzahl von Ereignissen langsam, da für jedes Ereignis je nach Renderereinstellungen Berechnungen durchgeführt werden. Da sie für jedes Ereignis unabhängig voneinander erfolgen, können sie parallel ausgeführt werden. Aufgrund des Zugriffs auf die Quelldaten über das Inkrementieren von Zeigern, muss eine Einteilung der Berechnungen in eine feste Anzahl von Threads geschehen, deren Ermittlung mit der C++11 Standardbibliothek *thread* erfolgt.

6.4.3. VISUALISIERUNG DER CLUSTERBERECHNUNG

Um die Qualität der Clusterberechnung zu untersuchen, wird allen Partikeln desselben Clusters dieselbe Farbe zugeordnet. Im ersten berechneten Frame erfolgt diese Farzuzuweisung für jeden Cluster zufällig, in den darauffolgenden Frames wird die Farbe desjenigen Partnerclusters des vorhergehenden Frames übernommen, der die meisten gemeinsamen Partikel aufweist. Sollte der Cluster keinen Partnercluster besitzen, wird eine neue zufällige Farbe zugewiesen. Dies wird parallel

6.4.4. WEITERLEITUNG DER DATEN

Wie werden die Variablen im Call gefüllt bzw. die Funktionen zum Füllen aufgerufen? Gefüllt werden sie entweder durch den Calculator oder die Source. Verwendung von Zeigern, um keine Daten wiederholt zu kopieren.

6.4.5. PROGRAMMAUFBAU FÜR DIE VISUALISIERUNG

7. EVALUATION

Diskussion: was funktioniert/nicht

7.1. ERMITTLEMENT DER EREIGNISSE

keine Bestimmung der Güte der Heuristik bei der Ereigniserkennung

7.2. EINFLUSS DER PARAMETER DER BERECHNUNGSSCHRITTE

7.2.1. EINFLUSS DER NACHBARSCHAFTSSUCHE

FLÜSSIGKEITSPARTIKEL OHNE CLUSTER UND ERHÖHTES EREIGNISRAUSCHEN BEI ZU KLEINER NACHBARSCHAFTSSUCHE

Bei kleinen Radien (dreifacher Radius oder kleiner) ist das Volumen für die Nachbarschaftssuche so klein

Daher ist der in der Benutzeroberfläche zugelassene Minimalwert für den Radiusmultiplikator auf vier gesetzt. Ein weiterer Grund ist, dass ein sehr kleines Suchvolumen das Aufkommen winziger Cluster stark erhöht, da die Partikel der flüssigen Phase nicht dicht gepackt sein müssen und so nebeneinanderliegende Partikel nicht als solche erkannt werden. Damit ist es wahrscheinlicher, dass der *Fast-Depth-Algorithmus* [Abschnitt 5.5](#) sie unterschiedlichen Clustern zuordnet, weil räumlich benachbarte, aber bei der Nachbarschaftssuche nicht als Nachbarn erkannte Partikel bei der Suche nach dem nächsten, tiefer liegenden Partikel nicht beachtet werden. Eine erhöhte Clusteranzahl innerhalb von Zusammenhangskomponenten führt zu einer erhöhten Anzahl von detektierten Ereignissen. Wenn der Nutzer den Einfluss kleiner Cluster ignorieren möchte und sie als Rauschen statt als gewinnbringende Information betrachtet, so ist ein kleiner Radius bei der Nachbarschaftssuche von Nachteil.

GASPARTIKELCLUSTER BEI DER CLUSTERREDUKTION

Wenn andererseits der Radius zu groß gewählt wird (fünffacher Radius oder höher), können auch Partikel der Gasphase Nachbarn erhalten und werden im *Fast-Depth-Algorithmus* Clustern zugeordnet. Dies wiederum kann dazu führen, dass beim Reduzieren der kleinen Cluster [Abschnitt 5.6](#) diese Partikel in Clustern mit nur diesem einen Partikel zurückbleiben. Das liegt darin begründet, dass die Clusterreduktion begrenzte Iterationsschritte aufweist. Falls der von den umliegenden Cluster am Weitesten entfernte Partikel als erstes ausgewählt wird, weisen die umliegenden Partikel noch denselben Cluster auf. Wenn die Suche nach drei Nachbarschaftsebenen abbricht und die durchsuchten Partikel alle denselben Cluster aufweisen, verbleibt der Partikel

im aktuellen Cluster. Hingegen werden die umliegenden Partikel aufgrund der geringeren Entfernung zu den benachbarten, größeren Clustern in diese aufgenommen. So verbleibt der zuerst ausgewählte Partikel als einziger im Cluster. Das führt in der Ereigniserkennung [Abschnitt 5.8](#) zu falschen Birth und Death Erkennungen. Allerdings ist dieser Fehler nicht aufgetreten, es werden keine einzelnen Partikel oder kleine Partikelgruppen durch die Clusterbildung erzeugt. Das kann durch den Vergleich der Einpartikel- sowie der minimalen Cluster direkt nach der Clusterbildung mit deren Anzahl nach der Clusterreduktion überprüft werden. Da die Anzahl in allen Messungen gleich oder geringer ist, das Ausbleiben dieses Fehlers lässt sich auf die verwendete minimale Clustergröße von maximal zehn Partikeln zurückführen. Somit ist die dreifache Iteration der Clusterreduktion ausreichend. Hingegen wird mit höheren Mindestclustergrößen der Fehler verstärkt auftreten. Das kann leicht vermieden werden, indem die Anzahl der durchsuchten Nachbarschaftsebenen erhöht wird.

GASPARTIKEL IN CLUSTERN BEI GROSSEN RADIEN

Zur Beschleunigung des *Fast-Depth-Algorithmus'* [Abschnitt 5.5](#) werden alle Partikel, die beim Suchen entlang des Pfades getroffen wurden, ebenfalls zum Cluster hinzugefügt. Diese Beschleunigung kann Fehler verursachen, wenn der Radius bei der Nachbarschaftssuche [Abschnitt 5.4](#) groß gewählt wurde, da so auch Gaspartikel am Rand von Clustern mit hinzugefügt werden. Diese Fehler sind jedoch vernachlässigbar, da die Heuristik der Ereigniserkennung [Abschnitt 5.8](#) Mengen betrachtet und somit einzelne Gaspartikel in der Clustergruppe einen geringen Einfluss auf das Ergebnis haben.

NULLTIEFWERT EINPARTIKELCLUSTER PHÄNOMEN

Aufgrund der Abbruchbedingung der Traversierung über die Tiefenwerte kann es passieren, dass Partikel mit denselben Tiefenwerten in einzelne, separate Cluster eingeteilt werden, die alle nur diesen einen Partikel enthalten. Da Gaspartikel vom *Fast-Depth-Algorithmus* aussortiert werden und der Algorithmus für das Eintreten dieses Falls keine tieferen Nachbarn finden darf, müssen die Partikel einen Tiefenwert von null aufweisen. Damit nebeneinander angeordnete Partikel diesen Tiefenwert aufweisen können, muss der Cluster sehr klein sein. Die Überprüfung der Positionen der betroffenen Einpartikelcluster bestätigt diese Annahme. Dieses Phänomen tritt ab dreifachem Clusterradius in der Nachbarschaftssuche auf. Bei kleineren Radien werden die nebeneinanderliegenden Partikel mit einem Tiefenwert von null nicht als Nachbarn erkannt. Da diese Partikel somit nachbarlos sind, werden sie vom *Fast-Depth-Algorithmus* ignoriert und nicht in Cluster eingeteilt.

7.3. AUSWERTUNG DER VISUALISIERUNG

7.3.1. GLYPHDESIGN

- qualitative Umfrage Erkennung der Symbole: Tester (Vorstellung im Bekannten- und Freundekreis) haben gezeigt, dass sie den Symbolen die korrekte Semantik zuordnen, bei der abstrakten Vorstellung fiel die Zuordnung wesentlich ungenauer aus.

7.3.2. GLYPHDARSTELLUNG IM PARTIKELRAUM

- schlechte Sichtbarkeit bei gleichzeitiger Ansicht von Partikeln und Events, ClipPlane bei Simple-SphereRenderer hilft etwas

7.4. RESSOURCENVERBRAUCH

Zeit- und Ressourcenangaben beziehen sich auf folgendes System: - Intel Core i5-3210M - 2x4GB DDR3-1600 - Windows 8.1

Die Zeitmessungen konnte separat für jeden Schritt durchgeführt werden, da die Berechnungsschritte sequentiell durchgeführt werden.

Wie zu erkennen, ist die Nachbarschaftssuche sowohl der zeit- als auch der speicherintensivste Schritt, gefolgt von der Clusterbildung. Der Clustervergleich und die Erkennung der Strukturereignisse benötigen eine um fast zwei Größenordnungen geringere Zeit.

7.4.1. OPTIMIERUNG DURCH PARALLELISIERUNG

Die Nachbarschaftssuche ist der zeit- und speicherintensivste Schritt. Eine Parallelisierung der Schleifen für die Partikel wäre wünschenswert, da die Partikel voneinander unabhängig behandelt werden und jeder einzelne Schleifendurchgang rechenaufwändig ist, vor allem bei Beachtung der periodischen Randbedingungen und dem damit verbundenen, achtfachen Aufruf des Suchalgorithmus des kD-Baums. Allerdings haben Versuche mit OpenMP als auch mit der Parallel Patterns Library (PPL) gezeigt, dass ANN bei der Parallelisierung durch eines der beiden Verfahren trotz rein lesenden Zugriffs Zugriffsfehler aufweist, da jeder Thread auf denselben Baum und dessen Zeiger zugreift. Das Kopieren des Suchbaums für jeden Thread könnte eine mögliche Lösung sein, würde mit steigender Partikel- und Threadanzahl jedoch sehr viel Speicher verbrauchen. Der Blocker für die Parallelisierung mit der ANN Bibliothek ist allerdings, dass sie während der Programmlaufzeit für alle Suchstrukturen denselben Speicherbereich nutzt und ihn dadurch auch beim Löschen einzelner Suchstrukturen nicht deallokiert [Mou10, S. 8]. Somit ist ANN für eine parallele Nutzung nicht geeignet und ein Wechsel zu einer anderen Bibliothek ist empfehlenswert, wie beispielsweise zu FLANN [O+13] [Wij14].

Bei der Clusterbildung als zweitlängster Schritt kann die Verarbeitung für jeden Partikel parallelisiert werden. Allerdings wird die Beschleunigung nur gering oder sogar negativ ausfallen. Zum Einen müssen die Clustererstellung sowie die Clusterzuordnung des Wurzelpartikels threadsicher umgestaltet werden, zum Beispiel mittels Mutex-Verfahren, um eine mehrfache Clusterzuweisung auf denselben Partikel zu vermeiden. Das führt zu einer Verringerung der Parallelisierungseffizienz, da die Prozesse gegenseitig aufeinander warten müssen, bis die Erstellung und Zuordnung abgeschlossen ist. Zum Anderen müsste das Hinzufügen der traversierten Partikel zum Cluster entfernt werden, wodurch die auch in vorhergehenden Schritten getroffenen Partikel untersucht werden müssen, was vor allem bei langen Traversierungspfaden aufgrund der sequentiell ablaufenden Traversierung die Effizienz stärker senken kann, als die durch die Parallelisierung hinzugewonnene.

Die Clusterreduktion ist beim Durchlaufen der Partikel parallelisiert und dadurch kann auf dem Zweikernprozessor mit vier Threads des Testsystems in Stichproben eine Geschwindigkeitsverbesserung von über 15% festgestellt werden (Frame 75 5622 ms zu 6733 ms, Frame 76 5631 ms zu 6802 ms). Die geringe Skalierung ist damit zu erklären, dass nur ein Bruchteil der Partikel reduziert wird (in den Stichprobe der Frames 75 und 76 sind es 66 bzw. 55 Partikel, damit 0,03% aller Partikel). Die Berechnung der Winkel zu den benachbarten Clustern geschieht in einer Schleife innerhalb des Durchlaufes eines Partikels und ist noch einmal parallelisiert, was aufgrund der geringen Clusteranzahl nur in einem minimalen Geschwindigkeitsgewinn von unter 4% resultiert. Die Mutex in beiden Schleifen (Zählvariable und minimaler Winkel) haben keinen messbaren Einfluss auf die Geschwindigkeit. Die Parallelisierung der drei einzelnen Suchschritte nach den benachbarten Clustern ist nicht sinnvoll, da dort zum Ersten lediglich Vergleiche und Inkrementationen durchgeführt werden und zum Zweiten das Hinzufügen des ermittelten Wertes zum der die benachbarten Cluster enthaltenen Container nicht threadsicher ist und durch einen Mutex geschützt sein muss. Versuche haben gezeigt, dass der bei der Parallelisierung erzeugte Verwaltungsaufwand den Vorgang sogar verlangsamt. Bei Frame 75

wird ohne Parallelisierung der Clustersuche eine Dauer von 6791 ms und mit Parallelisierung eine Dauer von 8711 ms ermittelt, respektive bei Frame 76 6688 ms zu 8394 ms.

Beim Clustervergleich sowie der Ereigniserkennung wäre aufgrund des geringen Berechnungsaufwandes, des nötigen Mutex-Verfahren der Listen sowie der sequentiell zu erfolgenden Ausgabe eine Parallelisierung in der Geschwindigkeit kaum spürbar bzw. sogar nachteilig.

Falsch: In der Implementation wird so oft wie möglich der Parallelmodus von libstdc++ genutzt, um den Zugriff auf die Listen zu beschleunigen. Gelegentlich werden OpenMP Direktiven unmittelbar an den Schleifen verwendet.

8. ZUSAMMENFASSUNG

Kurzzusammenfassung/Abstract ausführlicher, auf Details eingehen

9. AUSBLICK

Verbesserung der Berechnungen:

- Clustergröße, die zur Bildung des Ereignisses geführt hat, im Ereignis abspeichern. So kann der Benutzer eine Gewichtung der Ereignisse abhängig von der Clustergröße vornehmen bzw. die Auswirkungen kleiner Cluster dynamisch ausblenden.
- Die Mindestclustergröße auf wesentlich höhere Werte, zum Beispiel größer 100 einstellen, um das Ereignisrauschen in Zusammenhangskomponenten zu senken. Voraussetzung dafür ist die Erhöhung der durchsuchten Nachbarschaftsebenen. Dies kann zum Beispiel durch Umwandlung des iterativen Vorgehens in ein Rekursives erreicht werden. Empfehlenswert ist die Skalierung der Rekursionsschritte mit der Mindestclustergröße, um die Berechnungsdauer der Reduktion nicht unnötig zu erhöhen. Da kleinere Cluster, die nur eine Zusammenhangskomponente bilden, bei der Reduktion erhalten bleiben, gibt es keinen Verlust dieser Ereignisinformationen. Die im Berechnungsmodul vorhandenen Ausgaben der berechneten, quantitativen Daten ermöglichen leicht eine vorangehende statistische Analyse für eine zielgerichtete Einstellung der Mindestclustergröße, etwa durch die Ausgabe der Anzahl von Clustern bestimmter Größen.
- Einbeziehung von sowohl der Forward- als auch der Backwardsliste bei der Ermittlung von Merge und Split, Vergleich der Durchschnittswerte der Verhältnisse der gemeinsamen Partikel der Elternpartner in der jeweiligen anderen Liste und eine daraus abgeleitete Gewichtung.
- Vergleich von Clustern über mehrere Zeitschritte. Dazu eine Signatur für jeden Cluster nutzen, die beispielsweise mit der Summe der Partikelidentifikatoren gebildet werden kann.
- Nutzung der Nachbarschaften von Clustern, wie mit dem Nachbarschaftsgraphen vorschlagen
- Geschwindigkeitserhöhung:
 - Nachbarschaftssuche: Nutzung von FLANN o.ä., die Parallelisierung nutzen. [O+13]
 - Parallelisierung bei Clusterbildung: Mehrere Partikel parallel verarbeiten. Dazu muss die Clusterliste eine vordefinierte Größe aufweisen, was bereits durch die Speicherplatzreservierung implementiert ist. Des Weiteren müssen die Sprünge in der Schleife durch Abfragen ersetzt werden, was einen geringen Einfluss auf die Geschwindigkeit hat. Jedoch muss die Clustererstellung sequentiell erfolgen, um eine mehrfache Clusterzuweisung auf denselben Partikel zu vermeiden, was zu einer Verringerung der Parallelisierungseffizienz führt, da die Prozesse gegenseitig aufeinander warten müssen, bis die Clusterzuordnung des Wurzelpartikels abgeschlossen ist. Weiterhin müsste das Hinzufügen der traversierten Partikel zum Cluster entfernt werden, was ebenfalls die Effizienz senkt.

- Performanceerhöhung beim Lesen (und Speicherplatzverringerung) MMSE Datenformat: Eventeinteilung nach Zeit in Unterlisten (so dass beim Auslesen direkt auf den Zeitschritt gesprungen werden kann) und dort auch Einteilung nach Typ.

Verbesserung der Visualisierung

- Bessere Juxtaposition von Glyphen und Partikeln ermöglichen: Senken der Partikelopazität, damit die sich hinter Partikeln befindlichen Glyphen leichter sichtbar werden.

Weitere Untersuchungsmöglichkeiten der Visualisierung:

- Nutzung von Animationen mit visuellen Bewegungsattributen (Bewegungsrichtung, Bewegungsbahn, Beschleunigung, Geschwindigkeit) sowie Veränderung der anderen Attribute; die Zuweisung der Bewegungsattribute wäre an die Parameter Zeit und Position denkbar, die Zuweisung der Veränderung der anderen Attribute kann an alle drei Parameter erfolgen
- Attribut *Position* als *Vokabular* bei örtlicher Häufung von Ereignissen
- Wirkung der Farbzuordnung untersuchen
- Elementattribute (Größe, Opazität) abhängig vom Zoom

Verbesserungen Plugin und MegaMol

- Calculation: Annahme von MPDC mit Clustereinfärbung, so dass die Nachbarschaftssuche und die Clusterberechnung wegfallen. Dadurch sind die Auswirkungen der Parameter für die Ereignisberechnung viel schneller sichtbar (wenige Sekunden)!
- Renderer: Ausgabe von quantitativen Ereigniswerten, analog zur Calculation.
- SimpleSphereRenderer: Filtern (Ausblenden) nach Partikelfarben.
- Globales Kennzeichen für die Logausgabe (vgl. Kennzeichen Calculation).
- WASDQERF Steuerung (siehe Mail)
- Hinzufügen eines Pfades für Texturen, indem zur megamol.cfg ein weiteres Element <texturedir> mit dem Attribut path hinzugefügt wird, analog zum Element <shaderdir>.
- Hinzufügen eines Pfades für Logdateien zur Ausgabe von quantitativen Daten, indem zur megamol.cfg ein weiteres Element <outputFiledir> mit dem Attribut path hinzugefügt wird, analog zum Element <shaderdir>.

Benutzung von flexiblen Isooberflächen nach [CSP10].

A. ANHANG

A.1. BERECHNUNGEN MOCKUPS

WERTE VERALTET BZW. ZEIGEN DER TRIVIALEN BERECHNUNG SINNLOS/KANN GELÖSCHT WERDEN

In [Gleichung A.1](#) ist die Berechnung des Farbwerts h in linearer Abhängigkeit von der Zeit t zu finden.

$$\begin{aligned} h[^\circ] &= m \cdot t + n \\ h_{\max} &= m \cdot t_{\max} + n && \text{mit } h_{\max} = 320, t_{\max} = 23 \\ h_{\min} &= m \cdot t_{\min} + n && \text{mit } h_{\min} = 0, t_{\min} = 1 \\ n &= -14, \overline{54} \\ m &= 14, \overline{54} \\ h[^\circ] &= \boxed{[14, \overline{54} \cdot t + 14, \overline{54} + 0, 5]} \end{aligned} \tag{A.1}$$

In [Gleichung A.2](#) wird die Berechnung der Helligkeit b in linearer Abhängigkeit von der Zeit t gezeigt.

$$\begin{aligned} b[%] &= m \cdot t + n \\ b_{\max} &= m \cdot t_{\max} + n && \text{mit } b_{\max} = 100, t_{\max} = 23 \\ b_{\min} &= m \cdot t_{\min} + n && \text{mit } b_{\min} = 40, t_{\min} = 1 \\ n &= 37, \overline{27} \\ m &= 2, \overline{72} \\ b[%] &= \boxed{[2, \overline{72} \cdot t + 37, \overline{27} + 0, 5]} \end{aligned} \tag{A.2}$$

In [Gleichung A.3](#) ist die Berechnung der Durchmesser d in linearer Abhängigkeit von der Zeit t zu finden.

$$\begin{aligned} d[px] &= m \cdot t + n \\ d_{\max} &= m \cdot t_{\max} + n && \text{mit } d_{\max} = 75, t_{\max} = 23 \\ d_{\min} &= m \cdot t_{\min} + n && \text{mit } d_{\min} = 25, t_{\min} = 1 \\ n &= 22, \overline{72} \\ m &= 2, \overline{27} \\ d[px] &= \boxed{[2, \overline{27} \cdot t + 22, \overline{72} + 0, 5]} \end{aligned} \tag{A.3}$$

Tabelle A.1. Die Beispieldatensätze für die Mockups. Die x-Koordinate des Ortes ist mit dem Zeitpunkt t gekoppelt über $x = \lfloor \frac{4}{3} \cdot t + 0,5 \rfloor$. Die y, z-Koordinaten sind, bis auf wenige Ausnahmen zur Demonstration der Häufung, zufällig verteilt im Intervall [1, 10].

Datensatznummer	Zeitpunkt t	Ort (x, y, z)	Art
1	1	1, 8, 4	Split
2	1	1, 4, 3	Split
3	1	1, 1, 8	Split
4	1	1, 4, 1	Split
5	1	1, 4, 9	Split
6	1	1, 4, 4	Split
7	1	1, 4, 4	Merge
8	1	1, 4, 4	Birth
9	1	1, 4, 4	Death
10	1	1, 4, 6	Split
11	2	3, 4, 1	Split
12	2	3, 8, 7	Birth
13	2	3, 7, 8	Split
14	2	3, 6, 1	Split
15	2	3, 1, 2	Merge
16	2	3, 5, 8	Split
17	3	4, 3, 1	Merge
18	3	4, 10, 10	Merge
19	3	4, 5, 4	Merge
20	3	4, 3, 1	Merge
21	4	5, 5, 7	Merge
22	4	5, 3, 1	Split
23	4	5, 3, 10	Merge
24	4	5, 1, 1	Split
25	5	7, 9, 1	Merge
26	5	7, 4, 2	Merge
27	6	8, 3, 9	Split
28	6	8, 10, 10	Death
29	6	8, 1, 5	Split
30	6	8, 8, 1	Death
31	7	9, 3, 5	Death
32	7	9, 6, 4	Death
33	8	11, 3, 8	Split
34	8	11, 3, 2	Split
35	10	13, 2, 7	Merge
36	10	13, 2, 7	Split
37	10	13, 10, 5	Merge
38	10	13, 2, 2	Split
39	13	17, 8, 1	Merge
40	13	17, 1, 8	Merge
41	15	20, 5, 3	Split
42	15	20, 7, 6	Split
43	16	21, 7, 3	Death
44	16	21, 6, 6	Merge
45	18	24, 8, 4	Merge
46	20	27, 6, 5	Merge
47	20	27, 6, 5	Merge
48	20 ³⁸	27, 6, 5	Death
49	21	28, 5, 4	Merge
50	21	28, 5, 3	Merge

Tabelle A.2. Festlegung der Attributeigenschaften für den Parameter *Zeit* in den Mockups.
VERALTET

Zeit	Farbwert [°]	Helligkeit [%]	Anzahl Ecken	Größe [px]
1	0	40	1	25
2	15	43	2	27
3	29	45	3	30
4	44	48	4	32
5	58	51	5	34
6	73	54	6	36
7	87	56	7	39
8	102	59	8	41
10	131	65	10	45
13	175	73	13	52
15	204	78	15	57
16	218	81	16	59
20	276	92	20	68
21	291	95	21	70
23	320	100	23	75

Analog erfolgen die Berechnungen für den Parameter *Ort x* mit den Grenzwerten

$$\begin{aligned}x_{\min} &= 1 \\x_{\max} &= 31\end{aligned}\tag{A.4}$$

Das Ergebnis ist in [Tabelle A.3](#) zu finden.

A.2. WERKZEUGE UND BIBLIOTHEKEN

MOCKUP VISUALISIERUNG

Adobe Illustrator
Blender mit Regular Solids
Unity3D mit [UIComboBox](#), Simple FlyCamera

DOKUMENTATION

Adobe Photoshop
Blender mit Extra Objects
Fraps Aufnahme der visuellen MegaMol Ausgaben
IrfanView Bildumwandlung
TeXstudio, TeXlive mit tudscr, glossaries, tikz u.a.

MEGAMOL PLUGIN

ActivePerl
Adobe Illustrator zur Gestaltung der Glyphen
Adobe Photoshop zur Erstellung der Texturen
glm
lodepng zum Laden von Texturen
MegaMol mit ANN, AntTweakBar, mmstd_datatools, vislib
MegaMolConf

Tabelle A.3. Festlegung der Attributeigenschaften für den Parameter *Ort x* in den Mockups.
VERALTET

Ort x	Farbwert [°]	Helligkeit [%]	Anzahl Ecken	Größe [px]
1	0	40	1	25
3	21	44	3	28
4	32	46	4	30
5	43	48	5	32
7	64	52	7	35
8	75	54	8	37
9	85	56	9	38
11	107	60	11	42
13	128	64	13	45
17	171	72	17	52
20	203	78	20	57
21	213	80	21	58
27	277	92	27	68
28	288	94	28	70
31	320	100	31	75

VisualStudio 2013 mit OpenMP

LITERATUR

- [All+15] Andy Allan u. a. *Contours*. Englisch. 2015. URL: <http://wiki.openstreetmap.org/wiki/Contours> (besucht am 23.05.2015).
- [Au+08] Oscar Kin-Chung Au u. a. *Skeleton Extraction by Mesh Contraction*. Englisch. 2008. URL: <http://www.math.tau.ac.il/~dcor/articles/2008/Skeleton-Extraction.pdf> (besucht am 10.02.2015).
- [Bai04] Rudolf H. Baierl. *Konvexe Polyeder: Klassische und hybride numerische Generierungsmethoden*. Beuth Hochschule für Technik Berlin. 2004. URL: http://public.beuth-hochschule.de/~baierl/inhalt/polyhed/p_konst000530.pdf (besucht am 06.03.2015).
- [Blu67] Harry Blum. „A Transformation for Extracting New Descriptors of Shape“. In: *Models for the Perception of Speech and Visual Form*. Hrsg. von Weiant Wathen-Dunn. Cambridge: MIT Press, 1967, S. 362–380.
- [Bog15] Alexander Bogomolny. *Regular Polyhedra*. Englisch. Interactive Mathematics Miscellany und Puzzles. 2015. URL: http://www.cut-the-knot.org/do_you_know/polyhedra.shtml (besucht am 06.03.2015).
- [BPS97] Chandrajit L Bajaj, Valerio Pascucci und Daniel R Schikore. „The contour spectrum“. In: *Proceedings of the 8th conference on Visualization'97*. IEEE Computer Society Press. 1997, 167–ff.
- [Chi+05] Yi-Jen Chiang u. a. „Simple and optimal output-sensitive construction of contour trees using monotone paths“. In: *Computational Geometry* 30.2 (2005). Special Issue on the 19th European Workshop on Computational Geometry, S. 165–195. ISSN: 0925-7721. DOI: <http://dx.doi.org/10.1016/j.comgeo.2004.05.002>. URL: <http://www.sciencedirect.com/science/article/pii/S0925772104000811>.
- [CLM54] H. S. M. Coxeter, M. S. Longuet-Higgins und J. C. P. Miller. „Uniform Polyhedra“. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 246.916 (1954), S. 401–450. ISSN: 0080-4614. DOI: [10.1098/rsta.1954.0003](https://doi.org/10.1098/rsta.1954.0003).
- [Cor+05] Nicu D Cornea u. a. „Computing hierarchical curve-skeletons of 3D objects“. In: *The Visual Computer* 21.11 (2005), S. 945–955.
- [CS09] Hamish Carr und Jack Snoeyink. „Representing Interpolant Topology for Contour Tree Computation“. English. In: *Topology-Based Methods in Visualization II*. Hrsg. von Hans-Christian Hege, Konrad Polthier und Gerik Scheuermann. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, S. 59–73. ISBN: 978-3-540-88605-1.

- DOI: 10.1007/978-3-540-88606-8_5. URL: http://dx.doi.org/10.1007/978-3-540-88606-8_5.
- [CSA01a] Hamish Carr, Jack Snoeyink und Ulrike Axen. *Computing Contour Trees in All Dimensions*. Englisch. 2001. URL: <http://www.csi.ucd.ie/staff/hcarr/home/research/publications/contourtreealgorithm.pdf> (besucht am 16.03.2015).
 - [CSA01b] Hamish Carr, Jack Snoeyink und Ulrike Axen. *Efficient computation of Contour Trees*. Englisch. 2001. URL: <http://www.csi.ucd.ie/staff/hcarr/home/research/contourtrees/contourtrees.html> (besucht am 16.03.2015).
 - [CSM07] Nicu D Cornea, Deborah Silver und Patrick Min. „Curve-skeleton properties, applications, and algorithms“. In: *Visualization and Computer Graphics, IEEE Transactions on* 13.3 (2007), S. 530–548.
 - [CSP10] Hamish Carr, Jack Snoeyink und Michiel van de Panne. „Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree“. In: *Computational Geometry* 43.1 (2010). Special Issue on the 14th Annual Fall Workshop, S. 42–58. ISSN: 0925-7721. DOI: <http://dx.doi.org/10.1016/j.comgeo.2006.05.009>. URL: <http://www.sciencedirect.com/science/article/pii/S0925772109000455>.
 - [DS06] Tamal K. Dey und Jian Sun. „Defining and Computing Curve-skeletons with Medial Geodesic Function“. In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, S. 143–152. ISBN: 3-905673-36-3. URL: <http://dl.acm.org/citation.cfm?id=1281957.1281975>.
 - [Ede+04] Herbert Edelsbrunner u. a. „Time-varying reeb graphs for continuous space-time data“. In: *Proceedings of the twentieth annual symposium on Computational geometry*. ACM. 2004, S. 366–372.
 - [Gra15] Sam Gratrix. *WebGL Uniform Polyhedra*. Englisch. 2015. URL: <http://gratrix.net/polyhedra/webgl/> (besucht am 06.03.2015).
 - [Har08] George W. Hart. *Procedural Generation of Sculptural Forms*. Englisch. Computer Science Department, Stony Brook University. 2008. URL: <http://www.georgehart.com/ProceduralGeneration/Bridges08-Hart10pages.pdf> (besucht am 06.03.2015).
 - [HF05] M Sabry Hassouna und Aly A Farag. „Robust centerline extraction framework using level sets“. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Bd. 1. IEEE. 2005, S. 458–465.
 - [Hop96] Edward J. Hopkins. *Surface Weather Analysis Chart*. Englisch. 1996. URL: <http://www.meteor.wisc.edu/~hopkins/aos100/sfc-anl.htm> (besucht am 20.05.2015).
 - [Hur10] Lorenz Hurni. „Cartographic Relief Presentation Revisited - Forty Years after Eduard Imhof“. English. In: *Landform - Structure, Evolution, Process Control*. Hrsg. von Jan-Christoph Otto und Richard Dikau. Bd. 115. Lecture Notes in Earth Sciences. Springer Berlin Heidelberg, 2010, S. 1–20. ISBN: 978-3-540-75760-3. DOI: 10.1007/978-3-540-75761-0_1. URL: http://dx.doi.org/10.1007/978-3-540-75761-0_1.
 - [Joh66] Norman W. Johnson. „Convex Solids with Regular Faces“. In: *Canadian Journal of Mathematics* 18 (1966), S. 169–200. ISSN: 0008-414x.
 - [Kre+97] Marc van Kreveld u. a. „Contour Trees and Small Seed Sets for Isosurface Traversal“. In: *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*. SCG '97. Nice, France: ACM, 1997, S. 212–220. ISBN: 0-89791-878-9. DOI: <10.1145/262839.269238>. URL: <http://doi.acm.org/10.1145/262839.269238>.
 - [KT03] Sagi Katz und Ayellet Tal. *Hierarchical mesh decomposition using fuzzy clustering and cuts*. Bd. 22. 3. ACM, 2003.

- [KWB08] K. Klasing, D. Wollherr und M. Buss. „A clustering method for efficient segmentation of 3D laser data“. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. Mai 2008, S. 4043–4048. DOI: [10.1109/ROBOT.2008.4543832](https://doi.org/10.1109/ROBOT.2008.4543832).
- [Lan+06] D. Laney u. a. „Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities“. In: *Visualization and Computer Graphics, IEEE Transactions on* 12.5 (Sep. 2006), S. 1053–1060. ISSN: 1077-2626. DOI: [10.1109/TVCG.2006.186](https://doi.org/10.1109/TVCG.2006.186).
- [Mil63] John Willard Milnor. *Morse theory*. 51. Princeton university press, 1963.
- [Mou10] David M. Mount. *ANN Programming Manual Version 1.1*. Englisch. University of Maryland. 2010. URL: http://cs.umd.edu/~mount/ANN/Files/1.1.2/ANNmanual_1.1.pdf (besucht am 18.07.2015).
- [MWL02] Cherng-Min Ma, Shu-Yen Wan und Jiann-Der Lee. „Three-dimensional topology preserving reduction on the 4-subfields“. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.12 (Dez. 2002), S. 1594–1605. ISSN: 0162-8828. DOI: [10.1109/TPAMI.2002.1114851](https://doi.org/10.1109/TPAMI.2002.1114851).
- [O+13] Stephen O’Hara, Bruce Draper u. a. „Are you using the right approximate nearest neighbor algorithm?“ In: *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*. IEEE. 2013, S. 9–14.
- [Pal08] Kálmán Palágyi. „A 3D fully parallel surface-thinning algorithm“. In: *Theoretical Computer Science* 406.1–2 (2008). Discrete Tomography and Digital Geometry: In memory of Attila Kuba, S. 119–135. ISSN: 0304-3975. DOI: <http://dx.doi.org/10.1016/j.tcs.2008.06.041>. URL: <http://www.sciencedirect.com/science/article/pii/S0304397508004350>.
- [Pas+07] Valerio Pascucci u. a. „Robust on-line computation of Reeb graphs: simplicity and speed“. In: *ACM Transactions on Graphics (TOG)*. Bd. 26. 3. ACM. 2007, S. 58.
- [PCS04] Valerio Pascucci, Kree Cole-McLaughlin und Giorgio Scorzelli. „Multi-resolution computation and presentation of contour trees“. In: *IASTED Conference on Visualization, Imaging, and Image Processing*. Citeseer. 2004. URL: http://www.pascucci.org/topology/contour_tree/.
- [Ray94] Eric Raymond. *Criticism of C-FAQ submission invited*. Englisch. comp.lang.c. 1994. URL: <http://c-faq.com/struct/align.esr.html> (besucht am 02.06.2015).
- [SB06] B.-S. Sohn und C. Bajaj. „Time-varying contour topology“. In: *Visualization and Computer Graphics, IEEE Transactions on* 12.1 (Jan. 2006), S. 14–25. ISSN: 1077-2626. DOI: [10.1109/TVCG.2006.16](https://doi.org/10.1109/TVCG.2006.16).
- [Sha+07] Andrei Sharf u. a. „On-the-fly Curve-skeleton Computation for 3D Shapes“. In: *Computer Graphics Forum*. Bd. 26. 3. Wiley Online Library. 2007, S. 323–328.
- [SKK91] Yoshihisa Shinagawa, Tosiyasu L Kunii und Yannick L Kergosien. „Surface coding based on Morse theory“. In: *IEEE Computer Graphics and Applications* 5 (1991), S. 66–78.
- [SW97] D. Silver und X. Wang. „Tracking and visualizing turbulent 3D features“. In: *Visualization and Computer Graphics, IEEE Transactions on* 3.2 (Apr. 1997), S. 129–141. ISSN: 1077-2626. DOI: [10.1109/2945.597796](https://doi.org/10.1109/2945.597796).
- [Szy05] A. Szymczak. „Subdomain aware contour trees and contour evolution in time-dependent scalar fields“. In: *Shape Modeling and Applications, 2005 International Conference*. Juni 2005, S. 136–144. DOI: [10.1109/SMI.2005.45](https://doi.org/10.1109/SMI.2005.45).
- [Tag+12] Andrea Tagliasacchi u. a. „Mean curvature skeletons“. In: *Computer Graphics Forum*. Bd. 31. 5. Wiley Online Library. 2012, S. 1735–1744.

- [Tan+14] San Tang u. a. „Segmentation and 3D visualization of pheochromocytoma in contrast-enhanced CT images“. In: *Audio, Language and Image Processing (ICALIP), 2014 International Conference on*. Juli 2014, S. 39–43. DOI: [10.1109/ICALIP.2014.7009753](https://doi.org/10.1109/ICALIP.2014.7009753).
- [Tre+13] Alexander JB Trevor u. a. „Efficient organized point cloud segmentation with connected components“. In: *Semantic Perception Mapping and Exploration (SPME)* (2013).
- [TZC09] Andrea Tagliasacchi, Hao Zhang und Daniel Cohen-Or. „Curve skeleton extraction from incomplete point cloud“. In: *ACM Transactions on Graphics (TOG)*. Bd. 28. 3. ACM. 2009, S. 71.
- [Web+07] Ofir Weber u. a. „Context-Aware Skeletal Shape Deformation“. In: *Computer Graphics Forum*. Bd. 26. 3. Wiley Online Library. 2007, S. 265–274.
- [Wei15] Eric W. Weisstein. *Kepler-Poinsot Solid*. Englisch. MathWorld—A Wolfram Web Resource. 2015. URL: <http://mathworld.wolfram.com/Kepler-PoinsotSolid.html> (besucht am 06. 03. 2015).
- [Wij14] Maheshakya Wijewardena. *Performance Evaluation of Approximate Nearest Neighbour Search Implementations*. Englisch. University of Moratuwa. 2014. URL: <http://maheshakya.github.io/gsoc/2014/06/14/performance-evaluation-of-approximate-nearest-neighbor-search-implementations-part-2.html> (besucht am 15. 06. 2015).
- [WL08] Yu-Shuen Wang und Tong-Yee Lee. „Curve-Skeleton Extraction Using Iterative Least Squares Optimization“. In: *Visualization and Computer Graphics, IEEE Transactions on* 14.4 (Juli 2008), S. 926–936. ISSN: 1077-2626. DOI: [10.1109/TVCG.2008.38](https://doi.org/10.1109/TVCG.2008.38).
- [WPP07] Robert Y Wang, Kari Pulli und Jovan Popović. „Real-time enveloping with rotational regression“. In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), S. 73.

INDEX

Agglomeration, 23, 27
archimedischer Körper, 21
Art, 26, 27

Billboard, 7
Blender, 27

catalanischer Körper, 21
Clustergröße, 17
Clustervergleichsmatrix, 16

Farbe, 22, 27
Fast-Depth-Algorithmus, 6, 15, 30, 31
Form, 22, 26, 27

Gesetz der Ähnlichkeit, 22
Gesetz der fortgesetzt durchgehenden Linie, 24
Gesetz der gemeinsamen Bewegung, 22
Gesetz der gemeinsamen Region, 22
Gesetz der Geschlossenheit, 22, 24
Gesetz der Gleichzeitigkeit, 22
Gesetz der Kontinuität, 22
Gesetz der Nähe, 22
Gesetz der Prägnanz, 22
Gesetz der verbundenen Elemente, 24, 25
Größe, 22, 26
Häufung, 23–25, 27, 28

Isolinie, 10
Isooberfläche, 10, 11, 16
Isowert, 10, 11

Johnson-Körper, 21
Kepler-Poinsot-Körper, 21
Kontur, 9, 10, 16
Konturbaum, 9

MegalMol, 24
MegaMol, 23, 24, 26
Opazität, 24, 25, 27, 28
Ort, 24–26, 39, 40

Partnercluster, 17
platonischer Körper, 21, 27
Position, 22–25, 36

Saatpartikel, 14
Tiefe, 14
Tiefenwert, 6, 7, 13–15
Unity3D, 25, 27
Vokabular, 21, 22, 36
Zahlenwert, 21, 22
Zeit, 25, 39