TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät für Informatik Institut für Software- und Multimediatechnik

Professur Computergraphik und Visualisierung

Bachelorarbeit

VISUALISIERUNG VON STRUKTURVERÄNDERUNGEN IN MOLEKULARDYNAMIKDATEN

Richard Hähne

Geboren am: 10. Februar 1982 in Dresden
Matrikelnummer: 2873574
Immatrikulationsjahr: 2011

BACHELOR-ARBEIT

zur Erlangung des akademischen Grades

BACHELOR OF SCIENCE (B.SC.)

Betreuer
Sebastian Grottel
Ludwig Schmutzler

Betreuender Hochschullehrer
Prof. Dr. Stefan Gumhold

Eingereicht am: 10. August 2015

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelor-Arbeit 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, 10. August 2015

Richard Hähne

INHALTSVERZEICHNIS

Akronyme 5

1. Aufgabenstellung 6

2. Einleitung 7

3. Verwandte Arbeiten 10
   - 3.1. Skelettextraktion . . . 10
   - 3.2. Zusammenhangskomponenten und Konturbäume . . . 11
   - 3.3. Visualisierung großer Datenobjekte . . . 14

4. Überblick 17
   - 4.1. Beschreibung der Simulation . . . 17

5. Erkennung der Cluster und Strukturereignisse 19
   - 5.1. Datenbehandlung . . . 19
   - 5.2. Schritt 1: Bestimmung der nächsten Nachbarn . . . 21
   - 5.3. Schritt 2a: Cluster-Fast-Depth Algorithmus zur Bestimmung der Cluster . . . 22
   - 5.4. Schritt 2b: Reduzierung der kleinen Cluster . . . 25
   - 5.5. Schritt 3: Bestimmung der Clusterzusammenhänge . . . 26
   - 5.6. Schritt 4: Erkennung von Ereignissen . . . 27
   - 5.7. Protokollierung und Datenspeicherung . . . 29

6. Visualisierung 31
   - 6.1. Grundlagen . . . 31
   - 6.2. Strukturereignistaxonomie . . . 33
   - 6.3. Gestaltung der Glyphen . . . 36
   - 6.4. Renderer und Shader . . . 40
   - 6.5. Visualisierung der Cluster . . . 43
   - 6.6. Darstellung des Zeitverlaufs . . . 43

7. Evaluation 44
   - 7.1. Clusterbildung . . . 44
   - 7.2. Quantitative Analyse der Ereignisermittlung . . . 47
   - 7.3. Qualitative Analyse der Ereignisermittlung . . . 49
   - 7.4. Visualisierung der Cluster und Ereignisse . . . 52
   - 7.5. Ressourcenverbrauch . . . 57

8. Diskussion der Ergebnisse 60
8.1. Einfluss der Nachbarschaftssuche . . . 60
8.2. Farbgebung der Glyphen . . . 62
8.3. Verdeckung der Glyphen . . . 62
8.4. Optimierung des Ressourcenverbrauchs durch Parallelisierung . . . 62
8.5. Optimierung der Algorithmen . . . 63
8.6. Nützlichkeit von Visualisierungen . . . 64

9. Zusammenfassung 65

10. Ausblick 67

A. Anhang 69
A.1. Aufbau des Plugins . . . 69
A.2. Aufzählung der Cluster- und Ereigniserkennungsschritte . . . 72
A.3. Glyphprototyp . . . 72
A.4. Umfrage . . . 75
A.5. Berechnungen Glyph . . . 75
A.6. Verwendete Hardware . . . 76
A.7. Werkzeuge und Bibliotheken . . . 76
A.8. Alternative Bestimmung der Cluster ohne Nachbarschaftskenntnisse . . . 77

Literatur 78

AKRONYME

ANN	Approximate Nearest Neighbor Bibliothek	20, 61, 70, 74
CFD	Cluster Fast Depth Algorithmus	6, 17, 21, 22, 25, 26, 42, 48, 57–61, 63, 65
CSV	Comma Separated Values	27
DSE	Determine Structure Events	45, 56
FLANN	Fast Library for Approximate Nearest Neighbors	61
GiB	Gibibyte	55, 57, 67, 74
glm	OpenGL Mathematics	38, 74
HSB	Hue-Saturation-Brightness	71
HSV	Hue-Saturation-Value	73
kiB	Kibibyte	57
MIT	MIT Lizenz	16
MMPLD	MegaMol Multi Particle List Dump	16, 21, 27, 57, 67–69, 71
MMSE	MegaMol Structure Event	27, 28, 67, 69
MPDC	Multi Particle Data Call	21, 65, 68–70
ms	Millisekunden	29, 61
OGL	OpenGL	38, 74
PNG	Portable Network Graphics	38
PPL	Parallel Patterns Library	61
RGB	Rot-Grün-Blau	67
SECalc	Structure Events Calculation Modul	65, 68–70
SECC	Structure Events Cluster Compare	7, 25, 26, 42, 56, 57, 62, 63, 66, 70
SEDC	Structure Events Data Call	68
SPH	Smoothed-particle hydrodynamics	62
XML	Extensible Markup Language	16, 27, 67

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, können sie als Zusammenhangskomponente betrachtet werden.

Molekulardynamische Berechnungen ermöglichen die Vorhersage von Materialverhalten, etwa unter bestimmten äußeren Bedingungen wie beispielsweise thermische Einflüsse. 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 und einer begrenzten Zeitspanne 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 (siehe Abschnitt 4.1). Dabei verändert sich die Struktur der Flüssigkeit und diese Veränderungen sollen verfolgt werden, um daraus Struktureereignisse abzuleiten. Die Erkennung von Struktureereignissen kann dabei helfen, die Abhängigkeit des Materialverhaltens von äußeren Faktoren sowie des Materials selbst besser einschätzen zu können. Die Struktureereignisse werden, wie in Abbildung 2.1 zu sehen, in vier Kategorien einsortiert. *Birth* beschreibt dabei die Bildung von Agglomerationen aus Gaspartikeln, *Death* dagegen ihr Verschwinden, also der Übergang all ihrer Teilchen in die Gasphase. *Merge* steht für die Verschmelzung von Agglomerationen, *Split* für ihre Trennung.

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. Naiv betrachtet könnte so jede Zusammenhangskomponente eine Gruppe bilden. Da jedoch auch Struktureereignisse innerhalb dieser Komponenten erkannt werden sollen, erfolgt eine wesentlich differenziertere 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 Cluster Fast Depth Algorithmus (CFD) entwickelt wurde. Er ermöglicht eine reproduzierbare Einteilung der Partikel innerhalb von Zusammenhangskomponenten, erzeugt über mehrere Zeitschritte jedoch ein *Ereignisrauschen*, da Änderungen in der Partikelstruktur eine andere Clusterbildung zur Folge

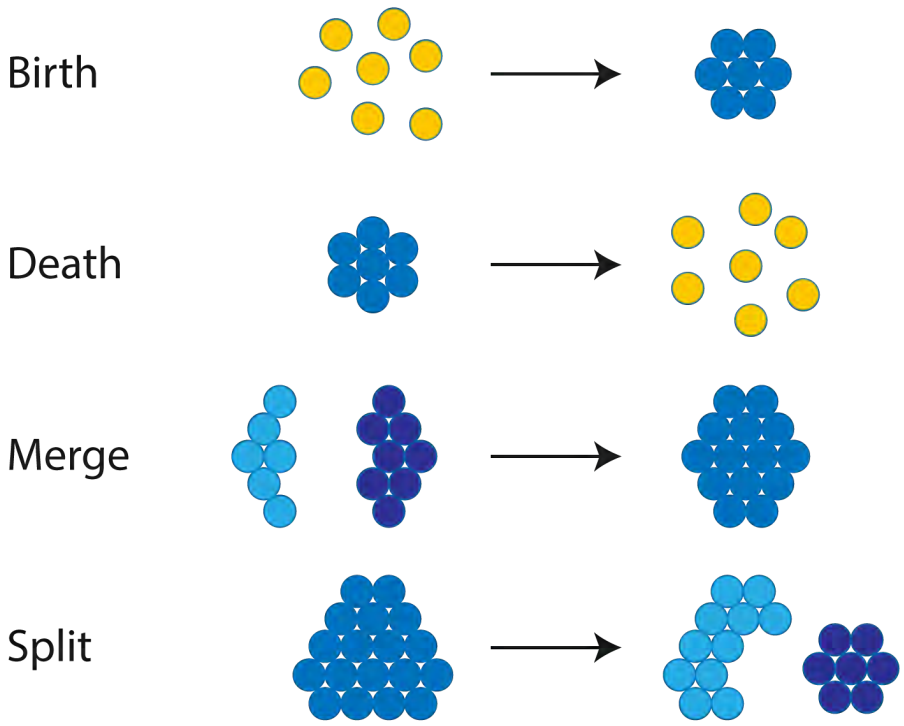


Abbildung 2.1. Die vier Arten der Strukturereignisse: *Birth*, *Death*, *Merge* und *Split*. Die gefüllten Kreise symbolisieren Partikel. Die gelbe Füllung steht für die Gasphase, die blaue für die Flüssigkeitsphase.

haben.

Der zweite und dritte Schwerpunkt dieser Arbeit liegen auf der Erkennung der Ereignisse sowie deren Visualisierung im Raum der Partikel. Die Erkennung erfolgt auf Basis eines Vergleichs der gebildeten Cluster über zwei Zeitschritte im hier entwickelten Structure Events Cluster Compare (SECC) und nutzt eine heuristische Auswertung der so entstandenen Daten. 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 (siehe Abbildung 2.2), soll 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 dreidimensionale 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 stellte 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. Auch musste eine eigene Methode entwickelt werden, um die Aufteilung der Zusammenhangskomponenten in einzelne Cluster vorzunehmen. Weiterhin muss 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

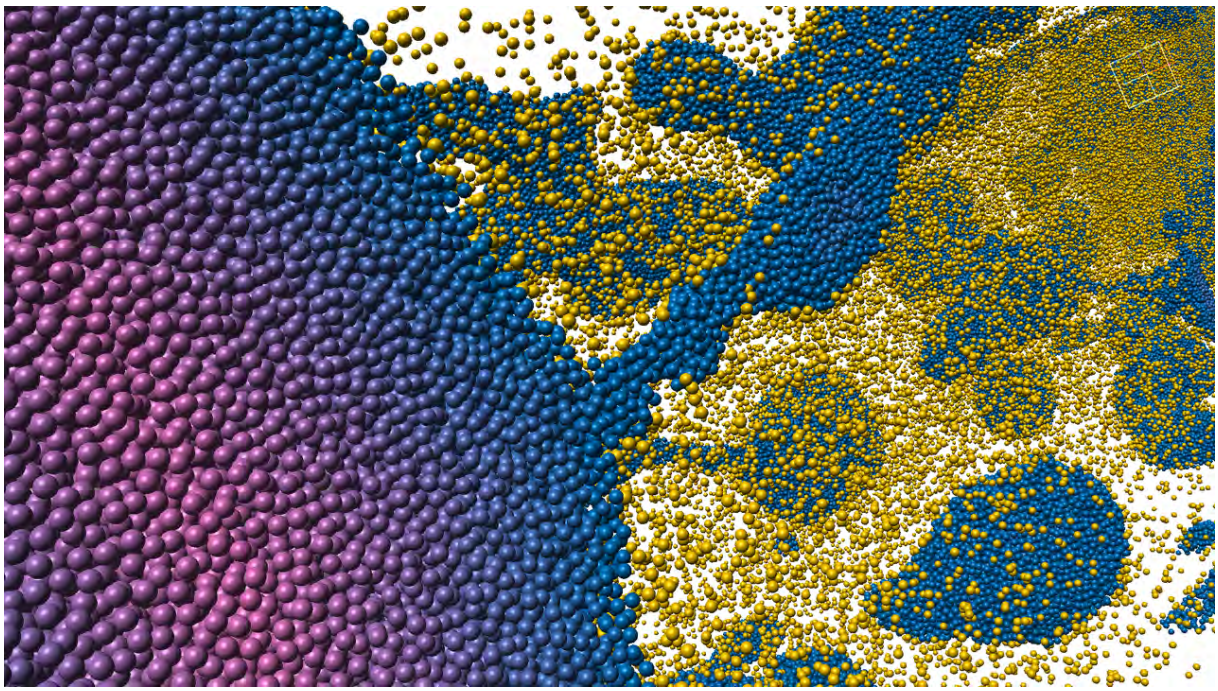


Abbildung 2.2. Kugelförmige Partikeldarstellung im Simulationsraum. Gelbe Partikel befinden sich in der Gasphase, blau-rote in der Flüssigkeitsphase.

erkennbar sind.

Dabei soll die Implementation eine akzeptable Berechnungszeit besitzen, um ihre Parameter für die Evaluation justieren zu können. Akzeptabel bedeutet, dass die Berechnungen der Cluster und der Ereignisse für einen Zeitpunkt des zwei Millionen Partikel großen Datensatzes innerhalb von Minuten und nicht von Stunden abgeschlossen sein müssen. Es wurde geprüft, inwieweit Erkenntnisse aus der Skelettextraktion, der Behandlung von Laserscandaten sowie die Nutzung von *Konturbäumen* bei der Entwicklung helfen können.

Zur Prüfung der Skalierbarkeit der eingesetzten Algorithmen wurden quantitative Messungen mit verschiedenen Parametern verglichen. Weiterhin wurde die Qualität der Ereignisermittlung durch einen visuellen Vergleich mit den Partikeldaten durchgeführt. Die der Visualisierung zugrundeliegenden Thesen hinsichtlich der Glyphgestaltung sind durch eine Umfrage geprüft worden (Beschreibung der Umfrage siehe **??**), die durch wenig Erklärungen des Sachverhaltes eine unvoreingenommene Einstellung der Teilnehmer sicherstellt.

3. VERWANDTE ARBEITEN

Für die Zuordnung von Teilchen zu Agglomerationen beziehungsweise Zusammenhangskomponenten ist das Wissen über deren räumliche Begrenzung Voraussetzung, was Kenntnisse über den Oberflächenverlauf und das Volumen der Strukturen erfordert. Die Volumen- und Oberflächenbestimmung von großen Datensätzen wurde in Arbeiten über Skelettextraktionen untersucht, die zum Beispiel die durch Laserscanverfahren ermittelte Daten auswerten. Weiterhin existieren Datenstrukturen, um räumliche Datensätze beliebiger Dimensionen anhand ihrer Funktionswerte in Zusammenhangskomponenten einzuteilen. Jene werden als *Konturbäume* bezeichnet.

Anschließend wird auf die Visualisierung großer Datenmengen 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 Skeletttypen wie zum Beispiel 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 zum Beispiel Schwarz-Weiß Bilder) durch schrittweises Entfernen der äußeren Datenpunkte [Pal08], die Berechnung mittels mittleren Krümmungsflusses, wobei 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 Auflistung ist nur ein Auszug von üblichen Methoden. Für Kurvenskelette [CSM07] gibt es 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 sich somit ihr Einsatzgebiet vor allem in Echtzeitanwendungen befindet. 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 in dieser Arbeit vorliegenden Partikeldatensatzes zu Agglomerationen dienen jene in der Skelettextraktion verwendeten Methoden als Ideengeber. Sie können jedoch nicht direkt implementiert 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 Verlagerungen von Partikelmengen zwischen diesen Gruppen im Verlauf der Zeit erfasst werden können. Die 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 5.2).

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 identifizierenden Strukturereignissen. Sie sind für beliebige Dimensionen geeignet, setzen jedoch wie die Skelettextraktionsverfahren 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 Entstehen 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 liegt ein kritischer Funktionswert vor, wenn der *Isowert* durch den Funktionswert verläuft und sich dabei die Topologie der Niveaumenge ändert. 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, im Skalarfeld enthaltenen, 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, sich teilen, sich verbinden oder verschwinden. Geschlechtsänderungen werden nicht beachtet. Die Kanten bilden äquivalente *Konturen*. Dabei entstehe qn ein Verbindungs- und ein Teilungsbaum [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 dargestellte Methodik als Abtastalgorithmus bezeichnet [CSA01a] [Chi+05, S. 176]. In Abbildung 3.1 ist ein solcher Baum abgebildet.

Da die *Konturbäume* nur kritische Funktionswerte beinhalten, sind sie eine spezielle Version der auch bei Skelettextraktionsverfahren 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 Baumes, 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 Skelettextraktionsverfahren ist gemeinsam, dass sie kritische Werte des über den Daten liegenden Skalarfeldes nutzen, um räumliche Strukturen zu analysieren und zu speichern. Diese

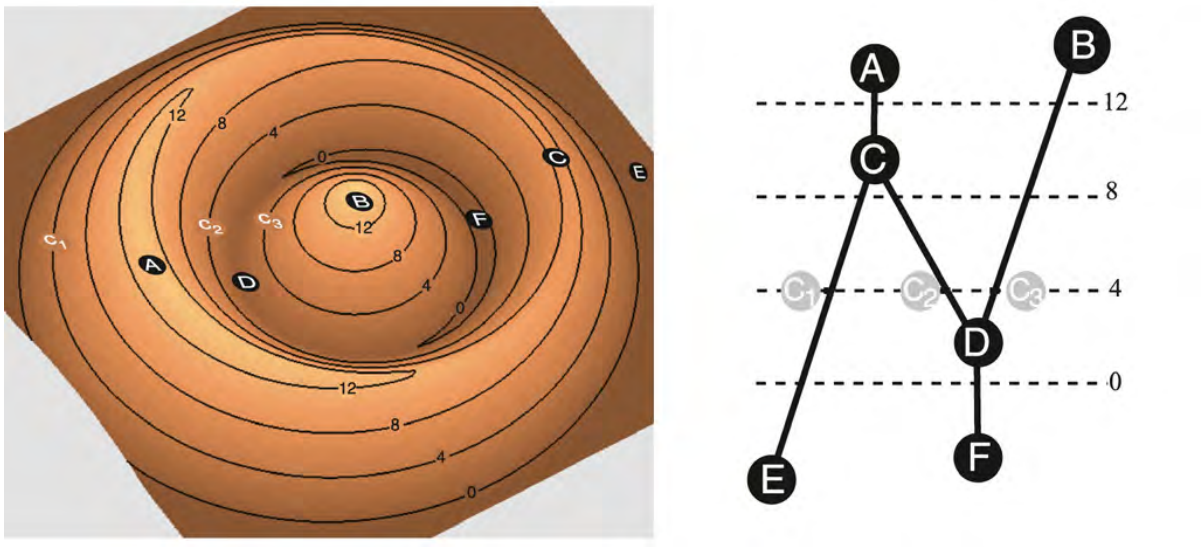


Abbildung 3.1. Oberflächenrelief eines Vulkankratersees mit einer Insel in der Mitte (links) sowie der zugehörige *Konturbaum* (rechts). A: Maximum der Kraterkante; B: Maximum des inneren Berges; C und D: Sattelpunkte; E,F: Minima. Aus [CSP10, S. 44], Bild 2.

räumlichen Strukturen liegen als Gitter oder Netze vor beziehungweise werden sie 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]

Auch aktuelle Arbeiten über *Konturbäume* benutzen Netze als Grundlage [RS14] [AN15].

Aufgrund der Voraussetzung von Gittern oder Netzen können diese Methoden auf den dieser Arbeit zugrundeliegenden Datensatz nicht direkt angewandt werden. Zwar enthält der Partikeldatensatz ebenfalls ein Skalarfeld, das jedem Datenpunkt beziehungsweise Partikel einen Funktionswert zuweist, jedoch sind keine Zusammenhänge zwischen den Partikeln bekannt. Erst im Laufe der Arbeit konnten Zusammenhänge zwischen den Partikeln erzeugt werden (siehe Abschnitt 5.2) und so konnte die Bildung des Verbindungsbaumes im Abtastalgorithmus auf das Ablaufen der Partikel bei der Clustersuche übertragen werden, indem der ihnen zugewiesene Funktionswert genutzt wird (siehe Abschnitt 5.3).

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 topologischem Änderungsgraph gespeichert werden [SB06]. Diese Idee kann auf den Vergleich der Cluster und die damit verbundene Ereigniserkennung angewandt werden (siehe Abschnitt 5.5).

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 DATENOBJEKTE

Datensammlungen haben inzwischen einen so großen Umfang, dass eine Analyse ohne Hilfsmittel nicht mehr möglich ist. Die Computervisualisierung bietet Möglichkeiten, große Datensätze so aufzubereiten, dass der Nutzer sich einen Überblick verschaffen oder gezielte Ausschnitte der Daten analysieren kann.

ANORDNUNG VON DATENOBJEKTEN

In der Arbeit von Gleicher et al ist eine umfangreiche Betrachtung zahlreicher Informationsvisualisierungen mit komplexen Objekten enthalten und es wird eine Taxonomie vorgeschlagen, um diese miteinander vergleichen zu können [Gle+11]. Dabei werden die vergleichenden, visuellen Gestaltungsmethoden in die drei Grundkategorien *Gegenüberstellung* (Juxtaposition), *Überlagerung* (Superposition) und *explizite Verschlüsselung* (explicit encoding) eingeordnet. Weiterhin werden jeweils zwei dieser Kategorien miteinander verknüpft, so dass die Taxonomie sechs Kategorien enthält.

Die Glyphdarstellung und die Darstellung der Partikel der vorliegenden Arbeit können vor allem aus Sicht der (Informations)Visualisierung als zwei getrennte Objekte betrachtet werden. Die Zielstellung ist eine Visualisierung der Partikel und *Ereignisglyphen* im selben Raum, was der Kategorie der *Überlagerung* in der von Gleicher et al vorgeschlagenen Taxonomie entspricht. Möglichkeiten für die Nutzung anderer Kategorien werden in Kapitel 8 diskutiert.

FILTERUNG VON DATENOBJEKTEN

Um Objekte mit vielen Datenelementen auswerten zu können, bieten Informationsvisualisierungen Werkzeuge wie Filter- und Zoomtechniken. Aus Mangel an verbindlichen Regeln für Darstellungen leitet Lima, losgelöst von einschränkenden Taxonomien zur grafischen Präsentation von Daten [Bri14], anhand der Analyse zahlreicher Darstellungen [Lim15] Prinzipien zur Herangehensweise an Visualisierungen großer Datensätze ab [Lim13, S. 81-95]. Trotz dessen, dass sich diese Prinzipien auf Netzwerke konzentrieren, werden zeitliche Änderungen über die Bewegung und Veränderung von Knoten und Verbindungen vorgeschlagen analog zur Bewegung der Partikel. Genauso lässt sich die Betrachtung der Sichten übertragen. Dort betont der Autor, dass die Umsetzung einer Makrosicht und einer Mikrosicht dabei hilft, dem Nutzer sowohl eine quantitative als auch eine qualitative Erfassung zu ermöglichen. Die quantitative Erfassung dient zum Erkennen von zusammengehörigen und alleinstehenden Elementen sowie der Struktur im Ganzen. Individuelle Verbindungen zwischen den Elementen sind dabei nicht relevant. Die qualitative Erfassung dient zur Bewertung einzelner Elemente, zu detaillierten Informationen über sie und damit zur Darstellung von Fakten.

Die Unterstützung beider Sichten wird durch die Bewegungs- und Zoomfähigkeit der Kamera des verwendeten *MegaMol* Viewmoduls (siehe Abschnitt A.1) grundsätzlich ermöglicht. Um

dem Nutzer auch in die Lage zu versetzen, die Daten sowohl quantitativ als auch qualitativ zu erfassen, wird eine Anpassung der Größe der *Ereignisglyphen* erlaubt.

Auch erwähnt Lima eine Beziehungssicht zur Analyse des Systems, welche die Existenz und Stärke von Verbindungen aufzeigt. Dafür wird die Nutzung mehrerer Perspektiven empfohlen. Dies kann beispielsweise mit der Kombination von Gegenüberstellung und expliziter Verschlüsselung erreicht werden (siehe Abschnitt 3.3, Kapitel 8).

ELEMENT- UND ZEITDARSTELLUNG IN MOLEKULARDYNAMIKSIMULATIONEN

Die Molekulardynamik untersucht Wechselwirkungen zwischen Atomen und Molekülen durch beispielsweise numerische Verfahren wie Runge-Kutta. Simulationen der Molekulardynamik erzeugen große Datenmengen, da selbst bei Prozessen, die in Größenordnungen um die $10^{-9}m$ stattfinden, zahlreiche Teilchen betroffen sind und sich durch chemische Bindungen oder physikalische Stoßprozesse gegenseitig beeinflussen. Hier wird sich auf die Simulation im dreidimensionalen Raum beschränkt, zumal auch die natürlichen Prozesse selbst in dieser Dimension stattfinden. Als vierte Dimension kommt die Zeit hinzu, deren Umfang oftmals aufgrund der Komplexität der stattfindenden Prozesse auf kurze Zeiträume beschränkt ist [BWW13]. Weiterhin werden aufgrund der eng begrenzten Simulationsräume oft periodische Randbedingungen genutzt, wie es auch in dieser Arbeit erfolgt.

Relevanz für diese Arbeit haben Visualisierungen von molekulardynamischen Simulationen im dreidimensionalen Raum, die Atome und Moleküle als geometrische Objekte repräsentieren. So finden gewundene Bänder für Monomere [CS07], Röhren verknüpft mit Bändern für Graphen [Sat+11], Röhren für Trajektoren von Daten einer Molekularsimulation [Gro+14] oder für Proteinkomplexe [Sma+13] [Cao+13] oder Federn für Bindungskräfte zwischen den Teilchen [Kee13] [McB13] Anwendung. Ebenfalls werden [Gro+07] Atome als Kugeln und Agglomerationen von zusammenhängenden Atomen in einer Ellipse zusammengefasst, was analog zur Darstellung der Partikel des vorliegenden Datensatzes ist. Diesen Visualisierungen gemeinsam (ausgenommen das Papier von Keesman) ist eine Farbkodierung bestimmter Eigenschaften der Elemente, was in dieser Arbeit für die Visualisierung der Cluster (siehe Abschnitt 6.5) als auch der Zeit (siehe Abschnitt 6.3) Anwendung findet.

Die Visualisierung des Zeitverlaufs erfolgt hier durch eine Bewegung der Partikel sowie durch ein dynamisches Einblenden der (statischen) *Ereignisglyphen*. Schon vor 30 Jahren wurden in Molekulardynamiksimulationen Animationen zur Visualisierung genutzt [DBW84]. Auch werden Punkt- und Liniendiagramme benutzt, in denen die Zeit auf der horizontalen Achse abgetragen ist [NRG08]. In Flussdiagrammen von Fluidsimulationen wird der Zeitverlauf ebenfalls in der horizontalen Richtung dargestellt [Lan+06]. Lima hat eine umfassende Aufstellung möglicher Zeitdarstellungen komplexer Daten zusammengetragen [Lim13].

GLYPHDESIGN

Ereignisglyphen können in dreidimensionaler oder zweidimensionaler Form erstellt werden. Durch die Nutzung eines Glyphprototypen kann in Abschnitt 6.3 gezeigt werden, dass dreidimensionale Glyphen ungeeignet sind, so dass hier nur auf die zweidimensionale Darstellung eingegangen wird.

Die Arbeit von Borgo et al. beinhaltet eine aktuelle Zusammenfassung zahlreicher Faktoren, die bei der Glyphdarstellung eine Rolle spielen [Bor+13]. So wird eine Glypheinteilung in drei Kategorien (mit drei Subkategorien) vorgestellt, wovon in dieser Arbeit ausschließlich die symbolischen Darstellung gewählt wird, da die beiden anderen Kategorien in Form der ikonischen und der Indexdarstellung physische beziehungsweise räumliche Korrelationen zum dargestellten Objekt aufweisen [Bor+13, S. 3], die bei Strukturereignissen nicht vorhanden sind. Ebenfalls Verwendung finden – in leicht abgewandelter Form – die elf aufgezählten, *visuellen Variablen* sowohl im Glyphprototyp (siehe Abschnitt A.3) als auch im entwickelten Plugin. Weiterhin

stellen Borgo et al. eine Veranschaulichung der Gestaltgesetze vor. Diese Gesetze helfen bei der Einordnung der Wirkung der Animation und der Glyphen (siehe Abschnitt 6.1) sowie bei der Auswahl der *visuellen Variablen* für die *Struktureignistaxonomie*.

Für die Glyphgestaltung in Abschnitt 6.3 dienen die Arbeiten von Maguire et al. als Inspiration, da dort die Gestaltung unter Berücksichtigung des Einsatzes in großen Graphen erfolgt und jeder Glyph mehrere Daten visualisieren kann [Mag+13] sowie die Wirkung mit zahlreichen Kombinationen aufgezeigt werden [Mag+12].

4. ÜBERBLICK

Trotz der in Abschnitt 3.2 beschriebenen Schwierigkeit, die vorgestellten Datenstrukturen und Methoden für diese Arbeit zu verwenden, liefern sie Ideen für die Entwicklung eigener Algorithmen. Diese Arbeit stellt diese Algorithmen vor, die es ermöglichen, in Punktdaten Strukturen zu erkennen und diese Strukturen zu nutzen, um daraus Strukturereignisse abzuleiten und sie zu visualisieren.

Die Entwicklung und Berechnungen erfolgen auf handelsüblichen Computern (siehe Abschnitt A.6). *MegaMol* dient als Framework und bietet Klassen für das Benutzerinterface, für die Behandlung großer Daten sowie für ihre Modifizierung, Visualisierung und Speicherung. Insbesondere die annähernd verzögerungsfreie Darstellung großer Datenmengen durch Streamingmethoden sowie das Pluginsystem sind hervorzuheben. Letzteres ermöglicht eine Kombination verschiedener Datenquellen, Berechnungs- und Ausgabemodulen. Im Rahmen dieser Arbeit entstand ein Plugin unter der MIT Lizenz (MIT). Es enthält vier Module und eine Kommunikationsklasse. Das *StructureEventsCalculation Modul* führt sämtliche, in Kapitel 5 beschriebenen Berechnungen durch, das *StaticRenderer Modul* dient zur Visualisierung der Ereignisse (siehe Abschnitt 6.4), das *StructureEventsDataSource Modul* sowie das *StructureEventsWriter Modul* dienen zum Lesen und Schreiben der Ereignisdaten. Für den Datentransfer der Ereignisdaten zwischen den Modulen wird der *StructureEventsDataCall* verwendet. Calls sind spezielle Klassen zur intermodularen Kommunikation der *MegaMol* Module. Beispielhafte Programmaufbauten können in Abschnitt A.1 betrachtet werden. Zur effizienten Erstellung der Aufbauten dient das dem *MegaMolprojekt* zugehörige Werkzeug *MegaMol Configurator*.

Zur Berechnung wird das Jobsystem von *MegaMol* verwendet, welches es ermöglicht, die Zeitschritte eines MegaMol Multi Particle List Dump (MMPLD) (das Dateiformat, in dem der dieser Arbeit zugrundliegende Datensatz vorliegt) sequentiell nacheinander und automatisiert aufzurufen. Zur Ausführung der Jobs werden Extensible Markup Language (XML) Dateien genutzt, die einen Aufbau von Modulen, ihrer Eigenschaften sowie von Kommunikationsklassen beschreiben. Sämtliche verwendete Bibliotheken und Werkzeuge sind thematisch in Abschnitt A.7 aufgeführt.

4.1. BESCHREIBUNG DER SIMULATION

Die dieser Arbeit zugrundeliegende Animation simuliert eine von einem Vakuum umgebenen Flüssigkeit, die rasch expandiert. Bedingt durch das quaderförmige, längliche Simulationsvolumen und der vollständigen Füllung mit Flüssigkeit in die zwei kurzen Raumrichtungen des Quaders sowie deren zentrale Positionierung, entstehen zwei Flüssigkeitsfronten, die entlang der langen Raumrichtung des Simulationsgebiets entgegengesetzt auseinander driften. Dazwischen entstehen Filamente und Tröpfchen und Gas durch sich aus den Fronten lösende Partikel

und Partikelagglomerationen. Der Begriff Filament beschreibt dabei, dass es sich um Gebilde handelt, die im Vergleich zu ihrer Länge dünn sind. Tröpfchen sind von den Fronten unabhängige Gebilde mit eher kugelartiger Form. In den Leerräumen dazwischen befindet sich Gas. Die Gasphase wird dadurch definiert, dass alle Partikel große Abstände zueinander haben und sich beispielsweise gegenseitig nicht berühren.

Mit zunehmender Entfernung der Flüssigkeitsfronten zerreißen die Filamente zunehmend. Bei Zeitschritt 89 kollidieren beide Fronten miteinander, vermischen sich und driften etwa ab Zeitschritt 94 wiederholt auseinander. Das geschieht, weil das Simulationsvolumen periodische Randbedingungen aufweist. Zur Vereinfachung der Vorstellung kann der Quader als Ring betrachtet werden, auch wenn dieses Bild die Periodizität der anderen beiden Raumrichtungen nicht berücksichtigt.

Nach die Kollision bewegen sich beide Fronten in die entgegensetzte Richtung „zurück". Dabei fangen sie vor der Kollision gebildete Filamente und Tröpfchen und hinterlassen „hinter" sich vor allem Tröpfchen. Die Animation endet mit Zeitschritt 149, bevor sich die Fronten noch einmal begegnen. Die Visualisierung der Animation kann online in einem Video[1] begutachtet werden und enthält bereits eine durch den CFD berechnete, vererbte Einfärbung der Cluster.

[1] https://www.youtube.com/watch?v=GiAkZTbgKsQ

5. ERKENNUNG DER CLUSTER UND STRUKTUREREIGNISSE

Zur Erkennung der Strukturereignisse ist eine Analyse der Struktur erforderlich, die in der Bildung von Clustern resultiert. Der gesamte Berechnungsprozess wird in vier Schritte unterteilt. Als erstes wird eine Partikelliste erstellt und in dieser werden Informationen über die benachbarten Partikel hinterlassen. Als nächstes werden unter Nutzung dieser Nachbarschaftsverhältnisse Cluster erstellt und anschließend diese Cluster über zwei aufeinanderfolgende Zeitschritte verglichen. Als vierter und letzter Schritt folgt die Erkennung der Strukturereignisse über eine Heuristik, die den Vergleich aus Schritt drei nutzt. Eine zusammenfassende Liste befindet sich in Abschnitt A.2.

Vor der Beschreibung der Berechnungen werden die zugrundeliegenden Daten und die genutzten Datenstrukturen erklärt.

5.1. DATENBEHANDLUNG

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 Entfernungsangabe. 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*.

Die Partikel-, Cluster- und Ereignisdaten werden als eine dichtgepackte Struktur gespeichert (siehe Tabelle 5.1, Tabelle 5.2, Tabelle 5.3). 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].

Die Referenzen der Partikel untereinander sowie die der Partikel zu Clustern und umgekehrt werden mit Hilfe von Identifikatoren umgesetzt, die direkt in den Strukturen gespeichert sind. Dies hat Vorteile im Vergleich zur Nutzung von Zeigern als Referenz oder zur Speicherung der Objekte selbst. Letzteres erzeugt durch das Anlegen von Objektkopien Redundanz und kann daher zu inkonsistenten Daten führen. Weiterhin bedingt dies skalierend mit der Objektanzahl eine enorme Speicherbelastung und sorgte bei Tests auf dem Entwicklungssystem (siehe

Tabelle 5.1. Struktur des Partikelobjekts.

Bytes	Datentyp	Beschreibung
0...11	3x float (32 bit)	Position (*x*, *y*, *z*)
12...15	float (32 bit)	Radius
16...27	3x float (32 bit)	Farbe (*r*, *g*, *b*)
28...31	float (32 bit)	*Tiefenwert*
32...39	uint64_t	Identifikator
40...43	int (32 bit)	Clusteridentifikator
44...(variabel)	std::vector<uint64_t>	Zeiger auf einen Vektor, der die Nachbaridentifikatoren enthält

Tabelle 5.2. Struktur des Clusterobjekts.

Bytes	Datentyp	Beschreibung
0...7	uint64_t	Identifikator des *Wurzelpartikels*
8...15	uint64_t	Anzahl an Partikeln
16...19	int (32 bit)	Identifikator
20...31	3x float (32 bit)	Farbe (*r*, *g*, *b*)

Tabelle 5.3. Struktur des Ereignisobjekts.

Bytes	Datentyp	Beschreibung
0...11	3x float (32 bit)	Position (*x*, *y*, *z*)
12...15	float (32 bit)	Zeitpunkt
16...19	enum (32 bit)	Ereignisart

Abschnitt A.6) zu ständigem Auslagern auf die Festplatte. Vorteile der Objektkopien sind der schnelle Zugriff auf die Daten sowie die Unabhängigkeit von Referenzen auf andere Objekte. Die Verwendung von Zeigern ermöglicht einen unwesentlich langsameren Zugriff und es kann auf Kopien der referenzierten Objekte verzichtet werden. Allerdings darf sich bei Nutzung dieser Methode die Position der Objekte im Speicher nicht verändern, was je nach verfügbaren Ressourcen nicht garantiert werden kann und zu undefiniertem Verhalten führt. Denn es kann, aufgrund der während der Laufzeit dynamisch allokierten und nicht vorhersagbaren Größe, nicht ausgeschlossen werden, dass die Clusterliste im Speicher verschoben wird. 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. Dies lässt sich an den ausgelesen Werten erkennen. So weist der Wert -572662307 beim Auslesen eines Integers darauf hin, dass der Speicherallokator einer Liste diesen Wert gesetzt haben könnte, was wiederum sagt, dass sich vorher eine Liste in diesem Speicherbereich befunden haben könnte.

Um dem Problem zu begegnen, werden statt Zeigern Identifikatoren eingesetzt (siehe Tabelle 5.1, Tabelle 5.2, Tabelle 5.3). Diese Identifikatoren werden genutzt, um zur Referenzierung auf die Listenzeiger zurückgreifen zu können. Die Methode hat drei Vorteile. Erstens ist die Geschwindigkeit nicht langsamer als mit einer direkten Zeigerreferenz, zweitens verweisen die Referenzen auch bei einer Verschiebung der die Objekte enthaltenen Listen im Speicher noch auf die korrekten Objekte und drittens kann geprüft werden, ob die Referenz mit dem im Speicher befindlichen Objekt übereinstimmt. Diese Prüfmöglichkeit ist dann wichtig, wenn die Objektlisten umsortiert werden, zum Beispiel durch das Hinzufügen oder Entfernen von Elementen, die sich nicht am Ende der Liste befinden.

5.2. SCHRITT 1: BESTIMMUNG DER NÄCHSTEN NACHBARN

Ausgehend von der in Abschnitt 3.2 referenzierten Bestimmung von Zusammenhangskomponenten über einen Maximalabstand zwischen Punkten im Netz, der ohne Baumstrukturen auskommt, könnte hier ebenfalls eine radialbasierte Nachbarschaftsbeziehung verwendet werden. Diese wurde geprüft und aufgrund des hohen Zeitaufwandes, verursacht durch die Entfernungsbestimmung zwischen Partikeln, verworfen (siehe Abschnitt A.8).

Eine Einteilung des Raumes in ein Gitter abhängig des Partikelradius wäre eine weitere Option. Einen ähnlichen Ansatz nutzt der kD-Baum, der die Partikel abhängig von ihrer Position in seine Äste verteilt und dadurch die Suche nach benachbarten Partikeln stark beschleunigt. Die Nutzung des Baumes bietet sich an, da die Anzahl der Partikel und ihre Positionen sich während eines Zeitschrittes nicht verändern und der Baum somit nicht modifiziert werden muss. Der Baum hat aufgrund der drei Raumrichtungen die Dimension $k = 3$. Die Identifikation der Partikel erfolgt über den im kD-Baum eingetragenen Index, der der Position der Partikel in der Partikelliste entspricht. Daher wird während der Laufzeit über den beim Partikel gespeicherten Identifikator überprüft, ob die Reihenfolge der Liste modifiziert wurde, um Falschzuweisungen registrieren zu können.

Die periodischen Randbedingungen des Datensatzes werden standardmäßig beachtet, indem die Nachbarschaftssuche für jeden Partikel achtmal durchgeführt wird. Dabei wird die Position des Partikels für jede Raumrichtung auf die gegenüberliegende Seite des Simulationsvolumens versetzt, was bei drei Raumrichtungen acht Kombinationen ergibt. Für Datensätze ohne periodische Randbedingungen kann der Nutzer dieses Vorgehen deaktivieren, so dass nur die Originalposition des Partikels verwendet wird.

MegaMol stellt mit der Approximate Nearest Neighbor Bibliothek (ANN)-Bibliothek eine Bibliothek für die Nutzung von kD-Bäumen bereit. Von dieser wird die radienbasierte Suche annkFRsearch verwendet, da sich so eine Reichweite bestimmen lässt (siehe Abschnitt 3.2). Als Suchparameter werden neben des Radius' selbst auch eine maximale Anzahl von zu prüfenden Nachbarn festgelegt. Dies ist nötig, da die ANN-Bibliothek die Größe des Containers

Tabelle 5.4. Experimentell ermittelte Werte für die maximal zu erfassenden Nachbarn, damit alle Partikel innerhalb des durch den Radiusmultiplikator vorgegebenen Suchvolumens erfasst werden.

Partikelradiusmultiplikator	Maximal erfassbare Nachbarn
4	35
5	60
6	100
7	155
10	425
20	3270

zur Speicherung der Identifikatoren der festgestellten Nachbarn (Nachbarschaftsliste) im Voraus festlegt. Dieser Wert wird abhängig vom Radius durch die Berechnung automatisch gesetzt. Die passenden, in Tabelle 5.4 aufgeführten 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.

Bei der Bestimmung wurde ein empirisches Annäherungsverfahren genutzt, in welcher die Anzahl der hinzugefügten Nachbarn bei gleichbleibendem Radius überprüft wurde. Dabei wurde mit kleinen Werten für die maximal erfassbaren Nachbarn begonnen und dieser Wert stückweise erhöht. So lange die Gesamtanzahl der zu den Partikeln hinzugefügten Nachbarn stieg, war die Anzahl an erfassbaren Nachbarn zu klein. Sobald die Nachbarn identisch blieben, war der optimale Wert gefunden. Aus Effizienzgründen wurde dieser Wert sowohl aus großer als auch als kleiner Richtung überprüft, um schneller den optimalen Wert zu erreichen.

Der Suchradius wird durch den Nutzer einstellbar gestaltet, da mit der Reichweite der Nachbarschaftssuche die Anzahl der Cluster und damit die gesamte Berechnung beeinflusst wird (siehe Abschnitt 8.1). Der Parameter wird dabei mit dem Partikelradius multipliziert, so dass die Bezeichnung *Partikelradiusmultiplikator* gegeben wird.

Die Suche wird für jeden Partikel durchgeführt. Dabei werden die Identifikatoren der gefundenen Nachbarn in die dafür vorgesehene Liste jedes Partikelobjekts hinzugefügt (siehe Abschnitt 5.1). Zuvor werden alle Partikel des MMPLD aus dem eingehenden Multi Particle Data Call (MPDC) in diese Partikelobjekte umgewandelt und in die Partikelliste geschrieben. Da die Anzahl der im MMPLD enthaltenen Elemente bekannt ist, findet eine Speicherplatzreservierung der Partikelliste statt, um eine Reallokation während der Eintragung der Partikel zu vermeiden.

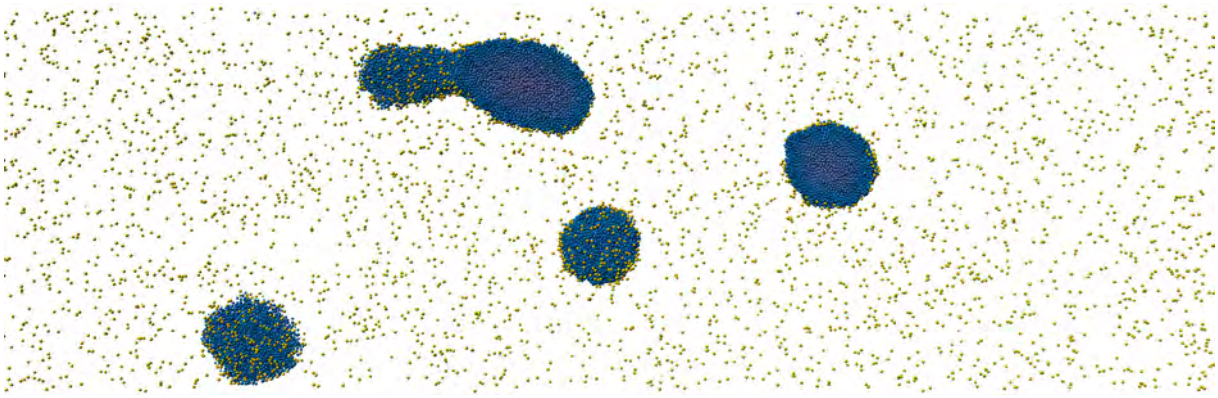
Mit der Erzeugung dieser Partikelliste, der Eintragung aller Partikel in den kD-Baum und der Eintragung der Nachbarn in die Partikelobjekte ist der erste Schritt für die Berechnung der Strukturereignisse abgeschlossen und es folgt die Clusterbestimmung.

5.3. SCHRITT 2A: CLUSTER-FAST-DEPTH ALGORITHMUS ZUR BESTIMMUNG DER CLUSTER

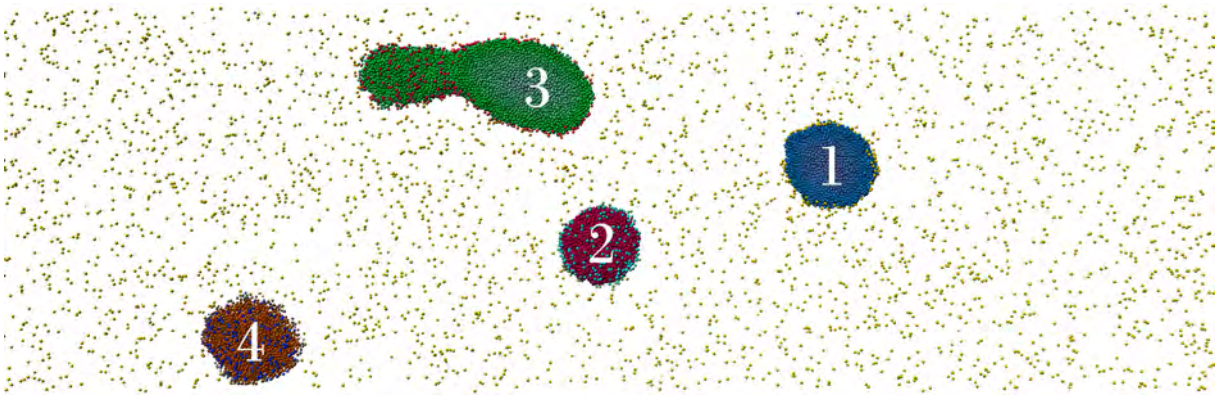
Zur Clusterbestimmung wurde der CFD entwickelt. Der Begriff wurde ausgehend von der Nutzung des *Tiefenwertes* gewählt, angelehnt an die Depth-First Suche in Baumstrukturen.

Da auch Verschmelzungen und Teilungen innerhalb von Zusammenhangskomponenten erkannt werden sollen, genügt es nicht, wenn die Clusterbildung alle benachbarten Partikel einer Gruppe zuordnet wie in Abbildung 5.1 abgebildet. Vielmehr sollen auch Zusammenhangskomponenten selbst unterteilt werden. Diese Idee ist in Abbildung 5.2 visualisiert.

Dabei wird die Idee des Reebgraphen sowie die Bildung des Verbindungsbaumes beim

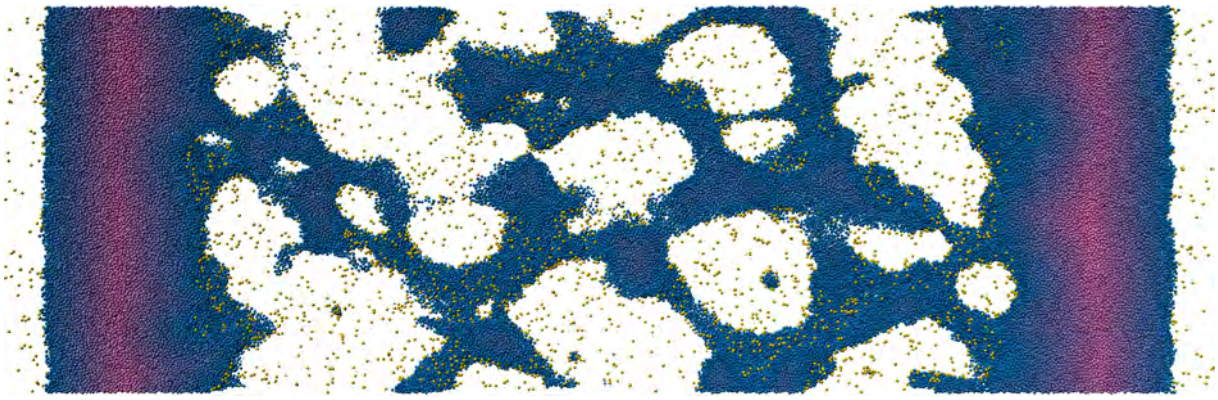


a) Tröpfchen separat verteilt im Raum.

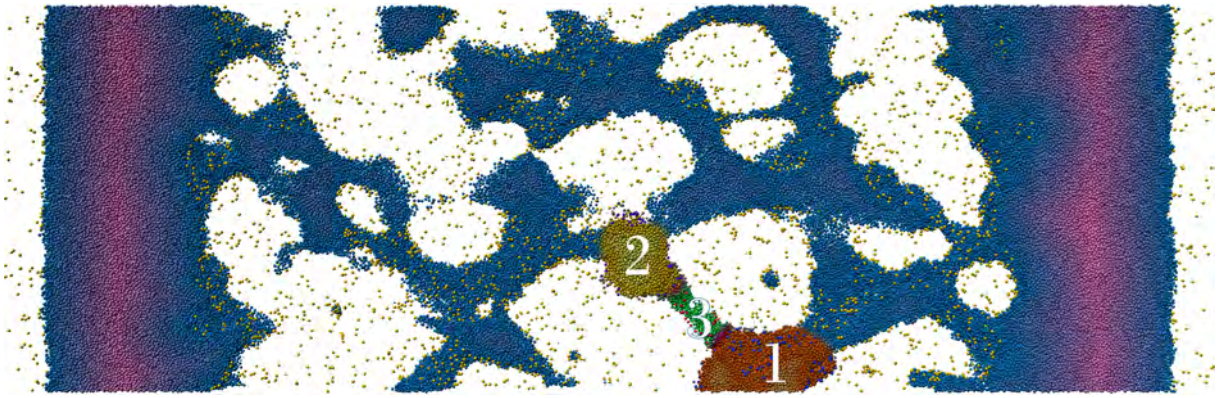


b) Einfärbung der Tröpfchen zum Zeigen einer möglichen Clusteraufteilung. Die Einfärbungen und die Zahlen eins bis vier markieren dabei jeweils einen Cluster. Trivia: Cluster drei würde vom CFD in mehrere Cluster eingeteilt werden aufgrund der Einschnürung in der Mitte.

Abbildung 5.1. Ein Cluster füllt jeweils eine gesamte Zusammenhangskomponente. Zu sehen sind Tröpfchen (blau) in einem Ausschnitt der Simulation bei Zeitschritt 90. Gelbe Partikel befinden sich in der Gasphase.



a) Filamentstrukturen innerhalb einer Zusammenhangskomponente zwischen den beiden Flüssigkeitsfronten.



b) Das Ziel ist, Partikel auch innerhalb von Zusammenhangskomponenten in Cluster aufzuteilen. Die Einfärbungen und die Zahlen eins bis drei markieren dabei jeweils einen Cluster.

Abbildung 5.2. Clusteraufteilung innerhalb einer Zusammenhangskomponente. Zu sehen ist ein Schnitt durch den Simulationsraum bei Zeitschritt 16. Gelbe Partikel befinden sich in der Gasphase.

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.

Die Partikelliste wird iterativ durchlaufen. Dabei werden die bei der Nachbarschaftssuche erstellten Beziehungen zwischen Partikeln genutzt, um für jeden Partikel denjenigen Nachbarn zu finden, der den höchsten *Tiefenwert* aufweist. Ist dieser Nachbar gefunden, werden für diesen noch einmal alle Nachbarn nach dem „tiefsten" durchsucht und dasselbe für diesen Partikel durchgeführt. Diese Suche geschieht ebenfalls iterativ. Sobald die umliegenden Partikel nur noch den gleichen oder einen geringeren *Tiefenwert* als den des zuletzt betrachteten Partikels aufweisen, gilt die Suche als abgeschlossen. Anschließend wird ein neuer Cluster erstellt mit diesem Partikel als *Wurzelpartikel*. Sollte bereits ein Cluster mit diesem Partikel als Wurzel existieren, so wird kein neuer Cluster erstellt. Danach werden der ursprüngliche Partikel als auch alle Partikel, die während des iterativen Ablaufs der tiefsten Nachbarn getroffen wurden, zum Cluster hinzugefügt. Dies geschieht, indem der Clusteridentifikator des neu erstellten oder bereits vorhandenen Clusters in das Referenzfeld der Partikel für ihren zugehörigen Cluster (siehe Abschnitt 5.1) eingetragen wird. TODO HIER KOMMT EIN SCHEMA HIN

Die Traversion erstellt einen kürzesten Pfad vom Ausgangspartikel zum Partikel mit der größten, lokalen Tiefe, wovon sich der Name für diesen Algorithmus ableitet. Da alle bei der Traversion getroffenen Partikel zum Cluster hinzugefügt werden, kann der Algorithmus beschleunigt werden. Bevor die Fast-Depth Suche begonnen wird, wird der aktuell betrachtete Partikel auf seine Clusterzugehörigkeit geprüft. Sollte er einem Cluster angehören, so wird die Suche nicht erst begonnen, sondern es wird zum nächsten Partikel gesprungen.

Weiterhin werden Gaspartikel anhand ihres *Tiefenwertes* erkannt und übersprungen. Auch Partikel ohne Nachbarn werden übersprungen. Diese Überprüfung scheint redundant, jedoch können bei sehr kleinen Suchradien der Nachbarschaftssuche auch Flüssigkeitspartikel keine Nachbarn aufweisen, deren fehlende Nachbarschaft dann die Bedingung für Cluster, einer Zusammenhangskomponente anzugehören, nicht erfüllen würde.

Damit ist der wesentliche Teil der Clusterbildung abgeschlossen. Der zweite Schritt beinhaltet lediglich noch eine Reduktion kleiner Cluster. Die Definition für kleine Cluster kann dabei vom Nutzer getroffen werden.

5.4. SCHRITT 2B: 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.

Es könnte eine Menge aus diesen sehr kleinen Clustern gebildet werden. Mit jedem Cluster diese Menge werden die naheliegensten „großen" Cluster gesucht, zum Beispiel durch Anordnung der Rootpartikel der Cluster in einem kD-Baum. Allerdings ist eine Prüfung, ob sich die Cluster in einer Zusammenhangskomponente befinden, sehr aufwändig.

Aufgrund dessen, dass sich in der Arbeit auf eine kleine Mindestclustergrößen von zehn Partikeln beschränkt wird, kann eine iterative Prüfung auf benachbarte, große Cluster unter Nutzung der bestehenden Nachbarschaftsbeziehungen der Partikel genutzt werden. Dabei werden bei jedem Partikel, der ein Flüssigkeitspartikel mit Nachbarn ist und dessen referenzierter Cluster die Mindestclustergröße unterschreitet, die Nachbarn durchsucht. Sollten die Nachbarn alle keinem, dem selbem oder einem anderen sehr kleinen Cluster angehören, wird mit ihren Nachbarpartikeln die selbe Suche durchgeführt. In der aktuellen Implementation wird diese Suche maximal dreimal durchgeführt, so dass ausgehend vom ursprünglich betrachteten

Partikel der maximal drittentfernteste Nachbar nach anderen Cluster durchsucht wird. Sollten zuvor schon andere Cluster gefunden worden sein, wird die Suche bereits dort abgebrochen. Nachbarpartikel, die keinem Cluster angehören (hier vor allem Gaspartikel, siehe Abschnitt 5.3), werden übersprungen.

Sollte nach Abschluss dieser Suche kein anderer, „großer" Cluster gefunden sein, wird der sehr kleine Cluster bestehen gelassen, da dann davon ausgegangen wird, dass der Cluster eine Zusammenhangskomponente bildet und daher mit keinem anderen Cluster verschmolzen werden kann. Diese Annahme ist für eine Mindestclustergröße von zehn berechtigt, denn bei der dichten Struktur der Partikel in der Flüssigkeit erreicht spätestens die zweite Nachbarschaftsebene diese Anzahl an Partikeln. Der dritte Suchlauf dient als Puffer für etwas größere Mindestclustergrößeneinstellungen.

Sollte ein Cluster gefunden worden sein, so wird der betrachtete Partikel diesem Cluster zugeordnet. Die getroffenen Nachbarn werden nicht hinzugefügt, um die Schleife in mehreren Prozessen ohne Mutex ablaufen lassen zu können. Sollten mehrere Cluster gefunden worden sein, wird eine Entscheidung anhand der Richtung des Partikels zu den benachbarten Clustern bestimmt. Er wird zu demjenigen Cluster hinzugefügt, der die ähnlichste Richtung aufweist, die der Richtung des Partikels zum *Wurzelpartikel* seines Clusters entspricht.

Dazu wird der Winkel zwischen der Richtung vom *Wurzelpartikel* des betrachteten Partikels und seiner Position zur Richtung des Partikels zum in Frage kommenden Cluster berechnet. Dies wird für alle benachbarten Cluster durchgeführt. Der benachbarte Cluster mit dem geringsten Winkel im Intervall $[0, \pi]$ wird der neue Cluster des Partikels.

Da die Partikel voneinander unabhängig sind, können sie parallel durchlaufen werden (Umsetzung mittels *OpenMP*). Weiterhin wird die Winkelberechnung zu jedem benachbarten Cluster parallelisiert. Die Cluster werden nicht aus der Clusterliste gelöscht, um die Referenzen auf die Liste nicht neu erstellen zu müssen. Stattdessen wird durch den Algorithmus ihre Partikelanzahl auf 0 reduziert (sollten alle Partikel neuen Clustern zugeordnet werden), so dass sie in den folgenden Schritten herausgefiltert werden können.

5.5. SCHRITT 3: BESTIMMUNG DER CLUSTERZUSAMMENHÄNGE

Zur Bestimmung der Ereignisse ist es notwendig, die Entwicklung der Cluster zu verfolgen. Dazu werden Cluster über zwei Zeitschritte hinweg miteinander verglichen. Dieser Berechnungsschritt wird als SECC bezeichnet.

CLUSTERNACHBARSCHAFTSGRAPH

Eine Variante ist die Erzeugung eines Graphen, der den Zusammenhang zwischen den Clustern zeigt und die bestehenden Nachbarschaftsbeziehungen der Partikel nutzt. Dazu kann die Clusterzugehörigkeit der Nachbarn jedes Partikels 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. Durch Begrenzung auf Partikel mit geringen *Tiefenwerten* kann diese Suche stark beschleunigt werden und dennoch alle Nachbarschaftsbeziehungen gefunden werden, da einerseits der CFD vor allem in die Tiefe und wenig in die Breite sucht und andererseits diese Beziehungen zwischen zwei Clustern aus beiden Richtungen geprüft werden. Weiter soll darauf nicht eingegangen werden, da die Schwierigkeit dieser Variante in der Clusteridentifizierung über Zeitschritte hinweg besteht. Der Rootpartikel ändert sich von Zeitpunkt zu Zeitpunkt, die Partikelzugehörigkeit der Cluster ebenfalls (siehe Abschnitt 7.3). Sie könnte als Ergänzung des hier implementierten Clustervergleichs dienen (siehe Abschnitt 8.5).

VERGLEICH DER ZUGEHÖRIGEN PARTIKELMENGEN

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 (siehe 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 und deren Spalten die Cluster des alten Zeitschrittes enthalten. Ein Partikel wird in diese *Clustervergleichsmatrix* eingetragen, indem die Clusterzugehörigkeit des aktuellen Zeitschrittes sowie die Clusterzugehörigkeit des vorherigen Zeitschrittes ermittelt werden.

Anschließend werden zwei Schritte durchgeführt, um die für die Auswertung und schließlich die für die Eventerkennung notwendigen Werte zu ermitteln. In jedem Schritt werden alle Cluster eines Zeitschrittes betrachtet und für jeden werden Cluster des anderen Zeitschrittes gesucht, die mit dem betrachteten Cluster gemeinsame Partikel aufweisen. Diese werden als *Partnercluster* oder Partner bezeichnet. Im ersten Schritt werden alle Cluster des vorherigen Schrittes mit jedem des aktuellen Zeitschrittes verglichen (vorwärtsgerichteter Vergleich), im zweiten Schritt alle derzeitigen mit jedem Cluster des vorherigen Zeitschrittes (rückwärtsgerichteter Vergleich).

PARTNERCLUSTERSTRUKTURLISTEN

Zur Speicherung der Vergleichsergebnisse wird eine Partnerclusterstruktur eingeführt. Diese enthält eine Liste der *Partnercluster* zusammen mit den jeweils gemeinsamen Partikeln sowie Methoden zur Berechnung von Gesamt- und Durchschnittswerten. Für beide Vergleichsrichtungen werden diese Strukturen in separate Listen geschrieben, so dass eine *vorwärtsgerichtete* und eine *rückwärtsgerichtete Partnerclusterstrukturliste* entstehen. Damit lassen sich die Anzahl der gemeinsamen Partikel für jeden vorhergehenden Cluster und jeden aktuellen bestimmen, woraus sich die weiteren für die Analyse notwendigen Werte ermitteln lassen.

5.6. SCHRITT 4: ERKENNUNG VON EREIGNISSEN

Es werden jeweils ein Cluster und seine *Partnercluster* betrachtet. Dies geschieht sowohl für den vorwärtsgerichteten als auch den rückwärtsgerichteten Vergleich.

Zur Analyse der durch die Partnerclusterstruktur zurückgegebenen Daten wird ein heuristisches Vorgehen genutzt. 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 ist der intuitiv wirkende Vergleich über den Clusteridentifikator für die Beobachtung von Veränderungen nicht möglich und kann ausgeschlossen werden. Dagegen kann ein Vergleich der Anteile von Partikeln in den Clustern erkennen lassen, 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 Größe eines Clusters hat für die Gewichtung von Ereignissen keine Bedeutung, da sie durch die Vorgehensweise der CFD stark von der Topologie der Strukturen abhängt. Ein durch einen kleinen Cluster an einer Engstelle eines Filaments ausgelöstes Ereignis kann für die Analyse bedeutender sein als ein Ereignis innerhalb der Flüssigkeitsfront, ausgelöst durch große Cluster. Um die Abhängigkeit der Ereigniserkennung von der *Clustergröße* zu eliminieren, werden relative Werte verwendet. Das heißt, die Methoden der Strukturen des SECC geben

Werte im Verhältnis zur Größe der Cluster an. Dazu zählen der Anteil der gemeinsamen Partikel aller *Partnercluster* a_{gesamt}, der Anteil für jeden einzelnen Partner a_{partner}, der Durchschnittsanteil der gemeinsamen Partikel für alle Partner $a_{\text{durchschnitt}}$ sowie Maxima (a_{max} p_{max}), Minima (a_{min} p_{min}) und die Anzahl der Partner p.

Um festzustellen, ob ein Cluster einen ähnlichen Vorgänger oder Nachfolger hat, können der durchschnittliche Anteil der gemeinsamen Partikel über alle Partner, der Anteil des größten Partners im Vergleich dazu sowie die Gesamtanzahl der Partner herangezogen werden. Sollten die ersten beiden Werte hoch und der dritte Wert klein sein, kann mit einer, von diesen Werten abhängigen, Wahrscheinlichkeit davon ausgegangen werden, dass es sich um denselben Cluster handelt. Dies ist eine undurchsichtige Methode zur Erkennung von Ereignissen, da sie von drei verschiedenen, wenig intuitiven Werten abhängig ist. Sie dient als schlechtes Beispiel. Darüber hinaus ist es für die Erkennung der Ereignisse, aufgrund der zuvor genannten Fluktuation der Partikel zwischen Clustern, nicht relevant, ob ein Cluster einen direkten Vorgänger hat.

ERMITTLUNG VON *BIRTH* UND *DEATH*

Wie eingangs erwähnt, werden die Ereignisse ermittelt, indem die Cluster in den beiden *Partnerclusterstrukturlisten* (siehe Abschnitt 5.5) nacheinander betrachtet und die genannten Verhältnisse sowie Absolutwerte der *Partnercluster* berücksichtigt werden. In den folgenden formalen Ausdrücken steht der Index v für die Nutzung des vorwärtsgerichteten und der Index r für die Nutzung des rückwärtsgerichteten Vergleichs.

Birth und *Death* bieten die einfachste Herangehensweise, da nur eine Überprüfung auf einen Übergang der Partikel von oder in die Gasphase vorgenommen werden muss. Auf die *Partnercluster* übertragen bedeutet das, im vorwärtsgerichteten Vergleich resultiert ein Cluster ohne Partner ($p_{\text{gesamt}_v} = 0$) in einem *Death*, im rückwärtsgerichteten Vergleich (mit $p_{\text{gesamt}_r} = 0$) in einem *Birth*. Um Phänomene untersuchen zu können, wird dem Nutzer ein Parameter s_{bd} bereitgestellt, mit dem auch Cluster mit einem benutzerdefiniertem Anteil an gemeinsamen Partikeln (a_{gesamt}) als *Birth* oder *Death* erkannt werden.

Damit muss für die Erkennung eines *Birthereignisses* eine der in Gleichung 5.1 formulierten und für die Erkennung eines *Deathereignisses* eine der in Gleichung 5.2 formulierten Gleichungen erfüllt sein.

$$\begin{aligned} p_{\text{gesamt}_r} &= 0 \quad (B) \\ s_{\text{bd}} &\leq a_{\text{gesamt}_r} \quad (C) \end{aligned} \tag{5.1}$$

$$\begin{aligned} p_{\text{gesamt}_v} &= 0 \quad (D) \\ s_{\text{bd}} &\leq a_{\text{gesamt}_v} \quad (E) \end{aligned} \tag{5.2}$$

Birth wird erkannt, wenn $(B \vee C) = \top$ und *Death* wird erkannt, wenn $(D \vee E) = \top$.

ERMITTLUNG VON *MERGE* UND *SPLIT*

Merge und *Split* können von vielen Faktoren abhängen. Etwa einer Kombination aus einem hohen durchschnittlichen Anteil gemeinsamer Partikel aller *Partnercluster* und damit einhergehend einer kleinen Anzahl an Partnern, um auszuschließen, dass das Ereignis als Rauschen einzuordnen ist. Rauschen kann entstehen, wenn ein Cluster zahlreiche kleine Partner besitzt. Der Nachteil darin liegt, dass diese Parameter wenig intuitiv sind und es daher zahlreicher Messungen bedarf, ab welchem Grenzwert des durchschnittlichen Anteils beziehungsweise der Partner solch ein Ereignis auftritt.

Aus diesem Grund wird der Begriff des *Großen Partners* eingeführt. *Große Partner* sind solche *Partnercluster*, die einen bestimmten Schwellwert des Anteils gemeinsamer Partikel überschreiten. Diesen Schwellwert kann der Anwender durch den Parameter s_a festlegen. Dieser Parameter wird zur Ermittlung der Anzahl der *Großen Partner* $p_{\text{groß}}$ genutzt. Diese Anzahl

wird für jeden Cluster sowohl der *vorwärtsgerichteten* als auch der *rückwärtsgerichteten Partnerclusterstrukturliste* separat ermittelt. Dazu werden die jedem Cluster zugehörigen *Partnercluster* durchlaufen. Beim Prüfen der Cluster der *vorwärtsgerichteten Liste* muss ein *Partnercluster* die in Gleichung 5.3 formulierte Bedingung erfüllen, damit $p_{\text{groß}_v}$ inkrementiert wird.

$$a_{\text{partner}_v} \geq s_a \tag{5.3}$$

Beim rückwärtsgerichteten Vergleich ist die in Gleichung 5.4 aufgeführte Bedingung für die *Partnercluster* zur Inkrementation von $p_{\text{groß}_r}$ analog.

$$a_{\text{partner}_r} \geq s_a \tag{5.4}$$

Über den Vergleich ihrer Anzahl $p_{\text{groß}}$ mit einem bereitgestellten Parameter s_p kann der Nutzer bestimmen, ob zwei, drei, vier oder mehr Partner mit der zu erfüllenden Bedingung in Gleichung 5.5 als *Merge* beziehungsweise der zu erfüllenden Bedingung in Gleichung 5.6 als *Split* erkannt werden sollen.

$$p_{\text{groß}_r} \geq s_p \quad (M) \tag{5.5}$$

$$p_{\text{groß}_v} \geq s_p \quad (S) \tag{5.6}$$

Merge wird erkannt, wenn $M = \top$ und *Split* wird erkannt, wenn $S = \top$.

So kann der Nutzer zwei Eigenschaften von Verschmelzungen beziehungsweise Abspaltungen festlegen. Zum einen die Anzahl der beteiligten Cluster. Zum anderen können durch das Setzen niedrigerer Anteile an gemeinsamen Partikeln auch Abspaltungen oder Verschmelzungen kleiner Cluster von oder mit großen Clustern registriert werden. Durch die Betrachtung der *Großen Partner* muss keine separate Behandlung des bereits erwähnten Rauschens erfolgen. Rauschen entsteht durch ständiges Hinzufügen und Lösen von Randpartikeln eines Clusters, was sich in vielen kleinen Partnern äußern kann.

5.7. PROTOKOLLIERUNG UND DATENSPEICHERUNG

Zur Überprüfung der Funktionalität der einzelnen Berechnungsschritte sowie zu ihrer quantitativen Analyse wird ein umfangreiches Protokoll erstellt, das bei Bedarf deaktiviert werden kann. Dazu werden Text- und Comma Separated Values (CSV)-Dateien mit den Parametern und Berechnungsergebnissen erstellt.

Da die Berechnungen eine gewisse Zeit andauern, sind sie für eine interaktive Visualisierung nicht geeignet. Daher werden die berechneten Ereignisse gespeichert, so dass auf sie schnell zugegriffen werden kann. Zur Speicherung wurde das Dateiformat MegaMol Structure Event (MMSE) entwickelt. Dieses hält die berechneten Ereignisse in einem Bytecode analog zu existierenden *MegaMol*-Formaten wie MMPLD fest, um zum einen Konsistenz im Aufbau zu den *MegaMol*-Modulen zu wahren und zum anderen die Entwicklung der Ein- und Ausgabe zu beschleunigen. Der Vorteil im Vergleich zu XML oder anderen textbasierten Speicherverfahren ist die höhere Performance beim Lesen und Schreiben sowie die kleinere Dateigröße.

Im Kopf befinden sich Daten für die Identifizierung des Dateityps, die Abmessungen des Simulationsraumes, die Ereignisanzahl und für den höchsten gespeicherten Zeitschritt der Ereignisse (siehe Tabelle 5.5). Nach den Kopfdaten sind die Ereignisdaten (siehe Abschnitt 5.1) sequentiell abgelegt.

Tabelle 5.5. Kopf des Dateiformats MMSE .

Bytes	Datentyp	Beschreibung
0...3	char*	Dateitypidentifikation
4...27	6x float (32 bit)	Simulationsraum
28...51	6x float (32 bit)	Beschnittbereich
52...59	uint64_t	Ereignisanzahl
60...63	float	Maximalzeit

6. VISUALISIERUNG

Dieses Kapitel beschreibt die Erstellung einer Taxonomie, die für eine systematischen Zuordnung der Ereignisparameter zu *visuellen Variablen* dient. Für die Überprüfung der Zuordnung dienen Mockups, die mit einem sogenannten Glyphprototypen erstellt werden. Weiterhin werden die Umsetzung der Glyphgestaltung als auch der Glyphvisualisierung mit einem Renderer beschrieben sowie die Visualisierung der Cluster im Partikelraum.

Zur Einordnung der Darstellung in die Taxonomie von Gleicher et al (siehe Abschnitt 3.3) werden der Partikel- und der Ereignisraum als zwei separate Objekte betrachtet. Dazu werden die Begriffe *Partikelobjekt* und *Ereignisobjekt* eingeführt; nicht zu verwechseln mit den Strukturen aus Abschnitt 5.1. Der erste Begriff steht für die Visualisierung der Partikel und beinhaltet den gesamten Verlauf von 150 Zeitschritten. Der zweite steht für die Visualisierung der Ereignisse für ebenfalls alle Zeitschritte.

6.1. GRUNDLAGEN

Zu Beginn werden zwei wiederkehrende Begriffe eingeführt und anschließend eine Vorbetrachtung zur Gestaltung von dreidimensionalen Glyphen durchgeführt.

PRÄATTENTIVE WIRKUNG VISUELLER VARIABLEN

Zu *visuellen Variablen* zählen unter anderem Anordnung, Farbe, Form, Größe, Maserung, Orientierung, Position, Schärfe und Transparenz (siehe Abschnitt 3.3). Sie entfalten eine präattentative Wirkung, wenn sie einen Reiz erzeugen, der vom Nervensystem des Beobachters zwar wahrgenommen wird und einen Effekt erzeugt, jedoch nicht in das Bewusstsein dringt [FF06].

Vorteile der präattentiven Wahrnehmung sind eine parallele, schnelle Verarbeitung (kürzer als 250 Millisekunden (ms)), die nicht gehemmt werden kann. Nachteil ist, das sie oberflächlich sind. So sind zwar räumliche Verknüpfungen oftmals präattentiv, zum Beispiel die Bewegung verknüpft mit Farbe, Form oder 3D-Disparität, doch die meisten Verknüpfungen sind es nicht. Daher fördert die Nutzung möglichst weniger Variablen eine präattentive Informationsaufnahme [HE12].

Um eine präattentive Wahrnehmung nicht zu erschweren, wird eine Mehrfachzuweisung vermieden, wie beispielsweise die Zuweisung von sowohl der Farbe, als auch der Form zur Zeit des Ereignisses. Auch wenn Mehrfachzuweisungen scheinbar zur Verstärkung beitragen, genügt eine *visuelle Variable* zur Kodierung eines Ereignisparameters.

GESTALTGESETZE

Die Gestaltgesetze, formuliert von Wertheimer und erweitert von Stephen Palmer und unter anderem visualisiert von Borgo et al. (siehe Abschnitt 3.3), 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.

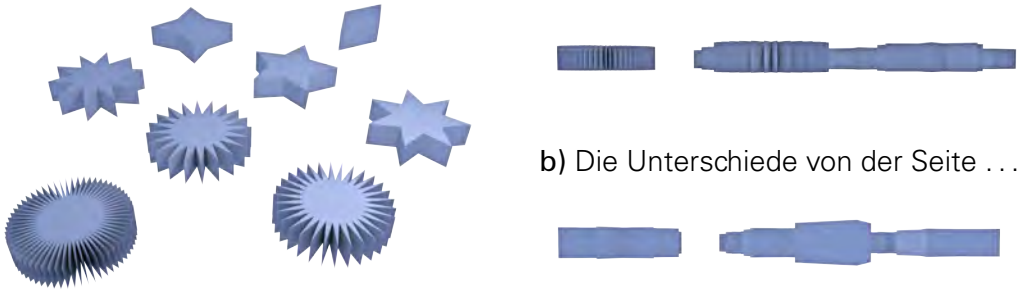
Die Partikel in der Animation werden auch durch das *Gesetz der gemeinsamen Bewegung* als zusammengehörig empfunden; sich gleichzeitig und in dieselbe Richtung bewegende Elemente werden als eine Gestalt wahrgenommen. Das spielt aufgrund der statischen Position der *Ereignisglyphen* für diese Arbeit keine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

Die zum Testen der Clustererkennung und zur Visualisierung der Cluster verwendete Einfärbung der Partikel (siehe Abschnitt 6.5) nutzt das *Gesetz der Kontinuität*. Es besagt, dass Reize, die eine Fortsetzung vorangehender Reize zu sein scheinen, als zusammengehörig angesehen werden.

DREIDIMENSIONALE 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 dreidimensionale 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 dreidimensionale Polytope und im engeren Sinne eine Teilmenge des dreidimensionalen Raums, 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 sternenförmigen



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. Erstellt in *Blender*.

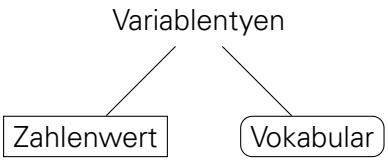


Abbildung 6.2. Variablentypen der *Strukturereignistaxonomie*

Prismen.

Andererseits erfüllen die sternförmigen *Kepler-Poinsot-Körper*, auch als „Sternpolyeder" bezeichnet, 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]

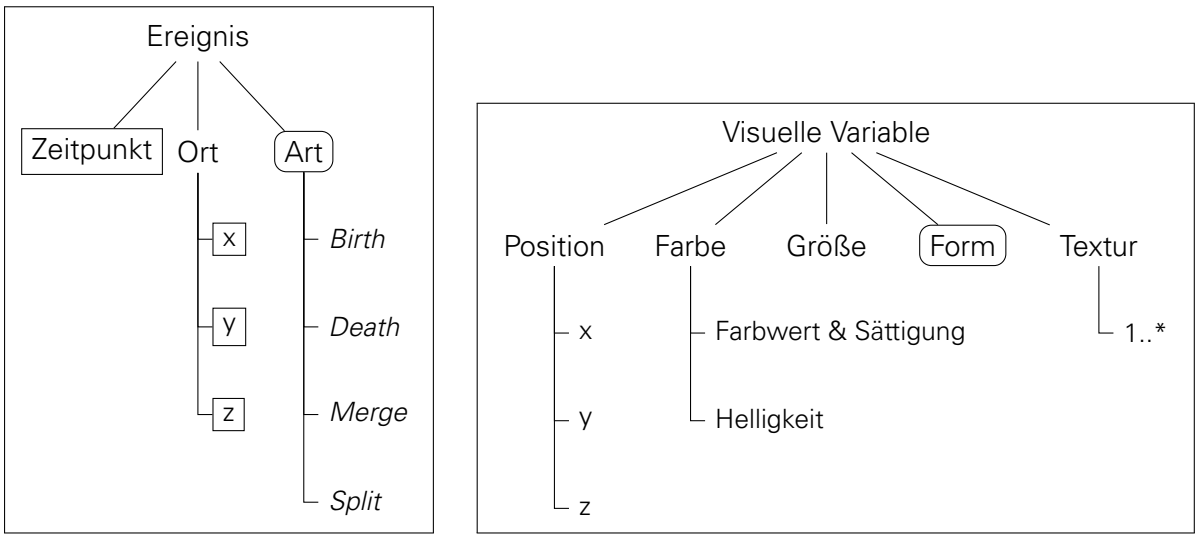
Somit gibt es mit der auf nichtprismatischen Polyedern basierenden Formgebung eine hohe Anzahl verschiedener Formen, die alle die Anforderung an eine einheitliche Größe, an einen aus allen Blickwinkeln ähnlichen Durchmesser und an die Wiedererkennbarkeit gewährleisten. Lediglich die Unterscheidung innerhalb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

6.2. STRUKTUREREIGNISTAXONOMIE

Die ermittelten Ereignisse besitzen Parameter. Die Aufgabe der Visualisierung ist es, diese Parameter für den Nutzer verständlich mit Hilfe der Anwendung von *visuellen Variablen* (siehe Abschnitt 3.3) aufzubereiten und anzuzeigen. Um die möglichen Kombinationen zu ermitteln, wird eine *Strukturereignistaxonomie* vorgeschlagen.

In dieser werden den Ereignisparametern und *visuellen Variablen* die beiden Variablentypen *Zahlenwert* und *Vokabular* zugeordnet (siehe Abbildung 6.2), die eine Überprüfung der Kompatibilität zwischen Parameter und Variable ermöglichen. Der Typ *Zahlenwert* steht dabei für die Nutzung eines kontinuierlichen Zahlenraumes, während der Variablentyp *Vokabular* eine diskrete, beschränkte Zuordnung mit voneinander disjunkten Elementen beschreibt.

Wie in Tabelle 5.3 zu sehen, besitzt ein Ereignis drei Parameter, die jeweils zwischen ein bis drei Freiheitsgraden aufweisen. Die Position hat drei Freiheitsgrade und ist ein *Zahlenwert*, der Zeitpunkt hat einen Freiheitsgrad und ist ebenfalls ein *Zahlenwert*, wohingegen die Ereignisart den Typ *Vokabular* besitzt.



a) Ereignisparameter sowie Parameterattribute mit zugehörigem Variablentyp.

b) Visuelle Variablen und ihre Attribute. Nicht gekennzeichnete Elemente sind für beide Variablentypen geeignet. Der Asterix steht für eine begrenzte Anzahl an Freiheitsgraden.

Abbildung 6.3. Schema der *Strukturereignistaxonomie* mit Datenparametern und der visuellen Attribute. Spitze Ecken entsprechen dem Typ *Zahlenwert*, abgerundete dem Typ *Vokabular* (Abbildung 6.2).

VISUELLE VARIABLEN EINES EREIGNISELEMENTS

Die *visuellen Variablen* können sowohl als fließender *Zahlenwert* als auch als festes *Vokabular* umgesetzt werden. Für diese Taxonomie werden sieben der elf von Borgo et al. gezeigten, *visuellen Variablen* verwendet (siehe Abschnitt 3.3). Die Variablen werden teilweise zusammengefasst. So werden Farbwert, Sättigung und Helligkeit als Attribute deklariert und zur neuen, *visuellen Variable* Farbe hinzugefügt. Die Anzahl der Attribute wird analog zu den Ereignisparametern als Freiheitsgrad bezeichnet.

Da Farbwerte bei geringer Sättigung nicht mehr voneinander unterschieden werden können, werden die beiden Attribute, Farbwert und Sättigung, zu einem zusammengefasst. Die Sättigung verbleibt dabei auf ihrem Maximalwert.

Dadurch ergibt sich die in Abbildung 6.3 gezeigte Taxonomie. Die Ereignisparameter beziehungsweise *visuellen Variablen* ohne Attribute oder des Typs *Vokabular* besitzen einen Freiheitsgrad von eins.

Obwohl bei der Form mehrere Freiheitsgrade denkbar wären – etwa eine Kombination aus Anzahl von Ecken mit einem variablen Abstand dieser vom Körperzentrum – wird die Variable auf einen Freiheitsgrad von eins beschränkt. Dies liegt daran, dass der variable Abstand der Ecken mit einer anderen *visuellen Variable* kollidiert, mit der Größe. In der dreidimensionalen Darstellung werden für die Form die in Abschnitt 6.1 beschriebenen Polyeder verwendet. Da die Anzahl an Polyedern begrenzt ist und die Unterscheidungsmöglichkeit mit zunehmender Flächenanzahl sinkt, ist die Form nur sehr eingeschränkt als Zahlenwert verwendbar. Dies gilt für die zweidimensionale Darstellung, etwa mittels Polygonen genauso. Daher wird sie in der Taxonomie auf den Typ *Vokabular* beschränkt (siehe Abbildung 6.3b).

Die Textur ist eine eigenständige, visuelle Variable, auch wenn sie sich mit der Farbe überschneidet. Da die Textur jedoch lokale Farbunterschiede innerhalb eines Elements kennzeichnet, kann sie parallel dazu verwendet werden. Sie vereint die in der Arbeit von Borgo et al. genannten *visuellen Variablen* Orientierung, Maserung und Anordnung (siehe Abschnitt 3.3) und

enthält mindestens einen Freiheitsgrad. Dieser kann jedoch durch die lokale Bestimmung weit höher liegen, etwa wenn verschiedene Bereiche der Textur für unterschiedliche Datenparameter verwendet werden.

Der Freiheitsgrad einer *visuellen Variable* gibt die Anzahl der Datenparameter an, die ihr zugewiesen werden kann. Umgekehrt zeigt der Freiheitsgrad eines Datenparameters die Anzahl der benötigten *visuellen Variablen*.

Bei der Bestimmung der Attribute wurde darauf geachtet, ein umfassendes Spektrum der Gestaltgesetze nutzen zu können (siehe Abschnitt 6.1). Alle *visuellen Variablen* 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 der Position mit den anderen *visuellen Variablen* 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 der Position angewendet werden, falls durch entsprechende Zuweisung die Elemente nah beieinander lägen und sie eine geschlossene Form bilden. Mit den hier definierten Variablen können somit sechs Gestaltgesetze genutzt werden.

Die Opazität wird aufgrund des Ziels der überlagerten Darstellung von *Ereignisobjekt* und *Partikelobjekt* in der Taxonomie trotz Vorteilen bei überlagerter Darstellung innerhalb eines Objekts (siehe Abschnitt A.3) ausgeklammert. Sie würde die Sichtbarkeit der Glyphen im Vergleich zur Partikeldarstellung oder des Hintergrundes zu sehr einschränken.

Animationen bieten weitere visuelle Variablen 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 Variablen* lassen sich durch den Betrachter unterschiedlich gut unterscheiden. Solche mit höherer Unterscheidungsgüte sollten bevorzugt gewählt werden. Form, Helligkeit und Textur besitzen eine geringere Güte. Daher sind sie besser zur Verwendung mit dem Typ *Vokabular* geeignet, denn die Anzahl der Varianten ist im Vergleich zu einer kontinuierlichen (unendlichen) Menge an Varianten, wie es beim Typ *Zahlenwert* der Fall ist, gering. Bei einem *Vokabular* können die Varianten einer *visuellen Variable* weit auseinanderliegen, was die Unterscheidungsmöglichkeit verbessert.

PROBLEM DER LOKALEN UND TEMPORALEN HÄUFUNG

Die Anzahl der Ereignisse, die denselben Ort oder denselben Zeitpunkt aufweisen können, ist unbekannt. Schon eine geringe Anzahl identischer Orte oder Zeitpunkte kann je nach Zuordnung der *visuellen Variablen* zu einer überlagerten, nicht mehr unterscheidbaren Darstellung der Ereignisglyphen führen.

Daher sollte eine *visuelle Variable* reserviert werden, um eine solche Anhäufung von Ereignissen anzuzeigen und programmatisch zu behandeln. Formal ausgedrückt bedeutet diese Reservierung das Hinzufügen der Häufung zu den Datenparametern der *Strukturereignistaxonomie*.

Bei der Visualisierung der Häufung kann die intuitive Wirkung der visuellen Attribute genutzt werden. So würde groß, opaque und dunkel eine starke Häufung symbolisieren und klein, transparent und hell hingegen eine niedrige.

ZUWEISUNG DER VISUELLEN VARIABLEN

Aufgrund der Zielstellung der Arbeit, die Strukturereignisse im Raum der Partikel anzuzeigen, wird die Position dem Ortsparameter des Ereignisses als *visuelle Variable* zugewiesen. Darüber hinaus wird jedem Ereignisparameter nur ein visuelles Attribut zugeordnet, um eine Verminderung der präattentiven Wahrnehmung durch Mehrfachzuweisungen zu vermeiden (siehe Abschnitt 6.1).

Tabelle 6.1. Zuweisungsmatrix der *visuellen Variablen* zu den Ereignisparametern. Berücksichtigt werden ausreichender Freiheitsgrad, Variablentyp (Typ) sowie Regeln (fest, vergeben). + steht für das Intervall $[1, \infty)$.

Parameter / Variable	Freiheitsgrad	Zeitpunkt	Ereignisort	Ereignisart	Häufung
		1	3	1	1
Position	3	× (vergeben)	fest	× (vergeben)	× (vergeben)
Größe	1	✓	×	✓	✓
Farbe	2	✓	×	✓	✓
Form	1	× (Typ)	×	✓	× (Typ)
Textur	+	✓	× (vergeben)	✓	✓

Weiterhin werden die Anzahl an Freiheitsgraden sowie die Kompatibilität der Variablentypen berücksichtigt und die Häufung als Parameter aufgenommen, so dass sich die in Tabelle 6.1 aufgeführte Zuweisungsmatrix ergibt. Der Ereignisart stehen durch ihren Typ als *Vokabular* die meisten *visuellen Variablen* zur Verfügung. Zu beachten ist, dass die Farbe und die Textur gleichzeitig mehrere Parameter visualisieren können.

GLYPHPROTOTYP

Zur Überprüfung der Zuweisung von Datenparametern zu den *visuellen Variablen* dienen Mockups. Aufgrund der Vielzahl der in Abschnitt 6.2 genannten Kombinationsmöglichkeiten wird eine programmatische Erstellung der Mockups gewählt. Diese nutzt die in Abschnitt 6.1 beschriebenen Grundlagen für die Darstellung der dreidimensionalen Glpyhen. Einzelheiten zur Funktionsweise werden in Abschnitt A.3 dargelegt.

Da die für die Simulation gewählte Kamera in *MegaMol* perspektivisch ist, werden die Aufnahmen im *Glyphprototyp* ausschließlich in einer perspektivischen Ansicht vorgenommen, um die Vergleichbarkeit zum Plugin zu wahren. Als Lichtquelle wird ein direktionales Licht verwendet.

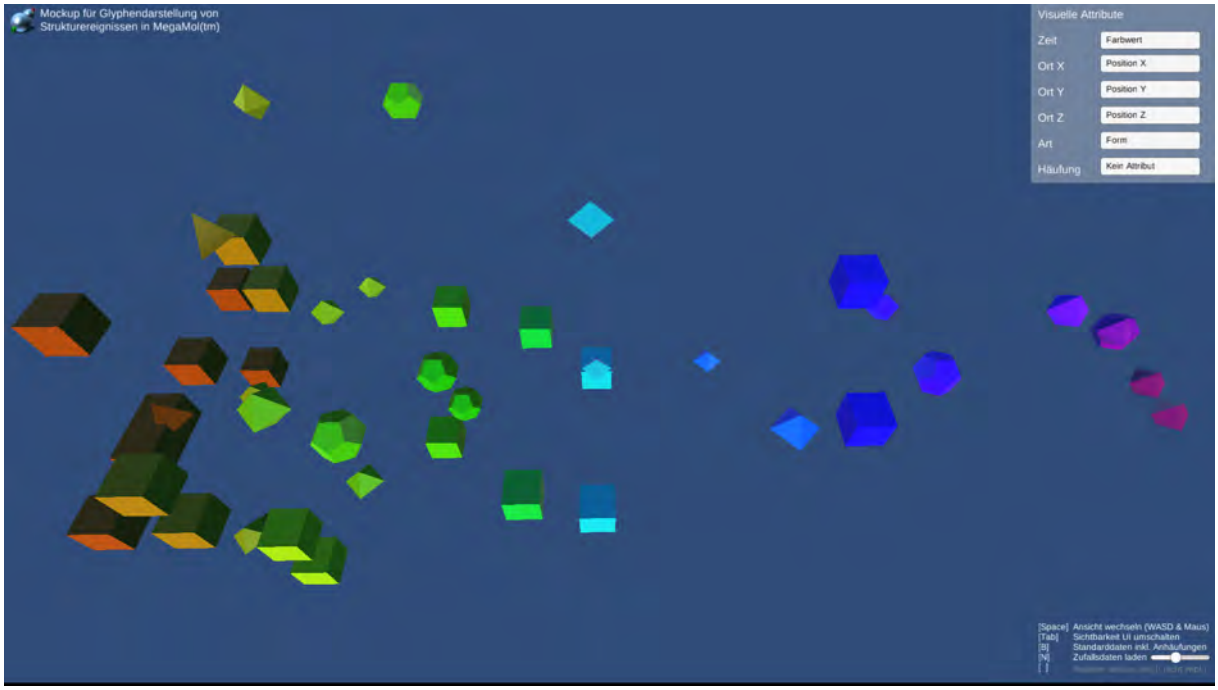
6.3. GESTALTUNG DER GLYPHEN

„[...] so that the first acceptable interpretation to occur to the hearer is the one [the speaker] intended to convey." [Inf01, S. 64]

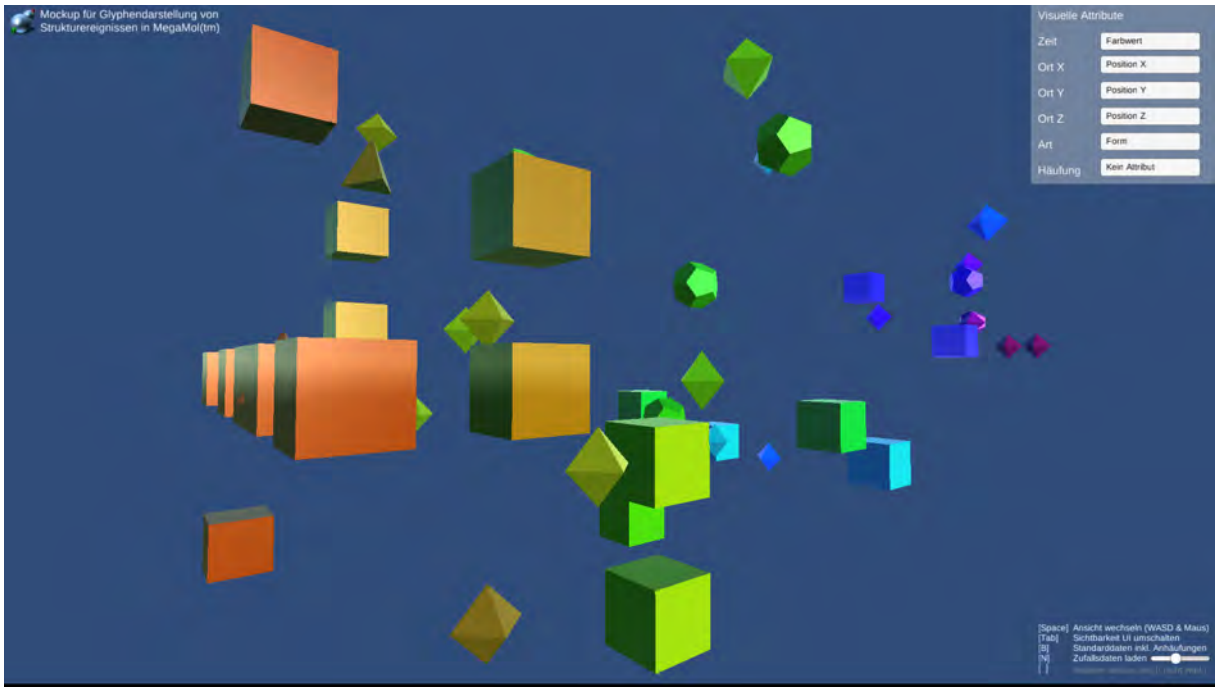
Die *Ereignisglyphen* sind die Schnittstelle zwischen den berechneten Ereignisdaten und dem Anwender. Die *Strukturereignistaxonomie* stellt die Grundlage für ihre Gestaltung dar. Die Untersuchung der Eignung von dreidimensionalen Glyphen erfolgt mit Hilfe des *Glyphprototypen*.

DIMENSION DER GLYPHDARSTELLUNG

Trotz der in Abschnitt 6.1 durchgeführten Vorüberlegungen sind die in den Mockups verwendeten Polyeder immer noch schlecht voneinander zu unterscheiden und ihre Erscheinung ist stark vom Blickwinkel der Kamera abhängig. Damit wird die präattentive Wirkung stark gesenkt, da zusätzliche kognitive Prozesse notwendig sind, um die Glyphen beim Wechsel des Blickwinkels mit den vorangegangenen Eindrücken zu verknüpfen Aufgrund der räumlichen Ausdehnung sind Überschneidungen zu beobachten (siehe Abbildung 6.4a), was bei der *Überlagerung* mit dem *Partikelobjekt* zusätzlich zur Verdeckung naheliegender Partikeln führen wird. Verhindert werden kann das durch sehr kleine Glyphen, allerdings können so die Formen nicht mehr differenziert werden. Dies wird in Abbildung 6.5 verdeutlicht. Auch ist dort die richtungsabhängige Beleuchtung der dreidimensionalen Glyphen zu sehen, was die Behandlung der Helligkeit als separate



a) Sicht von schräg oben. Links im Bild ist eine Überschneidung des orangenen Tetraeders (*Death*) mit dem orangenen Hexaeder (*Merge*) zu sehen.



b) Sicht von der Seite.

Abbildung 6.4. Mockups des Glyphprototypen mit Standarddaten (siehe Abschnitt A.3). Der Ereignisart ist die Form zugewiesen, der Zeit der Farbwert. Deutlich ist eine unterschiedliche Helligkeit der Flächen innerhalb der Polyeder zu erkennen.

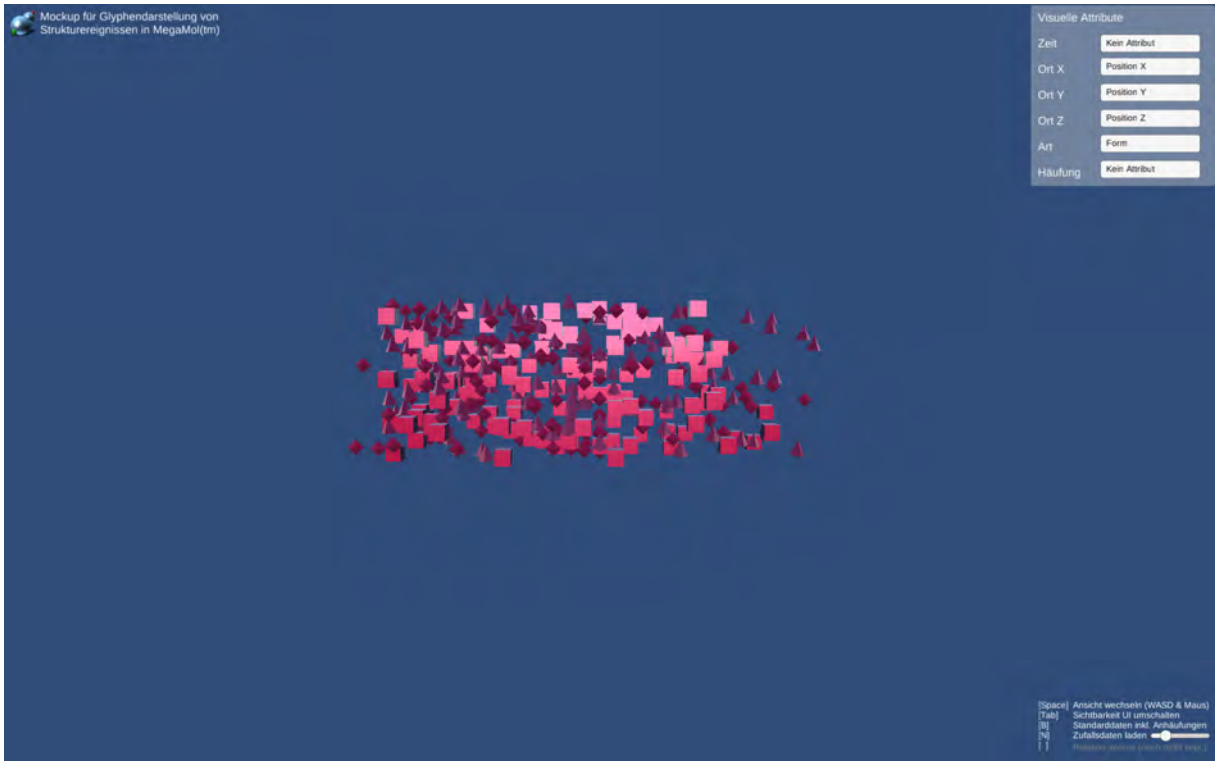


Abbildung 6.5. Mockup des Glyphprototypen aus großer Entfernung mit zufällig erzeugten Daten (siehe Abschnitt A.3). Der Ereignisart ist die Form zugewiesen. Deutlich ist zu sehen, dass die Helligkeit der Glyphen von der Beleuchtung abhängig ist wegen der mit unterschiedlichen Winkeln zur Kamera und zur Lichtquelle geneigten Flächen.

visuelle Variable erschwert. Bei der Applikation einer Textur auf die Glyphoberfläche kommt es durch die unterschiedliche, blickwinkelabhängige Darstellung der einzelnen Polyederflächen zu Verzerrungen der Textur und damit ebenfalls zur Wahrnehmungsbeeinträchtigung (dieser Sachverhalten ist mit der aktuellen Implementation des *Glyphprototyp* nicht darstellbar).

Aufgrund der vier genannten Gründe – Blickwinkelabhängigkeit der Form und der Textur, Abhängigkeit der Helligkeit der einzelnen Polyederflächen vom Winkel zur Lichtquelle und Kamera sowie Überschneidungsprobleme – wird von einer dreidimensionalen Formgebung der Glyphen abgesehen. Deswegen sind weitere Untersuchungen mit dem *Glyphprototypen* über die Zuweisungsmöglichkeiten der *visuellen Variablen* für die vorliegende Arbeit nicht relevant.

So wird eine zweidimensionale Darstellung gewählt, die immer zur Kamera ausgerichtet wird (siehe Abschnitt 6.4). So können die beiden Nachteile der Blickwinkelabhängigkeit beseitigt werden und es gibt keine Flächen im selben Glyph, die unterschiedlich angeleuchtet werden.

FORM UND ABGESCHLOSSENHEIT

Es wird eine annähernd rechteckige Form verwendet, um einen Kontrast zur runden Gestalt der kugelförmigen Partikelvisualisierung zu schaffen. Falls der innere Bereich des Glyphs dem Nutzer keinen Anhaltspunkt für die Orientierung geben kann, wird die Unterkante als eine stumpfe, nach unten zulaufende Spitze gestaltet (siehe Abbildung 6.6). Ihre auffällige Wirkung wird bei der Positionierung der Glyphen (siehe Abschnitt 6.4) genutzt. Die Nutzung eines nach unten zeigenden, gleichseitigen Dreiecks wäre ebenfalls denkbar, allerdings ist der zur Verfügung stehende Platz für die innere Gestaltung wesentlich geringer. Außerdem könnten ungewollte Assoziationen zu Verkehrschildern geschehen.

Die Außenkante des Glyphs ist mit einem Rahmen versehen, um das *Gesetz der Geschlossenheit* zu berücksichtigen und dabei im Inneren Platz für die Visualisierung der Ereignisart und -zeit zu erhalten. Dieser Rand wird nicht zu dünn gestaltet, um Aliasing an hochfrequenten (schmalen, nah beieinanderliegenden) Kanten und das damit verbundene Flimmern während der Anzeige zu verringern.

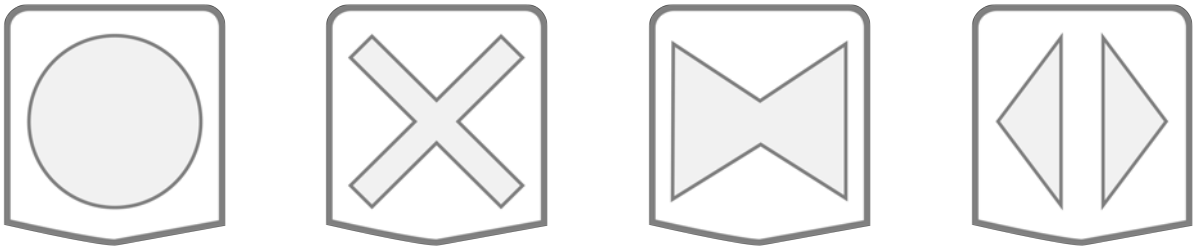
KODIERUNG DER EREIGNISART

Es werden zwei symbolische Darstellungsvarianten (siehe Abschnitt 3.3) entworfen. Die abstrakte Variante hat keinen Bezug zum Inhalt, wogegen die narrative Variante eine Konnektivität zu den Begriffen der Strukturereignisse beinhaltet. So steht bei der zweiten Variante der Asterisk für Geburt, ein Kreuz für den Tod sowie eine umschließende als auch eine nach außen geöffnete Klammer für Verschmelzung beziehungsweise Trennung (siehe Abbildung 6.6b). Ein Ziel der für die Auswertung entwickelten Umfrage stellt die Prüfung da, ob beide Varianten eine intuitive Deutung zulassen und ob eine der beiden leichter interpretiert werden kann.

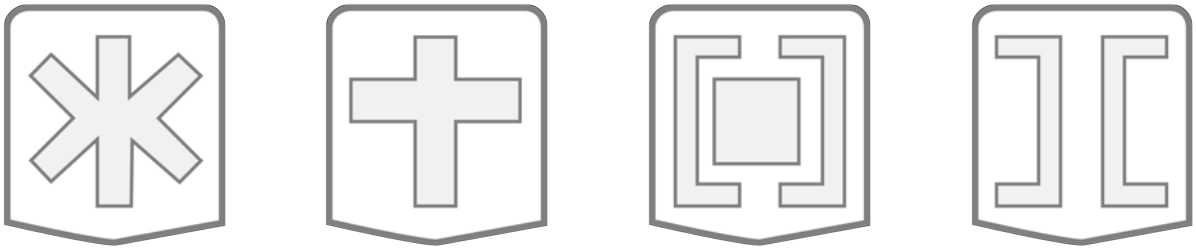
Die Kodierungsvarianten könnten mit den in Tabelle A.1 vorgeschlagenen Werten für Farbwert und Helligkeit erweitert werden. Allerdings werden beide Farbattribute für die Kodierung der Zeit reserviert und eine Doppelbelegung für die Ereignisart wird vermieden. Denn die *Strukturereignistaxonomie* fordert eine eindeutige Zuweisung der visuellen Attribute, so dass bei einer Doppelbelegung eine automatische Änderung der Zuweisung des jeweils anderen Parameters erfolgen muss. Solch eine automatische Änderung wird im *Glyphprototyp* unterstützt und sie führt bei der Bedienung zu Verwirrung. Daher wird bei der Implementation des *MegaMol*-Plugins darauf verzichtet.

KODIERUNG DER ZEIT

Die Visualisierung der Zeit wird unter Verwendung von Farbwert, Helligkeit oder Textur durchgeführt. Scheinbar ist die Textur zusammen mit der Ereignisart doppelt belegt, jedoch unterstützt diese *visuelle Variable* mehrere Freiheitsgrade (siehe Abschnitt 6.2). Gemäß der *Strukturereignistaxonomie* wird die Sättigung unveränderbar auf den Maximalwert eingestellt.



a) Abstrakt symbolische Darstellungsvariante



b) Narrativ symbolische Darstellungsvariante

Abbildung 6.6. Kodierung aller vier Ereignisarten mit zwei unterschiedlichen, symbolischen Darstellungsvarianten. Von links nach rechts sind Glyphen für *Birth*, *Death*, *Merge* und *Split* abgebildet. Außerdem ist die Glyphform sichtbar.

Alle drei *visuellen Variablen* ermöglichen einen uneingeschränkten Einsatz des Typs *Zahlenwert*, was für die Darstellung der Zeit von Vorteil ist.

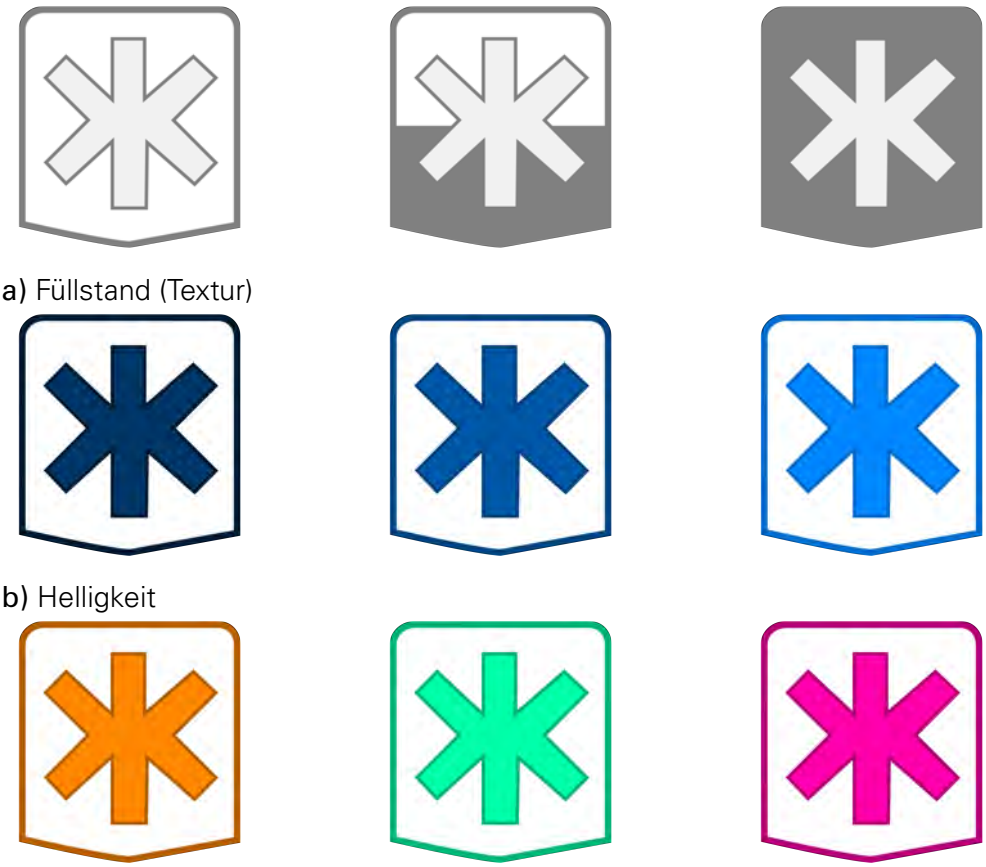
Es wird zu prüfen sein, welche der drei Varianten am Besten für die Beurteilung des Zeitfortschritts geeignet ist. Darüber hinaus wird mit der Umfrage für jede Variante die Unterscheidbarkeit der Zeitschritte voneinander ermittelt.

6.4. RENDERER UND SHADER

Der Glyph wird immer zur Kamera ausgerichtet. Dazu wird innerhalb des Geometrieshaders im Modelviewkoordinatensystem ein Quad aufgespannt. Das so entstandene Objekt wird unter dem Begriff 3D-Sprite beziehungsweise *Billboard* eingeordnet und der Shader damit als *Billboard*-Shader bezeichnet. Er ist zur Wahrung der Kompatibilität mit den *MegaMol* Kernmodulen in OpenGL (OGL) 2.1 geschrieben. *MegaMol* stellt eine direktionale Lichtquellen bereit und sie kann automatisiert mit der Kamera ausgerichtet werden, um die Blickwinkelabhängigkeit der Beleuchtung zu eliminieren. Die Färbung der Glyphen erfolgt jedoch unabhängig von der Lichtquelle, so dass eine identische Wirkung gewährleistet werden kann.

Zur Kodierung der Ereignisart sowie der Zeit als Textur werden Bilder im Portable Network Graphics (PNG)-Format mittels *lodepng* geladen und zur Anzeige an den *Billboard*-Shader weitergeleitet. In seinem Fragmentshader wird die Farbe des Fragments in Abhängigkeit von den Parametereinstellungen des Renderers gesetzt. Als weitere externe Bibliothek wird die OpenGL Mathematics (glm)-Bibliothek genutzt, da dadurch die Vektordatentypen in *C++* und in *GLSL* identisch behandelt werden können.

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 *Billboard*-Shader zur Darstellung des Zeitpunktes eines Ereignisses auf bestimmte Farben in der Textur angewiesen ist. So wird die Farbe lila (rgb 255, 0, 255) abhängig von den Einstellungen der Zeitkodierung dynamisch ersetzt. Bei der Auswahl der Zeitkodierung als Farbe oder Helligkeit wird durch den Shader ein heller Grauton in die lilagefärbten Bereiche geschrieben und alle anderen Pixel der Textur werden mit einer von der Zeit abhängigen Farbe ersetzt, bei der Auswahl als Textur wird



a) Füllstand (Textur)

b) Helligkeit

c) Farbwert

Abbildung 6.7. Kodierung der Ereigniszeit in drei Varianten. Von links nach rechts wird der Zeitfortschritt von 0% bis 100% durch die Glyphen visualisiert.

der Bereich abhängig von der Zeit mit Dunkel- oder Hellgrau überschrieben und die anderen Bereiche der Textur bleiben unangetastet (siehe Abschnitt A.5). Zunächst wird nur die narrativ symbolische Darstellungsweise für die Ereignisart implementiert.

POSITIONIERUNG DER GLYPHEN

Durch die in der Aufgabenstellung formulierte Bedingung, die Daten der Partikel als auch die Daten der Strukturereignisse im selben Raum zu visualisieren, wird die überlagernde Darstellung angewandt (siehe Abschnitt 3.3) und die Position der *Ereignisglyphen* wird an die Positionen der entsprechenden Strukturveränderungen im Partikeldatensatz gesetzt. Dazu wird der Ort des *Wurzelpartikels* eines betroffenen Clusters verwendet. Die Positionen des oder der *Partnercluster* sind dabei nicht relevant, was bei großen Clustern zu Problemen der Lokalisierung des Ereignisses führen könnte. Eine Implementation, die in Abhängigkeit der Parameter der *Ereignisheuristik* sowie der *Clustergröße* der betroffenen (Partner-)Cluster eine gewichtete Positionierung entlang der Verbindungslinie zwischen den *Wurzelpartikeln* vornimmt, wäre für eine Weiterentwicklung wünschenswert.

Im Verhältnis zur Position des Ereigniselements wird der *Ereignisglyph* genau oberhalb platziert. Aufgrund der in Abschnitt 6.3 beschriebenen äußeren Form wird angenommen, dass der Nutzer das Ereignis intuitiv unterhalb des Glyphen einordnet.

PARAMETERSLOTS

Parameterslots bieten dem Nutzer die Möglichkeit, während der Laufzeit des Programms Parameter ändern zu können.

Um dem in Abschnitt 6.2 beschriebenen Problem der lokalen und temporalen Häufung zu begegnen, wird unter Berücksichtigung der in Tabelle 6.1 aufgeführten Zuweisungsmöglichkeiten ein änderbarer Parameter für die Glyphgröße bereitgestellt. Somit wird der Häufung die Größe als visuelles Attribut zugewiesen. Weiterhin unterstützt die veränderbare Glyphgröße den Wechsel zwischen der Makro- und Mikrosicht, indem der Parameter je nach Benutzerpräferenzen bei der Makrosicht große und bei der Mikrosicht kleine Werte annehmen kann.

Ein zweiter Parameterslot wurde zur benutzerdefinierte Kontrolle über die Visualisierung der Zeit mit einer der drei in Abschnitt 6.3 vorgestellten Varianten implementiert. Auch gibt es einen dritten und vierten Slot für die Ereignisposition sowie die -art. Zwar bieten diese beiden nur jeweils eine Auswahlmöglichkeit, jedoch ist das *StaticRenderer Modul* dadurch leicht erweiterbar.

Ein fünfter Slot ermöglicht das gezielte Einblenden eines bestimmten Strukturereignisses oder aller.

PARALLELISIERUNG DER ANZEIGEBERECHNUNGEN

Bei der Zuweisung der *visuellen Variablen* zu den Ereignisparametern sind Berechnungen notwendig. Diese bewirken, dass die Anzeige bei einer hohen Anzahl von Ereignissen verzögert erfolgt, etwa nach Änderungen von Parametern des *StaticRenderer Moduls*. Da die Berechnungen 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. Eine andere Möglichkeit wäre gewesen, diese Berechnungen in den Geometrieshader der Grafikkarte zu verlegen.

6.5. VISUALISIERUNG DER CLUSTER

Die *MegaMol* Molekularsimulation der Partikel nutzt entweder nur die Position als *visuelle Variable* oder zusätzlich die Farbe. Beide werden zur Visualisierung der Cluster verwendet.

Um die Qualität der Clusterberechnung zu untersuchen und dem Nutzer Vergleichsmöglichkeiten mit den Ereignissen an die Hand zu geben, wird allen Partikeln desselben Clusters dieselbe Farbe zugeordnet. Im ersten berechneten Zeitschritt erfolgt diese Farbzuweisung für jeden Cluster zufällig oder anhand von Eigenschaften des *Wurzelpartikels* (siehe Abschnitt 5.3), in den darauffolgenden Zeitschritten wird die Farbe desjenigen Partnerclusters des vorhergehenden Zeitschrittes übernommen, der die meisten gemeinsamen Partikel aufweist. Sollte der Cluster keinen *Partnercluster* besitzen, wird eine neue zufällige Farbe zugewiesen.

Weiterhin wird eine zufällige Färbung für jeden Zeitschritt implementiert, die als alternative Darstellungsform gewählt werden kann.

6.6. DARSTELLUNG DES ZEITVERLAUFS

Durch eine ähnliche Bewegungsrichtung und Geschwindigkeit sowie durch örtliche Nähe der Partikel lassen sich Zusammenhangskomponenten während der Animation erkennen. Dies nutzt die Clustereinfärbung mittels einer Farbvererbung, so dass die Wanderung von Clustern verfolgt werden kann.

Das *StaticRenderer Modul* bietet eine statische, gleichzeitige Anzeige der *Ereignisglyphen* für alle Zeitschritte bzw. für die folgenden oder vorhergehenden Zeitschritte im Vergleich zum aktuelle gewähltem. Um das Entstehen der Strukturereignisse besser zu verfolgen, wird auch ein dynamisches Einblenden während der Animation ermöglicht. Dazu wird der Renderer um einen Parameterslot für den Zeitmodus und einen weiteren erweitert, um eine benutzerdefinierte Reichweite an angezeigten Zeitschritten zu ermöglichen.

Unterstützt werden diese Varianten der zeitlichen Anzeige durch die visuelle Kodierung der Zeit durch die Glyphen (Abschnitt 6.3).

7. EVALUATION

Die Berechnungen werden auf einem dedizierten Rechner (siehe zweites System in Abschnitt A.6), unabhängig vom Entwicklungssystem, und für verschiedene Parametereinstellungen durchgeführt. Die für die Entwicklung und Auswertung notwendigen Berechnungen erstreckten sich aufgrund der in Abschnitt 7.5 gezeigten Berechnungsdauer über mehrere Wochen. Einzelheiten zum Aufbau der dafür benötigten Module sind in Abschnitt A.1 zu finden.

Die Berechnungen werden sowohl qualitativ als auch quantitativ bewertet und die Thesen der Visualisierung werden durch eine Umfrage sowohl unter Experten von Molekularsimulationen als auch unter Laien geprüft. Auch fließt ein Teil der Umfrage bei der Beurteilung des Ereignisverlaufes ein. Erklärungen in der Umfrage[1] wurden nur sparsam eingesetzt, um die Benutzer unvoreingenommen urteilen zu lassen und zu prüfen, ob die Visualisierung auch ohne Hintergrundwissen Rückschlüsse zulässt.

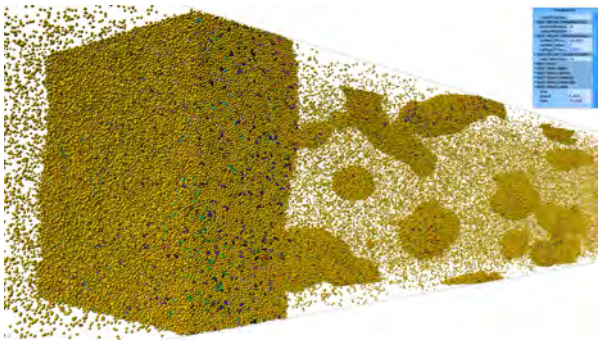
Die Erfassung quantitativer Daten war nicht Teil der Aufgabenstellung, wurde jedoch während der Entwicklung erforderlich aufgrund des experimentellen Charakters einer Heuristik und der Überprüfung der Funktionalität der beiden entwickelten CFD und SECC Algorithmen. Eine Analyse dieser Daten ergänzt die qualitative Betrachtung.

7.1. CLUSTERBILDUNG

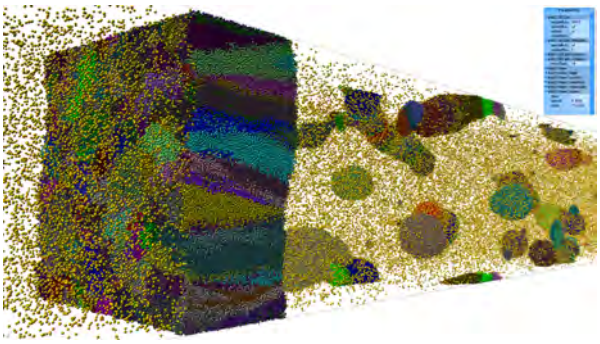
In Abbildung 7.1 ist deutlich zu sehen, dass bei zweifachem Partikelradius bei der Nachbarschaftserkennung nur sehr kleine Cluster gefunden werden. Dies liegt daran, dass bei dieser geringen Suchreichweite kaum Nachbarn vorhanden sind. Somit ist diese Radiuseinstellung zu klein, auch ist die Clusteranzahl mit zwei Größenordnungen über der Anzahl bei dreifachem Radius viel zu hoch (siehe Abbildung 7.2). Bei fünffachem Radius beträgt die Anzahl der gefundenen Cluster bei Zeitschritt 75 mit 722 Clustern nur noch 40% der Anzahl mit vierfachem Radius. Der Zeitaufwand liegt mit etwa 500 Sekunden nur 30% höher, was ein guter Kompromiss ist. So wird der fünffache Radius als Basiseinstellung für die Berechnung über den gesamten Simulationszeitraum verwendet und als ergänzende Daten der sieben- und der zehnfache Radius ebenfalls betrachtet.

Der große Einfluss des gewählten Radiusmodifikators bei der Nachbarschaftssuche ist in Abbildung 7.3 auch über den gesamten Simulationszeitraum sichtbar. Mit zunehmendem Suchradius nimmt die Anzahl an Clustern stark ab. Je größer die Suchreichweite desto höher ist die Anzahl der Nachbarn pro Partikel und auch Partikel, die etwas weiter entfernt sind, werden zu Nachbarn. Damit verringert sich effektiv die Anzahl an lokalen Maxima, denn wenn bei kleinerem Suchradius, wie dem fünffachen Partikelradius, zwei lokale Maxima nebeneinanderliegen, so werden diese bei größeren Suchradien während des SECC zu einem zusammengefasst, da die

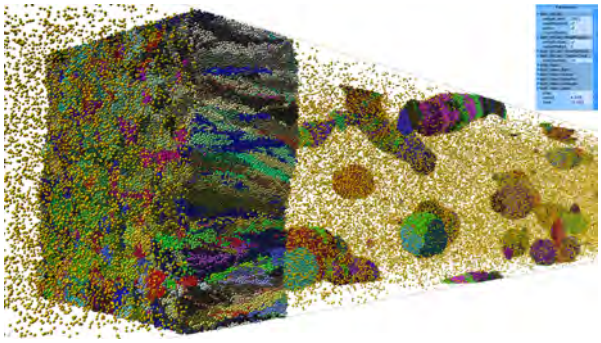
[1] http://www.instant.ly/s/wY5Ax



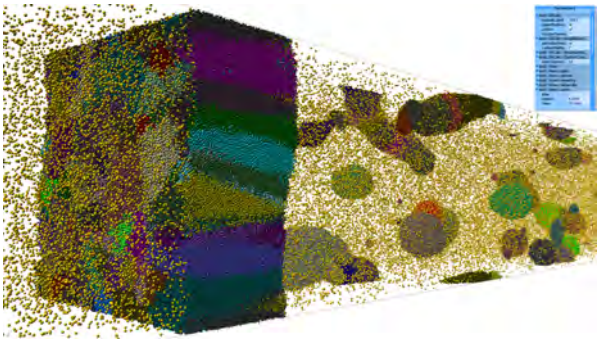
a) Zweifacher Radius.



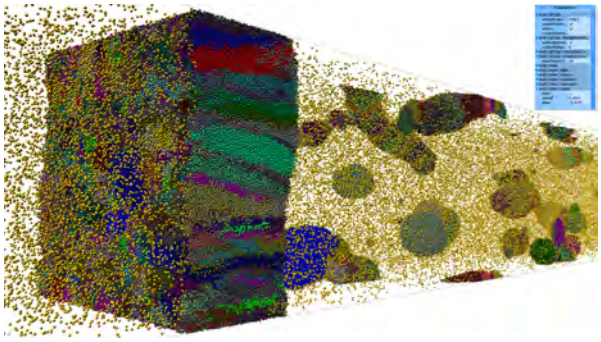
d) Fünffacher Radius.



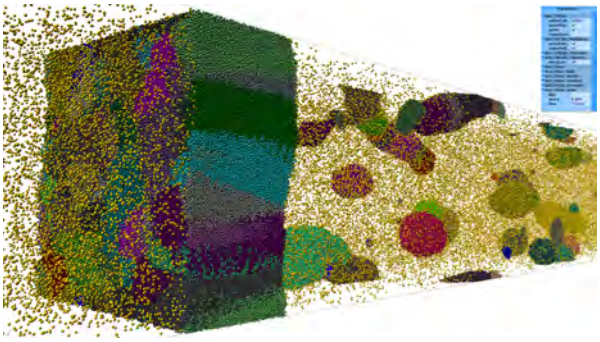
b) Dreifacher Radius.



e) Siebenfacher Radius.



c) Vierfacher Radius.



f) Zehnfacher Radius.

Abbildung 7.1. Clusterbildung abhängig von der Reichweite bei der Nachbarschaftserkennung. Dabei ist das Vielfache des Partikelradius' angegeben. Die Aufnahmen sind bei Zeitschritt 75 entstanden.

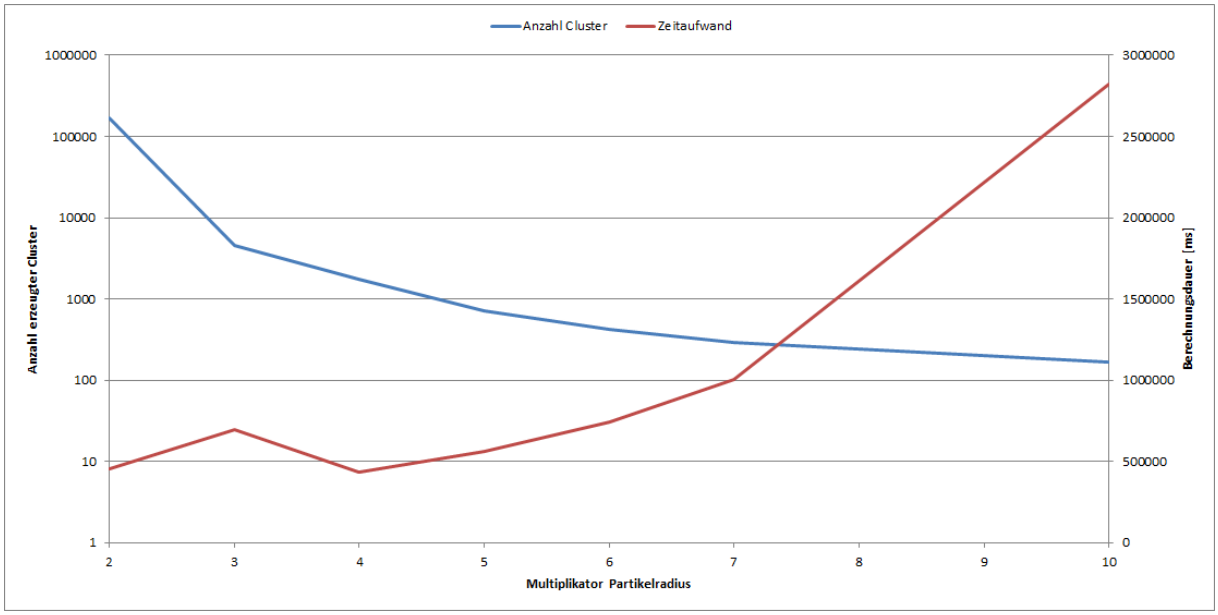


Abbildung 7.2. Logarithmische Anzahl der erzeugten Cluster sowie die Berechnungszeit in Abhängigkeit des Radius' der Nachbarschaftssuche bei Zeitschritt 75.

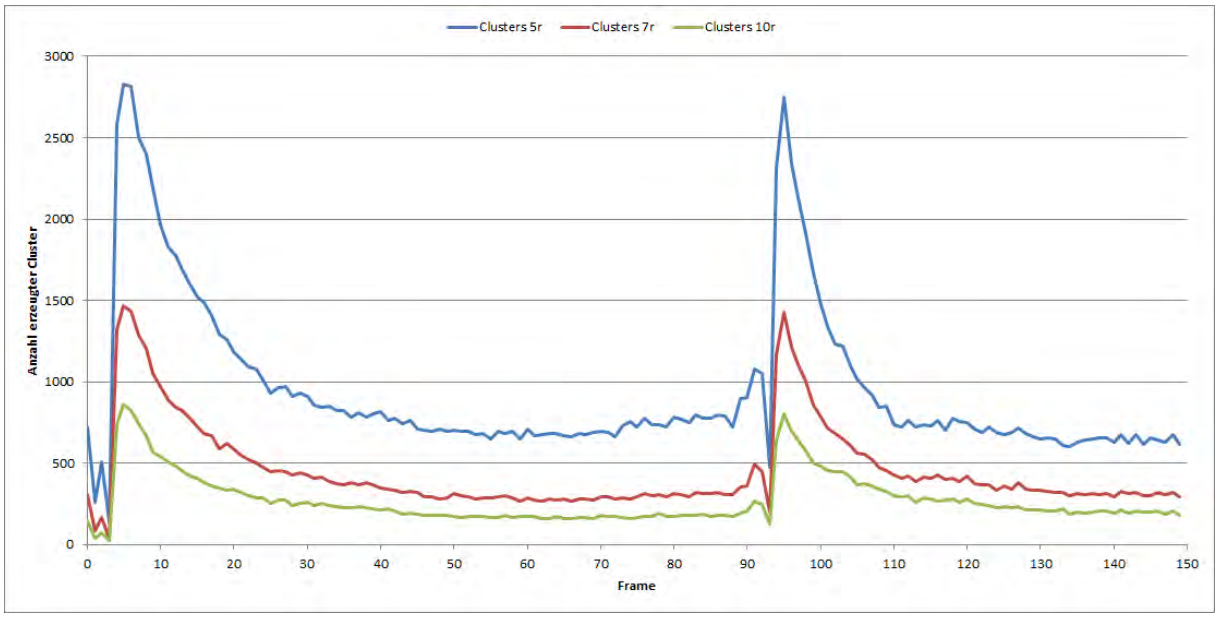


Abbildung 7.3. Anzahl der erzeugten Cluster in Abhängigkeit des *Partikelradiusmultiplikators* der Nachbarschaftssuche über alle Zeitschritte.

beiden Maxima dann selbst direkte Nachbarn sind und so das tiefere, höhere Maximum das kleinere verdrängt.

Weiterhin ist der große Einfluss der durch den Simulationsverlauf bedingten Strukturänderungen auf die Clusteranzahl zu erkennen. Dies ist durch die veränderte Topologie und damit der variierenden Anzahl an lokalen Maxima bedingt. Die Auswirkungen werden bei der Ereignisanalyse in Abschnitt 7.2 genannt.

7.2. QUANTITATIVE ANALYSE DER EREIGNISERMITTLUNG

Bei der Erkennung von *Birth* und *Death* Ereignissen wurde eine zusätzliche Erkennungsmöglichkeit über die Anteile an gemeinsamen Partikeln implementiert (siehe Parameter in Abschnitt 5.6). Die Überprüfung der *Birth*- und *Deathereignisse* mit Anteilen von 0-5% an gemeinsamen Partikeln ergab einen über alle Anteile identischen Wert (siehe digitale Logs des Determine Structure Events (DSE)). Das bestätigt die Stabilität der gewählten *Birth* bzw. *Death* Erkennung über Cluster ohne Partner. Auf diesen Parameter braucht daher nicht weiter eingegangen zu werden.

ABHÄNGIGKEIT VON DER NACHBARSCHAFTSSUCHE

In Abbildung 7.4 ist zu sehen, dass *Birth* und *Death* weit weniger von der Radiuseinstellung bei der Nachbarschaftssuche abhängen als *Merge* und *Split*. Das kommt daher, dass diese Ereignisse vor allem bei kleinen Tröpfchenstrukturen vorkommen, bei denen die Anzahl von Clustern aufgrund der geringen Partikelanzahl unabhängig der Radiuseinstellung ist.

Merge und *Split* hingegen zeigen eine starke Abnahme ihrer Anzahl mit zunehmendem Radius (Helligkeit in Abbildung 7.4b). Dies hängt mit der geringeren Clusteranzahl und der damit verbundenen, kleineren *Clustervergleichsmatrix* zusammen, wodurch bei denselben Ereigniserkennungsparametern weniger Ereignisse erkannt werden können.

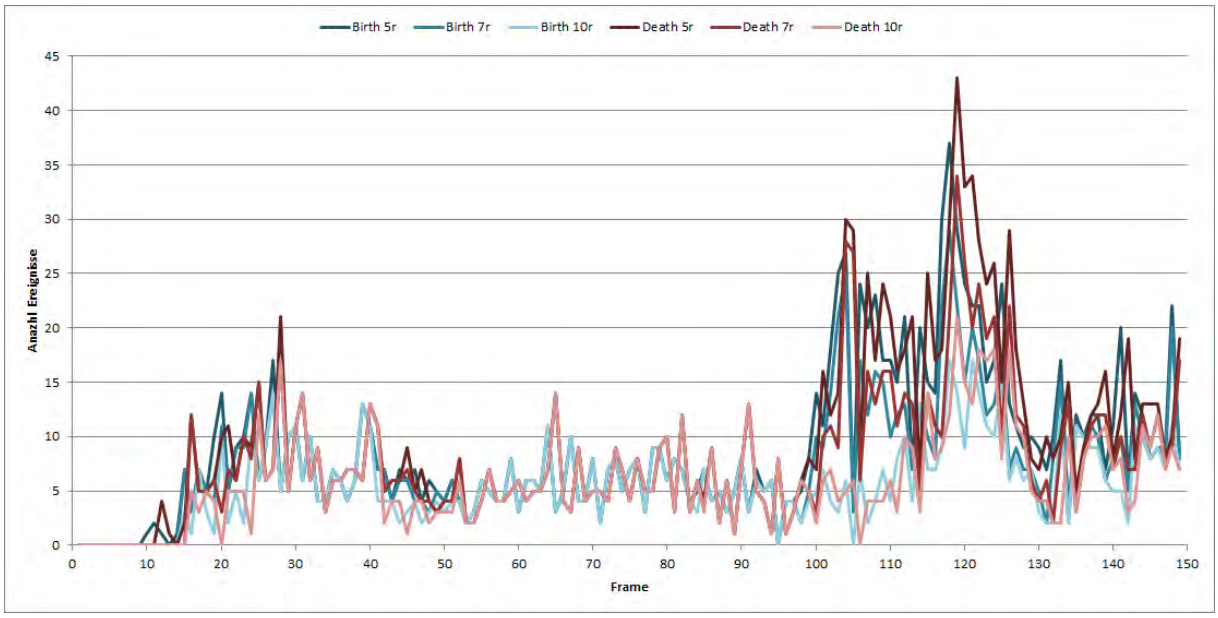
ABHÄNGIGKEIT VOM SIMULATIONSVERLAUF

Die Anzahl der Ereignisse ändert sich in Abhängigkeit vom Simulationsverlauf stark (siehe Abschnitt 4.1). Dies korreliert mit der Anzahl der Cluster (siehe Abbildung 7.3) und fällt insbesondere bei der Trennung und dem Zusammenstoßen der Flüssigkeitsfronten ins Auge. Auf *Merge* und *Split* hat dieses Geschehnis einen großen und unmittelbaren Einfluss, auf *Birth* und *Death* einen kleineren. Daraus kann abgeleitet werden, dass viele *Merge*- und *Splitereignisse* innerhalb der Fronten stattfinden, während *Birth* und *Death* eher bei Tröpfchen vorkommen. Weiterhin nehmen *Merge* und *Split* ab, je länger die Trennung bzw. der Zusammenstoß zurückliegen. Bei Zeitschritt 90 ist ihre Anzahl auf ein Fünftel im Vergleich zum Beginn der Frontentrennung zusammengeschrumpft. Dieses Verhältnis gilt für einen fünffachen Radius bei der Ermittlung der Nachbarn. Je höher dieser Suchradius gewählt wird, desto geringer sind die Differenzen bei der Anzahl der Strukturereignisse, da auch die Gesamtzahl der Ereignisse abnimmt.

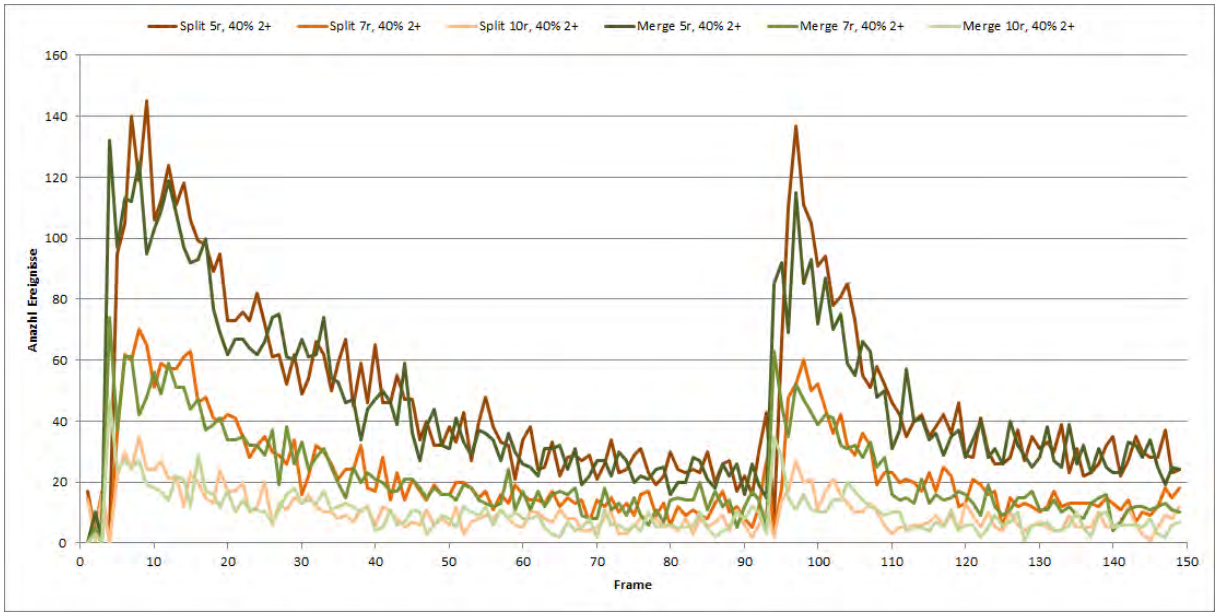
Die *Merge*- und *Splitereignisse* weisen eine ähnliche Anzahl bei der Trennung und beim Zusammenstoß auf. Die *Birth*- und *Deathereignisse* ebenfalls. Als Besonderheit nehmen diese Ereignisse nach dem Zusammenstoß nicht ab, sondern stark zu. Das könnte daran liegen, dass sich viele sehr kleine Cluster bilden oder Phänomene wie der „Nulltiefenwert *Einpartikelcluster*“ auftreten (siehe Abschnitt 7.3, Abschnitt 8.1). In Abschnitt 7.5 wird die Annahme über die Bildung kleiner Cluster untermauert.

ABHÄNGIGKEIT VON DEN EREIGNISPARAMETERN

Die durch den Nutzer einstellbaren Parameter für die Ereigniserkennung sind die Anzahl an *Partnerclustern* und die Anteile an gemeinsamen Partikeln dieser Cluster. *Birth* und *Death* werden mit einer festen Anzahl von null Partnern erkannt, besitzen aber einen einstellbaren



a) *Birth* (blau) und *Death* (rot).



b) *Merge* (grün) und *Split* (orange) bei einer Einstellung von zwei *Partnerclustern* mit jeweils 40% gemeinsamen Partikeln.

Abbildung 7.4. Anzahl der Ereignisse über den Zeitraum der Simulation in Abhängigkeit des Radius' der Nachbarschaftssuche.

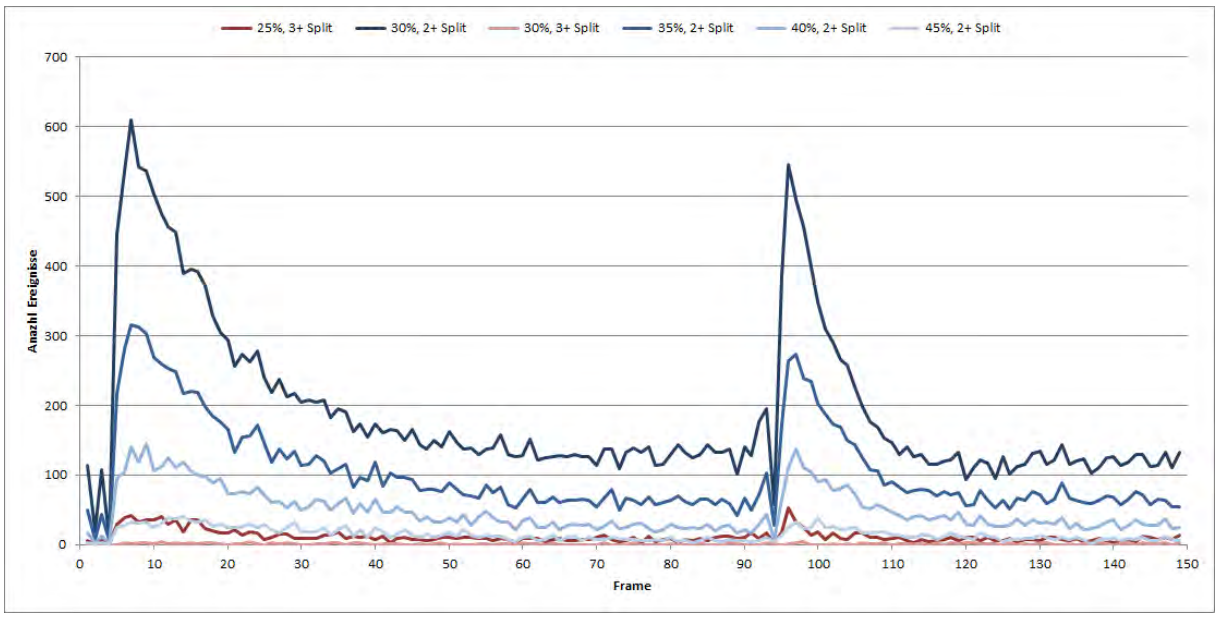


Abbildung 7.5. Anzahl der Ereignisse in Abhängigkeit der Parameter. Rote Verläufe sind Messungen für drei *Partnercluster*, blaue Verläufe für zwei. Für *Merge* sind Verlauf und Verhältnisse analog. Die Prozentangaben stehen für die Anteile gemeinsamer Partikel pro Partner.

Anteil von gemeinsamen Partikeln. Wie am Anfang dieses Kapitels beschrieben, hat dieser keinen Einfluss auf die Ereignisanzahl bei relevanten, geringen Werten. Somit werden nur die *Merge*- und *Splitereignisse* betrachtet.

Die in Abbildung 7.5 abgebildeten Verläufe zeigen mit strikteren Parametereinstellung einen erwarteten Abfall der Ereignisanzahl. Insbesondere die Erhöhung der Mindestanzahl an benötigten *Partnerclustern* schlägt sich enorm nieder. Drei *Partnercluster* mit jeweils 30% gemeinsamen Partikeln treten fast nie auf, eine Senkung auf 25% bewirkt eine im Vergleich deutliche Erhöhung, liegt jedoch gerade einmal auf ähnlichem Niveau wie zwei Partner mit 45% gemeinsamen Partikel. Eine mögliche Einstellung mit mindestens vier benötigten Partnern wird daher einen wesentlich geringeren Anteil erfordern, um mit der Heuristik *Merge*- und *Splitereignisse* erkennen zu lassen.

Trotz dieser Unterschiede ist die Verteilung der Ereignisanzahl über den Simulationsverlauf hinweg für alle Parametereinstellungen ähnlich. Insbesondere zeigt auch die Einstellung mit drei Mindestpartnern und 25% Anteil ähnliche Ausschläge wie die Verläufe mit deren zwei, so zum Beispiel bei den Zeitschritten 95 und 97.

7.3. QUALITATIVE ANALYSE DER EREIGNISERMITTLUNG

Aufgrund des Mangels eines optimalen Systems kann keine Bestimmung der Güte der bei der Ereigniserkennung zum Einsatz kommenden Heuristik getroffen werden. Allerdings ermöglicht die überlagerte Visualisierung des Partikel- und des *Ereignisobjektes* (siehe Abschnitt 6.4) eine Bewertung der Ereignisermittlung durch einen Vergleich mit den Positionen von Clustern. Durch eine ähnliche Bewegungsrichtung und Geschwindigkeit der Partikel sowie durch ihre örtliche Nähe oder Ferne lassen sich Agglomerationen von Partikeln während der Animation erkennen. Unterstützt wird dies durch die Einfärbung der Partikel nach ihrer Clusterzugehörigkeit. Dies hilft auch, ohne Zuhilfenahme von *Ereignisglyphen* eine Verschmelzung oder Trennung über mehrere Zeitschritte hinweg zu erkennen, so dass in einigen Fällen eine Beurteilung über die korrekte Ermittlung der Ereignisart getroffen werden kann. Zu diesen Fällen zählt das Trennen

Abbildung 7.6. Alle Ereignisse visualisiert im Simulationsraum. Die Zeit ist über den Farbwert der Glyphen visualisiert. Die Glyphgröße beträgt 0,012. Der *Partikelradiusmultiplikator* beträgt fünf, die *Ereignisheuristik* arbeitet mit zwei *Partnerclustern* mit jeweils 40% Anteil an gemeinsamen Partikeln.

oder Verschmelzen von Filamenten oder von Tröpfchen.

EREIGNISSE IM SIMULATIONSVERLAUF

In Abbildung 7.6 sind alle Ereignisse bei einem *Partikelradiusmultiplikator* von fünf visualisiert. Die Zeit der Ereignisses ist mit der Farbe kodiert. Vor allem im zweiten und vierten Fünftel sind deutlich horizontal langgestreckte, perlenkettenartige Verläufe der *Ereignisglyphen* zu sehen. Ihre Anordnung zeigt, dass die durch die Bewegung der Flüssigkeit nach außen entstehenden Filamentstrukturen, die meisten der Strukturereignisse verursachen. Auch werden Ereignisse durch die Tröpfchen oder die Flüssigkeitsfronten erzeugt, jedoch ist die Zahl der durch die Tröpfchen verursachten Ereignisse geringer. Dagegen sind die innerhalb der Fronten entstandenen Ereignisse verteilter angeordnet über die kurzen Achsen des Simulationsraumes. Dies kann nur während des Auftauchens der Glyphen während der Animation beurteilt werden, was ein Video der Animation mit zufälliger Clustereinfärbung[2] zeigt.

Diese Eigenschaft der Berechnung wurde ebenfalls in der Umfrage behandelt, um zu überprüfen, ob die Visualisierung korrekte Rückschlüsse ziehen lässt. Dabei ist vor allem die Expertenbeurteilung von Belang. Aufgrund der kleinen Anzahl werden in Abbildung 7.7 auch die Antworten der Laien aufgeführt.

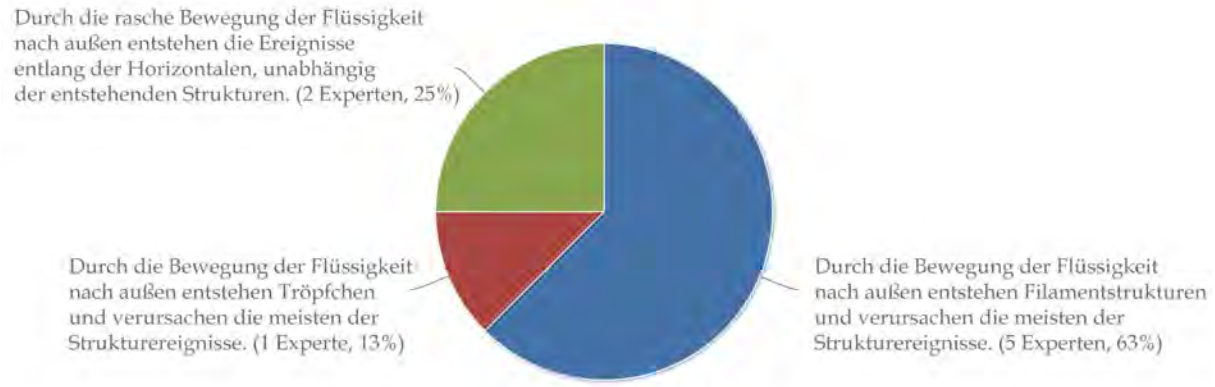
Vier Antwortmöglichkeiten wurden gegeben. Durch die Bewegung der Flüssigkeit nach außen entstehen (a) Filamentstrukturen und verursachen die meisten der Strukturereignisse oder (b) Tröpfchen und verursachen die meisten der Strukturereignisse oder (c) die Ereignisse entlang der Horizontalen, unabhängig der entstehenden Strukturen erzeugen die meisten Strukturereignisse oder (d) die während der Simulation verbleibenden Flüssigkeitsblöcke verursachen die meisten Ereignisse und zufälligerweise sind sie perlenartig angeordnet. Insgesamt 46% der Teilnehmer beantworteten die Frage korrekt mit (a), von den Experten waren es mit 63% bedeutend mehr.

Überraschend war, dass selbst zwei der Experten die Ereignisse unabhängig der Strukturen einordneten (Antwort c), was im Sinne der Ereignisvisualisierung falsch ist. Dies deutet die Schwierigkeit an, die Visualisierung ohne Erklärungen zu interpretieren. Das mit 34% weit mehr Laien diese Antwort gewählt haben zeigt, dass die Beurteilung ohne Hintergrundwissen schwerer fällt und eher geraten wurde. Auch bezog sich ein Großteil der Rückfragen seitens der Laien auf diese Fragestellung.

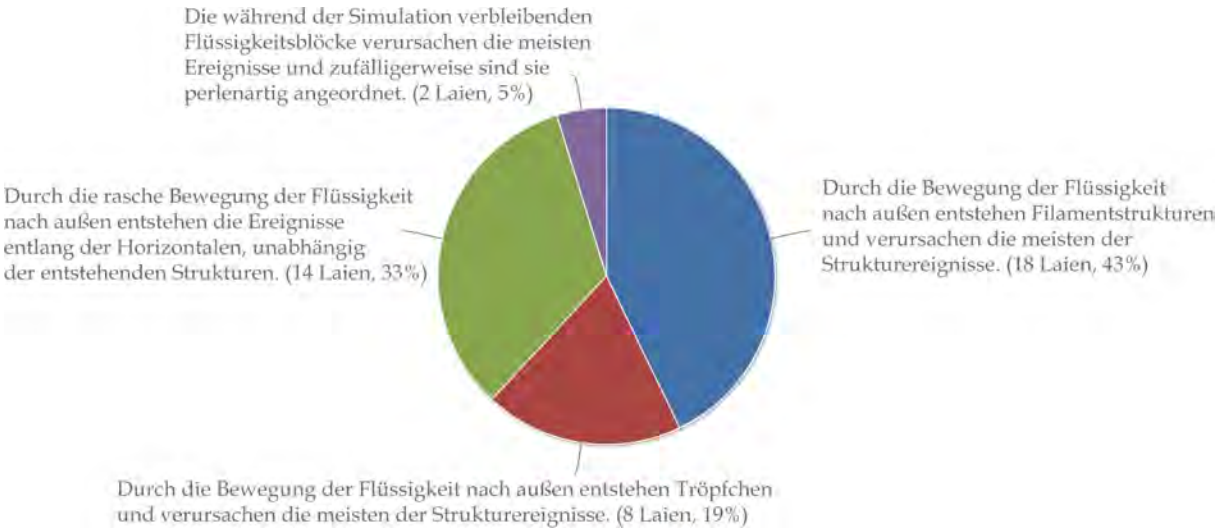
Die zahlreichen, innerhalb der beiden Flüssigkeitsfronten entstehenden Strukturereignisse lassen sich visuell nur schwer beurteilen. Ihre Entstehung kann jedoch mit der Abhängigkeit des CFD von lokalen Maxima (siehe Abschnitt 7.1) und der damit verbundenen Schwankung der Clusterzugehörigkeit von Partikeln begründet werden, wodurch die *Ereignisheuristik* viele Ereignisse erkennt. Durch einen Schnitt im Simulationsraum kann ... TODO SCREENSHOT

Diese Art von Ereignissen innerhalb von Strukturen ohne deren sichtbare Trennung oder Vereinigung werden als *Ereignisrauschen* bezeichnet. Es muss vom in Abschnitt 5.6 erwähnten Rauschen abgegrenzt werden, da es nicht die Bildung kleiner *Partnercluster* beschreibt.

[2] https://www.youtube.com/watch?v=0K7J3ISMrDM



a) Antwortverteilung von acht Experten.



b) Antwortverteilung von 42 Laien.

Abbildung 7.7. Inhaltliche Beurteilung der Ereignisvisualisierung nach Vergleich eines Videos der Simulation mit der statischen Visualisierung aller Ereignisse analog zu Abbildung 7.6.

Birth- und *Deathereignisse* treten sehr gehäuft an kleinen Clustern auf. Interessanterweise bilden sie bei der Visualisierung über den gesamten Zeitverlauf mehrere, wie in Abbildung 7.8 abgebildete Ketten.

Diesem Phänomen wird der Name „Nulltiefenwert *Einpartikelcluster*" gegeben, welches in Abschnitt 8.1 näher beschrieben wird.

ABHÄNGIGKEIT VON DEN EREIGNISPARAMETERN

Der Vergleich in Abbildung 7.9 zeigt beispielhaft die erwartete Abnahme der Erkennung der Ereignisse bei restriktiveren Einstellungen der Parameter. So wird die Trennung des Filaments bei einem Anteil gemeinsamer Partikel von 40% der beiden beteiligten Cluster als Ereignis erkannt, bei einem Anteil von 45% jedoch nicht mehr, so dass dieser Wert für dieses Filament zu hoch eingestellt ist. Andererseits ermöglichen restriktivere Werte die Verringerung des *Ereignisrauschens*, was an der Erkennung des *Mergeereignisses* innerhalb des Filaments deutlich wird.

Diese Art der Abhängigkeit der Ereigniserkennung von den gewählten Parametern kann über die gesamte Simulation beobachtet werden.

7.4. VISUALISIERUNG DER CLUSTER UND EREIGNISSE

DIESE SEKTION IST NOCH UNFERTIG!

FARBZUWEISUNG DER CLUSTER

Wenn die Cluster durch die Vererbung der Farben des vorhergehenden Zeitschrittes visualisiert werden, können benachbarte Cluster eine identische Farbe aufweisen, so dass die Clusterunterscheidung erschwert ist. Bei der zufälligen Zuweisung der Farbe geschieht dies nicht. In Abbildung 7.10a sind deutlich weniger Cluster unterscheidbar als in Abbildung 7.10b. Aus diesem Grund wurden für die qualitative Analyse nur Datensätze mit zufälligen Clustereinfärbung verwendet.

In der Visualisierung als Animation ist der Verlauf der Cluster bei der Farbvererbung jedoch besser zu verfolgen, da bei der zufälligen Zuweisung ein starkes Farbflackern auftritt.

ANORDNUNG

Kapitel 6 Die Komplexität der Ereignisvisualisierung (*Ereignisobjekt*) ist dabei geringer ist als die der Partikel (*Partikelobjekt*). Das hat drei Gründe. Erstens ist die Anzahl der Ereignisse um mehrere Größenordnungen kleiner und zweitens werden im Rahmen dieser Arbeit keine Verbindungen zwischen den Ereignissen analysiert und visualisiert, was die Differenz der Elementanzahl erhält. Schließlich ist drittens, was als Hauptgrund betrachtet werden sollte, die Visualisierung der Ereignisse statisch. Sie behalten im Gegensatz zu den Partikeln dieselbe Position, was sie für den Betrachter während der Animation einfacher erfassbar macht. Andererseits steigern die in den Glyphen gespeicherten Informationen über den Zeitpunkt und die Ereignisart den Komplexitätsgrad des *Ereignisobjektes*, worauf in der Auswertung eingegangen wird.

In der Übersichtsansicht Komplexitätsgrad Glyphen doof, rangezoom nicht.x

SICHTEN

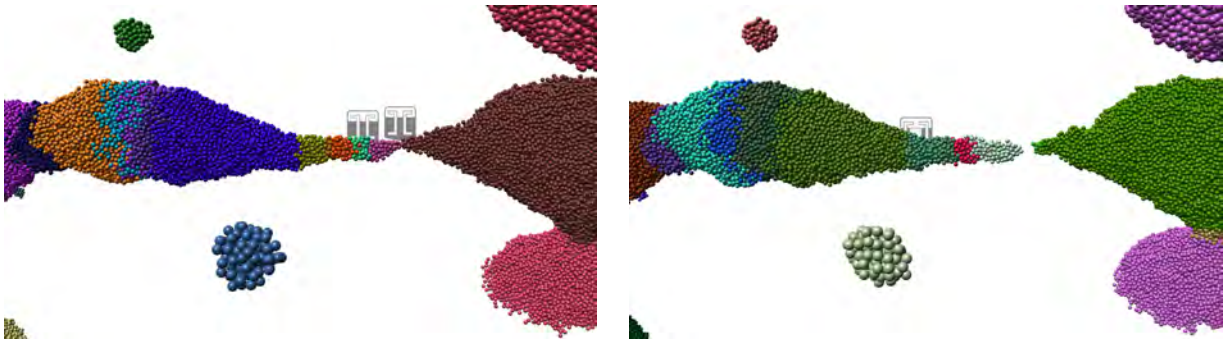
Die Makrosicht dient zur quantitativen Erfassung:

Die Mikrosicht zur qualitativen Erfassung
(siehe Abschnitt 3.3)

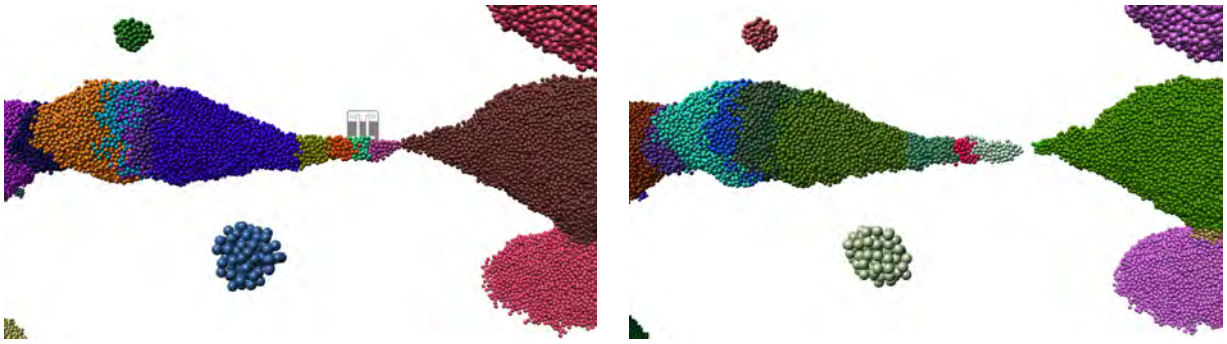
a) *Birth*ereignisse im gesamten Simulationsverlauf. Deutlich sind schnürenartige Verläufe zu erkennen. Die Zeit wird über den Farbwert wiedergegeben und die Glyphgröße beträgt 0,049. *Death* verhält sich analog.

b) *Birth* und *Death* und die zugehörigen *Einpartikelcluster* in der Nahaufnahme mit einer Glyphgröße von TODO.

Abbildung 7.8. Kettenverläufe von *Birth*- und *Death*ereignissen bei einem *Partikelradiusmultiplikator* von fünf.

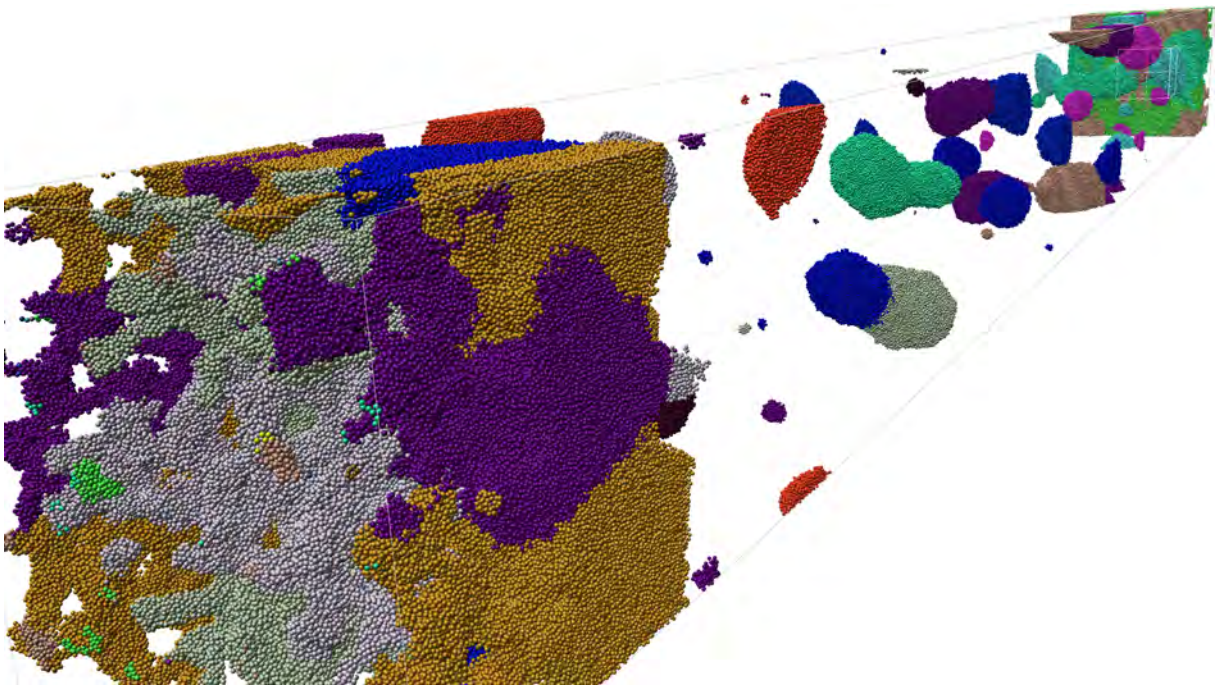


a) Das *Splitereignis* (das Rechte der beiden *Split*ereignisse) wird erkannt bei einem Anteil von 40% gemeinsamer Partikel. Das *Merge*ereignis (rechts) kann als *Ereignisrauschen* betrachtet werden.

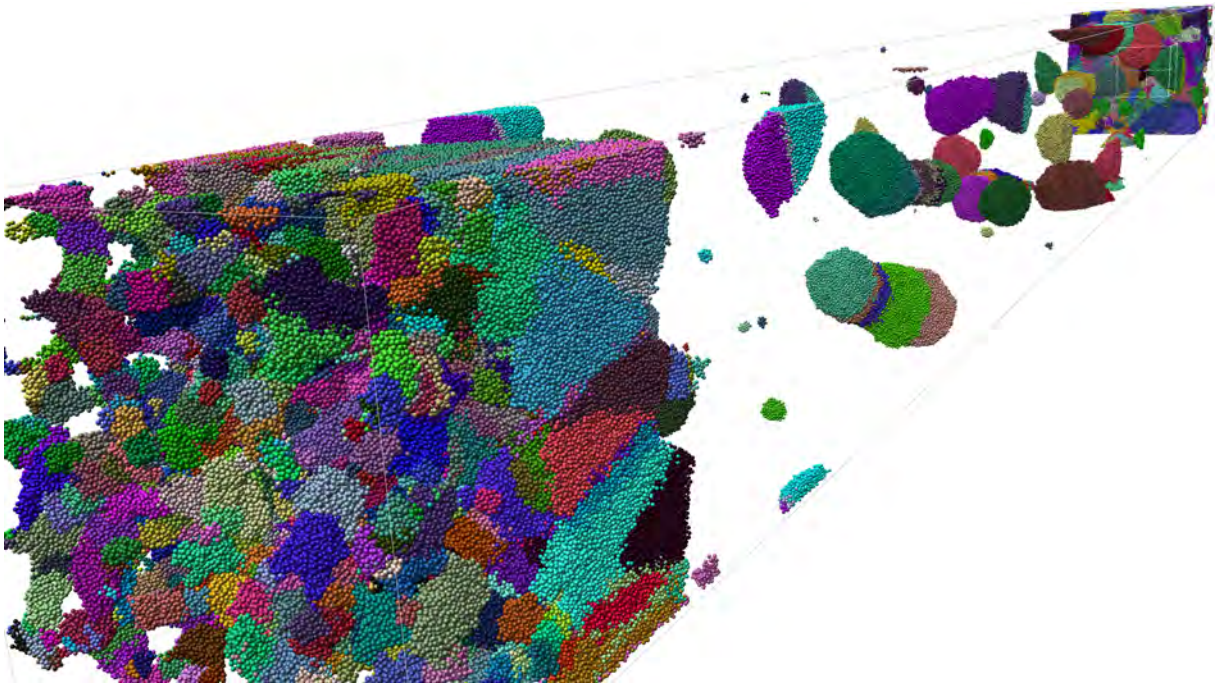


b) Das *Splitereignis* wird bei einem Anteil von 45% gemeinsamer Partikel nicht erkannt. Gleichzeitig ist jedoch das *Ereignisrauschen* geringer, da kein *Merge*ereignis innerhalb des Filaments auftaucht (rechts).

Abbildung 7.9. Abhängigkeit der Erkennung eines *Split*ereignisses an einem Filament vom Anteil der gemeinsamen Partikel. Der braune Cluster hat sich während des Zeitschrittes 101 (links) bereits vom lilafarbenen Cluster gelöst, was im Zeitschritt 102 (rechts) besser zu sehen ist. Durch die zufällige Clusterfärbung verändern sich die Farben zwischen den Zeitschritten. Partikel der Gasphase sind ausgeblendet.

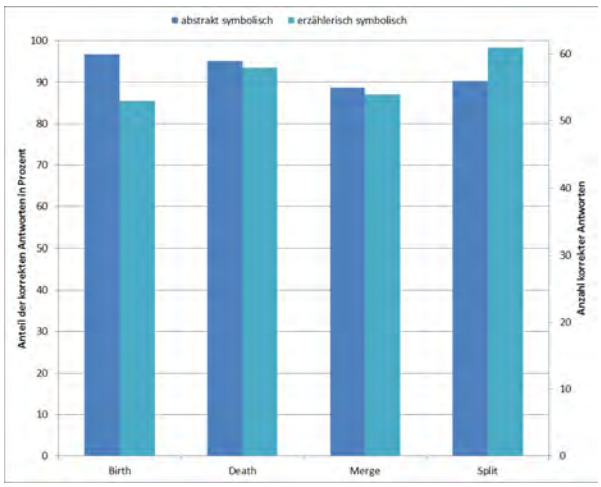


a) Clusterfärbung mit Vererbung.

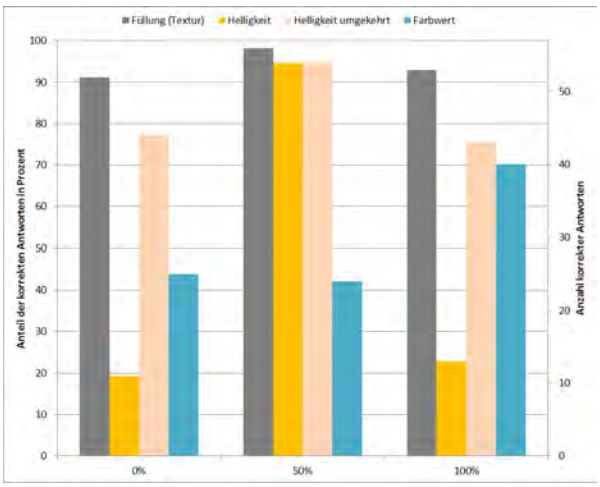


b) Zufällig Clusterfärbung ohne Vererbung.

Abbildung 7.10. Clustereinfärbung mit und ohne Vererbung bei Zeitschritt 101. Der *Partikelradiusmultiplikator* beträgt fünf. Partikel der Gasphase sind ausgeblendet.



a) Zuordnung der mit der Textur als abstrakt und narrativ symbolisch kodierten Ereignisart.



b) Zuordnung der mit der Textur (grau), Helligkeit (gelb, rosa) und Farbwert (blau) kodierten Ereigniszeit.

Abbildung 7.11. Umfrageergebnisse zur korrekten Zuordnung der Visualisierung der Ereignisglyphen.

GLYPHGESTALTUNG

Die überwiegende Mehrheit hat die korrekte Semantik

Damit wurden die Annahmen bei der Glyphgestaltung bestätigt und sie kann als erfolgreich betrachtet werden.

Ein Vorschlag eines Teilnehmers lautete, die ikonische Darstellung des *Birth*ereignisses mit den symbolischen Darstellungen der drei anderen Ereignisse zu kombinieren, da der Teilnehmer „den Kreis nur nach Ausschlussprinzip“ zuordnen konnte. Hingegen sagen die Werte, dass der Kreis BLABLA

Ein Ziel war es, den Laien möglichst unvoreingenommen urteilen zu lassen, um nicht von der Wirkung der *visuellen Variablen*. Aufgrund von Rückmeldungen der Form „[erbitte] kurze Erklärung von Filament, Strukturereignis” und „eventuell hätte ich mehr verstanden, wenn ich wüßte was Glyphen sind und wozu das ganze wirklich dienen soll” ist dieses Ziel teilweise erreicht worden. Andererseits kann der bewusste Informationsmangel auch von der Konzentration auf die Gestaltung ablenken „Bitte schreib manchmal dazu, wieso du die Fragen stellst; ich musste erstmal nachdenken, warum zur Hölle ich jetzt Symbole den Bedeutungen zuordnen soll”.

GLYPHDARSTELLUNG IM PARTIKELRAUM

Die in Abschnitt 6.4 getroffene Annahme, dass die Positionierung des Glyphen oberhalb des Ereignisses intuitiv sei, wurde ebenfalls mit der Umfrage überprüft. Dazu wurde jeweils ein Partikel unter die Spitze, knapp über die Spitze und in die Mitte des Glyphen platziert. Überraschenderweise wurde der Annahme durch eine große Mehrheit widersprochen, denn für die implementierte Variante entschieden sich mit zehn Teilnehmern nur 16% der Befragten. Eine Platzierung des Partikels knapp oberhalb der Spitze wurde zwar lediglich von zwei Personen favorisiert, dafür entschieden sich 46 der Teilnehmer (75%) für die Positionierung in der Mitte des Glyphen.

Ein Kommentar bewertet den hohen Komplexitätsgrad des *Ereignisobjektes* in der Makrosicht als negativ und spricht von einer „überladenen Glyphdarstellung, die unter Verdeckungen leidet“. Dies ist der Darstellung in der Umfrage geschuldet, da dort die Makrosicht gewählt wurde und aufgrund der statischen Darstellung als Bild mit begrenzter Zoommöglichkeit eine

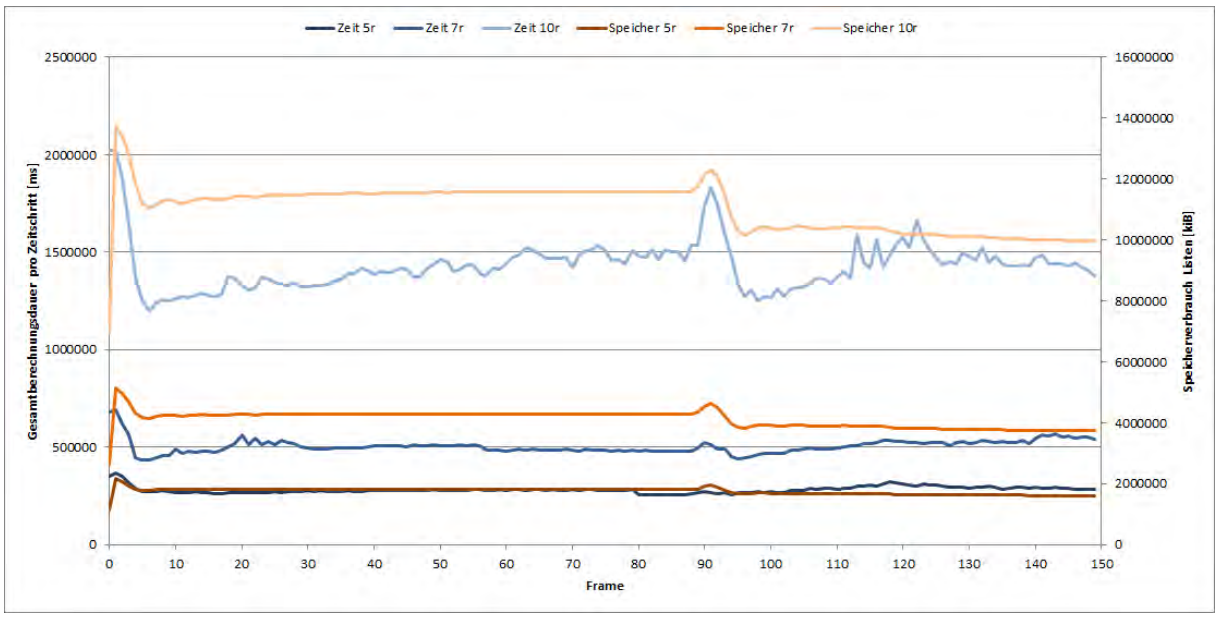


Abbildung 7.12. Umfrageergebnis über die favorisierte, *visuelle Variable* zur Zeitdarstellung in der Makrosicht.

große Glyphgröße gewählt wurde. In der Anwendung können die Glyphen durch den Nutzer skaliert werden, da das *StaticRenderer Modul* einen entsprechenden Parameter besitzt (siehe Abschnitt 6.4).

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

7.5. RESSOURCENVERBRAUCH

In Abbildung 7.13 ist wiederholt eine starke Abhängigkeit vom *Partikelradiusmultiplikator* bei der Nachbarschaftssuche festzustellen. Mit größerem Suchvolumen nehmen Zeit- und Speicherverbrauch stark zu. Der aufgezeigte Speicherverbrauch spiegelt lediglich die Größe der die Datenstrukturen enthaltenen Listen wider. Bei der Berechnung hat der Prozess mit dem gegebenen Datensatz und ohne Visualisierung bis zu 13 Gibibyte (GiB) bei einem Multiplikator von zehn benötigt.

Der Ressourcenverbrauch ist über den Zeitraum annähernd gleichbleibend, bis auf zwei Ausnahmen. Bei der Trennnung und beim Zusammenstoß der Flüssigkeitsfronten ist er stark erhöht, was auf die erhöhte Anzahl an Nachbarschaftsbeziehungen durch die dichte Packung der Partikel in den Fronten sowie die erhöhte Anzahl an Clustern zu diesem Zeitpunkt (siehe Abschnitt 7.1) zurückzuführen ist. Dies lässt sich gut in Abbildung 7.14 erkennen. Vor allem die Nachbarschaftssuche benötigt deutlich länger und der Speicherplatz für die Partikelliste ist größer, da die durchschnittliche Anzahl von Nachbarn pro Partikel aufgrund der dichteren Nähe vieler Partikel zueinander während dieser Zeiten höher ist.

Nach dem Zusammenstoß beider Fronten kann eine leichte Abnahme der durchschnittlichen Nachbarn pro Partikel im Vergleich zur Zeit vor dem Zusammenstoß beobachtet werden. Das ist ein Indiz auf mehr Randpartikel mit weniger Nachbarn und damit ein erhöhtes Tröpfchenaufkommen. Diese Vermutung wird durch ein stark erhöhtes Aufkommen von *Mindestgrößencluster* nach dem Zusammenstoß in Abbildung 8.1 gestützt. Weiterhin könnten auch mehr Partikel in die Gasphase übergegangen sein als zuvor, was durch die Messungen in Abbildung 7.15 bestätigt wird.

Der nahezu halbierte Speicherverbrauch der Partikelliste des aktuellen Zeitschrittes und dem

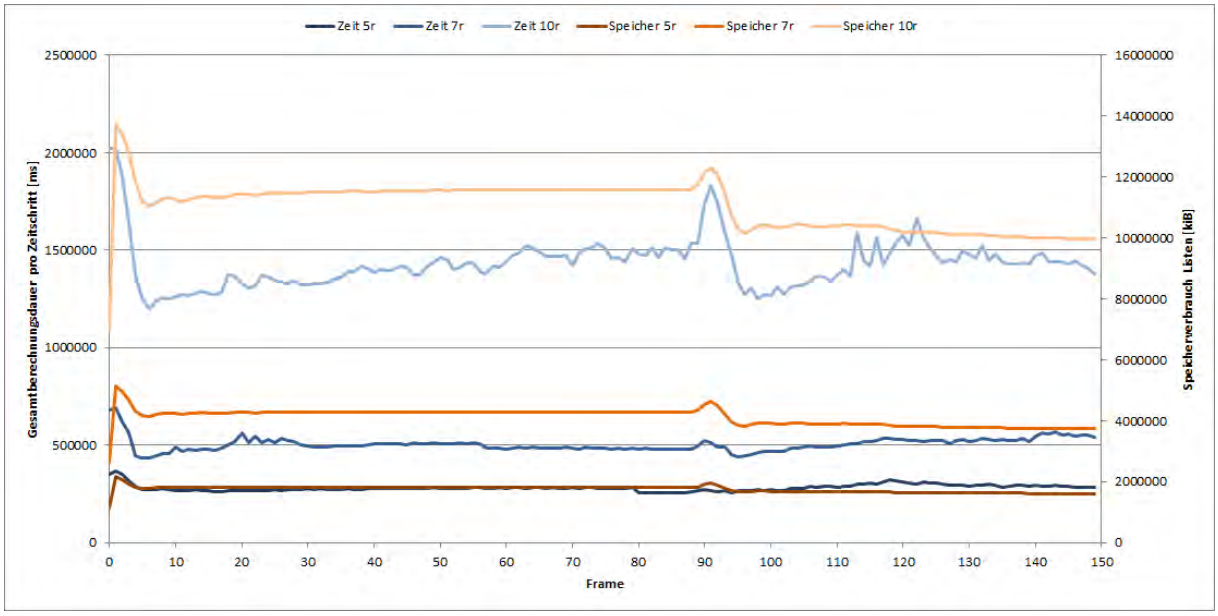


Abbildung 7.13. Ressourcenverbrauch über alle Zeitschritte in Abhängigkeit des *Partikelradiusmultiplikators*. Der Speicherverbrauch spiegelt nur die Listengröße und nicht den durch die Berechnungen benötigten Speicher wider.

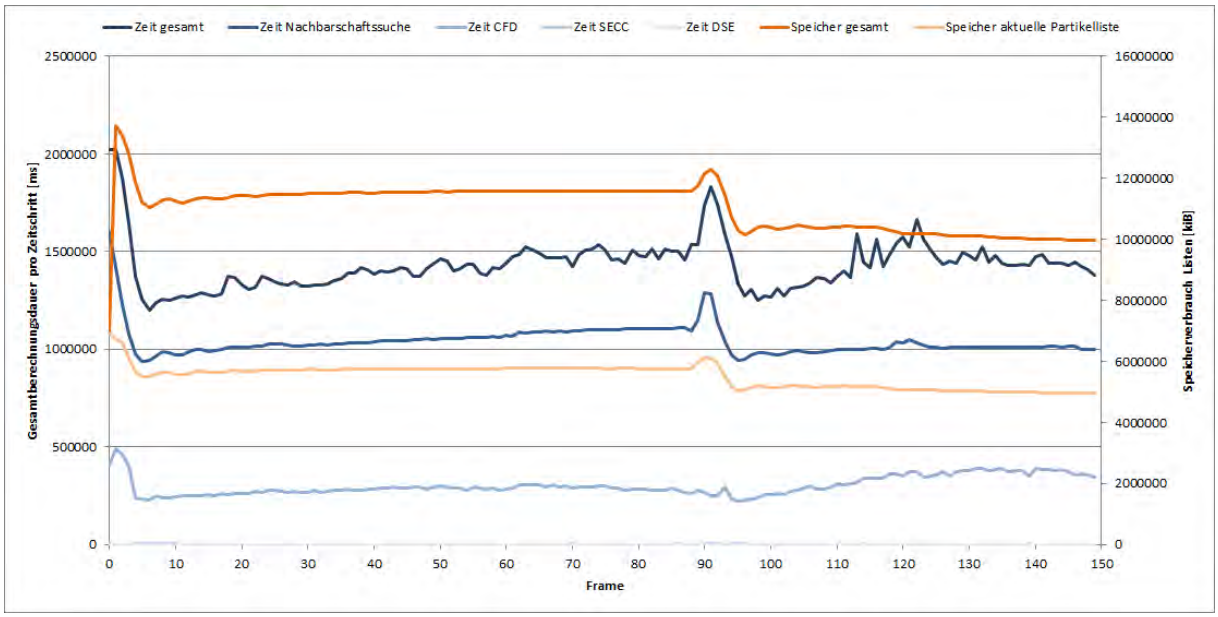


Abbildung 7.14. Ressourcenverbrauchsverteilung der Berechungsschritte über den gesamten Zeitraum bei einem *Partikelradiusmultiplikator* von zehn. Die Speicherdaten für SECC sowie für DSE wurden ausgeblendet, da ihre geringen Werte wie auch ihre Zeitdaten unerheblich sind.

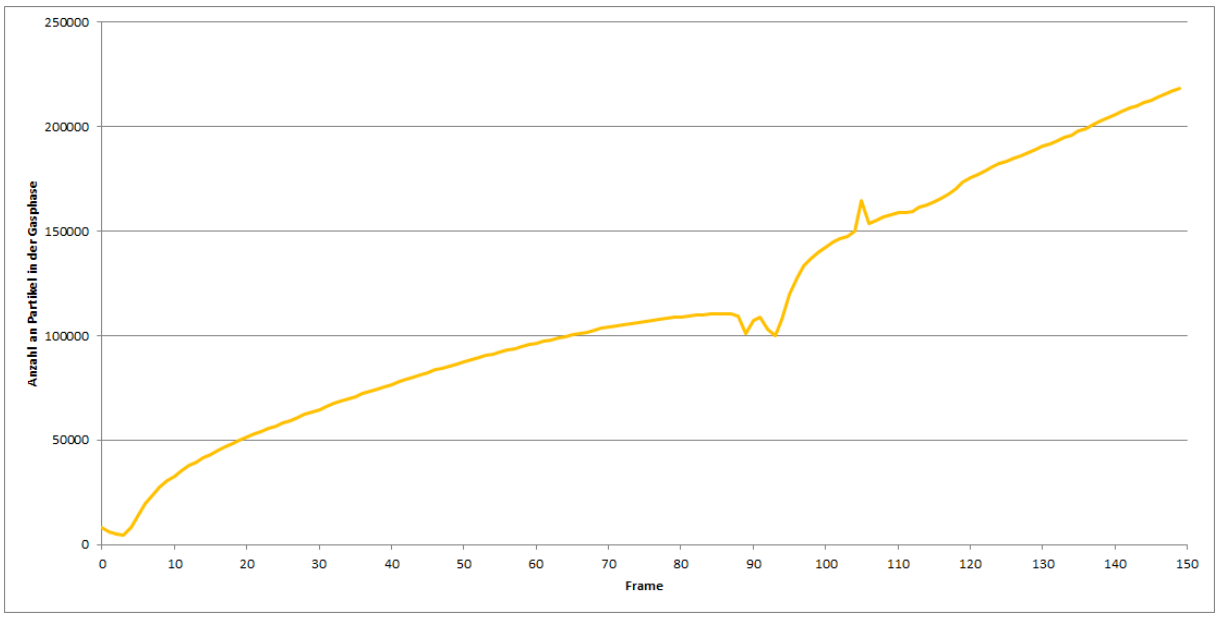


Abbildung 7.15. Anzahl der Partikel in der Gasphase über den Zeitraum der Simulation verteilt.

Gesamtspeicherverbrauch liegt daran, dass die Partikelliste des vorhergehenden Zeitschrittes in den Gesamtverbrauch einfließt. Hingegen sind die anderen Listen für die Cluster, *Partnercluster* sowie Ereignisse mit einer Größe von wenigen Kibibyte (kiB) vernachlässigbar klein und wurden daher ausgeblendet.

Diese Daten liefern eine wichtige Erkenntnis. Die Nachbarschaftssuche benötigt mit großem Abstand die meiste Zeit, danach folgt der CFD mit etwa ein Viertel des Zeitverbrauchs. Die anderen Schritte liegen mit einer Dauer von maximale zwei Sekunden für den SECC und wenigen Millisekunden für die Ereignisheuristik um zwei und mehr Größenordnungen darunter.

Der Speicherverbrauch der Listen ist mit maximal 1,4 GiB bei einem *Partikelradiusmultiplikator* von zehn überschaubar. Der in Stichproben registrierte wesentlich höhere Gesamtspeicherverbrauch des Berechnungsprozesses zwischen elf und 13 GiB ist vor allem auf das Vorhalten des gesamten Datensatzes der Simulation in Form des MMPLD im Speicher zurückzuführen (über fünf GiB) sowie auf den hohen Verbrauch bei der Durchsuchung des kD-Baums.

8. DISKUSSION DER ERGEBNISSE

8.1. EINFLUSS DER NACHBARSCHAFTSSUCHE

Wie die Evaluation zeigt, hat die bei der Nachbarschaftssuche gewählte, maximale Reichweite einen maßgeblichen Einfluss auf alle Berechnungsschritte. Deswegen ist die Diskussion des *Partikelradiusmultiplikators* und die damit verbundenen Phänomene bedeutend für diese Arbeit.

FLÜSSIGKEITSPARTIKEL OHNE CLUSTER UND ERHÖHTES EREIGNISRAUSCHEN BEI ZU KLEINER NACHBARSCHAFTSSUCHE

In Abschnitt 7.1 ist zu sehen, dass bei kleinem *Partikelradiusmultiplikator* nur sehr kleine Cluster erkannt werden, denn das Volumen für die Nachbarschaftssuche ist so klein, dass viele Partikel der Flüssigkeitsphase keine Nachbarn besitzen. Weiterhin ist die Berechnungsgeschwindigkeit bei einem vierfachen Modifikator 20% geringer als bei einem dreifachen, so dass der in der Benutzeroberfläche zugelassene Minimalwert für den Radiusmultiplikator vier beträgt. 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 CFD Abschnitt 5.3 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 (siehe Abbildung 7.1). 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 *Ereignisrauschen* statt als gewinnbringende Information betrachtet, so ist ein kleines Suchvolumen bei der Nachbarschaftssuche von Nachteil.

GASPARTIKELCLUSTER BEI DER CLUSTERREDUKTION

Wenn andererseits der *Partikelradiusmultiplikator* zu groß gewählt wird (fünffacher Radius oder höher), können auch Partikel der Gasphase Nachbarn erhalten und werden im CFD Clustern zugeordnet. Dies wiederum kann dazu führen, dass beim Reduzieren der kleinen Cluster (siehe Abschnitt 5.4) die Partikel in Clustern mit nur diesem einen Partikel zurückbleiben. Das liegt darin begründet, dass die Clusterreduktion begrenzte Iterationsschritte aufweist. Falls der von den umliegenden Clustern 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

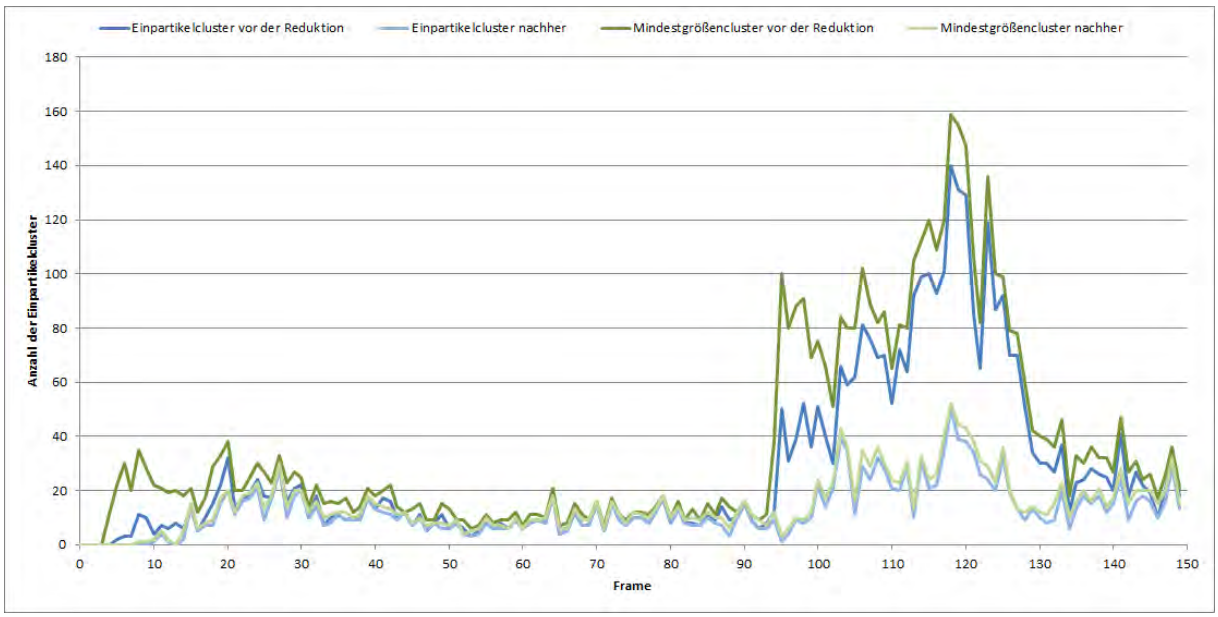


Abbildung 8.1. Anzahl der Mindestgrößencluster (grün) sowie der *Einpartikelcluster* (blau) vor und nach der Clusterreduktion bei einem *Partikelradiusmultiplikator* von fünf.

der zuerst ausgewählte Partikel als einziger im Cluster. Das führt in der Ereigniserkennung (siehe Abschnitt 5.6) zu falschen Erkennungen von *Birth* und *Death*. Allerdings ist dieser Fehler nicht aufgetreten, es werden keine einzelnen Partikel oder kleine Partikelgruppen durch die Clusterreduktion erzeugt. Das kann durch den Vergleich der Einpartikel- sowie der minimalen Cluster direkt nach der Clusterbildung mit deren Anzahl nach der Clusterreduktion überprüft werden. In Abbildung 8.1 bleibt in allen Zeitschritten die Anzahl nach der Clusterreduktion gleich oder ist geringer als vorher. Damit ist gezeigt, dass durch die Reduktion keine Gaspartikelcluster erzeugt werden.

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, etwa durch die Umstellung der Clusterreduktion in einen iterativen oder rekursiven Prozess, dessen Durchgänge abhängig der gewählten Mindestclustergröße sind.

GASPARTIKEL IN CLUSTERN BEI GROSSEN RADIEN

Zur Beschleunigung des CFD werden alle Partikel, die beim Suchen entlang des Pfades getroffen wurden, ebenfalls zum Cluster hinzugefügt. Diese Beschleunigung kann Fehler verursachen, wenn der *Partikelradiusmultiplikator* bei der Nachbarschaftssuche (siehe Abschnitt 5.2) groß gewählt wurde, da so auch Gaspartikel am Rand von Clustern mit hinzugefügt werden. Diese Fehler sind jedoch vernachlässigbar, da die Heuristik der Ereigniserkennung Mengen betrachtet und somit einzelne Gaspartikel in der Clustergruppe einen geringen Einfluss auf das Ergebnis haben.

NULLTIEFENWERT 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 CFD 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. Da diese Daten sehr umfangreich sind und sich schwer statisch visualisieren lassen, wird hierbei auf die digitalen Protokolle des CFD in den einzelnen Messungen verwiesen. Dieses Phänomen tritt ab dreifachem Partikelradius 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 CFD ignoriert und nicht in Cluster eingeteilt.

Diese Phänomen bewirkt die Bildung zahlreicher *Birth*- und *Deathereignisse* (siehe Abschnitt 7.3). Da diese Cluster jedoch korrekt erkannt werden, wird dies nicht als Fehler betrachtet. Eine Einordnung als *Ereignisrauschen* kann hingegen vorgenommen werden.

8.2. FARBGEBUNG DER GLYPHEN

DIESE SEKTION IST NOCH UNFERTIG!

Die Umfrage über die Positionierung des Glyphen relativ zum Ereignis widerspricht der Implementation und scheint damit dem in Abschnitt 6.1 formulierten Ziel zur Förderung der präattentiven Wahrnehmung entgegen zu wirken. Dem kann begegnet werden, indem entweder die Glyphdarstellung durch einen deutlich spitzer zulaufender Pfeil am unteren Rand eindeutiger gestaltet wird oder die Positionierung des Glyphs mittig über dem Ereignis erfolgt.

Farbverläufe: Feedback Umfrage Erfahrungswerte mit Farbe gekoppelt, so dass eine Farbgebung nach bekanntem Muster (z.B. Verkehrsampel) sinnvoller ist.

8.3. VERDECKUNG DER GLYPHEN

Wie in der Evaluation in Abschnitt 7.4 gezeigt, wurde in der Umfrage die Verdeckung der Glyphen untereinander bei starken Zoomfaktoren bemängelt. Dies ist der Darstellung in der Umfrage geschuldet, da in der Anwendung die Glyphen skaliert werden können. Falls auch das nicht ausreicht, kann die Einführung einer Beziehungssicht nach Lima helfen (siehe Abschnitt 3.3). Um diese Sicht herstellen zu können, kann nach Gleicher et al (siehe Abschnitt 3.3) eine Kombination von Gegenüberstellung und expliziter Verschlüsselung genutzt werden und so durch die Trennung des Ereignisraums vom Raum der Partikel die Bindung der *Ereignisglyphen* an die Position aufgehoben werden, wie es die in dieser Arbeit vorgeschlagene *Strukturereignistaxonomie* ermöglicht (siehe Abschnitt 6.2) Weiterhin zeigte sich bei der Nutzung des Programms das Problem, dass die Glyphen durch die Partikel verdeckt werden, was dadurch ebenfalls beseitigt werden kann. Aufgrund der Komplexität einer solchen Implementation, kann sie, wie in Kapitel 10 dargelegt, in einer separaten Arbeit untersucht werden.

Eine optionale, automatisierte Abhängigkeit der Größe der Ereignisglyphen von der Entfernung zur Kamera kann dem Nutzer den Wechsel zwischen Makro- und Mikrosicht erleichtern (siehe Abschnitt 3.3) und Verdeckungsprobleme verringern.

Eine Senkung der Partikelopazität wäre ebenfalls eine Option, um Glyphen sichbar zu machen, die sich hinter Partikeln befinden. Allerdings wird eine Einstellung der Opazität von Partikeln durch *MegaMol* derzeit nicht unterstützt.

8.4. OPTIMIERUNG DES RESSOURCENVERBRAUCHS 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. Dies tritt bei der Beachtung

der periodischen Randbedingungen und dem damit verbundenen, achtfachen Aufruf des Suchalgorithmus des kD-Baums umso stärker hervor. Allerdings haben Versuche mit *OpenMP* als auch mit der Parallel Patterns Library (PPL)-Bibliothek gezeigt, dass die ANN-Bibliothek 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, die unabhängigen Suchen würden mit steigender Partikel- und Threadanzahl jedoch sehr viel Speicher verbrauchen (siehe Abschnitt 7.5). Der Grund für die fehlende Parallelisierung der Schleifen, welche die ANN-Suchmethoden enthalten, ist allerdings, dass die ANN-Bibliothek 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-Bibliothek für eine parallele Nutzung nicht geeignet und ein Wechsel zu einer anderen Bibliothek, wie beispielsweise zu Fast Library for Approximate Nearest Neighbors (FLANN)-Bibliothek [O+13] [Wij14], ist empfehlenswert.

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. Zum Ersten werden dort lediglich Vergleiche und Inkrementationen durchgeführt. Zum Zweiten ist das Hinzufügen des ermittelten Wertes zum der die benachbarten Cluster enthaltenen Container nicht Threadsicher und muss durch einen Mutex geschützt sein. 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-Verfahrens der Listen sowie der sequentiell zu erfolgenden Ausgabe eine Parallelisierung in der Geschwindigkeit kaum spürbar bzw. sogar nachteilig.

8.5. OPTIMIERUNG DER ALGORITHMEN

Der CFD kann optimiert werden, indem beim Treffen auf den ersten Partikel, der einem Cluster angehört, die Suche beendet ist und die Clusterzuweisung erfolgt. Dies kann enorme Geschwindigkeitszuwächse des Algorithmus bewirken, ohne dass dessen Qualität beeinflusst wird.

Die SECC kann um einen Clusternachbarschaftsgraphen erweitert werden. Dieser speichert wie in Abschnitt 5.5 vorgeschlagen für jeden Zeitschritt die Nachbarschaften der Cluster. In Kombination mit dem Mengenvergleich können so mit einer messbaren Wahrscheinlichkeit die Cluster zugeordnet werden (siehe Abschnitt 6.5). Dies stellt eine weitere Option bereit, um die Struktureereignisse daraus abzuleiten.

8.6. NÜTZLICHKEIT VON VISUALISIERUNGEN

Die Nützlichkeit der Visualisierung von Molekulardynamiksimulationen schätzen Experten und vereinzelt auch Laien als sehr hoch ein. Letztere Gruppe verbindet mit der Visualisierung jedoch eher andere Big-Data Domänen wie „[die] Bereiche Pharmazie und Medizin" und „wo bewegliche komplexe Systeme auftauchen [wie] Straßenverkehr, Games etc." Die Ausnahme bildet ein Laie, der unter Verwendung der Ellipsoid-Splatting-Technik für Smoothed-particle hydrodynamics (SPH)-Simulationen in einem verwandten Gebiet arbeitet. Er ordnet die Wichtigkeit der Visualisierung für „die Forschung und die Entwicklung" sehr hoch ein mit der Begründung, „da man irgendeine Visualisierung benötigt, um die Daten überhaupt irgendwie sinnvoll auswerten zu können."

Ein Experte begründete die große Relevanz von Visualisierungen in Molekulardynamiksimulationen mit drei Argumenten. So dienen seiner Ansicht nach die Visualisierungen dazu, um „Simulationsergebnisse [zu] validieren (erscheint das Verhalten des Systems plausibel?)", um „Effekte sichtbar [zu] machen, die bei naiver Auswertung oder Standardauswertung verborgen bleiben" und um die „Simulationsergebnisse einer Zuhörerschaft verständlich [zu] präsentieren".

9. ZUSAMMENFASSUNG

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 Struktur*ereignisse* abzuleiten.

Die Berechnungen lassen mit Hilfe ihrer sowohl quantitativen als auch qualitativen Ausgabe Erkenntnisse über die Veränderung der Strukturen während der Simulation zu, da die Anzahl der Cluster und die Verteilung der Partikel stark auf Topologieänderungen reagiert. Der Nutzer kann Parameter ändern, um etwa *Ereignisrauschen* zu verringern und durch die Möglichkeit, die Berechnungen permanent zu speichern, auch schnell zwischen den Parametern für die einzelnen Berechnungsschritte wechseln und so die Nachteile der individuellen Einstellungen auszugleichen; beispielsweise die Erfassung aller wesentlichen Ereignisse durch einen kleinen *Partikelradiusmultiplikator* und einen niedrigen Schwellwert für die Anteile gemeinsamer Partikel mit dem Nachteil von starkem *Ereignisrauschen*. Dagegen können restriktivere Einstellungen mit höheren Schwellwerten geringes *Ereignisrauschen* verursachen mit dem Nachteil, einige wesentliche Ereignisse nicht zu erfassen (siehe Abschnitt 7.2, Abschnitt 7.3).

Trotz dessen, dass die erhoffte Umsetzung der Berechnungen mit *Konturbäumen* und ihren bereitgestellten Schwellwerten nicht erfolgen konnte, wurde die Zielstellung, Strukturveränderung mittels *Ereignisglyphen* abzubilden, durch die Entwicklung eigener Algorithmen in Form des CFD und der SECC sowie der darauf aufbauenden *Ereignisheuristik* erreicht. Diese geben dem Nutzer Mittel in die Hand, Schwellwerte zu setzen, die die Qualität und Anzahl der erkannten Ereignisse stark beeinflussen. Weiterhin ergab eine Umfrage unter Experten und Laien, dass die Bedeutung der Glyphen intuitiv eingeschätzt werden kann. Lediglich die Zuordnung der Helligkeit für die Darstellung der Ereigniszeit sollte vertauscht werden. Auch kann unter Berufung auf die Umfrageergebnisse die Empfehlung erfolgen, die Farbdarstellung für die Ereigniszeit nur in der Makrosicht zu nutzen und für die Mikrosicht stattdessen die Helligkeit oder den Füllstand zu verwenden.

Seitens der Experten wurde der Prozess, der zu den visualisierten Ereignissen führte, in der Mehrheit korrekt eingeschätzt. Hingegen fällt es schwer, ohne Hintergrundwissen allein durch die Visualisierung Prozesse richtig zu beurteilen (siehe Abschnitt 7.3).

Durch die Clustervisualisierung, die umfangreiche Bereitstellung quantitativer Ergebnisse, die vorgeschlagene *Struktur*ereignistaxonomie, mehrere Möglichkeiten zur Glyphdarstellung sowie durch die Auswertung unter Einbeziehung von Laien wurden die geplanten Ziele der Arbeit übertroffen. Die entstandene Software in Form des *MegaMol*-Plugins (siehe Abschnitt A.1) als auch des *Glyphprototypen* (siehe Abschnitt A.3) kann für Anwendungsfälle über diese Arbeit hinaus verwendet werden. So kann mit dem Plugin eine Strukturanalyse anderer Partikeldatensätze mit einem Skalarfeld für den *Tiefenwert* erfolgen. Bei einer Lockerung der *Struktur*ereignistaxonomie durch die Entfernung der Bindung der Position an den Ereignisort, kann der *Glyphprototyp* zur Überprüfung der Wirkung vieler Verknüpfungsvarianten der *visuellen Variablen* an die Parameter

zum Einsatz kommen.

Ein Problem der entwickelten Algorithmen stellt das *Ereignisrauschen* dar, dass bei allen vier Arten der Strukturereignisse auftritt. Auch gibt es Verdeckungen der Glyphen untereinander sowie durch die Partikel, was die Erfassung der Ereignisinformationen erschwert. Im folgenden Kapitel werden diesbezügliche Verbesserungsmöglichkeiten beschrieben.

10. AUSBLICK

In der Arbeit konnte nur eine Untermenge der Einstellungsmöglichkeiten der entwickelten Cluster- und Ereigniserkennung überprüft werden, da die Berechnungen sehr lang dauern (siehe Abschnitt 7.5. Wenn das Structure Events Calculation Modul (SECalc) mit der Fähigkeit erweitert werden würde, aus einem MPDC mit Clustereinfärbungen direkt die Clusterlisten zu bilden, könnten die Nachbarschaftssuche und der CFD einmalig für jeden ihrer Parameter durchgeführt werden. Anschließend wären Änderungen der Parameter der *Ereignisheuristik* auch ohne Vorberechnungen der Ereignisse innerhalb weniger Sekunden sichtbar, so dass eine vollständige Überprüfung der Auswirkung aller Parameterkombinationen der *Ereignisheuristik* stattfinden kann.

In der Diskussion wurden Verbesserungen der Algorithmen vor allem hinsichtlich des Ressourcenverbrauchs aufgeführt (siehe Abschnitt 8.4 und Abschnitt 8.5). Zur Verringerung des starken *Ereignisrauschens* werden zwei weitere Modifikationen der Algorithmen vorgeschlagen, die in einer weiterführenden Arbeit implementiert und getestet werden könnten. So kann die *Clustergröße* desjenigen Clusters, der zusammen mit seinen *Parnterclustern* zur Bildung des Ereignisses geführt hat, direkt im Ereignis abgespeichert werden. Damit wird dem Benutzer eine Gewichtung der Ereignisse abhängig von der Clustergröße ermöglicht und er kann beispielsweise die Auswirkungen kleiner Cluster dynamisch ausblenden. Darüberhinaus könnte die Nutzung einer Mindestclustergröße mit wesentlich höheren Werten (100 Partikel oder höher) zur Senkung des *Ereignisrauschens* beitragen. Voraussetzung dafür ist die Erhöhung der durchsuchten Nachbarschaftsebenen während der Clusterreduktion (siehe Abschnitt 5.4). Dies kann zum Beispiel durch Umwandlung des iterativen Vorgehens in ein Rekursives erreicht werden. Empfehlenswert ist die Skalierung der Rekursionschritte 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. Um einen zielgerichteten Wert für die Mindestclustergröße zu finden, können die im Berechnungsmodul vorhandenen Ausgaben der berechneten, quantitativen Daten für eine (statistische) Analyse genutzt werden, etwa durch die Ausgabe der Anzahl von Clustern bestimmter Größen.

Neben dem Problem des *Ereignisrauschens* tritt das Problem der Verdeckung auf. Dies ist durch die Zielstellung der Arbeit bedingt, da die Partikel als auch die Daten der Strukturereignisse im selben Raum visualisiert werden. Dies kann zu Verdeckungsproblemen der Glyphen untereinander sowie der Verdeckung durch die Partikel führen (siehe Kapitel 8). Zur Lösung können die in Abschnitt 3.3 vorgestellten Kategorien genutzt werden, um die überlagerte Darstellung beispielsweise durch eine *explizite Verschlüsselung* in Verbindung mit der *Gegenüberstellung* zu ersetzen. Dies würde dem Verdecken entgegenwirken und könnte zudem eine quantitative Erfassung der einzelnen Ereignisarten etwa durch eine Einstellung der Dicke der Verbindungselemente zwischen dem Partikelraum und dem Ereignisraum erleichtern. Die *Strukturereignistaxonomie* aus

Abschnitt 6.2 unterstützt dabei die *explizite Verschlüsselung*, etwa durch eine freie Anordnung der Glyphen, angepasst an die Anforderungen der Anzeige der Verbindungselemente zwischen beiden Ansichten.

Zusätzliche Erkenntnisse für die Strukturveränderungen können durch, in dieser Arbeit nicht betrachtete, Zusammenhänge zwischen Clustern gebildet werden. Für die derzeitige Funktionsweise der *Ereignisheuristik* sind Clusterzusammenhänge nicht relevant (siehe vorgeführtes Negativbeispiel in Abschnitt 5.6). Sie kann jedoch damit erweitert werden, dass Cluster über mehrere oder alle Zeitschritte verglichen werden. Dies kann über die Nutzung der im SECC gebildeten Partnerclusterstrukturen erfolgen. In einer neuen Struktur können die *Großen Partner* von Clustern permanent gespeichert werden. So können die Cluster und damit die zugehörigen Ereignisse über einen längeren Zeitraum zueinander in Beziehung gestellt werden. Eine Visualisierung dieser Ereignisbeziehungen kann durch die eben vorgeschlagene Nutzung weiterer Visualisierungsarten für den Vergleich von *Ereignisobjekt* und *Partikelobjekt* erfolgen. So kann das *Ereignisobjekt* als Zeitverlaufsdiagramm dargestellt werden und trotzdem mit Hilfe der *expliziten Verschlüsselung* auf die Positionen im *Partikelobjekt* verweisen, indem zum Beispiel Linien von den Ereigniselementen zu den entsprechenden Positionen im *Partikelobjekt* gezogen werden.

Neben der weiteren Untersuchung und Verbesserung der in dieser Arbeit entwickelten Algorithmen können auch die bei der Nachbarschaftssuche entstandenen Strukturen genutzt werden, um eine mögliche, direktere Anwendung der in Abschnitt 3.2 zitierten *Konturbaumalgorithmen* zu überprüfen.

Damit können die in dieser Arbeit entstandenen Algorithmen, Strukturen und Visualisierungsmethoden für verschiedene, auf ihnen aufbauende Weiterentwicklungen sowie für weitere Strukturuntersuchungen von Molekulardynamikdaten genutzt werden.

A. ANHANG

A.1. AUFBAU DES PLUGINS

Das Plugin[1] ist in *C++* verfasst und enthält mehrere Module und Kommunikationsklassen (Call):

StaticRenderer Modul Visualisierung der Strukturereignisse mit einem *Billboard*-Shader
StructureEventsCalculation Modul Berechnungen der Strukturereignisse
StructureEventsDataCall Kommunikationsklasse zur intermodularen Weiterleitung von Daten
StructureEventsDataSource Modul Liest MMSE Dateien
StructureEventsWriter Modul Dateiausgabe für die Strukturereignisse, schreibt MMSE Dateien

Die Prämisse beim Entwerfen und Programmieren des Plugins lautete, nah bei den Kernmodulen von *MegaMol* zu bleiben. Dies erhöht die Lesbarkeit für die mit *MegaMol* Vertrauten und es können Performancevorteile genutzt werden. Im Besonderen beinhaltet dies die Verwendung von *OpenGL* 2.1 im Renderer und im *Billboard*-Shader, von Zeigern im Callmodul, von der Beschränkung auf Felder im Schreib- und Lesemodul sowie die Verwendung von Bytecode beim Dateiformat selbst.

Das Werkzeug zum Erstellen der für das Betreiben der *MegaMol* Konsole notwendigen XML Dateien und zur Visualisierung dieser Aufbauten ist der *MegaMol Configurator*.

AUFBAU DER AUSGANGSDATEN

Die vorliegenden Quelldaten der Simulation enthalten den Wert der vorzeichenbehafteten Distanzfunktion als Gleitkommawert im für die Farbdaten vorgesehenen Speicherbereich des MMPLD. Für den Farbwert wird der Typ „MultiParticleDataCall::Particles::COLDATA_FLOAT_I“ vorausgesetzt. Diese Daten sind neben den Vertexdaten für jeden Partikel im Speicher sequentiell, direkt hintereinander angeordnet. Weiterhin kann der Radius global oder für jeden Vertex gespeichert sein. Der Elementabstand wird durch den Stride bzw. Typ bestimmt (jeweils für den Vertex und die Farben). Weitere Feinheiten wie die Zusammenfassung in Partikellisten können der Spezifikation entnommen werden [Gro14, S. 3]. Die Ausgabe der Clustereinfärbung nutzt keine Partikellisten und speichert einen Rot-Grün-Blau (RGB) Farbwert sowie den Radius für jeden Partikel separat. Daher ist die Dateigröße mit etwa 7,8 GiB um 30% größer als die 5,6 GiB des Ausgangsdatensatzes.

[1] https://github.com/roidanton/mmvis_static

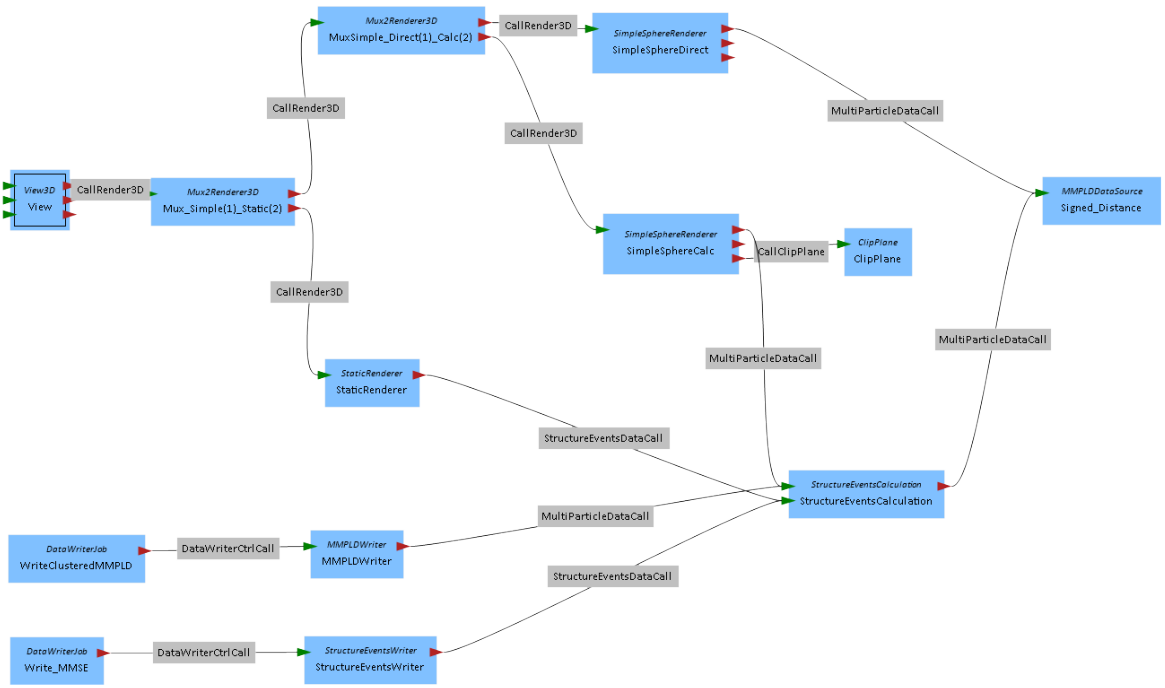


Abbildung A.1. Anordnung der *MegaMol* Module für die Überprüfung und Ausgabe der Berechnungen.

KOMMUNIKATIONSKLASSEN (CALLS)

Die Module kommunizieren über spezielle Kommunikationsklassen. Der für diese Arbeit entwickelte *StructureEventsDataCall* nutzt wie andere *MegaMol*-Calls Zeiger, um die Daten zwischen den Modulen weiterzuleiten, ohne sie im Speicher kopieren zu müssen. Das spart vor allem Speicherplatz und auch minimal Zeit für den Kopiervorgang, der durch die hohen Zugriffs- und Schreibraten des Arbeitsspeichers allerdings sehr schnell vonstatten geht. Erleichtert wird seine Arbeit, da sämtliche Listenelemente, auf die mit Zeigern zugegriffen wird – wie Partikel, Cluster und Struktureereignisse – dichtgepackte Strukturen mit Datentypen der Länge vier oder acht Byte sind, damit der Compiler kein zusätzliches Padding einfügt und so die Strides und damit die Zeiger ungültig macht.

PROGRAMMAUFBAU FÜR DIE BERECHNUNG

Wie in Abbildung A.1 zu sehen, erfolgt die Speicherung der Clustervisualisierung in einem MMPLD, wozu das *MMPLDWriter Modul* mit dem zugehörigen Job zum Einsatz kommt. Die Speicherung der Struktureereignisse geschieht durch das *StructureEventsWriter Modul* bzw. innerhalb des SECalc aufgrund spezieller Anforderungen des *MegaMol* Jobsystems. Um eine schnelle visuelle Überprüfung der Ergebnisse vornehmen zu können, werden mehrere *SimpleSphereRenderer* für die Partikel sowie die Ereignisvisualisierung mittels des *StaticRenderer Moduls* im selben Raum angezeigt. Dies ist durch die mehrfache Verschaltung des *MuxRenderers* möglich.

Verbunden sind die Daten-, Berechnungs- und Visualisierungsmodule über den Structure Events Data Call (SEDC) beziehungsweise den MPDC. Wie der MPDC sind die weiteren Module und Kommunikationsklassen Teil des *MegaMol* Kerns. Das Viewmodul steuert unter anderem den angezeigten Frame. Die Auslösung der Berechnung für die unterschiedlichen Frames erfolgt durch den View über den verbundenen *SimpleSphereRenderer*, da der aus SECalc ausgehende MPDC für den Wechsel der Frames verantwortlich ist.

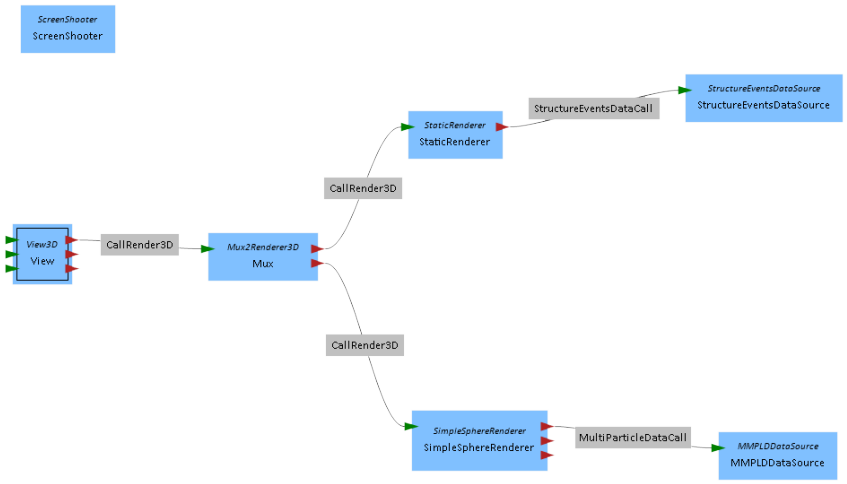


Abbildung A.2. Anordnung der *MegaMol* Module für die Ausgabe der erzeugten MMPLD und MMSE Dateien.

Für die Berechnungen aller Frames wird Jobsystem von *MegaMol* genutzt. Der Aufbau für die Jobs ist einfacher, da auf sämtliche Visualisierungsmodule verzichtet werden kann. Auch wird das *StructureEventsWriter Modul* aufgrund von Restriktionen des Jobsystems in SECalc integriert. Der Job wird auf dem *MMPLDWriter Modul* aufgerufen, an dem das SECalc mittels eines MPDC angeschlossen ist, welches seine Daten durch einen weiteren, mit dem Lesemoduls für MMPLD Dateien verknüpften MPDC bekommt. Dieses Modul liest den der Arbeit zugrundeliegenden Datensatz aus.

PROGRAMMAUFBAU FÜR DIE VISUALISIERUNG

Die Ausgabe erfordert zwei Renderer. Den *SimpleSphereRenderer* für die Visualisierung der Clustereinfärbungen und das *StaticRenderer Modul* für die Struktureereignisse. Diese werden gemäß der geforderten, überlagerten Darstellung (siehe Superposition in Abschnitt 3.3) mit Hilfe des *MuxRenderers* im selben Simulationsraum angezeigt. Ihre Daten bekommen die Renderer jeweils von einem entsprechenden Lesemodul.

Für die Ausgabe wurden Kernmodule modifiziert. Die wichtigste Modifikation ist die Ausblendung der in der Gasphase befindlichen Partikel mittels Deaktivierung von Partikeln mit der die Gasphase markierenden Farbe im entsprechenden Fragmentshader des *MegaMol* Kerns. Diese Farbe ist in der aktuellen Implementation des Plugins über einen Slot in SECalc festgelegt. Für eine langfristige Implementation dieses Features in *MegaMol* müsste sie im *SimpleSphereRenderer* durch den Nutzer einstellbar sein.

```
1   // Discard fragments with certain color.
    if (gl_Color.r == .98 &&
3     gl_Color.g == .78 &&
      gl_Color.b == 0.0)
5     discard;
```

WEITERES OPTIMIERUNGSPOTENTIAL

- *SimpleSphereRenderer*: Filtern (Ausblenden) nach Partikelfarben wie oben vorgeschlagen.
- *StaticRenderer Modul*: Visuelle Ausgabe von quantitativen Ereigniswerten, analog zur Protokollierung in SECalc .

- Globales Kennzeichen für die Logausgabe analog zum Protokollkennzeichen in SECalc .
- WASD-QE-RF Steuerung, als Vorbild dienen die guten User Interfaces von Spielen wie Elite Dangerous.
- Hinzufügen eines Pfades für Assets, indem zur megamol.cfg ein weiteres Element <assetdir> mit dem Attribut path hinzugefügt wird, analog zum Element <shaderdir>.

A.2. AUFZÄHLUNG DER CLUSTER- UND EREIGNISERKENNUNGSSCHRITTE

1. a) Aufbau der Partikelliste mit den Daten des MPDC.
   b) Erstellung von kD-Bäumen für die Nachbarschaftserkennung.
   c) Nutzung des kD-Baum Suchalgorithmus der ANN-Bibliothek , um Nachbarn jedes Partikels zu diesem hinzuzufügen unter Berücksichtigung des *Partikelradiusmultiplikator*.
2. a) Cluster erstellen unter Nutzung der Nachbarn.
   b) Reduktion der *Mindestgrößencluster* innerhalb von Zusammenhangskomponenten, die weniger Partikel besitzen als für die benutzerdefinierte minimale *Clustergröße* erlaubt.
3. SECC durch Nutzung der *Clustervergleichsmatrix* und Erstellung von zwei Listen mit *Partnerclustern* in vorwärtiger beziehungsweise rückwärtiger Richtung.
4. Anwendung von Verhältnisberechnungen auf diese Listen und Nutzung von benutzerdefinierten Grenzwerten für die *Ereignisheuristik*.

A.3. GLYPHPROTOTYP

Zur Überprüfung einer sinnvollen Zuweisung von Datenparametern zu den *visuellen Variablen* dienen Mockups. Aufgrund der Vielzahl der in Abschnitt 6.2 genannten Kombinationsmöglichkeiten wird eine programmatische Erstellung der Mockups gewählt.

Dazu wird ein Glyphprototyp geschrieben, der im Webbrowser verwendet werden kann[2]. 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. Die Geschwindigkeit der graphischen Anzeige spielt bei der Erstellung von Mockups eine untergeordnete Rolle. Als Sprache wird *C#* gewählt.

Der Glyphprototyp basiert auf einer älteren Entwicklungsstufe der *Struktureignistaxonomie*, in der die Opazität vorhanden war und die Textur fehlte. Dies wirkt sich nicht nachteilig aus. Zum einen erfolgt im Prototyp keine überlagerte Darstellung mit einem anderen Datenobjekt. Zum anderen wird er in dieser Arbeit nur zur Überprüfung der dreidimensionalen Glyphdarstellung genutzt. Der Vorteil in der Verwendung der Opazität liegt darin, dass so das in Abschnitt A.3 beschriebene, günstige Verhalten der Opazität bei der Zuweisung zur Häufung genutzt werden kann.

ERSTELLUNG DER QUELLDATEN

Um eine Vergleichbarkeit der verschiedenen Mockups untereinander zu gewährleisten, werden die Quelldaten fest vorgegeben. Sämtliche Werte wurden für die Datenparameter Ort und Zeit so gewählt, dass eine Überlappung der Ereigniselemente von nebeneinanderliegenden

[2] http://mmm.takera.de

Ereignissen bei einer Zuweisung an die *visuelle Variable* 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 A.1 direkt an die Zeit t gekoppelt.

$$x = \left\lfloor \frac{4}{3} \cdot t + 0,5 \right\rfloor \tag{A.1}$$

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 Ereignisart erfolgt manuell. Da die Partikel des MMPLD Quelldatensatzes zu Beginn der Simulation einen einzigen Cluster in flüssiger Phase bilden (siehe Abschnitt 4.1), werden im Prototypen bei kleinen Zeitwerten vornehmlich *Splits* zugewiesen.

Zur Darstellung der in Abschnitt 6.2 beschriebenen Häufung, werden einige Ereignisse jeweils gruppiert und händisch an denselben Ort und dieselbe Zeit gesetzt. Weiterhin wird, wie in Abschnitt 6.2 empfohlen, die Häufung als Ereignisparameter implementiert, um ihr visuelle Variablen zuweisen zu können.

ZUWEISUNG DER ZAHLENWERTE

Zur Gewährleistung der in Abschnitt A.3 beschriebenen Vermeidung unnötiger Überlappungen der Ereigniselemente wird der Standardwert der *visuellen Variable* Größe, d.h. wenn die Variable keinem Parameter zugewiesen ist, entsprechend skaliert. Um nah bei der Visualisierung der Molekularsimulation selbst zu bleiben, wird in den Mockups, die die *visuelle Variable* Form nicht beinhalten, eine Kugel verwendet.

Die Variablen in den Mockups weisen folgende Zahlenbereiche auf:

- x-, y-, z-Position: $[-\infty, \infty]$, beim Parameter Ort erfolgt eine direkte Zuweisung
- Farbwert: $[0, 340]°$ 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 Variablen* a_{min} und a_{max} bekannt sind und sich für die Parameter p des Datensatzes ein minimaler und maximaler Wert p_{min} und p_{max} bestimmen

Tabelle A.1. Festlegung der Attributeigenschaften für die Ereignisart in den Mockups.

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

lässt, wird für die Berechnung der Variablenwerte die Linearfunktion in Gleichung A.2 verwendet.

$$\begin{aligned}
a &= m \cdot p + n \\
a_{\max} &= m \cdot p_{\max} + n \\
a_{\min} &= m \cdot p_{\min} + n \\
m &= \boxed{\frac{a_{\max} - n}{p_{\max}}} \\
n &= a_{\min} - m \cdot p_{\min} = a_{\min} - \frac{a_{\max} - n}{p_{\max}} \cdot p_{\min} \\
&= a_{\min} - a_{\max} \cdot \frac{p_{\min}}{p_{\max}} + n \cdot \frac{p_{\min}}{p_{\max}} \\
n \cdot p_{\max} &= a_{\min} \cdot p_{\max} - a_{\max} \cdot p_{\min} + n \cdot p_{\min} \\
n \cdot (p_{\max} - p_{\min}) &= a_{\min} \cdot p_{\max} - a_{\max} \cdot p_{\min} \\
n &= \boxed{\frac{a_{\min} \cdot p_{\max} - a_{\max} \cdot p_{\min}}{p_{\max} - p_{\min}}}
\end{aligned} \tag{A.2}$$

Anschließend erfolgt eine Rundung zur handlichen Verwendung der Ergebnisse.

$$a = \lfloor m \cdot p + n + 0,5 \rfloor \tag{A.3}$$

Aufgrund des großen Umfangs und der geringen Relevanz der Berechnungsergebnisse sind sie hier nicht aufgeführt.

ZUWEISUNG DES VOKABULARS

Da die Mächtigkeit M des Vokabulars der Ereignisart bekannt ist, kann das Intervall v der *visuellen Variable a* in Gleichung A.4 zur rechnerischen Ermittlung der Variablenwerte a_i genutzt werden.

$$\begin{aligned}
M &= 4 \\
\Rightarrow p_{0\ldots3} &= \left(0, \frac{1}{3}, \frac{2}{3}, 1\right) \\
v &= a_{\max} - a_{\min}
\end{aligned} \tag{A.4}$$

$$a_i = \left\{ p_i \cdot v + a_{\min} \,|\, 0 \leq i \leq M - 1 \right\}$$

Ausnahmen bilden die *visuellen Variablen* Farbe und Form. Wegen der kleinen Mächtigkeit des Vokabulars und zur Sicherung einer optimale Aufteilung des Farbwerts, wird dieser in Tabelle A.1 manuell definiert. Für die Form werden zur optimalen Unterscheidung vier Varianten aus der Menge der *platonischen Körper* ausgewählt.

KONSTANTER VARIABLENWERT BEI OPAZITÄT

Die Reservierung der Opazität für die Häufung ist sinnvoll, 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 wird zum Beispiel bei der Zuweisung des visuellen Attributes Größe eine Größenveränderung nachträglich für jedes Ereignis in der Anhäufung auf der Basis eines temporalen oder lokalen Häufungswertes durchgeführt.

Aufgrund dieses günstigen Verhaltens der Opazität wird ihr ein konstanter Variablenwert zugewiesen, so dass keine zusätzlichen Ereigniswerte für die Häufung benötigt werden. Dieser konstante Wert wird auf 50% Opazität festgesetzt.

ZUFÄLLIGE DATEN

Mit den festgestellten Grenzen und entwickelten Gleichungen können neben den festen Quelldaten auch zufällig verteilte Daten erzeugt werden, die den Simulationsraum nicht überschreiten. Da dieser Prozess automatisiert ist, wird dem Nutzer die Möglichkeit gegeben, über einen Slider die Anzahl der zu erzeugenden Datensätze in einem großen Intervall einzustellen.

UNTERSCHEIDBARKEIT VON POLYEDERN

Da die Unterscheidung von Polyedern mit zunehmender Flächenanzahl schwieriger wird, findet im Glyphprototyp 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 Polygonnetze die maximale Dimension von 1 (einheitenlos) entsprechend der Dimension der kugelförmigen Standardelemente in *Unity3D* gewählt.

A.4. UMFRAGE

A.5. BERECHNUNGEN GLYPH

Die folgenden Berechnungen sind abhängig von der Ereigniszeit t mit t_{max} als höchsten Zeitpunkt aller Ereignisse. Die Farben werden im Hue-Saturation-Value (HSV)-Farbraum berechnet.

Berechnung des Farbwertes h im Intervall $[0, 1]$ mit $h_{offset} = \frac{30°}{360°}$ als Versatz zur Vermeidung identischer Farbwerte am Anfang und Ende des Zeitraums. Die Sättigung s und Helligkeit v sind maximal $s = v = 1$.

$$h = \left(\frac{t}{t_{max}} \right) \cdot (1 - h_{offset}) + h_{offset} \tag{A.5}$$

Berechnung der Helligkeit v im Intervall $[0, 1]$ mit $b_{offset} = 0,4$ als Versatz zur Vermeidung identischer Helligkeitswerte am Anfang und Ende des Zeitraums. Der Farbwert ist ein Blauton $h = \frac{208°}{360°}$ und die Helligkeit v ist maximal $v = 1$.

$$v = \left(\frac{t}{t_{max}} \right) \cdot (1 - b_{offset}) + b_{offset} \tag{A.6}$$

Berechnung der Höhe der dunkelgrauen Texturfarbe p im Intervall $[0, 1]$ mit $p_{\text{offset}} = 0,01$ als Versatz zur Korrektur des unteren Transparenzbereiches der Textur.

$$p = \left(\frac{t}{t_{\text{max}}} \right) \cdot (1 - p_{\text{offset}}) + p_{\text{offset}} \tag{A.7}$$

A.6. VERWENDETE HARDWARE

Das Plugin wurde auf folgendem System entwickelt:

- Prozessor: Intel Core i5-3210M @2,5GHz
- Arbeitsspeicher: 2x4 GiB DDR3-1600
- Betriebssystem: Windows 8.1

Die Berechnung aller Zeitschritte wurde aufgrund des hohen Ressourcenverbrauchs auf folgendem System durchgeführt:

- Prozessor: Intel Core i5-3570K @3,4GHz
- Arbeitsspeicher: 2x8 GiB DDR3-1600
- Betriebssystem: Windows 8.1

A.7. 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 eine Headerbibliothek zur Speicherung von Vektoren analog zu OGL
lodepng zum Laden von Texturen
MegaMol mit der ANN-Bibliothek , AntTweakBar, mmstd_datatools, vislib
MegaMolConf
VisualStudio 2013 mit OpenMP und Parallel Patterns Library

KONTURBÄUME

In der Implementation nicht genutzt, aber aufgrund des Schwerpunktes in der Analyse mit aufgeführt.

SimpleCT Repository[3] [AH13]

[3] https://github.com/baranaydogan/SimpleCT/

libtourtre Website[4], basierend auf [CSA01a] und [PCS04]

A.8. ALTERNATIVE BESTIMMUNG DER CLUSTER OHNE NACHBARSCHAFTSKENNTNISSE

Partikel eines Clusters können über eine radialbasierte Nachbarschaftsbeziehung bestimmt werden, 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, analog zum ersten Schritt des Abtastalgorithmus (siehe Abschnitt 3.2). 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 zum anderen müsste der nächste *Tiefenwert* 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 betrachtetem 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.

Dieser Algorithmus ist wegen der Entfernungsbestimmung über den Radius sehr langsam. Zur Beschleunigung könnte die Entfernungsbestimmung bei den Partikeln auf solche mit ähnlichen Tiefenwerten eingeschränkt werden oder es könnte eine Einteilung des Raumes in ein Gitter erfolgen und jede Zelle erhält ihre eigene Partikelliste. Bei der Suche nach benachbarten Partikeln werden nur die Nachbarzellen durchsucht. Allerdings ist auch bei diesen Varianten immer noch eine Entfernungsbestimmung notwendig, wenn auch mit wesentlich weniger Überprüfungen je Partikel.

Daher werden diese Methoden nicht weiter verfolgt und stattdessen wird auf eine Baumstruktur zur Nachbarschaftsbildung zurückgegriffen und von diesen Daten aus eine Clusterbestimmung vorgenommen (siehe Abschnitt 5.2).

[4] http://graphics.cs.ucdavis.edu/ sdillard/libtourtre/doc/html/

LITERATUR

[AH13] D.B. Aydogan und J. Hyttinen. „Contour tree connectivity of binary images from algebraic graph theory". Englisch. In: *IEEE International Conference on Image Processing (ICIP)*. Melbourne, Australia, 2013.

[All+15] Andy Allan, Steve Hill, Christoph Eckert und Harry Wood et al. *Contours*. Englisch. 2015. URL: http://wiki.openstreetmap.org/wiki/Contours (besucht am 23. 05. 2015).

[AN15] Aditya Acharya und Vijay Natarajan. „A parallel and memory efficient algorithm for constructing the contour tree". In: *Visualization Symposium (PacificVis), 2015 IEEE Pacific*. Apr. 2015, S. 271–278. DOI: 10.1109/PACIFICVIS.2015.7156387.

[Au+08] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or und Tong-Yee Lee. *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).

[Bor+13] Rita Borgo, Johannes Kehrer, David H Chung, Eamonn Maguire, Robert S Laramee, Helwig Hauser, Matthew Ward und Min Chen. „Glyph-based visualization: Foundations, design guidelines, techniques and applications". In: *Eurographics State of the Art Reports* (2013), S. 39–63.

[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.

[Bri14] W.C. Brinton. *Graphic Methods for Presenting Facts*. Industrial management Library. Engineering Magazine Company, 1914. URL: https://books.google.de/books?id=LbQYAAAAMAAJ.

[BWW13] Eric J. Bylaska, Jonathan Q. Weare und John H. Weare. „Extending molecular simulation time scales: Parallel in time integrations for high-level quantum chemistry and complex force representations". In: *The Journal of Chemical Physics* 139.7, 074114 (2013). DOI: http://dx.doi.org/10.1063/1.4818328. URL: http://scitation.aip.org/content/aip/journal/jcp/139/7/10.1063/1.4818328.

[Cao+13] Xiaobo Cao, Jeremy L Yap, MK Newell-Rogers, Chander Peddaboina, Weihua Jiang, Harry T Papaconstantinou, Dan Jupitor, Arun Rai, Kwan-Young Jung, Richard P Tubin u. a. „The novel BH3 α-helix mimetic JY-1-106 induces apoptosis in a subset of cancer cells (lung cancer, colon cancer and mesothelioma) by disrupting Bcl-xL and Mcl-1 protein–protein interactions with Bak". In: *Molecular Cancer* 12.1 (2013), S. 42.

[Chi+05] Yi-Jen Chiang, Tobias Lenz, Xiang Lu und Günter Rote. „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.

[Cor+05] Nicu D Cornea, Deborah Silver, Xiaosong Yuan und Raman Balasubramanian. „Computing hierarchical curve-skeletons of 3D objects". In: *The Visual Computer* 21.11 (2005), S. 945–955.

[CS07] Jordi Cohen und Klaus Schulten. „O2 migration pathways are not conserved across proteins of a similar fold". In: *Biophysical journal* 93.10 (2007), S. 3591–3600.

[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.

[DBW84] P Delhaise, M Bardiaux und S Wodak. „Interactive computer animation of macromolecules“. In: *Journal of Molecular Graphics* 2.4 (1984), S. 103–106. ISSN: 0263-7855. DOI: http://dx.doi.org/10.1016/0263-7855(84)80002-8. URL: http://www.sciencedirect.com/science/article/pii/0263785584800028.

[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, John Harer, Ajith Mascarenhas und Valerio Pascucci. „Time-varying reeb graphs for continuous space-time data“. In: *Proceedings of the twentieth annual symposium on Computational geometry*. ACM. 2004, S. 366–372.

[FF06] Joachim Funke und Peter A Frensch. *Handbuch der Allgemeinen Psychologie-Kognition*. Hogrefe Verlag, 2006.

[Gle+11] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen und Jonathan C. Roberts. „Visual comparison for information visualization“. In: *Information Visualization* 10.4 (2011), S. 289–309. DOI: 10.1177/1473871611416549. eprint: http://ivi.sagepub.com/content/10/4/289.full.pdf+html. URL: http://ivi.sagepub.com/content/10/4/289.abstract.

[Gra15] Sam Gratrix. *WebGL Uniform Polyhedra*. Englisch. 2015. URL: http://gratrix.net/polyhedra/webgl/ (besucht am 06. 03. 2015).

[Gro+07] Sebastian Grottel, Guido Reina, Jadran Vrabec und Thomas Ertl. „Visual Verification and Analysis of Cluster Detection for Molecular Dynamics“. In: *Proceedings of IEEE Visualization '07*. Bd. 13. 6. 2007, S. 1624–1631. DOI: 10.1109/TVCG.2007.70614. URL: http://doi.acm.org/10.1109/TVCG.2007.70614.

[Gro+14] S. Grottel, J. Heinrich, D. Weiskopf und S. Gumhold. „Visual Analysis of Trajectories in Multi-Dimensional State Spaces“. In: *Computer Graphics Forum* 33.6 (2014), S. 310–321. ISSN: 1467-8659. DOI: 10.1111/cgf.12352. URL: http://dx.doi.org/10.1111/cgf.12352.

[Gro14] Sebastian Grottel. *File Format Specification MMPLD*. Englisch. 2014. URL: https://cloud.visus.uni-stuttgart.de/public.php?service=files&t=b859555761a09c5857a705b&download= (besucht am 28. 04. 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).

[HE12] Christopher G Healey und James T Enns. „Attention and visual memory in visualization and computer graphics“. In: *Visualization and Computer Graphics, IEEE Transactions on* 18.7 (2012), S. 1170–1188.

[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.

[Inf01] Elly Infantidou. *Evidentials and relevance*. John Benjamins Publishing, 2001. ISBN: 9789027251053. DOI: 10.1075/pbns.86. URL: https://benjamins.com/#catalog/books/pbns.86/main.

[Joh66] Norman W. Johnson. „Convex Solids with Regular Faces“. In: *Canadian Journal of Mathematics* 18 (1966), S. 169–200. ISSN: 0008-414x.

[Kee13] R Keesman. „Dynamical eigenmodes of various bead-spring systems“. In: (2013).

[Kre+97] Marc van Kreveld, René van Oostrum, Chandrajit Bajaj, Valerio Pascucci und Dan Schikore. „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.

[Lan+06] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller und V. Pascucci. „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.

[Lim13] M. Lima. *Visual Complexity: Mapping Patterns of Information*. Princeton Architectural Press, 2013. ISBN: 9781616892197. URL: https://books.google.de/books?id=59xlmQEACAAJ.

[Lim15] Manuel Lima. *Visual complexity*. Englisch. 2015. URL: http://www.visualcomplexity.com/vc/ (besucht am 14. 04. 2015).

[Mag+12] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies und Min Chen. „Taxonomy-Based Glyph Design - with a Case Study on Visualizing Workflows of Biological Experiments“. In: *Visualization and Computer Graphics, IEEE Transactions on* 18.12 (Dez. 2012), S. 2603–2612. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.271.

[Mag+13] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies und Min Chen. „Visual Compression of Workflow Visualizations with Automated Detection of Macro Motifs“. In: *Visualization and Computer Graphics, IEEE Transactions on* 19.12 (Dez. 2013), S. 2576–2585. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.225.

[McB13] Carl McBride. *Path integral formulation*. Englisch. Universidad Nacional de Educación a Distancia. 2013. URL: http://www.sklogwiki.org/SklogWiki/index.php/Path_integral_formulation (besucht am 22. 07. 2015).

[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.

[NRG08] Dmitry Nerukh, Vladimir Ryabov und Robert C. Glen. „Complex temporal patterns in molecular dynamics: A direct measure of the phase-space exploration by the trajectory at macroscopic time scales". In: *Phys. Rev. E* 77 (3 März 2008), S. 036225. DOI: 10.1103/PhysRevE.77.036225. URL: http://link.aps.org/doi/10.1103/PhysRevE.77.036225.

[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, Giorgio Scorzelli, Peer-Timo Bremer und Ajith Mascarenhas. „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).

[RS14] Benjamin Raichel und C. Seshadhri. „A Mountaintop View Requires Minimal Sorting: A Faster Contour Tree Algorithm". In: *CoRR* abs/1411.2689 (2014). URL: http://arxiv.org/abs/1411.2689.

[Sat+11] Chaitanya Sathe, Xueqing Zou, Jean-Pierre Leburton und Klaus Schulten. „Computational investigation of DNA detection using graphene nanopores". In: *ACS nano* 5.11 (2011), S. 8842–8851.

[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.

[Sha+07] Andrei Sharf, Thomas Lewiner, Ariel Shamir und Leif Kobbelt. „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.

[Sma+13] Meagan C Small, Pedro Lopes, Rodrigo B Andrade und Alexander D MacKerell Jr. „Impact of ribosomal modification on the binding of the antibiotic telithromycin using a combined grand canonical monte carlo/molecular dynamics simulation approach". In: (2013).

[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.

[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.

[Tag+12] Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson und Hao Zhang. „Mean curvature skeletons“. In: *Computer Graphics Forum*. Bd. 31. 5. Wiley Online Library. 2012, S. 1735–1744.

[Tan+14] San Tang, Yi Guo, Yuanyuan Wang, Wanli Cao und Fukang Sun. „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.

[Tre+13] Alexander JB Trevor, Suat Gedikli, Radu B Rusu und Henrik I Christensen. „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, Olga Sorkine, Yaron Lipman und Craig Gotsman. „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.

[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

Überlagerung, 14, 36

ANN-Bibliothek, 21, 63, 72, 76
archimedischer Körper, 32

Billboard, 8, 40, 69
Birth, 6–8, 28, 34, 40, 47, 48, 52, 53, 56, 61, 62, 74
Blender, 33, 75

catalanischer Körper, 32
CFD-Algorithmus, 7, 18, 22, 23, 26, 27, 44, 50, 59–63, 65, 67
Clustergröße, 27, 42, 67, 72
Clustervergleichsmatrix, 27, 47, 72

Death, 6–8, 28, 34, 37, 40, 47, 48, 52, 53, 61, 62, 74

Einpartikelcluster, 47, 52, 53, 61, 62
Ereignisglyph, 14, 15, 32, 36, 42, 43, 49, 50, 62, 65
Ereignisheuristik, 42, 50, 65, 67, 68, 72
Ereignisobjekt, 31, 35, 49, 52, 56, 68
Ereignisrauschen, 7, 50, 52, 54, 60, 62, 65–67
explizite Verschlüsselung, 14, 67, 68

FLANN-Bibliothek, 63

Gegenüberstellung, 14, 67
Gesetz der Ähnlichkeit, 32, 35
Gesetz der gemeinsamen Bewegung, 32, 35
Gesetz der gemeinsamen Region, 35
Gesetz der Geschlossenheit, 32, 35, 39
Gesetz der Gleichzeitigkeit, 35
Gesetz der Kontinuität, 32, 35
Gesetz der Nähe, 32, 35
Gesetz der Prägnanz, 32, 35
Gesetz der verbundenen Elemente, 73
Glyphprototyp, 36, 39, 65
Großer Partner, 28, 29, 68

Isolinie, 12
Isooberfläche, 12–14, 27
Isowert, 12, 14

Johnson-Körper, 32

Kepler-Poinsot-Körper, 33
Kontur, 12, 27
Konturbaum, 9–14, 25, 27, 65, 68

MegaMol, 14, 17, 21, 29, 36, 39, 40, 43, 62, 65, 69–71
MegaMol Configurator, 17, 69
Merge, 6–8, 28, 29, 34, 37, 40, 47–49, 52, 54, 74
Mindestgrößencluster, 57, 72
MMPLDWriter Modul, 70, 71
MMSE-Datei, 29, 30, 69, 71
MuxRenderer, 70, 71

OpenGL, 40, 69, 76
OpenMP, 26, 63

Parallel Patterns Library, 63
Parntercluster, 67
Partikelobjekt, 31, 35, 36, 52, 68
Partikelradiusmultiplikator, 22, 46, 50, 53, 55, 57–61, 65, 72
Partnercluster, 27–29, 42, 43, 47–50, 59, 72
Partnerclusterstrukturliste, 27–29
platonischer Körper, 32, 33, 74, 75

Saatpartikel, 77

SECC-Algorithmus, 8, 26, 27, 44, 58, 59, 64, 65, 68, 72
SimpleSphereRenderer, 70, 71
Split, 6–8, 28, 29, 34, 40, 47–49, 54, 73, 74
StaticRenderer Modul, 17, 42, 43, 57, 69–71
StructureEventsCalculation Modul, 17, 67, 69–72
StructureEventsDataCall, 17, 69, 70
StructureEventsDataSource Modul, 17, 69
StructureEventsWriter Modul, 17, 69–71
Strukturereignistaxonomie, 16, 33–36, 39, 62, 65, 67, 72

Tiefenwert, 7, 8, 19, 20, 22, 25, 26, 61, 62, 65, 77

Unity3D, 72, 75

visuelle Variable, 15, 16, 31, 33–36, 39, 40, 42, 43, 56, 57, 65, 72–74
Vokabular, 33–36

Wurzelpartikel, 20, 25, 26, 42, 43, 63

Zahlenwert, 33–35, 40