BERGISCHE
UNIVERSITÄT
WUPPERTAL

Fakultät für Elektrotechnik
Informationstechnik und Medientechnik

Elektromagnetische Felder in der virtuellen Realität

Dissertation zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

eingereicht durch

Robert Roth

bei der

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Fakultät für Elektrotechnik, Informationstechnik und Medientechnik

Prüfungsausschuss

Erstprüfer	Prof. Dr. Reinhard Möller
Zweitprüfer	Prof. Dr. Dietmar Tutsch

Wuppertal, 2023

Elektromagnetische Felder in der virtuellen Realität

Robert Roth

19. Oktober 2023

Inhaltsverzeichnis

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Entwicklung von Algorithmen zur Darstellung elektromagnetischer Felder in der virtuellen Realität. Dabei werden unter anderem didaktische und akademische sowie Anwendungsfälle aus der Spieleentwicklung untersucht und dargelegt, in welchen dieser Anwendungsfälle dieser Algorithmen zur Verbesserung des Benutzererlebnisses sowie der Qualität der dargestellten Ergebnisse zuträglich sein kann. Dazu werden die entwickelten Algorithmen vorgestellt und ihre Anwendung in verschiedenen Szenarien untersucht.

Zunächst wird erläutert, wie Felder üblicherweise durch Methoden der Computergraphik dargestellt werden können und welche Probleme aus diesen Methoden erwachsen. Besonderes Augenmerk liegt dabei auf Abwägungen von Pfeilen, Linien und Isoflächen als geeignete Darstellungsmethoden und den mit ihm zusammenhängenden Problematiken. Im Anschluss daran werden die alternativen Techniken der schrittweisen Suche und des Shrink-Wrap-Verfahrens unter Berücksichtigung einer Simulation zur Darstellung von Isoflächen vorgestellt. Probleme, die sich aus der Startgeometrie beim Einsatz des Shrink-Wrap-Verfahrens ergeben, werden ebenfalls aufgezeigt und mögliche Anwendungsbereiche der schrittweisen Suche untersucht. Als Verfahren zur Erkennung von Fehlannahmen bezüglich der Äquivalenz zweier Punktladungen werden Punktbeinhaltungstests und in diesem Sonderfall anwendbare Alternativen sowie als Möglichkeiten der Reaktion auf diese Erkennung das Zusammenfügen und das selektive Einblenden von Meshes eingeführt. Schließlich werden die drei diskutierten Verfahren verglichen, um eine allgemeine Empfehlung für ihre Anwendbarkeit in verschiedenen Einsatzgebieten abzuleiten. Als Sonderfall für die schrittweise Suche wird der Anwendungsbereich der makro-kontrollierten Generierung von Videospielgeometrie vorgestellt und darauf aufbauende Verfahren zur Verbesserung und effizienteren Nutzung diskutiert.

Abstract

The present work deals with the development of algorithms for the representation of electromagnetic fields in virtual reality. Among other things, didactic and academic as well as use cases from game development are examined and it is explained in which of these use cases these algorithms can be beneficial for improving the user experience as well as the quality of the displayed results. For this purpose, the developed algorithms are presented and their application in different scenarios is examined.

First, it is explained how fields can usually be represented by methods of computer graphics and which problems arise from these methods. Particular attention is paid to considerations of arrows, lines and isosurfaces as suitable methods of representation and the problems associated with it. Subsequently, the alternative techniques of stepwise search and the shrink-wrap method are presented, taking into account a simulation for the representation of isosurfaces. Problems arising from the starting geometry when using the shrink-wrap method are also pointed out and possible areas of application of the stepwise search are examined. As procedures for the detection of false assumptions regarding the equivalence of two point charges, point leg retention tests and alternatives applicable in this special case are introduced, as well as merging and selective blending of meshes as possibilities for reacting to this detection. Finally, the three methods discussed are compared in order to derive a general recommendation for their applicability in different fields of application. As a special case for step-by-step search, the application area of macro-controlled generation of video game geometry is presented and methods based on it for improvement and more efficient use are discussed.

Einführung

Diese Arbeit befasst sich mit der Visualisierung elektrischer Felder. Bei diesen handelt es sich um theoretische Konstrukte zur Erklärung elektrotechnischer Phänomene wie Blitze oder physikalische Kräfte, die von geladenen Körpern auf andere ausgeübt werden. Der Zusammenhang zwischen diesen Phänomenen und elektrischen Feldern wurde erst spät durch Physiker wie Carl Friedrich Gauß erforscht und im 19. Jahrhundert durch James Clerk Maxwell zu einer Theorie vereinigt.

Diese Theorie ist allerdings aufgrund ihrer abstrakten Natur Menschen nur schwer zugänglich; Laien werten elektrische Felder oft als ein weiteres der benannten physikalischen Phänomene und legen für ihr Verständnis unwissenschaftliche Modelle dieser Felder wie „Elektrosmog“ zugrunde. Kontroversen über den Einfluss elektrischer Felder auf die Gesundheit verstärken das Unbehagen einer wissenschaftlichen Auseinandersetzung mit diesem Thema; Dass elektrische Felder nicht unmittelbar wahrnehmbar sind und dass jedes elektrische Gerät und jede elektrische Leitung ein elektrisches Feld von nicht intuitiv abschätzbarer Stärke erzeugt, nährt Ängste von Menschen vor den Folgen, elektrischen Feldern ausgesetzt zu sein, denen sie sich nicht entziehen können. Aber auch wissenschaftlich interessierten Personen mit dem Wunsch, elektrische Felder zu verstehen, fällt das Verständnis der mathematischen Zusammenhänge und Modelle ohne zumindest abstrakte bildliche Darstellung schwer. Diese Tatsache hat über die Zeit verschiedene Verfahren zu visuellen Darstellungen solcher Felder motiviert: Im Schulunterricht wird die Feldliniendarstellung gelehrt, da sie sich gut anhand kleiner Experimente, beispielsweise mit Magneten und Metallspänen, rekonstruieren lässt. Die Computergraphik ist heutzutage imstande, die einstmals in Büchern abgedruckten Visualisierungen elektrischer Felder auf Bildschirmen darzustellen. Die weit verbreiteten Feldlinien- und Pfeildarstellungen weisen allerdings Defizite auf, die bei dem Versuch sichtbar werden, sie in den Anschauungsraum zu übertragen, in der Komplexität der Ladungsverteilung über primitive Anordnungen wie einige punktförmige Ladungen oder einen Stabmagneten hinauszugehen, oder eine Analyse des Feldes aus großer oder sehr kleiner Entfernung der Feldquelle durchzuführen. Daher wird eine Darstellung der intuitiveren, allerdings in der Berechnung und

Darstellung komplexeren Isoflächen angestrebt. Ein Beispiel einer solchen Isofläche ist in Abbildung 2 dargestellt. Die Idee der Darstellung von elektrischen Feldern auf diese Weise folgt dem Konzept der Strahlungscharakteristiken [SB18], die die Stärke der von Antennen ausgestrahlten Radiowellen abhängig von ihrer Richtung darstellen. Isoflächen finden bereits im Bereich der Darstellung elektrischer Felder durch die Finite-Elemente-Methode Anwendung. Die oben angeführten Darstellungsprobleme der herkömmlichen Repräsentation elektrischer Felder bestehen dabei fort, da zweidimensionale Bildschirme keinerlei Tiefeneindruck der dargestellten dreidimensionalen Geometrie ermöglichen. Augmented Reality – die Erweiterung der reinen Realität durch virtuelle Elemente – kann hier als eine relativ neuartige Technologie grundlegend dazu beitragen, elektrische Felder zu visualisieren. Mittels Augmented Reality können durch virtuelle Realität angezeigte Ergebnisse mit der rein realen Quelle des elektrischen Feldes visuell überlagert und deren Zusammenhang somit verständlich gemacht werden. Die erforderlichen Rendering-Algorithmen für die Erstellung von Isoflächen liegen zwar vor, sie weisen jedoch bezüglich der Darstellung durch Augmented Reality Nachteile auf. Beispielsweise können sie auf Raytracing-Techniken basieren, die mit der virtuellen Realität aus mehreren Gründen, unter anderem ihrer Unvereinbarkeit ihrer Beleuchtungsmodelle mit der reinen Realität nicht kompatibel sind, oder viele Berechnungen fernab der Oberflächen des darzustellenden Körpers durchführen. Methoden, die die Isofläche durch ein Mesh annähern, so dass ihre Darstellung durch andere Rendering-Verfahren erzeugt werden könne, leiden unter inhärenten Defekten, unter anderem, dass sie keine sauberen Meshes produzieren, was zu Darstellungsfehlern führen kann. Dies wirft die Frage nach effizienteren Methoden zur Berechnung von Isoflächen auf, die zu ermitteln der Kern dieser Arbeit ist. Es werden Methoden vorgestellt, die das Problem überflüssiger Berechnung überwinden und gleichzeitig Defizite bei der erstellten Geometrie beheben. Weiterhin wird eine weitere Anwendungen für die vorgestellten Algorithmen im Bereich der prozeduralen Generierung von Geometrie aufgezeigt.

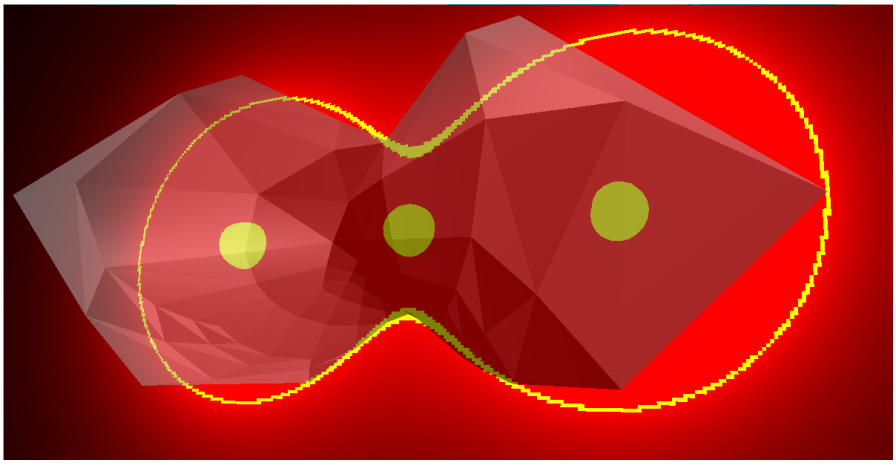


Abbildung 2: Ein durch ein Mesh (grau), eine Höhenkarte in drei Dimensionen und den Marching Squares-Algorithmus (gelb) in zwei Dimensionen repräsentiertes elektrisches Feld

Aufbau der Arbeit

Ziel dieser Arbeit ist das Vorstellen von Methoden zur beschleunigten Generierung von Geometrie, die Isoflächen repräsentiert. Dazu werden in Kapitel 1 zunächst gängige Darstellungsmethoden besprochen und miteinander verglichen, wobei ihre Vor- und Nachteile bei der Verwendung in der virtuellen Realität herausgearbeitet werden. Besonderes Augenmerk liegt dabei auf den Isoflächen und weit verbreiteten Verfahren ihrer Darstellung sowie deren Eignung für die Verwendung in VR-Anwendungen. Schließlich befasst sich dieses Kapitel mit Software-Anwendungen, die sich diese Verfahren zu Nutze machen und evaluiert die Aussichten, diese um einen Zugang zur virtuellen Realität zu erweitern.

Danach liefert die Arbeit die Grundlagen für das Verständnis der im Verlauf der Arbeit diskutierten Eigenschaften von Isoflächen erforderlichen mathematischen Konzepten und führt in Kapitel 2 die benötigte Theorie von Skalar- und Vektorfeldern und Konzepte der Informatik wie Monoiden und der Computergraphik wie das Begrenzungsflächenmodell ein. Im weiteren Verlauf des Kapitels werden die verwendeten Begriffe für elektrische Felder geklärt und auf Skalarfelder zurückgeführt. Da im Verlauf des Hauptteils neue Algorithmen vorgestellt werden, führt dieses Kapitel ebenfalls das Hoare-Kalkül als Mittel zum Beweis der Terminierung und der Korrektheit sowie die Landau-Symbole für die Analyse der Zeit- und Speicherkomplexität ein.

Auf diesen Grundlagen aufbauend werden in Kapitel 3 Voraussetzungen für Isoflächen herausgestellt, die ein Skalarfeld sinnvoll repräsentieren. Dazu werden Eigenschaften dieser Felder ermittelt und begründet, die für die sie zu repräsentierenden Meshes ebenfalls gelten müssen. Die darauf aufbauenden Untersuchungen stellen Versuche an, Algorithmen herzulei-

ten, die diese Meshes mit möglichst geringer Zeitkomplexität generieren. Dafür wird zunächst die Möglichkeit untersucht, dieses Ziel in konstanter Zeit durch algebraische Berechnungen in konstanter oder durch einen Greedy-Algorithmus in linearer Zeit zu erreichen. Dabei wird aufgezeigt, dass diese Versuche fehlschlagen müssen.

Daher widmet sich als zentraler Teil dieser Arbeit Kapitel 4 neben der schrittweisen Suche dem Ansatz, die Vorgaben für Isopotentialflächen durch eine Simulation in polynomialer Zeit zu erfüllen. Diese basiert darauf, ein Mesh oder mehrere Meshes durch Zusammenfügen oder durch selektives Ein- und Ausblenden zu manipulieren, um die Kongruenz zu den Anforderungen aus Kapitel 3 herzustellen oder zu gewährleisten. Diese Operationen werden hinsichtlich der Nutzbarkeit für Algorithmen innerhalb der Simulation beleuchtet. Für die schrittweise Suche, die Annäherung durch Simulation und den Marching Cubes-Algorithmus werden Stärken und Schwächen in Abhängigkeit von Laufzeit- und Speicherkomplexität, Problemklassen und Kompatibilität zu vorgestellten Manipulationsoperationen aufgezeigt und gegenübergestellt. Um auf eine weitere Erhöhung der Geschwindigkeit der entwickelten Verfahren hinzuwirken, bespricht Kapitel 5 die Möglichkeit, diese zu parallelisieren.

Kapitel 6 nutzt den in Kapitel 4 vorgestellten Algorithmus der schrittweisen Suche, um ihn für das Anwendungsgebiet der Videospielentwicklung nutzbar zu machen. Dazu wird das Kriterium nach der Anforderung eines festen Isowertes aufgehoben und seine Geometrie extrudiert. Ziel dieser Untersuchung ist, ein Mittel bereitzustellen, das eine kontrollierte Generierung von Spielgeometrie ermöglicht. Um zu zeigen, dass die schrittweise Suche sich für diese Aufgabe besser eignet, wird diese mit dem etablierten Marching Squares-Algorithmus kontrastiert. Anschließend werden Überlegungen angestellt, elektrische Felder und aus ihnen gewonnene Spielgeometrie in weiteren Bereichen der Spieleentwicklung zu nutzen.

Zur Validierung der Ergebnisse werden in Kapitel 7 die Implementierungen der in dieser Arbeit neu vorgestellten Algorithmen diskutiert. Es stellt die für die unterschiedlichen Projekte verwendeten Technologien vor, hält die verwendeten Datenstrukturen in UML-Klassendiagrammen fest und erläutert wichtige Attribute und Methoden der besprochenen Datenstrukturen.

Um ein abschließendes Fazit der in dieser Arbeit durchgeführten Untersuchungen zu ziehen, werden in Kapitel 9 die Verfahren sowie deren Stärken und Schwächen der Verfahren zusammengefasst und deren Besonderheiten hervorgehoben.

Publikationen und betreute Arbeiten

[Rot+18]

Robert Roth, Tomasz Grzejszczak, Michał Niezabitowski und Reinhard Möller
Adapting Strategies to Display Simulations of Electric Fields Spawned by Point Charges for Augmented Reality
2018 IEEE International Conference on Consumer Electronics (ICCE)

[RMP20]

Robert Roth, Reinhard Möller und Pursche, Thomas
Extensible Augmented Reality Assisted Contact-Free Patient Surveillance in Emergency Context
2018 IEEE International Conference on Consumer Electronics (ICCE)

[RB22]

Robert Roth und Bernard Beitz
Macro-Controlled Generation of Geometry Using Vector Fields and their Application
Eurosis GAME-ON'22
Best Paper Award
Vorgemerkt zur Veröffentlichung in *New Trends in Computer Sciences* der VILNIUS TECH, `https://journals.vilniustech.lt/index.php/NTCS`

[GMR20]

Tomasz Grzejszczak, Reinhard Möller und Robert Roth
Tracking of dynamic gesture fingertips position in video sequence
Archives of Control Sciences, Vol 30, 2020

[Lep+16]

Thomas D. Lepich, Robert Roth, Reinhard Möller und Christoph Brandau
Semi-automated sanitizing of component dependencies after subsystem reconfiguration in multimodal VR and AR frameworks
2016 IEEE 6th International Conference on Consumer Electronics

KAPITEL 1

Ausgangspunkt der Forschung und Stand der Technik

Augmented Reality und die Simulation und Darstellung elektrischer Felder sind zwar für sich gesehen keine neuen Forschungsgebiete, sie bergen aber besondere Probleme, wenn sie mit dem Ziel kombiniert werden, elektrische Felder auf einem Augmented-Reality-fähigen Headset darzustellen. Dieses Kapitel stellt den Stand der Forschung zu der Kombination dieser beiden Themengebiete vor. Wir zeigen zunächst auf, dass elektrische Felder verschiedenster Art und Isoflächen zwar häufig Gegenstand der Forschung sind, allerdings selten zusammen. Wir führen die derzeitig gängigen visuellen Darstellungsmethoden für elektrische Felder auf und erarbeiten das Verständnis für Probleme, die sich mit ihnen im Zusammenhang mit der Anwendung in der virtuellen und erweiterten Realität ergeben. Außerdem beleuchten wir die derzeitigen Einsatzgebiete von Isoflächen abseits von elektrischen Feldern und ermitteln, inwieweit wir uns die bereits gewonnenen Erkenntnisse nutzbar machen können. Im weiteren Verlauf untersuchen wir bereits vorliegende Software-basierte Lösungen, die auf Isoflächen rekurrieren, um elektrische Felder darzustellen und ihre Anschlussfähigkeit an die virtuelle Realität. Schließlich fassen wir den Forschungsstand zu Isoflächen zusammen, die zeitlich veränderliche elektrische Felder repräsentieren.

1.1 Probleme beim Stand der Technik

Die Hardware zur Darstellung virtueller Realität in Form von IBM-PCs mit dedizierter Graphikhardware verliert bei Abschluss der Arbeit zusehends an Bedeutung und gibt Marktanteile an eigenständige VR-Umgebungen wie Oculus Quest, HTC Vive Focus und Pico Neo 2 ab. Die Niedrigpreissegmente halten bereits Smartphones mit Funktionalität für die Darstellung sowohl virtueller als auch erweiterter Realität vor. Durch Einsetzen in günstige Headsets lassen diese sich zu einem voll funktionsfähigen Standalone-VR-Headset kombinieren. Die Vorteile dieser Art von Geräten manifestieren sich in hoher Mobilität und einfacherer Handhabung. Der Preis für diesen Komfort ist allerdings die mobile Graphikhardware, die dem Rendering der Simulationsszenen zugedacht ist. Diese ist nicht für das hochperformante Darstellen komplexer

Geometrie konzipiert, sondern auf kleine Baugröße und niedrigen Energieverbrauch ausgelegt. Zudem verfügen mobile Graphikprozessoren in der Regel über einen für diese Aufgaben optimierten, geringeren Befehlssatz wie OpenGL ES oder Direct3D Mobile.

1.1.1 Darstellung elektrischer Felder

Die Darstellung elektrischer Felder erfolgt sowohl auf Bildschirmen als auch in der virtuellen Realität häufig durch die Linien- und Pfeildarstellung. Diese Darstellungsformen haben in der Didaktik ihren festen Platz, bergen aber eigentümliche Probleme bei der Interpretation:

1.1.1.1 Feldliniendarstellung

Die physikalische Grundlage zu der Feldliniendarstellung bildet das Metallspanexperiment: Um das magnetische Feld eines Stab- oder Elektromagneten zu untersuchen, wird dieser durch eine flache, nicht-magnetische Schicht abgetrennt, auf die gleichmäßig Metallspäne verteilt werden. Die ferromagnetische Eigenschaft der Späne führt zu ihrer Ausrichtung an dem elektrischen Feld, das auf sie wirkt [Bab16, S. 34]. [Cav66] stellt ein Experiment vor, das darauf abzielt, durch einen Staub aus Magnesiumoxid-Partikeln unter einer Glasglocke das elektrische Feld zweier Aluminiumelektroden dreidimensional sichtbar zu machen. Resultate dieser Experimente sind in Abbildung 1.1 dargestellt. Auf ähnliche Art und Weise werden auch für die Darstellung von Laminar- und Wirbelströmen in Flüssigkeiten Visualisierungen durch Stromlinien verwendet [Ada19, S. 112].

Aufgrund der Magnetisierung neigen die Metallspäne im elektrischen Feld zum Verklumpen. Dadurch stellen sich drei Effekte ein, die in Abbildung 1.1a zu beobachten sind.

1. Die Strukturen der Verklumpung erinnern optisch an Linien: Nachträglich in das Feld zwischen zwei Linien eingebrachte Späne werden sich unter Einfluss des Feldes auf eine der beiden Linien zubewegen. Durch die Lücken tritt die Linienstruktur hervor.

2. Die Metallspäne wirken aufgrund der Magnetisierung wie Magnete, die sich gegenseitig anziehen. In Bereichen hoher Feldstärken zeigt sich diese Wirkung entsprechend stärker. Daher ist in einem Bereich um die Quelle des magnetischen Feldes herum mit einer größeren Lückenbildung zu rechnen.

3. Bei großem Abstand zur Quelle des magnetischen Feldes wirken auch die magnetisierten Metallspäne schwächer. Bei zunehmendem Abstand behalten die Metallspäne in zunehmendem Maß ihre Position nach der gleichmäßigen Verteilung bei, so dass Aussagen über die Stärke des Feldes nicht mehr möglich sind und nur noch auf die Richtung geschlossen werden kann. Die Fähigkeit des Feldes, länglich geformte Metallspäne in Richtung des

Feldes zu rotieren, nimmt ab, bis Position und Ausrichtung eines einzelnen Spans keine Informationen mehr über das Feld liefert.

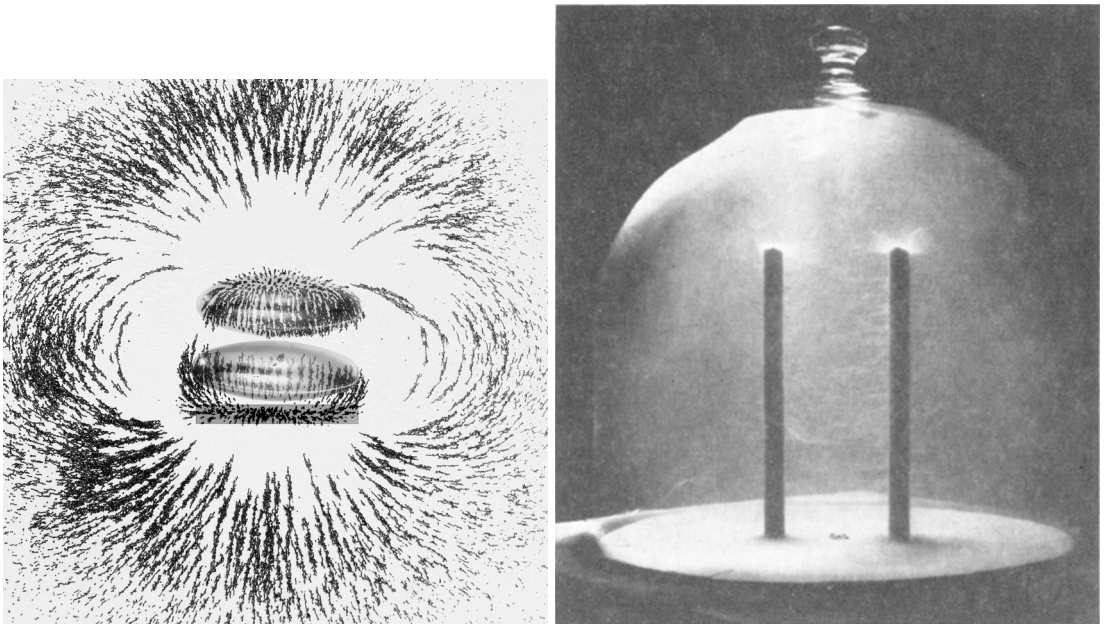
4. Die mechanischen Eigenschaften von Feldquelle und Spänen verhindern eine Darstellung des Feldes innerhalb der Quelle.

Aus den visuellen Phänomenen, die Experimente mit Metallspänen zeitigen, motiviert sich die Darstellung durch kontinuierliche Linien, die *Feldlinien* heißen. Die daraus folgenden Abbildungen fußen zusätzlich auf der Einsicht, dass elektrische Felder Kontinuen sind. Auf diese Weise ist eine Darstellung des Feldes durch unterbrechungsfreie Linien auch im Inneren des Körpers bzw. der Quelle möglich. Da das Feld ein Kontinuum ist, gibt es auch unendlich viele Feldlinien, die auf diese Weise dargestellt eine Interpretation verunmöglichen. Daher wird in der Regel eine Auswahl an Feldlinien dargestellt, deren geeignete Anzahl von einem Designer durch Mutmaßen festgelegt wird. Unabhängig von dieser Wahl sinkt der Abstand der Linien zueinander proportional mit der Stärke des elektrischen Feldes [KMR06, S. 149]. Auf diese Weise lässt sich der Unterschied des Skalarfeldes zwischen zwei Punkten, die auf oder nah an den Feldlinien liegen, intuitiv abschätzen. Ebenso verläuft das elektrische Feld vektoriell tangential zu den Feldlinien, wodurch sich in der Nähe der Feldlinien die Richtung des elektrischen Feldes abwägen lässt. In größerer Entfernung von den Feldlinien geht die Information über das elektrische Feld bei konstanter Anzahl der Feldlinien verloren, bei zunehmender Anzahl der Feldlinien entsprechend in der Nähe der Feldquelle. Ein Beispiel für die Darstellung eines Feldes durch Feldlinien, das durch zwei Punktladungen erzeugt wird, findet sich in Abbildung 1.2b.

[Mat+13] legen eine Implementierung der Darstellung von magnetischen Feldlinien in drei Dimensionen vor. Anhand von Abbildung 1.3 lässt sich ersehen, dass die durch Projektion verloren gegangenen Tiefeninformationen maßgeblich für die Interpretation des Simulationsergebnisses sind. In der Abbildung ist die Erkennbarkeit der Feldlinien durch weiträumige Szenenverdeckung erschwert und verlässt an mehreren Stellen die Sichtpyramide. Für die in der Abbildung durch zwei bis drei optische Tracker repräsentierten Ladungen bzw. leitenden Gegenstände scheint eine Anzahl von vier Feldlinien angemessen. Innerhalb einer Simulation kann die Situation eine höhere oder niedrigere Anzahl erfordern.

1.1.1.2 Pfeildarstellung

Die Pfeildarstellung elektrischer Felder macht sich die Eigenschaft der Kontinuität elektrischer Felder zunutze. Durch sie ist es möglich, jeden Punkt eines Vektorfeldes durch einen Pfeil darzustellen, dessen Richtung bzw. Länge die Richtung bzw. Stärke des elektrischen Feldes darstellt. Alleine aus der Ladungsverteilung und dem Isowert sind keine objektiv bedeutenden Punkte zu ermitteln, die sich als Position zum Platzieren von Pfeilen besser als andere eignen. Daher wird das Feld üblicherweise einer Rasterung unterzogen und jeder Rasterpunkt mit einem



(a) Zweidimensionale elektrische Feldlinien durch Metallspäne [PDS16]

(b) Dreidimensionale elektrische Feldlinien durch Magnesiumoxid-Staubpartikel [Cav66]

Abbildung 1.1: Experimente durch Metallpartikel zur Darstellung des elektrischen Felds durch Feldlinien

Pfeil versehen. Diese Darstellungsweise ist geeignet, bei der Visualisierung Bereiche höherer Feldstärke hervorzuheben, weil durch sie bedingt an diesen Stellen die Länge des Pfeils größer ausfallen, hinter denen kürzere Pfeile in Bereichen mit schwächerem Feld visuell zurücktreten. Allerdings teilt sich dieses Verfahren einige Probleme mit dem Marching-Cubes-Algorithmus: Eine Abschätzung des Darstellungsraums ist ohne zusätzliche Information nicht möglich und wird auch ohnedies erwartungsgemäß einen großen Bereich irrelevanter Informationen abbilden: Bereiche fernab der Ladung, an denen das Feld naturgemäß klein ausfallen wird, werden durch Pfeile dargestellt, deren Linie annähernd hinter die Spitze zurücktritt, wodurch die Information verloren geht. Andererseits werden Pfeile mit Ursprung in der Nähe der Ladung bedingt durch die lange Linie entsprechend viel Raum einnehmen, ohne dass dadurch zusätzliche Information vermittelt würde. Die Berechnung der Pfeile muss durch Iteration über den gesamten Darstellungsraum erfolgen. Im schlimmsten Fall entspricht dieser der gesamten Sichtpyramide. Diese setzt allerdings die geeignete Wahl einer *Far Clipping Plane* voraus. Diese Wahl muss aber unter Anderem in Abhängigkeit der Komplexität der darzustellenden Geometrie und der berechnenden Hardware getroffen werden, was erst nach Abschluss zumindest eines Teils dieser Berechnungen möglich ist. Außerdem ist eine angemessene Skalierung der Pfeilspitzen nicht immer möglich: Soll verhindert werden, dass einige Pfeilspitzen nicht im Darstellungsbereich liegen, müssen die Pfeile gegebenenfalls unter die Erkennbarkeitsgrenze skaliert werden. Überschreiten die Pfeile die Länge der Rastergröße, können Überschneidungen

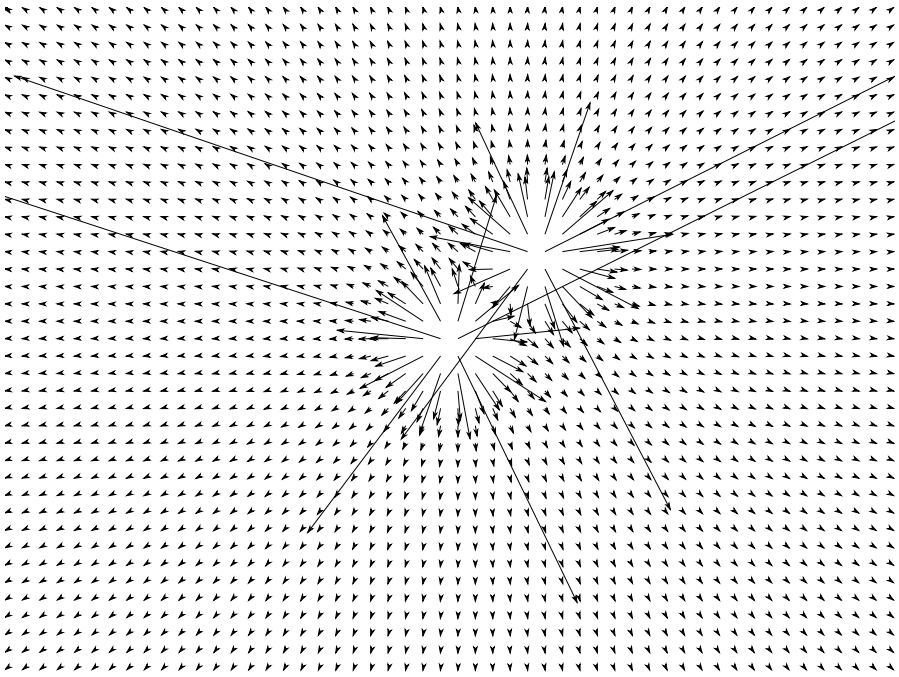
auftreten, die die Übersichtlichkeit der Abbildung herabsetzen können. Ein Beispiel für die Darstellung eines Feldes durch Feldlinien, das durch zwei Punktladungen erzeugt wird, findet sich in Abbildung 1.2b.

[Wie18] präsentiert eine Umsetzung der Darstellung elektrischer Felder in drei Dimensionen. Das Problem der Länge der Pfeile wurde durch Beschränken der Länge auf Kosten des Informationsgehalts umgesetzt [Wie18, S. 19]. Die Anzahl der Überschneidungen von Pfeilen und der durch den Bildschirmrand abgeschnittenen Pfeilen kann so reduziert werden. Die Interpretation der Darstellung kann allerdings auch hier nur unter Zuhilfenahme stereoskopischer Tiefeninformationen oder durch Positionswechsel der Kamera innerhalb der Simulation vollständig vorgenommen werden.

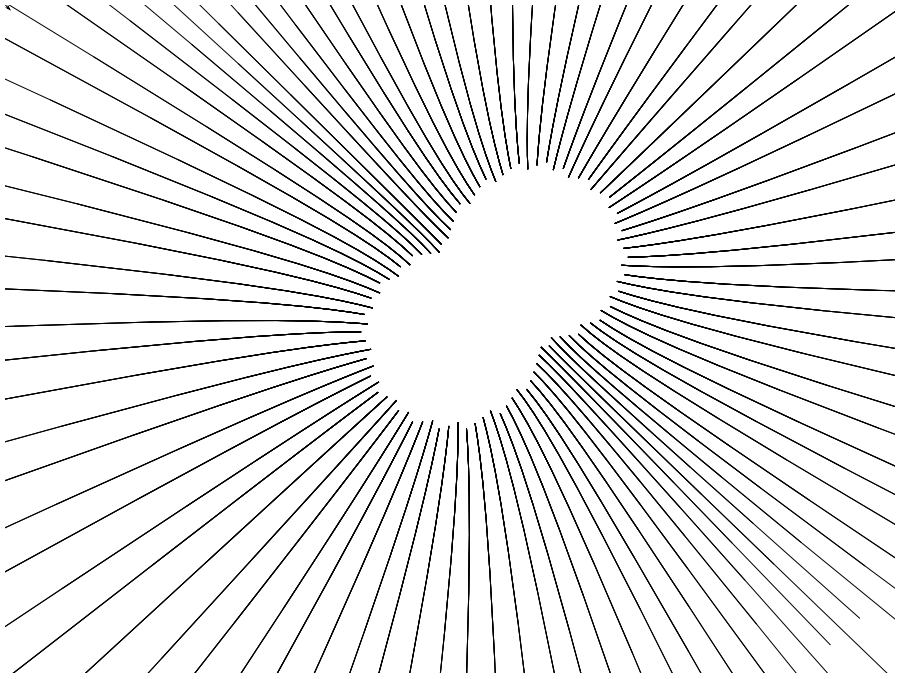
Der bei Abschluss dieser Arbeit einzige bekannte erfolgreiche Versuch, ein Skalarfeld – in diesem Fall ein magnetisches Feld – durch Virtual Reality darzustellen, wurde von [Mat+13] durchgeführt. Beispiele, die aus dem Einsatz entstanden, sind in Abb. 1.3 dargestellt. Das implementierte System wird als didaktisches Medium eingesetzt. Einfache magnetische Ladungsverteilungen können virtuell mittels Papiermarkern beliebig im einsehbaren Raum platziert werden. Die Arbeit gibt zwar keinen Aufschluss auf die Genauigkeit des Tracking-Verfahrens und der Darstellung. Allerdings sind auch die als Ergebnis dargestellten Geometrien, die das berechnete elektromagnetische Feld repräsentieren, nicht für den Gebrauch im wissenschaftlichen Bereich vorgesehen. Der Ansatz besteht darin, Feldlinien von statischen Feldern darzustellen. Für die Sichtbarkeit wurde eine Dicke der Feldlinie definiert sowie eine Pfeilspitze eingefügt, um die Orientierung des Feldes zu visualisieren. Die unendliche Ausdehnung und hohe Komplexität der darzustellenden Szene, die aus der hohen Anzahl der Feldlinien resultieren würde, wird durch Abschneiden und Reduzieren der Feldlinien realisiert.

1.1.2 Darstellung als Isoflächen

Auch die Darstellung von Isoflächen wird im Forschungsbereich der Computergraphik seit langem vertieft [BW01, passim] [DFMD02]. Die mitunter durchaus zeitaufwändige Berechnung ihrer Ausdehnungen, die nicht trivial ist, wird die Durchführung des Marching-Cube-Algorithmus benötigt. Aus diesem Grund sind Vereinfachungen vorgenommen worden: [Bli82] vereinfacht die Gleichung der elektrischen Flussdichte, indem er Teile der Gleichung entfernt, deren Berechnung zeitaufwendig sind. Durch diesen Ansatz erreicht Blinn die Visualisierung ansehnlicher Isoflächen bei höherer Bildwiederholrate, disqualifiziert die Ergebnisse aber gleichzeitig für den Einsatz im wissenschaftlichen Bereich, da der Gewinn an Geschwindigkeit zu Lasten der Genauigkeit geht. Darüber hinaus wird der Ansatz vom Autor selbst als „nicht besonders schnell“ („not terrifically fast“ [Bli82, S. 249]) beschrieben, was möglicherweise darauf zurückzuführen ist, dass der Algorithmus auf Raytracing basiert. Augmented-Reality-Anwendungen hingegen erfordern eine hohe Bildwiederholrate, um das Risiko einer Simulatorkrankheit in



(a) Pfeildarstellung



(b) Feldliniendarstellung

Abbildung 1.2: Ungeeignete Darstellungsmethoden in 2D

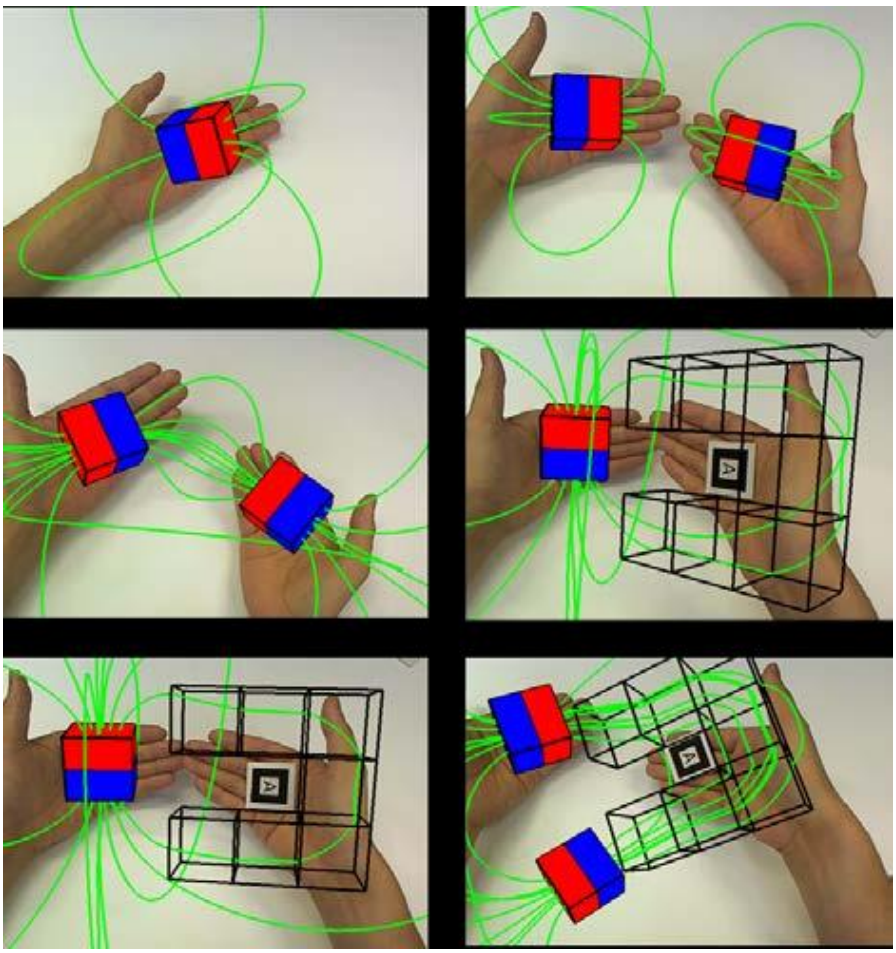


Abbildung 1.3: Durch Feldlinien dargestelltes Magnetfeld, das durch Ladungen an Positionen der Marker erzeugt und durch Dielektrika verformt wird [Mat+13]

Verbindung mit Unwohlsein zu verringern. Der Marching-Cube-Algorithmus baut auf der Ausgabe einer gerasterten Isofläche auf, die Teile zwischen den Rasterebenen approximiert. Bei präzisen Darstellungen kann der Einsatz eines sehr feinen Rasters erforderlich sein, was zu zunehmenden Iterationen im Raum führt, der keinen Teil der tatsächlichen Oberfläche enthält. Zusätzlich ist für jedes Bild eine Neuberechnung der gesamten Szene erforderlich; Der Marching Cubes-Algorithmus greift nicht auf alte Berechnungen zurück. Ein durch den Marching-Cube-Algorithmus erstelltes Mesh ist in Abbildung 1.4 dargestellt.

1.1.3 Raytracing

[Bli82] legte das erste Verfahren zur Darstellung von Isoflächen vor, das zu seiner Zeit auf einem Raymarching-Verfahren basierte. Dort wird zunächst anhand einer primitiven Isofläche für eine Punktladung analog zu Gleichung (3.3) ein Suchintervall für den Schnittpunkt des Strahls mit der eigentlichen Isofläche durch den *regula falsi*-Algorithmus ermittelt. Darauf aufbauend wurden einige Verbesserungen vorgeschlagen:

- [Bel21, S. 28f.] findet Argumente, die Suche nach der Äquipotentialfläche durch Einführen eines Suchraums in Form eines achsenparallelen Quaders *Bounding Box* für Punktladungen zu vereinfachen. Einschränkungen, wie

  > „Damit solche Fälle [mehrerer Schnittpunkte mit den Äquipotentialflächen] keine Probleme verursachen, werden mehrere Punkte vor dem Schätzpunkt gewählt [Bel21, S. 29, 57]."

  werden nicht näher begründet und die Anzahl der heuristischen Punkte werden nicht angegeben [Bel21, S. 29, Zz. 17ff], so dass dieses Verfahren auf makroskopische Sonderfälle beschränkt bleibt, in denen angenommen werden kann, dass der Schätzpunkt nicht innerhalb einer anderen Fläche liegt.

- Für extrem große Datensätze präsentieren [Par+98] ein Raytracing-Verfahren, das das Rendern im Bereich des Medical Imaging so weit beschleunigt, dass eine interaktive Auswahl des Isowertes möglich ist. Die Autoren geben eine Abschätzung für die Zeitkomplexität im Unterschied zu [VG97] mit $\mathcal{O}(n \cdot m_x \cdot m_y)$ an, und können schließlich Optimierungen durch Anwenden eines Oktonärbaumalgorithmus anbringen. Dadurch reduzieren sie den durchschnittlichen Zeitaufwand auf $\sqrt[3]{n}$ Rechenoperationen, ohne sich mit einer Begründung etwa durch eine probabilistische Analyse [CLR09, S. 115] aufzuhalten. Im schlimmsten Fall bedeutet dieses Ergebnis eine Zeitkomplexität von $\mathcal{O}(n^3)$.

Diese Verbesserungsvorschläge konnten das Problem der Unentscheidbarkeit der geometrischen Optik [RTY94, passim], dem die hohe Laufzeitkomplexität von Raytracing-Algorithmen

geschuldet ist, allerdings nicht entscheidend verringern [VG97]. Im schlimmsten Fall belaufen sich ihre Laufzeiten auf

$$\mathcal{O}\left((l+1)(n+k)\log(n) + (l+1)\cdot m_x \cdot m_y \cdot \log(n)\right)$$

wobei l die Anzahl der Lichtquellen, n die Anzahl der Vertices, $k \in \mathcal{O}(n^2)$ ein szenenabhängiger Parameter und m_x und m_y die Anzahl der Pixel in x bzw. y-Richtung bezeichnet. [SLM88, S. 16]

1.1.4 Volumengraphik

Die Volumengraphik (Volume Rendering) hat ihren Ursprung in der wissenschaftlichen Darstellung von Sensordaten und befasst sich mit verschiedenen Methoden zur Erzeugung von Bildern zur Visualisierung dreidimensionaler skalarer Daten [Eng+06b, S. 2ff]. In der Volumengraphik wird dreidimensionale Geometrie durch Voxelgitter dargestellt. Sie findet hauptsächlich im Bereich des Medical Imaging, der Werkstoffkunde, Computational Chemistry, Quantenphysik und der Atmosphärenoptik Anwendung [LG95, S. 17 und passim]. Auch für das Darstellen von Isopotentialflächen wurden Verfahren der Volumengraphik angewendet, wobei das größte Einsatzgebiet das Medical Imaging darstellt [Had+05]. In diesem Bereich stellt die Volumengraphik Methoden zur Beschleunigung bestehender Verfahren zur Darstellung von Isoflächen wie dem Marching Cubes-Algorithmus bereit.

1.1.5 Marching Cubes

Im Bereich des Medical Imaging kann der Suchbereich sehr genau abgeschätzt werden, weil diese Verfahren das Vorliegen der nötigen Datenpunkte bereits voraussetzen. Eine Beschleunigung des Verfahrens ist in diesen Anwendungsfällen durch Vorsortieren der Daten zu erreichen, um so Iterationen über Leerraum zu reduzieren [Cig+97; IK95]. Durch die Aufgabenstellung dieser Arbeit können wir aber an diese Erkenntnisse nicht anlehnen, weil im vorliegenden Fall abgesehen von der Ladungsverteilung keine weiteren Daten gegeben sind, sondern diese bestimmt werden müssen.

Eine Abschätzung der Komplexität aus dem Marching Cubes-Verfahren resultierender Meshes ist nicht möglich; Tatsächlich können ungünstige Konfigurationen von Punktladungen und dem Suchraum zu einer Anzahl an Polygonen führen, die auf mobiler Hardware nicht mehr in akzeptabler Zeit gerendert werden können. Verfahren wie das in [NH92] können zwar die Anzahl der Polygone um den Faktor 10 senken, nehmen dazu jedoch einen zusätzlichen Aufwand von $\mathcal{O}(n^2 \log n)$ in Kauf. Das Ziel einer genauen Darstellung des durch diese Punktladungen erzeugten Feldes schließt dieses Vorgehen aber aus.

1.1.6 Zeitveränderliche Isoflächen

Das Thema zeitveränderlicher Isoflächen ist weitgehend unerforscht. [BW01, S. 10] vermelden hierzu zwar steigendes Interesse (*increasing interest*), können aber zum damaligen Forschungsstand nur drei einschlägige Veröffentlichungen anführen: [WB98] drücken zeitlich veränderliche Datenreihen repräsentierende Isoflächen durch eine vierdimensionale Funktion aus und triangulieren diese durch Simplizes. Dabei gehen die Autoren davon aus, dass das Skalarfeld während des gesamten Zeitraums als gegeben vorliegt. [She98] und [SH99] legen Vorschläge zur Optimierung der Laufzeit- und Speicherkomplexität durch ein vorgelagertes Sortierverfahren in Binärbäumen bzw. Bäume durch Oktonärbäume vor. Die beim Abschluss dieser Arbeit einzigen neueren Veröffentlichungen zu diesem Thema operieren unter den gleichen Voraussetzungen und berechnen für jeden Zeitabschnitt die baumartigen Speicherstrukturen einzeln, um aus dieser einen Topologie-Übergangsgraphen (*topology change graph*) abzuleiten, der die zeitlichen Übergang der Konturen in *Events* festhält. Dadurch ermöglichen [SB06] das interaktive Verfolgen und Extrahieren von Konturstrukturen innerhalb des zeitlich variablen Skalarfelds, indem alle Änderungen der Konturen im Voraus berechnet werden – einen Vorgang, den die Autoren ohne diese Maßnahme als ungeeignet für interaktive Applikationen („not suitable for interactive applications“), da zeitaufwändig (*expensive*) bezeichnen. [Gre+04] spezialisiert die vorgenannten Verfahren auf Daten, die nach einem nicht näher ausgeführten Verfahren komprimiert sind und parallelisiert sie auf dedizierter Hardware.

Die oben vorgestellten Verfahren zur Darstellung zeitveränderlicher Isoflächen setzen voraus, dass der gesamte Verlauf der zeitlichen Veränderung vor ihrer Darstellung zu einem gegebenen Zeitpunkt bekannt ist. Obwohl [She98; SH99; WB98] Anwendungsbeispiele vorlegen, in denen diese Annahme tragfähig ist, behält sich diese Arbeit die Möglichkeit einer Interaktion vor. Diese bedingt unvorhersehbare Veränderungen an den Eingabedaten – hier der Ladungsverteilung, der Ladungsstärke und des Isowerts – weshalb die oben angeführten Verfahren nicht mehr zur Anwendung kommen können.

1.1.7 Software-Anwendungen

Eine Marktanalyse für den Bereich akademischer und gestalterischer Software-Programme fördert eine breite Palette von Anwendungen aus verschiedenen Epochen und verschiedenen Einsatzgebieten zur Analyse und Anzeige elektromagnetischer Felder auf Bildschirmen zu Tage. Eine Auswahl dieser Projekte wird im Folgenden anhand einiger Repräsentanten auf die Tauglichkeit hin, wissenschaftlich verwertbare Erkenntnisse aus ihnen abzuleiten, untersucht. Bei den ersteren, proprietären Projekten ist eine hohe Entwicklungsreife der Programmierung und hohe Anwendbarkeit zu erwarten, weswegen ihren technischen Details ein hohes Augenmerk der Untersuchung zugedacht ist. Die Liste endet mit Software, die durch einen zumindest teilweise quelloffenen Teil des Projekts eine Analyse der verwendeten Algorithmen erlaubt.

Electrostatics3D Bei *Electrostatics3D* handelte es sich um eine Applet-Anwendung zur zweidimensionalen und dreidimensionalen Darstellung elektromagnetischer Felder zu didaktischen Zwecken. Die Entwicklung wurde eingestellt, und der Download ist nicht mehr verfügbar. Aus den archivierten Webseiten lässt sich noch ersehen, dass das Programm über die Funktionalität zum Zeichnen von Feldlinien und *Isoflächen* hatte, aber über die verwendeten Technologien kann nur spekuliert werden. Darüber hinaus wurde für diese Anwendung keine Anbindung an Virtual- oder Augmented Reality vorgesehen [Sha02].

Charges and Fields *Charges and Fields* ist eine browserbasierte Anwendung für den gleichen Zweck. Eine Analyse des Quellcodes ergibt, dass die Berechnung und Darstellung der dreidimensionalen Geometrie über WebGL erfolgt, was eine Anbindung an die virtuelle Realität prinzipiell ermöglicht. Charges and Fields sieht lediglich eine Abwandlung der Pfeildarstellung und eine zweifarbige Höhenkarte des Feldes vor. [DR04].

3-D Electrostatic/Magnetostatic Fields Applet Als Teil einer Sammlung diverser Java-Applets zur Simulation mathematischer und physikalischer Phänomene dienen 3-D Electrostatic Applet und Magnetostatic Fields Applet der Darstellung elektrostatischer und magnetostatischer Felder in 2D und 3D. Neben einigen nicht für die virtuelle Realität geeigneten Anzeigemethoden können verschiedene unter anderem bewegte Darstellungen mit Pfeilen, Feldlinien und Testladungspartikeln ausgewählt werden. Mit der Einstellung „*Equipotentials*“ kann eine Ebene in zwei Koordinaten („*Slice*“) oder eine Darstellung ohne Slices gewählt werden. Die letztere Option legt nahe, dass das Programm imstande ist, ein Drahtgittermodell der Ladungsanordnung zu erstellen. Der Quellcode kann auch hier leider keiner Analyse unterzogen werden. Die Ladungsverteilungen selbst sind allerdings nicht frei wählbar, sondern können lediglich aus einer Liste ausgewählt werden. Dieser Umstand gibt Grund zu der Annahme, dass auf die wählbaren Sonderfälle spezialisierte Algorithmen zum Einsatz kommen. Bei der Umsetzung dieses Projekts stand die Portabilität für Augmented oder Virtual Reality darüber hinaus offensichtlich nicht im Vordergrund [Fal21].

COMSOL Multiphysics Bei *COMSOL Multiphysics* handelt es sich um eine CAE-Software-Lösung zur Durchführung von Finite-Elemente-Analysen in verschiedenen physikalischen Anwendungen [May19, S. 90]. Die Entwicklerfirma COMSOLAB veröffentlicht keine Informationen über die zugrundeliegenden Algorithmen, allerdings ist es der Methodik zu eigen, dass sie auf der Unterteilung des Problemraums in kleinere, häufig quadrilaterale Geometrien („Elemente“) [Tab15, S. 19] [MKH13, passim] basiert, um diese dann als homogene Einheiten den entsprechenden physikalischen Modellen – im Falle elektromagnetischer Analyse den Maxwell-Gleichungen – zuzuführen [MKH13, S. 373]. Darüber hinaus lassen die Einstellungsmöglichkeiten der Benutzeroberfläche darauf schließen, dass eine Darstellung von Äquipotentialflächen nur über einen be-

grenzten, quaderförmigen Raum möglich ist. Das darzustellende Mesh entsteht also wahrscheinlich durch Iteration über diesen Raum. In der Fachliteratur zu Themengebieten der Finite-Elementen-Analysen findet sich unter anderem ausdrückliche Verweise auf den Marching-Cubes-Algorithmus [Son18, S. 356] und ähnliche Verfahren [MKH13, S. 401]. [JNS17] tragen einen ersten, vielversprechenden Versuch bei, die Ergebnisse von durch COMSOL erstellte Finite-Elemente-Analysen in der virtuellen Realität darzustellen, können aber aufgrund des proprietären Charakters der Software über die Entstehung der Darstellung des Feldes ebenfalls keine Angaben machen. Die Analyse muss mit dem Befund abschließen, dass das Frontend die Daten binär empfängt und diese durch einen eigens programmierten Shader auf einem hinterlegten Mesh darstellt.

ANSYS Das kommerzielle Programm *ANSYS* verfügt mit ANSYS Maxwell über eine dedizierte Umgebung zur Analyse elektromagnetischer Felder durch Finite-Elemente-Methoden [May19, S. 89ff]. Der proprietäre Charakter dieses Projekts verhindert eine Analyse des Quellcodes ebenfalls. Da es sich bei ANSYS über eine Software-Suite zur Durchführung von Finite-Elemente-Analysen handelt [Moa99, passim], erscheint die Annahme sinnvoll, dass ANSYS sich auch bei der Darstellung von elektrischen Feldern durch Isoflächen auf ähnliche Vorgehensweisen zur Anwendung bringt, wie COMSOL. Mit der Simulationsplattform ANSYS VXPERIENCE steht eine Produktfamilie mit Schnittstellen zur virtuellen Realität zur Verfügung, die in der Hauptsache auf realistische Fahrsimulation abzielt [San20, S. 15], wobei auch Daten von verschiedene Sensoren, die im Bereich der Fahrassistenzsysteme Einsatz finden, Teil der Simulation sein können [Sys20]. Diese können im Sinne einer Anwendung von der Aufgabenstellung erfasst werden, allerdings fungiert die virtuelle Realität hier als reines Darstellungsmedium der Ausgaben von ANSYS und bedient sich somit der gleichen Rendering-Methoden wie auch bei der Bildschirmarbeit.

Verschiedene Funktionen in MATLAB/GNU Octave Die numerische Rechenumgebung *MATLAB* stellt in ihrer gleichnamigen Programmiersprache unter Anderem die `isosurface`-Funktion bereit [Cla18, S. 208ff]. Auch hier lässt sich anhand erstellter Darstellungen wie in Abbildung 1.4 nur mutmaßen, dass sich die MATLAB-Suite auf den Marching Cubes-Algorithmus stützt. Aus der Analyse des Quellcodeabschnitts in der quelloffenen Alternative *GNU Octave* resultiert, dass auch hier der Marching Cubes-Algorithmus zum Einsatz kommt.

Da sich proprietäre, undokumentierte Software hinsichtlich ihrer Methodik jeder Analyse entzieht, können diese trotz ihrer teilweise vielversprechenden Ansätze in den folgenden Überlegungen nicht berücksichtigt werden. Quelloffene Software und proprietäre Software, deren Verfahren dokumentiert sind setzen allerdings in der Regel auf die Anzeige von Feldlinien. Software, die Isoflächen anzeigt, bedient sich Varianten von Algorithmen, die abgesehen von

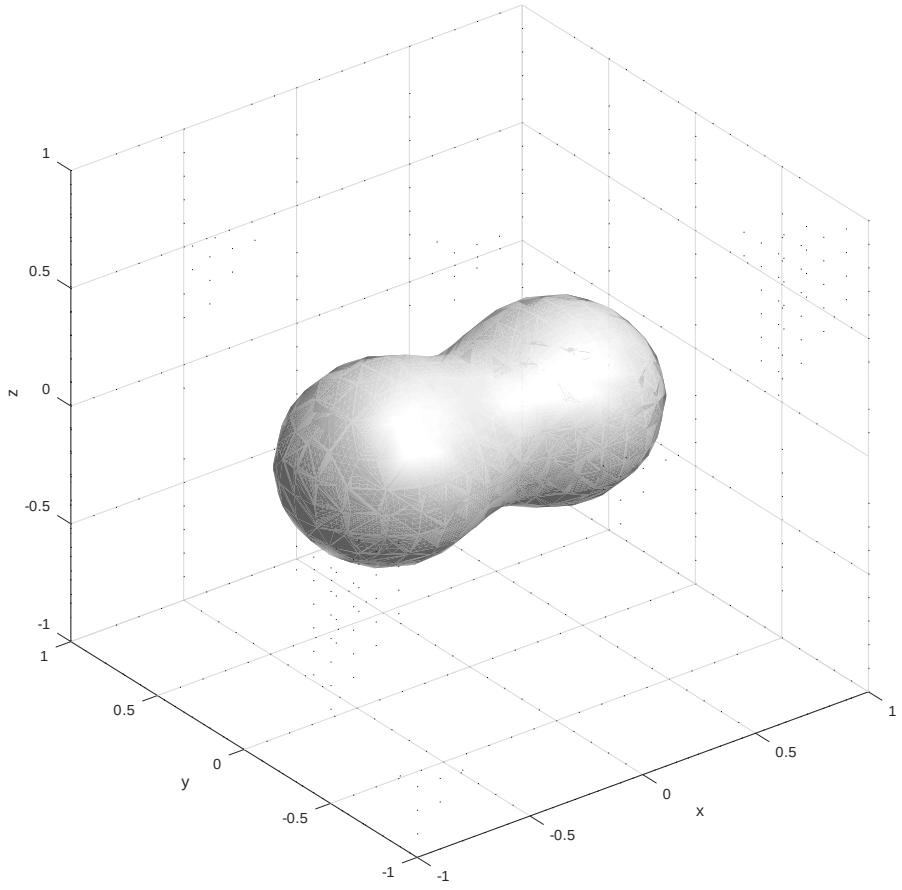


Abbildung 1.4: Darstellung einer Isosurface in GNU Octave

anderen Problemen über den gesamten darstellbaren Bereich – zum Großteil leeren Raum – iterieren müssen und das Ergebnis durch die Grenzen des Anzeigemediums beschnitten werden. Eine genauere Untersuchung dieser Verfahren findet in Kapitel 4 statt.

KAPITEL 2

Grundlagen

2.1 Definitionen

Ein *Vektor* in n Dimensionen wird mit $\mathbb{R}^n$ notiert. Es gelten die bekannten Axiome der Vektoralgebra. Diese Arbeit verwendet besonders das Skalarprodukt ($\mathbf{a} \bullet \mathbf{b}$, zur Unterscheidung von der Multiplikation zweier Zahlen und der Vektor-Matrix-Multiplikation: $\mathbf{c} \cdot \mathbf{A}$), das Vektorprodukt $\mathbf{a} \times \mathbf{b}$ so wie die komponentenweise Addition und Subtraktion mit der Schreibweise $\mathbf{a} + \mathbf{b}$ und $\mathbf{a} - \mathbf{b}$, wobei $\mathbf{a}$ und $\mathbf{b}$ Vektoren sind.
Die Komponenten eines Vektors aus $\mathbb{R}^n$ ergeben sich durch die Auswahlfunktion

$$(\cdot)_k : \mathbb{R}^n \times \mathbb{N} \to \mathbb{R}, \quad \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_k \\ \mathbf{v}_{k+1} \\ \vdots \\ \mathbf{v}_n \end{pmatrix}_k \mapsto \mathbf{v}_k$$

Als gängige Werte für k werden die Kurzschreibweisen

x für $k = 1$
y für $k = 2$
z für $k = 3$
w für $k = 4$

verwendet.

Beispiel

Für einen Vektor $\mathbf{v} = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$ ist $\mathbf{v}_1 = 2$, $\mathbf{v}_2 = 3$.

Halbgerade

Eine *Halbgerade* oder ein Strahl ist eine Gerade, die einer Seite zu begrenzt ist.

Ein Strahl lässt sich durch Einschränken des Definitionsbereichs einer Geraden auf einen höchsten oder niedrigsten Wert oder durch Angeben eines vektoriellen Aufpunkts $\mathbf{a}$ und eines Richtungsvektors $\mathbf{b}$ in der Form

$$\mathbf{s}(x) = \mathbf{a} + x\mathbf{b} \qquad x \in \mathbb{R}, x \leq 0 \tag{2.1}$$

formulieren. In Abbildungen kennzeichnen wir die unbegrenzte Seite der Halbgeraden durch eine doppelte Pfeilspitze ($\twoheadrightarrow$).

2.2 Mengen und Folgen

Für die wohlunterschiedenen Elemente $m_1, m_2, \ldots$ einer *Menge* M werden die Schreibweisen $m_1, m_2, \cdots \in M$ und $M = \{m_1, m_2, \ldots\}$ verwendet. Die Vereinigung $M \cup N$ und Schnittmenge $M \cap N$ mit den Mengen M und N folgen abgesehen von den folgenden Bemerkungen den Axiomen der Zermelo-Fränkel-Mengenlehre. Bei der Definition von Operatoren bezeichnen m bzw. f die Klasse der Mengen bzw. der Folgen.

Operationen auf Mengen

Aus der Vereinigung der Mengen $M \cup N = P$ folgt als $M = P \setminus N$ die *Differenz* der Menge M mit allen Elementen in P außer derer in N. Die *Kardinalität* einer endlichen Menge M wird mit dem Operator:

$$\# : m \to \mathbb{N}, \qquad \{m_1, m_2, \ldots m_k\} \mapsto k$$

bezeichnet.

Als *Folgen* bezeichnen wir durch die natürlichen Zahlen indizierte, geordnete Mengen mit den gleichen Schreibweisen und dem Auswahloperator

$$(\cdot)_k : F \to a, \qquad \{f_1, f_2, \ldots, f_k, f_{k+1}, \ldots, f_n\}_k \mapsto f_k$$

für alle Folgen in F, die das Element f_k beinhalten. Das Kriterium der Wohlunterscheidbarkeit entfällt bei Folgen. Für *periodische Folgen* f sind alle Elemente mit Indizes $i > \#f$ definiert durch $f_i := f_{i-\#f}$.

Operationen auf Folgen

Die Konkatenation zweier Folgen $F = \{f_1, f_2, \ldots\}$ und $G = \{g_1, g_2, \ldots\}$ ist

$$+ : f \times f \to f, \qquad F + G = \{f_1, f_2, \ldots, g_1, g_2, \ldots\}$$

Stellenweise erfolgt die Konkatenation mit einem einzelnen Element $e \in F$:

$$F + e = \{f_1, f_2, \ldots, e\}\,. \tag{2.2}$$

$$e + F = \{e, f_1, f_2, \ldots\}\,. \tag{2.3}$$

2.3 Gleichheit

Für verschiedene Aussagen müssen verschiedene Definitionen der Gleichheit und dem Gleichheitssymbol $=$ verwendet werden, um Missverständnisse zu vermeiden.

Aussagen Sind A und B Aussagen, bedeutet die Schreibweise $A \Rightarrow B$, dass die Aussage B aus der Aussage A folgt. Der Pfeil kann mit analogen Bedeutungen in umgekehrte oder beide Richtungen Verwendung finden.

Werte Sind A und B Variablen, folgt aus der Aussage $A = B$, dass beide Variablen den gleichen Wert besitzen. Diese Regel schließt Zeiger auf die gleiche Adresse als „zeigeridentisch“ ein.

Mengen und Folgen Sind A und B Mengen, folgt aus der Aussage $A = B$

1. $\#A = \#B$ (Gleichheit der Werte) und
2. Für alle Elemente $a \in A$ existiert ein Element $b \in B$ mit $a = b$ (Gleichheit der Werte) und umgekehrt

Für Folgen gilt statt des Punkts 2: Für alle $i \in \mathbb{N}$ mit $i \leq \#A$ gilt $A_i = B_i$.

Hoare-Tripel Wenn A das Hoare-Tripel $\{\!|\ A_1\ |\!\}\ \ S_1\ \ \{\!|\ A_2\ |\!\}$ und B das Hoare-Tripel $\{\!|\ B_1\ |\!\}\ \ S_2\ \ \{\!|\ B_2\ |\!\}$ mit den Aussagen A_1, A_2, B_1, B_2 und den Folgen S_1 und S_2 bezeichnen, folgt aus $A = B$:

- $A_1 = B_1$ (Gleichheit der Aussagen),
- $A_2 = B_2$ (Gleichheit der Aussagen),
- $S_1 = S_2$ (Gleichheit der Folgen)

Sei $M(x)$ eine Aussage über ein beliebiges x und die obigen Definitionen der Gleichheit nicht auf A und B anwendbar, bedeutet die Gleichung $A = B$: $M(A) \Leftrightarrow M(B)$.

Die in dieser Arbeit ebenfalls verwendeten Begriffe der Isomorphie und der Äquivalenz bezeichnen Gleichheitsbeziehungen unter schwächeren Bedingungen. Diese stützen sich an einigen Stellen auf Morphismen.

Morphismus, Homomorphismus, Funktion

Ein *Morphismus* $f : A \to B$ ist eine Abbildung zwischen den zwei Objekten A und B. Erhält f dabei die Struktur der Objekte, handelt es sich bei f um einen *Homomorphismus*. Sind A und B Mengen und bildet f jedes Element aus A eindeutig auf ein Element aus B ab, dann ist f eine *Funktion*.

Homomorphie konstituiert hier eine schwache Form der Gleichheit zwischen zwei Objekten. Auch Isomorphismen sind insbesondere Morphismen:

Isomorphie und Isomorphismus

Zwei Objekte A und B, die durch einen Morphismus $f : A \to B$ verknüpft sind, sind *isomorph*, wenn ein Morphismus $f^{-1} : B \to A$ existiert, so dass $f(f^{-1}(a)) = f^{-1}(f(a)) = a$. f und f^{-1} heißen dann *Isomorphismen* [Mil14], wobei f^{-1} das Inverse von f ist.

Beispiel

- Die Folgen von Zeichenketten: $\{\texttt{Katzen}, \texttt{sind}, \texttt{Monster}\}$ und $\{\texttt{Monster}, \texttt{sind}, \texttt{Katzen}\}$ sind isomorph mit dem Isomorphismus $f : \Sigma^3 \to \Sigma^3$, $\{a, b, c\} \mapsto \{c, b, a\}$. In diesem Fall ist f ein Automorphismus, da $f = f^{-1}$.
- Vektoren in kartesischen und homogenen Koordinaten sind isomorph mit den Isomorphismen $f : \mathbb{R}^4 \to \mathbb{R}^3$ und $f^{-1} : \mathbb{R}^3 \to \mathbb{R}^4$

$$f : \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \end{pmatrix} \qquad f^{-1} : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

  sofern $w \neq 0$.
- Die Mengen der Fließkommazahlen und der Zeichenketten sind nicht isomorph: Die Zeichenkette `Fuchs` kann nicht als Fließkommazahl dargestellt werden.

Äquivalenz und Äquivalenzklasse

Für alle Objekte einer Äquivalenzklasse muss gelten: [Mil14, S. 369]

- Reflexion: $a \sim a$ (Alle Objekte sind äquivalent zu sich selbst)
- Symmetrie: Wenn $a \sim b$, dann $b \sim a$.
- Transitivität: Wenn $a \sim b$ und $b \sim c$, dann $a \sim c$.

Gelten alle dieser Aussagen, heißt a *äquivalent* zu b. Eine *Äquivalenzklasse* enthält alle zueinander äquivalenten Elemente einer Oberklasse.

Beispiel

Zwei rationale Zahlen (a_1, a_2) und (b_1, b_2), wobei a_1, a_2, b_1 und b_2 ganze Zahlen sind, sind äquivalent, wenn gilt: $a_1 \cdot b_2 = a_2 \cdot b_1$.

Aus den obigen Definitionen von Gleichheit ergeben sich automatisch weitere intuitiv verständliche Gleichheiten wie die Gleichheit von Meshes über die Gleichheit der Mengen.

Monoid

Ein *Monoid* ist eine assoziative algebraische Struktur mit einer binären Operation und einem neutralen Element [CB14, S. 257]. Für *assoziative* Strukturen gilt $(a + b) + c = a + (b + c)$. Für ein *neutrales Element* e gilt für alle Elemente a: $a + e = e + a = a$. a, b, c und e bezeichnen hier Elemente der Struktur und $+$ ihre Verknüpfung.

2.3.1 Repräsentation von Objekten durch Begrenzungsflächen (B-Rep)

Mesh, Punkte, Face, Kante

Ein *Mesh* (auch: Drahtgittermodell, geometrisches Primitivum) ist eine dreidimensionale Repräsentation eines Polyeders. Zur Unterscheidung von Flächen verwenden wir für Meshes den Begriff *Face*. Sie begrenzen die Ausdehnung des modellierten Körpers. Die Kanten der Faces sind identisch mit den *Kanten* des Meshes. Diese Punkte, die die Kanten begrenzen, heißen auch Vertices. Ihre Definition ist identisch mit der des Vektors.

Anschaulich ist ein Face eine durch eine Menge Kanten $\{k_1, k_2, \ldots\}$ begrenzte Fläche mit einem Vertex einer der Kanten als Stützvektor. Da die Kombination zweier Monoiden ebenfalls einen Monoiden ergibt, ist ein Mesh, das als Kombination mehrerer Meshes entsteht, die als Folgen selbst Monoiden sind, auch ein Monoid. In dieser Arbeit liegt das Augenmerk auf Dreiecken, also Faces, die durch drei Kanten begrenzt werden und sich untereinander drei Vertices $\mathbf{a}$, $\mathbf{b}$ und $\mathbf{c}$ teilen. Die Umlaufrichtung dieser Punkte bestimmt die Flächennormale

eines Face, wobei eine Umlaufrichtung entgegen dem Uhrzeigersinn eine nach außen zeigende Flächennormale bedingen soll. Diese kann damit durch

$$\frac{(b-a)\times(c-a)}{\|(b-a)\times(c-a)\|}$$

berechnet werden. Da im Begrenzungsflächenmodell die Punkte, Kanten und Flächen im Sinne ihrer Definition weiter oben als Folgen mit den Namen „Punktfolge“, „Kantenfolge“ und „Flächenfolge“ abgefasst werden, sind diese Monoiden.

Für diese Arbeit werden an Meshes an manchen Stellen zusätzliche Forderungen gestellt:

Eigenschaften von Meshes

Sauberkeit Ein Mesh ist *sauber*, wenn für alle Vertices a und b und alle Kanten k
- die Kantenfolge entweder die Kante (a, b) oder die Kante (b, a) enthält und
- genau zwei Elemente der Flächenfolge k enthalten

Geschlossenheit Ein Mesh ist *geschlossen*, wenn es für jede Kante k genau zwei Elemente f, g in der Kantenmenge gibt, für die $k \in f$ und $k \in g$ gilt.

Regularität Ein Mesh ist *regulär*, wenn alle Elemente der Flächenfolge die Kardinalität n besitzen. Sonderfälle sind:

Triangularität Ein reguläres Mesh ist *triangular*, für $n = 3$

Quadrilateral Ein reguläres Mesh ist *quadrilateral*, für $n = 4$.

Selbstüberschneidungsfreiheit Ein geschlossenes, reguläres Mesh ist *selbstüberschneidungsfrei*, wenn keines seiner Vertices einen Polyeder-Beinhaltungstest für dieses Mesh besteht.

Die Ansprüche an Sauberkeit, Geschlossenheit und Triangularität erhalten die monoidale Struktur geometrischer Primitiva.

2.4 Elektrostatik

Das Ziel dieser Arbeit ist das Entwickeln von Methoden zur Darstellung elektrischer Felder mit Mitteln der Augmented- und Virtual Reality. Das Hauptaugenmerk der Untersuchungen liegt dabei auf der Darstellung dieser Felder. Später werden Isopotentialflächen in den Mittelpunkt der Betrachtung rücken, deren Ursprung zwar ebenfalls in der Elektrostatik liegt, deren Algorithmen aber lediglich in Abhängigkeit eines Vektor- bzw. Skalarfelds formuliert werden. In Abschnitt 1.1.1 haben wir bereits einige Anwendungsbereiche aus dem elektrotechnischen Bereich vorgestellt. Weitere liegen in der Visualisierung von Flüssigkeitssimulationen, die sich das organische, tropfenförmige Aussehen von Icospheres zunutze machen [Bri08], was für Isoflächen die Beinamen „Blob“ [Bri08, passim] [Sta16, S. 101ff] [Eng+06a, S. 260ff] [Mar+21,

S. 551ff] und „Meta-ball“ [Sta16, S. 101] [Mar+21, S.551ff] [Luc19, S. 22] etabliert hat. Weniger bekannte Einsatzgebiete umfassen die Themengebiete der Meteorologie [Wat+10], das Industriedesign [Sch+20], Pflanzenwachstumssimulationen [JK00, S. 10] und die Atomphysik [Har16].

In späteren Kapiteln wird sich herausstellen, dass eine algebraische Berechnung der Punkte einer Isofläche allgemein nicht möglich ist. Dies führt zu der Erkenntnis, dass eine einfache Berechnung des elektrischen Feldes aus einer gegebenen Ladungsverteilung nicht ausreicht, um dieses aussagekräftig durch eine Isofläche wiederzugeben. Daher stützt auch diese Arbeit die Generierung des Meshes, das das elektrische Feld repräsentieren soll, in großen Teilen auf dessen Simulation.

Eine Isofläche wird üblicherweise durch Software berechnet, deren Implementierung sich auf ein Framework, beispielsweise eine Graphik-Engine stützt. Die Simulation des elektrischen Feldes und seine Darstellung durch ein Mesh erfolgen in diesem Rahmen durch eine Funktion, die einen der im Verlauf der Arbeit vorgestellten Algorithmen implementiert. Grundlage der Berechnung sind hier in der Szene der Simulation frei platzierbare, gegebenenfalls während der Simulation veränderliche und bewegliche Ladungselemente, die in diesem Kapitel vorgestellt werden.

Obwohl diese Arbeit von „elektrischen Feldern“ spricht, erfolgen die Argumentationen im Verlauf dieser Arbeit anhand von Skalar- und Vektorfeldern. Dieses Vorgehen kann auf der theoretischen Ebene durch den Hinweis darauf gerechtfertigt werden, dass die Permittivität $\varepsilon_0\varepsilon_r$ in der Gleichung zur Berechnung des elektrischen Feldes

$$\mathbf{E} = \frac{\mathbf{D} - \mathbf{P}}{\varepsilon_0\varepsilon_r}$$

lediglich eine multiplikative Konstante darstellt. Dies geht mit einigen Einschränkungen einher: Wir begrenzen unsere Argumentation damit auf homogene, nicht polarisierte Dielektrika. Diese Einschränkungen sind für die Betrachtungen in dieser Arbeit aber nicht von Belang. In der Tat handelt es sich hierbei um Sonderfälle, für die bessere Arten der visuellen Darstellung vorliegen, wie durch Vektoren in Polarkoordinaten. Zum Abschluss dieser Arbeit liegen keine wissenschaftlichen Erkenntnisse vor, die die betrachterfreundliche Darstellung elektrischer Felder besprechen und dabei die Möglichkeit der Inhomogenität oder Polarisation berücksichtigen. Die Quellen in Kapitel 1 nehmen zumindest implizit ein Vakuum als Medium an. In jedem Fall kann das resultierende Feld nach erneutem Hinzufügen der Permittivitätskonstanten den selben Methoden zur Darstellung zugeführt werden, wie ohne diese.

Diese Sichtweise erlaubt eine abstrakte, einheitenlose Betrachtung elektrischer Felder, die sich nun auf alle Fälle – auch abseits der Elektrotechnik – anwenden lässt, solange ein Vek-

torfeld vorliegt: James Blinn, der erstmals mit der Darstellung von Isopotentialflächen durch Raytracing die Thematik aus Sicht der Computergraphik beleuchtete, legte in seinen Versuchen keine elektrischen Felder sondern die Elektronendichtefunktion von Wasserstoffatomen zugrunde [Bli82, S. 237]. Ebenfalls ist eine Anwendung der später vorgestellten Algorithmen zur Darstellung frequenzabhängiger, durch Wechselstrom induzierter elektromagnetischer Felder oder die bereits oben erwähnten Abstrahlcharakteristiken von Antennen denkbar. Der Fokus auf ausgewählte Aspekte der Elektrostatik zu legen schränkt also die Allgemeinheit des Begriffes des darzustellenden Feldes nicht ein.

Dennoch eignet sich die Elektrostatik aufgrund der Anschaulichkeit und guten mathematischen Beschreibbarkeit von Punktladungen und darauf aufbauenden anderen Ladungsverteilungen besonders als Grundlage der folgenden Untersuchungen. Ob die Betrachtungen anhand der elektrischen Flussdichte oder des elektrischen Feldes vorgenommen werden, ist nach den obigen Befunden nicht relevant. Diese Arbeit verwendet wahlfrei das Symbol des elektrischen Feldes ($\mathbf{E}$)

2.4.1 Punktladungen

Elektrische Ladungen und Felder sind theoretische Konstrukte, die herangezogen werden, um elektrotechnische Phänomene wie mechanische Kräfte zu erklären, die ein geladener Gegenstand auf eine andere untersuchte Ladung q ausübt. Ladung kann nicht ohne Raumausdehnung auskommen. Allerdings ist für die folgenden Überlegungen diese nur kurz nach Aufbringen bzw. Auslenken der Ladung relevant, und spielt nach einer ausreichend groß annehmbaren „Einschwingzeit“ für Untersuchungen im Rahmen der Elektrostatik keine weitere Rolle [Bab16, S. 33]. Dieser Umstand motiviert die Argumentation anhand einer Punktladungsverteilung q, die auf einen einzelnen Punkt im Raum konzentriert ist.

Punktladung, Testladung

Eine *Punktladung* an der Position $\mathbf{p}$ mit der Ladungsstärke q ist definiert durch das Tupel

$$q = (\mathbf{p}, q)$$

mit $\mathbf{p} \in \mathbb{R}^3$ und $q \in \mathbb{R}$.

Eine *Testladung* ist eine Punktladung, auf die das elektrische Feld zwar wirkt, deren Ladung allerdings selbst bei der Berechnung des Feldes keine Berücksichtigung findet.

Eine Punktladung ist im diesem Kontext eine Ladungsmenge ohne Masse, die mit ihr verbunden ist. Wenn eine Testladung eingebracht wird, wird sie durch eine Kraft, die wir vereinfacht als elektrisches Feld bezeichnen, von der Punktladung in Bewegung versetzt.

Elektrisches Feld

Eine Punktladung $(q, \mathbf{p})$ erzeugt ein elektrisches Feld

$$\mathbf{E}(\mathbf{p}) = \frac{q}{4\pi \left\|\mathbf{p}\right\|^3}\mathbf{p} \tag{2.4}$$

Mehrere Punktladungen addieren ihre Wirkungen vektoriell. Die beobachtete Kraft nimmt ab, je weiter die Testladung von der Punktladung entfernt ist [Bli91]. Die Wirkung dieses Kraftfeldes wird *elektrischer Fluss* genannt, der bezogen auf den Raum eine physikalische Dichte beschreibt.

Das elektrische Feld ist ein Vektorfeld, das durch eine Menge von Punktladungen $Q = \{(\mathbf{p}_1, q_1), (\mathbf{p}_2, q_2), \ldots\}$ erzeugt wird:

$$\mathbf{E}(\mathbf{r}) = \sum_i \frac{q_i}{4\pi \left\|\mathbf{r}_i - \mathbf{p}_i\right\|^3} (\mathbf{r}_i - \mathbf{p}_i) \tag{2.5}$$

Obwohl wir uns in unseren Betrachtungen auf elektrostatische Felder fokussieren, werden wir an verschiedenen Stellen unsere Betrachtungen auf dynamische elektrische Felder Übertragen.

Statisches und dynamisches Feld

Ein elektrisches Feld heißt *dynamisch*, wenn es eine zeitlich veränderliche Komponente enthält. Ihm steht das *statische* Feld gegenüber. [Bab16, S. 33]

Gemäß der obigen Definition des elektrischen Feldes liegt dessen Quelle ausschließlich in den Punktladung. Mithin kommen diese alleine als Ursprung der zeitlichen Veränderung in Frage. Dadurch ist in beiden Komponenten einer Punktladung – ihrer Position und ihrer Ladungsstärke – eine Erweiterung auf die Funktionen $q_i : \mathbb{R} \to \mathbb{R}$ und $\mathbf{p}_i : \mathbb{R} \to \mathbb{R}^3$ für die i-te Punktladung vorzusehen. Die Darstellung eines dynamischen elektrischen Feldes ergibt sich also zu:

$$\mathbf{E}(\mathbf{r}, t) = \sum_{i \in \mathbb{N}} \frac{q_i i(t)}{4\pi\varepsilon_0\varepsilon_r \left\|\mathbf{r}_i - \mathbf{p}_i(t)\right\|^3} (\mathbf{r}_i - \mathbf{p}_i(t))$$

wobei sich hier die Menge aller betrachteten Punktladungen ebenfalls als Erweiterung des Mengenbegriffs durch die Zeitkomponente als Funktion $Q(t) = Q(t) = \{(\mathbf{p}_1(t), q_1), (\mathbf{p}_2(t), q_2), \ldots\}$ mit den Zeitvektoren $\mathbf{p}_i(t)$ für alle $i \in \mathbb{N}$ darstellen lässt.

Einige Überlegungen im Verlauf der Arbeit – besonders die, die sich auf die Generierung von Isoflächen beziehen – erfordern für die Darstellung als Grundlage ein Kontinuum als Feld.

Stetigkeit, Kontinuum

Eine Funktion $f : [a, b] \to \mathbb{R}$ [Sim08, S. 107] ist unter folgenden Bedingungen [AK19, S. 212] *stetig* für alle Punkte c im Intervall $[a, b]$:

- $f(x) = c$
- $\lim_{x \to c^-} f(x) = c$
- $\lim_{x \to c^+} f(x) = c$

Eine Funktion, für die diese Bedingungen für alle Punkte $c \in \mathbb{R}$ gelten, heißt *Kontinuum.*

Kontinuen bilden Gruppen bezüglich der Addition und der Multiplikation der Funktionen mit der Subtraktion und der Division als ihren Inversen. Laut [AK19, S. 198] sind Polynome Kontinuen, so auch Konstanten, die insbesondere Polynome sind. Daraus folgt, dass es sich beim Rechten Teil der Gleichung (2.4) um ein Kontinuum handelt. Bei Gleichung (2.5) handelt es sich als Polynom um die benannte Untergruppe der Kontinuengruppe, wodurch diese Struktur nach Anwenden der Addition erhalten wird. $\mathbf{E}(\mathbf{p})$ ist als linker Teil der Gleichung also ebenfalls ein Kontinuum.

Aufbauend auf Punktladungen wird für die im Hauptteil der Arbeit verwendete Theorie der elektrischen Felder neben dem Begriff des elektrischen Feldes auch der des Potentialfelds relevant.

Potentialfeld

Das Potentialfeld des elektrischen Feldes $\mathbf{E}$ berechnet sich durch[Bab16, S. 39]

$$\nabla \varphi = -\mathbf{E} \tag{2.6}$$

Dabei ist der Operator $\nabla : \mathbb{R}^n \to \mathbb{R}^n$ (Nabla-Operator) die *Divergenz*, also der Vektor der partiellen Ableitungen seines Urbilds liefert. Ihm ist zueigen, dass sein Abbild stets orthogonal zum Urbild ist [Rai18, Ss. 464ff]. Isoflächen stehen in der Physik weit häufiger in ihrer prominenteren Form als *Äquipotentialflächen* – Flächen mit gleichem elektrischen Potential – im Mittelpunkt der Betrachtung [Bab16, S. 34], [Nav16; Sch21, passim]. In Gleichung (2.6) zeigt sich allerdings, dass zwischen elektrischem und Potentialfeld ein konstanter bzw. differentieller Zusammenhang besteht. Da Kontinuen bezüglich der Multiplikation eine Gruppenstruktur aufweisen, die Differentiation eines Polynoms wiederum ein Polynom ergibt, und Polynome Kontinuen sind, eignen sich die Mittel zur Darstellung von Äquipotentialflächen ebenso zur Darstellung von Isoflächen elektrischer Felder.

Anhand Abb. 2.1 kann eingesehen werden, dass diese Grundlagen für die annähernde flächenmäßige Darstellung elektrischer Felder ausreichen: Dort ist das elektrische Feld am Punkt p durch eine Fläche angenähert. Diese ist ein Teil der Ebene in Normalenform, die durch den Ortsvektor von p, und den Normalenvektor das elektrische Feld im Punkt p definiert ist. Für die Parameterform können die Vektoren $\boldsymbol{\varphi}$ und $\boldsymbol{\varphi} \times \mathbf{E}$ verwendet werden. Der Rest des Feldes kann durch weitere solcher Flächen angenähert werden. Die Güte dieser Annäherung hängt dann von der Größe der verwendeten Flächen ab.

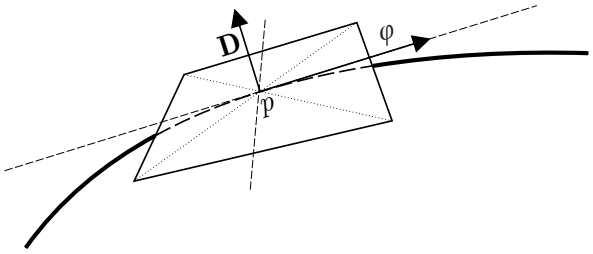


Abbildung 2.1: Tangentialfläche an p

2.4.2 Nicht-punktförmige Ladungsverteilungen

Eine erschöpfende Bearbeitung des Themas erfordert die Betrachtung anderer häufiger Arten von Ladungsverteilungen, allerdings fokussieren wir uns in dieser Arbeit auf die Betrachtung von Punktladungen. Für die Berechnung von Ladungsverteilungen anhand von Kurven bzw. Flächen stehen die Theorie über Kurvenintegrale [Men07, S. 5ff] bzw. Flächenintegrale [Men07, S. 23ff] zur Verfügung. Der Einfluss auf die Berechnung der Zeit- und Speicherkomplexitäten muss entsprechend berücksichtigt werden. Um Argumentationen anhand von diesen Ladungsverteilungen durchzuführen und deren Erkenntnisse nutzbar zu machen, werden diese durch Punktladungen angenähert, um sie somit als Sonderfälle der Gleichung (2.4) ansehen zu können.

Eine *linienförmige Ladungsverteilung* entlang einer Kurve C mit der gleichförmigen Ladungsverteilung q_l – also der Verteilung einer Ladung mit konstanter Ladungsdichte – kann durch eine gleichförmige Verteilung von Punktladungen angenähert werden, deren Auswirkungen auf das elektrische Feld sich überlagern. Ihr elektrisches Feld berechnet sich durch [Bab16, S.30]

$$\mathbf{E}_{\mathrm{L}}(\mathbf{r}) = \sum_i \mathbf{E}(q, \mathbf{q}_i) \tag{2.7}$$

wobei $\mathbf{E}(q, \mathbf{q}_i)$ das elektrische Feld der i-ten aus einer Menge indizierter Punktladungen ist. Bei Bekanntheit der Ableitung $\dot{\rho}$ der Verteilungsfunktion ρ oder der Stammfunktion des

Vektorfeldes kann die exakte Berechnung durch das Kurvenintegral

$$\mathbf{E}_{\mathrm{L}}(\mathbf{r}) = \frac{1}{4\pi \|\mathbf{r}\|^3}\mathbf{r}\int_C q\rho(t)\cdot\dot{\rho}(t)\mathrm{d}s$$

mit dem Parameterbereich $t \in [\alpha, \beta]$ durchgeführt werden [Men07, S. 9]. In Anbetracht der Vorarbeit von [Wie18, S. 23] mag der Fall einer Ladungsverteilung entlang eines Spline von gesteigertem Interesse sein:

Linienladung entlang eines Spline

Gegeben sei eine an dem Spline [Moe03] mit der parametrischen Darstellung $\mathbf{p}(t)$ im Parameterbereich $[\alpha, \beta]$ homogen verteilte Ladung q. Das elektrische Feld berechnet sich durch:

$$\mathbf{E}(\mathbf{r}) = \frac{1}{4\pi \|\mathbf{r}\|^3}\mathbf{r}\int_\alpha^\beta q\mathbf{p}(t)\dot{\mathbf{p}}(t)\mathrm{d}t$$

Ist der Aufwand zum Lösen des Integrals oder der Bestimmung von $\dot{\mathbf{p}}$ nicht zu rechtfertigen, lässt sich das Feld in Anlehnung an Gleichung (2.7) durch n Punktladungen annähern durch:

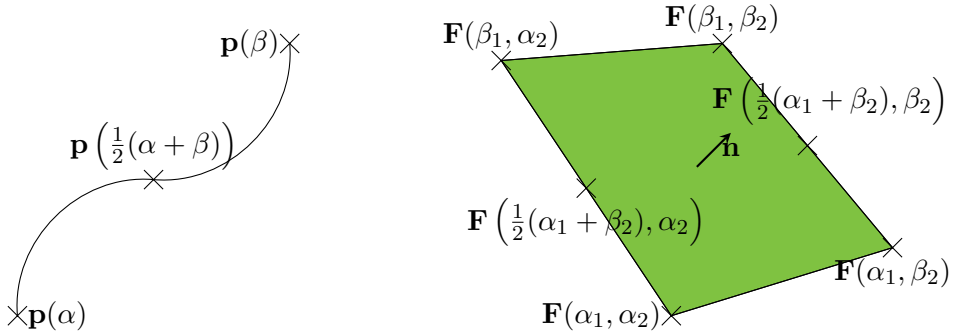
$$\mathbf{E}(\mathbf{r}) \approx \frac{1}{4\pi \|\mathbf{r}\|^3}\mathbf{r}q\left(\mathbf{p}(\alpha) + \mathbf{p}\left(\alpha\left(1-\frac{1}{n}\right)+\frac{1}{n}\beta\right) + \cdots + \mathbf{p}(\beta)\right)$$

Abb. 2.2a zeigt eine beispielhafte Anordnung für $n = 3$.

Eine *Oberflächenladung* berechnet sich analog durch das Bereichsintegral [Men07, S. 30]

$$\int_B \mathbf{f}(\mathbf{r}(u,v))\cdot\|\mathrm{d}\|\,u\mathrm{d}v$$

mit den Tangentialvektoren $\mathbf{r}_u$ und $\mathbf{r}_v$.



(a) Durch Punktladungen angenäherte linienförmige Ladungsverteilung

(b) Durch Punktladungen angenäherte Oberflächenladung

Feld einer Oberfläche

Gegeben sei eine auf der Mantelfläche $\mathbf{F}$ mit den Parameterbereichen $[\alpha_1, \beta_1]$ und $[\alpha_2, \beta_2]$ gleich verteilte Ladung q. Die Flächennormale lässt sich aus den beiden Tangentialvektoren berechnen, womit das Integral

$$\mathbf{E}(\mathbf{r}) = \frac{q}{4\pi \left\|\mathbf{r}\right\|^3}\mathbf{r} \int_{\alpha_1}^{\beta_1} \int_{\alpha_2}^{\beta_2} \left\| \frac{\mathrm{d}}{\mathrm{d}x}\mathbf{F} \times \frac{\mathrm{d}}{\mathrm{d}y}\mathbf{F} \right\| \mathrm{d}o_2 \mathrm{d}o_1$$

bestimmt ist.
Sollte wiederum der Aufwand zur Berechnung der Integrale oder die Bildung der Ableitung nicht zu rechtfertigen sein, kann analog zum vorherigen Beispiel, nur hier in zwei Dimensionen, verfahren werden. Abb. 2.2b zeigt diese anhand einer rechteckigen Fläche im Raum.

2.4.3 Rechengeschwindigkeit

Häufig werden an der Gleichung (2.4) verschiedene Vereinfachungen vorgenommen, um Berechnungen auf Kosten der Genauigkeit vorzunehmen. Blinn schlägt vor, den Exponenten im Nenner zu erhöhen, um die Berechnung des Betrags zu beschleunigen[Bli82, S. 238].

Dieser Überlegung können wir hier allerdings nicht wörtlich folgen: Die Problemstellung verlangt die Berücksichtigung der Permittivitätskonstanten $\varepsilon_0\varepsilon_r$, um die Anwendung einer elektrotechnischen Untersuchung zugänglich zu halten. Jedoch können wir mit Hilfe des Distributivgesetzes die nicht von der Laufvariablen abhängigen konstanten $4\pi\varepsilon_0\varepsilon_r$ ausklammern und zu einer Konstante c zusammenfassen. Dieser geht als multiplikativer Anteil in die Modellmatrix ein, die für die Darstellung des Meshes aufgestellt wird. Er fungiert als Skalierungsfaktor und kann als solcher, sofern die Konstanten bekannt sind, unabhängig von der Simulation berechnet werden. In Darstellungsformen, die invariant bezüglich der Skalierung sind, z. B. *fit-to-screen*, kann $c = 1$ gesetzt werden und entfällt somit bei der Berechnung.

Darüber hinaus befassen wir uns im Abschnitt 2.8 ausführlich mit der Zeitkomplexität der Algorithmen zur Berechnung der Geometrie.

2.5 Isoflächen, Flächen, Cluster

In Kapitel 1 haben wir die Nachteile der Darstellung elektrischer Felder durch Pfeil- und Feldliniendarstellungen herausgearbeitet. Diesen Darstellungsmethoden und ihren Mängeln wollen wir in diesem Kapitel die Isopotentialflächen entgegenstellen. Die folgenden Abschnitte beleuchten die Thematik des Clusterings näher. Wir finden zunächst für das Verständnis von Flächen und Isoflächen notwendige Definitionen. Danach leiten wir aus diesen Definitionen und anderen bekannten Gesetzmäßigkeiten der Elektrostatik wichtige Eigenschaften ab, die das Problem des Clusterings motivieren und in den folgenden Abschnitten dieses Kapitels bei den verschiedenen Möglichkeiten, den potentiell disjunkten Charakter von Isoflächen zu berücksichtigen, als Maßgabe für das Erreichen das Ziels der Arbeit dienen.

Isofläche und Isowert

Eine *Isofläche*, *Isopotentialfäche* oder *iso-surface* ist die abgeschlossene Hülle $\bar{\mathbf{p}}$ aller Punkte $\mathbf{p}$, die in einem elektrischen Feld $\mathbf{E}$ für einen festgelegten Wert ξ die Bedingung

$$\mathbf{E}(\mathbf{p}) > \xi$$

erfüllen. Der Wert ξ heißt Isowert.

Diese Definition beschreibt die Menge aller Punkte, für die $\mathbf{E}(\mathbf{p}) = \xi$ gilt. Bei dieser Betrachtungsweise bilden Isoflächen den Rand eines kompakten, berandeten Raumes und sind ihrerseits kompakt, aber unberandet. Anschaulich werden kompakte Flächen auch *geschlossen* genannt, um auszudrücken, dass diese keine „Öffnungen“ aufweisen. Die Eigenschaft der Geschlossenheit folgt unmittelbar aus der Kontinuität elektrischer Felder. Aus diesem Grund verwenden wir den Begriff der „einwertigen Mengen von Flächen“ synonym mit dem der „nicht disjunkten Isoflächen“.

Für manche Ladungsverteilungen und Werte für ξ ist die Isofläche disjunkt. Abbildung 2.3 zeigt eine solche Situation. Dieser Umstand macht eine Unterscheidung der Flächen notwendig.

Fläche

Eine *Fläche* einer Isofläche bezeichnet die abgeschlossene Hülle einer Untermenge der Punkte dieser Isofläche, sofern diese Fläche geschlossen ist.

Da Flächen geschlossen sind, können diese sich weder gegenseitig noch selbst schneiden: Dies würde voraussetzen, dass auf keinem Punkt der betreffenden Flächen der Wert des elektrischen Feldes den Isowert unterschreitet. Dies ist unmöglich, wie die Intuition zeigt und unten genauer ausgeführt wird. Im zweidimensionalen Fall stellt eine Isofläche also eine Menge von Jordan-Kurven dar [Jor87]. Für Flächen ist auch der Begriff „Zellen“ gebräuchlich [She98], den wir aber, weil er ebenfalls als Iterationsschritt des Marching Cube-Algorithmus [NH92] und seiner Abwandlungen [Par+98] Verwendung findet, vermeiden.

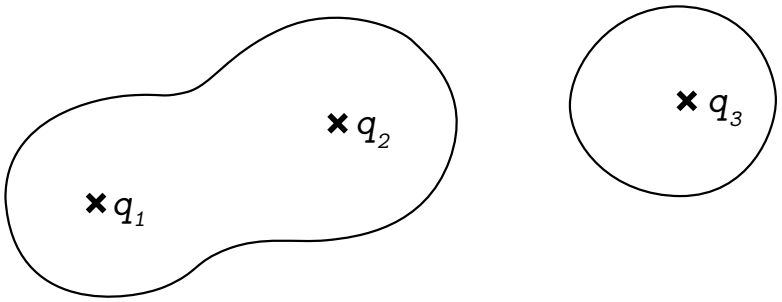


Abbildung 2.3: Punktladungscluster mit disjunkter Isofläche

Aus Abbildung 2.3 lässt sich ersehen, dass die Isoflächen die Punktladungen, die das betrachtete Feld erzeugen, beinhalten können. Die Erweiterung der Isofläche als Punktmenge auf ihre abgeschlossene Hülle ermöglicht hier eine intuitive Interpretation des Begriffs des *Beinhaltens* und mit diesem folgende Bemerkungen:

1. Da $\lim_{\mathbf{p}' \to \mathbf{p}} \mathbf{E}(\mathbf{p}) = \infty > \xi$ für beliebige $\mathbf{p}'$ und ξ, können keine Punktladungen außerhalb von Isoflächen existieren. .

2. Da alle Flächen geschlossen sind, kann jede Punktladung nur in einer Fläche enthalten sein.

3. Alle Punkte $\mathbf{p}$, die in mehreren Flächen enthalten sind, sind potenzielle Positionen für Punktladungen. Aufgrund Punkt 2 können aber keine zwei Flächen eine und dieselbe Punktladung enthalten. Somit können sich Flächen nicht ganz oder teilweise gegenseitig überschneiden – also sich weder gegenseitig beinhalten noch Punkte auf der Fläche miteinander teilen.

4. Das Gaußsche Gesetz lautet:
$$\oint_{\partial V} \mathbf{E} \mathrm{d}\mathbf{A} = q$$
für eine geschlossene Fläche ∂V, die eine Ladung $q \in \mathbb{R}$ mit beliebiger Verteilung vollständig beinhaltet und ein elektrisches Feld $\mathbf{E}$ [Bab16, S. 32]. Ist nun ∂V die Fläche einer

Isofläche, so gilt für mindestens einen Punkt $f \in \partial V$: $\mathbf{E}(f) > 0$. Daraus folgt, dass auch

$$\oint_{\partial V} \mathbf{E} \mathrm{d}\mathbf{A} > 0$$

und somit ebenfalls

$$q > 0$$

Negative Ladungen ermöglichen zwar Punkte $\mathbf{E}(f') < 0$, was allerdings erfordert, dass ein solcher Punkt innerhalb ∂V liegt, wodurch die enthaltene Ladung ebenfalls negativ ist. In diesem Fall gilt die gleiche Argumentation mit negativem Vorzeichen. Jede Fläche muss also Ladung enthalten. In dieser Arbeit kommen dafür nur Punktladungen in Frage. Punktladungen sind im Raum nicht ausgedehnt und können mithin nur ganz oder überhaupt nicht in einer Fläche enthalten sein. Jede Fläche einer Isofläche enthält somit mindestens eine Punktladung.

5. Der Begriff des „Beinhaltens“ stellt eine Äquivalenzrelation $\sim$ („Wird von der gleichen Fläche beinhaltet“) dar. Die Eigenschaften: $q_1 \sim q_1$ und $q_1 \sim q_2 \Rightarrow q_2 \sim q_1$ und $q_1 \sim q_2 \wedge q_2 \sim q_3 \Rightarrow q_1 \sim q_3$ für beliebige Punktladungen q_1, q_2, q_3 lassen sich anhand Abbildung 2.3 nachvollziehen. Die Flächen bilden die Äquivalenzklassen von $\sim$ und bilden eine Partition der Isofläche.

6. Komplementär zu Punkt 1 ist die Bedingung, dass Q' eine Äquivalenzklasse von $\sim$ ist, kein hinreichendes Kriterium, dass es einen Punkt $\mathbf{p}$ in $\bar{Q}'$ gibt, für den $\mathbf{E}(\mathbf{p}) < \xi$ gilt.

Cluster

Die Äquivalenzklassen $Q/\sim$ einer Menge von Punktladungen Q werden in dieser Arbeit als *Cluster* bezeichnet.

Ein Cluster ist also anschaulich eine Menge von Punktladungen, die in einer Fläche enthalten sind.

Für einige der in späteren Teilen dieser Arbeit vorgestellten Algorithmen spielt die Bestimmung der Cluster eine wichtige Rolle. Diesen Vorgang und sein Ergebnis nennen wir *Clustering*. Wie wir sehen werden, ist das Clustering unter Umständen mit hohem rechnerischen Aufwand verbunden, der sich aufgrund der Anforderungen der Eignung für virtuelle und erweiterte Realität nicht immer rechtfertigen lässt.

Soll das elektrische Feld durch Isoflächen dargestellt werden, muss dem Umstand, dass das es in disjunkten Flächen auftreten kann, Rechnung getragen werden. In diesem Ansatz ist die Ermittlung der Clustergrenzen unter Umständen mit hohem Aufwand verbunden, da in diesem

Fall auch zwischen den Flächen an jedem Iterationspunkt das elektrische Feld berechnet werden muss.

Da Meshes Flächen nur annähern können, tut eine Betrachtung des entstandenen Fehlers Not, der sich im Abstand jedes Punkts auf dem Mesh zum jeweiligen nächsten Punkt auf der Fläche manifestiert. [Rho+02] legen in einer Anwendung zur Darstellung von 3D-Scans durch Isoflächen den Marching Cubes-Algorithmus zugrunde. Dabei führen sie eine Erweiterung ein, durch die sie für jeden Vertex den Abstand zur angenäherten Fläche messen, diesen als Farbe auf den Vertex codieren und ihn durch einen Shader über das Dreieck interpolieren können. Über die Methodik der Berechnung des Abstands gibt der Artikel keine Auskunft. Die Abbildungen lassen aber vermuten, dass das als Maß für die Unsicherheit (*uncertainty*) die betragsmäßige Differenz zwischen gemessenem Wert des Feldes an der Fläche des Gitterpunkts und dem Isowertverwendet wurde.

Der exakte Verlauf der Fläche innerhalb eines Dreiecks in einem Mesh lässt sich zwar algorithmisch nicht bestimmen und ist im Allgemeinen beliebig, weswegen auch die Genauigkeit in der Entfernung der Mesh-Vertices nicht abgeschätzt, sondern lediglich interpoliert werden kann. Ist aber für die Anwendung der Darstellung elektrischer Felder vorausgesetzt, dass der Abstand der Vertices im Mesh den zu den Punktladungen deutlich übersteigen, lassen sich die obigen Ergebnisse präzisieren. In anderen Algorithmen zur Konstruktion von Meshes, die Flächen repräsentieren, als dem Marching Cubes-Algorithmus, der nicht auf Iteration in einem Gitter basiert, kann so die Genauigkeit gegenüber dem Ergebnis des Marching Cubes-Algorithmus erhöht werden.

Eine genaue Darstellung der Isofläche kann nur in den Vertices der repräsentierenden Meshes sichergestellt werden. Bei größer werdendem Abstand eines Punktes innerhalb eines Dreiecks von seinen Vertices erhöht sich der Abstand des entsprechenden Punkts auf dem Mesh zum nächsten Punkt auf der Fläche. Da eine steigende Entfernung eines Vertices einer Kante des Dreiecks – einer Linie – gleichzeitig eine geringer werdende Entfernung zum anderen Vertex dieser Kante bedeutet, aber lediglich auf den Kanten eine Abweichung von 0 vorliegt, ist zu erwarten, dass die Abweichung des Meshes zur Fläche auf der Kante zur Mitte hin maximal ist. Da sowohl die Fläche als auch das Dreieck abgesehen von seinen Kanten kontinuierlich sind, ist auch der Abstand zwischen beiden eine kontinuierliche Funktion. Somit ist der größte Anstieg des Abstands beim Entfernen vom Vertex innerhalb eines Dreiecks auf der Winkelhalbierenden zu finden. Diese schneiden sich im Schwerpunkt des Dreiecks, wo schließlich der größte Abstand zwischen Mesh und repräsentierter Fläche zu erwarten ist. Eine Illustration dieser Überlegung findet sich in Abbildung 2.4.

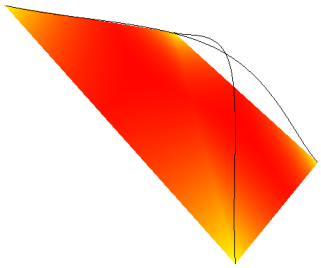


Abbildung 2.4: Fläche eines triangularen Meshes, Abstand zur Fläche (schwarz angedeutet) analog zu [Rho+02] durch Shader eingefärbt

2.6 Beweise der Terminierung und Korrektheit

Die im Hauptteil dieser Arbeit vorgestellten Algorithmen folgen dem Paradigma der imperativen Programmierung. Im Abstrakten ist ein Algorithmus ein Schema zur Lösung eines vorgegebenen Problems. Sowohl im ingenieurmäßigen als auch im wissenschaftlichen Kontext sind beim Entwickeln neuer Algorithmen neben Aufwandsanalysen auch Beweise der Korrektheit und Terminierung wünschenswert. Zu diesem Zweck müssen zunächst einige Begriffe spezifiziert werden.

Algorithmus, Programm und Anweisung

Ein *Algorithmus* ist eine mindestens einwertige Folge von Anweisungen mit einer Eingabe- und Ausgabemenge. Die zustandsbehaftete Repräsentation eines Algorithmus nennen wir *Programm*. Eine *Anweisung* ändert den Zustand eines Programms.

Algorithmen werden in dieser Arbeit nicht nur in der Mengenschreibweise wie in Abschnitt 2.2 sondern auch zeilenweise als Pseudocode ausformuliert werden. Ein Beispiel für diese Darstellungsweise zeigt Algorithmus 1.

Das Hoare-Kalkül schließt unter Annahme der Gültigkeit einer Nachbedingung $(\!| n |\!)$ und der Annahme einer Vorbedingung $(\!| v |\!)$ die Korrektheit eines Algorithmus S. Dieser Umstand wird durch das *Hoare-Tripel* $(\!| v |\!) \quad S \quad (\!| n |\!)$ beschrieben. Bei v und n handelt es sich um boolesche Aussagen – also Aussagen, die entweder *wahr* ($\top$) oder *falsch* ($\bot$) sind. Aus dem Wahrheitsgehalt von n wird schließlich der Wahrheitsgehalt der Aussage

$$v \wedge n \Rightarrow S \text{ ist korrekt}$$

Algorithmus 1 : Algorithmus zur Collatz-Vermutung

eingabe : $i \in \mathbb{N}$
ausgabe : $\varnothing$

```
1 solange i ≠ 1 tue
2     wenn i gerade dann
3         i ← i/2 ;                                  // Zuweisung
4     Ende
5     sonst
6         i ← 3 · i + 1
7     Ende
8 Ende
```

gefolgert. Muss die der letzten Zeile folgende Zeile eines Programms ausgeführt werden, wenn schon gezeigt ist, dass diese letzte Zeile in jedem Fall erreicht wird, ist implizit die Terminierung des Algorithmus gezeigt.

Neutrale Anweisung

Für die *neutrale Anweisung* nop gilt

$$(\!| a |\!) \quad \texttt{nop} \quad (\!| a |\!)$$

Diese Anweisung verändert den Zustand des Programms also nicht.

2.7 Axiome des Hoare-Kalküls

2.7.1 Zuweisungsaxiom / Einsetzungsregel

Seien $a(x)$ eine Aussage in Abhängigkeit einer beliebigen Variablen x und n_1 und n_2 beliebige Variablen der gleichen Menge. Nach Einsetzen einer dieser Variablen in a gilt für den Zustand des Programms:

$$(\!| a(n_1) |\!) \quad n_2 \leftarrow n_1 \quad (\!| a(n_2) |\!)$$

Durch die Zuweisung gilt in der Nachbedingung die Aussage a nun für die Variable n_2.

Beispiel

$(\!| i = n_1 |\!)$
$n_2 \leftarrow n_1$
$(\!| i = n_2 |\!)$

In diesem Beispiel ist die Aussage $a(n_1) = (i = n_1)$. Nach der Zuweisung gilt ebenfalls die Aussage $a(n_2)$.

2.7.2 Kompositionsaxiom / Sequenzregel

Seien n_1 und n_2 beliebige Variablen, a_1 und a_2 beliebige Aussagen über diese Variablen und S_1 und S_2 Algorithmen in den Hoare-Tripeln:

$$\{\!|\, a(n_1) \,|\!\}\quad S_1 \quad \{\!|\, a_1 \,|\!\} \tag{2.8}$$

und

$$\{\!|\, a_1 \,|\!\}\quad S_2 \quad \{\!|\, a(n_2) \,|\!\} \tag{2.9}$$

Nach Konkatenation der beiden Algorithmen ergibt sich wegen der Gleichheit der Nachbedingung aus Gleichung (2.8) mit der Vorbedingung aus Gleichung (2.9):

$$\{\!|\, a(n_1) \,|\!\}\quad S_1 + S_2 \quad \{\!|\, a(n_2) \,|\!\}$$

Analog zum letzten Beispiel gilt nach Abarbeitung des $S_1 + S_2$: $a(n_1) = a(n_2)$.

2.7.3 Auswahlaxiom / if-then-else-Regel

Seien B eine Bedingung und S_1, S_2 Algorithmen und S_3 die Anweisung:

```
1 wenn B dann
2 |  S1
3 Ende
4 sonst
5 |  S2
6 Ende
```

in den Hoare-Tripeln:

$$\{\!|\, a \wedge B \,|\!\}\quad S_1 \quad \{\!|\, b \,|\!\}$$

und

$$\{\!|\, a \wedge \overline{B} \,|\!\}\quad S_2 \quad \{\!|\, b \,|\!\}$$

Die Bedingung in S_3 bewirkt, dass unter Abhängigkeit von B entweder S_1 oder S_2 zur Ausführung kommt. Für die Korrektheit eines Beweises muss nun die Vorbedingung eines der

beiden Tripel oder beide gelten, wodurch sich als Hoare-Tripel für $S3$:

$$\begin{aligned} &\{\!|\ (a \wedge B) \vee (a \wedge \overline{B})\ |\!\} \quad S_3 \quad \{\!|\ b \vee b\ |\!\} \\ =&\{\!|\ a\ |\!\} \quad S_3 \quad \{\!|\ b\ |\!\} \end{aligned}$$

ergibt. Der Else-Zweig kann in Algorithmen entfallen. In diesem Fall gilt die gleiche Regel mit $S_2 = \texttt{nop}$.

2.7.4 Iterationsaxiom / while-Regel

Sei I eine Aussage, B eine Bedingung und S_1 der Algorithmus in Alg. 2:

Algorithmus 2 : While-Schleife

```
1 solange B tue
2 |  S1
3 Ende
```

Die Aussage I fungiert als Bindeglied zwischen zwei Durchläufen der Schleife: Sie muss während der Schleifendurchläufe und somit sowohl vor als auch nach der Schleife gelten. Aus diesem Grund wird sie *Schleifeninvariante* genannt [GG05]. In der Nachbedingung von S_2 muss berücksichtigt werden, dass die Schleife, geendet haben muss, weswegen in der Nachbedingung B nicht mehr wahr sein kann. Daraus ergibt sich für die die Korrektheit von S_2 das Hoare-Tripel:

$$\{\!|\ I \wedge a\ |\!\} \quad S_2 \quad \{\!|\ I \wedge \overline{B}\ |\!\}$$

Für Beweise mittels des Hoare-Kalküls muss die Schleifeninvariante derart ermittelt werden, dass sie dem Iterationsaxiom Rechnung trägt. Nicht für alle Schleifen ist eine Schleifeninvariante bestimmbar, und es existiert kein Algorithmus für ihre Bestimmung, falls sie existiert. Obwohl für Sonderfälle Techniken zu ihrer Bestimmung existieren [Ham17], ähnelt die Suche nach einer Schleifeninvariante allgemein stark der Ermittlung des Induktionsschritts beim Führen eines mathematischen Beweises durch vollständige Induktion [Dij80]. Der Beweis erfolgt dadurch, dass die Wahrheit der Schleifeninvariante vor, während und nach der Schleife gewährleistet wird. Außerdem muss für die Bedingungen B innerhalb der Schleife, nach der Schleife allerdings $\overline{B}$ der Wahrheitsgehalt nachgewiesen werden.

Für einen Beweis der Korrektheit einer Schleife können mehrere Schleifeninvarianten bestimmbar sein und zur Auswahl stehen [Lut09, S. 62]. Als triviale Schleifeninvariante ist die Aussage „wahr“ vor und nach jedem Algorithmus wahr. Als Kriterium der Qualität werden in dieser Arbeit nur Schleifeninvarianten gewählt, die in Abhängigkeit von den im Algorithmus veränderten Variablen abhängen und gezeigt, dass diese sich trotz dieser Veränderung selber

nicht verändern. Abgesehen von diesen Überlegungen hängt die Qualität des Beweises nicht von der Wahl der Schleifeninvarianten ab.

2.7.5 Stärken der Vorbedingungen und Schwächen der Nachbedingungen

Für logische Schlussfolgerungen im Zuge von Beweisen mit dem Hoare-Kalkül müssen Bedingungen, von denen Aussagen abhängen, umgeformt werden. Die Stärke der Bedingung kann dabei nicht beliebig geändert werden, ohne dabei die Argumentation zu devalidieren.

Stärke der Bedingung

Eine Bedingung a ist stärker als eine andere Bedingung b, wenn

- $a \Rightarrow b$ und
- $a \neq b$ bzw. äquivalent $a \not\Leftarrow b$.

Aus der Wahrheitstabelle in Abb. 2.5 wird ersichtlich, dass in der Aussage $a \Rightarrow b$ die Aussage b spezifischere Aussagen zulässt als a.

Beispiel zur Stärke der Bedingung

Aus:

- „Rainer ist Rudolfs Erstgeborener“ $\Rightarrow$ „Rudolf ist Rainers Vater“
- „Rainer ist Rudolfs Erstgeborener“ $\not\Leftarrow$ „Rudolf ist Rainers Vater“

folgt, dass die Aussage „Rainer ist Rudolfs Erstgeborener“ stärker ist, als „Rudolf ist Rainers Vater“.

Ebenso lässt sich erkennen, dass die Aussage $\bot$ die stärkste Bedingung ist, da $\bot \Rightarrow b = \top$ für alle b. Dem Beweis ist es zuträglich, wenn die Vorbedingungen des Algorithmus möglichst allgemein sind, da er so für mehr Fälle gültig ist. Gleichzeitig sollten die Nachbedingungen möglichst spezifisch sein, da die Aussage auf diese Art mehr Informationen enthält. Aus diesem Grund werden in dieser Arbeit Vorbedingungen ausschließlich gestärkt und Nachbedingung ausschließlich geschwächt. Diese Schritte schränken die Allgemeinheit des Beweises nicht ein.

Unter den Voraussetzungen dass $a \Rightarrow b$ und $c \Leftarrow d$, gelten also für beliebige Aussagen m und n die Hoare-Tripel

$$(\!| a \wedge b |\!) \quad S_1 \quad (\!| n |\!) \qquad \text{und} \qquad (\!| m |\!) \quad S_2 \quad (\!| c \wedge d |\!)$$

a	b	$a \Rightarrow b$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

Abbildung 2.5: Wahrheitstabelle für Implikation

2.8 Komplexitätanalyse und Landau-Symbole

Wir verwenden die konventionellen Landau-Symbole, um das asymptotische Verhalten von Funktionen zu beschreiben. Die zu untersuchenden Funktionen sind in diesem Fall die Anzahl der abzuarbeitenden elementaren Operationen in Abhängigkeit von der Größe der Eingabe. In dieser Arbeit sind damit Fließkommaoperationen gemeint. Für den Vergleich der Algorithmen steht die Betrachtung des Symbols $\mathcal{O}$ im Mittelpunkt. Es beschreibt die Menge aller Funktionen, die durch die Funktion im Parameter von $\mathcal{O}$ beschrankt sind.

Asymptotische obere Schranke

Sei $f(x)$ eine Funktion mit beliebiger Definitionsmenge. Die *asymptotische obere Schranke* $\mathcal{O}(g)$ ist die Menge aller Funktionen g, für die es positive Konstanten $x_0 \in \mathbb{R}$ und $c \in \mathbb{R}$ gibt, so dass: $f(x) \leq c \cdot g(x)$ für $x > x_0$ [CLR09, S. 44] gilt. In Klassenschreibweise:

$$\mathcal{O}(g) = \{j \quad | \quad \exists x_0 \in \mathbb{R} \ \wedge \ \exists c \in \mathbb{R} \ \wedge \ f(x) \leq c \cdot j(x) \forall x > x_0\} \text{ für alle } f(x) \tag{2.10}$$

Die obige Definition besagt anschaulich, dass die Aussage $f \in \mathcal{O}(g)$ äquivalent ist zu der Aussage: f Wächst höchstens so schnell wie g; Für alle Parameterwerte x, die größer sind als x_0, ist f immer kleiner oder gleich g. Eine Klasse enthält demnach auch immer alle Klassen langsamer wachsenden Funktionen.

Beispiel

n^2 wächst schneller als n, also $\mathcal{O}(n) \in \mathcal{O}(n^2)$

Für die Analyse von Funktionen wurde in Kapitel 2.6 das Konzept des Algorithmus als Folge von Anweisungen eingeführt. In diesem Sinne können Elementaroperationen in etwa mit Elementen einer solchen Folge verglichen werden, solange diese keine Sprunganweisungen beinhalten.

Mit Einführen eines für Folgen distributiven zeitlichen Verzögerungsoperators $\cdot : \mathbb{R} \times A \to A$, der als „ein Element a aus der Menge der Anweisungen A dauert c-mal so lang, und $\cdot : \mathbb{R} \times f \to f$ ergibt sich eine Art Skalarprodukt für Folgen, die somit einerseits Algorithmen soweit in dieser Arbeit nötig mit den gleichen Mitteln wie Funktionen der Laufzeitanalyse unterzogen werden können. Andererseits ist mit diesem Operator ein Maß für eine hardwareunabhängige Laufzeit gefunden. In einem Term wie $c \cdot S$ mit einem beliebigen Algorithmus S gäbe somit ein reelles c ein Maß für die Leistungsfähigkeit der Hardware an. Für diese Arbeit erübrigen sich also die Angabe von gemessenen Laufzeiten: In einer Laufzeitanalyse behandeln wir einen mit einer Verzögerung c beaufschlagten Algorithmus so, als würde er c-mal nacheinander ausgeführt werden.

Die vorangegangenen Überlegungen motivieren „Rechenregeln“, die die Laufzeitanalyse mit Mitteln der Analysis vereinfachen können:

1. Die Multiplikation einer Funktion mit einer beliebigen Konstanten $c' \in \mathbb{R}$ bedeutet, dass $c' \cdot f(x) \leq c \cdot g(x)$. Eine Division durch diese Konstante bewirkt, dass $f(x) \leq \frac{c}{c'} \cdot g(x)$. Da sowohl $c \in \mathbb{R}$ als auch $c' \in \mathbb{R}$, muss auch $\frac{c}{c'} \in \mathbb{R}$ liegen und kann demnach als diejenige Konstante angesehen werden, von der die Aussage: $f(x) \in \mathcal{O}(g) \Rightarrow c \cdot f(x) \in \mathcal{O}(g)$ abhängt. Anschaulich kann diese Multiplikation durch den oben eingeführten Verzögerungsoperator ausgedrückt werden: Alle Gleitkommaoperationen in $c \cdot S_f$ dauern c-mal länger, als in S_f, Mit linearem Anwachsen der Eingabe wächst auch die Laufzeit von $c \cdot S_f$ höchstens linear.

2. Die Konkatenation zweier Algorithmen S_a und S_b führt zu einer Addition der Komplexitätsklassen $\mathcal{O}(S_a + S_b)$. Diese können unter Gleichung (2.10) interpretiert werden als

$$\mathcal{O}(S_a + S_b) = \{(a+b) \quad | \quad \exists x_0 \in \mathbb{R} \ \wedge \ \exists c \in \mathbb{R} \ \wedge \ f(x) \leq c \cdot (a+b)(x) \forall x > x_0\}$$

   Um von einer gemeinsamen Funktion g beschrankt zu sein, müssen beide Funktionen den Klassenbedingungen mit ihren eigenen individuellen Schranken genügen, so dass

$$\begin{aligned} \mathcal{O}(S_a + S_b) = \Big\{ g | \\ & \exists x_{0_a}, c_a \in \mathbb{R} && \wedge && \exists x_{0_b}, c_b \in \mathbb{R} \\ \wedge \quad & a(x) \leq c_a \cdot g(x) && \wedge && b(x) \leq c_b \cdot g(x) \\ \wedge \quad & x > x_{0_a} && \wedge && x > x_{0_b} \quad \Big\} \end{aligned} \tag{2.11}$$

Gleichung (2.11) lässt sich zu $x > \max(x_{0_a}, x_{0_b})$ als gemeinsamen Grenzwert der beschrankenden Funktion vereinfachen und somit gilt auch für die Klasse

$$\begin{aligned}\mathcal{O}(S_a + S_b) = \Big\{ g | \exists x_{0_a}, x_{0_b} \quad & \wedge \quad \exists c \in \mathbb{R} \\ & \wedge \quad a(x) > c \cdot g(x) \\ & \wedge \quad b(x) > c \cdot g(x) \Big\} \\ & \qquad \text{für alle } x > \max(x_{0_a}, x_{0_b}) \\ \in \mathcal{O}(\max(S_a, S_b))\end{aligned}$$

3. Wird ein Algorithmus S_f k Mal bei nicht konstantem $k \in \mathbb{N}$ ausgeführt, so ergibt sich $k \cdot S_f \in \mathcal{O}(k \cdot \mathcal{O}(f))$.

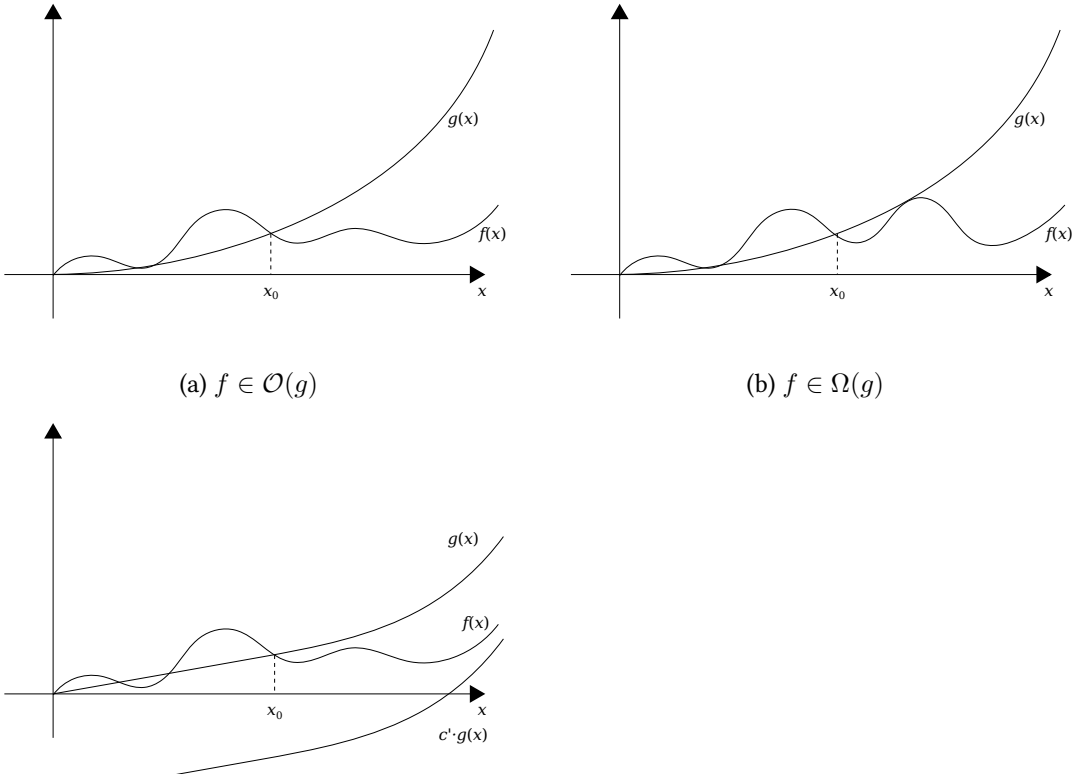
4. Kommt eine Folge von Anweisungen nur bedingt zur Ausführung, kann für das O-Kalkül immer davon ausgegangen werden, dass diese Bedingung mit einer Wahrscheinlichkeit von p erfüllt würde. Nach Regel 1 ergibt sich $\mathcal{O}(p \cdot S) \in \mathcal{O}(S)$. Außerdem kann eine Verzweigung in zwei Anweisungsfolgen S_f und S_g als deren Konkatenation $S_f + S_g$ und nach Regel 2 als Element der Komplexitätsklasse $\mathcal{O}(\max(S_f, S_g))$ betrachtet werden.

Für eine kurze anschließende Bemerkung werden die folgenden Landau-Symbole eingeführt:

Weitere Schranken

- Die *strenge asymptotische untere Schranke* Ω wird analog zu $\mathcal{O}$ definiert mit „$<$“ statt „$\leq$“. Die Menge Ω schließt also im Vergleich zu $\mathcal{O}$ diejenigen Funktionen aus, die gleich schnell wachsen, bzw. ab einem bestimmten Parameterwert gleich sind. [CLR09, S. 48].
- Die *asymptotische Ober- und Unterschranke* Θ wird analog zu $\mathcal{O}$ definiert mit einer zusätzlichen Konstante c', für die gilt: $f(x) \geq c' \cdot g(x)$ für $x > x_0$ [CLR09, S. 48].

Die Bedeutung der Definitionen der obigen Landau-Symbole wird in Abb. 2.6 veranschaulicht. Weitere Landau-Symbole existieren [Wil94, S.10], finden aber in dieser Arbeit keine Verwendung. Für Aufwandsanalysen wird meist das Symbol $\mathcal{O}$ herangezogen. Seine schwache Aussagekraft ist für die so leichtere Vergleichbarkeit meist ausreichend. Auch in dieser Arbeit werden Zeit- und Speicherkomplexität durch $\mathcal{O}(g)$ angegeben, sofern aber nicht anders angegeben, gilt diese Aussage ebenfalls für $\Omega(g)$ und $\Theta(g)$, ohne dass die entsprechende Argumentation weiter ausgeführt wird.



(a) $f \in \mathcal{O}(g)$

(b) $f \in \Omega(g)$

(c) $f \in \Theta(g)$

Abbildung 2.6: Veranschaulichung der Symbole $\mathcal{O}$, Ω und Θ

2.8.1 Speicherkomplexität

Die Ausführungen zur Zeitkomplexität im Abschnitt 2.8 mit der Bedeutung „Anzahl der Fließkommaoperationen" gelten analog für die Speicherkomplexität mit der Bedeutung „Anzahl der Fließkommavariablen".

2.9 Bird-Meertens-Notation

Diese Arbeit verwendet an verschiedenen Stellen die Bird-Meertens-Notation (*Bird Meertens Formalism*, BMF), um Algorithmen zu spezifizieren und herzuleiten. Dieser Formalismus findet seinen Ursprung in der Programmiersprache APL, die wiederum auf mathematischer Notation basiert. Elemente sowohl der Bird-Meertens-Notation als auch der APL-Syntax finden sich heutzutage in den Sprachen Matlab, Wolfram Language, Haskell und in verschiedenen Tabellenkalkulationsprogrammen. Die Bird-Meertens-Notation wird ebenfalls verwendet, um implementierungsunabhängig Algorithmen zur Lösung allgemeiner Probleme zu spezifizieren und durch algebraische Umformungen herzuleiten.

Auch wenn die Bird-Meertens-Notation auf andere Speicherstrukturen anwendbar ist [Bun94, passim], beschränken wir unsere Betrachtung auf Folgen. Unter dieser Voraussetzung kennt die Bird-Meertens-Notation als einzige Primitiva die Operatoren $\star : (l \to l) \times l^n \to l^n$ (gesprochen: „map“) und $/ : (l \times l \to l) \times l^n \to l$ (gesprochen: „reduce“) für ein beliebiges Objekt l. Der Operator $\star$ bildet eine Folge und eine Funktion auf eine andere Folge ab, indem er die übergebene Funktion auf jedes Element der übergebenen Folge anwendet [Bir86, S. 6]:

$$f \star \{l_1, l_2, \ldots\} = \{f(l_1), f(l_2), \ldots)\}$$

Der Operator $/$ bildet ebenfalls eine Funktion und eine Folge auf ein Objekt l ab. Typischerweise handelt es sich bei der übergebenen Funktion um einen weiteren binären Operator $\oplus : l \times l \to l$, der zwei Elemente aus l aufeinander abbildet. Der Operator $/$ listet die Elemente der Folge auf und verbindet sie durch den Operator $\oplus$:

$$\oplus / \{l_1, l_2, \ldots\} = l_1 \oplus l_2 \oplus \ldots$$

Beispiel: $\star$ und $/$

Die folgende Funktion berechnet den Betrag eines Vektors beliebiger Dimension, dessen Komponenten als Folge V gegeben sind:

$$\sqrt{(+/) \circ ((\,)^2 \star V)}$$

Abgesehen von $\star$ und $/$ finden die folgenden Funktionen in dieser Arbeit Verwendung:

Minimum Die Funktion $\min : O \times O \to O$, $\min(a,b) = \begin{cases} a & \text{wenn } a < b \\ b & \text{sonst} \end{cases}$ bestimmt den kleineren der beiden Werte a und b einer beliebigen Ordnung O.

Spezialisierung Die Funktion $g : (O \to O) \times O^2 \to O$,

$$g_f(a,b) = \begin{cases} a & \text{wenn } g(f(a), f(b)) = f(a) \\ b & \text{sonst} \end{cases}$$

bestimmt den Wert von a und b, der nach Anwenden der Funktion f auf beide Werte durch g ausgewählt würde. a und b sind dabei Teil einer beliebigen Ordnung O.

> **Beispiel: Spezialisierung**
>
> Das folgende Beispiel zeigt den Unterschied zwischen der nicht spezialisierten min-Funktion und der Spezialisierung $g = \min$ und $f(x) = x^2$ auf:
> $\min(-5, 2) = -5$
> $\min_{()^2}(-5, 2) = 2$

Erstes und letztes Element Die Funktionen $\text{head} : l^n \to l$, $\text{head}(\{a_1, a_2, a_3, \ldots\}) = a_1$ bzw $\text{last} : l^n \to l$, $\text{last}(\{a_1, a_2, \ldots, a_n\}) = a_n$ liefern das erste bzw. letzte Element einer beliebigen Liste.

Entfernen der ersten Elemente Die Funktion $\text{drop} : \mathbb{N} \times l^n \to l$,

$$\text{drop}(n, \{a_1, a_2, \ldots, a_n, a_{n+1}, \ldots, a_k\}) = \{a_n, a_{n+1}, \ldots, a_k\}$$

gibt die übergebene Liste ohne die ersten n Elemente zurück.

2.9.1 Currying

Durch Currying ist es möglich, Funktionen mit mehreren Parametern als mehrere Funktionen mit je einem Parameter darzustellen. Nach Teilanwendung der gecurryten Funktionen liefern diese als Rückgabe eine weitere Funktion, die entweder einen weiteren Parameter erwartet oder – falls alle Parameter vorliegen – das gewünschte Ergebnis.

Beispiel: Currying

Als einführendes Beispiel für das Currying dient hier die Funktion addiere : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, addiere$(x, y) = x + y$. Diese lässt sich umformen in eine Funktion addiere$'$: $\mathbb{N} \to \mathbb{N}$, die nach Anwenden auf den ersten Parameter eine weitere Funktion zurückgibt, die ihrerseits nach Anwenden auf den zweiten Parameter das gewünschte Ergebnis berechnen kann:

$$\text{addiere}'(c) = (\text{addiere}(y) = c + y)$$

.
Die gecurryte Funktion könnte folgendermaßen angewendet werden:
addiere$'(c) =:$ addiere$(3) = 3 + c$
addiere$'(5) = 3 + 5 = 8$

Currying erlaubt das Weglassen gleicher Parameter von Funktionen und Operatoren auf beiden Seiten einer Gleichung. Die Aussage $f(x, y) = g(x, y)$ lässt sich dadurch vereinfachen zu $f = g$. Außerdem lassen sich binäre Operatoren wie $+$ auch teilangewendet als Funktionen auffassen, was diese ebenfalls dem Currying zugänglich macht. Somit lassen sich der Schreibweise von [HZ22] folgend Operationen – auch nach Teilanwendung – als Funktion verstehen:

$$+(a, b) = (a+)(b) = (+b)(a) = a + b$$

Diese Erkenntnisse erlauben uns die Definition einer Funktion prefix, einer Funktion, die von einer Folge die ersten Elemente mit einer bestimmten Eigenschaft zurückgibt als

$$\text{prefix}\,(p) = \oplus \not\leftarrow$$

Hierbei sind:

- p ein Prädikat $p : l \to \{\top, \bot\}$ für ein beliebiges Objekt l [Bir86].
- Der Operator $\oplus : l \times l^n \to l^n$ definiert durch

$$a \oplus x = \begin{cases} \{a\} + x & \text{wenn } p(a) \text{ wahr} \\ \{\} & \text{sonst} \end{cases}$$

- Der Operator $\not\leftarrow$: $l^n \to l^n$ ein Operator, der die linksgerichtete Reduktion bezeichnet. Er erlaubt die Festlegung der Reihenfolge der Auswertung der reduzierten Folge von rechts

nach links:

$$\oplus \not\leftarrow \{a_1, a_2, a_3, \cdots, a_n\} = (a_1 \oplus (a_2 \oplus (a_3 \oplus (\cdots \oplus a_n))))$$

Beispiel: prefix

Sei F eine Folge ganzer Zahlen: $\{4, 1, -6, 3, 11, 3\}$ und p das Prädikat (≥ 0). Die Anwendung der prefix-Funktion auf diese Parameter ergibt:

$$\text{prefix}\,(p, F) = \{4, 1\}$$

KAPITEL 3

Repräsentation der Flächen durch Meshes in konstanter und linearer Zeit

In Abschnitt 2.4 haben wir den kontinuierlichen Charakter der Isoflächen herausgearbeitet. Ihrer Gestalt nach erfolgt eine möglichst präzise Darstellung durch Raytracing-Verfahren, die wir mitsamt den mit ihnen einhergehenden Nachteilen, besonders dem hohen Rechenaufwand, in Kapitel 1 besprochen haben. Im Laufe der Zeit entwickelten sich standardisierte Verfahren zum Zeichnen begrenzungsflächenrepräsentierter Geometrie, die zunächst in Softwarebibliotheken, später in durch ebensolche Bibliotheken steuerbaren Realisierungen durch Graphikprozessoren vorzufinden waren. Auf moderner Graphik-Hardware, auch auf mobiler, finden sich heutzutage Instanzen des *Rendering-Pipeline*-Modells, die durch Programmbibliotheken wie OpenGL oder DirectX angesprochen werden können [Eis+16, S. 303]. Diese Darstellungsarten ermöglichen die Anwendung von Beleuchtungsmodellen, die eine schnellere Berechnung eines zweidimensionalen, anzuzeigenden Bildes eines dreidimensionalen Modells zulassen, da in diesem Fall weniger zeitkomplexe Algorithmen anwendbar sind, als das oben besprochene Raytracing [Gam21, S. 30ff], aber in Kombination vergleichbare Ergebnisse liefern [Gam21, S. 4]. Für den Fall, dass ein dreidimensionales Modell unter Anwendung dieser Methoden gezeichnet werden soll, aber nicht im Begrenzungsflächenmodell vorliegt, ist eine Konvertierung erforderlich. Solange eine Isomorphie zwischen der vorliegenden Repräsentation und dem Begrenzungsflächenmodell nachgewiesen werden kann, erfolgt die beschleunigte Berechnung sogar verlustfrei, denn die Umkehrabbildung gewährleistet ihrerseits die Rückführung in die ursprüngliche Form. Konstruktionsbäume, die keine Primitiva mit Rundungen beinhalten, geben Beispiele für zu ihren Repräsentationen im Begrenzungsflächenmodell isomorphe 3D-Modelle. Die intensive Forschungsarbeit, auf deren Schultern diese Mechanismen ruhen, motiviert eine ausführliche Untersuchung von Möglichkeiten, die Konzeption der Isoflächen aus Abschnitt 3.2 diesen zugänglich zu machen.

Anders als Isoflächen bestehen begrenzungsflächenrepräsentierte *Meshes* aus einer endlichen Anzahl von Vertices, die für dieses Kapitel vereinfacht als Punkte verstanden werden

können. Dieses Kapitel befasst sich mit Möglichkeiten, die Position dieser Punkte zu bestimmen, so dass die ihnen übergeordnete Struktur eine Fläche bestmöglich repräsentiert. Kann diese Position nicht im Voraus bestimmt werden, müssen sie fehlerbehaftet angenommen und ihre Position unter Zuhilfenahme neuer Informationen korrigiert werden. Die einzigen infrage kommenden Informationen bestehen hier in der Position des Vertex, der Punktladung und dem Feld an der Position des Vertex, so dass dieser entweder in Richtung des Feldes oder auf einer Halbgeraden bewegt werden kann, die Anhand Gleichung (2.1) mit dem Aufpunkt der Punktladung und der Richtung zum Vertex konstruiert ist. Erstere Möglichkeit ist innerhalb einer Simulation möglich, wie wir sie später beleuchten werden. Die folgenden Abschnitte untersuchen die Eignung linearer Trajektorien zur Beschreibung der Bewegung der Messpunkte und beleuchten die Konsequenzen für eine solche Entscheidung.

3.1 Anpassen der Vertex-Positionen durch lineare Trajektorien

Ein großer Teil dieser Arbeit befasst sich mit der Aufgabe, die Repräsentation einer Fläche durch ein Mesh zu verbessern, indem die Position der Vertices der Fläche angenähert wird, falls das Mesh die gewünschten Kriterien der Annäherung noch nicht erfüllt. Da die Geometrie der Fläche nicht bekannt ist, steht als einziges Maß für die Qualität der Annäherung die Differenz des Isowertes zur Feldstärke an der Position des Vertex zur Verfügung, die berechenbar ist. Um eine neue Position des Vertex zu bestimmen, der zu einer Verbesserung der Annäherung führt, können wir hier nur auf diese Feldstärke und die Positionen des Vertex und der Punktladungen zurückgreifen. Diese Informationen genügen, um durch Gleichung (2.1) eine lineare Trajektorie für jedes Vertex $\mathbf{v}$ zu konstruieren. Diese wird zwei oder mehr Schnittpunkte mit der anzunähernden Fläche aufweisen, sofern ein Vertex einer Punktladung q zuzuordnen oder nur eine solche vorhanden ist: $\mathbf{s}(x) = \mathbf{q}_p + x\mathbf{v}$. Ist im Voraus bekannt, dass die anzunähernde Fläche mehrere Punktladungen umfasst, kann auch ein Hilfspunkt $\mathbf{m}$ wie ein Mittelpunkt zu Hilfe gezogen werden. Abbildung 3.1 zeigt beispielhaft die Möglichkeiten, aus den gegebenen Informationen lineare Trajektorien zu erzeugen.

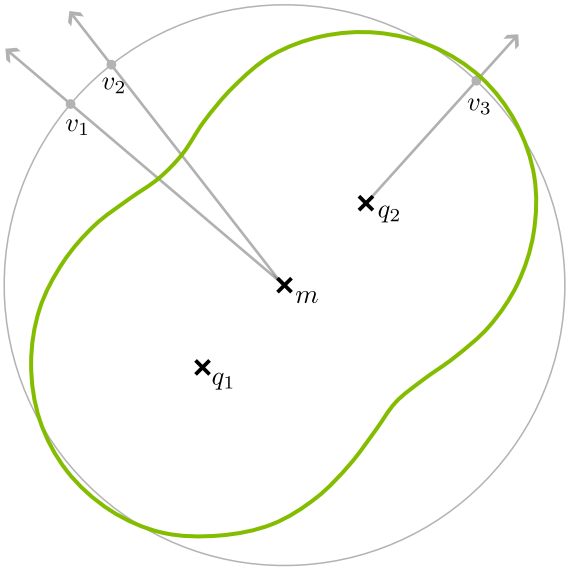


Abbildung 3.1: Das graue Mesh mit Vertices v_1 und v_2 soll die grüne Fläche annähern. Eine Halbgerade ist an eine Punktladung orientiert, eine am Hilfspunkt.

3.2 Arbitrarität und Konkavität in Isoflächen

Ein durch Isoflächen dargestelltes Feld ist für Menschen intuitiv verständlich: Durch vergleichsweise niedrige Informationsdichte verdeutlicht es dem Betrachter Bereiche höherer und geringerer Feldstärken durch größere Entfernung der Punkte auf den Flächen zur Ladung. Bereiche unterhalb des Isowertes sind auch bei disjunkten Flächen sofort erkennbar. Die Richtung des Feldes ist punktweise senkrecht zu den Flächen. Aufgrund dieser Plastizität verführen Isoflächen zu Fehlannahmen hinsichtlich ihrer Erzeugung. Beispielsweise sind die Punkte geringster Feldstärke zwischen den Isoflächen mit dem Auge sehr leicht auszumachen – sie liegen mittig zwischen den Grenzflächen. Dieser Punkt liegt aber nicht etwa auch immer mittig zwischen den Ladungen. Es ist nicht einmal ungewöhnlich, wenn die Feldstärke in diesem Punkt sogar größer als der Isowert ist. Andernfalls könnte das Problem der Suche nach Disjunktionen in der Isofläche mit konstanter Laufzeit gelöst werden. Beispiele wie diese zeigen, dass die Isoflächen, die in ihnen enthaltenen oder nicht enthaltenen und die gemeinsam oder nicht gemeinsam in einer Fläche enthaltenen Ladungen als arbiträr anzusehen sind. Dieser Erkenntnis nachzugehen und Trugschlüsse bezüglich der Gestalt der Flächen aufzudecken ist das Ziel dieses Abschnitts. Die folgenden Ausführungen sprechen nacheinander die Aussicht darauf, Punkte unterhalb des Isowertes zu berechnen, die Aufschluss auf die Gestalt der Isofläche geben. Daraufhin wird der Versuch angestellt, durch algebraische Berechnungen den Schnittpunkt der Bewegungshalbgeraden mit der Isofläche zu finden. Zuletzt wird untersucht, wie aussichtsreich der Versuch ist,

aus gegebenen Isoflächen durch das Superpositionsprinzip die Gestalt einer neuen Isofläche zu berechnen.

Ebenso wie die Wahl einer der Isofläche annähernden Oberfläche nicht beliebig gewählt werden kann, kann auch nicht ohne weiteres bestimmt werden, ob eine Menge von Punktladungen Q paarweise in unterschiedlichen Clustern liegt. Im Folgenden werden einige naheliegende Trugschlüsse des Aufstellens der Äquivalenzklassen für Punktladungen falsifiziert.

3.2.1 Punkte unterhalb des Isowertes zwischen den Punktladungen

Der Versuch, das Clustering durch Suchen von Punkten zwischen Punktladungen, etwa auf einer durch die Position dieser Punktladung begrenzten Strecke, um vom Feld an diesen Punkten auf die Äquivalenzklassen Rückschlüsse ziehen zu können, birgt Schwierigkeiten: Ihm liegt die Beobachtung zugrunde, dass zwischen den Flächen von Clustern stets mindestens ein Punkt existiert, der auf dieser Strecke liegt, für den $\mathbf{E}(\mathbf{p}) < \xi$ gilt. Isoflächen sind aber nicht notwendigerweise konvex, weswegen, auch mit Hinweis auf Punkt 6 der obigen Bemerkungen, die Existenz eines solchen Punkts kein Notwendiges Kriterium für den Schluss darstellt, dass die beiden Punktladungen nicht äquivalent sind. In Situationen mit schnell bewegten Punktladungen, die ein heuristisches Clustering erlauben, kann diese aber im Sinne einer „Daumenregel“ erfolgreich genug sein. Umso bessere Ergebnisse sind zu erwarten, je näher sich die Ladungen in einem Cluster im Vergleich zu den Ladungen in anderen Clustern befinden: Flächen, die durch vergleichsweise große ξ entstehen, nähern sich in ihrer Form Kugeln an, tendieren also zur Konvexität, wobei hingegen bei vergleichsweise kleinem ξ die Cluster klein werden und Paare von Punktladungen, die vergleichsweise weit voneinander entfernt liegen, wider Erwarten nicht die gleiche Äquivalenzklassen repräsentieren. In solchen Sonderfällen liegt es jedoch näher, die Entfernung der Punktladungen als alleiniges heuristisches Kriterium heranzuziehen. Ebenso ist zu beachten, dass für ein vergleichsweise großes ξ keine ausgezeichneten Punkte wie etwa der Mittelpunkt der Strecke als Ansatzpunkt verwendet werden können. Um dies zu verdeutlichen, betrachten wir die einfache Ladungsverteilung in Abb. 3.4. Hier soll festgestellt werden, dass $q_1 \not\sim q_2$. Daher sollte es auf der Strecke $\mathbf{g} = \overline{q_{1,\text{pos}}, q_{q,\text{pos}}}$ zwischen diesen beiden Punktladungen einen nicht ausgezeichneten Punkt $\mathbf{p}$ mit $\|\mathbf{E}(p) < \xi\|$ geben. Die Strecke $g(t)$ formulieren wir in der parametrischen Form:

$$g(t) = q_{0,\text{pos}} \cdot (1 - t) + q_{1,\text{pos}} \cdot t, \quad t \in [0, 1] \tag{3.1}$$

wobei sich $\mathbf{p}$ nun durch $t \in [0, 1]$ berechnet. Diese Formulierung erlaubt nun eine zweidimensionale Betrachtung des Problems. Eine beispielhafte Darstellung findet sich in Abbildung 3.2. Die Feldstärke zu jedem t erhalten wir durch Einsetzen von Gleichung (3.4) in Gleichung (3.1) mit dem Ergebnis einer Funktion $\mathbf{E}_g : [0, 1] \to \mathbb{R}^3$

$$\mathbf{E}_g(t) = \sum_{q \in Q} \frac{1}{(g(t) - q_{\text{pos}})^2} \tag{3.2}$$

Für $t = 0{,}5$ beispielsweise, also die Mitte der Strecke, können wir nicht voraussetzen, dass $\mathbf{E}_g(t)$ den Isowert unterschreitet, obwohl die Punktladungen nicht äquivalent sind. Lokale Suchverfahren können hier bestenfalls zur Verifizierung, keinesfalls aber zur Falsifizierung herangezogen werden. Da das elektrische Feld ein Kontinuum ist, ist auch $\mathbf{E}_g(t)$ auf dem gesamten Intervall $[0, 1]$ definiert; Außerdem können wir uns durch Betrachten der Gleichung (3.2) davon überzeugen, dass der Wertebereich von $\mathbf{E}_g$ im Intervall $[o, \infty]$ mit einem reellen $o < \xi$ liegt, wenn $q_0 \not\sim q_2$. $\mathbf{E}_g(t)$ hängt von $q_{3,\text{q}}$ ab, das beliebig in $\mathbb{R}$ liegen kann. Obwohl die Existenz des Punkts sicher ist, können wir für jedes t ein $q_{3,\text{q}}$ so wählen, dass $\mathbf{E}_g(t)$ größer, gleich oder kleiner als ξ ausfällt. Der Punkt kann also auf diese Weise nur durch Zufall gefunden werden.

Einen Anhaltspunkt für eine Abschätzung kann die Richtung des Feldes geben: Grenzwerte von t nah bei 0 und 1 ergeben antiparallele Feldvektoren, so dass es aussichtsreich sein kann, den Grenzwert

$$\lim_{a \to 0} \lim_{b \to 1} \mathbf{E}_g(a) \bullet \mathbf{E}_g(b)$$

aufgrund der Kontinuität des elektrischen Feldes als Entscheidungskriterium für das Clustering zu verwenden. Ebenfalls kann die Richtung des Feldes das Suchverfahren beschleunigen. Eine qualitative Darstellung dieses Terms, bei dem $a \approx 0$ fixiert und b von 0 an 1 angenähert wird, ist in Abb. 3.2 dargestellt: In der obigen beispielhaften Anordnung sind die Feldstärken für kleine b parallel, für große b antiparallel. Dieser Befund lässt eine Nicht-Äquivalenz der Punktladungen vermuten. Eine Äquivalenz liegt nahe, wenn der Verlauf der orange Kurve in Abbildung 3.2 annähernd konstant ist. Eine Verteilung aus zwei Punktladungen ist so nicht zu beurteilen.

3.2.2 Sonderfälle

An späterer Stelle in dieser Arbeit führen wir ein Verfahren zur exakten Bestimmung durch sukzessive Annäherung eines Meshes an die Isofläche ein. Teil dieses Verfahrens ist, bewegliche Messpunkte in die Szenerie einzuführen, die sich abhängig von der Stärke des Feldes linear auf je einer Punktladung q zu- oder von ihnen wegbewegen. Ihre Position kann dabei in Flächen von Clustern liegen, die q nicht repräsentiert. Darauf aufbauend kann versucht werden, den Übergang eines Messpunkts von einem Bereich unterhalb des Isowerts in eine Fläche festzustellen.

Vor Beginn der Iteration kann festgestellt werden, ob sich ein bestimmter Messpunkt bei einer linearen Bewegung auf einer Halbgeraden, auf der q_1 liegt, einer anderen Punktladung q_2 aus einer Ladungsverteilung wie in Abb. 3.3 annähert. Für den allgemeinen Fall kann dieses

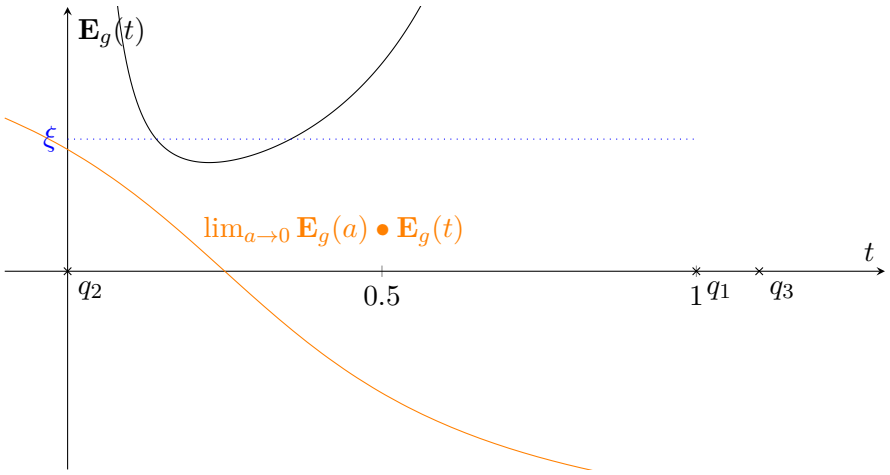


Abbildung 3.2: Qualitative Darstellungen der Feldstärke (schwarz) und des Verlaufs der Richtung (orange) in der Anordnung aus Abbildung 3.4 auf der Strecke $\overline{q_{1,\text{pos}}q_{2,\text{pos}}}$. $\mathbf{E}_g(0{,}5) > \xi$ kann nicht $q_1 \sim q_2$ verifizieren.

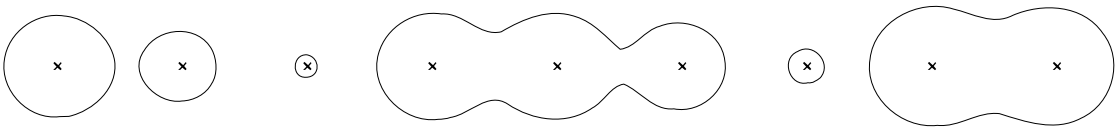


Abbildung 3.3: Unter bestimmten Bedingungen ist das Clustering durch Messen der Feldstärke zwischen den Ladungen möglich. In diesem Fall sind die Punktladungen kollinear und äquidistant, weisen aber unterschiedliche Stärke auf.

Verfahren nicht angewendet werden. Unter folgenden Bedingungen ist eine Zuordnung von Messpunkten in eine gegebene oder keine Fläche möglich:

- Für beliebige, paarweise unterschiedliche $q_1, q_2 \in Q$ gilt: $q_1 \approx q_2$,
- Die Anordnung der Punktladungen ist annähernd kollinear,
- Die Ladungen sind annähernd homogen oder ihre Anordnung annähernd äquidistant,
- Der Isowert ist sehr klein oder
- Der Qualität der Ergebnisse wird ihre schnelle Verfügbarkeit vorgezogen

Die hier verwendeten Begriffe sind qualitativ zu verstehen: Eine höhere Ungleichheit zwischen den Ladungsstärken, eine größere Störung der Kollinearität etc. führt zu einer Verschlechterung der Ergebnisse. Der Grad der Verschlechterung ist von den jeweiligen Parametern der betrachteten Ladungsverteilung abhängig. Als Sammelbegriff für diese Bedingungen kann hier das Wort *makroskopisch* herhalten, der ebenso qualitativ zu verstehen ist.

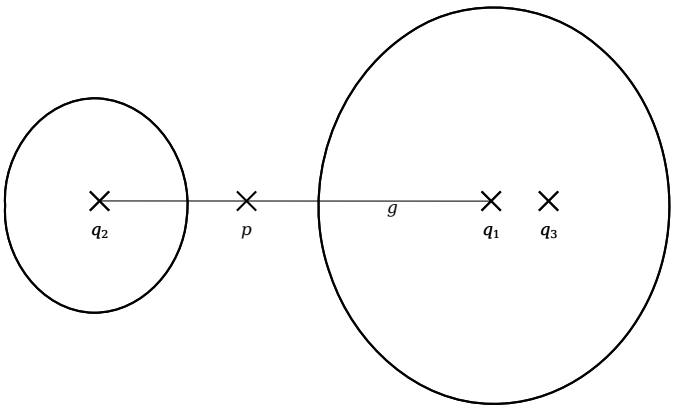


Abbildung 3.4: Punkte $\mathbf{p}$ mit $\|\mathbf{E}(p)\| < \xi$ existieren zwar, können aber nicht lokalisiert werden

3.2.3 Artefakte

Die Isofläche einer einzelnen Punktladung ist stets eine Kugel. Liegt eine Konfiguration mit mehreren Punktladungen vor, können die Flächen konkave Bereiche aufweisen. Diese können in Simulationen, die sich auf den in Abschnitt 3.2.1 besprochenen Ansatz einer Halbgerade zur Bestimmung der Trajektorie der Vertices stützt, zu Problemen führen. Zunächst beinhalten diese Stellen eine Vergrößerung des Radius und führen somit zu einer geringeren dichte von Vertices der Isofläche. Weiterhin können Halbgerade von einem Vertex und der Punktladung das Mesh an weiteren Stellen schneiden. Diese Probleme manifestieren sich in *Artefakten* – Bereichen, in denen das Mesh die Fläche fehlerhaft wiedergibt, indem beispielsweise Konkavitäten und Konvexitäten nicht erkennbar sind. Mit diesen Phänomenen ist zu rechnen, wenn keine Maßnahmen angestrengt werden, auf erkannte Fehlannahmen der Nicht-Äquivalenz zweier Punktladungen zu reagieren. Das Auftreten dieser Artefakte ist nicht erkennbar, jedoch können das Auftreten von Gebieten mit stark unterschiedlicher Auflösung einen Anhaltspunkt (vgl. Abschnitt 3.3.13.2) geben.

Die Motivation, Artefakte zu verhindern gibt zusätzlichen Anlass, Möglichkeiten zur Modifikation von Meshes zu untersuchen. Beim Zusammenfügen von Meshes können die vorangehend beschriebenen Situationen nicht auftreten, da nach dieser Operation die Vertices beider Meshes auf Halbgeraden mit den jeweils eigenen Ladungen bewegt werden: Konvexe Bereiche in Flächen entstehen durch den Einfluss mehrerer Punktladungen. Durch das Zusammenfügen zweier Meshes, die vorher jeweils ein Cluster repräsentierten, entsteht jeweils ein konkaver Bereich auf dem neuen Mesh. Damit würde anschaulich gesprochen eine Disjunktion der Isofläche an den konkaven Stellen der Fläche entstehen, wenn sich die beiden Cluster auseinander bewegten. Somit können keine konkaven Bereiche innerhalb der zusammengefügten Bereiche entstehen.

3.2.4 Bestimmung der neuen Vertex-Position durch algebraische Berechnung

Im Verlauf dieser Arbeit finden Untersuchungen hauptsächlich an Gleichung (2.4) und Gleichung (2.5) statt, die in geschlossener Form vorliegen, so dass anhand durch den Ring $(\mathbb{R}, +, \cdot)$ ermöglichter Umformung von Gleichungen argumentiert werden kann. Außerdem haben wir in Abschnitt 3.1 die Aufgabe, Flächen durch Meshes zu repräsentieren, auf die Suche eines Schnittpunktes der Bewegungshalbgerade der Vertices mit der Fläche selbst zurückgeführt. Dieser Umstand wirft die Frage nach der algebraischen Bestimmbarkeit der Punktmenge auf einer Isofläche von $\mathbf{E}$ durch iteratives Hinzufügen von Punktladungen auf. Diesem Ansatz gehen wir vereinfachend in skalarer Form nach, um aufzuzeigen, dass das Problem selbst in dieser Form nicht lösbar ist. Die Gleichung (2.4) lautet in skalarer Form:

$$E(x) = \sum_{i \in \mathcal{Q}} \frac{q_i}{(x - x_i)^2} = \xi \tag{3.3}$$

wobei ξ weiterhin den Isowert darstellt, für den ein Funktionswert in x_0 gesucht wird und $\mathcal{Q}$ die Folge der Punktladungen $\{(x_1, q_1), (x_2, q_2), \ldots\}$. Gleichung (3.3) hat für eine einwertige Punktladungsmenge $\mathcal{Q} = \{(x_0, q_0)\}$ eine Lösung in x mit

$$x = x_0 \pm \frac{q_0}{\sqrt{\xi}}$$

x_0 kann somit als Punkt auf einer eindimensionalen Isofläche des Feldes E angesehen werden. Eine beispielhafte Anordnung mit einer Punktladung im Ursprung ist in Abbildung 3.5a illustriert. Ausgehend von diesem Feld für eine Punktladung wollen wir im Folgenden feststellen, ob sich aus der gegebenen Position x_0 eines Vertex auf der Icosphere mit $\mathbf{E}(x_0) = \xi$ eine neue Position x' mit $\mathbf{E}'(x') = \xi$ als dem Feld nach Einbringen einer weiteren Punktladung q zu berechnen ist. Gesucht ist also nach einer Beziehung zwischen der faktoriellen Darstellung der Gleichung mit einer Ladung, die sich leicht umstellen lässt und der Summe nach dem Hinzufügen der neuen Ladung, die als Polynom gelöst werden kann. Diese Argumentation ließe sich dann auf Anordnungen Punktladungsmengen $\mathcal{Q}$ beliebiger Kardinalität übertragen.

Für die Bestimmung der Position auf der x-Achse bzw. ihre entsprechende Ladungsstärke der Punktladungen $q \in \mathcal{Q}$ definieren wir die beiden Projektionen $()_{\text{pos}} : Q \to \mathbb{R}$ bzw $()_{\text{q}} : Q \to \mathbb{R}$. Bei Vorliegen eines Feldes einer einwertigen Punktladungsmenge, in das eine zusätzliche Ladung eingebracht werden soll, berechnen wir das Feld also durch

$$\left(\sum_{i \in \mathcal{Q}} \frac{1}{\left(x_0 - i_{\text{pos}}\right)^2} \right) = \xi \tag{3.4}$$

und nach Einbringen einer neuen Ladung w mit

$$\left(\sum_{i \in \mathcal{Q}} \frac{1}{\left(x' - i_{\text{pos}}\right)^2} \right) + \frac{1}{\left(x' - w_{\text{pos}}\right)^2} = \xi \tag{3.5}$$

In Gleichung (3.5) steht nun x' für die Position eines Punkts auf der neu entstandenen Isofläche. Da aus den gegebenen Größen in Gleichung (3.4) auf diese neue Position geschlossen werden soll, müssen die beiden Terme, die nach Vorgabe den Wert ξ haben, gleichgesetzt werden.

$$\Leftrightarrow \left(\sum_{i \in \mathcal{Q}} \frac{1}{\left(x_0 - i_{\text{pos}}\right)^2} \right) = \left(\sum_{i \in \mathcal{Q}} \frac{1}{\left(x' - i_{\text{pos}}\right)^2} \right) + \frac{1}{\left(x' - w_{\text{pos}}\right)^2}$$

Das Ausschreiben des Summenterms ermöglicht dann das sukzessive Hinzufügen weiterer Summanden:

$$\left(\sum_{i \in \mathcal{Q}} \frac{1}{\left(x' - i_{\text{pos}}\right)^2} \right) = \frac{1}{\left(x' - q_{0,\text{pos}}\right)^2} + \frac{1}{\left(x' - q_{1,\text{pos}}\right)^2} + \cdots + \frac{1}{\left(x' - w_{\text{pos}}\right)^2}$$

$$= \frac{\left(x' - q_{0,\text{pos}}\right)^2 + \left(x' - q_{1,\text{pos}}\right)^2}{\left(x' - q_{1,\text{pos}}\right)^2 \left(x' - q_{2,\text{pos}}\right)^2} + \frac{1}{\left(x' - q_{3,\text{pos}}\right)^2} + \cdots + \frac{1}{\left(x' - w_{\text{pos}}\right)^2}$$

$$= \frac{\left(x' - q_{2,\text{pos}}\right)^2 \left(\left(x' - q_{0,\text{pos}}\right)^2 + \left(x' - q_{1,\text{pos}}\right)^2 \right) + \left(x' - q_{1,\text{pos}}\right)^2 \left(x' - q_{2,\text{pos}}\right)^2 \left(x' - q_{3,\text{pos}}\right)^2}{\left(x' - q_{1,\text{pos}}\right)^2 \left(x' - q_{2,\text{pos}}\right)^2 \left(x' - q_{3,\text{pos}}\right)^2} + \frac{1}{\left(x' - q_{4,\text{pos}}\right)^2} + \cdots + \frac{1}{\left(x' - w_{\text{pos}}\right)^2}$$

Der Hauptnenner der Summenterme bildet ein Produkt, das eine zusammengefasste Schreibweise durch das Produktzeichen ermöglicht. Den Zähler substituieren wir durch eine Funktion $r : \mathbb{N} \to \mathbb{R}$. Für $k+1$ Terme erhalten wir:

$$\frac{r(k)}{\prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2} + \frac{1}{\left(x' - w_{\text{pos}}\right)^2} - \sum_{i=0}^{k} \frac{1}{\left(x_0 - i_{\text{pos}}\right)^2} = 0 \tag{3.6}$$

Um weiter nach x' auflösen zu können, muss der Kehrwert der linken Seite der Gleichung gebildet werden:

$$\frac{\left(x' - w_{\text{pos}}\right)^2 r(k) + \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2 - \left(x' - w_{\text{pos}}\right)^2 \sum_{i=0}^{k} \frac{1}{\left(x_0 - i_{\text{pos}}\right)^2} \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2}{\left(x' - w_{\text{pos}}\right)^2 \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2} = 0$$

Der Nenner entfällt bei der weiteren Bestimmung der Lösungen, so dass sich die Suche im Folgenden auf die Lösungen des Zählers beschränken kann:

$$\Leftrightarrow \left(x' - w_{\text{pos}}\right)^2 r(k) + \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2 - \left(x' - w_{\text{pos}}\right)^2 \sum_{i=0}^{k} \frac{1}{\left(x_0 - i_{\text{pos}}\right)^2} \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2 = 0 \tag{3.7}$$

Die Funktion r in Gleichung (3.6) offenbart eine rekursive Struktur, die wir zunächst in eine Fallunterscheidung umwandeln:

$$r(n) = \begin{cases} \left(x' - q_{n,\text{pos}}\right)^2 \left(r(n-1) + \prod_{i=0}^{n-1} \left(x' - q_{i,\text{pos}}\right)^2 \right) + \prod_{i=0}^{n} \left(x' - q_{i,\text{pos}}\right)^2 & \text{für } n > 1 \\ \left(x' - q_{0,\text{pos}}\right)^2 + \left(x' - q_{1,\text{pos}}\right)^2 & \text{für } n = 1 \end{cases}$$

Nach Auflösen der Fallunterscheidung und Ausmultiplizieren der Summanden ergibt sich:

$$r(n) = \sum_{k=0}^{n} \prod_{n_1=0}^{i-1} \left(x' - q_{n_1,\text{pos}}\right)^2 \prod_{n_2=i+1}^{n} \left(x' - q_{n_2,\text{pos}}\right)^2 \tag{3.8}$$

Der Term für $k+1$ Ladungen lautet also, nach Einsetzen von Gleichung (3.8) in Gleichung (3.7):

$$\Leftrightarrow \left(x' - w_{\text{pos}}\right)^2 \left(\sum_{k=0}^{k} \prod_{k_1=0}^{i-1} \left(x' - q_{k_1,\text{pos}}\right)^2 \prod_{k_2=i+1}^{k} \left(x' - q_{k_2,\text{pos}}\right)^2 \right) + \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2 - \left(x' - w_{\text{pos}}\right)^2 \sum_{i=0}^{k} \frac{1}{\left(x_0 - i_{\text{pos}}\right)^2} \prod_{i=0}^{k} \left(x' - q_{i,\text{pos}}\right)^2 = 0$$

Die linke Seite der Gleichung liegt nun in polynomialer Form vor, die eine Bestimmung der Nullstellen ermöglicht. In dieser Form kann an der Gleichung abgelesen werden, dass hier ein Polynom des Grades $(k-1)+2$ zu lösen ist, desgleichen für zwei weitere Dimensionen. Für Polynome in höheren Graden existiert kein algebraisches Lösungsverfahren [Gre19, S. 20]. Für die Bestimmung der Methode muss also das elektrische Feld angenähert [Gre19, S.20] oder ein iteratives Verfahren wie es in Abschnitt 4.5 vorgestellt wird, angewendet werden [Gre19, S. 32]. Hilfreich kann sein, dass wir nicht alle Lösungen brauchen, sondern nur die kleinste reelle. Da aber k und somit das Polynom nicht bekannt sind, kann auch diese Position nicht eingegrenzt werden.

3.2.5 Bestimmung der neuen Vertex-Position durch einen Greedy-Algorithmus

In Abschnitt 2.4 zeigen wir auf, dass das elektrische Feld eine additive, vektorielle Überlagerung seiner Einzelfelder ist. Aus diesem Grund konnten alle Argumentationen in dieser Arbeit anhand der Gleichung (2.5) das Verschieben von Vertices auf einer Halbgeraden ebenfalls als Vektoraddition realisiert werden, jedoch drängt sich der Gedanke eines Zusammenhangs zwischen diesen beiden Phänomenen auf. Schlägt wie in Abschnitt 3.2.4 die Rückführung der Überlagerung auf ein zu lösendes Polynom fehl, kann dieser Zusammenhang auch abstrakter als Bewertungsfunktion eines Greedy-Algorithmus verstanden werden. Ein *Greedy-Algorithmus* ist ein Algorithmus, der in jedem Schritt den besten Folgeschritt berechnet. Die Bewertungsfunktion bemisst dabei die Güte der möglichen Schritte. Ein solcher Algorithmus müsste also die Form des Alg. 3 haben [Vö+08]. Dieser wäre, sollte er existieren, imstande, durch

- die berechneten Feldstärke eines Vertex, der sich im schlechtesten Fall noch auf der Startposition befindet, und
- die Bewertungsfunktion

die Verschiebung des Vertex um die korrekte Länge in Richtung der Halbgerade als „besten Schritt“ zu berechnen. Im Folgenden widerlegen wir die Existenz dieses Algorithmus.

Algorithmus 3 : Allgemeine Form eines Greedy-Algorithmus zur Berechnung des elektrischen Feldes .

eingabe : Elektrisches Feld, Isowert ξ, Vertex einer Isofläche v, Bewertungsfunktion f
ausgabe : Elektrisches Feld

1 Feld am Vertex $\mathbf{e}_v = \mathbf{E}(v)$;
2 Iso-Fehler $\varepsilon = \|\xi - e_v\|$;
3 Neue Position des Vertex: $\mathbf{v}' = \mathbf{v} + f(\varepsilon)$; `// Überlagerung`

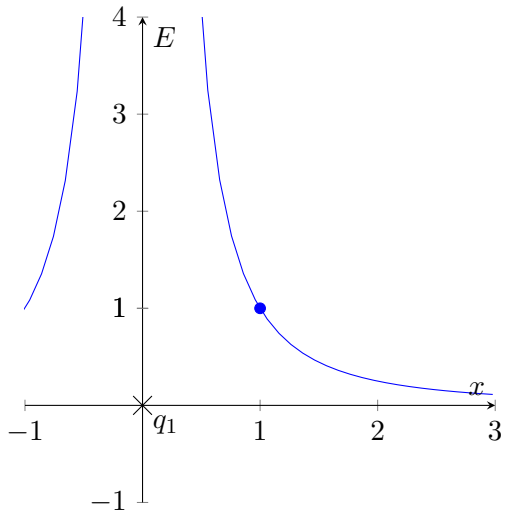
Um die Argumentation zu vereinfachen, reduzieren wir analog zu Abschnitt 3.2.4 die Gleichung (2.5) auf eindimensionale Probleme und eine einzige Punktladung. Auf diese Weise erhalten wir

$$E = \frac{1}{x^2}$$

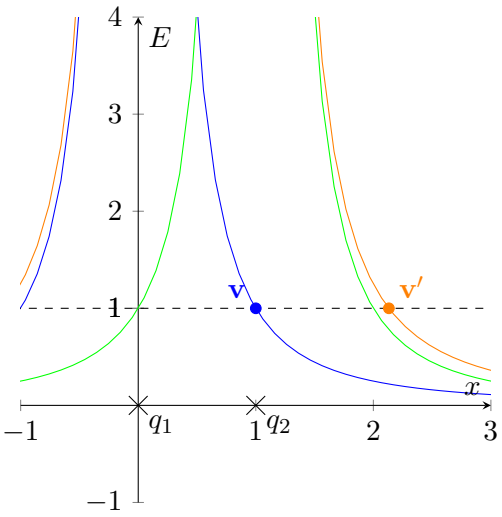
Die Beschränkung der Betrachtungen auf eindimensionale Probleme schränkt zwar die Allgemeinheit der Argumentation ein, die Existenz des Alg. 3 für eindimensionale Probleme ist aber ein notwendiges Kriterium für die Existenz dieses Algorithmus für drei Dimensionen. Das Feld wird für eine Punktladung an der Stelle $x = 0$ in Abbildung 3.5a dargestellt. Außerdem nehmen wir einen Isowert von $\xi = 1$ an. Dieser Wert wird im von q_1 erzeugten Feld bei $x = 1$ erreicht.

Selbst unter der obigen Annahme, dass auch für mehrere Punktladungen v' durch Überlagerung berechnet werden könnte, als gäbe es nur eine einzige Punktladung, kann ein Ansteigen des Fehlers ε nur durch Einbringen neuer Punktladungen verursacht werden.

Nach Einbringen der zweiten Punktladung muss die x-Position des überlagerten Feldes für $\xi = 1$ bestimmt werden. Bei einer Gleichung der Gestalt von Gleichung (2.5) sind vier Lösungen, davon zwei reelle zu erwarten. Da wir ein Gegenbeispiel konstruieren, dürfen wir die unphysikalischen Lösungen als bekannt annehmen und verwerfen. Es verbleibt eine physikalisch sinnvolle Lösung für die Position des neuen Vertex. Diese ist in Abbildung 3.6 angedeutet. Nach der obigen Argumentation muss dieses nun durch die Funktion $f(\varepsilon)$ bestimmt werden, so dass die neue Vertex-Position schließlich die Summe aus $f(\varepsilon)$ und der alten Vertex-Position ist. Anhand Abbildung 3.6 ist aber ebenfalls ersichtlich, dass kein Punkt bekannt ist, aus dessen Überlagerung mit v das Ergebnis $f(\varepsilon)$ zu berechnen ist. Dafür müsste dieser Punkt ein ausgezeichneter Punkt auf dem Kreis mit dem Radius $f(\varepsilon)$ mit dem Mittelpunkt v' sein.



(a) Feld einer Ladung in einer Dimension und Vertex einer Icosphere.



(b) Feld beider Ladungen (blau und grün) und überlagertes Feld (orange).

Abbildung 3.5: Suchen einer Lösung mittels des Greedy-Algorithmus.

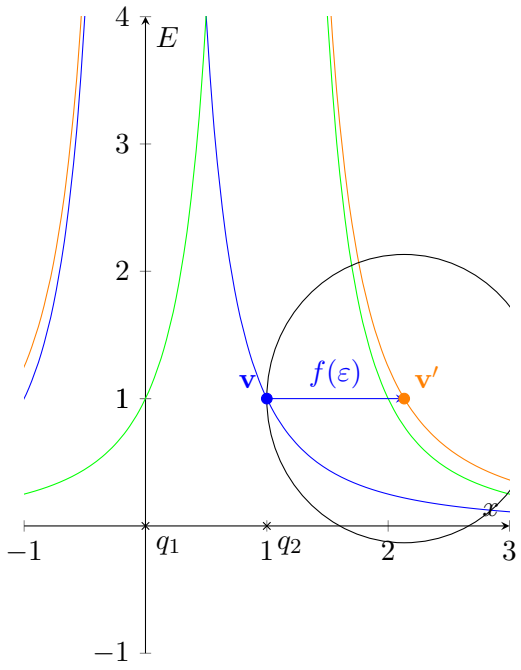


Abbildung 3.6: Der Kreis mit dem Radius der Lösung schneidet keine ausgezeichneten Punkte.

3.3 Operationen auf Meshes

Wegen der Arbitrarität der Cluster kann bei der Erstellung des Meshes, das die Isofläche repräsentiert, erst im Verlauf die Äquivalenz zweier Punktladungen ausgeschlossen oder zugesichert werden, wobei sich herausstellen kann, dass zuvor verwendete Heuristiken versagt haben. Ebenso können sich die Äquivalenzklassen bei dynamischen Ladungsanordnungen ändern. Backtracking-Verfahren, die sich als Mittel zur Fehlerkorrektur anbieten, können dann die Simulation so stark verlangsamen, dass die Eignung für die Anwendung in der virtuellen Realität nicht mehr in Frage steht. Dieser Umstand motiviert die Untersuchung von Mitteln, Fehler oder Änderungen in den Äquivalenzklassen zu erkennen und die entsprechende Korrektur durch geeignete Manipulation der Meshes durchzuführen.

Die folgenden Abschnitte besprechen weitere, allgemein anwendbare Möglichkeiten der Fehlererkennung und -korrektur. Die möglichen auftretenden Fehlersituationen beschränken sich dabei auf falsch angenommene Äquivalenz, in der ein Mesh eine Menge nicht äquivalenter Ladungen umschließt, und falsch angenommene Nicht-Äquivalenz, in äquivalente Ladungen durch Meshes voneinander trennen. Als mögliche Reaktionen auf beide Situationen diskutieren wir einerseits das „Wegschneiden“ als falsch erkannter und das „Hinzufügen“ als fehlend erkannter Geometrie, andererseits das vollständige und verzögerte („lazy“) Vorberechnen aller möglicherweise entstehenden Meshes und dem ein- bzw. ausblenden des passenden Meshes in Abhängigkeit von der erkannten Fehlersituation.

3.3.1 Fehlererkennung

Abhängig von den Randbedingungen der Simulation, des Anwendungsfalls, der Erstellung des Meshes, etc., können naheliegende Verfahren zur Fehlerfeststellung angewendet werden. Beispielsweise wird bei falsch angenommener Nicht-Äquivalenz eine Fläche durch zwei Meshes repräsentiert. Diese werden sich, besonders wenn Verfahren wie in Abschnitt 4.5 angewendet werden, überschneiden, was die Kriterien einer Isofläche verletzt. Somit ist ein positiver Punktbeinhaltungstest eines Vertex des einen Meshes mit dem anderen Mesh hinreichendes Kriterium für eine falsch angenommene Nicht-Äquivalenz. Die für zweidimensionale Probleme bekannteste Anwendung besteht darin, die Schnittpunkte eines Strahls mit beliebigen Winkeln mit den Kanten des zu testenden Polygons zu zählen. Ist diese Anzahl ungerade, beinhaltet das Polygon den Punkt. Dieses Vorgehen kann anhand Abbildung 3.7 nachvollzogen werden.

3.3.2 Polygon-Punktbeinhaltungstest

Der in [Moe03, Kap. 3.3, F. 5] hergeleitete Algorithmus kann nur auf geschlossenen Polygonen durchgeführt werden und ist nicht resistent gegen Fälle, in denen der zu testende Punkt mit einer Kante des Polygons kollinear ist. Obwohl solchen Fälle selten sind und auch wenn unter

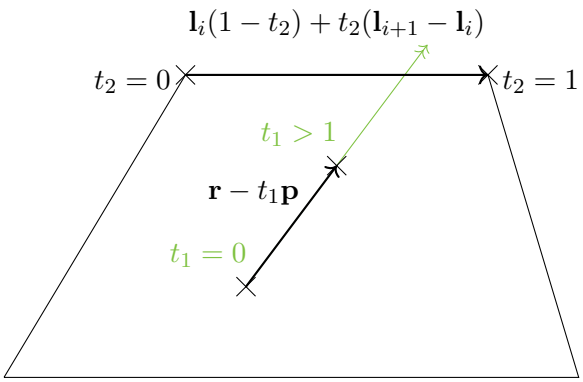


Abbildung 3.7: Polygon-Punktbeinhaltungstest mithilfe eines Strahls.

Umständen eine leichte Verschiebung des Punktes denkbar ist, um den Algorithmus mit diesem neuen Punkt erfolgreich durchzuführen, ist unter diesen Umständen dennoch eine Bestimmung der asymptotischen Laufzeit nicht möglich. [Hao18] stellt eine aktuelle Sammlung verschiedener Punktbeinhaltungstests aus der Literatur bzw. einen eigenen Algorithmus vor, die gegen kollineare Fälle verträglich sind und gibt die jeweiligen Zeitkomplexitätsklassen an bzw. berechnet diese. Die Laufzeitkomplexitäten übersteigen jedoch mit mindestens $\mathcal{O}(n \log n)$ die Laufzeit des Verfahrens aus [Moe03]. Wir wollen daher im Folgenden eine Möglichkeit herleiten, den kollinearen Sonderfällen durch Anpassen der Parametergrenzen des Streckenabschnitts und einer bedingten Komplementierung Rechnung zu tragen. Die Schnittpunktberechnung eines Strahls mit einer Kante $k \in K$ eines Polygons P möge durch die Funktion $f : E \to \{\bot, \top\}$ erfolgen. Hierbei bezeichnet E die Menge aller Kanten. Da Polygone Folgen sind, lassen sie sich in offene Teilpolygone

$$P = P_1 + P_2 + P_3 \tag{3.9}$$

aufteilen. Sowohl Folgen als auch der Boolesche Ring sind insbesondere Halbgruppen, was die Untersuchung von f auf die Eigenschaft eines Halbgruppenhomomorphismus ermöglicht. Laut dem Homomorphismen-Lemma aus [Bir86, S. 12] ist jede Funktion ein Homomorphismus, die sich als Abbildung (*map*) mit anschließender Reduktion ausdrücken lässt. Dieses Vorgehen nach der Wahl des Teststrahls ist zulässig: Zunächst erfolgt der Test des Strahls mit jeder Kante des Polygons durch die Funktion f, anschließend die Verknüpfung der Teilergebnisse zu einem Gesamtergebnis. In der Bird-Meertens-Notation kann dieser Vorgang als

$$h = (\oplus/) \circ (f\star)$$

formuliert werden. Diese Gleichung beschreibt die übliche Vorschrift für Homomorphismen: $f(P_1+P_2) = f(P_1) \oplus f(P_2)$ mit einer noch zu bestimmenden Verknüpfung $\oplus$. Dieser Operator muss die Eigenschaft haben, dass er – auch mehrfach angewendet – nach der Reduktion den Wert $\top$ genau dann annimmt, wenn in einem Teilpolygon der Strahl eine ungerade Anzahl von

P_1	P_2	$P_1 \oplus P_2$
$\top$	$\top$	$\bot$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$

Tabelle 3.1: Wahrheitstabelle für die Kontravalenz.

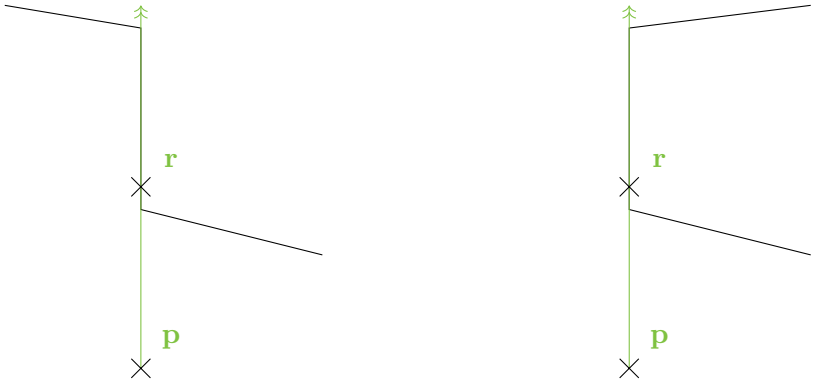


Abbildung 3.8: Teilpolygone mit je einer Kante, die auf dem Strahl verläuft.

Kanten schneidet. Ergibt in der Reduktion eine Funktion $f(P_k) = \top$, so komplementiert sie das bisher reduzierte Ergebnis, während das Ergebnis $\bot$ das Ergebnis nicht verändert. Dadurch ergibt sich die Wahrheitstabelle Tabelle 3.1. Anhand dieser kann nun $\oplus$ als Kontravalenz (exklusive Disjunktion) identifiziert werden. Durch diese Erkenntnis ist es uns nun möglich, die Argumentation für die Bearbeitung der Sonderfälle an dem offenen Teilpolygon P_2 aus Gleichung (3.9) unter Vernachlässigung der Teilpolygone P_1 und P_3 fortzuführen, ohne dabei die Allgemeinheit einzuschränken.

Tritt nun in einem Polygon $P = \{\mathbf{l}_1, \ldots, \mathbf{l}_{i-1}, \mathbf{l}_i, \mathbf{l}_{i+1}, \ldots\}$ eine Situation wie in Abbildung 3.8 auf, in der durch die Punkte $\mathbf{l}_i$ und $\mathbf{l}_{i+1}$ beschriebene Kante auf dem Strahl verläuft, muss der im linken Fall gezeigte als eine ungerade Anzahl von Schnitten (z. B. einer) und der im rechten Fall gezeigte als eine gerade Anzahl von Schnitten (z. B. null) gezählt werden. Da diese Zählung ebenfalls für mehrere aufeinander folgende auf dem Strahl verlaufende Kanten gilt, können wir ohne Einschränkung der Allgemeinheit annehmen, dass das untersuchte Teilpolygon nur aus den drei Kanten $\mathbf{l}_1$, $\mathbf{l}_2$ und $\mathbf{l}_3$ besteht. Die korrekte Zählung ermitteln wir durch Messen der Winkel zwischen $\mathbf{l}_1$ und $\mathbf{l}_3$: Ist $(\mathbf{l}_i - \mathbf{l}_{i-1}) \bullet (\mathbf{l}_{i+2} - \mathbf{l}_{i+1})$, wird ein Schnitt gezählt, ansonsten nicht. Eine weitere Behandlung der Kante erübrigt sich.

Für die Berechnung der Schnittpunkte durch Algorithmus 4 eignet sich eine Betrachtung in homogenen Koordinaten. Daher mögen für den Rest dieses Abschnitts Vektoren und Matrizen die homogene Eigenschaft verfügen. Die beiden Geraden für den Strahl und die zu prüfende Kante konstruieren wir in der Zweipunktform:

- den Strahl durch die Position des zu testenden Punkts $\mathbf{p}$ und einem beliebigen Punkt $\mathbf{r}$: $\mathbf{p} + t_1\mathbf{r}$
- die Kante durch die Position zweier aufeinanderfolgender Vertices: $\mathbf{l}_i(1-t_2)+t_2(\mathbf{l}_{i+1}-\mathbf{l}_i)$

mit reellen Parametern t_1 und t_2. Die Parameter für den Schnittpunkt ergeben sich durch Lösen des Gleichungssystems:

$$\mathbf{p} + t_1\mathbf{r} = \mathbf{l}_i(1 - t_2) + t_2(\mathbf{l}_{i+1} - \mathbf{l}_i) \tag{3.10}$$

Wenn der Schnittpunkt „in Richtung des Strahls“ ($t_1 > 0$) und im Streckenabschnitt der Kante liegt ($0 \leq t_2 \leq 1$), ist der Schnittpunkt gültig. Durch Anpassung des Parameterbereichs für t_2 zu

$$0 < t_2 \leq 1 \tag{3.11}$$

erreichen wir, dass nur ein Schnitt gezählt wird, falls der $\mathbf{p}$, $\mathbf{r}$ und ein Vertex kollinear sind. Andererseits genügt es, den Fall zu betrachten, dass $\mathbf{p}$, $\mathbf{r}$ und zwei im Polygon aufeinanderfolgende Vertices kollinear sind und somit unendlich viele Schnittpunkte vorlägen. In diesem Fall wäre Gleichungssystem 3.10 aufgrund einer Division durch Null nicht lösbar. In diesem Fall verfahren wir wie oben beschrieben.

Da dieser Test mit jeder Kante des Polygons genau ein Mal ausgeführt und durch entsprechende kollineare Fälle nicht wiederholt werden muss, beläuft sich der Aufwand dieses Algorithmus auf $\mathcal{O}(n)$, wobei n die Anzahl der Kanten des Polygons bezeichnet.

Dieses Prinzip lässt sich auf dreidimensionale Anwendungsfälle erweitern [Eis+16, S. 52], muss aber dahingehend abgeändert werden, dass der Schnitt des Strahls mit allen Faces des Meshes überprüft werden muss.

3.3.3 Polyeder-Punktbeinhaltungstest

Ein Polyeder-Punktbeinhaltungstest kann auf einfache Weise auf den zweidimensionalen Punktbeinhaltungstest zurückgeführt werden. Die Ebene $\mathbf{P}$ der einzelnen Polyederflächen ist in Parameterform bereits jeweils durch drei Punkte der Fläche gegeben. In Alg. 5 berechnet sich der Schnittpunktparameter dieser Ebene mit dem Strahl durch Lösen des Gleichungssystems

$$\mathbf{P} = \mathbf{r} - t_3\mathbf{p} \tag{3.12}$$

und der Überprüfung, ob mit $t_3 > 0$ der Schnittpunkt in Richtung der Halbgerade liegt. Nach Einsetzen von t_3 werden wir den Schnittpunkt $\mathbf{s}$ in der Ebene $\mathbf{P}$ vorfinden und haben somit nur noch einen Polygon-Punktbeinhaltungstest durchzuführen.

Algorithmus 4 : Punktbeinhaltungstest für zweidimensionale Polygone.

eingabe : Punkt $\mathbf{p}$, Polygon $P = \{\mathbf{l}_1, \mathbf{l}_2, \ldots \mathbf{l}_3\}$
ausgabe : Wahrheitswert w

$r \leftarrow \mathbf{1}$; // beliebig
$w \leftarrow \top$;
für *jeden Punkt* $\mathbf{l}_i \in L + \mathbf{l}_1$ **tue**
  $\mathbf{v} \leftarrow \mathbf{l}_{i+1} - t_2\mathbf{l}_i$; // Richtung der Kante
  wenn $\mathbf{r}_x + \mathbf{v}_x\mathbf{v}_y\mathbf{r}_y = 0 \vee \mathbf{v}_x\mathbf{r}_y - \mathbf{v}_y\mathbf{r}_x = 0$ **dann**
    wenn $(\mathbf{l}_i - \mathbf{l}_{i-1}) \bullet (\mathbf{l}_{i+2} - \mathbf{l}_{i+1})$ **dann**
      $w \leftarrow \bar{w} > 0$;
    Ende
    Abbruch des Schleifendurchlaufs
  Ende
  $(t_1, t_2) =$ Lösung des Gleichungssystems (3.10);
  wenn $t_1 > 0 \wedge t_2 > 0 \wedge t_2 \leq 1$ **dann**
    $w \leftarrow \bar{w}$
  Ende
Ende

Algorithmus 5 : Punktbeinhaltungstest für Polyeder.

eingabe : Punkt $\mathbf{p}$, Polyeder $L = \{P_1, P_2, \ldots P_3\}$
ausgabe : Wahrheitswert b

$r \leftarrow \mathbf{1}$; // beliebig
$t_3 \leftarrow$ Lösung des Gleichungssystems (3.12);
für *alle Polygone* $P_i \in L$ **tue**
  $\mathbf{s} =$ Schnittpunkt des Strahls $\mathbf{p} + t_3\mathbf{r}$;
  wenn *Polygon* P_i *den Punkt* $\mathbf{s}$ *beinhaltet* **dann**
    $w \leftarrow \bar{w}$
  Ende
Ende

Anhand Alg. 5 erkennen wir, dass dieser ebenfalls eine Laufzeitkomplexität $\mathcal{O}(nm)$ aufweist, wobei m hier für die Anzahl der Polygone steht. Der Faktor n begründet sich aus der $\mathcal{O}(m)$-mal durchgeführten Polygon-Beinhaltungstests aus Algorithmus 4. Da die Aufgabe darin besteht, zu testen, ob die Polyeder sich gegenseitig überschneiden, muss dieser Test l-mal durchgeführt werden, wenn das andere Polygon aus l Vertices besteht, wodurch sich der Aufwand auf $\mathcal{O}(lmn) \in \mathcal{O}(n^3)$ erhöht. Die Komplexität kann durch folgende Mittel herabgesetzt werden:

- Wenn während einer Simulation wie in Abschnitt 4.5 bereits eine Iteration über alle Vertices eines Meshes durchgeführt wird, können die in Schleife durchgeführten Anweisungen in Alg. 5 hinzugefügt werden, statt die Zeitkomplexität eines einzelnen Schleifendurchlaufs mit dem Faktor
  $O(n^2)$ zu beaufschlagen.

- Für einen Polyeder mit einer begrenzten Kantenanzahl für Flächen läuft Algorithmus 4 mit linearer Zeit.

- Kann in einer Simulation von Geometrien ausgegangen werden, die deutlich größer sind, als die größte Strecke, um die die Vertices während einer Bildwiederholung verschoben werden können, ist ein zeitweises Versagen der Erkennung falsch angenommener Nicht-Äquivalenzen tolerabel. Die Polyeder-Beinhaltungstests können somit auf eine feste Anzahl von zufälliger Vertices eingeschränkt werden.

Können alle drei Punkte Anwendung finden, läuft das Verfahren in linearer Zeit.

3.3.3.1 Heuristiken zur Erkennung falsch angenommener Nicht-Äquivalenz

Bei der Anwendung schnellerer Methoden zur Erkennung falsch angenommener Nicht-Äquivalenz kommt die Problematik der Arbitrarität der Cluster (s. Abschnitt 3.2) zum Tragen, allerdings können im Falle einer Simulation zusätzliche Informationen aus den vorangegangenen Simulationsschritten zur Verfügung stehen. Einige mögliche Vorgehensweisen, diese zu nutzen, werden wir im Folgenden kurz darlegen.

Beispielsweise können bewegliche Messpunkte in der Datenstruktur um eine Bewegungshalbgerade entlang einer zugeordneten Punktladung und eine Kennzeichnung, ob sich der Messpunkt im letzten Durchgang außer- oder innerhalb einer Fläche aufgehalten hat, erweitert werden. Für annähernd homogene Ladungsanordnungen kann aus diesen Informationen ein Kriterium für falsch angenommene Nicht-Äquivalenz abgeleitet werden, wenn der Abstand zu einer anderen Punktladung als der zugeordneten geringer ist, oder durch Änderung der Kennzeichnung ein Übergang in eine Fläche erkannt wird. Steht eine Fläche der übertretenen Grenze im Zusammenhang mit einer Ladung in einer Situation wie in Abbildung 3.9, konnte dessen

Äquivalenz mit der zugeordneten Ladung in vorhergehenden Simulationsschritten zugesichert werden.

Das Ergebnis einer einfachen Berechnung des Abstands zwischen Messpunkt und zwei Graden kann keinen Hinweis auf eine Äquivalenz liefern. Kann aber festgestellt werden, dass in einer Situation wie in Abbildung 3.10 ein beweglicher Messpunkt während einer Simulation „hinter“ eine andere Punktladung bewegt, können wir zweifelsfrei von einer Äquivalenz des übertretenen mit dem verbundenen Punkt ausgehen. Über dieses Kriterium können wir durch Konstruktion einer Ebene und einem Test ermitteln, ob der Messpunkt über der Ebene liegt. Diese lässt sich leicht in der Normalenform

$$\mathbf{n} \bullet (\mathbf{p} - \mathbf{q_2}) = 0 \tag{3.13}$$

mit dem Aufpunkt q_2 und der Normalen $(q_1 - q_2)$ konstruieren. Diese Gleichung beschreibt alle Punkte der Ebene, die den Bereich „hinter“ von dem Bereich „vor“ der Ladung q_2 trennt. Das Ergebnis ihres linken Terms nach Einsetzen der Koordinaten des Messpunkts gibt Aufschluss darüber, ob der Punkt „vor“ (< 0), „auf“ ($= 0$) oder „hinter“ (> 0) der Ebene liegt. Mit einer anderen Anschauung lässt sich diese Aussage als Messen des Winkels zwischen dem Normalenvektor $\mathbf{n}$ der Ebene aus Gleichung (3.13) und dem Richtungsvektor $\mathbf{p} - q_2$ verstehen: Zeigen die Vektoren wie in Abbildung 3.11 gegeneinander, wird das Skalarprodukt negativ. Nach Übertritt durch die Ebene zeigen die Vektoren in etwa die gleiche Richtung, und das Skalarprodukt wird positiv. Für einfache Fälle, in denen beispielsweise nur zwei Punktladungen oder eine annähernd lineare Anordnung vorliegt, kann dieses Kriterium bereits hinreichend sein und durch seine geringe Laufzeitkomplexität mit $\mathcal{O}(1)$ ein gangbarer Weg bei Präferenz für Geschwindigkeit gegenüber der Zuverlässigkeit der Ergebnisse sein. Bei komplexen Anordnungen sind leicht Beispiele zu finden, in denen so gefundene Äquivalenzen als falsch angenommen zu werten sind.

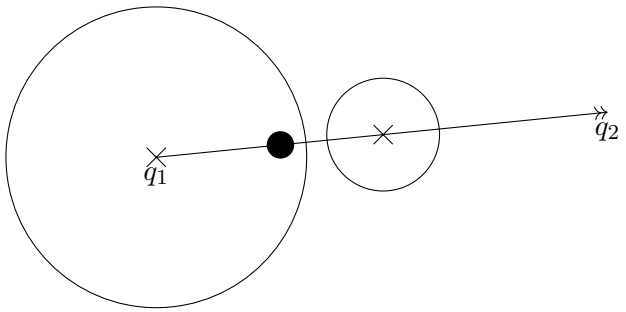


Abbildung 3.9: Messpunkt innerhalb der Fläche, die q_1, aber nicht q_2 beinhaltet, aber näher an q_2.

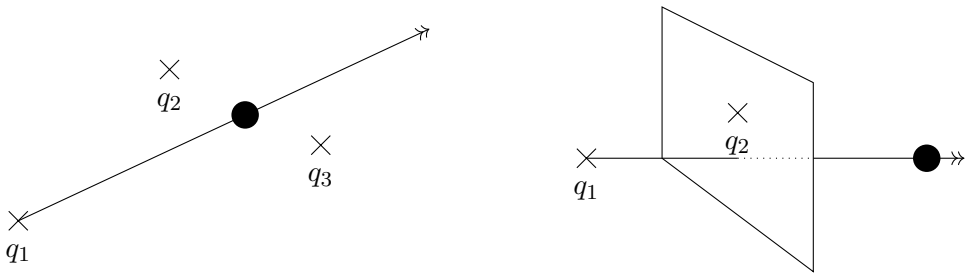


Abbildung 3.10: Messpunkt befindet sich „hinter“ q_2 aus Sicht von q_1, was auf Äquivalenz hinweist.

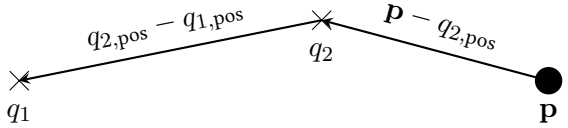


Abbildung 3.11: Das Skalarprodukt der beiden Vektoren ist hinreichend für das Kriterium: „hinter“ q_2 aus Sicht von q_1.

3.3.3.2 Erkennen falsch angenommener Äquivalenzen

Wenn bezüglich zweier Punktladungen von einer Äquivalenz ausgegangen werden kann, werden diese durch ein einzelnes Mesh repräsentiert. Werden die Vertices dieses Meshes auf Halbgeraden bewegt, wird sich, falls die Äquivalenz falsch angenommen wurde, ein Artefakt auf diesem Mesh herausbilden: Da das Mesh zwei Flächen repräsentiert, werden die Vertices mindestens eines Faces dieses Meshes auf unterschiedlichen Flächen zu liegen kommen. Bei ausreichend großer Fläche sind diese Faces daran erkennbar, dass das elektrische Feld an den Vertices annähernd antiparallel ausgerichtet ist, während das Feld an Vertices, annähernd parallel ist. Die Analyse eines Faces, ob seine Vertices verschiedene Flächen repräsentieren, läuft für reguläre Meshes in konstanter Zeit. In einer Simulation, in der ohnehin eine Iteration über alle Faces durchgeführt wird, fällt kein asymptotischer Zusatzaufwand an.

3.3.4 Fehlerkorrektur durch Zusammenfügen von Meshes

Wenn die Isofläche in einer Simulation durch iterative Verfeinerung angenähert wird und sich dabei durch Anwenden der Kriterien in Abschnitt 3.3.1 eine Annahme einer Nicht-Äquivalenz als falsch herausstellt, muss ein adäquates Mittel zur Reaktion bereitstehen. Ziel einer solchen Fehlerkorrektur ist es, die Anordnungen der Meshes in einen Zustand zu überführen, der vorgelegen hätte, wäre statt der falschen die korrekte Annahme getroffen worden. Im Falle der Nicht-Äquivalenz bedeutet das, dass mehrere Meshes eine Fläche der Isofläche repräsentieren. Prinzipiell ist als Fehlerkorrektur denkbar, eines der kollidierenden Meshes zu löschen oder beide Meshes zusammenzufügen, sofern nicht mehrere falsch angenommene Nicht-Äquivalenzen

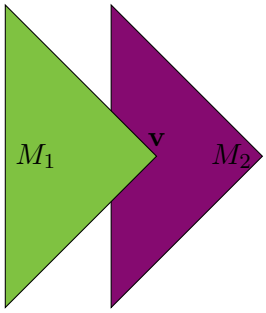


Abbildung 3.12: Übertreten eines Vertex in ein anderes Mesh.

gleichzeitig festgestellt werden können. In den folgenden Abschnitten besprechen wir diese beiden Optionen.

3.3.5 Zusammenfügen zweier Meshes

Gegenüber dem Löschen von Geometrie unter Verlust von Informationen besteht die Möglichkeit, zwei Meshes M_1 und M_2 nach Feststellen der falschen Annahme der Nicht-Äquivalenzen an einem geeigneten Zeitpunkt aufzutrennen, wobei wie in Abbildung 3.12 illustriert M_2 einen Vertex $\mathbf{v}$ von M_1 beinhaltet. Dabei entstehen zwei offene Meshes, die sich durch geeignete Mittel wieder zu einem Mesh zusammenzufügen lassen. Dieser Vorgang ist als Vereinigung aus dem Themengebiet der konstruktiven Festkörpergeometrie (CSG) bekannt und stützt sich auf besondere Methoden der Berechnung durch Raytracing [Gha08], die bei Vorliegen als begrenzungsflächenrepräsentierte Geometrie durch Löschen der überschneidenden Flächen und Schließen der so entstandenen Öffnungen eine anschließende Triangulation umgesetzt werden müsste [Sil10], wenn möglich. Allerdings beschreiben die Kanten dieser Öffnung ein Polygon im Raum – also einen Kantenzug mit Punkten in dreidimensionalen Koordinaten – und entsprechen jeweils der Folge der Kanten, die eine entfernte von einer verbleibenden Fläche trennen.

Die zu löschenden Flächen des M_1 sind gleichzeitig das Ergebnis des Punkt-Polyederbeinhaltungstest in Abschnitt 3.3.1. Da wir anhand von Öffnungspolygonen argumentieren, können wir ohne die Allgemeinheit zu beschränken, davon ausgehen, dass die Punktbeinhaltungstests der Vertices genau einen positiven Punkt ergeben haben: In allen anderen Fällen sind die Flächen zusammenhängend und bilden wiederum ein Öffnungspolygon, oder sie sind nicht zusammenhängend und somit Ergebnis mehrerer positiver Punktbeinhaltungstests mit unterschiedlichen Vertices, weswegen die folgenden Überlegungen die jeweiligen Punkte getrennt zutrifft.

Zu löschen sind alle Flächen in M_1, deren Kanten M_2 teilen. Dabei entsteht eine Öffnung, die ein Polygon G beschreibt. Ebenfalls muss mindestens eine geeignete Fläche des M_2 gelöscht

werden, wodurch das Öffnungspolygon K entsteht. Für die Bestimmung dieser Fläche stehen verschiedene Möglichkeiten zur Auswahl, die sich abhängig vom Anwendungsfall anbieten:

- Da M_2 sowohl mindestens eine Punktladung an der Position $\mathbf{p}$ als auch $\mathbf{v}$ beinhaltet, muss eine Strecke $\overline{\mathbf{pv}}$ mindestens eine – bei einer frühen Fehlererkennung genau eine – Fläche des M_2 schneiden. Der Test aller Flächen führt zu einer Zeitkomplexität von $O(n)$. Bei einer linearen Trajektorie von $\mathbf{v}$ wird dabei exakt die durchstoßene Fläche des M_2 erkannt. Wird der Fehler zu spät erkannt, muss die Fläche ausgewählt werden, deren Schnittpunkt mit der Strecke am nächsten bei der Punktladung liegt

- Die Kanten, die $\mathbf{v}$ schneiden, müssen jeweils auch mindestens eine – bei einer frühen Fehlererkennung genau eine – Fläche des jeweils anderen Meshes schneiden. Da hier alle Kanten des einen Meshes gegen alle Flächen des anderen getestet werden müssen, entsteht so eine Zeitkomplexität von $O(n^2)$. Weiterhin kann bei einer späten Fehlererkennung der Schnittest auf mehrere Flächen anschlagen. Daher ist hier ein weiterer Test mit einer Zeitkomplexität von $O(n^2)$ auf zu entfernende isolierte Flächen durchzuführen. Dieses Vorgehen ist zu empfehlen, wenn eine flächenmäßig ähnliche Größe der Öffnungspolygone gegenüber niedriger Laufzeit bevorzugt wird.

Allgemeine Triangulierungsverfahren wie die Delaunay-Triangulierung – selbst wenn sie auf dreidimensionale Fälle erweitert werden – können hier aufgrund eines Typenkonflikts nicht zur Anwendung kommen: Diese bilden eine Menge von Punkten auf ein triangulares Mesh ab [ABL03]. Im vorliegenden Sonderfall sollen aber zwei Polygone durch ein offenes Mesh miteinander verbunden werden. Dabei ist die Forderung nach Regularität oder sogar Triangularität des Meshes fakultativ, dafür muss die besondere Anforderung nach der Geschlossenheit des Polygons abgesehen von zwei Öffnungen gestellt werden. Die Polygone für die Eingabe der Triangulierung können wir zwar wie etwa auch in Abschnitt 3.3.1 als Folgen von Punkten verstehen, aber eine Triangulierung schließt ebenfalls die Öffnungspolygone ein. Dieser Effekt ist unerwünscht: Ein Mesh, das eine Fläche einer Isofläche repräsentiert, darf keine zusätzlichen Faces enthalten. Dies würde implizieren, dass die repräsentierende Fläche von einer anderen Fläche geschnitten würde.

3.3.6 Konstruktion eines schließenden Polygons durch Quads und Triangle Fans

Da in vielen Fällen – besonders in solchen, in denen während einer Simulation relativ frühzeitig eine falsch angenommene Nicht-Äquivalenz erkannt wird – das Entfernen der beteiligten Flächen ohne das Entfernen informationstragender Messpunkte nicht möglich ist, bietet es sich an, ein offenes Mesh M_3 zu finden, das $M_1 + M_2$ schließt, ohne dass M_3 neue Punkte

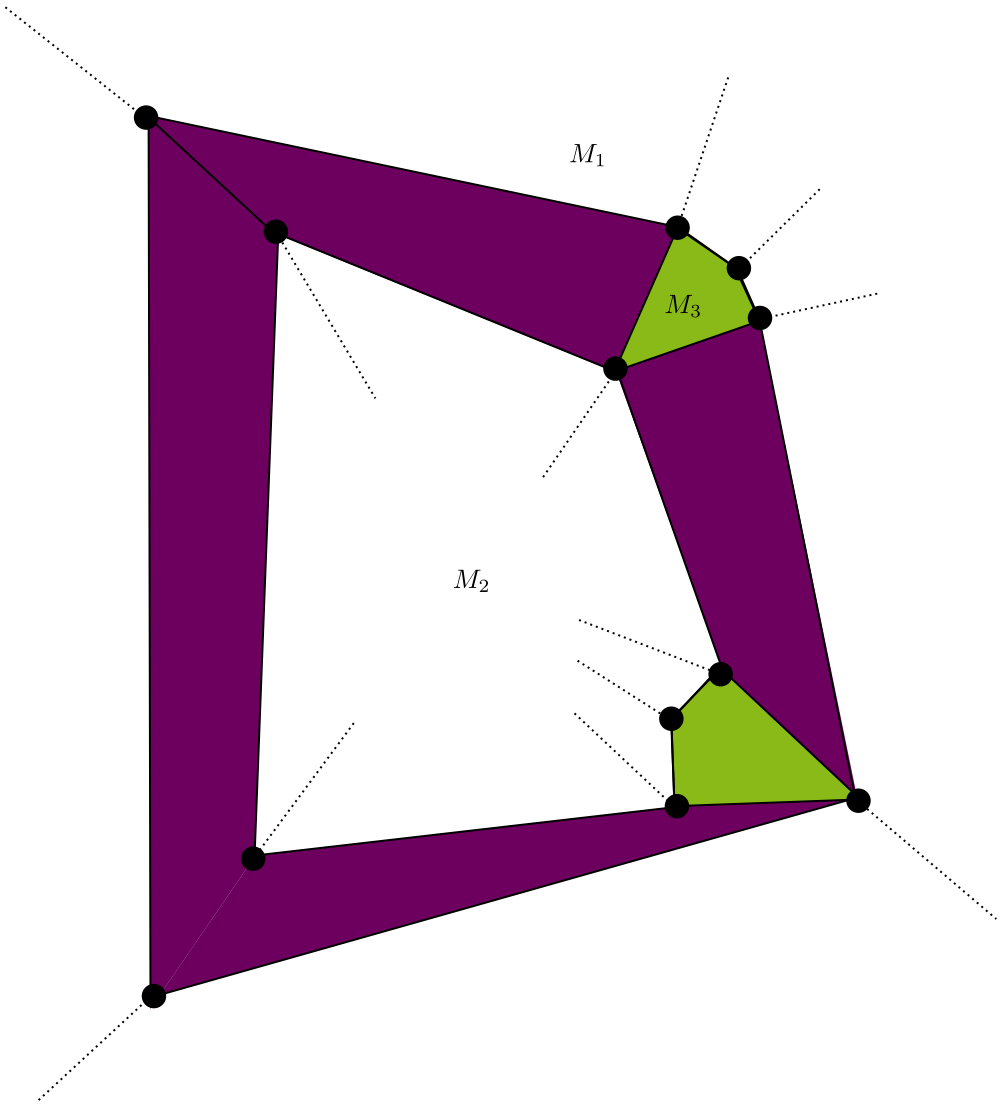


Abbildung 3.13: Das farbige Mesh M_3 verbindet das äußere Polygon M_1 mit dem inneren Polygon M_2 durch Quads (dunkelmagenta) und Triangle Fans (grün)

enthält. Dazu suchen wir eine Funktion $f : P \times P \to M$, die ein solches Mesh zum Ergebnis hat. Dabei bezeichnet P die Menge aller Polygone, M die Menge aller Meshes. Für diese Problemstellung existieren Lösungen zur Überbrückung von Polygonen, die bereits in 3D-Modellierungswerkzeugen wie Blender angewendet werden [Vil14, S. 79]. Eine Möglichkeit zur Umsetzung dieses Algorithmus ist, die Kanten des M_3 so zu wählen, dass sie den Abstand der Punkte zueinander minimieren. Der Abstand der Punkte zueinander muss in einer projektiven Ebene erfolgen, um sicherzustellen, dass jeder Punkt aus beiden Polygonen mindestens ein mal der nächste Punkt zu einem Punkt des jeweils anderen Polygons in Frage kommt. Somit verbinden diese Kanten jeden Punkt aus M_1 zu dem jeweils nächsten Punkt aus M_2. Sind

mehrfach hintereinander die gleichen Punkte aus M_2 einem Punkt aus M_1 am nächsten, ergibt sich als verbindendes Teilmesh ein Triangle Fan, ansonsten liegt eine einzelne Kante vor, die zusammen mit der nächsten Kante ein Quad ergibt. Diese Argumentation gilt umgekehrt analog für Triangle Fans mit einem Mittelpunkt aus M_2. Triangle Fans mit Mittelpunkten aus M_1 und aus M_2 müssen durch ein Quad getrennt sein, weil sonst der Mittelpunkt des einen Triangle Fans ein Randpunkt des anderen wäre. Auch können zwei Triangle Fans mit Mittelpunkten aus dem gleichen Polygon nicht unmittelbar aufeinander folgen, da diese den gleichen Mittelpunkt hätten und es somit nur ein Triangle Fan wäre. Ein Beispiel für ein nach diesen Überlegungen entstandenes Verbindungsmesh ist in Abbildung 3.13 bildhaft dargestellt.

Im Folgenden spezifizieren wir einen Algorithmus, der zwei Polygone nach den obigen Vorgaben durch Konstruktion eines dritten Meshes miteinander zu einem geschlossenen Mesh verbindet. Die beiden zu verbindenden Polygone bezeichnen wir als die Folgen $k + K$ und $g + G$ und die Konstruktionsvorschrift:

$$\text{polygon}(k + K, g + G) = t + q + r$$

konstruiert werden, wobei t eine Punktfolge für ein Triangle Fan oder eine leere Punktfolge, q das Quad das letzte Triangle Fan bzw. Quad mit dem nächsten verbindet und r der Rest des Polygons bezeichnen. Diese Komponenten bestimmen wir im Folgenden so, dass jede Kante ein Punkt des einen Öffnungspolygons mit dem anderen mit der minimalen Kantenlänge verbindet. Sowohl Triangle Fans als auch Quads lassen sich als Punktfolgen beschreiben. Wir richten uns hier nach der Konvention, den Mittelpunkt der Triangle Fans an den Anfang der jeweiligen Punktfolge zu stellen. Somit bildet die Funktion polygon Punktfolgen auf Punktfolgen ab.

Die Hilfsfunktion $d : P \times P^n \to P$ hat zum Zweck, aus der Punktliste der Eingabe den Punkt zurückzugeben, der dem Punkt der Eingabe am nächsten ist. Durch Currying erhalten wir die Funktionssignatur: $d : (P \to P^n) \to P$. Nach der Teilanwendung auf einen Punkt $\mathbf{x}$ erhalten wir eine neue Funktion $d'_{\mathbf{x}} : P^n \to P$, die sich schließlich geeignet formulieren lässt:

$$d'_x = \min_{\|()\| \circ (x-)}/$$

Die Funktion $d'_{\mathbf{x}}$ subtrahiert von jedem Punkt der übergebenen Folge den Punkt $\mathbf{x}$, bildet den Betrag des Ergebnisses und liefert schließlich denjenigen Punkt zurück, für den der Betrag am kleinsten ist.

Da immer der nächste zu einem bestimmten Punkt gesucht wird, ist immer mindestens eine der beiden Folgen n_k und n_g einwertig. Sind beide einwertig, wird kein Triangle Fan

konstruiert.

$$t = \begin{cases} \varnothing & \text{wenn } \#n_k = \#n_g = 1 \\ \{k + n_k\} & \text{wenn } \#n_g = 1 \\ \{g + n_g\} & \text{wenn } \#n_k = 1 \end{cases}$$

Die Funktion t konstruiert – sofern nicht beide Punktmengen einwertig sind – eine Folge, die zu Beginn den Mittelpunkt und anschließend die äußeren Punkte eines Triangle Fans enthält. Der Mittelpunkt a oder b ist der einzige Wert der jeweils einwertigen Menge. Die äußeren Punkte repräsentieren die Folge aller Elemente am Beginn der Punktfolge des anderen Polygons, die ebenfalls am nächsten am Mittelpunkt liegen. Dadurch lassen sich n_k und n_g konstruieren als:

$$n_k = \text{prefix}\left((= k), d'_{d'_k(G)}(K)\right) \qquad n_g = \text{prefix}\left((= g), d'_{d'_k(G)}(K)\right)$$

Dieser Schritt ist für n_g in Abbildung 3.14 illustriert.

Die Quads, die zwei Triangle Fans voneinander trennen, lassen sich jeweils als zwei Dreiecke und damit ebenfalls als Triangle Fans mit je einem Mittel- und zwei äußeren Punkten auffassen. Durch diese Erkenntnis lässt sich die Funktion q formulieren:

$$q = \{\text{last}\,(n_k)\,, \text{last}\,(n_g)\,, \text{head}\,(r_g)\} + \{\text{head}\,(r_k)\,, \text{head}\,(r_g)\,, \text{last}\,(r_k)\}$$

Der Rest der Folge r vollführt die Rekursion mit den verbleibenden Elementen der Punktfolgen:

$$r = f\,(\text{drop}\,(\#n_k, K)\,, \text{drop}\,(\#n_g, G))$$

.

Zur Berechnung des Aufwands ist zu bemerken, dass die Funktion d'_a nicht Teil der Iteration ist, sondern im Voraus berechnet wird. Darüber hinaus weisen beide d-Funktionen aufgrund der Reduktion einen Aufwand von $\mathcal{O}(n)$ auf. Die zweite d-Funktion ist ihrerseits Teil der Generierung der Liste, die für die prefix-Funktion verwendet wird. prefix weist ebenfalls einen Aufwand von $\mathcal{O}(n)$ auf. Die asymptotische Gesamtlaufzeit beläuft sich also auf $\mathcal{O}(n^2)$.

3.3.7 Fehlerkorrektur durch selektives Einblenden von Meshes

Dieser Abschnitt untersucht eine Möglichkeit, auf das Erkennen falsch angenommener Äquivalenzverhältnisse zu reagieren, die sich zum Vorgehen in Abschnitt 3.3.4 dahingehend unterscheidet, dass auf die initiale Erstellung der während einer Simulation durch Auftrennen und Zusammenfügen möglicherweise entstehenden Meshes während der Initialisierungsphase abstellt, statt diese Operationen während der Simulation durchzuführen. Daraus werden unter anderem Meshes entstehen, die Äquivalenzverhältnisse falsch abbilden, und diese sind

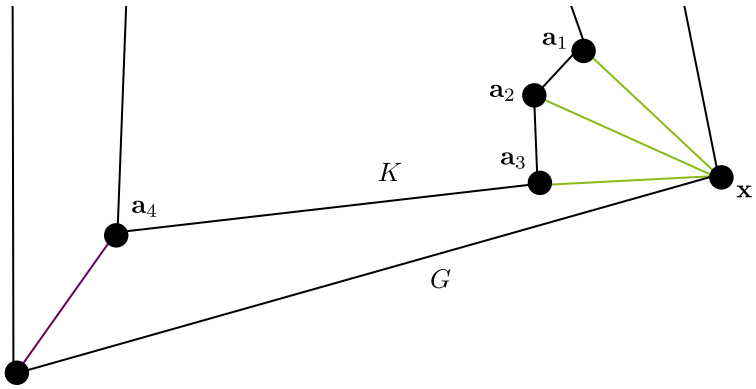


Abbildung 3.14: Bildhafte Darstellung eines Rekursionsschritts. $\mathbf{a}_1$ bis $\mathbf{a}_3$ aus K sind dem Punkt $\mathbf{x}$ aus G am nächsten. a_4 ist somit nicht Teil von n_g.

erst während der Simulation identifizierbar. Um keine Szene aus Meshes zu erstellen, die den Kriterien für Eigenschaften von Isoflächen widersprechen – beispielsweise, dass diese nicht geschachtelt auftreten oder immer mindestens eine Punktladung enthalten – finden wir das Kriterium der Pareto-Effizienz. Wir entwickeln den markierten Konturbaum als logische Struktur, an denen Pareto-Schritte durchgeführt werden können. Für die neue Datenstruktur wird ein Zeiger-Diagramm entwickelt, das die Grundlage einer Komplexitätsanalyse bildet.

Nach dem Erkennen falsch angenommener Äquivalenzen gemäß Abschnitt 3.3.1 liegt die Situation vor, dass ein Mesh nicht äquivalente Punktladungen beinhaltet. Nicht-Äquivalente Ladungen werden von unterschiedlichen Flächen umgeben. Wurde aber ihre Äquivalenz angenommen, werden diese Flächen unkorrekterweise von einem Mesh repräsentiert. In diesem Fall muss die Szene in einen Zustand überführt werden, in dem jede dieser Flächen eine Repräsentation durch ein Mesh findet.

Bei diesem steht die Entscheidung an, ob, wenn eine Überschneidung aufgetreten ist, die Simulation der beiden betroffenen Meshes in dem Wissen fortzuführen ist, dass eines von beiden überflüssig ist (S. Abschnitt 3.3.7).

Ein Analogon zum Zusammenfügen zweier Meshes wie bei einer festgestellten falsch angenommenen Nicht-Äquivalenz – das Aufschneiden des Meshes und das anschließende Verbinden der Öffnungen durch ein drittes Mesh zu einem gesamten Mesh – steht in diesem Fall nicht zur Verfügung. Zwar lässt sich leicht einsehen, dass nach dem Auftrennen die gleiche Ausgangssituation wie in Abschnitt 3.3.4 nach dem Löschen des vom anderen Mesh enthaltenen Vertices entsteht: Zwei Meshes mit einer Öffnung in Gestalt eines Polygons. Allerdings lassen sich einerseits die Vertices des aufzuteilenden Meshes nicht eindeutig einem Cluster zuordnen. Lediglich durch Heuristiken wie dem Skalarprodukt des elektrischen Feldes an zwei Vertices lässt sich feststellen, ob diese in unterschiedlichen Clustern liegen. Werden die Vertices auf

Halbgeraden bewegt, wird bei Erkennen falscher Annahmen bezüglich der Äquivalenz eine Neuausrichtung der Bewegungshalbgerade nötig. Dieses Problem ist aufgrund der Arbitrarität der Cluster nicht in allen Fällen lösbar. Da die Flächen unterschiedlich groß sein und die Erkennung der falsch angenommenen Nicht-Äquivalenz früh eintreten können. In solchen Fällen können sich Vertices noch sehr nah an der falschen Punktladung befinden. Andererseits ist der Algorithmus aus Abschnitt 3.3.4 nicht auf das Schließen der Öffnungspolygone mit mehr als drei Punkten anwendbar, weil sich die schließende Geometrie nicht in jedem Fall als Triangle-Strip darstellen lässt. Dessen unbeschadet liegt ein Algorithmus zur vollständigen Triangulierung einfacher Polygone vor, der annähernd lineare Laufzeit aufweist [CTW88].

Eine Alternative dazu bietet das selektive Ein- und Ausblenden von Meshes. Zu diesem Zweck kann in einer Szene nach dem Erkennen einer falsch angenommenen Äquivalenz das mehrere Flächen repräsentierende Mesh aus- und stattdessen mehrere kleine Meshes eingeblendet werden, die eine bessere Repräsentation der Isoflächen darstellen. Die erreichte Verbesserung lässt sich durch die Pareto-Effizienz bemessen.

Pareto-Verbesserung, Pareto-Optimum, Pareto-Menge

In einem zustandsbehafteten System, dessen Zustände nach mehreren Eigenschaften ordinal bewertet werden können, bezeichnet der Übergang eines Zustandes in einen höher bewerteten Zustand *Pareto-Verbesserung*, wenn der höher bewertete Zustand durch Verbesserung einer Eigenschaft erreicht wird, ohne dass dadurch eine andere Eigenschaft verschlechtert wird. Ist keine Pareto-Verbesserung mehr erreichbar, befindet sich das System in einem *Pareto-Optimum*. Die Menge aller Pareto-Optima heißt *Pareto-Menge*.

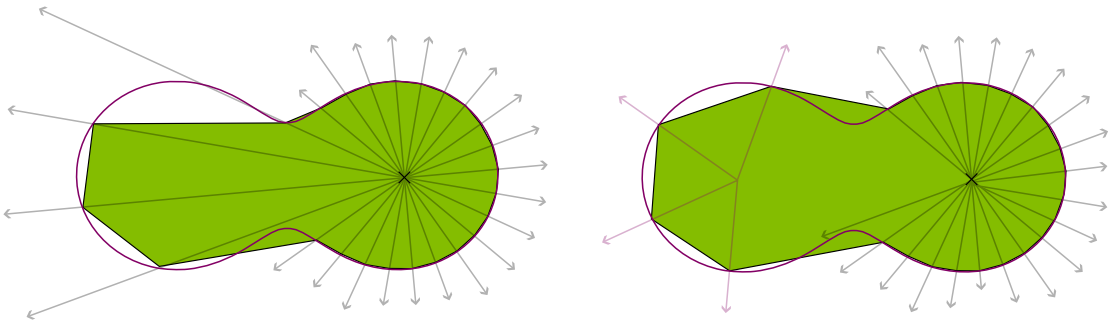
Eine Szene mit ein- und ausgeblendeten Meshes kann als ein solches System verstanden werden, wobei die aufgrund falscher Annahmen bezüglich der Äquivalenz von Punktladungen unkorrekterweise aus- oder eingeblendete Meshes, und korrekt aus- oder eingeblendete Meshes die Eigenschaften bezeichnen. Eine dieser Eigenschaften ist bei Abwesenheit von falschen Annahmen bezüglich der Äquivalenzen als hoch zu bewerten, bei Anwesenheit derselben als niedrig. Eine Überführung der Szene in einen korrekteren Zustand stellt sich also als eine Pareto-Verbesserung dar. Sobald keine Pareto-Verbesserungen mehr vorgenommen werden können, ist ein Pareto-Optimum erreicht. Im vorliegenden Fall kann jede Veränderung der Repräsentation der Flächen durch Meshes nur eine Pareto-Verbesserung oder eine Verschlechterung sein, weswegen die Pareto-Menge des vorliegenden Systems einwertig sein muss. In diesem Zustand ist die Repräsentation der Isofläche korrekt, d. h. keine falsch angenommenen Äquivalenzen liegen mehr vor. Die Pareto-Effizienz stellt sich also als geeignetes Werkzeug heraus, die Entscheidung, ein Mesh ein- oder auszublenden, zu bewerten und ein eindeutiges Optimum zu identifizieren.

3.3.7.1 Naives Ausblenden

Zu Beginn einer Simulation ist die Äquivalenzrelation der Punktladungen nicht bekannt und damit ebenfalls nicht, welche Punktladungen durch welches Mesh repräsentiert werden könnten. Wird eine falsch angenommene Nicht-Äquivalenz erkannt, erweist sich eines der Meshes als überflüssig, so dass es aus dem Szenegraphen entfernt (ausgeblendet) werden kann. Da die Fläche, die diese Meshes repräsentieren sollen, keine Bereiche mit einer Feldstärke unterhalb des Isowertes enthält, werden beide Meshes während einer Simulation etwa die Form der Fläche annehmen, so dass eine eingehende Analyse, welches der beiden überschneidenden Meshes ausgeblendet werden sollte, nicht lohnend erscheint. Ohnedem laufen viele dieser Situationen auf symmetrische Konstellationen hinaus, in denen arbiträr eines der Meshes ausgeblendet werden kann.

Ein Ausblenden des Meshes ohne weitere Maßnahmen führt im Fortlaufen der Simulation zu Bereichen ungleichmäßiger Verteilung von Vertices auf dem Mesh, außerdem zu den in Abschnitt 3.2.3 besprochenen Artefakten. In den Überlegungen in Abschnitt 3.3.4 trat dieses Problem nicht auf, da sich die Vertices der zusammengefügten Meshes auf verschiedenen Trajektorien bewegten. Da gemäß Abschnitt 3.2.3 konkave Bereiche des Meshes nur zwischen Ladungen entstehen können, ist es möglich, nach dem Entfernen eines der Meshes, die Bewegungshalbgerade eines Teils der Vertices auf dem Mesh neu auszurichten. Die Ermittlung der neu auszurichtenden Vertices mündet in einem Suchproblem: Da wir ohne die Allgemeinheit zu verletzen eine symmetrische Anordnung nicht voraussetzen können, wird die Liste der Vertices abhängig von einem Kriterium – denkbar wäre hier etwa der kürzere Abstand des Vertex zu den betreffenden Punktladungen – aus unterschiedlich großen Bereichen mit Vertices bestehen, die dieses Kriterium erfüllen, während die Vertices außerhalb dieser Bereiche es nicht erfüllen. Bei einem Aufwand von $\mathcal{O}(1)$, die Neuausrichtung vorzunehmen, wird dieses Vorhaben eine Laufzeitkomplexität von $\mathcal{O}(n)$ aufweisen. Das Beispiel in Abbildung 3.15a kann auf diese Weise von einigen Artefakten befreit werden. Ein mögliches Ergebnis ist in Abbildung 3.15b angedeutet. Dem Problem variierender Dichte von Vertices, das ebenfalls das Auftreten von Artefakten begünstigt, widmen wir an gegebener Stelle einen eigenen Abschnitt.

In den folgenden Abschnitten erläutern wir, dass sich die Menge möglicher Meshes als Baum konstruieren lässt, der seinerseits einen zur Simulation parallel laufenden Greedy-Algorithmus auf einer baumartigen komplexen Datenstruktur mit konstanter Laufzeit ermöglicht. Dabei orientieren wir uns an dem Vorschlag von [SB06] und [She98], die unabhängig voneinander Konturbäume (bei [She98] „indizierte Bäume“ (*index tree*)) verwenden, um zeitlich veränderliche Isoflächen abzubilden bzw. die Speicherkomplexität und -zugriffsanfragen zu reduzieren.



(a) Bedingt durch die Konstellation der Halbgeraden zur Fläche weist es Kanten unterschiedlicher Länge, Artefakte und mehrdeutige Lösungen auf.

(b) Durch Neuausrichtung der Bewegungshalbgerade konnten zwei Artefakte und mehrdeutige Lösungen beseitigt werden.

Abbildung 3.15: Ein Mesh (grün), das die Fläche (dunkelmagenta) und zugehörige Halbgeraden (grau)

Baum

In der Graphentheorie bezeichnet ein *Baum* einen zyklenfreien, zusammenhängenden, gerichteten Graphen ohne geschlossene Pfade. Der Fuß einer gerichteten Kante in einem Baum heißt *Elternelement* ihres Kopfes, dem *Kindelement* ihres Fußes. Knoten ohne Kindelemente bezeichnen wir als *Blätter*, Knoten ohne Elternelemente als *Wurzel*.

Diese Definition findet ihre Entsprechung in der praktischen Informatik als Datenstruktur, in der wir die Kanten eines Baums beispielsweise als Zeiger und die Knoten als eigenen komplexen Datentyp mit weiteren Nutzdaten abbilden können.

Bäume lassen sich auch durch die Funktion $\{()\} : \mathcal{T} \to \mathcal{T}$, wobei $\mathcal{T}$ die Menge der Bäume ist, mit der Bedeutung „ist Kindknoten von“, darstellen. Diese und ähnliche Notationen finden mehr oder weniger spontan motiviert breite Verwendung in der Fachliteratur verschiedener Disziplinen, für die Bäume relevant sind [Goz+14] [Glu20] [MR97][Str03, Kap. 2. 6]. Diese Form der Darstellung verwenden wir im Folgenden, um das Konzept der Markierbarkeit zu erweitern und an dieser neuen Struktur logische Operationen vorzunehmen.

Beispiel

Der in Abbildung 3.17 dargestellte Baum lässt sich in dieser Schreibweise wiedergeben als

$$\{\{q_1, q_2\}, q_3\}$$

Die äußere Klammer stellt hier das Wurzelelement dar, die mit q indizierten Elemente die Blätter, die inneren Klammern den Knoten $\{q_1, q_2\}$ der durch die Kindknoten q_1 und q_2 identifizierbar ist.

3.3.8 Markierung

Der Zweck des Baumes innerhalb der Simulation wird durch die Nutzdaten der Datenstruktur realisiert. Diese können sich, wenn sie beispielsweise in C99 implementiert würden, in einem Zeiger auf ein Mesh `struct Mesh*` und eine Markierung, die den Zustand des Meshes hinsichtlich der Sichtbarkeit `bool isVisible` manifestieren. Diese Markierung erweist sich angesichts der Eigenschaften von Isoflächen, dass eine Fläche selbst keine Fläche beinhalten kann, als notwendig: Wird eine falsche Annahme über die Rolle einer Punktladung in der Äquivalenzklasse festgestellt, muss in der Szene ein anderes Mesh eingeblendet werden, das das Cluster besser repräsentiert und diese Ladung ebenfalls beinhaltet. Da aber eine Punktladung nicht von mehreren Flächen – ebenso wenig von mehreren Meshes – repräsentiert werden darf, müssen während eines solchen Schritts immer gleichzeitig Meshes aus- und andere eingeblendet werden.

Auf diese Weise haben wir die Struktur des Baumes durch Einführen der Markierung um die Darstellung eines Zustandes erweitert. Eine für C nutzbare, dynamische Datenstruktur ist als Zeiger-Diagramm in Abbildung 3.18 abgebildet.

Sofern sich eine Heuristik, die sich beispielsweise den Abstand von Punktladungen mit homogener Ladungsstärke zunutze macht, finden lässt, ergibt sich die Situation, dass alle Meshes in einem solchen Baum während der Simulation ganz von einem anderen Mesh beinhaltet werden. Abbildung 4.3 veranschaulicht diesen Umstand. Übertragen wir die Beziehung „ist Kindknoten von“ der Bäume auf eine entsprechende Beziehung „beinhaltet“ bezüglich Flächen und Meshes, offenbart sich die zugrunde liegende Struktur des Konturbaumes. Die Markierung spiegelt die Konfiguration der sichtbaren und unsichtbaren Meshes wider. Meshes, die in dem Beispiel in Abbildung 4.3 etwa ein Mesh für die Repräsentation einer Fläche $\{q_1, q_2\}$ beinhaltet, sind per Konstruktion nicht Teil dieses Baumes und spielen in einer Simulation keine Rolle.

Einen markierbaren Knoten n eines Graphen wollen wir im Folgenden mit n' kennzeichnen. n° bezeichnet einen Knoten explizit ohne Markierung. Die Bezeichnung n ohne Zusatz verwenden wir weiterhin agnostisch zur Markierung.

3.3.9 Markierter Konturbaum und markierter DAG

Das Voraussetzen einer Heuristik, die das Generieren einer Baumstruktur beispielsweise unter Zuhilfenahme des Abstands der Punktladungen ermöglicht, kann in häufigen Fällen ein einfacher, gangbarer Weg sein, schränkt aber die Allgemeinheit der Argumentation ein: Aufgrund der Arbitrarität der Cluster können wir lediglich als bekannt voraussetzen, dass den für die Repräsentation der Flächen in Frage kommenden Meshes eine Baumstruktur zugrunde liegen muss. Dessen Knoten sind aber abgesehen von den Blättern und der Wurzel unbekannt. Allerdings ist die Menge aller Knoten – derer, die Teil des Baumes sind, der bei einer korrekten Markierung

genau diejenigen Meshes einblendet, die die Isofläche repräsentieren und derjenigen, die nicht Teil dieses Baumes sind – bestimmbar, und Teil eines Baums, der keine Markierung erreichen kann, so dass die entsprechend eingeblendeten Meshes die Isofläche repräsentieren können.

Die Aufgabe kann also auch als Suche nach einem aus vielen möglichen korrekten Bäumen verstanden werden. All diesen Bäumen ist gemein, dass das Wurzelelement die Isofläche repräsentiert, die alle Ladungen beinhaltet. Wenn aber mehrere Bäume die gleiche Wurzel miteinander teilen, so handelt es sich nur um einen Baum mit ebendieser Wurzel.

Dieser Baum ist isomorph zur Potenzmenge aller Punktladungen ohne die leere Menge, wenn wir die möglichen Meshes mit den Elementen der Potenzmenge und Beziehung „beinhaltet“ bezüglich zweier Meshes bzw. Mengen in der Potenzmenge miteinander identifizieren. Diese Beziehung deutet sich in der didaktischen Literatur an, in der Bäume bisweilen als Illustration der Konstruktion von Potenzmengen herangezogen werden [PDP01, S. 136] [Hac08, S. 110]. Im Folgenden werden wir sehen, dass die leere Menge explizit ausgeschlossen werden muss. Sie wäre ein Blatt und würde bei Markierung ein Mesh einblenden, das aber keine Ladungen enthält, also ein Mesh, das keine Fläche repräsentiert. Dies ist nach Punkt 4 auf Seite 37 ausgeschlossen. Die Funktion $\wp : \mathcal{M} \to \mathcal{M}$ bildet auf einfache Weise nach der Vorschrift in Gleichung (3.14) rekursiv die Potenzmenge einer beliebigen Menge $\{a\} \cup M \in \mathcal{M}$ ab, wobei $a \neq M$. Gleichung (3.14) orientiert sich an üblichen Implementierungen [Ros23], sieht aber einen Abbruch bereits bei einwertigen Mengenparametern vor, um die leere Menge als Element auszuschließen.

$$\wp(\{a\} \cup M) = \{a\} \cup \wp(M) \cup (\{a\} \times \wp(M)) \qquad \wp(\{a\}) = \{a\} \tag{3.14}$$

Die Potenzmenge enthält natürlicherweise die Blattknoten mehrfach in verschiedenen Untermengen als Elemente. Die obige Definition des Baums sowie der Potenzmenge erlauben das mehrfache Auftreten von Elementen, allerdings wird sich in einer Szene mit Punktladungen jede nur einfach finden. Dopplungen müssten sonst von der Berechnung des elektrischen Feldes explizit ausgeschlossen werden. In einer wie oben bereits angedachten Implementierung in C wären die Datenobjekte dieser Punktladungen zeigeridentisch, wodurch sich die Dopplung aufhöbe. Das Beispiel, das im Zeiger-Diagramm in Abbildung 3.18 dargestellt ist, berücksichtigt diesen Fall. Unter Umständen erfordert die Implementierung der Simulation eine Identifikation der Zeigeridentität mit der „beinhaltet“-Beziehung, etwa um ein Mesh-Objekt mit mehreren doppelt auftretenden Clustern, die es repräsentieren soll, zu verknüpfen. Eine solche Beziehung kann als schließende Kante verstanden werden, die der Definition des Baums widerspricht. Da wir in diesem Kapitel ausschließlich über gerichtete Graphen sprechen, wird dadurch die Eigenschaft der Zyklenfreiheit von Bäumen nicht verletzt, auch sind Wurzel-, Blätter und Eltern- bzw. Kindelemente noch eindeutig identifizierbar, allerdings können nun Knoten mehrere

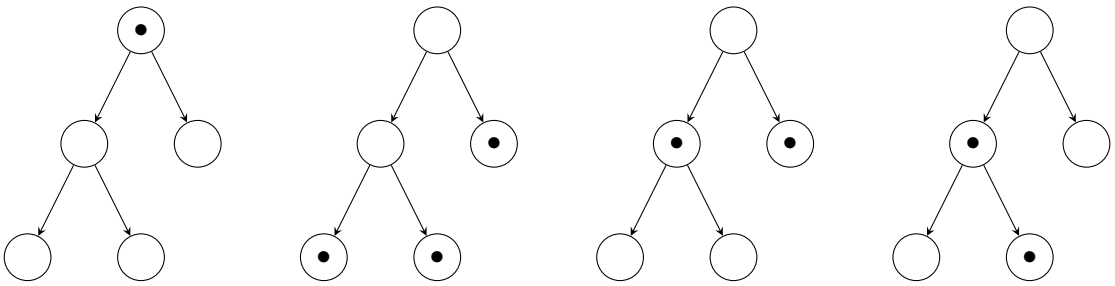


Abbildung 3.16: Drei Bäume mit gültiger und ein Baum mit ungültiger Konfiguration der Markierung

Elternknoten besitzen, weswegen in diesem Fall von einem *gerichteten, azyklischen Graphen* (directed acyclic graph, DAG) gesprochen werden muss.

3.3.10 Gültige Markierungen

Die Menge der gültigen Markierungen leitet sich unmittelbar aus den Eigenschaften der Cluster ab: Zum einen darf die Szene keine verschachtelten Meshes anzeigen, zum anderen muss jedes angezeigte Mesh mindestens eine Ladung enthalten und jede simulierte Ladung in genau einem Mesh enthalten sein. Weitere Eigenschaften wie etwa die Überschneidungsfreiheit betreffen die Form der Meshes selbst, die abzubilden ein Konturbaum nicht zum Ziel hat.

Dass zwei ineinander liegende Meshes nicht auftreten dürfen, muss sich dahingehend im Baum widerspiegeln, dass zwischen dem Wurzel- und jedem Blattknoten genau ein markierter Knoten liegt. Ein markierter Knoten darf also selbst keine mittelbaren, markierten Kindknoten besitzen und – mit Ausnahme des Wurzelknotens – ebenfalls in jedem Nachbarzweig genau einen markierten Knoten zwischen dem gemeinsamen Eltern- und jedem abstammendem Kindknoten aufweisen. Jede Konfiguration der Markierung eignet sich als Startkonfiguration, sofern sie diesen Regeln genügt.

Beispiel

Einige Beispiele für gültige Markierungen des Baums aus dem obigen Beispiel:

- $\{\{q_1^\circ, q_2^\circ\}^\circ, q_3^\circ\}'$ (Nur die Wurzel ist markiert)
- $\{\{q_1', q_2'\}^\circ, q_3'\}^\circ$ (Nur die Blätter sind markiert.)
- $\left\{\{q_1^\circ, q_2^\circ\}', q_3'\right\}^\circ$

Die Markierung $\left\{\{q_1^\circ, q_2'\}', q_3^\circ\right\}^\circ$ ist in mehrerlei Hinsicht ungültig: Zwischen Wurzel und q_3 ist kein Knoten markiert, während q_2 und sein Elternknoten markiert sind. Diese Konfigurationen sind in Abbildung 3.16 illustriert.

Die Konfiguration der Markierung des Konturbaums spiegelt den Zustand der Szene wider. Als Mittel zu ihrer Bewertung haben wir weiter oben die Pareto-Effizienz gefunden. Diese

ermöglicht aber lediglich eine ordinale Bewertung. Daher eignet sich die eine gültige Startmarkierung so gut wie jede andere.

3.3.11 Funktionen

Die Änderungen, die an der Konfiguration eines Konturbaums in eine Pareto-bessere überführen, können als Funktionen verstanden werden, die die Markierung der Knoten des einen Baums auf die eines anderen abbilden.

Die Operationen überführen die Struktur durch Änderungen in einen anderen Zustand. Dies ist nötig, wenn entweder eine Äquivalenz bzw. eine Nicht-Äquivalenz zweier Punktladungen erkannt wurde. Die beiden korrespondierenden Funktionen, die jeweils einen Zustand der Daten auf einen anderen abbilden, nennen wir f bzw. g. Diese beiden Funktionen sind vergleichbar mit den Operationen auf Meshes, die zwei Meshes mit jeweils einem Öffnungspolygon vereinen und ein Mesh auftrennen und dabei zwei Meshes mit zueinander gerichteten Öffnungspolygonen zurück lassen. f versetzt die Daten dabei in einen Zustand, der im Vergleich zum ursprünglichen Zustand das Mesh im Parameter aus- und eine Menge von Meshes eingeblendet wird. Die auslösende Fehlannahme der Äquivalenz wird dadurch aufgelöst, ohne dass der Rest der Daten beeinflusst wird. Die neue Konfiguration ist also Pareto-besser zu bewerten. g hingegen versetzt analog die Daten in einen Zustand, in dem im Vergleich zum ursprünglichen Zustand eine Menge Meshes aus- und eines eingeblendet wird. Auch hierdurch wird der zugrundeliegende Widerspruch aufgelöst, wodurch sich eine Pareto-Verbesserung ergibt. Auch wenn dieser Umstand nicht möglich ist, kann bemerkt werden, dass Operationen f und g, die die gleichen Meshes betreffen wegen $f \circ g = g \circ f = \mathrm{id}$, also nacheinander angewandt den Urzustand wiederherstellen und somit $g = f^{-1}$ ist.

Im folgenden sind die Funktionen f und g jeweils so zu bestimmen, dass:

1. sie gültige Markierungen auf gültige Markierungen abbilden und
2. die gültigen ursprüngliche Markierungen des Zielbaums Pareto-besser zu der ursprünglichen sind.

Die Funktion f formulieren wir als

$$f(\{n_1, n_2, \ldots\}') = \{n_1', n_2', \ldots\}^\circ \tag{3.15}$$

Wenn die ursprüngliche Markierung gültig ist, können wir voraussetzen, dass keine mittelbaren Kindknoten markiert sind. Nach der Operation ist die Markierung der Wurzel aufgehoben, während alle unmittelbaren Kindknoten markiert sind. Somit ist die Eigenschaft, dass zwischen dem Wurzelknoten und jedem Blattknoten genau ein Knoten markiert ist, gewahrt. Betrifft die

Funktion nur einen Teilbaum, bleibt der Rest des Baums unberührt. Somit ist die Konfiguration des Zielbaums gültig, wenn die ursprüngliche Konfiguration des Unterbaums und die des Oberbaums gültig sind. f kommt zur Anwendung, wenn eine falsch angenommene Äquivalenz erkannt wurde. Zwar kann nicht gesagt werden, ob das Bild den entsprechenden Teil der Isofläche korrekt repräsentiert, aber die Menge der zu repräsentierenden Flächen ist größer, als im Urbild. Somit stellt f eine Pareto-Verbesserung dar.

Die zu f inverse Funktion lautet $g(\{n_1, n_2, n_3, \ldots\}$ und ermöglicht in dieser Form die Reaktion auf mehrere gleichzeitig als falsch erkannte Nicht-Äquivalenzen. Da diese jedoch in unterschiedlichen Bäumen auftreten oder auf das Aktivieren eines gemeinsamen Elternelements hinauslaufen, ist es möglich, diese Funktion mit nur zwei Parametern in den Form $\{g(\{n_1, n_2\}), \ldots, g(\{n_3, n_4\}), \ldots\}$ und $g(\{n_1, g(\{n_2, n_3\})\})$ äquivalent zu verwenden, ohne die Allgemeinheit einzuschränken. g lässt sich somit formulieren als:

$$g(\{n_1, n_2\}^\circ) = \{g(n_1)^\circ, g(n_2)^\circ\}' \tag{3.16}$$

.

Die Argumentation für die Abbildung gültiger Markierungen auf gültige Markierung und die Pareto-Verbesserung lauten analog zu denen der Funktion f. Blätter b erfordern die Sonderbehandlung $g(b) = b^\circ$.

Da bei einer einwertigen Isofläche keine falsch angenommenen Äquivalenzen und bei einer diskreten Isofläche keine falsch angenommenen Nicht-Äquivalenzen denkbar sind, können die g und f für die entsprechenden Parameter, also beispielsweise g für den Wurzelknoten, undefiniert sein. Zusätzlich ist zu bemerken, dass die Sonderbehandlung $g(\varnothing)$ bei einer ungültige Konfiguration herbeiführen kann, wenn sie fehlerhaft auf ein Blatt angewendet wird, obwohl ein markiertes Blatt eine falsch angenommene Nicht-Äquivalenz ausschließt. Diese Fälle sind bei der Implementierung beispielsweise durch Ausnahmefehlerbehandlung zu berücksichtigen.

Die Funktion g muss außerdem auf dem korrekten Elternknoten der fälschlich als nicht äquivalent angenommenen Punktladungen angewendet werden. Dieser bestimmt sich über den *transponierten Graphen* (transposition graph) des Baumes bzw. des DAG – also den Graphen mit den gleichen Knoten und umgekehrten Kanten. Die Transposition eines Graphen lässt sich in einer Implementierung entweder durch die Transposition der Adjazenzmatrix [CLR09, S. 592] mit einem Aufwand von $\mathcal{O}(n^2)$ [CLR09, S. 831] oder ohne Zusatzaufwand parallel zum Konturbaum erstellen. Dass mindestens ein solcher gesuchter Elternknoten existiert, ergibt sich aus der Tatsache, dass die Funktion $\wp$ in Gleichung (3.14) auf m vollständig definiert ist. In einem Baum kommen die Punktladungen allerdings mehrfach vor, bzw kann ein DAG mehrere

Elternelemente haben, wohingegen neben dem Ausblenden der Meshes fälschlich als nicht äquivalent angenommenen Punktladungen immer genau ein Mesh eingeblendet werden muss.

- $\{a\} = \wp M$ gilt nicht, da nach Voraussetzung a nicht in M enthalten ist. Somit ist a auch nicht in $\wp M$ enthalten und somit können die beiden Mengen erst recht nicht gleich sein.
- $\{a\} = \{a\} \times \wp M$, wenn $\wp M = \varnothing$. Die leere Menge ist allerdings nicht im Wertebereich von $\wp$ enthalten.
- $\wp M = \{a\} \times \wp M$, wenn die Menge $\{a\}$ leer ist. Sie enthält aber immer a.

Der gesuchte Elternknoten existiert also in jedem Fall und ist eindeutig. Eine mögliche Implementierung ist in Alg. 6 angegeben. Seine Terminierung folgt aus der Existenz des Elternelements, die Korrektheit aus seiner Eindeutigkeit.

Algorithmus 6 : Bestimmung des gemeinsamen Elternelements zweier Knotenobjekte n_1 und n_2 in einem DAG

eingabe : Knoten des transponierten Graphen n_1, n_2
ausgabe : Elternknotenobjekt p

```
1 für jedes Elternknotenobjekt p' von n_2 tue
2     für jedes Kindknotenobjekt c von p' tue
3         wenn c = n_1 dann
4             p ← p';
5             Fertig;
6         Ende
7     Ende
8 Ende
```

Beispiel

Das letzte der obigen Beispiele wurde durch f gewonnen:

$$f\left(\left\{\left\{q_1^\circ, q_2^\circ\right\}^\circ, q_3^\circ\right\}'\right) = \left\{\left\{q_1^\circ, q_2^\circ\right\}', q_3'\right\}^\circ$$

3.3.12 Komplexitätsanalyse

Die Funktion f benötigt für die intendierte Funktionsweise denjenigen Knoten, der die fälschlich als äquivalent angenommenen Ladungen enthält. Das auszublendende Mesh wird durch ebendiesen Knoten, die einzublendenden Meshes als dessen Kindknoten repräsentiert. Die Anzahl der unmittelbaren Kinder eines Knotens im Konturbaum hängt von der Anzahl der vom entsprechenden Mesh vor dem Ausblenden umfassenden Punktladungen ab: Über die Äquivalenzen der enthaltenen Punktladungen liegen noch keine Informationen vor, allerdings ist sicher, dass die aufgrund der vorliegenden Nicht-Äquivalenz jedes einzublendende Mesh

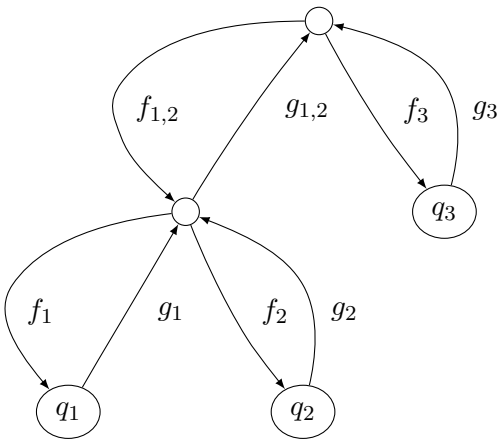


Abbildung 3.17: Diagramm eines Baums für die Ladungsverteilung in Abbildung 4.3

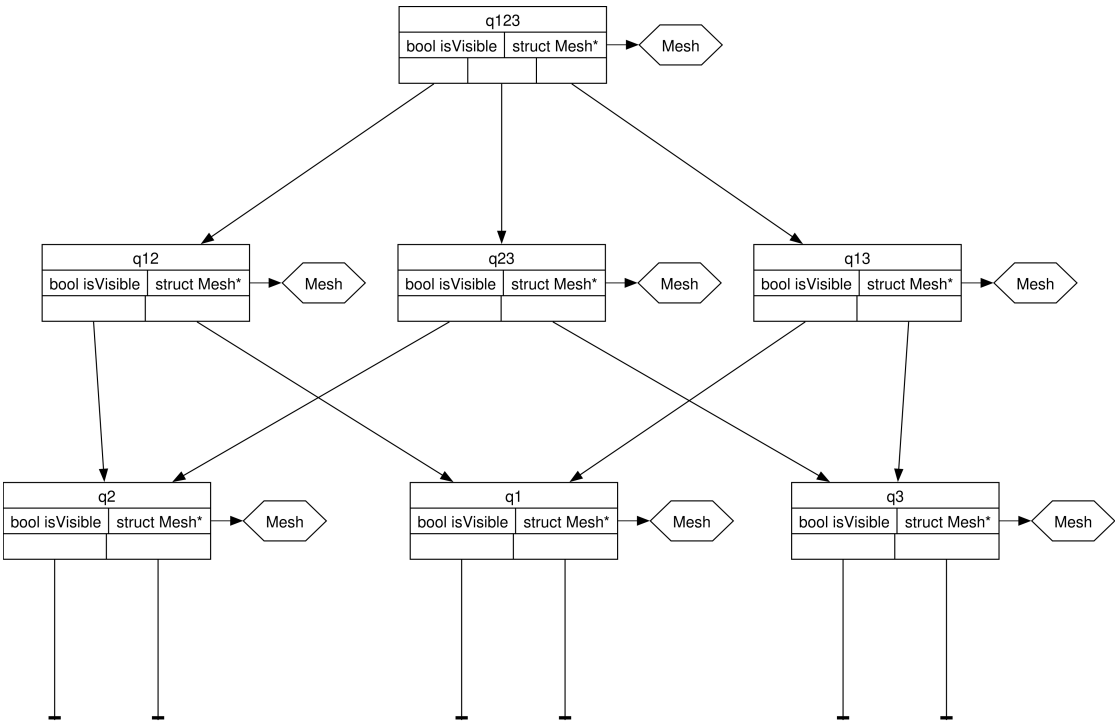


Abbildung 3.18: Zeiger-Diagramm für einen DAG, der das Beispiel in Abbildung 4.3 abbildet.

eine Ladung weniger enthalten muss. Die Anzahl der Kindknoten entspricht also der Anzahl der Ladungen n, womit sich die Zeitkomplexität auf $\Theta(n)$ beläuft.

Die Funktion g durchläuft die beiden als Parameter angegebenen Unterbäume, was eine Zeitkomplexität von $\Theta(n)$ bedeutet [CLR09, S. 288]. Der Zusatzaufwand der Bestimmung des Elternelements beläuft sich auf $\mathcal{O}(n^2)$ für n Punktladungen, was somit den Gesamtaufwand für g ergibt.

Diese Befunde werfen die Frage auf, ob eine Simulation so gestaltet werden kann, dass die Verwendung der komplexeren Funktion g ganz vermieden werden kann. Für statische Ladungsverteilungen lässt sich leicht einsehen, dass $g(x)$ und $f(x)$ für den Knoten x nicht beide eine Pareto-Verbesserung darstellen können. Auch zeigt ein obiges Beispiel, dass jeder Konturbaum gültig markiert ist, wenn nur das Wurzelelement markiert ist. Daraus folgt, dass in einem Konturbaum für eine statische Ladungsverteilung, in dem nur das Wurzelelement Markiert ist, g nie eine Pareto-Verbesserung beitragen kann, ihre Verwendung sich also vermeiden lässt. Für dynamische Ladungsverteilung gilt dieses Argument nicht: Durch eine Veränderung der Position oder Stärke der Punktladungen können sich Fehlannahmen bezüglich der Nicht-Äquivalenz von Punktladungen ergeben, die durch g behandelt werden müssten. Hier kann aber die oben getroffene Erkenntnis $g = f^{-1}$ verwendet werden, indem ähnlich der Memoisation f die Markierung des Baums für einen etwaigen Aufruf von g zwischenspeichert. Ist in dem Konturbaum initial nur das Wurzelelement markiert, ist die Existenz eines solchen zwischengespeicherten Ergebnisses garantiert. Somit wäre auch hier die Verwendung von g vermeidbar.

Der Term $\wp M$ tritt in der Gleichung (3.14) zwar mehrfach auf, würde aber in einer Implementierung nur einmal berechnet und das Ergebnis mehrfach verwendet werden. Jeder Rekursionsschritt berechnet also ein eigenes Unterproblem. Als Zusatzaufwand entsteht im dritten Term durch das Kreuzprodukt ein Aufwand von $\mathcal{O}(n)$. Die Verknüpfung eines Elements mit einer Menge erfolgt pro Aufruf in konstanter Zeit. Die Differenzengleichung der Bestimmung der Potenzmenge ergibt sich also zu

$$T(n) = 2 \cdot T(n-1) + 1 \tag{3.17}$$

In der allgemeinen Form für k Rekursionsschritte ist $T(n) = 2^k \cdot T(n-k) + n$, wobei im Basisfall $T(1) = 1$. Somit ist bei Rekursionsabbruch $n - k = 1$ wodurch in diesem Fall $n = k + 1$. Nach Einsetzen in Gleichung (3.17) erhalten wir $T(n) = 2^k \cdot T(1) + n = 2^n \cdot 1 + n$. Das Verfahren ist also in der Komplexitätsklasse $\Theta(2^n)$. Mit analoger Argumentation lässt sich feststellen, dass diese auch die Komplexitätsklasse für den Speicheraufwand darstellt [Goo21].

Dieser Aufwand kann entweder vorbereitend vollständig erzeugt oder beim Erkennen einer falschen Annahme über das Äquivalenzverhältnis zweier Punktladungen mit dem Aufwand

$\mathcal{O}(n)$ berechnet werden. In diesem Fall kann auch der Speicherplatz für ausgeblendete Meshes freigegeben werden, womit eine Speicherkomplexität von ebenfalls $\mathcal{O}(n)$ entsteht.

3.3.13 Artefakte und Homogenität

Aufgrund der Arbitrarität der Cluster kann die Größe der Meshes in einer Simulation und somit eine angemessene Dichte der Vertices schlecht abgeschätzt werden, weswegen abhängig vom Ansatz der Grundgeometrie das Mesh bereichsweise „zu fein“ oder „zu grob“ empfundene Auflösung aufweisen kann. Das Auftreten dieses Effekts ist mit dem des ansteigenden Abstands zwischen Feldlinien bei steigendem Isowert vergleichbar. Abhilfe kann hier das Aufteilen und Zusammenführen von Dreiecken bieten. Außerdem können während einer Simulation – besonders wenn auf Erkennen falsch angenommener Äquivalenzverhältnisse zweier Punktladungen mit selektivem Einblenden reagiert wird – zum Auftreten von Artefakten kommen, die der Eignung eines Meshes, die Fläche einer Isofläche zu repräsentieren, beeinträchtigen. In den folgenden Erläuterungen führen wir Begriffe zur Beurteilung der Verteilung von Vertices auf einem Mesh ein und qualifizieren diese, indem wir variierende Dichte und das Auftreten von Artefakten als Eigenschaften der nach dieser Metrik als schlecht zu bewertenden Meshes aufzeigen. Schließlich leiten wir geeignete Operationen auf Meshes her, den so aufgedeckten Problemen entgegenzuwirken. Dazu machen wir uns die monoidale Struktur des Meshes nutzbar, durch die wir Dreiecke eines triangularen Meshes ersetzen können, ohne die Sauberkeit des Meshes zu stören.

3.3.13.1 Inhomöogenitäten

Eine Definition für ein isotropes Mesh hält das Gebiet der isotropen Neuvermaschung von Oberflächen (*isotropic surface remeshing* nach [Kha+22; Xu+19]) bereit, die ein triangulares Mesh mit einer Flächenliste, die möglichst gleichseitige Dreiecke enthält, bezeichnet. Im Folgenden wollen wir allerdings Untersuchungen anstellen, die über die Begriffe in dieser Definition hinausreichen, alleine schon, weil wir uns in diesem Teil der Arbeit nicht auf triangulare Meshes beschränken wollen. Daher führen wir geeignetere Begriffe ein, anhand derer wir die Analyse der Auswirkungen durchführen, die eine Simulation eines elektrischen Feldes auf ein Mesh, das eine Fläche repräsentieren soll, hervorrufen kann.

Homogenität, Homöogenität

Ein reguläres Mesh ist *homogen*, wenn alle Flächen dieses Meshes die gleiche Kantenlänge aufweisen. Variieren die Kantenlängen innerhalb eines festzulegenden Intervalls, ist das Mesh *homöogen*. Abweichende oder nicht in diesem Intervall enthaltene Kantenlängen bezeichnen wir als *inhomogen*. Ebenso ist kontextabhängig ein sauberer und regulärer Teil des Meshes oder das gesamte Mesh inhomogen. Abweichende Kanten oder saubere und reguläre Teile des Meshes nennen wir auch *Inhomogenität*.

Das Adjektiv „homöogen“ ist steigerbar. Ein Mesh bezeichnen wir als umso homöogener, je gleichmäßiger die Vertices auf der Oberfläche verteilt sind. Ein hohes Maß an Homöogenität ist wünschenswert, da sie bei einer etwaigen Texturierung der Meshes das Generieren von UV-Maps erleichtern und insgesamt einen optisch besseren Eindruck des dargestellten Meshes liefern. Ein niedriges Maß an Homöogenität hingegen zeichnet sich durch stark variierende Dichte von Vertices über dem Mesh aus. In diesem Fall ist zu erwarten, dass auf einem Teil des Meshes mit vielen Vertices der Informationsgehalt eines einzelnen Vertex stark abgenommen hat, wodurch sich nicht zu rechtfertigender Ressourcenverbrauch einstellt. Außerdem kann ein hohes Maß an Wechsel zwischen den Dichten der Vertices optisch störend wirken – die Bereiche vergleichsweise niedriger Vertex-Dichte als „Artefakt“ wahrgenommen werden. Auch steht uns ein Ansatz wie etwa in [KBS00] nicht zu Verfügung, eine Oktonärbaum-ähnliche Datenstruktur zur Speicherung von Meshes in verschiedenen Auflösungen zu verwenden, da dieses abgesehen vom erhöhten Speicherverbrauch der bei jeder Veränderung der Äquivalenzverhältnisse neu berechnet werden müsste und die Problematik der inhomogenen Gestalt der erzeugten Geometrie fort tragen würde. [DFPS00] stellen fassen einige weitere Neuvermaschungsverfahren zusammenfassend vor, die aber zumeist auf einem gesamten statischen Mesh arbeiten und ein statisches Endergebnis liefern, also keine kleineren, zielgerichteten Veränderungen vornehmen oder andere nicht akzeptable Nachteile wie eine hohe Laufzeit aufzählen. Selbst in Verfahren, die auf der Idee einer linearen Trajektorie aufbauen, profitieren wir also von Mitteln, Bereiche geringer Homogenität zu erkennen und von Operationen, diese zu steigern. Am geeignetsten scheint der Ansatz der quaternären Triangulation (*quaternary triangulation*) aus [DFPS00, Ff. 28f] zu sein, für den die Autoren allerdings keine auffindbaren Quellen liefern, die die skizzierten Vorgänge detailliert genug erläutern, um eine Analyse des Verfahrens anzustrengen. Wir erlauben uns daher, uns nur den Ansatz dieser Idee zu übernehmen und diesen nach unseren Bedürfnissen anzupassen und zu erweitern.

Im primitiven Fall einer einzelnen Punktladung, die von einer Ico- oder UV-Sphere umgeben wird und auf der die Vertices entlang von Halbgeraden, die sich durch Gleichung (2.1) mit dem Vertex selbst und der Position der Punktladung konstruieren lässt, wird eine Simulation die Homogenität nicht beeinträchtigen. Vielmehr wird sie ein von bestimmten Inhomogenitä-

ten behaftetes Mesh in einen homogenen Zustand überführen, nämlich von solchen, die aus ungleichmäßigem Translatieren der Vertices auf der Halbgerade resultieren. Die Motivation, auf eine höhere Homogenität des Meshes hinzuwirken sinkt also mit dem Isowert. Mit dem Begriff der Homogenität lässt sich an dieser Stelle korollarisch verstehen, warum Isoflächen bei sinkendem ξ eine Kugelform annehmen: Bei $\xi = 0$ haben alle Punkte unendlich großen und somit gleichen Abstand zueinander und somit ist das Mesh homogen. Sie entsprechen somit den Vertices eines gleichseitigen Polyeders oder, wenn die Anzahl der Vertices dieses Polyeders gegen Unendlich geht, einer Kugel. Sehr große ξ rufen Situationen hervor, in denen alle Punktladungen von einer Fläche repräsentiert werden. Analog zum trivialen Fall ist auch hier mit kugelförmigen homöogenen Meshes zu rechnen. Aus diesen Gründen bieten sich kugelförmige Meshes – bekannte Beispiele sind UV-Spheres als quadrilaterale und Icospheres als triangulare, kugelförmige Meshes – als Grundform für Startgeometrien in Simulationen an: In den beschriebenen Fällen wird eine Simulation die anfängliche Homogenität stören, nicht aber die Sauberkeit, Geschlossenheit oder Triangularität bzw. Quadrilateralität, weil lediglich Transformationen auf die Vertices angewendet werden.

3.3.13.2 Auflösung des Meshes

Ein Polygon kann als zweidimensionales Mesh aufgefasst werden. Dieses kann eine zweidimensionale Fläche einer Isofläche für ein zweidimensionales Skalarfeld repräsentieren. Sollte nach einer beliebigen Maßgabe dieses Polygon „zu wenige“ Vertices beinhalten, um die Fläche angemessen zu repräsentieren, kann die Auflösung durch Aufteilen der Kanten, einfügen eines neuen Punkts auf der Mitte der Kante und Verbinden der beiden Punkte der ursprünglichen Kanten mit dem neuen Punkt erhöht werden. Auf analoge Weise kann die Auflösung auch reduziert werden.

In Situationen, in denen Fläche mehrere aber nicht alle Ladungen beinhaltet – vor allem in solchen, in denen keine vollständige Heuristik zur Ermittlung der Äquivalenzklassen vorliegt – ist nicht mit gleichmäßigen Veränderungen der Meshes während der Simulation zu rechnen. Daher wird dass das Mesh im Verlauf Flächen hoher und niedriger Feinheit aufweisen. Wir wollen die Argumentation unabhängig von einer Entscheidung über die Maßnahme der lokalen Feinheit eines Meshes fortführen, weil uns einerseits die Möglichkeiten dafür zu mannigfaltig erscheinen, wir ihr andererseits keine allzu große Bedeutung beimessen. Wir führen aber einige einfache Beispiele an, die dieser Aufgabe gerecht werden und in konstanter Zeit berechenbar sind:

Beispiele

- Ober- und Untergrenzen für Kantenlänge der Flächen oder ihrer Flächeninhalte bei triangularen oder quadrilateralen Meshes geben ein gutes Maß für die lokale Vertexdichte eines Meshes. Diese Größen sind durch eine Vektoraddition bzw. durch Bilden eines Skalar- und eines Kreuzprodukts – Operationen mit konstanter Laufzeit – berechenbar [But11, S. 957].
- Das Volumen des minimalen achsenparallelen die Geometrie umgebenden Quaders (*minimum AABB*) für die jeweiligen Dreiecke im Fall eines triangularen Meshes. Diese ist durch einige Vergleiche und Fließkommamultiplikationen zu berechnen. Für eine feste Wortbreite der Fließkommazahl lassen sich diese Berechnungen ebenfalls in konstanter Zeit berechnen.

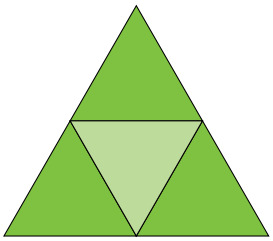
Für den jeweiligen Bedarf lassen sich andere Rahmengeometrien generieren und gegebenenfalls in einer höheren Komplexitätsklasse berechnen [OSu+01, S. 9f].

Ist das entsprechende Kriterium für ein Dreieck nicht bzw. nicht mehr erfüllt, ist während der Simulation dieses Dreiecks für nach Maßgaben dieses Kriteriums „zu groß“ oder „zu klein“ gewesen bzw. geworden. UV-Spheres erlauben Subdivision auf eine einfache Art und Weise. Da es sich bei Icospheres um geodätische Polygone handelt, ist auch auf diesen eine Subdivision-Operation möglich, die wiederum in einer Icosphere resultiert [Kob+00]. Verschiedene Implementierungen wie [Hav08] und [Kah09] realisieren diese Lösung mit einer Laufzeitkomplexität von $\mathcal{O}(n^2)$. Auf diese Weise kann der Effekt eines Meshes erreicht werden, das sich beim Anwachsen automatisch verfeinert.

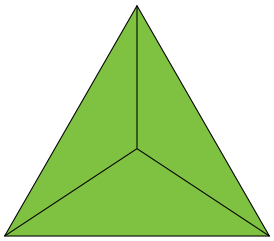
Auch Faces lassen sich dieser Behandlung unterziehen: Wird in einer statischen Simulation ein Dreieck „zu groß“, kann das Dreieck gemäß dem in Abbildung 3.19a abgebildeten Schema aufgeteilt werden: Auf diese Weise entstehen vier dem ursprünglichen Dreieck und untereinander ähnliche Dreiecke. Die Reduktion von Dreiecken nach diesem Schema erfordert die Kollinearität aller drei Punkte auf den Kanten des Zieldreiecks. Ist also der Erhalt der Ähnlichkeit der Dreiecke auf diese Weise während einer statischen Simulation ein Ziel, sollte die Startgeometrie möglichst klein gewählt werden, um Reduktionen unnötig zu machen. In dynamischen Simulationen kann auf das umkehrbare Schema in Abbildung 3.19b zurückgegriffen werden. Es folgt dem Vorschlag von [DF+99, F. 2] als einzigem, der einzelne Dreiecke gezielt und auf umkehrbare Weise verarbeitet. Darüber hinaus bilden Dreiecke bezüglich dieser Operation einen Monoiden. Allerdings ist dieses Verfahren nicht einsetzbar, wenn die Ergebnisgeometrie aus dem ursprünglichen Dreieck ähnlichen Dreiecken besteht oder das resultierende Mesh regelmäßig sein muss.

Beide Operationen haben eine Laufzeitkomplexität von $\mathcal{O}(1)$, allerdings muss bei der Umkehrung ein Zeit- bzw. Speicheraufwand von $\mathcal{O}(n)$ aufgebracht werden, um zu erkennen, ob das Dreieck einen Punkt mit zwei anderen Dreiecken teilt, bzw. diejenigen Punkte zwischenzuspeichern, die durch das Aufteilen hinzugekommen sind.

Die in diesem Abschnitt vorgestellten Verfahren operieren durch direkte Modifikationen des Meshes und erfordern dafür einen zusätzlichen Speicheraufwand von $\mathcal{O}(1)$. Auf AR-Hardware mit geringem Arbeitsspeicher können betrachterabhängige Mesh-Vereinfachungen wie die aus [ESC00] zum Einsatz kommen. Die Autoren verzichten auf eine Komplexitätsanalyse, stützen sich aber auf die Voronoi-Triangulation, die eine Laufzeitkomplexität von $\mathcal{O}(n \log n)$ aufweist [Ver19]. Die Ergebnisse des Laufzeitexperiments von bis zu knapp vier Minuten während der Vorverarbeitungsphase sind auf dem Hintergrund der von den Autoren verwendeten Hardware (Workstation mit höchstens 400 MHz und höchstens 128 MB freiem Arbeitsspeicher) zu beurteilen.



(a) Nicht-invertierbare Aufteilung eines Dreiecks in ähnliche Dreiecke.



(b) Invertierbare Aufteilung eines Dreiecks in nicht-ähnliche Dreiecke.

KAPITEL 4

Bestimmung der Isopotentialflächen in polynomialer Zeit

Aus den obigen Abschnitten wird deutlich, dass eine algebraische Berechnung der Positionen eines Vertices auf einer Bewegungshalbgeraden oder ihre Bestimmung durch einen Greedy-Algorithmus in linearer Zeit nicht möglich ist. Es stellt sich also die Frage, ob diese Ergebnisse in polynomialer Zeit erreicht werden können. Dieses Kapitel will diese Frage positiv beantworten. Dafür setzen wir Algorithmen ein, die das Problem der Berechnung von Vertices von Meshes, die Isoflächen repräsentieren, durch Simulation lösen. Dieses Kapitel stellt jeweils einen Algorithmus:

1. zur Berechnung des Feldes einer Punktladung,
2. zur Berechnung eines Feldes aus mehreren Punktladungen,
3. zur Generierung eines eine dreidimensionale Isofläche eines Feldes aus mehreren Punktladungen repräsentierenden Meshes durch Marching Cubes,
4. zur Generierung eines eine zweidimensionale Isofläche eines Feldes aus mehreren Punktladungen repräsentierenden Polygonen durch schrittweise Umläufe und
5. zur Generierung eines eine dreidimensionale Isofläche eines Feldes aus mehreren Punktladungen repräsentierenden Meshes durch Annäherung in einer Simulation

vor. Wir leiten diese Algorithmen jeweils aus den Grundlagen her und gleichen diese, wenn anwendbar, gegen die Vorgaben aus Abschnitt 3.2 ab. Anschließend beweisen wir jeweils deren Korrektheit und Terminierung und analysieren ihre Laufzeit.

4.1 Elektrisches Feld einer Punktladung

In Abschnitt 2.4 haben wir erläutert, dass das elektrische Feld berechnet wird, indem die Beiträge aller Punktladungen summiert werden. Das Feld an einem Punkt $\mathbf{p}$ einer einzelnen Punktladung berechnet sich gemäß Gleichung (2.4) mit:

$$\mathbf{E}(\mathbf{p}) = \frac{q}{4\pi \left\|\mathbf{p}\right\|^3}\mathbf{p}$$

.

Algorithmus 7 : Berechnung des elektrischen Felds einer einzelnen Punktladung am Aufpunkt

eingabe : Punktladung $q \in \mathbb{R}^3 \times R$, Aufpunkt $\mathbf{p} \in \mathbb{R}^3$
ausgabe : Ergebnisvektor $\mathbf{e} \in \mathbb{R}^3$

1 **wenn** $q_q = 0$ **dann**
2 $\quad \mathbf{e} \leftarrow \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$;
3 **Ende**
4 **sonst**
5 $\quad \mathbf{d} \leftarrow \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$; `// Initialisierung`
6 $\quad \mathbf{d} \leftarrow \mathbf{p}$;
7 $\quad \mathbf{d} \leftarrow \mathbf{d} - \mathbf{q}_p$;
8 $\quad \mathbf{a} \leftarrow \|\mathbf{d}\|$;
9 $\quad n \leftarrow \frac{1}{a^3}$;
10 $\quad \mathbf{e} \leftarrow \mathbf{d}$;
11 $\quad \mathbf{e} \leftarrow \mathbf{e} \cdot q_q \cdot n$;
12 **Ende**

Ein einfacher Algorithmus zur Berechnung ist in Alg. 7 angegeben. Die Terminierung folgt aus der Abwesenheit von Kontrollstrukturen innerhalb der bedingten Verzweigung. Die Korrektheit wird durch das Hoare-Kalkül bewiesen. Zu beweisen ist das Hoare-Tripel:

$$\langle\!| \top |\!\rangle \quad S \quad \langle\!| \mathbf{e} = \frac{q_q}{\|p - q_p\|} |\!\rangle$$

mit S als Alg. 7:

- Korrektheit des Else-Zweiges:
  $\langle\!| q_q \neq 0 |\!\rangle$
  Nach Stärken der Vorbedingung durch das Idempotenzgesetz ($\top \wedge a = a$):
  $\langle\!| \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p} - q_p) = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p} - q_p) \wedge q_q \neq 0 |\!\rangle$

Die Anweisung $\mathbf{d} \leftarrow \mathbf{0}$ dient der Initialisierung, und verändert somit die Zusicherung nicht:
$\{\!| \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) \wedge q_q \neq 0 |\!\}$
$\mathbf{d} \leftarrow \mathbf{p}$
$\{\!| \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{d}-q_p) = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) \wedge q_q \neq 0 |\!\}$
$\mathbf{d} \leftarrow \mathbf{d} - \mathbf{q}_p$
$\{\!| \frac{q_q}{\|d\|^3}\mathbf{d} = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) |\!\}$
$\mathbf{a} \leftarrow \|\mathbf{d}\|$
$\{\!| \frac{q_q}{a^3}\mathbf{a} = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) |\!\}$
Nach Anwenden der Erweiterungsregel für Brüche:
$\{\!| q_q\frac{1}{a^3}\mathbf{d} = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) |\!\}$
$n \leftarrow \frac{1}{a^3}$
$\{\!| q_q \cdot n\mathbf{d} = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) |\!\}$
$\mathbf{e} \leftarrow \mathbf{d}$
$\{\!| q_q \cdot n\mathbf{e} = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) |\!\}$
$\mathbf{e} \leftarrow q_q \cdot n\mathbf{e}$
$\{\!| \mathbf{e} = \frac{q_q}{\|\mathbf{p}-q_p\|^3}(\mathbf{p}-q_p) |\!\}$

- Korrektheit des If-Zweiges:
  $\{\!| \top |\!\}$
  Nach Stärken der Vorbedingung:
  $\{\!| \mathbf{e} = (\mathbf{p}-q_p) \cdot q_q \cdot \frac{1}{\|\mathbf{p}-q_p\|^3} |\!\}$
  $\mathbf{e} \leftarrow \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$
  $\{\!| \mathbf{e} = (\mathbf{p}-q_p) \cdot q_q \cdot \frac{1}{\|\mathbf{p}-q_p\|^3} |\!\}$
  Nach Voraussetzung in der Verzweigung gilt $q_q = 0$. Unter Anwendung des Skalarproduktes:
  $\{\!| \mathbf{e} = \mathbf{0} |\!\}$

4.1.1 Zeit- und Speicherkomplexität

Für die Analyse der Laufzeit betrachten wir den Wenn-Zweig S_w des Alg. 7 und dessen Sonst-Zweig S_s getrennt. Beide sind Folgen arithmetischer Fließkommaoperationen ohne Kontrollstrukturen. Sie können nach Regel 4 in Abschnitt 2.8 für das Zeitkomplexität als eine Konkatenation beider Zweige S_wS_s angesehen werden. Sie weist nach Regel 1 unabhängig von der Anzahl der Fließkommaoperationen eine Zeitkomplexität von $\mathcal{O}(W+S)$ auf. Nach Regel 2 gilt $\mathcal{O}(W+S) \in \mathcal{O}(1+1) \in \mathcal{O}(1)$.

Der Algorithmus verwendet eine endliche Menge an Fließkommavariablen, weswegen beide eine Speicherkomplexität von $\mathcal{O}(1)$ aufweisen.

4.2 Elektrisches Feld mehrerer Punktladungen

Das Gesamtfeld an einem Punkt $\mathbf{p}$ berechnet sich nun gemäß Gleichung (2.5) durch die Summierung mehrerer Punktladungen über Gleichung (2.4):

$$\mathbf{E}(\mathbf{r}) = \sum_i \frac{q_i}{4\pi \left\|\mathbf{r}_i - \mathbf{q}_q\right\|^3} (\mathbf{r}_i - \mathbf{q}_q)$$

über eine Menge von Punktladungen $\{(\mathbf{q}_1, q_1), (\mathbf{q}_2, q_2), \ldots\}$. Es handelt sich hierbei um eine indizierte Menge, die somit auch als Folge wohlunterscheidbarer Elemente betrachtet werden kann. Als solche ist sie einer Iteration durch Schleifen zugänglich. Folgen würden in einer Implementierung als Liste oder Array gespeichert. Moderne Programmiersprachen wie z. B. JavaScript erlauben für eine Iteration über diese Datenstrukturen spezielle Kontrollstrukturen wie „for each“. Diese erschweren die Beweisführung und werden daher in dieser Arbeit als einfache Zählschleifen umformuliert. Gleichung (2.5) lässt sich auf diese Weise durch die Summe auf das Feld einer einzelnen Punktladung zurückführen:

$$\mathbf{D}_{\text{Summe}}(\mathbf{p}) = \sum_{j=1}^{j} \mathbf{E}\left(\mathbf{p}, q_j\right) \tag{4.1}$$

Die Gleichung (4.1) lässt sich anhand Alg. 8 implementieren.

Algorithmus 8 : Berechnung des gesamten elektrischen Feldes aus einer Folge von Punktladungen

eingabe : Folge von Punktladungen $Q \in \left(\mathbb{R}^3 \times \mathbb{R}\right)^n$, Aufpunkt $\mathbf{p} \in \mathbb{R}^3$
ausgabe : Ergebnisvektor $\mathbf{e} \in \mathbb{R}^3$

```
1 e ← 0;
2 i ← 1;
3 solange i ≤ #Q tue
4 |   a ← E(p, q_i); /* Ergebnis des Algorithmus 7                    */
5 |   e ← e + a;
6 |   i ← i + 1;
7 Ende
```

Die Terminierung des Alg. 8 erfolgt, sobald die Bedingung der solange-Schleife nicht mehr erfüllt ist, also die Aussage $\overline{i \leq \#Q}$ wahr ist. Durch das Hoare-Kalkül lässt sich zeigen, dass im Schleifenrumpf i streng monoton anwächst und somit nach abzählbar vielen Schritten die Konstante $\#Q$ übersteigt. Die Vorbedingung $(\!|\, i = c \,|\!)$ für ein beliebiges konstantes $c \in \mathbb{N}$ kann

stellvertretend für einen beliebigen Schleifendurchlauf, zu dem die Schleifenbedingung noch gilt, als gültig angesehen werden, während die Nachbedingung $(\!| i > c |\!)$ zu beweisen ist.

$(\!| i = c |\!)$
$a \leftarrow \mathbf{E}(\mathbf{p}, q_i)$
$\mathbf{s} \leftarrow \mathbf{s} + \mathbf{E}_l$
$(\!| i = c |\!)$
$i \leftarrow i + 1$
$(\!| i = c + 1 |\!)$

Die Nachbedingung können wir zu $(\!| i > c |\!)$ abschwächen, was der Forderung entspricht. Am Schlüsselwort „Ende“ hält i den Wert für den Anfang des nächsten Schleifendurchlaufs. Ist Q eine endliche Menge, dann ist auch $\#Q$ endlich, wodurch nach endlich vielen Wiederholungen die Aussage $c > \#Q$ gilt. Diese widerspricht aber der Schleifen-Bedingung, wodurch diese abbricht. Überdies ist das Ende der Anweisungsfolge erreicht, was die Terminierung des Algorithmus bedingt.

Die Korrektheit des Schleifenrumpfs wurde bereits oben gezeigt und kann für die folgenden Überlegungen vorausgesetzt werden. Der Algorithmus endet mit dem Ende der Schleife, wodurch die Gültigkeit der Schleifeninvarianten zum Ende des Algorithmus aus ihrer Gültigkeit nach dem letzten Schleifendurchlauf folgt.

Für den Schleifenkopf bleibt anhand einer Schleifeninvariante I zu zeigen, dass

$$(\!| I \wedge (\mathbf{e} = 0) |\!) \quad S \quad (\!| I \wedge \mathbf{e} = \sum_{j=1}^{\#Q} \mathbf{E}\,(\mathbf{p}, q_j) |\!)$$

mit S als Alg. 8 ein gültiges Hoare-Tripel ist. $\mathbf{e} = 0$ ergibt sich aus der Initialisierung vor der Schleife. Eine mögliche Schleifeninvariante ist mit

$$I = \left(\sum_{j=1}^{l} \mathbf{E}\,(\mathbf{p}, q_j) = \left[\sum_{j=1}^{l-1} \mathbf{E}\,(\mathbf{p}, q_j) \right] + \mathbf{E}\,(\mathbf{p}, q_l) \right)$$

gefunden. Das Gleichheitszeichen außerhalb der Klammern ist hier als „Gleichheit der Aussagen“ zu verstehen. Gemäß Abschnitt 2.7.4 schränkt diese Wahl die Allgemeinheit nicht ein. Zu zeigen ist nun die Korrektheit der Schleifeninvariante vor, während und nach der Schleife sowie ihre Abbruchbedingung.

Vor der Schleife

$(\!| I |\!)$
$i \leftarrow 1$

$\mathbf{e} \leftarrow 0$
$\{\!| \sum_{j=1}^{1} \mathbf{E}(\mathbf{p}, q_j) = \left[\sum_{j=1}^{0} \mathbf{E}(\mathbf{p}, q_j)\right] + \mathbf{E}(\mathbf{p}, q_1) \wedge i = 1 |\!\}$
Der Term $\sum_{j=1}^{0} \mathbf{E}(\mathbf{p}, q_j)$ stellt eine leere Summe dar und entfällt [Bun03, S. 7]. Der linke Teil der Gleichung vereinfacht sich nach Aufsummieren zu

$$\sum_{j=1}^{1} \mathbf{E}(\mathbf{p}, q_j) = \mathbf{E}(\mathbf{p}, q_1)$$

womit schließlich der initiale Wert für den ersten Schleifendurchlauf berechnet ist:
$\{\!| \sum_{j=1}^{1} \mathbf{E}(\mathbf{p}, q_j) = \mathbf{E}(\mathbf{p}, q_1) |\!\}$

Während der Schleife

Aus dem vorherigen Punkt ist bekannt, dass

$$\sum_{j=1}^{1} \mathbf{E}(\mathbf{p}, q_j) = \left[\sum_{j=1}^{1-1} \mathbf{E}(\mathbf{p}, q_j)\right] + \mathbf{E}(\mathbf{p}, q_1)$$

so dass die Invariante

$$\sum_{j=1}^{i} \mathbf{E}(\mathbf{p}, q_j) = \left[\sum_{j=1}^{i-1} \mathbf{E}(\mathbf{p}, q_j)\right] + \mathbf{E}(\mathbf{p}, q_i)$$

mit $i \geq 1$ für den i-ten Schleifendurchlauf vorausgesetzt werden kann. Die Gültigkeit der Invariante ist nun für $i' := i + 1$ zu zeigen. Die Gleichung kann zu $i + 1$ erweitert werden. Nach Einsetzen von i' erhalten wir:

$$\begin{aligned} \sum_{j=1}^{i+1} \mathbf{E}(\mathbf{p}, q_j) &= \left[\sum_{j=1}^{i+1-1} \mathbf{E}(\mathbf{p}, q_j)\right] + \mathbf{E}(\mathbf{p}, q_{i+1}) \\ = \sum_{j=1}^{i'} \mathbf{E}(\mathbf{p}, q_j) &= \left[\sum_{j=1}^{i'-1} \mathbf{E}(\mathbf{p}, q_j)\right] + \mathbf{E}(\mathbf{p}, q_{i'}) \end{aligned}$$

was der Schleifeninvariante entspricht.

Nach der Schleife

Nach Abbruch der Schleife muss die Vorbedingung mit $\overline{i \leq \#Q}$ konjugiert werden.
$\{\!| \sum_{j=1}^{i} \mathbf{E}(\mathbf{p}, q_j) = \left[\sum_{j=1}^{i-1} \mathbf{E}(\mathbf{p}, q_j)\right] + \mathbf{E}(\mathbf{p}, q_i) \wedge \overline{i \leq \#Q} |\!\}$
Die Gültigkeit der Schleifeninvarianten ergibt sich aus ihrer Gültigkeit nach dem letzten Schleifendurchlauf. Die Nachbedingung der Schleife wird als Vorbedingung für den folgenden Schritt zu $i = \#Q + 1$ verstärkt. Nach Ende des Algorithmus steht auch der Endzustand mit $\mathbf{s} = \sum_{j=1}^{i} \mathbf{E}(\mathbf{p}, q_j)$ fest. Durch Einsetzen der verstärkten Abbruchbe-

dingung der Schleife bleibt als Nachbedingung:
$(\!| \sum_{j=1}^{\#Q} \mathbf{E}(\mathbf{p}, q_j) |\!)$

4.2.1 Zeit- und Speicherkomplexität

Alg. 8 unterteilt sich in die Teile S_1 vor der Schleife und S_2 als die Schleife selbst. Während S_1 lediglich eine reine Konkatenation von Fließkommaoperationen ist und daher analog zur Argumentation in Abschnitt 4.1.1 unabhängig von der Anzahl der Fließkommaoperationen eine Zeitkomplexität von $\mathcal{O}(1)$ aufweist, wird der Rumpf von S_2 selbst $\#Q$ mal ausgeführt. Damit beläuft sich die Gesamtlaufzeit der Schleife aufgrund Regel 3 auf $\#Q \cdot \mathcal{O}(1)$. Algorithmus 2 weist also eine Zeitkomplexität von $\mathcal{O}(1 + n) \in \mathcal{O}(n)$ auf, wobei $n = \#Q$ auf.

Der Algorithmus verwendet eine endliche Menge an Fließkommavariablen, weswegen beide eine Speicherkomplexität von $\mathcal{O}(1)$ aufweisen.

4.3 Analyse des Marching Cubes-Algorithmus

Der Marching Cubes-Algorithmus wurde erstmals durch [LC87] vorgelegt und ist zum Zeitpunkt des Abschlusses dieser Arbeit das am häufigsten verwendete Mittel, um Meshes aus Punktladungen zu generieren, die Isoflächen annähern. Seine Grundidee basiert darauf, das Skalarfeld zu diskretisieren, indem es einen festen Suchraum in würfelförmige Zellen aufteilt. Auf beliebige Dimensionen verallgemeinerte Variationen legen Simplizes als Zellgeometrie zugrunde [Gre+04; SB06]. Der Alg. 9 gibt den Marching Cubes-Algorithmus nach [CX14] unter behutsamer Korrektur und Anpassung an die Begrifflichkeiten dieser Arbeit wieder. Der Marching Squares-Algorithmus liegt in einer zum Alg. 9 analogen Variante vor und eignet sich zur Lösung zweidimensionaler Probleme. Der Zellenvektor beschreibt die Größe einer Zelle, gibt also die Schrittweite in x-, y- und z-Richtung an. Der achsenparallele Hüllkörper $\mathbf{b}$ beschreibt den Suchraum. Die Funktion L liefert den passenden Eintrag einer Lookup-Tabelle für die Aussage $\mathbf{E}(\mathbf{p}) < \xi$ für die aktuelle und die sechs umliegenden Zellen. In der zweidimensionalen Variante sind alle Vektoren zweidimensional, außerdem werden nur vier Parameter für den achsenparallelen Hüllkörper und nur fünf für die Lookup-Funktion verwendet. Ebenso entfällt die äußere der drei Schleifen. Alle folgenden Ausführungen sind sowohl auf den Marching Squares als auch auf den Marching Cubes-Algorithmus anwendbar, dienen aber vorrangig der Analyse des Marching Cubes-Algorithmus. Den besonderen Anwendungsfällen des Marching Squares-Algorithmus wenden wir uns in Abschnitt 6.2.1 zu.

[CX14] legen ebenfalls einen Beweis für die Terminierung und Korrektheit des Marching Cubes-Algorithmus vor. Dieser wird unter Zuhilfenahme maschinengestützter Beweise durch das Programm Coq geführt, das ebenfalls auf das Hoare-Kalkül zurückgreift [Aff15].

4.3.1 Zeitkomplexität

Hinsichtlich der Zeitkomplexität liegen zwar Ansätze vor, in denen etwa durch Zuhilfenahme von Oktonärbäumen eine Zeitkomplexität von

$$\mathcal{O}\left(p + p\log\left(\frac{n}{p}\right)\right)$$

erreicht wird. Diese setzen aber voraus, dass dafür p Zellen als nicht leer bekannt sind [NY06]. Da in unserem Fall weder an den Gitterpunkten diskretisierte Werte des elektrischen Feldes noch die Anzahl leerer Bereiche als bekannt vorausgesetzt werden können, müssen diese in einer Simulation durch den Alg.8 berechnet werden, der gemäß Kapitel 4 eine Zeitkomplexität von $\mathcal{O}(n)$ aufweist. Aus diesem Grund ist auch im Gegensatz zu anderen Zeitkomplexitätsanalysen – unter anderem in [CX14] – die Berechnung des Feldes durch die Funktion $\mathbf{E}$ mit $\mathcal{O}(k \cdot n)$ nicht vernachlässigbar. k bezeichnet hier die Anzahl der Funktionsaufrufe.

Darüber hinaus wirft der 9 die Frage auf, warum der asymptotische Aufwand des Marching Cube-Algorithmus in $\mathcal{O}(n^2)$ liegt, obwohl dieser über drei **solange**-Schleifen iteriert und im Inneren die Funktion $\mathbf{E}$ aufruft, für die ihrerseits ein Aufwand von $\mathcal{O}(n)$ anfällt. Dieser Frage liegt die Annahme einer Datenstruktur der Gitterpunkte als „dreidimensionales Array“ zugrunde, die wir hier als verschachtelte Folge $G = \{\{\{\mathbf{p}_1, \mathbf{p}_2, \ldots\}, \ldots\}\}$ auffassen wollen. Weiterhin verwenden wir Funktion L, die aus acht berechneten und ausgewerteten Werten des elektrischen Feldes an den Eckpunkten einer Zelle die passende Geometrie aus der Lookup-Tabelle bestimmt. Diese läuft wie gewohnt in konstanter Zeit [Wen13, S. 22]. Der Marching Cubes-Algorithmus lässt sich nun in der Bird Meertens-Form spezifizieren als:

$$(+/) \circ (+/) \circ L \circ (\mathbf{E} \star\star\star)G$$

In dieser Spezifikation lässt sich die dreimalige Anwendung des Operators $\star$ erkennen, der gemeinsam mit der Funktion $\mathbf{E}$ einen Zeitaufwand von $\mathcal{O}(n^4)$ suggeriert. Gemäß einem Sonderfall des Homomorphismen-Lemmas aus [Bir86] für die Verringerung der Tiefe („flatten“) verschachtelter Folgen gilt für beliebige Funktionen f:

$$(+/)(f \star\star) = (f\star) \circ (+/)$$

. Nach zweimaliger Anwendung ergibt sich:

$$(+/) \circ (+/) \circ L \circ (\mathbf{E} \star\star\star)G = L \circ (\mathbf{E}\star) \circ (+/) \circ (+/)G$$

Nach der Umformung ist ersichtlich, dass sich der Marching Cubes-Algorithmus mit einer $\star$-Operation über der Funktion $\mathbf{E}$ formulieren lässt, woraus sich seine asymptotische Laufzeit von $\mathcal{O}(n^2)$ ergibt.

Die obige Umstellung kann durch eine Umformulierung der Schleifenkaskade im äußeren Teil des Alg. 9 realisiert werden:

```
1  solange s_z < z_max ∧ s_y < y_max ∧ s_x < x_max;                    // 3 · O(n)
2  tue
3  |  M ← M + L(E(s + ϱ_x) < ξ, E(s − ϱ_x) < ξ, E(s + ϱ_y) < ξ, E(s − ϱ_y) <
   |    ξ, E(s + ϱ_z) < ξ,
4  |      E(s − ϱ_z) < ξ, E(s) < ξ);                                   // 7 · O(n)
5  |  s_x ← s_x + ϱ_x;
6  |  wenn s_x > x_max dann
7  |  |  s_x ← x_min;
8  |  |  s_y ← s_y + ϱ_y;
9  |  |  wenn s_y > y_max dann
10 |  |  |  s_y ← y_min;
11 |  |  |  s_z ← s_z + ϱ_z;
12 |  |  Ende
13 |  Ende
14 Ende
```

Dieses Ergebnis bezieht sich auf die einfachste vorliegende Form des Marching Cubes-Algorithmus, die die niedrigste Zeitkomplexität aufweist. Hierbei ist zu berücksichtigen, dass das Ergebnis lediglich als eine gerasterte Repräsentation der Isofläche vorliegt und ein in mehrerlei Hinsicht topologisch unkorrektes Mesh darstellt. Daher schließen sich üblicherweise an den Marching Cubes-Algorithmus unter anderem die folgenden Maßnahmen zur Verbesserung des Ergebnisses an. Die häufigste und im Hinblick auf die Zeitkomplexität günstigste Verbesserung wird durch eine lineare Interpolation [And14, Ff. 18f] der Vertices auf den Gitterpunkten erreicht. Bei Wiederverwendung der berechneten Feldstärken ist dies mit einem Zeitaufwand von $\mathcal{O}(1)$ möglich. Die Genauigkeit des Ergebnisses ist dabei nicht abzuschätzen und verbessert sich im schlimmsten Fall nicht wahrnehmbar. Das Ergebnis des Marching Cubes-Algorithmus ist die Konkatenation der einzelnen Meshes, die die Lookup-Tabelle liefert. Anhand Abbildung 4.1 lässt sich ersehen, dass es sich dabei ausnahmslos um offene Meshes handelt. Benachbarte Zellen berechnen die Vertices bzw. Kanten an den geteilten Würfelseiten mehrfach, wodurch diese auch mehrfach in die Vertex- bzw. Kantenliste des konkatenierten Meshes eingehen werden. Das Mesh wird also Doppelpunkte enthalten. In Anwendungen, die ein sauberes Mesh voraussetzen, ist ein anschließender Aufwand von $\mathcal{O}(n^2)$ für das Auffinden dieser Doppelpunkte durch erschöpfende Suche und die anschließende Bereinigung notwendig, wobei n die Anzahl der Vertices im Mesh ist. Ebenfalls ist anzuführen, dass die Lookup-Tabelle nicht eindeutig formuliert werden kann: Beispielsweise kann bei Auftreten des Falls 10 in Abbildung 4.1 nicht ermittelt werden, in welche Richtung die beiden Flächen zu orientieren sind, ohne zusätzliche Berechnungen durchzuführen, die die umliegenden Zellen einbeziehen. [NH91] steht hierbei stellvertretend für die schnellsten verschiedener vorgeschla-

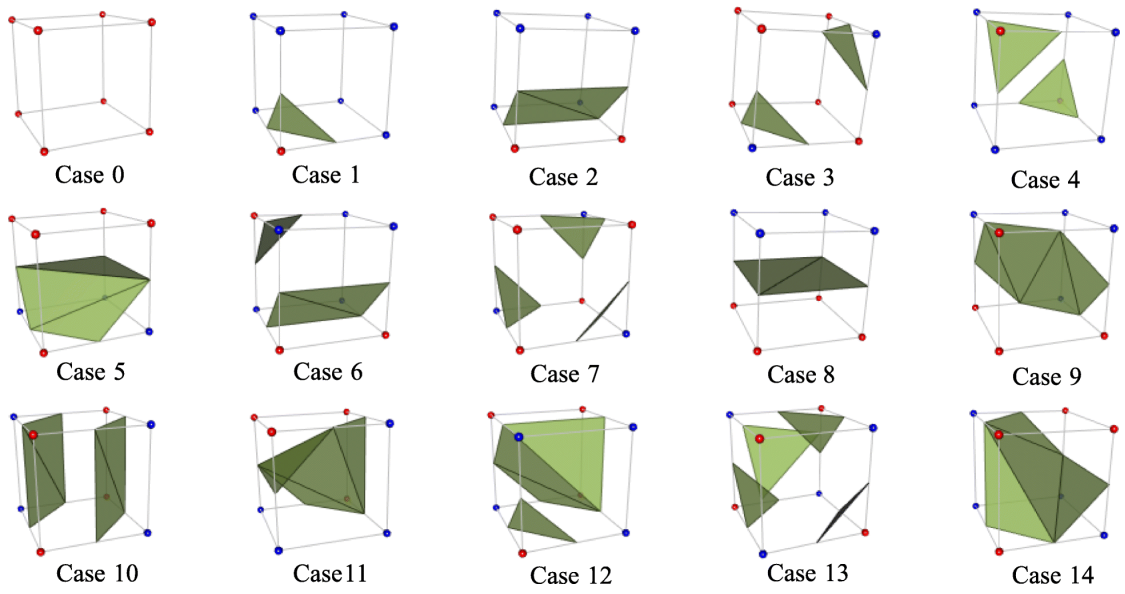


Abbildung 4.1: Lookup-Tabelle des Marching Cubes-Algorithmus [CPS19]

gener Lösungen für dieses Problem. Die Autoren erstellen bei Auftreten einer uneindeutigen Zelle die bilineare Interpolation einer Würfelfläche und untersuchen diese statt des elektrischen Feldes. Aufgrund der hyperbolischen Gestalt des Interpolanten können die Orientierungen der Flächen ermittelt werden. Als Kriterium dafür wird die Stärke des elektrischen Feldes am Treffpunkt der Asymptoten der beiden Hyperbeln herangezogen. Liegt dieser oberhalb des Isowerts, müssen die Flächen nach außen zeigen, ansonsten nach innen. Für die Probleme, die dem Ergebnis-Mesh anhaften, sind also bereits Lösungsvorschläge vorhanden. Allerdings gehen diese immer mit einer Erhöhung der Zeitkomplexität einher. Wir werden später weitere Verfahren untersuchen, die in der Komplexitätsklasse des Marching Cubes-Algorithmus liegen, die angeführten topologischen Probleme der Meshes aber umgehen und eine höhere Genauigkeit als eine lineare Interpolation erreichen. Daher verzichten wir hier auf die Gegenüberstellung der Verfahren zur Verbesserung des Marching Cubes-Algorithmus.

Algorithmus 9 : Marching Cubes

eingabe : Zellenvektor $\varrho \in \mathbb{R}^3$,
AABB-Parameter $\{\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{max}}, \mathbf{y}_{\text{min}}, \mathbf{y}_{\text{max}}, \mathbf{z}_{\text{min}}, \mathbf{z}_{\text{max}}\} \in \mathbb{R}^6$, Funktion $\mathbf{E} : G \to R$, Isowert ξ, Lookup-Funktion $L : \{\top, \bot\}^7 \to M$

ausgabe : Mesh M, das die Isofläche $\mathbf{E}(\mathbf{p}) = \xi$ repräsentiert

$M \leftarrow \varnothing$
$\mathbf{s} \leftarrow \begin{pmatrix} \mathbf{x}_{\text{min}} \\ \mathbf{y}_{\text{min}} \\ \mathbf{z}_{\text{min}} \end{pmatrix}$
solange $\mathbf{s}_z < \mathbf{z}_{max}$ **tue**
  solange $\mathbf{s}_y < \mathbf{y}_{max}$ **tue**
    solange $\mathbf{s}_x < \mathbf{x}_{max}$ **tue**
      $M \leftarrow M + L(\mathbf{E}(\mathbf{s} + \varrho_x) < \xi, \mathbf{E}(\mathbf{s} - \varrho_x) < \xi, \mathbf{E}(\mathbf{s} + \varrho_y) < \xi,$
        $\mathbf{E}(\mathbf{s} - \varrho_y) < \xi, \mathbf{E}(\mathbf{s} + \varrho_z) < \xi, \mathbf{E}(\mathbf{s} - \varrho_z) < \xi, \mathbf{E}(\mathbf{s}) < \xi)$
      $\mathbf{s}_x \leftarrow \mathbf{s}_x + \varrho_x$
    Ende
    $\mathbf{s}_y \leftarrow \mathbf{s}_y + \varrho_y$
    $\mathbf{s}_x \leftarrow \mathbf{x}_{\text{min}}$
  Ende
  $\mathbf{s}_z \leftarrow \mathbf{s}_z + \varrho_z$
  $\mathbf{s}_y \leftarrow \mathbf{y}_{\text{min}}$
Ende

4.4 Berechnung zweidimensionaler Flächen durch schrittweise Umläufe

4.4.1 Beschreibung des Verfahrens in zwei Dimensionen

Bei dem in [Rot+18] vorgestellten Ansatz verwenden wir den von einem Startpunkt p bestimmten Isowert, um stückweise eine Isofläche um ein Cluster von Punktladungen mit Stücken einer gegebenen Länge $\mathbf{s}$ zu zeichnen. Innerhalb einer Simulation kann dieser Startpunkt beispielsweise durch einen Mausklick oder das Betätigen eines Triggers eines VR-Controllers an einer bestimmten Position ausgewählt werden. Die Vorarbeit für das Interaktionsmodell liefern unter anderem [Tom15] und [JH12]. Ausgehend vom Startpunkt werden sukzessive Kontrollschritte der Länge von $\mathbf{s}$ in Richtung $\hat{\varphi}$ durchgeführt, bis der Abstand zwischen Startpunkt und Kontrollposition kleiner als die Länge von $\mathbf{s}$ ist; in diesem Fall wird die Schleife beendet. Somit besteht das erzeugte Mesh aus Strecken, die alle die gleiche Länge aufweisen, mit Ausnahme der letzten Strecke, die normalerweise kürzer ist. Es kann also bis auf diese Strecke als homogen bezeichnet werden. Das Ergebnis erfordert eine Nachbehandlung in zweierlei Hinsicht:

4.4.1.1 Der neue Punkt liegt nicht auf der Isofläche

Der Kontrollpunkt befinde sich bei $\mathbf{a}$. $\|\mathbf{E}(\mathbf{a}+s)\| = \xi$ gilt nicht immer, selbst wenn $\|\mathbf{E}(\mathbf{a})\| = \xi$ für den Kontrollpunkt $\mathbf{a}$ gilt. An einen Schritt muss sich also ein Verfahren anschließen, das einen korrigierten Punkt $\mathbf{p}_c$ findet, für den innerhalb einer angemessenen Toleranz $\|\mathbf{E}(\mathbf{p}_c)\| = \xi$ gilt. Die Toleranz ist abhängig vom Anwendungsfall.

Das vorliegende Problem ist numerischer Natur und beispielsweise durch das Newton-Raphson-Verfahren lösbar, wenn wir das Problem in einer Schnittebene des elektrischen Feldes mit dem Aufpunkt $\mathbf{p}'$ betrachten. Zusätzlich erfordert die Ebene einen Richtungsvektor, der sicherstellt, dass diese Ebene die Fläche schneidet. Dies kann durch einen Richtungsvektor erreicht werden, der orthogonal auf $\mathbf{p}' - \mathbf{p}$ steht. Ein Beispiel für ein verwandtes Problem mit ähnlicher Lösung findet sich in [Gre19, S. 152], eine Angabe des Algorithmus mit Implementierung auf [Gre19, Ss. 679ff].

In Abschnitt 3.3.13 thematisieren wir die Homogenität von Meshes. Ein zweidimensionales Mesh kann durch den schrittweisen Algorithmus per Konstruktion erreicht werden, wenn wir statt wie oben einer Gerade einen Kreis mit dem Radius $\|\mathbf{s}\|$ als Schrittweite zugrundelegen. Auf diese Weise ist gewährleistet, dass bis auf die letzte Kante alle die gleiche Länge aufweisen. Das Newton-Raphson-Verfahren ist hier nicht mehr anwendbar. Ebenso wenig können das Regula-Falsi- oder Bisektionsverfahren zum Einsatz kommen, weil mit $\mathbf{p} + s$ nur ein Punkt-Wertepaar des elektrischen Feldes auf dem Kreis bekannt ist. Dennoch ist ein Rückgriff auf den Mittelwertsatz möglich, in dem wiederum schrittweise die Richtung des Vektors $\mathbf{p} + \mathbf{s}$ durch Rotation verändert und bei jedem Schritt das elektrische Feld berechnet wird. Wechselt bei einem Winkel φ_1 das elektrische Feld im Vergleich zum Winkel des vorangegangenen Schrittes φ_2 das Vorzeichen, ist ein Intervall gefunden, das die Lösung enthält, so dass mit dem Bisektionsverfahren fortgefahren werden kann.

4.4.1.2 Die Wahl von s kann unpraktisch sein

Der Punkt bei $\mathbf{a} + s\hat{\boldsymbol{\varphi}}$ möglicherweise näher an einem anderen Cluster. Diese Konstellation ist bei vergleichsweise großen Werten von s möglich. In diesem Fall kann die Suche der inneren Schleife bei $\mathbf{a}' = \mathbf{a} - s\hat{\mathbf{E}}$ begonnen werden. Das Problem besteht weiterhin bei Werten von s, die größer sind als der Durchmesser einer Fläche, was in unrealistischen Ergebnissen resultieren würde. Um sicherzustellen, dass eine Lösung $\mathbf{p}_c$ gefunden werden kann, könnte s so gewählt werden, dass der ermittelte Isowert nirgendwo auf dem Kreis zu finden ist. Die Auswirkungen von s auf unpraktische Wahlmöglichkeiten sind in Abb. 4.3 dargestellt. Allgemeiner kann gesagt werden, dass nur s, dessen Betrag kleiner geringer als der Abstand der Punktladungen ist, zu sinnvollen Ergebnissen führen kann.

Algorithmus 10 : Zeichnen durch schrittweise Umläufe mit konstanter Schrittweite

eingabe : Startpunkt $\mathbf{p}$, Länge s, Fehlerschwelle t_0, Initialer Winkelschritt t_s

Ziel-Isowert $\xi \leftarrow \|\mathbf{E}(\mathbf{p})\|$;
Erster Punkt der Strecke $\mathbf{p}' \leftarrow \mathbf{p}$;
solange $\|\mathbf{p} - \mathbf{p}'\| > s$; `// Solange Linienende nicht in der Nähe des Anfangs` **tue**
    Zweiter Punkt der Strecke $\mathbf{p}_c \leftarrow \mathbf{p}' + s\hat{\boldsymbol{\varphi}}$;
    Winkel $\alpha \leftarrow \tan^{-1}(\mathbf{p}_c - \mathbf{p}')$;
    Winkelschritt $t \leftarrow t_s$;
    Vorzeichen $v \leftarrow \mathrm{sgn}\mathbf{E}(\mathbf{p}_c)$;
    solange $t > t_0$ **tue**
        $\mathbf{p}_c \leftarrow \mathbf{p}' + s\begin{pmatrix}\sin\alpha\\ \cos\alpha\end{pmatrix}$;
        wenn $\mathrm{sgn}(\mathbf{E}(\mathbf{p}_c) - \xi) \neq v$ **dann**
            $t \leftarrow -\frac{t}{2}$; `// Bei Vorzeichenwechsel: Suchrichtung umkehren`
            ; `// und Schrittweite reduzieren`
            $v \leftarrow \mathrm{sgn}(\mathbf{E}(\mathbf{p}_c) - \xi)$;
        Ende
        $\alpha \leftarrow \alpha + t$;
    Ende
    Verbinde $\mathbf{p}'$ mit $\mathbf{p}_c$;
    $\mathbf{p}' \leftarrow \mathbf{p}_c$;
Ende
Verbinde $\mathbf{p}'$ mit $\mathbf{p}$;

4.4.2 Terminierung und Korrektheit

Für die Beweise der Terminierung und Korrektheit setzen wir voraus, dass keine ungünstige Wahl für s getroffen wurde, außerdem unter dieser Voraussetzung ohne Einschränkungen, dass eine einwertige Isofläche zu repräsentieren ist. Der Alg. 10 ist in einer vereinfachten Form wiedergegeben, die an dieser Stelle irrelevante Schritte auslässt. So ist aufgrund der Tatsache, dass zwei Jordan-Kurven stets eine gerade Anzahl von Schnittpunkten haben [Sch16], zu prüfen, ob ein bereits bearbeiteter Punkt auf der Fläche gefunden wurde. Diese Details wurden in den Implementierungen entsprechend berücksichtigt.

Die Beweise für Terminierung und Korrektheit verlaufen analog zu denen des Alg. 8. Die äußere Schleife des Alg. 10 ergibt sich aus der in Abschnitt 3.2 getroffenen Feststellung, dass es sich bei zweidimensionalen Flächen um Jordan-Kurven handelt: Sofern für die innere Schleife S_i das Hoare-Tripel

$$\langle\!|\, t > t_0 \wedge I \,|\!\rangle \quad S_i \quad \langle\!|\, t \leq t_0 \wedge I \,|\!\rangle$$

gilt, kann für alle Punkte des resultierenden Polygons angenommen werden, dass sich diese im Vergleich zur Schrittweite annähernd auf der Fläche befinden. Da sich $\mathbf{p}$ exakt auf der Fläche

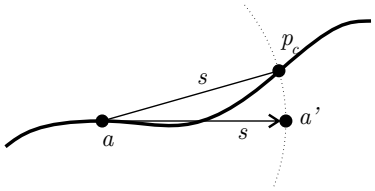


Abbildung 4.2: Bestimmung von p_c

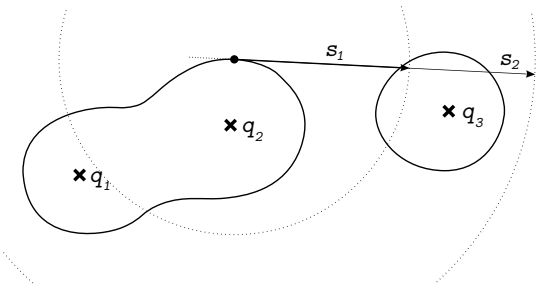


Abbildung 4.3: Auswirkungen unterschiedlicher gewählter Schrittweiten s

befindet, gibt es im Polygon zwei aufeinanderfolgende Punkte $\mathbf{p}'$ und $\mathbf{p}''$, die mit $\mathbf{p}$ annähernd kollinear sind, so dass $\mathbf{p}$ „zwischen“ diesen Punkten liegt. Da der Abstand dieser beiden Punkte aber s ist, muss für beide Punkte, besonders ersteren $\|\mathbf{p} - \mathbf{p}'\|$ gelten, wodurch die äußere Schleife abbricht.

Der Beweis der Terminierung der inneren Schleife folgt analog zu dem des Alg. 8: Durch die Anweisung $t \leftarrow -\frac{t}{2}$ sinkt der Wert von t monoton für jeden Schleifendurchlauf und muss nach endlich vielen Schritten unter t_0 fallen. Da s nicht ungünstig gewählt wurde, existiert der gesuchte Schnittpunkt zwischen Fläche und Suchkreis. Aus diesem Umstand folgt, dass nach endlich vielen Durchläufen der inneren Schleife die Vorzeichen der Differenz des Feldes an den gemessenen Positionen und dem Ziel-Isowert sich in zwei aufeinanderfolgenden Schleifendurchläufen unterscheiden. Ein passender Kandidat für die Invariante der inneren Schleife ist:

$$
\begin{aligned}
I = \Bigg(& \left\| \mathbf{E}\left(\mathbf{p}' + s\begin{pmatrix}\sin\alpha\\ \cos\alpha\end{pmatrix}\right) - \xi \right\| \leq \left\| \mathbf{E}\left(\mathbf{p}' + s\begin{pmatrix}\sin\alpha + t\\ \cos\alpha + t\end{pmatrix}\right) - \xi \right\| \\
\vee \quad & \left\| \mathbf{E}\left(\mathbf{p}' + s\begin{pmatrix}\sin\alpha\\ \cos\alpha\end{pmatrix}\right) - \xi \right\| \leq \left\| \mathbf{E}\left(\mathbf{p}' + s\begin{pmatrix}\sin\alpha - \frac{t}{2}\\ \cos\alpha - \frac{t}{2}\end{pmatrix}\right) - \xi \right\| \Bigg)
\end{aligned}
$$

Die Frage, ob das Vorzeichen während des Schleifendurchlaufs gewechselt hat, wird durch die Konjunktion geklärt: Ist die Bedingung der Schleife wahr, gilt der rechte Teil der Konjunktion; andernfalls gilt der linke Teil. Vor der Schleife sind die Variablen so belegt, dass der linke

Teil mit „=“ gilt. Nach der bedingten Modifikation von t werden keine weiteren Veränderungen vorgenommen, womit am Ende der Schleife die Belegung des letzten Schleifendurchlaufs vorliegt.

4.4.3 Laufzeitkomplexität

Der Alg. 10 berechnet das Ergebnis durch zwei Schleifen und greift dabei wieder auf die Berechnung des elektrischen Feldes durch Alg.8 zurück, der eine Laufzeitkomplexität von $\mathcal{O}(n)$ aufweist.

Die innere Schleife läuft, solange bis der Winkelschritt t unter die angegebene Fehlerschwelle t_0 fällt. Dafür wird der Fehlerschritt so lange halbiert, bis die Bedingung der Schleife nicht mehr erfüllt wird. Der Aufwand der inneren Schleife beläuft sich also auf $\mathcal{O}(\log n)$. Dass t dabei nicht in jedem Schleifendurchlauf reduziert wird, erhöht zwar die Anzahl der Schleifendurchläufe und damit etwa die Laufzeit für die Symbole Ω und T, allerdings bleibt die asymptotische Laufzeit von diesem Befund unberührt. Zum besseren Verständnis kann es hilfreich sein, die Bedingung der Wenn-Abfrage $\operatorname{sgn}(\mathbf{E}(\mathbf{p}') + c) \neq v$ als stochastischen Prozess zu verstehen, womit wir den Wahrheitswert dieser Aussage in der Aufwandsanalyse als rein zufälligen Koeffizienten betrachten können. Somit wird deutlich, dass die Variablen t und t_0, die die Schleife kontrollieren, nicht beeinflussen.

Die äußere Schleife läuft mit $\mathcal{O}(n)$ was einen zeitlichen Gesamtaufwand von $\mathcal{O}(n \log n)$ ergibt.

4.5 Bestimmung der Vertex-Positionen durch ein Shrinkwrap-Verfahren

In den folgenden Abschnitten betrachten wir die Möglichkeit, uns im Rahmen einer Simulation der korrekten Repräsentation von Isoflächen durch Meshes durch das in Abschnitt 3.1 angedachte Verschieben seiner Vertices schrittweise zu nähern. Einige Parameter dieser Möglichkeit haben wir im Laufe dieser Arbeit bereits beleuchtet, etwa in Abschnitt 3.2 und Abschnitt 3.3.13. Dabei orientieren wir uns grob am *Shrink-Wrap*-Verfahren von [Kob+00] und untersuchen Voraussetzungen, die die Autoren für die Wirksamkeit ihres Verfahrens zugrundelegen, dahingehend, ob sie unserem Fall ebenfalls gelten. Dafür leiten wir aus den Eigenschaften elektrischer Felder und triangularer Meshes Vereinfachungen her, die wir in unserer Simulation zur Vereinfachung und Beschleunigung unserer Berechnungen und der Verbesserung der Ergebnisse im Hinblick auf Genauigkeit und Ansehnlichkeit dienstbar machen. Da eine Betrachtung des Shrink-Wrap-Verfahrens in zwei Dimensionen eine Vereinfachung des Verständnisses und interessante Möglichkeiten der Visualisierung bietet, beleuchten wir die relevanten Unterschiede dieser Variation. Wir beweisen die Terminierung und die Korrektheit des Verfahrens und führen

eine Laufzeit- und Speicherkomplexitätsanalyse durch. Das Ziel kann dabei als mehrschrittiges Mesh Morphing [Ale01] verstanden werden, abgesehen davon, dass unsere Zielform (*target shape*) nicht bekannt ist und wir daher nicht auf baryzentrische Abbildungen zurückgreifen können, dafür aber der Pfad der Vertices (vgl. [S. 2, Schritt 3][Ale01]) nicht berechnet werden muss, sondern bereits vorgegeben ist.

In Abschnitt 3.3.7 haben wir die vorteilhafte Zeitkomplexität von Mesh-Bäumen gegenüber anderen Operationen auf Meshes wie dem Auftrennen und Zusammenfügen bei Erkennen falscher Annahmen über das Äquivalenzverhältnis von Punktladungen dargelegt. Diese Erkenntnisse können wir in einer Simulation nutzen, wodurch in dieser schnell auf falsche Annahmen der Äquivalenz oder Nicht-Äquivalenz reagiert werden kann.

Das in [Kob+00] vorgestellte Shrink-Wrap-Verfahren stellt in sich in Grundzügen ein Simulationsverfahren dar, bei dem die Bewegungshalbgerade des Vertex anhand eines Projektionsoperators P, der jeden Vertex eines kugelförmigen Mesh (*spherical mesh*) als *Startgeometrie* auf den nächsten Punkt des anzunähernden Meshes, der *Zielgeometrie*, abbildet. Gemäß Gleichung (2.1) entsprechen die Parameter $\mathbf{a}$ demnach der Position der Vertices auf der Startgeometrie und $\mathbf{b}$ den jeweiligen Vertex-Normalen. Diese Wahl ist dem Umstand geschuldet, dass außer den Vertices der Startgeometrie kein weiterer Punkt zur Konstruktion zur Verfügung steht. Die Autoren berechnen als Position der Vertices durch die Konstruktion von Halbgeraden (*shooting rays*) zwischen dem Vertex selbst und dem nächsten Punkt auf der Zielgeometrie. In unserem Fall steht allerdings mit der Position der Punktladung ein Kandidat für $\mathbf{b}$ zur Verfügung. Daher erscheint es als sinnvoll, abweichend von [Kob+00] in der Konstruktion der Halbgerade diesen als zweiten Punkt festzulegen.

Der Konturbaum kann zur Bestimmung von Startgeometrien herangezogen werden, das durch ihn bestimmte Ergebnis ist aber ambivalent. Für die folgenden Ausführungen können wir ohne Einschränkung der Allgemeinheit in Anlehnung an [Kob+00] als Startgeometrie eine Icosphere um jede Punktladung wählen. Diese Wahl beeinflusst die folgenden Ausführungen nicht, falls diese endgültig ist. Ansonsten ist der zusätzliche Aufwand dafür in der Komplexitätsanalyse gesondert vorzusehen. Sofern die Cluster nicht im Voraus bestimmt werden können, bietet sich der Konturbaum als Mittel zur Reaktion auf das Erkennen falscher Annahmen der Äquivalenz oder Nicht-Äquivalenz an. In jedem Fall muss für eine Simulation eine Startgeometrie gewählt werden. Außerdem kommt den Ausführungen in Abschnitt 3.2 potentiell das Vorliegen einer disjunkten Zielgeometrie in Betracht, wodurch gemäß Abschnitt 3.3.13 der Erhalt der Homogenität Teil der Zielsetzung sein kann.

4.5.1 Beschreibung des Verfahrens in zwei Dimensionen

In jedem Schritt wird für alle Vertices $\mathbf{v}$ der Startgeometrie das Skalarfeld $\|\mathbf{E}(\mathbf{v})\|$ berechnet und mit einem vorgegebenen oder analog zu Abschnitt 4.4.1 durch Mausklick bestimmten Isowert verglichen, um einen Fehlerwert ε zu berechnen. In Abhängigkeit dieses Fehlerwerts wird $f(\varepsilon) \cdot p$ auf der Halbgeraden

$$s(x) = \mathbf{v} + x\mathbf{m}, \qquad x \in \mathbb{R}, x \leq 0$$

dargestellt, die $\mathbf{v}$ selbst mit der Position einem gesondert zu benennenden Mittelpunkt $\mathbf{m}$ verbindet. $f : \mathbb{R} \to \mathbb{R}$ bezeichnet hier eine Funktion, die den Fehlerwert ε dem Verfahren zugänglich macht und wird abhängig vom Verwendungszweck gewählt. Beispielsweise sollte sie für den vorliegenden Fall so konstruiert werden, dass das Verfahren die Vertices um umso kleinere Abstände verschiebt, je weniger das elektrische Feld an dieser Position vom gesuchten Isowert abweicht. Dies leistet beispielsweise eine gestauchte Tangens-Funktion, die qualitativ mit hinreichender Genauigkeit und geringer Laufzeitkomplexität durch

$$f(x) = \max\left(-l, \min\left(l, \frac{x}{k}\right)\right) \tag{4.2}$$

mit einer Dämpfungskonstanten $k \in \mathbb{R}, 0 < k < 1$ und einem Sättigungswert $l \in \mathbb{R}$ angenähert werden kann. Damit verringert die Funktion f den Betrag der Schrittweite und beschränkt ihn auf den Wert l. Abbildung 4.4 veranschaulicht diese Variante. Unter der Annahme, dass für f eine Funktion mit einer Laufzeitkomplexität von $\mathcal{O}(1)$ gewählt wird, sind die folgenden Ausführungen unabhängig von dieser Wahl. Gleichung (4.2) stellt ein geeignetes Beispiel einer solchen Funktion dar.

In einfachen Konfigurationen kann dafür der Schwerpunkt der Startgeometrie oder der gewichtete Mittelpunkt der Punktladungen und deren Ladungsstärke q_i herhalten. Sollten sich während der Berechnung komplexere Geometrien ergeben, in denen Artefakte und Konvexitäten auftreten, bieten die einschlägigen Verfahren in Abschnitt 3.3.13 Abhilfe.

In der initialen Phase des Alg. 19 wird die Startgeometrie erzeugt. Im zweidimensionalen Fall entstehen dabei kreisförmige Polygone mit einer gewünschten Schrittweite, etwa durch eine Variation des Bresenhamschen Midpoint Circle-Algorithmus [Kot+19, S. 55ff], der als Schrittgröße eine Strecke statt Pixeln verwendet und die Ergebnispunkte durch Kanten verbinden könnte. Beim Erstellen dieser Kreise ist darauf zu achten, dass sie keine weiteren Punktladungen enthalten. Dafür kann etwa ein Radius gewählt werden, der kleiner ist, als der kürzeste Abstand zwischen zwei Punktladungen. Dieses Vorgehen ist nur für einzelne Punktladungen möglich, da es sich hierbei um eine diskrete Topologie handelt. Liegen mehrdimensionale Ladungsanordnungen wie etwa Linien- oder Volumenladungen vor, kann beispielsweise auf

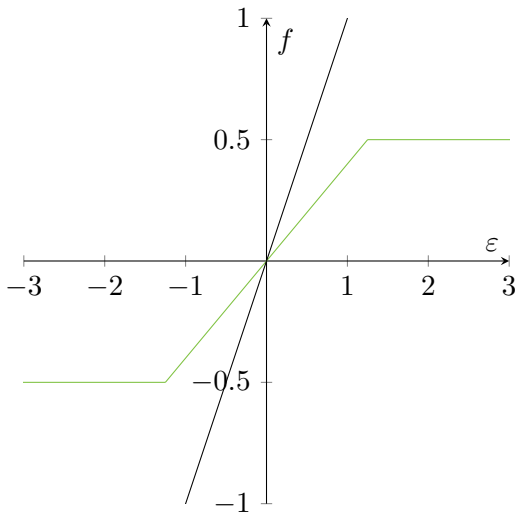


Abbildung 4.4: ε (schwarz) und mögliche Funktion $f(\varepsilon)$ mit $l = 0.5$ und $k = 0.4$ (blau)

die Konstruktion einer Startgeometrie durch die Minkowski-Summe eines kleinen Kreises und der Topologie der Ladungsanordnung zurückgegriffen werden. Allerdings weist diese eine zusätzliche Laufzeitkomplexität von $\mathcal{O}(n^2)$ auf [Ski09, S. 617].

Die Berechnung des Shrink-Wrap-Verfahrens kann im Sonderfall eines statischen Feldes mit annähernd kreisförmigen Flächen mit gut abschätzbarer Größe ebenfalls vor der Simulation durchgeführt werden. Liegen allerdings dynamische Felder vor oder liegen die Ausdehnungen der Flächen nicht nah beieinander, kann während der Simulation ein Rechenschritt des Verfahrens pro Wiederholung durchgeführt werden, bevor das Bild gerendert wird. Dies führt zu sukzessive präziseren Ergebnissen bei jedem Frame. Die Ergebnisse von fünf Iterationen des Alg. 19 sind in Abbildung 4.5 beispielhaft dargestellt.

Die Vertices passen sich automatisch an Veränderungen dynamischer Felder, beispielsweise Änderung der Position oder Ladungsstärke der Punktladungen an, solange die Äquivalenzmengen konstant bleiben. Andernfalls stehen die Erkenntnisse zur Reaktion auf falsch angenommene Äquivalenzverhältnisse der Punktladungen aus Abschnitt 3.3.7 als Abhilfe zur Verfügung. Die Vertexdichte der resultierenden Meshes ist dabei nicht immer leicht abzuschätzen und kann ebenfalls während der Simulation durch gegebenenfalls selektives Anpassen der Auflösung, wie in Abschnitt 3.3.13 adaptiert werden.

4.5.2 Terminierung und Korrektheit

Der Verständlichkeit halber wurde die clusterbezogene Behandlung der Meshes sowie die Behebung von Artefakten, die im Abschnitt 3.3.13 besprochen wurden, bei der Angabe des Alg. 19

Algorithmus 11 : Simulierter Algorithmus: Simulation und Schritt

eingabe : Menge der Punktladungen $\mathcal{Q}$, Isowert ξ oder Startpunkt $\mathbf{p}$

```
1  ξ ← ‖E(p)‖;                    // Falls statt ξ der Startpunkt p gegeben ist
2  für alle (p, q) ∈ Q tue
3      Mittelpunkt der Punktladungen m ← 1/#Q ∑ Q;
4      Lade kreisförmige Startgeometrie mit Mittelpunkt m, die alle betreffenden
        Punktladungen enthält;
5      für alle Vertices des resultierenden Polygons v tue
6          v ← v + m;
7      Ende
8      Vorbereitung für Fehlererkennung;
9  Ende
10 solange Simulation läuft tue
11     für alle Faces des Meshes f ∈ F tue
12         für alle (v, m) ∈ V tue
13             ε ← E(v) − ξ;
14             v ← v − f(ε) (v−m)/‖v−m‖;
15         Ende
16         Fehlerbehandlung für F;
17     Ende
18     Zeichne Kanten E;
19 Ende
```

vernachlässigt. Bei diesen handelt es sich um eigene Verfahren, denen sich eigene Abschnitte dieser Arbeit widmen.

Beide seiner für-alle-Schleifen iterieren über eine endliche Menge. Der Beweis der Terminierung erfolgt analog zu dem des Alg. 8; Die solange-Schleife terminiert, sofern die Simulation terminiert. Die Korrektheit der Konstruktion der Startgeometrie beweisen [Kob+00] bereits. Für die Korrektheit der Schleife im Rendervorgang ist zu zeigen, dass für die Fehlerwerte $\mathbf{E}(\mathbf{v}_1) - \xi$ und $\mathbf{E}(\mathbf{v}_2) - \xi$ in zwei aufeinanderfolgenden Schleifendurchläufen $\mathbf{E}(\mathbf{v}_2) - \xi \leq \mathbf{E}(\mathbf{v}_1) - \xi$ gilt, was als Aussage für eine Schleifeninvariante I gelten kann.

Für den Teil vor der Schleife darf die Schleifeninvariante auf $\mathbf{E}(\mathbf{v}_2) - \xi < \infty$ verstärkt werden und gilt trivialerweise. Innerhalb der Schleife gilt mit ebenso verstärkter Vorbedingung:

Aus dem vorherigen Schleifendurchlauf mit dem entsprechenden Fehlerwert ε_1 ist bekannt, dass
$(\!| \mathbf{E}(\mathbf{v}_1 - f(\varepsilon_1)\frac{\mathbf{v}_1-\mathbf{m}}{\|\mathbf{v}_1-\mathbf{m}\|}) - \xi = \varepsilon_1 |\!)$
Diese Vorbedingung lässt sich abschwächen zu:
$(\!| \mathbf{E}(\mathbf{v}_1 - f(\varepsilon_1)\frac{\mathbf{v}_1-\mathbf{m}}{\|\mathbf{v}_1-\mathbf{m}\|}) - \xi \leq \varepsilon_1 |\!)$
Da bisher $\mathbf{v}_1 = \mathbf{v}_2$, gilt auch:

$(\!| \mathbf{E}(\mathbf{v}_1 - f(\varepsilon_1)\frac{\mathbf{v}_1-\mathbf{m}}{\|\mathbf{v}_1-\mathbf{m}\|}) - \xi \leq \varepsilon_2 |\!)$
$\varepsilon_2 \leftarrow \mathbf{E}(\mathbf{v}_1) - \xi$
$(\!| \mathbf{E}(\mathbf{v}_1 - f(\varepsilon_1)\frac{\mathbf{v}_1-\mathbf{m}}{\|\mathbf{v}_1-\mathbf{m}\|}) - \xi \leq \mathbf{E}(\mathbf{v}_1) - \xi |\!)$
$\mathbf{v}_2 \leftarrow \mathbf{v}_1 - f(\varepsilon_2)\frac{\mathbf{v}_2-\mathbf{m}}{\|\mathbf{v}_2-\mathbf{m}\|}$
$(\!| \mathbf{E}(\mathbf{v}_2) - \xi \leq \mathbf{E}(\mathbf{v}_1) - \xi |\!)$

4.5.3 Beschreibung des Verfahrens in drei Dimensionen

Die im vorherigen Abschnitt dargelegte Variante des Shrink-Wrap-Verfahrens greift, abgesehen von der Konstruktion der Startgeometrie, ausschließlich auf affine Abbildungen zurück und lässt sich somit durch einfaches Erweitern der Dimensionen der Vektoren auf drei Dimensionen übertragen. Als dreidimensionale Startgeometrien kommen etwa UV- oder Icospheres in Frage.

4.5.4 Zeitkomplexität

Die initialisierende Phase des Alg. 19 kennzeichnet sich durch eine für-alle-Schleife, die über die Punktladungen und eine innere Schleife, die über alle Vertices der zuvor geladenen Startgeometrie iteriert. Der Aufwand der Initialisierungsphase beträgt also $\mathcal{O}(n^2 + T(n))$, mit T als Laufzeit für die Vorbereitung der Fehlererkennung.

Während der Renderschleife iterieren zwei Schleifen über die Faces bzw. Vertices dieser Faces. Bei Vorliegen eines regulären Meshes – bei UV- und Icospheres handelt es sich um solche – fällt ein Aufwand von $\mathcal{O}(n)$ an. Für die Fehlerbehandlung wurden in Abschnitt 3.3.1 verschiedene Verfahren mit unterschiedlicher Laufzeit vorgestellt. Die asymptotische Laufzeit können wir also mit $\mathcal{O}(n + T(n))$ mit der Laufzeit T des Fehlererkennungsverfahrens abschätzen. In Abschnitt 3.3.1 kommen wir zu der Feststellung, dass die Laufzeit für die Erkennung falsch angenommener Äquivalenzen die der für die Erkennung falsch angenommener Nicht-Äquivalenzen schlägt. Der Konturbaum aus Abschnitt 3.3.7 liefert die beste asymptotische Laufzeit während der Simulation, wenn bei der Initialisierung nur das Wurzelelement markiert ist. Dadurch bleibt die Laufzeit der Simulation bei $\mathcal{O}(n)$, dafür steigt der Aufwand während der Initialisierung auf $\mathcal{O}(2^n)$.

Das Ziel des Shrink-Wrap-Verfahrens, die Bildwiederholrate der Simulation zu minimieren, kann durch eine Betrachtung der asymptotischen Laufzeit nicht nachgewiesen werden: Beide Verfahren operieren in der Komplexitätsklasse $O(n^2)$. n steht hier für die Anzahl der Berechnungen des elektrischen Feldes aus der Ladungsverteilung durch Gleichung (2.5). Dennoch ist zu erwarten, dass bei einer gegebenen Ladungsverteilung und einem gegebenen Isowert das Shrink-Wrap-Verfahren die Berechnung der Geometrie schneller abschließen kann, als der Marching Cubes-Algorithmus. Der Grund dafür liegt im Leerraum, über den der Marching Cubes-Algorithmus iterieren muss, um zu einem Ergebnis zu gelangen.

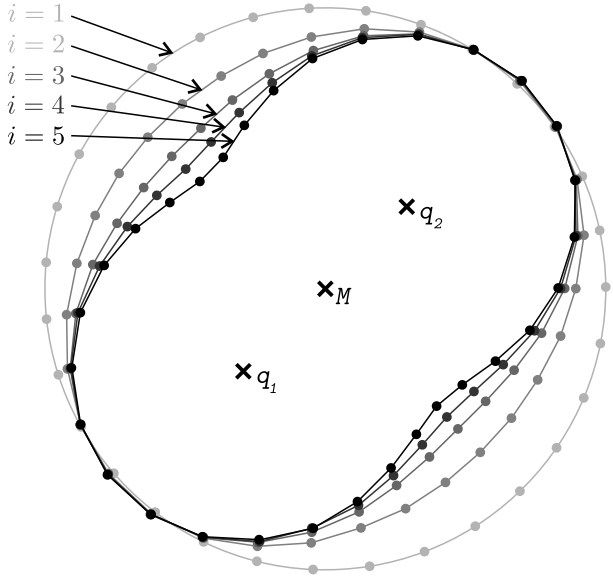


Abbildung 4.5: Die ersten fünf Durchläufe der Renderschleife mit Alg. 19 für eine Fläche, die zwei Punktladungen enthält

Abbildung 4.6 greift das Beispiel aus Abbildung 4.5 auf. Dargestellt ist der fünfte Schritt der Iteration. Für dieses Beispiel nehmen wir an, dass der Abstand aller Punkte des Shrink-Wrap-Verfahrens zur Isofläche vernachlässigbar ist. Außerdem zeigt die Abbildung das Gitter der Zellen, über die der Marching Cubes-Algorithmus iteriert. Im gegebenen Beispiel übersteigt die Anzahl der Iterationspunkte des Marching Cubes mit 104 die des Shrink-Wrap-Verfahrens mit 31 Iterationspunkten. Dieser Unterschied kommt durch leere Zellen innerhalb der Fläche. Das vorliegende Beispiel begünstigt den Marching Cubes-Algorithmus: Es ist so gewählt, dass außerhalb der Fläche nur einzelne leere Zellen vorkommen. Außerdem ist die Isofläche einwertig, wodurch keine etwaigen leeren Zellen zwischen den Flächen auftreten können. Dennoch muss der Marching Cubes-Algorithmus über den gesamten Suchraum, also über alle Gitterpunkte iterieren, da auch ihm als Eingabe nur die Ladungsverteilung und der Isowert, nicht aber die Isofläche vorliegt. Ob der Algorithmus gerade eine leere Zelle bearbeitet, steht also erst fest, wenn das Feld an allen Eckpunkten dieser Zelle berechnet wurde.

Um zu zeigen, dass das obige Beispiel repräsentativ für alle nicht destruktiven Fälle ist, versuchen wir nun Bedingungen herzuleiten, unter denen die Anzahl der Messpunkte für beide Algorithmen gleich ist. Zu diesem Zweck verwenden wir das Ergebnis des Shrink-Wrap-Verfahrens, welches wir als Vorgabe definieren, und stellen Modifikationen am Marching Cubes-Algorithmus an, mit dem Ziel, dessen Iterationspunkte in der Anzahl auf die des Shrink-Wrap-Verfahrens zu reduzieren.

Folgen wir der Definition des Marching Cubes-Algorithmus streng, muss der Abstand der Gitterpunkte in jeder Dimension gleich groß sein. Unter dieser Anforderung müsste, sollten alle

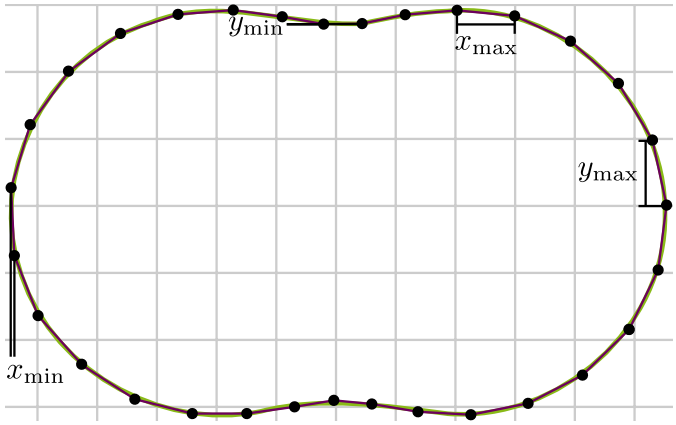


Abbildung 4.6: Beispielhafte Darstellung für die Anzahl der zu berechnenden Punkte für Shrink-Wrap-Verfahren (dunkelmagenta) und das Zellengitter des Marching Squares-Algorithmus (grau)

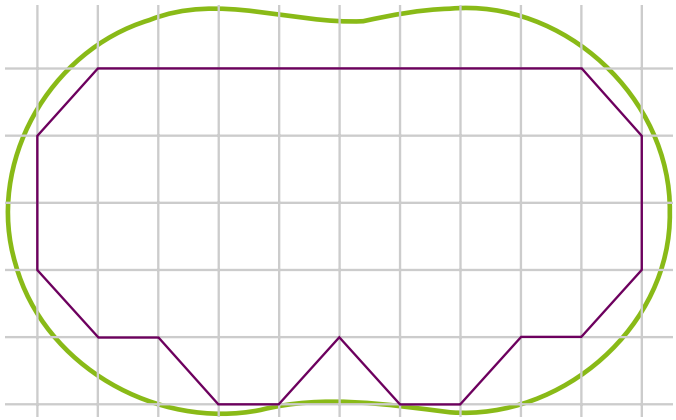


Abbildung 4.7: Ergebnis des Marching Squares-Algorithmus (dunkelmagenta) zu einer Isofläche (grün) mit den Gitterabständen $(x_{\max}, y_{\max})$

Punkte der Vorgabe abgebildet werden, der Abstand der Gitterpunkte in allen drei Richtungen dem kleinsten Abstand zweier Punkte der Vorgabe entsprechen. Ein größerer Abstand ginge mit einem Informationsverlust einher. Dieses Vorgehen bewirkt eine große Menge an Messpunkten und gleichzeitig eine deutlich höhere Detaildichte als die der Vorgabe.

Ebenso wie ein minimaler Punktabstand in allen Achsen kann auch ein maximaler bestimmt werden. Bei dieser Wahl der Gitterabstände kann der Marching Cubes nur noch die gröbsten Details der Vorgabe erfassen, aber nicht mehr alle. Dennoch minimieren wir so die Anzahl der nötigen Messpunkte: Eine noch geringere Anzahl würde bedeuten, dass die etwaige Details nur noch zufällig günstig fallende Gitterpunkte dargestellt werden. Die Qualität des Meshes bleibt, wie in Abbildung 4.7 anhand des Marching Cubes-Algorithmus für zweidimensionale Probleme dargestellt, bereits hier weit hinter der der Vorgabe zurück, wohingegen die Anzahl der Messpunkte weiterhin größer ist, als die der Vorgabe. Eine weitere Reduktion dieser Messpunkte ist nur noch durch Erhöhen ihrer Abstände über die größten Abstände der Vertices der Vorgabe hinaus möglich.

Für Fälle mit $x_{\max} \neq y_{\max}$ haben wir die Anforderung des Marching Cubes-Algorithmus an würfelförmige Zellen bereits aufgehoben. Lösen wir die Kriterien des Marching Cubes soweit, dass die Größe der Zellen untereinander variabel sein kann, ist es möglich, innerhalb einer Achse die Abstände so zu wählen, dass alle Punkte der Vorgabe wiedergegeben werden. Im zweidimensionalen Fall entspricht die Anzahl der Zellen in einer Achse dann bereits der Anzahl der Punkte der Vorgabe. In der anderen Dimension dürfte das Gitter nur aus einer Zelle mit der Höhe des gesamten Suchraums bestehen, um ein Ergebnis zu erzielen, das dem Anspruch genügt, gleich viele Messpunkte wie die Vorgabe zu haben. Eine so geringe Auflösung ließe keinerlei Details erkennen. Für dreidimensionale Fälle erfolgt die Argumentation analog: Die letzte verbleibende Dimension kann nur aus einer einzigen Zelle bestehen, die die gesamte Höhe des Gitters einnimmt. Ein Zustand mit gleicher Anzahl an Messpunkten für beide Verfahren lässt sich also herbeiführen, allerdings ist das Ergebnis des Marching Cubes-Verfahrens hier nicht mehr verwendbar.

Eine weitere Aufhebung der Kriterien des Marching Cubes-Algorithmus müsste die Quader des Gitters in allgemeinere Formen wie Parallelepipede und schließlich Rhomboeder oder Prismen überführen. Sofern diese parkettierbar sind, ergeben sich aber die gleichen Probleme, solange sie sich nicht in irgendeiner Weise die Form der Isofläche zunutze machen, ohne diese zu kennen.

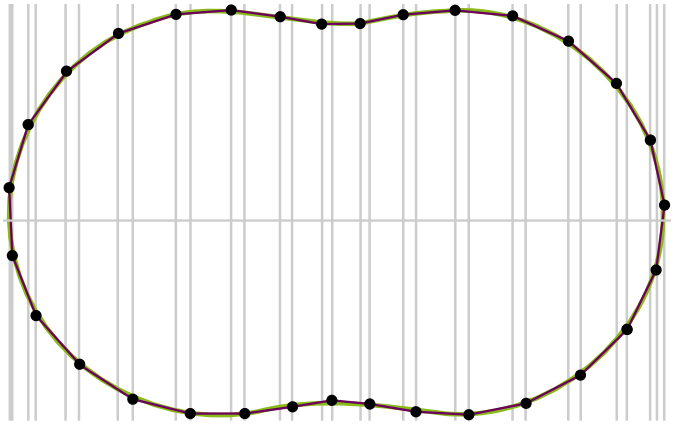


Abbildung 4.8: Das Gitter enthält die gleiche Anzahl der Messpunkte, aber nur eine Zelle in der $y - Achse$

4.6 Gegenüberstellung der Verfahren

Ziel dieses Kapitels ist die Gegenüberstellung der Verfahren, die sich im Verlauf dieser Arbeit als geeignet herausgestellt haben, eine Isofläche durch ein Mesh anzunähern, namentlich des Shrink-Wrap-Verfahrens im Rahmen einer Simulation, der schrittweisen Suche und dem Marching-Cubes-Algorithmus, um auf eine allgemeine Empfehlung zur Implementierung einer Lösung für allgemeine und spezifische Problemstellungen hinzuarbeiten. Dazu beleuchten wir verschiedene Aspekte wie die Laufzeitkomplexität und den Umgang mit disjunkten Isoflächen und Artefakten. Diese Aspekte betrachten wir aufgeschlüsselt nach Verfahren und wägen dieses gegeneinander ab.

Marching Cubes Der Marching Cubes-Algorithmus stellt sich als langsamster Kandidat für die Annäherung von Isoflächen durch Meshes heraus: Zwar ist seine asymptotische Laufzeit von $\mathcal{O}(n^2)$ vergleichbar mit der anderer Verfahren, jedoch wird die durchschnittliche Laufzeit durch Iteration über Leerraum erhöht. Darüber hinaus wird zusätzlicher Zeitaufwand zur Nachbearbeitung der Meshes benötigt, dessen genaues Ausmaß unter anderem von der verwendeten Lookup-Tabelle abhängt. Der Algorithmus kann für das Problem lediglich eine initiale Lösung berechnen; eine Verfeinerung der Ergebnisse beispielsweise durch Interpolation ist denkbar, aber bei den derzeitigen Einsatzgebieten des Algorithmus selten zielführend. Bei einer Veränderung der Ladungsverteilung ist aber eine Neuberechnung des Meshes nötig. Fehlannahmen von Äquivalenzen sind nicht möglich; Artefakte können bei ungünstiger Wahl der Auflösung entstehen, aber durch in dieser Arbeit vorgestellte Methoden weder erkannt noch behoben werden, so dass zeitaufwändige Neuvermaschungs-Algorithmen zum Einsatz kommen müssen. Der Marching Cubes-Algorithmus erstellt das Mesh agnostisch zu disjunkten Isoflächen: Da er über den gesamten Suchraum iteriert, stellt sich die Frage nach der Äquivalenz von Punktladungen

Kriterium	Marching Cubes	Schrittweise Suche	Shrink-Wrap
Laufzeit	$\mathcal{O}(n^2)$	$\mathcal{O}(n \log n)$	$\subseteq \mathcal{O}(n^2)$, $\mathcal{O}(n^2)$
Nachbearbeitung	mind. $\mathcal{O}(n^2)$	nicht nötig	nicht nötig
Speicherbedarf	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$ oder $\mathcal{O}(n)$
Auflösungsanpassung	nicht möglich	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Dimensionen	3D	2D	3D
Verfeinerung	Interpolation	nicht möglich	Simulation
Inkl. Leerraum	$T(n^4)$	$T(n^2)$	$T(n^2)$
Ladungsverteilung	Statisch	Statisch	Dynamisch
Flächen	beliebig	eine	beliebig
Disjunkte Isoflächen	automatisch	nicht möglich	gesondert
Suchraumbestimmung	nicht möglich	nicht anwendbar	Simulation
Artefakte	Neuvermaschung	Keine	Auflösungsanpassung

Tabelle 4.1: Gegenüberstellung der vorgestellten Verfahren

nicht. Die Darstellung einzelner gezielter Flächen ist nicht möglich. Außerdem muss der Suchraum im Voraus bekannt sein.

Schrittweise Suche Die schrittweise Suche bietet einzig hinsichtlich der asymptotischen Laufzeit mit $\mathcal{O}(n \log n)$ gegenüber den anderen vorgestellten Verfahren einen Vorteil. Iterationen über Leerraum finden nicht statt. Sie eignet sich lediglich zur Annäherung einer einzelnen Isofläche und bleibt im Einsatz auf zweidimensionale Fälle beschränkt. Auch kann die schrittweise Suche nicht dazu verwendet werden, disjunkte Flächen darzustellen. Ihre Verwendbarkeit grenzt sich daher auf Sonderfälle ein.

Shrink-Wrap Dem Shrink-Wrap-Verfahren ist zueigen, dass es neben der Anpassung des Meshes während der Simulation einen zusätzlichen Aufwand für die Generierung des Meshes benötigt. Die Wahl der Startgeometrie hängt davon ab, wie die Simulation auf das Feststellen einer falschen Äquivalenzrelation reagiert. Falls kein schnelleres Verfahren zur Generierung zur Verfügung steht oder die Simulation kritisch sowohl hinsichtlich der Speicher- als auch der Laufzeitkomplexität ist, keine Informationen über die Cluster benötigt werden und der Suchraum bekannt ist, kann dafür sogar der Marching Cubes-Algorithmus eingesetzt werden. Im Gegensatz zum Marching Cubes-Algorithmus kann zu Lasten der durchschnittlichen Laufzeitkomplexität die Auflösung des Resultats während des Verlaufs gezielt erhöht werden.

Die Tabelle 4.1 stellt die obigen Erkenntnisse gegenüber. Aus ihnen lässt sich ableiten, dass in allgemeinen, während der Simulation zeitkritischen Anwendungen mit dynamischer Ladungsverteilung das Shrink-Wrap-Verfahren zu wählen ist.

4.7 Heuristiken zur Abschätzung der Cluster

Die Aufgabe der oben aufgeführten Verfahren ist, die Struktur der Meshes so zu generieren, dass sie disjunkte Flächen korrekt repräsentieren. Der Marching Cubes-Algorithmus erreicht diese Repräsentation durch Iteration über den gesamten Suchraum, beim Shrink-Wrap-Verfahren während einer Simulation durch Reaktion auf falsch angenommene Äquivalenzverhältnisse der Punktladungen sichergestellt. Die schrittweise Suche hingegen vernachlässigt jedoch, dass sie die Isofläche möglicherweise nicht vollständig wiedergibt. Sie kann jedoch verwendet werden, um eine Heuristik zur Erstellung der Cluster zu ermöglichen. Ihr Ergebnis kann auch für das Shrink-Wrap-Verfahren nutzbringend sein, weil es einigermaßen zuverlässige Annahmen bezüglich der Äquivalenzverhältnisse der Punktladungen treffen kann. Bisher war das aufgrund der Arbitrarität der Cluster überhaupt nicht möglich.

4.7.1 Wiederholte schrittweise Suche

Das in Alg. 12 beschriebene Vorgehen erfordert die Erweiterung der Datenstruktur um eine doppelte Markierbarkeit, also zwei zusätzliche Datenelemente, die jeweils die betroffene Punktladung als „markiert“ oder „nicht markiert“ kennzeichnen. Die Bestimmung der Cluster der Ladungen kann durch Iteration des Algorithmus 10 mit einem Punkt bei $\mathbf{p}_+$ erfolgen, der so groß ist, dass ein Kreis mit dem Zentrum am Durchschnittspunkt aller Ladungen bei $\mathbf{m}$ und dem Radius $\|\mathbf{p}_+ - \mathbf{m}\|$ alle Ladungen umschließen würde. Eine Durchführung der schrittweisen Suche mit dem Anfangspunkt $\mathbf{p}_+$ würde also in einem Polygon resultieren, das alle Punktladungen beinhaltet. In jeder Iteration der schrittweisen Suche wird die nächstgelegene Punktladung markiert. Wann immer in einer folgenden Iteration beim Verschieben von $\mathbf{p}_+$ in Richtung $\mathbf{p}$ eine Ladung nicht markiert wurde, ist ein neues Cluster gefunden worden, und ein weiterer Punkt $\mathbf{r}_+$ mit $\|\mathbf{E}(\mathbf{r}_+)\| = \iota$ muss in Richtung des Zentrums der eingeschlossenen Ladungen gefunden werden. Es handelt sich also um ein rekursives Verfahren. Die Weite des Suchschritts kann beliebig gewählt werden: Eine höhere Schrittweite führt dazu, dass mehr Punktladungen in einem Iterationsschritt die Markierung wechseln, die in einem weiteren Rekursionsschritt behandelt werden müssen. Die Schrittweite in der Größenordnung des Abstands der Punktladungen zueinander zu wählen trägt der Laufzeit zu: Eine größere Schrittweite wird dazu führen, dass Punktladungen in verschiedenen Clustern zwischen zwei aufeinanderfolgenden Durchläufen ihre Markierung wechseln. Bei kleinerer Schrittweite werden häufig keine Punktladungen die Markierung wechseln, was für diesen Durchlauf keine Erkenntnis bezüglich des Clusterings liefert. Ein Beispiel für einen einzelnen Schritt dieses Vorgehens ist in Abbildung 4.9 dargestellt.

4.7.2 Vollständiges Clustering

Im folgenden entwickeln wir einen Ansatz zur Annäherung des vollständigen Clusterings. Dieses baut auf Alg. 12 auf. Als Eingabe erwartet es die Punktladungen, wobei die Positio-

Algorithmus 12 : Clustering

eingabe : Menge der doppelt markierbaren Punktladungen $\mathcal{Q} = \{(q_0, (m_0, n_0)), (q_1, (m_1, n_1))\}$, Startpunkt $\mathbf{p}$ oder Isowert ξ, optional Grenzpunkt $\mathbf{p}_+$

ausgabe : Polygone P

$\xi = \mathbf{E}(\mathbf{p})$;
$\mathbf{m} = \frac{1}{\#\mathcal{Q}} \sum_{q \in \mathcal{Q}} \mathcal{Q}_{\mathbf{p}}$;
wenn *Grenzpunkt nicht angegeben* **dann**
    $\mathbf{p}_+ = \mathbf{m} + \{q \in \mathcal{Q} | \max(\|\mathbf{m} - q\|)\} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$; `// Beliebiger Vektor mit Länge`
    ; `// des größten Abstands`
Ende
Schrittweise Suche an Punkt $\mathbf{p}_+$; Bestimme in jedem Schritt die nächste Punktladung q, m, n; $m \leftarrow \top$;
Ergebnispolygon e;
solange $\mathbf{p}_+$ *hinter* $\mathbf{p}$ **tue**
    Schrittweise Suche an Punkt $\mathbf{p}_+$; Bestimme in jedem Schritt die nächste Punktladung q, m, n; $n \leftarrow \top$;
    Verschiebe $\mathbf{p}_+$ in Richtung von $\mathbf{m}$;
    Folge $\mathcal{Q}'$;
    für $(q, m, n) \in \mathcal{Q}$ **tue**
        wenn $m \neq n$ **dann**
            Füge (q, m, n) $\mathcal{Q}'$ hinzu;
        Ende
        Füge P das Ergebnis von Clustering($\mathcal{Q}'$,ξ,q_p) hinzu;
        Tausche(m,n);
    Ende
Ende
Rückgabe e;

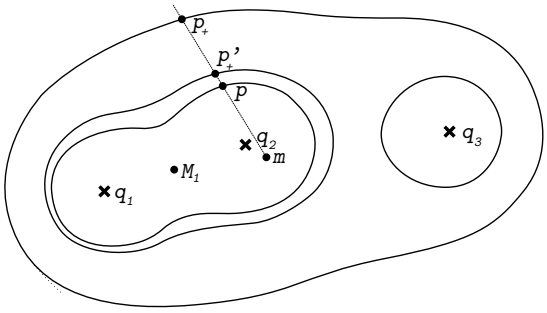


Abbildung 4.9: Einordnung der Punktladungen in Äquivalenzklassen

nen einer Koordinate nach sortiert sein müssen. Das Verfahren basiert darauf, die Strecke der aufeinanderfolgenden Punktladungen zu dritteln und an beiden Zwischenpunkten eine vorläufige Fläche durch die schrittweise Suche zu generieren. Der an diesen Punkten gemessene Isowert ist ebenfalls Teil der Datenstruktur. Das Ergebnis ist eine Liste, die in der Hinsicht einen Ersatz für einen Konturbaum darstellt, indem sie alle möglichen Meshes generiert mit einer Zusatzinformation, wann diese ein- und auszublenden sind: Eingeblendet werden alle Meshes, deren zugehörige Isowert überschritten wird. Dabei ist zu bemerken, dass es sich bei dem Ergebnis lediglich um eine Annäherung der Mesh-Struktur handelt: Generiert werden lediglich die Meshes selber. Da die Isowerte nicht genau bestimmt werden, geben sie die Isofläche nur grob wieder. Für makroskopische Probleme mit großen Clustern, die vergleichsweise weit voneinander entfernt sind oder denen ein kleiner Isowert zugeordnet ist. Ebenfalls kann während des Rückgriffs auf die schrittweise Suche nur zweidimensionale Geometrie generiert werden.

Algorithmus 13 : Heuristik für das vollständige Clustering

1 **für** $i \in x | x \in \mathbb{N}, x < \#Q - 1$ **tue**
2 $\quad \mathbf{s} \leftarrow (Q_{i+1})_p - (Q_i)_p$;
3 $\quad \mathbf{p}_1 = \mathbf{s} + (\mathbf{l}\frac{1}{3})$;
4 $\quad \mathbf{p}_2 = \mathbf{s} + (\mathbf{l}\frac{2}{3})$;
5 $\quad$ Polygon $p \leftarrow$ schrittweise Suche an Punkt $\mathbf{p}_1$;
6 $\quad$ Füge $(p, \mathbf{E}(\mathbf{p}_1))$ dem Clustering hinzu;
7 $\quad p \leftarrow$ schrittweise Suche an Punkt $\mathbf{p}_2$;
8 $\quad$ Füge $(p, \mathbf{E}(\mathbf{p}_2))$ dem Clustering hinzu;
9 **Ende**

KAPITEL 5

Parallelisierbarkeit der Algorithmen

Alle gängigen Programmiersprachen, darunter C und C++ [Ban20, S. 355], Java [SBG22, S. 489], Python [Hat16, S. 167] und Haskell [Men14, S. 187] durch Multithreading, JavaScript [DSK17, S. 2] durch *Web Workers*, bieten Möglichkeiten zur einfachen Umsetzung parallelisierter Algorithmen. Dies motiviert die Untersuchung der Parallelisierbarkeit der in dieser Arbeit vorgestellten Verfahren. Einige dieser Verfahren stellen Operationen auf Monoiden dar, die stets parallelisierbar sind. Wir leiten diese Argumentationsweise her, indem wir zunächst Monoiden definieren und einige Beispiele anführen. Danach begründen wir die Parallelisierbarkeit von Monoiden durch ihre assoziative Eigenschaft. Schließlich begründen wir für jedes Verfahren separat seine Parallelisierbarkeit – wenn möglich durch Aufzeigen der monoidalen Struktur – oder seine Nicht-Parallelisierbarkeit.

Aufgrund seiner weiten Verbreitung lohnt es sich hervorzuheben, dass der Marching Cubes-Algorithmus aus Abschnitt 4.3 nicht parallelisierbar ist: Es ist zwar möglich, alle Zellen des Marching Cube-Algorithmus unabhängig voneinander zu berechnen, allerdings würde dies ein eigenes offenes Mesh für jede Zelle produzieren, das bei Konkatenation aller Meshes ein einziges Mesh ergäbe, das neben dem Kriterium der Geschlossenheit ebenfalls das Kriterium der Sauberkeit nicht erfüllt. [GZ22] legen ein Verfahren für Sonderfälle für den Bereich des Medical Imaging vor, das die Vertices des rekonstruierten Meshes „fast überall“ (*almost everywhere*) exakt platziert, ohne allerdings den Begriff „fast überall“ zu quantifizieren. Dafür wenden sie den von [Nie04] vorgestellten Dual Marching Cubes-Algorithmus an. Dieser erstellt für jede Zelle den Vertex eines Meshes, die durch Nachbearbeitungsschritte zu einem gesamten quadrilateralen Mesh verbunden werden. Die Voxel lassen sich zellenweise unabhängig voneinander generieren. Durch Erweiterung der Datenstruktur um die Position der Zelle können die benachbarten Vertices als Information erhalten werden. Dadurch lassen sich auch die Kanten und Flächen des Meshes unabhängig voneinander erstellen. Somit ist der Gesamtprozess parallelisierbar, aber auch aufwändiger: Sowohl [Nie04] als auch [GZ22] geben keine Hinweise auf die Laufzeit des Algorithmus, allerdings baut dieser auf dem Marching

Cubes-Algorithmus auf und erweitert diesen um die Erstellung des Meshes aus den Vertices. Selbst unter der Annahme, dass die asymptotische Laufzeit gleich ist, wird doch die durchschnittliche Laufzeit die des Marching Cubes-Algorithmus übersteigen. Ein weiterer Versuch der parallelisierten Ausführung des Marching Cube-Algorithmus findet sich in [MN97]. Der Arbeit fehlt aber ein Beweis der erfolgreichen Parallelisierung, der über die Interpretation des Plots eines Laufzeitmessungsexperiments hinausgeht.

Diese Arbeit macht sich an unterschiedlichen Stellen Monoiden dienstbar. Monoiden ermöglichen auf natürliche Weise Parallelisierung unter anderem beim Einsatz von Listenkombinatoren [CB14, Ss. 259ff] oder der Konstruktion von Homomorphismen. Von besonderem Interesse ist im Folgenden die Eigenschaft der Assoziativität der Monoiden bezüglich der binären Operation. Einige der besprochenen Datenstrukturen weisen zwar neutrale Elemente auf. Diese sind aber, wie wir sehen werden, für das Aufzeigen einer Parallelisierbarkeit unerheblich. Die Argumentationen könnten somit auch anhand von Halbgruppen erfolgen. Die zugrundeliegenden Erkenntnisse entstammen allerdings der Kategorientheorie, die ihrerseits keine Halbgruppen kennt, die nicht auch Monoiden sind. Wir halten es aber nicht für nötig, das Thema der Kategorientheorie qualifiziert einzuführen, um die nachfolgenden Punkte darzulegen. Um Verwechslung zu vermeiden werden wir daher dennoch den Begriff „Monoid“ verwenden.

Beispiele

1. Zeichenketten, wie sie aus der Programmiersprache C bekannt sind, sind Monoiden: Laut Kapitel 2 bezeichnen Folgen eine geordnete Menge mit einer binären, Operation, der Konkatenation. Diese ist ebenfalls assoziativ [Bec07, F. 53]. Die leere Zeichenkette `""` ist als neutrales Element zu verstehen. Zeichenketten sind also Monoiden.
2. In Abschnitt 3.3.4 wird die Existenz eines Halbgruppenhomomorphismus zwischen der Punktliste eines Polygons und einem als Listenkombinator umgesetzten Prädikat zur Entscheidung eines Punktbeinhaltungstests.

Ein Algorithmus, der die Funktion f implementiert, die zwei Monoiden aufeinander abbildet, lässt sich aufgrund der assoziativen Eigenschaft parallelisieren. Seien m_1, m_2 usw. Elemente des Monoiden $(M_1, +)$. Dieser lässt sich auch aufgeschlüsselt nach den kombinierten Elementen oder mit dem Summenzeichen als Reduktion notieren:

$$(M_1, +) = m_1 + m_2 + \cdots = \sum_{i \in \mathbb{N}} m_i = +/M_1$$

notieren. Der gesamte Monoid soll nun durch diese Funktion f mit einem weiteren Monoid $(M_2, \star)$ als Ergebnis bearbeitet werden. Die Rechnung erfolgt durch die Operationen. Unter

Anwendung des Assoziativgesetzes können die Klammern der Addition hinzugefügt werden, um die Addition anzudeuten:

$$m_1 + (m_2 + (m_3 \dots)))$$

Die sequentielle Abarbeitung dieser Elemente erfolgt durch

$$f(m_1 \star f(m_2 \star f(m_3 \dots)))$$

Das erneute Anwenden des Assoziativgesetzes erlaubt das Entfernen der Klammern:

$$f(m_1) \star f(m_2) \star f(m_3) \dots$$

Verstanden als assoziative algebraische Struktur mit einer binären Operation und neutralem Element erlaubt die assoziative Eigenschaft bildlich gesprochen das „Auftrennen“ des Datentyps anhand des binären Operators und das individuelle Anwenden der Funktion auf einzelne Datenelemente. Ist das Bild ebenfalls ein Monoid, kann hier wiederum die assoziative Eigenschaft der Monoiden zu Hilfe genommen werden, um die Ergebnisse der Funktion anhand des dort vorliegenden binären Operators als Teilergebnisse zu einer Datenstruktur zusammenzusetzen, die das Ergebnis repräsentiert. Somit sind die Einzelergebnisse der Berechnung nicht voneinander abhängig und bilden – ebenfalls unabhängig von der Laufzeit der einzelnen Berechnungen – sobald alle ermittelt sind das Endergebnis.

Bei der Einführung aller in dieser Arbeit neu präsentierten Datenstrukturen wurde, sofern vorhanden, die ihre monoidale Struktur aufgezeigt; die folgenden neu präsentierten Algorithmen operieren auf diesen:

- Das Shrink-Wrap-Verfahren in Abschnitt 4.5 iteriert über die Vertex-Folge aller Meshes. Hierbei berechnet die parallelisierbare Funktion eine Verschiebungsmatrix für den zu bearbeitenden Vertex und wendet diese an.

- Das invertierbare der in Abschnitt 3.3.13 vorgestellten Verfahren zur selektiven Anpassung der Auflösung von Meshes operieren auf den Vertex-, Kanten und Flächenfolgen der Meshes. Zur Erläuterung sei $a_1, a_2, a_3, \dots, a_i, \dots$ die Kantenliste und $a_i = \mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3}$ die Punktliste der Fläche i-ten Fläche. Die zugehörige Abbildung lautet:

$$\begin{aligned} &\{\dots, a_{i-1}, \{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3}\}, a_{i+1}, \dots\} \\ \mapsto &\{\dots, a_{i-1}, \{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,m}\}, \{\mathbf{p}_{i,m}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3}\}, \{\mathbf{p}_{i,3}, \mathbf{p}_{i,1}, \mathbf{p}_{i,m}\}, a_{i+1}, \dots\} \end{aligned}$$

- Wie in Abschnitt 3.3.2 ausgeführt, stellt der Punktbeinhaltungstest für Polygone einen Isomorphismus in Halbgruppen – angepasst an die Bezeichnungen dieses Abschnitts:

in Monoiden – dar. Der Polyeder-Punktbeinhaltungstest aus Abschnitt 3.3.3 stellt eine Erweiterung dieses Verfahrens für drei Dimensionen dar.

Besondere Betrachtung erfordert die Konstruktion von Icospheres durch Subdivision. Zwar bilden sowohl Meshes als auch Polygone Folgen, die ihrerseits monoidale Struktur aufweisen. Allerdings sind die Elemente der Folgen des Meshes bezüglich jeder Subdivisionsoperation nicht abgeschlossen. In [Mla+19] gelingt es, die Catmull-Subdivision auf dünn besetzte Matrizen bezüglich der Multiplikation zurückzuführen, die insbesondere einen Monoiden darstellen, und ihre Berechnung parallelisiert auf mehreren Kernen der GPU durchzuführen. Durch diesen Ansatz ist auch die Subdivision parallel umsetzbar.

Die folgenden Verfahren können sich die monoidale Struktur der Datenstrukturen nicht zunutze machen und sind auch anderweitig nicht parallelisierbar:

- Das Auftrennen und Zusammenfügen von Meshes aus Abschnitt 3.3.4: Die Geometrie zwischen zwei Öffnungspolygonen ist nur sequentiell abzuarbeiten, da zu Beginn nur zwei Punkte miteinander verbunden werden können. Alle weiteren Kanten des schließenden Polygons sind von den Ergebnissen der vorherigen Schritte abhängig.

- Auch wenn die Potenzmenge parallelisiert bestimmbar ist, ist laut [Gib20, S. 10] die Konstruktionsoperation für Bäume zwar homomorph zur Konkatenation von Folgen, allerdings ohne dabei die Assoziativität zu erhalten. Eine Parallelität für die Konstruktion von Konturbäumen ist also aus der obigen Argumentation nicht ableitbar. Auch für die in Abschnitt 3.3.7 vorgestellten g und f sind nicht parallelisiert anwendbar. Dies lässt sich durch ein Gegenbeispiel aufzeigen: Gegeben sei der markierte Konturbaum $\{a', b'\}^\circ$ mit beliebigen Unterbäumen a und b. Die Funktionen $f(b)$ und $f(b)$ resultieren in unterschiedlichen Bäumen, je nachdem, in welcher Reihenfolge sie angewendet werden. Eine parallele Ausführung kann zum Auftreten einer Wettlaufsituation führen. Dazu ist zu bemerken, dass beide Anwendungen den Baum in einen undefinierten Zustand versetzen, wenn zuvor die jeweils andere ausgeführt wurde, weil f auf markierten Knoten nicht definiert ist. Analoge Überlegungen gelten für g. Da zu jedem Zeitpunkt nur eine Falschannahme bezüglich der Äquivalenzverhältnisse der Flächen erkannt werden kann, könnte eine gleichzeitige Ausführung mehrerer Operationen für g und f und die daraus resultierenden Probleme nur einer fehlerhaften Implementierung geschuldet sein. Korrekterweise kann immer nur eine der Funktionen gleichzeitig ausgeführt werden. Daraus folgt aber nicht die Parallelisierbarkeit dieser Operationen.

- Der schrittweise Suchalgorithmus aus Abschnitt 4.4.1 ermöglicht keine Möglichkeit der Parallelisierung: Jeder Schritt ist abhängig vom Ergebnis des vorherigen Schritts.

KAPITEL 6

Makro-kontrollierte prozedurale Generierung für Spielgeometrie

Ein untergeordnetes Ziel dieser Arbeit ist, die Erkenntnisse aus dem Hauptteil auf ein Themengebiet abseits der Elektrotechnik, aber noch innerhalb des Themenkreises der Virtual- und Augmented Reality anzuwenden. Kapitel 4.6 enthält die Feststellung, dass der Algorithmus, der eine einzelne Fläche einer Isofläche durch ein zweidimensionales Mesh annähert, auf Sonderfälle begrenzt ist. In diesem Kapitel heben wir die Videospielentwicklung als einen solchen Sonderfall hervor: Wir verwenden die schrittweise Suche, um zu einem beliebig konstruierten zweidimensionalen Vektorfeld eine Repräsentation seiner Isoflächen zu generieren und diese durch Mittel wie Extrusion in eine zweidimensionale Geometrie zu überführen. Videospiele sind zwei- oder dreidimensionale, interaktive und visuelle Simulationen, die der Unterhaltung dienen. Diese beinhalten in der Regel zwei- oder dreidimensionale Geometrie. Videospiele sind ein Beispiel für visuelle Simulationen mit einem besonderen Anspruch an die Komplexität der dargestellten Geometrie, sowohl was die Szenerie als auch Charaktere, Gegenstände, etc. angeht: Die PlayStation der ersten Generation war für das Rendern von 3600000 Polygonen ausgelegt, solange lediglich primitive Schattierungsverfahren wie *flat shading* zum Einsatz kamen. Die Erstellung dieser Szenerie durch die Bearbeitung von Meshes und der damit verbundene besondere künstlerische Aspekt macht die Erstellung von Videospiel-Szenerien besonders zeitaufwändig, insbesondere bei umfangreichen Open-World-Anwendungen.

Diesem Umstand wird häufig durch den Einsatz von Werkzeugen der *prozeduralen Generierung* entgegengewirkt [SA17]. Üblicherweise wird Rauschen verwendet, um eine *Height-Map* zu erzeugen, die von einem Designer manuell nachbearbeitet wird, um eine Umgebung für Geometrie für Spielobjekte wie Gebäude, Pickups, NPCs, Dekoration usw. zu schaffen, die ebenfalls manuell platziert werden müssen.

Einige Videospiele, vor allem aus dem Roguelike-/Roguelite- oder Open-World-Survival-Genre, enthalten spezielle Funktionen für die prozedurale Generierung der Spielumgebung

[MO17]. Zwar sind die Produktionsregeln in diesem Fall etwas aufwändiger, folgen aber dem selben Prinzip.

Spiellandschaften, insbesondere dreidimensionale, zeichnen sich dadurch aus, dass sie sich entweder auf eine bestimmte Art von Infrastruktur konzentrieren (z. B. einen *Dungeon*) oder eine dünne Population infrastrukturähnlicher geometrischer Primitiva aufweisen, die den Design-Vorgaben nach sich zwar nicht überschneiden dürfen, aber dennoch zufällig aussehen müssen und kaum Größenanpassungen zulassen, was den Einsatz von Relaxations-Algorithmen stark erschwert. Ein Beispiel hierfür sind Open-World-Sandboxes wie Minecraft.

Der Grund dafür hat eine ökonomische Dimension: Obwohl menschliches Handeln nicht quantifizierbar ist, können wir dennoch sagen, dass ein Designer, der eine Maschine im Sinne der Mensch-Maschine-Interaktion einsetzt, ein gewünschtes Ergebnis mit einem Minimum an Interaktionshandlungen erzielen möchte. Eine rein zufällig generierte Spielumgebung – also eine solche, die keine Interaktionshandlungen für ihre Erstellung erfordert – kann die spezifischen Vorstellungen eines Designers, die in einer informatischen Datenstruktur ausgedrückt werden könnten, möglicherweise nicht umsetzen, während die rein manuelle Erstellung einer exakten Darstellung der Vorstellungen des Designers im Detail die maximale Anzahl der Interaktionshandlungen erfordert. Je mehr menschliche Kontrolle wir also zulassen, desto mehr Automatisierung werden wir aufgeben müssen und umgekehrt. Wenn wir keine reine zufällige Generierung anstreben, ist ein gewisses Maß an Interaktion erforderlich, um die Vorstellungen einer Maschine mitzuteilen, die sie perfekt umsetzen würde. Jede darüber hinausgehende erforderliche Interaktion ist unerwünscht.

6.1 Makro-kontrollierte Erstellung von Geometrie für Videospiele

6.1.1 Rein prozedurales Generieren und prozedurales Generieren mit Mikro- und Makro-Kontrolle

Wir werden zunächst die Begriffe „rein prozedurales Generieren“ und „Mikro-Kontrolle“ einführen, bevor wir uns an einer Definition der Makro-Kontrolle versuchen. Dafür greifen wir auf den Begriff des informatischen Prozesses zurück.

Prozess

Ein Prozess $I \rightarrow O$ bezeichnet eine Abbildung einer Menge Eingabeobjekte I auf eine Menge Ausgabeobjekte O.

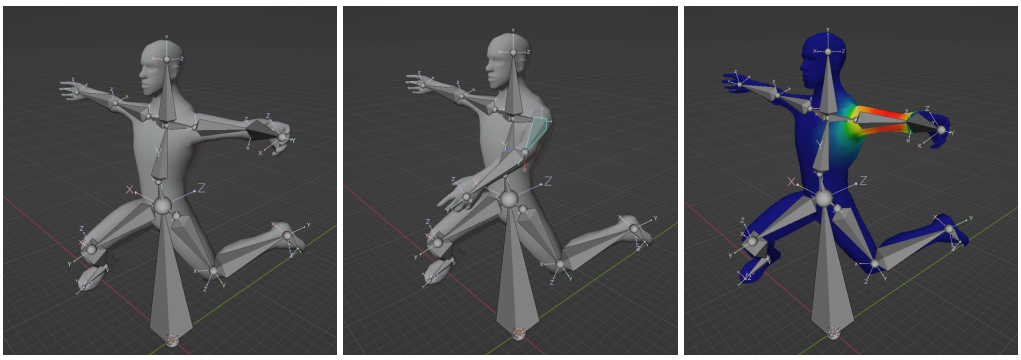
Ein Prozess unterscheidet sich insofern von einer Funktion, als dass Fragen der praktischen Informatik, wie nach der Programmiersprache der Implementierung oder ob eine Fehlerbehandlung durchgeführt wird und der Mathematik, wie nach der Wohldefiniertheit oder ob es sich um eine Bijektion oder um eine reine Funktion handelt, keine Beachtung finden. Diese Abgrenzung ist sinnvoll, weil wir Prozesse als prinzipiell von Menschen durchführbar verstehen wollen. Unter anderem um die Durchführung durch Menschen nicht zu einem notwendigen Kriterium zu machen, wollen wir aber die Kriterien der Komposition und Assoziativität aus beiden oben erwähnten Disziplinen für Prozesse übernehmen.

Die Definitionen hängen von der gewünschten Art der Ausgabe ab, an die zunächst nur der Anspruch gestellt wird, dass es sich dabei um ein Objekt eines beliebigen Typs handeln soll. Passende Beispiele für den Kontext der Spieleentwicklung sind Punkte, Flächen, Pixel oder Modellmatrizen.

Rein prozedurales Generieren

Wir nennen einen Prozess $P : I \to O$ einen *rein prozeduralen Generator*, wenn er lediglich angesichts der Eingabe I und des Prozesses selbst keine Aussage über den Ausgang O ermöglicht, außer durch Abarbeiten des P selbst.

Die Unmöglichkeit, eine Aussage über O zu treffen, impliziert, dass auch I nicht gewählt werden kann, um O gezielt zu beeinflussen. Rein prozedurales Generieren folgt typischerweise festen Regeln und hat zum Zweck, einen Spieler mit unerwartbaren Spielinhalten zu konfrontieren. Eine Abarbeitung durch einen Menschen ist also nicht wünschenswert und häufig auch nicht in angemessener Zeit möglich.



(a) Ursprüngliches Mesh mit Bewegungsskelett (b) Auswirkung der Rotation eines Knochens (c) Weight Map für einen Knochen

Abbildung 6.1: Geriggtes Modell [ras14]

Beispiele für makro-kontrollierte Prozesse

- Der Perlin-Rauschprozess (*Perlin Noise*) erzeugt als Ausgabe ein diskretes Skalarfeld, das als Graustufen aufgetragen ein Zufallsbild mit weichem, wolkenartigen Strukturen wie in Abb. 6.2 erzeugt. Seine Eingabe ist eine Zufallszahl (Seed) und eine Zellengröße. Wir können nur durch Kenntnis dieser beiden Werte weder Aussagen über die Helligkeit eines bestimmten Pixels treffen, noch können wir diese Werte wählen, um die Helligkeit eines Pixels zu erwirken oder gezielt Strukturen zu erzeugen.
- Das *Rigging* (Animation durch Bewegungsskelette) dreidimensionaler Modelle realisiert eine gewichtete Abbildung von Vertices des Meshes auf ein Bewegungsskelett. Durch diese Abbildung ist es möglich, das Mesh durch Manipulation des Skeletts makrokontrolliert zu bewegen [DP11, S. 424ff]. Dieser Vorgang ist in Abbildung 6.1 illustriert. Die Position vieler Vertices ist nach einer Operation auf dem Bewegungsskelett von Menschen gut abzuschätzen, sie sind jedoch nur durch den Prozess selbst genau berechenbar. Es handelt sich also um einen makro-kontrollierten Prozess, auch wenn dieser selbst keine Geometrie generiert.

Mikro-kontrolliertes Generieren

Als *mikro-kontrollierten Generator* bezeichnen wir einen isomorphen Prozess $P : I \to I$.

Diese Begriff bezeichnet also Prozesse, deren Ausgangsobjekte logisch ihrem Eingang gleichen. Mikro-kontrollierte Generatoren zeichnen sich dadurch aus, dass dem Designer keine nennenswerte Möglichkeit zur Automatisierung zur Verfügung steht: Nach dem Prozess hat er mit den gleichen Objekten zu tun, wie vorher. Mikro-kontrolliertes Generieren ist in hohem Maß auf die Verknüpfung ihrer Prozesse angewiesen: Erst in einem anderen mikro-kontrollierten

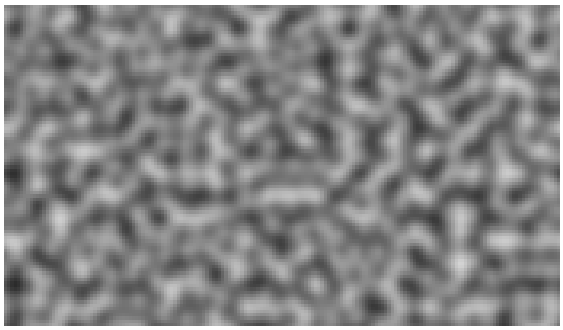


Abbildung 6.2: Perlin-Rauschen mit einer Zellengröße von 32 und dem Seed 1.

Abbildung 6.3: Aus Punktladungen generierte Height-Map

Prozess findet seine Ausgabe als Eingabe auf einer anderen Stufe Verwendung. Beispiele hierfür sind das Zeichnen einer Height-Mapmit Programmen wie Photoshop, die Manipulation von Vertices in Blenders Edit Mode oder das Verschieben, Drehen und Skalieren von Objekten in Unity, die weitgehend ohne Automatisierung auskommen.

Der Übergang zu makro-kontrollierter Generierung findet in der Vereinigung mikro-kontrollierter Eingabe mit der prozeduralen Generierung statt:

Makro-kontrolliertes Generieren

Als *makro-kontrollierten Generator* bezeichnen wir einen prozeduralen Prozess $P : I \to O$, wenn ein zweiter Prozess $P' : I \to O'$ existiert, so dass $O \approx O'$.

Das Potential zu hohem Automatisierungsgrad besteht durch P. Der Designer interagiert mit Parametern, die sich logisch von den Ausgangsobjekten unterscheiden. Auf diese Weise erhält er den Charakter der rein prozeduralen Generierung, erlaubt aber durch die Eingabe I eine einfache Vorhersage und Kontrolle durch P' über seine Ausgabe O'. Die Existenz eines Prozesses $Q : O \to O'$ kann nicht vorausgesetzt werden, da sonst die Komposition $Q \circ P'$ entstünde, die eine logische Verknüpfung zwischen Ein- und Ausgang des Prozesses darstellen würde. Stattdessen deutet der Operator $\approx: A \times B \to \{\top, \bot\}$ eine subjektive Ähnlichkeit der Objekte A und B an. Durch P' und durch die logische Gleichheit von O' und O wird ebenfalls Kontrollierbarkeit gewährleistet.

Solange irgendeine Art der kontrollierten Automatisierung des Entwurfsprozesses denkbar ist, beeinträchtigt der Design-Aspekt die Verfügbarkeit von Mikro-Kontrolle nicht. Dies ermöglicht dem Designer, den Prozess mit der geringsten Anzahl unerwünschter Interaktionen zu wählen.

Bézier-Kurve

Eine Bézier-Kurve besteht aus einer Menge speziell angeordneter Punkte O, die bestimmten Kontinuitätsanforderungen genügt [Kot+19]. Diese Punkte werden aus einer Menge von Kontrollpunkten I durch Kombination der Bernstein-Polynome in einen Prozess P prozedural berechnet.

- O und I unterscheiden sich insofern logisch, als dass alle Punkte in O auf der Kurve liegen, in I aber nur zwei notwendigerweise.
- P' besteht in der Fähigkeit eines Menschen, O durch eine bildliche Darstellung von I grob abzuschätzen.

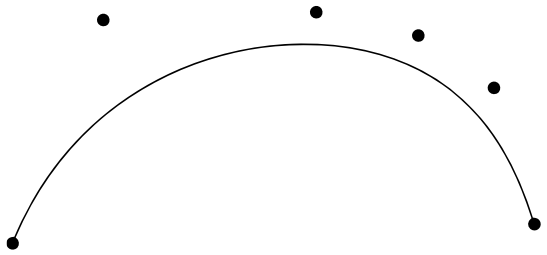


Abbildung 6.4: Bézier-Spline mit Kontrollpunkten

Ein weiteres bekanntes Beispiel ist der Marching Squares-Algorithmus in einigen seiner Ausprägungen, der verschiedene nicht übereinstimmende Typen der Ein- und Ausgabe haben kann.

Eine interessante Überlegung ist, ob das Ergebnis eines makro-kontrollierten Prozesses als mikro-kontrollierbar bezeichnet werden sollte. Dies wirft die Frage nach der Typengleichheit auf. Eine Height Map wie in Abb. 6.2 besteht aus einem Feld von Pixeln. Es könnte nachbearbeitet werden oder mit einem Standard-Bildbearbeitungsprogramm unter Erhaltung der logischen Struktur editiert werden: Seiner Gestalt nach wäre es dann immer noch eine Height-Map, aber nicht einer Art, die dem Ergebnis eines Perlin-Rauschens ähnelt. Im Gegensatz dazu kann eine durch Pixel angenäherte Bézier-Kurve nach der Manipulation ihrer Pixel weiterhin einer Bézier-Kurve, zwei Bézier-Kurven, einer diskontinuierlichen Kurve und einer Nicht-Kurve ähneln, je nach Menge und Position der hinzugefügten oder entfernten Pixel.

6.2 Punktladungen und Vektorfelder als Height-maps

Die Ausgaben eines makro-kontrollierten Prozesses unterscheiden sich logisch von seinen Eingaben. Vorzugsweise ist die Eingabe leichter zu kontrollieren als die Ausgabe, zumindest muss sie aber gleich gut kontrollierbar sein. Um Interaktion zu ermöglichen, muss die Ausgabe eine Art natürliches Ënde"haben, da der Mensch von Natur aus nicht im Unendlichen handeln kann. Das bedeutet ebenfalls, dass jeder Prozess mit einer unendlichen Ausgabemenge zumindest zu einem gewissen Grad rein prozedural erzeugt sein muss.

Für moderne Spiele wollen wir in der Lage sein, dreidimensionale Inhalte zu erzeugen. Beispielsweise werden diskrete Skalarfelder häufig als zweidimensionales Graustufenbild dargestellt, in dem die Helligkeit jedes Pixels einen Höhenwert auf einem Terrain gibt. Abbildung 6.3 zeigt eine solche *Height-Map.* Dies mag für die Darstellung einer voxelbasierten dreidimensionalen Umgebung ausreichen oder durch geometrische Primitiva interpoliert werden, aber in allen anderen Fällen benötigen wir ein skalares Feld, das bei Bedarf in ein Höhenfeld umgewandelt werden kann.

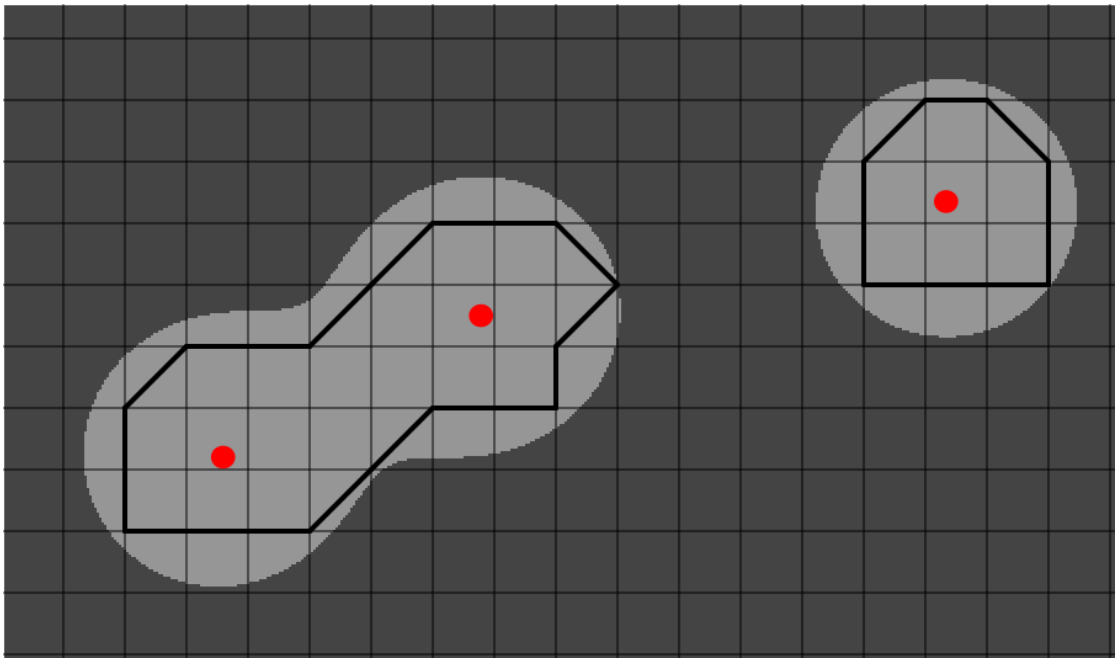


Abbildung 6.5: Ergebnis des Marching Squares-Algorithmus. Die Punktladungen sind rot dargestellt, die Ergebnispolygone in schwarz.

Elektrische Felder, die wir in Abschnitt 2.4 kennen gelernt haben, können die Eingabe von Prozessen sein, die diesen Anforderungen genügen. Sie sind ein schnelles und einfach zu verwendendes Mittel, um Skalarfelder aus Height-Maps auf makro-kontrollierte Weise zu erzeugen. Eine Height-Map drückt die Stärke des elektrischen Feldes durch die Helligkeit der Pixel aus. Isoflächen manifestieren sich in Height-Maps als Bereiche mit gleicher Helligkeit. Wandelt man Height-Maps in eine Isofläche um, können bereits disjunkt. Die Grenze zwischen dem dunklen und dem hellen Bereich der Height-Map in Abbildung 6.5 stellt eine disjunkte zweiwertige Isofläche dar. Zur Berechnung dieses Ergebnisses genügt uns die vereinfachte skalare, zweidimensionale Form der Gleichung (2.5). Für alle Pixel $\mathbf{p}$ einer Height-Map gilt somit:

$$\text{Farbe } (\mathbf{p}) = \begin{cases} \text{hell} & \text{wenn } \sum_i \left| \frac{c_i}{|\mathbf{r}_i - \mathbf{p}|^3} (\mathbf{r}_i - \mathbf{p}_i) \right| > \xi \\ \text{dunkel} & \text{sonst} \end{cases}$$

Obwohl das elektrische Feld dreidimensional sein kann, eignet sich der schrittweise Algorithmus aus Abschnitt 4.4.1 nicht dazu, dreidimensionale Geometrie zu erzeugen. Er liefert ein Polygon, das mit verschiedenen Mitteln in ein dreidimensionales Mesh verwandelt werden kann. Eine Gegenüberstellung dieser Methoden würde den Rahmen dieser Arbeit sprengen, so dass wir uns damit begnügen, das Polygon linear zu extrudieren. Die folgenden Ergebnisse hängen nicht von dieser Wahl ab.

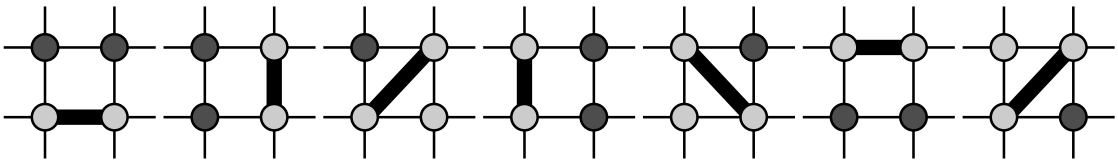


Abbildung 6.6: Illustration der Lookup-Tabelle für den Marching Squares-Algorithmus

6.2.1 Marching Squares

Der Marching Squares-Algorithmus ist wie der ihm verwandte Marching Cubes-Algorithmus (S. Abschnitt 1.1.5) weit verbreitet und wird im Kontext der Spieleentwicklung verwendet. Beide können als Prozess verstanden werden, aber im Gegensatz zum Marching Cubes-Algorithmus liegt bei Marching Squares ein zweidimensionales Raster zugrunde. Die Aufwandsanalyse verläuft analog zur dreidimensionalen Variation und resultiert ebenfalls in einem Aufwand von $\mathcal{O}(n^2)$. Der Marching Cubes-Algorithmus erwartet als Eingabe stets eine Menge von Punktladungen und einen Isowert und liefert einen oder mehrere Polyeder als Ausgabe zurück. Demgegenüber findet sich der Marching Squares-Algorithmus in mehreren Variationen. In vielen Fällen findet das zweidimensionale Analogon zum Marching Squares-Algorithmus Anwendung. Andere Ausprägungen umfassen makro-kontrollierte Prozesse, die ein Höhenfeld - möglicherweise die Ausgabe eines Rauschprozesses - in eine Kontur [Spi14], ein Array von Vektoren oder Arrays von Werten eines anderen Typs [Fli14] umwandeln können. Es gibt weitere Variationen, die unterschiedliche Ergebnisse in unterschiedlichen Typen liefern. In dieser Arbeit betrachten wir den Marching Squares-Algorithmus als eine Möglichkeit, ein Höhenfeld in zweidimensionale Meshes [Won14] umzuwandeln.

Zu diesem Zweck rastert der Algorithmus den gesamten Suchraum in quadratische *Zellen*. Für jede Zelle wird geprüft, ob das Skalarfeld an jeder Ecke des Quadrats über oder unter einem Isowert liegt. Abhängig davon, an welchen Ecken das Feld über oder unter dem Isowert liegt, wird dem Mesh keine Kante oder eine bestimmte Kante gemäß einer Lookup-Tabelle hinzugefügt. In dieser Arbeit wird die in Abb. 6.6 dargestellte Lookup-Tabelle verwendet. Beispiele für weitere mögliche Lookup-Tabellen finden sich in [Wen13, S. 19] Maßnahmen zur Erhöhung der Genauigkeit des Ergebnisses werden für diesen Zweck vernachlässigt, da sie hier nicht relevant sind. Die folgenden Überlegungen hängen nicht von diesen Entscheidungen ab.

Die Verwendung der Ausgabe eines Rauschprozesses als Eingabe für den Marching Squares-Algorithmus bietet eine Möglichkeit, Geometrie auf makro-kontrollierte Weise zu erzeugen: Der kombinierte Prozess hat beide Eingänge des ursprünglichen Prozesses, von denen keiner ein Mesh ist. Die Parameter des Marching Squares-Algorithmus erlauben jedoch einige wenn auch aufgrund der Zufälligkeit des Rauschens vage Vorhersagen und sogar eine gewisse Kontrolle über das resultierende Mesh. Wenn zum Beispiel die Gittergröße mit der Größe der erzeugten

Höhenkarte übereinstimmt oder wenn wir einen Isowert $\xi = 0$ wählen, wissen wir, dass das resultierende Polygon keine Kanten haben wird. Wir wissen auch, dass die Geometrie weder in der Nähe noch in der Ferne von Ladungen entstehen wird, eine direkte Linie zwischen zwei Ladungen, die weit voneinander entfernt sind, wird wahrscheinlich zwei Isoflächen kreuzen, usw.

Die Parameter des kombinierten Prozesses sind:

- Die Position der Punktladungen
- Die Größe des Gitters
- Der Isowert

6.2.2 Schrittweise Suche

Der in Abschnitt 4.4.1 vorgestellte schrittweise Suchalgorithmus bietet eine höhere und bessere Makro-Kontrolle als der Marching Squares-Algorithmus aus Abschnitt 6.2.1, wodurch er sich besser für eine Entwurfsaufgabe eignet, wenn er auf das von Punktladungen erzeugte elektrische Feld angewendet wird. Da das elektrische Feld einer Punktladung an der Position der Ladung unendlich hoch ist und quadratisch abfällt, kann es schwierig sein, einen geeigneten Isowert zu finden. Der Isowert, der zur Erzeugung des Bildes in Abb. 6.5 verwendet wurde, beträgt beispielsweise 0,00015, während alle Punktladungen eine Intensität von 1 haben.

Solange das Skalarfeld stetig ist und konvergiert, können wir sicher sein, dass sich $\mathbf{p}_c$ – gemäß Abschnitt 4.4.1 der in jedem Schritt berechnete Punkt in der Nähe der Fläche – nach einer endlichen Anzahl von Schritten in der Nähe unseres Ausgangspunkts befindet. Wie in Abb. 6.3 zu sehen ist, ist das Bild an der Position der Punktladungen weiß und wird immer dunkler, je weiter wir uns nach außen bewegen. Wir können immer geschlossene Formen auf Pixeln desselben Grautons zeichnen, mit Ausnahme der weißen und schwarzen Pixel. Bei einem konstanten s ist eine sinnvolle Bedingung für die Nähe zweier Punkte, dass ihr Abstand kleiner als s ist. Wenn diese Bedingung erfüllt ist, wird der Algorithmus abgebrochen.

Am Ende jedes Schritts kann der Sammlung eine neue Kante $(p, \mathbf{p}_c)$ hinzugefügt werden; $(\mathbf{p}_c, \mathbf{p}_s)$ im letzten Schritt, wobei p_s der zu Beginn gewählte Startpunkt ist.

Der kombinierte Prozess liefert uns die folgenden Parameter für die Makro-Kontrollierte Generierung:

- Die Position der Punktladungen
- Die Schrittweite s

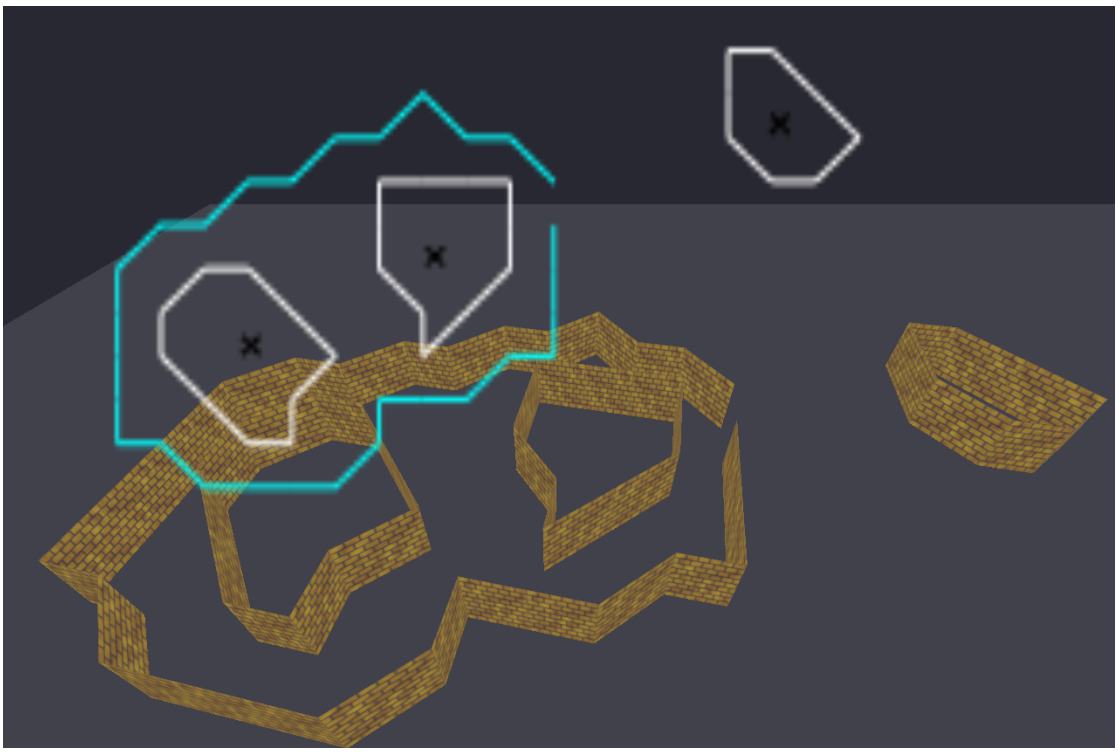


Abbildung 6.7: Vier Polygone, die durch den schrittweisen Suchalgorithmus generiert wurden. Der cyane Linienzug repräsentiert eine Fläche mit deutlich größerem Isowert, als die weißen Linienzüge. Der Hintergrund zeigt die extrudierte Geometrie als Videospiel-Szenerie.

- Die Menge der Flächen
- Der Startpunkt und
- Der Isowert der einzelnen Flächen.

Dieser Prozess kann auf verschiedene Weise erweitert werden, um mehr Freiheitsgrade zu erhalten, die sich in den Eingaben für den Prozess ausdrücken. In der Implementierung dieses Algorithmus, die im Abschnitt Abschnitt 6.2.4 verwendet wird, wurden die resultierenden Punkte beispielsweise gerastert. Damit steht uns ein zusätzlicher Parameter für die Größe des Rasters zur Verfügung, ähnlich wie beim Marching-Squares-Algorithmus. Ein Beispiel für eine Reihe extrudierter Polygone, die durch den schrittweisen Algorithmus erzeugt wurden, ist in Abb. 6.7 dargestellt. Die Position von drei Punktladungen mit gleicher Ladung ist durch ein schwarzes Kreuz gekennzeichnet. Der Algorithmus wurde verwendet, um vier Flächen für verschiedene Isowerte zu erzeugen.

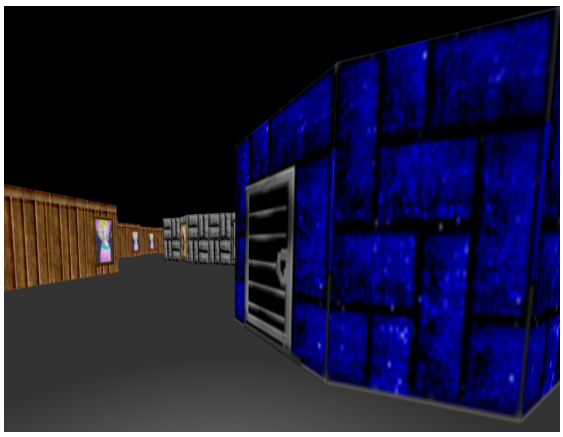


Abbildung 6.8: UV-Gemappte, generierte Geometrie

6.2.3 Auswahl spezieller Anwendungsfälle

Der Ansatz, elektrische Felder, die durch Punktladungen erzeugt werden, mit einem Verfahren zur Umwandlung in Maschen zu kombinieren, birgt durch Variation und Erweiterung eine Vielzahl von Möglichkeiten zur weiteren Verbesserung der Gestaltung von Ebenengeometrien, von denen einige im Folgenden skizziert werden.

6.2.4 Automatisches UV-Mapping

Der in Abschnitt 4.4.1 beschriebene schrittweise Suchalgorithmus ermöglicht die Erstellung von UV-Mapping der Geometrie während des Prozesses. In dem in Abbildung 6.8 gezeigten Beispiel wurde bei jedem Schritt die resultierende Kante zu einem Rechteck extrudiert, um schließlich die Geometrie für einen Dungeon zu erhalten. Für die Demonstration anhand eines einfachen Beispiels wurde ein vordefinierter Teil der UV-Map ausgewählt, indem ihre Koordinaten im Voraus angegeben wurden. Das Ergebnis war ein Dungeon mit wiederholenden Texturen. Wenn eine unterschiedliche Schrittweite benötigt wird, könnte ein Zeiger bei jedem Schritt entsprechend inkrementiert werden. Wenn der Zeiger die Textur überschreitet, kann das Rechteck nach Belieben geteilt oder geschnitten oder die Textur wiederholt werden. Im obigen Beispiel wurden verschiedenen Flächen mit fester Schrittweite eine Textur auf der Grundlage ihrer Reihenfolge zugewiesen. Bei jedem dritten Schritt wurde eine Variation angewendet. Mit solch einfachen Produktionsregeln lässt sich in kurzer Zeit ein beachtliches Ergebnis erzielen. Eine komplexere Abbildungslogik ist möglich, zum Beispiel durch Backtracking.

Der obige Ansatz kann durch eine Variation des Clustering-Algorithmus erweitert werden, um eine komplette, vollständig texturierte Spielkarte unter Zuhilfenahme von Makro-Kontrolle zu erstellen. Der Ansatz stützt sich auf der ursprünglichen Idee der schrittweisen Suche eines zweidimensionalen Polygons auf und erweitert dieses durch Extrusion zu einem dreidimensionalen Objekt, in diesem Fall eine Liste von Quads. Die Meshes werden automatisch beim

Platzieren der Ladungen durch eine Variation des Alg. 13 erstellt, der in abgewandelter Form automatisch korrekte Äquivalenzklassen für Ladungsverteilungen erstellt, in denen die Punktladungen vergleichsweise weit auseinander liegen. Außerdem wird gefordert, dass die Positionen der Punkte nach einer Koordinate – hier der x-Koordinate sortiert sind. Die unterschiedlichen Hierarchieebenen des Konturbaums werden im Editor-Modus durch unterschiedliche Farben der Polygone und im Spielmodus durch unterschiedliche Texturen der Wände angedeutet. Die Hierarchieebene für jedes Polygon wird durch einfaches Zählen der den gefundenen Punkten $\mathbf{p}_c$ nächsten Punktladungen während des Durchführens der schrittweisen Suche ermittelt.

Algorithmus 14 : Heuristik zur Cluster-Suche

eingabe : Menge der Punktladungen $\mathcal{Q}$
ausgabe : Liste Polygone und deren jeweilige Hierarchieebene

$r \leftarrow \varnothing$
wenn $\#\mathcal{Q} < 2$ **dann**
  zurück
Ende
$i = 0$
solange $i < \#\mathcal{Q} - 1$ **tue**
  $(\mathbf{p}_s, q) \leftarrow \mathcal{Q}_i$
  $\mathbf{d} \leftarrow \mathcal{Q}_{i+1} - \mathcal{Q}_i$
  $\mathbf{v}_1 \leftarrow \mathbf{p}_s + \frac{1}{3}\hat{\mathbf{d}}$
  $\mathbf{v}_2 \leftarrow \mathbf{p}_s + \frac{2}{3}\hat{\mathbf{d}}$
  Führe die schrittweise Suche mit Startpunkt $\mathbf{v}_1$ durch.
  $p_1 \leftarrow$ Resultat des Algorithmus.
  Ermittle für jeden Schritt die jeweilige Punktladung, die dem Punkt $\mathbf{p}_c$ am nächsten liegt.
  $C_1 \leftarrow$ Menge dieser Punktladungen.
  Führe die schrittweise Suche mit Startpunkt $\mathbf{v}_2$ durch.
  $p_1 \leftarrow$ Resultat des Algorithmus.
  Ermittle für jeden Schritt die jeweilige Punktladung, die dem Punkt $\mathbf{p}_c$ am nächsten liegt.
  $C_2 \leftarrow$ Menge dieser Punktladungen.
  $r \leftarrow r \cup (p_1, C_1)$
  $r \leftarrow r \cup (p_2, C_2)$
  $i \leftarrow i + 1$
Ende

6.2.5 Logische Bestimmung der Begriffe „innen“ und „außen“

Der schrittweise Suchalgorithmus ermöglicht nicht nur die Erstellung zweidimensionaler Meshes, sondern kann auch dazu verwendet werden, einen vorgegebenen Bereich der Spielszenerie mit Objekten zu füllen, die ungefähr den gleichen Abstand zueinander haben. Wird diese Aufgabe durch einen Zufallsprozess gelöst, können Situationen entstehen, in denen eine homöogene Verteilung der Objekte durch einen Relaxations-Algorithmus nachträglich hergestellt werden

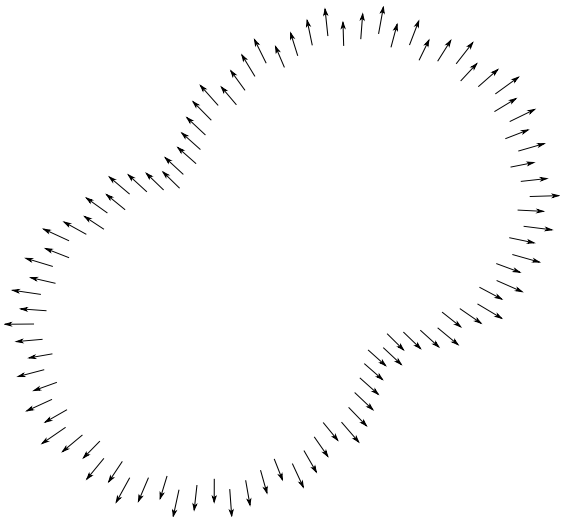


Abbildung 6.9: Pfeile an Punkten auf einer Isofläche, die nach „außen“ zeigen.

muss. Demgegenüber besteht die Möglichkeit, während der Durchführung der schrittweisen Suche bei jedem Schritt ein Spielobjekt am Ort $\mathbf{p}_c$ in der Spielszene zu platzieren. Da die Schrittweite s als Parameter der schrittweisen Suche beliebig gewählt werden kann, ist somit der Abstand der Objekte zueinander mit Ausnahme eines Objektepaars beliebig wählbar.

Oftmals sind diese Objekte im Bezug auf ihre Rotation nicht agnostisch. Bei Häusern in einer Stadt beispielsweise sollen nach Vorgabe die Türen zur Straße und nicht von der Straße weg ausgerichtet sein. Bei rein prozeduraler Generierung kann ein zusätzlicher Aufwand erforderlich sein, um eine “außerhalb”-Richtung zu bestimmen, falls diese verfügbar ist. Elektrische Felder bieten ein sinnvolles Konzept für die Definition der Begriffe „innen“ als „der Ladungsanordnung zugerichtet“ und „außen“ als „von der Ladungsanordnung abgekehrt“ bezogen auf eine Fläche, wobei die Punkte oberhalb des Isowerts das „innen“ versinnbildlichen: Testladungen bewegen sich immer von Ladungen weg, also zeigt auch das Feld an jedem Punkt vektoriell von den Ladungen weg. In Abb. 6.9 ist zu sehen, dass die Pfeile, die die Richtung des elektrischen Feldes andeuten, bei jedem Schritt aus Sicht der Isofläche nach „außen“ zeigen. In unserem Beispiel müssen die Häuser daher so gedreht werden, dass die Tür in Richtung der Pfeile zeigt. Andere Verfahren zur Bestimmung der erforderlichen Richtung für die Oberflächennormale sind denkbar, wie sie aus der 3D-Computergraphik als *Backface Culling* bekannt sind.

Auf ähnliche Weise können Begriffe für „weiter innerhalb“, „in der Mitte“ oder „entlang der Wand“ formuliert werden, um so Bonus-Gegenstände, Feinde oder ähnliche Spielobjekte zu platzieren.

6.2.6 Pfadsuche für KI-Objekte

Die Vektorfeld-Pfadsuche wird häufig in den Bereichen Simulationen, Spiele und Robotik eingesetzt. Sie erleichtert zum Beispiel die schnelle Berechnung von Partikelbewegungen. Typischerweise werden die Vektorfelder für diesen Ansatz aus den Positionen der Kollisionsobjekte und ihrer räumlichen Ausdehnung berechnet [Sid18]. Danach können sich die bewegten Objekte so verhalten, dass sie von den Ladungen „weggestoßen“ werden, indem sie dem elektrischen Feld folgen. Negative Ladungen können eingeführt werden, um diese Objekte stattdessen anzuziehen [Dur13].

Nach den obigen Erkenntnissen ist das Vektorfeld bereits als elektrisches Feld aus der gegebenen Ladungsverteilung berechnet worden und kann ebenfalls für die Pfadsuche verwendet werden, indem die Vertices des Meshes in jedem Durchlauf der Simulation beispielsweise um eine feste Länge in die Richtung des elektrischen Feldes verschoben werden. Dadurch wirkt es, als strebten die so kontrollierten Objekte auf natürliche Weise dem Minimum des Feldes zu. Bei elektrischen Feldern liegt dieses „außen“ – unendlich weit von der Ladungsanordnung entfernt. Es wird also nie stehen bleiben oder mit der generierten Geometrie kollidieren, unabhängig von der Entfernung zur Ladungsanordnung. Die „Vorwärtsrichtung“ ist durch das elektrische Feld an jedem Punkt gegeben. Bei aus gegebener Spielgeometrie erstellten Feldern wird dafür beispielsweise das Raster verfeinert [Sid18].

Ebenfalls ist es möglich, die Stärke der Ladungen zu erhöhen oder das Ziel als Volumen zu formulieren, das Anpassungen zulässt. Hierbei ist zu bemerken, dass dafür die Spielgeometrie nicht erneut erstellt werden muss, aber kann. Auch kann eine negative Ladung eingebracht werden. Dies wird ein künstliches lokales Minimum erzeugen, das als Ziel für die Spielobjekte fungiert, auf das diese sich zubewegen, bis sie es erreicht haben. Die Möglichkeit, unterschiedliche Felder zu verwenden, von denen nicht alle negative Ladungen enthalten, besteht darin, in einer Szene gleichzeitig sich zielgerichtet und nach außen bewegende Objekte zu simulieren. Die Kollisionsfreiheit mit der Spielgeometrie bleibt davon unberührt, allerdings muss die Kollisionserkennung der Spielobjekte untereinander anderweitig sichergestellt werden.

6.2.7 CSG

CSG (Constructive Solid Geometry) ist eine Technik, die ursprünglich aus dem Bereich des mechanischen CAD [Cli21] stammt, aber ähnliche Methoden wurden früher auch für die Definition von Datenstrukturen (BSP-Bäume) von Levels in Spielen wie Doom, Quake oder Half-Life verwendet. Die Grundidee von CSG besteht darin, vordefinierte überlappende Punktmengen, so genannte *Primitiva* (z. B. Kugeln, Quader, Polyeder usw.), mit Hilfe boolescher Operationen zu kombinieren, um gemischte Mengen wie Schnittmenge, Vereinigung und Differenz zu bestimmen. Auf diese Weise ist CSG ein Werkzeug zur intuitiven Modellierung von Geometrien, die mit Hilfe von Raytracing leicht gerendert werden können [Gha08]. Obwohl das am häu-

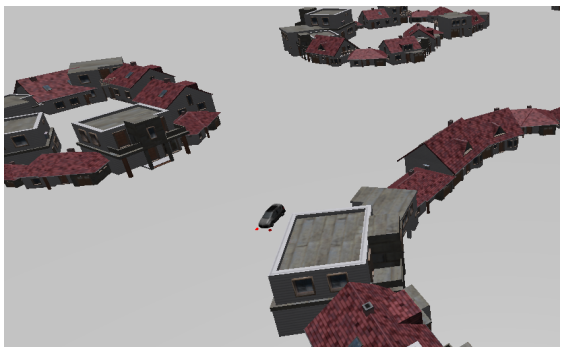


Abbildung 6.10: Hausobjekte an repräsentativen Punkten und KI-Auto

figsten bediente Anwendungsgebiet von CAD-Software das Design von 3D-Modellen durch Konstruktionsbäume ist, müssen die Grundkörper der CSG-Anwendung nicht zwangsläufig dreidimensional sein. Bereits in frühen Veröffentlichungen werden Konstruktionsbäume mit zweidimensionalen Grundkörpern thematisiert [SV93] [Gib+95], und CAD-Anwendungen, die sich auf die Lösung zweidimensionaler Probleme beschränken, sind sowohl in Open Source-Bereich mit LibreCAD und OpenSCAD vorhanden. Auch in proprietären CAD-Programmen wie BricsCAD [Gra15, S. 138ff.] und etablierten Vektorgraphikprogrammen wie Inkscape und Adobe Illustrator ist es möglich, die bei CSG üblichen Operationen – Vereinigung, Differenz und Schnittmenge – auf Pfade und Polygone anzuwenden. Als mögliche zweidimensionale Primitiva benennt [Gib+95] Kreise und Rechtecke, LibreCAD und OpenSCAD verwenden Ellipsen, Ellipsenbögen, Kurven, Polygone und viele ihrer Sonderformen. In Inkscape und Illustrator sind diese und weitere anwendungsspezifische Elemente wie Texte und Kalligraphie gängig.

CSG erfreut sich aufgrund seiner einfachen, intuitiven Verwendbarkeit großer Beliebtheit. Moderne Graphikhardware ist jedoch für das Rendern von Geometrien mit begrenzungsflächenrepräsentierter 3D-Darstellung gemäß Abschnitt 2.3.1 durch Vertices, Flächen usw. optimiert. Daher müssen Konstruktionsbäume für die Rendering-Pipeline umgewandelt werden. Für die Randflächensuche lässt sich 2D-Geometrien die schrittweise Suche verwenden: Abb. 6.11 zeigt zwei sich überschneidende Ellipsen. Für diese wird ein Polygon gesucht, das alle Punkte der blauen Ellipse enthält mit Ausnahme derer, die auch in der roten Ellipse enthalten sind. Ein Startpunkt auf der blauen Ellipse kann beliebig gewählt werden. Bei jedem Schritt wird zusätzlich geprüft, ob $\mathbf{p}_c$ innerhalb der roten Ellipse liegt. Wenn ja, wird der schrittweise Algorithmus im Folgenden nach Punkten auf der Oberfläche der roten Ellipse und innerhalb der blauen Ellipse suchen. Wird ein Punkt außerhalb der blauen Ellipse gefunden, wird der Algorithmus wie zuvor fortgesetzt.

Für Ellipsen mag dieser Algorithmus überflüssig erscheinen. Punkte auf einer Ellipse sind leicht zu finden, und es ist leicht zu prüfen, ob ein Punkt in einer Ellipse enthalten ist. Liegen aber zwei Skalarfelder zugrunde, die innerhalb der Ellipsen einen bestimmten Isowert übersteigen,

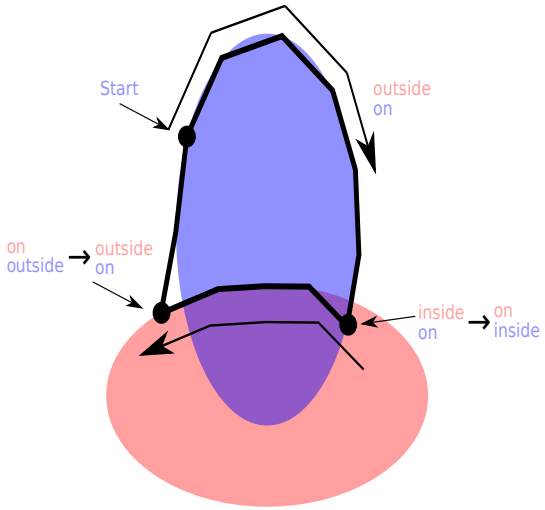


Abbildung 6.11: Berechnung der Punkte der blauen Ellipse ohne die Punkte der roten Ellipse gemäß Abschnitt 4.4.1

können die Erweiterung des schrittweisen Suchalgorithmus angewendet werden, indem ihm diese Isowerte und einen Startpunkt als Parameter übergeben werden, einer oder beide der Isowerte durch den Startpunkt bestimmt werden können. Auf diese Weise kann eine komplexere Geometrie erzeugt werden.

KAPITEL 7

Implementierung

Während der Entstehenszeit dieser Arbeit wurden die Forschungsergebnisse zu den vorliegenden Themen in Implementierungen umgesetzt, um die Gültigkeit der theoretischen Überlegungen für die Praxis zu validieren und deren Funktionsweise verständlich zu machen. Dieses Kapitel stellt die einzelnen Ergebnisse im Hinblick auf Umfang und Funktionsweise vor, begründet Entscheidungen, Annahmen und mögliche Abweichungen von den theoretischen Erläuterungen und gibt Hinweise zur Anwendung und Weiterentwicklung.

7.1 Darstellung von Isoflächen

Zur Validierung der Erkenntnisse aus Kapitel 4 legen wir verschiedene Implementierungen vor, die jeweils besondere Aspekte herauszuheben zum Ziel haben. Aufgrund der unterschiedlichen Anforderungen sind dabei unterschiedliche Frameworks und Programmiersprachen zur Anwendung gekommen, die es nahelegen, die Betrachtung in Realisierung der Implementierung zweidimensionaler und dreidimensionaler Probleme aufzuteilen.

7.1.1 Zweidimensionale Isoflächen

Die Implementierungen der Annäherung zweidimensionaler Isoflächen durch Polygone wurden in Java umgesetzt. Ziel war die Umsetzung der Algorithmen 7 und 8 als Prototypen für Algorithmen zur Annäherung der Fläche in polynomialer Zeit, sowie die Algorithmen 10 aus Abschnitt 4.4.1 und 19 aus Abschnitt 4.5. Für diese Aufgaben wurde das Framework *Processing* samt der gleichnamigen Entwicklungsumgebung verwendet. Abbildung 7.1 gibt eine Übersicht über die für diese Aufgabe erstellten Klassen als UML-Diagramm. Der vorliegende Fall erfordert nur einzelne rudimentäre Auszüge der Vektoralgebra, die der Einfachheit halber in der Klasse `Vec` realisiert wurden. Sie realisiert einfache Vektoraddition, das Skalarprodukt, Invertierung, Normierung und Rotation um ausgewählte feste Winkel, außerdem einige Methoden zum Datenmanagement und Debugging. Die Klasse `Charge` repräsentiert ein Punktladungsobjekt. Sie ist geeignet, eine Funktion zur Berechnung des Gesamtfelds als Teil ihres Parameters in Form

eines iterierbaren Objekts, etwa einer Liste von Punktladungsobjekten, zu übernehmen. Außerdem kann sie durch Angeben eines Mittelpunkts durch die statischen Funktionen `setCenterX` und `setCenterY` und eines Durchmessers durch `setScale` dazu verwendet werden, als ganze Ladungsverteilung zu fungieren. Auf diese Weise kann ein `Charge`-Objekt zum Beispiel als Kugeloberfläche fungieren, für die ein Mittelpunkt, der im Shrink-Wrap-Verfahren in Abschnitt 4.5 zur Konstruktion der Startgeometrien fungiert, manuell angegeben werden kann. Diese Funktionalität findet ihre Anwendung in Spezialfällen, in denen nur eine Äquivalenzklasse der Punktladungen vorliegt. Das Attribut `p` speichert die Ladungsstärke. Schließlich kann die Klasse `Color` eingesetzt werden, um Eigenheiten des repräsentierenden Polygons hervorzuheben, um etwa die einzelnen Flächen der Isofläche zu identifizieren oder die Nähe einer Kante zu einer bestimmten Punktladung anzudeuten. Durch die Funktion `Next` kann durch die Farben eines hinterlegtes Schemas iteriert werden; den Zustand der Iteration speichert die Klasse im Attribut `count`. Durch die Methode `Next` generiert sie ein Farbschema, durch das verschiedene Eigenschaften durch Farben codiert werden können.

7.1.1.1 Schrittweise Suche

Die schrittweise Suche eignet sich lediglich zur Annäherung einer einzigen Fläche durch ein Mesh. Sie kann sich dafür auf die zusätzliche Funktionalität der `Charge`-Klasse stützen. Ein UML-Diagramm dieser und des relevanten Auszugs der `Main`-Klasse finden sich in Abbildung 7.2. Die Maus-Interaktionsschnittstelle von Processing wird dazu genutzt, um vom Benutzer den Startpunkt per Mausklick abzufragen. Der im Folgenden über die Funktion `getDFieldAt` berechnete Isowert wird – zum Vergleich während der Suche auch in den Folgeschritten wieder über diese Funktion berechnet – verwendet, um den Rest des Algorithmus durchzuführen. Zusätzlich zu Alg. 10 wurden farbliche Hervorhebungen der Linien sowie eine Obergrenze der Anzahl der Kanten als zusätzliches Abbruchkriterium hinzugefügt. Das Ergebnis eines Algorithmusdurchlaufs ist beispielhaft in Abbildung 4.2 visualisiert.

7.1.1.2 Shrink-Wrap-Simulation

Während der Arbeit hat sich gezeigt, dass das Erkennen von Fehlannahmen bezüglich der Äquivalenzen und das Reagieren darauf mit geringerer Zeitkomplexität abläuft, als das Erkennen von Fehlannahmen von Nicht-Äquivalenz zweier Punktladungen. Im Bewusstsein, dass diese dennoch auftreten können und in dynamischen Ladungsanordnungen auch auf die Nicht-Äquivalenz erkannt werden müssen, wird auch hier davon ausgegangen, dass es nur eine Äquivalenzklasse gibt. Die entsprechende Main-Klasse gestaltet sich demnach analog zu Abbildung 7.2 aus. Die Annahme, dass nur eine Äquivalenzklasse existiert, erlaubt die Konstruktion der Startgeometrie des Konturbaums $\{q|q \in Q\}'$ mit Q als der Menge der Punktladungen in der Szene, also eines kreisförmigen Polygons, das alle Punktladungen enthält. Somit können während der Simulation mit statischen Anordnungen keine falschen Nicht-Äquivalenzen von

Punktladungen festgestellt werden. In der zweidimensionalen Implementierung wurden daher keine Anstrengungen in diese Richtung unternommen, wodurch sich auch Operationen auf Meshes sowie das Generieren und selektive Ein- und Ausblenden anderer Meshes erübrigen. Das Ergebnis eines Algorithmusdurchlaufs zeigt Abbildung 4.5.

7.1.2 Dreidimensionale VR-Anwendung

Da die schrittweise Suche nur im zweidimensionalen Raum sinnvoll durchführbar ist, liegt für die dreidimensionale Anwendung lediglich die Shrink-Wrap-Simulation als Implementierung vor. Zu Beginn der Forschung erschien die Entwicklung einer browserbasierten Anwendung zielführend, um bereits im frühen Entwicklungsstadium auch für Standalone-Datenbrillen Kompatibilität zu gewährleisten. Zum Abschluss der Arbeit werden bereits eine breite Palette von Standalone-VR-Headsets mit dem Android-Betriebssystem betrieben und unterstützen in den vorinstallierten Browser-Anwendungen WebXR, unter anderem Oculus Quest 1 und 2, HTC Vive Focus, Huawei VR Glasses, Pico 4 und Pico Neo 3. Als geeignete Graphik-Engine, die nativ die Entwicklung für WebXR-Anwendungen ermöglicht, bietet sich three.js an. Die enthaltene `WebXRManager`-Klasse ermöglicht auch das Umschalten der Anwendung in den Präsentationsmodus für Smartphones oder Tablets und unterstützt den Zugriff auf Controller.

Die vorgelegte Implementierung der Shrink-Wrap-Simulation für three.js berücksichtigt das Auftreten disjunkter Isoflächen, nimmt aber keine Operationen auf Meshes vor, sondern reagiert aufgrund der vorteilhaften Zeitkomplexität auf das Erkennen falsch angenommener Äquivalenzen durch die Heuristik des Vergleichs der Richtung der Vektorfelder sowie Nicht-Äquivalenzen durch den Punktbeinhaltungstest über das Skalarprodukt der betroffenen Vertices mit selektivem Einblenden von Meshes. Diese werden durch den Konturbaum vorläufig generiert, aber nur im eingeblendeten Zustand für die Berechnung berücksichtigt. Die Erkennung falsch angenommener Nicht-Äquivalenz zweier Punktladungen wird über die Methode `Icosphere.Match` durchgeführt. Die Startgeometrie wurde aus dem Konturbaum $\{q'|q \in Q\}$ generiert, wobei auf Icospheres als Primitivum zurückgegriffen wird, um von den Möglichkeiten der schnelleren und präziseren Auflösungsanpassung in Abschnitt 3.3.13 profitieren zu können, die ihren Weg in die Implementierung gefunden haben. Außerdem entfällt so die Notwendigkeit der Erkennung falsch angenommener Äquivalenzen für statische Ladungsanordnungen. Eine Konvertierung in Triangle-Strips wurde dabei nicht angestrengt: Laut [GE04] beträgt der Gewinn an Rechengeschwindigkeit lediglich zwei Prozent. Weiterhin würde sie eine Anpassung der Auflösung entweder hinsichtlich der Laufzeit- oder der Speicherkomplexität erschweren. Diese Anpassung ist selektiv über die Funktionen `SubdivideFace`, `SplitFace` `PinchFace` und `TriforceFace` und global über die Funktion `TriforceAll` und `PinchAll` möglich. Diese implementieren die in Abschnitt 3.3.13 vorgestellten umkehrbaren und nicht umkehrbaren Aufteilungen von dreieckigen Faces in Icospheres.

Für die Benutzerinteraktion orientiert sich die Umsetzung an [Wie18], die sich des HTML-basierten 3D-Frameworks A-FRAME bedient, das seinerseits funktional eine Obermenge von three.js darstellt und auf einer dreidimensionalen Navigation mittels Gamepad aufbaut. Damit ist dieser Ansatz für diese Arbeit sowohl in der virtuellen Realität durch eine Datenbrille als auch für die Bildschirmarbeit geeignet. Das Gamepad steuert ein eigens kreiertes Controller-Skript `cursor-update`, welches eine First Person-Steuerung realisiert. Die WebXR-Funktionalität wird von jedem A-FRAME-Projekt originär unterstützt. Die Implementierung im Anhang hingegen stützt sich auf die three.js-eigene `OrbitControls`, die eine Navigation im dreidimensionalen Raum ähnlich in gängigen 3D-Graphikanwendungen wie Autodesk Maya oder Blender realisieren. Der Wechsel in den VR- oder Präsentationsmodus funktioniert hinsichtlich der Benutzerinteraktion wie bei [Wie18], ist allerdings in den neueren Versionen von three.js Teil des Renderers der Graphik-Engine selbst und wird durch die Komponente `VRButton` realisiert. Auf diese Weise unterstützt sie die in handelsüblichen eigenständigen VR-Datenbrillen mittlerweile standardmäßig beiliegenden VR-Controller sowie eine Erweiterung der Funktionalität um Kontrolle des Programms durch erkennen von Handgesten mittels Computer Vision wie in [GMR20] oder in Anlehnung an [Mat+13] optisches Tracking oder andere Tracking-Methoden [Kat17, passim].

Die Implementierung orientiert sich generell an den Prinzipien der objektorientierten Programmierung, macht sich aber in den Hauptroutinen die Möglichkeit von JavaScript zur imperativen Programmierung zunutze. Die grundlegenden Funktionen zur Berechnung des elektrischen Feldes wurden aus den weiter oben besprochenen Implementierungen für zweidimensionale Probleme übernommen. Abbildung 7.3 zeigt ein UML-Klassendiagramm der wichtigsten erstellten und verwendeten Klassen.

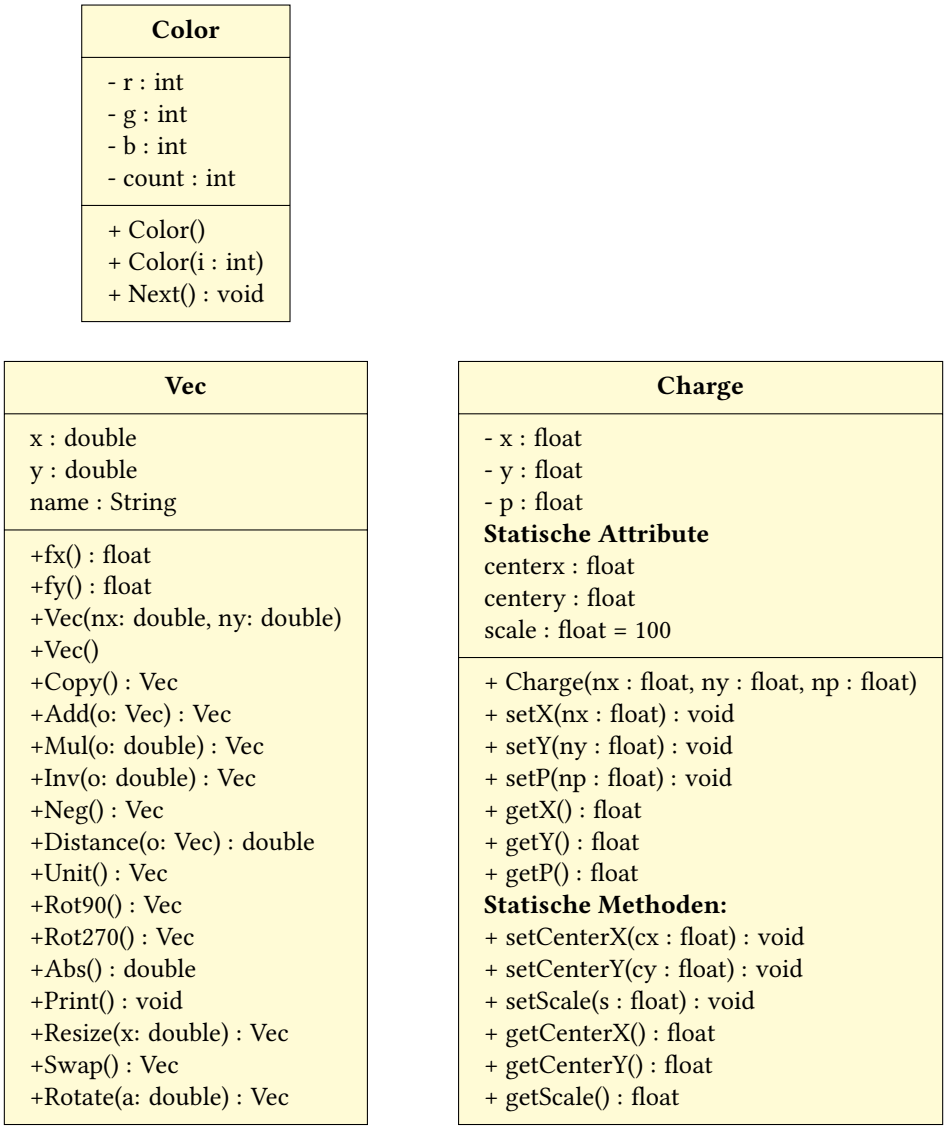


Abbildung 7.1: UML-Klassendiagramme der Hilfsklassen für Farbe, Vektoralgebra und der Verwaltung für Ladungsobjekte

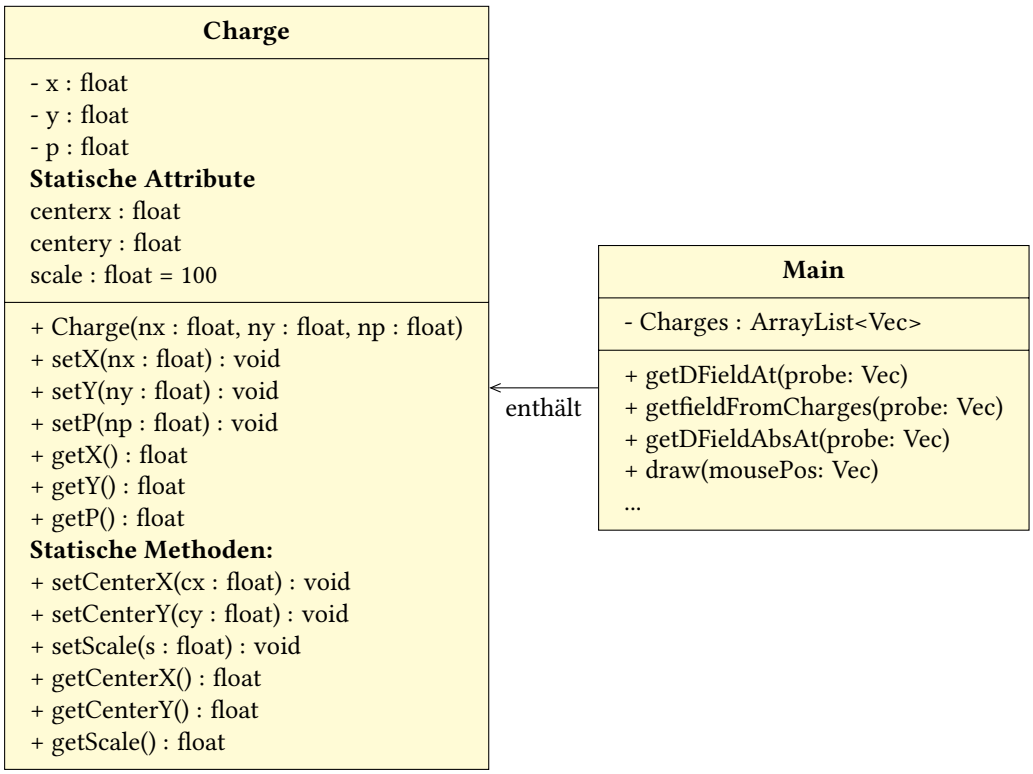


Abbildung 7.2: Hauptklasse der schrittweisen Suche

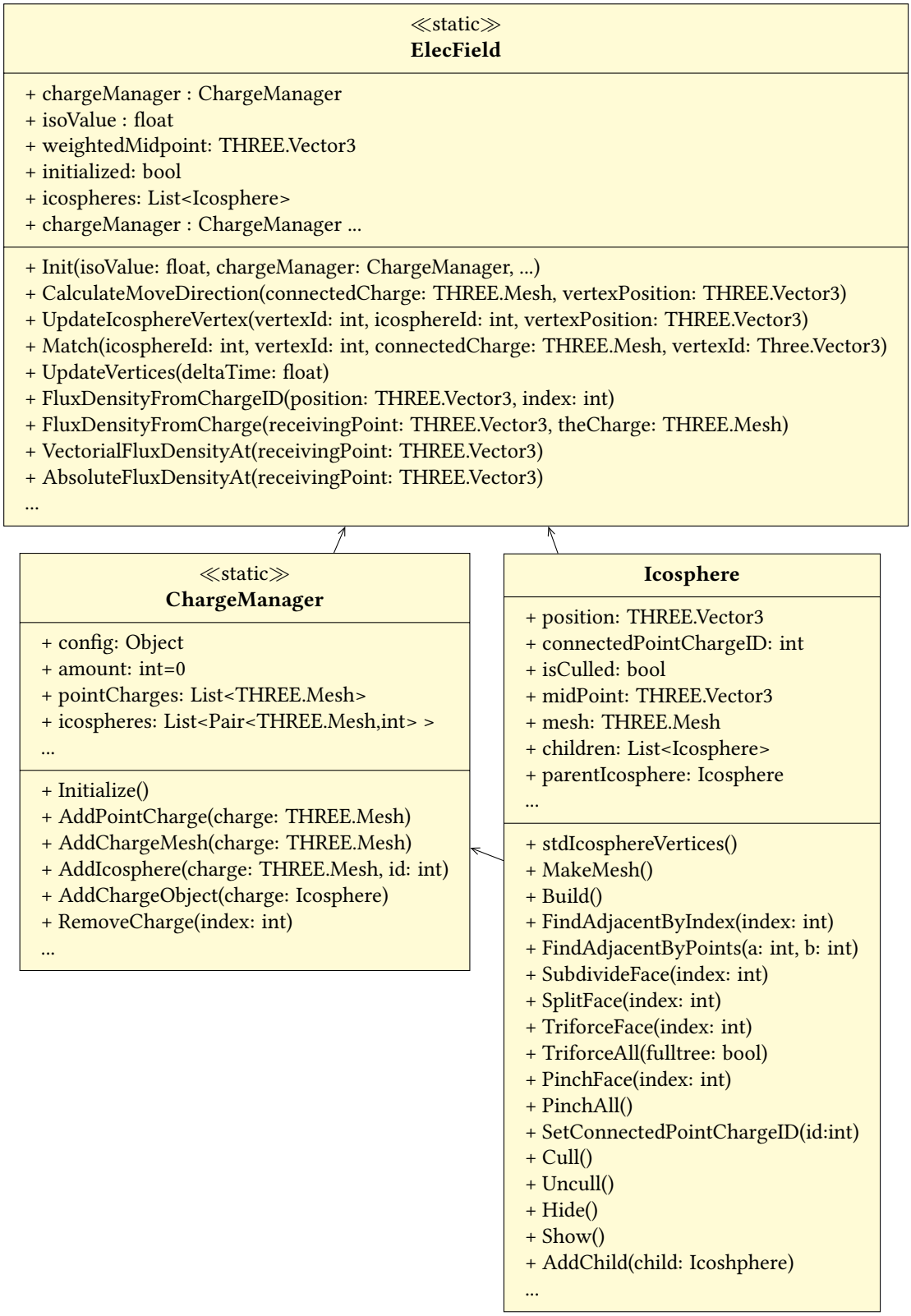


Abbildung 7.3: Wichtige Klassen der Implementierung der Shrink-Wrap-Simulation für dreidimensionale Probleme

7.2 Schrittweise-Suche für makro-kontrollierte Generierung für Spielgeometrie

Die Anwendung des Annäherns von Isoflächen durch Meshes für die Spieleentwicklung in Kapitel 6 hatte zum Ziel, einen Sonderfall für die Einsetzbarkeit der schrittweisen Suche herauszuheben. Durch sie ist es möglich, dreidimensionale Geometrie zu generieren, indem die durch sie generierte zweidimensionale Geometrie beispielsweise durch Extrusion um eine zusätzliche Dimension erweitert wird. Die beiden vorgelegten Implementierungen verwenden mit `fieldcalculator2D` eine zweidimensionale Variante der `ElecField`-Klasse aus der Implementierung für dreidimensionale Probleme. Die Implementierung erfolgte wieder in JavaScript und verwendet die Graphik-Engine three.js, unter Wiederverwendung der Klassen aus den voranstehenden Abschnitten. Da beide Implementierungen grundsätzlich die Umsetzung der makro-kontrollierten Erstellung von Spielgeometrie durch einen Level-Editor mit einem Spielmodus umsetzen, bieten sie eine Möglichkeit, den Modus über die Taste M zu wechseln. Der Editor-Modus zeigt eine Übersichtskarte der Spielgeometrie an und ermöglicht die Bearbeitung durch Hinzufügen von Punktladungen durch die Taste A und deren Verschiebung mit der Taste G. Die Ladungsverwaltung übernimmt hier die Klasse `Charges`, die ansonsten den für diese Anwendung relevanten Funktionsumfang der `ChargeManager`-Klasse leistet. Die schrittweise Suche wird in der Klasse `Extruder` durchgeführt, die ebenfalls in einer eigenen Methode die erzeugte Geometrie nachbearbeitet, extrudiert, texturiert und die notwendigen Objekte für den Szenegraphen von three.js erstellt. Das zweidimensionale Ergebnis des Algorithmus, das im Level-Editor angezeigt wird, wird von der Klasse `Minimap` beispielsweise durch Hervorhebungen und Einfärbung aufgearbeitet und verwaltet.

7.2.1 Stadt

Die erste Implementierung erzeugt durch die schrittweise Suche eine makro-kontrollierte Generierung einer Stadt, die aus Häuserblocks besteht. Die Klasse `Extruder` wurde dafür um eine Schnittstelle zu `THREE.OBJloader` erweitert, die eine vordefinierte Liste von Dateinamen über einen HTTP-Request lädt. Statt eines Polygons erstellt die Klasse an jedem gefundenen Punkt $\mathbf{p}_c$ ein Hausobjekt und dreht dieses parallel zur Richtung des an $\mathbf{p}_c$ gemessenen elektrischen Feldes. Außerdem wird das erzeugte Objekt mit einer in einer weiteren Liste von Dateinamen festgelegten Textur versehen. Der Algorithmus selbst erfordert einen Startpunkt für die Suche, der durch einen Mausklick festgelegt werden kann. Somit ist es möglich, beliebige durch Häuser repräsentierte Flächen mit unterschiedlichen Isowerten zu erstellen. Die Taste U löscht den zuletzt erstellten Häuserblock. Im Editor-Modus kann außerdem durch Drücken der Taste C ein Auto an die Position der Maus verschoben werden, das die in Abschnitt 6.2.6 beschriebene Pfadsuche realisiert.

7.2.2 Dungeon und Heuristik zur Generierung der Äquivalenzklassen

Ein weiteres three.js-Projekt implementiert die Erstellung einer vollständig texturierten Spielkarte aus Abschnitt 6.2.4. Es erstellt gemäß der ursprünglichen Idee der schrittweisen Suche ein zweidimensionales Polygon und erweitert dieses durch Extrusion zu einem dreidimensionalen Objekt. Die Meshes werden automatisch beim Platzieren der Ladungen durch eine Variation des Alg. 13 erstellt, der in seiner abgewandelten Form automatisch korrekte Äquivalenzklassen für makroskopische Ladungsverteilungen erstellt. Davon kann in diesem Fall ausgegangen werden, weil die nach Konstruktion des Algorithmus die Punktladungsobjekte dreimal weiter auseinander liegen, als die Grenzen der Isofläche. Der Algorithmus erfordert, dass die Positionen der Punkte nach einer Koordinate – beispielsweise nach der x-Koordinate sortiert sind. Die Klasse `Profile` tut dies durch JavaScript-eigene Mechanismen.

Die Variation des Algorithmus besteht darin, dass er die gesamte Menge der Flächen erstellt, die im Konturbaum dieser Ladungsanordnung auftreten würden, unabhängig vom Isowert. Ein Konturbaum ist daher auch nicht Teil der Implementierung. Die unterschiedlichen Hierarchieebenen des Konturbaums werden im Editor-Modus durch unterschiedliche Farben der Polygone angedeutet und verkörpern die unterschiedlichen Texturen der Wände im Spielmodus. Die Hierarchieebene jedes Polygons wird dabei durch einfaches Zählen der den gefundenen Punkten $\mathbf{p}_c$ nächsten Punktladungen während des Durchführens der schrittweisen Suche ermittelt.

KAPITEL 8

Ergebnisse dieser Arbeit

Im Rahmen dieser Arbeit wurden verschiedene Verfahren zur effizienten Annäherung von Isoflächen durch Meshes entwickelt und vorgestellt. Das **schrittweise Suche**-Verfahren wurde entwickelt, um ein bis auf eine Kante homogenes Polygon zu erstellen und dieses durch Extrusion als eine Menge von Flächen oder eine entlang der angenäherten Fläche angeordnete Objekte in einer Szene zu platzieren. Die Stärke der schrittweisen Suche liegt in ihrer niedrigen asymptotischen Laufzeit. Allerdings ist es auf zweidimensionale Anwendungen beschränkt und kann nur eine Fläche durch ein Polygon annähern.

Das **Shrink-Wrap-Verfahren** wurde als ursprüngliches Remeshing-Verfahren zum Zweck der Annäherung einer Isofläche statt eines Basis-Meshes durch Verschieben von Vertices auf einer geradlinigen Trajektorie modifiziert. Die Auflösung ist beliebig kontrollierbar, auch lokal, und es können kleine Veränderungen in der dynamischen Ladungsanordnung berücksichtigt werden. Die Stärke des Shrink-Wrap-Verfahrens liegt in seiner niedrigeren durchschnittlichen Laufzeit im Vergleich zur Marching Cubes-Methode und in der Möglichkeit, im Gegensatz zu diesem saubere Meshes zu erzeugen. Der Nachteil dieses Vorgehens liegt darin, dass in allgemeinen Fällen eine Aufstellung der Äquivalenzklassen erforderlich ist. Dazu muss ein Verfahren ausgewählt werden, das innerhalb einer Simulation so eingesetzt werden kann, dass es innerhalb der Komplexitätsklasse des Shrink-Wrap-Verfahrens während eines Simulationsschritts liegt.

Einer für das Shrink-Wrap-Verfahren verwendbaren Startgeometrie, deren Verwendung sowohl in Äquivalenz als auch Nicht-Äquivalenzen von Punktladungen resultieren kann, wurde einer in dieser Arbeit vorgestellte Methode zum **Zusammenfügen zweier Meshes** mit jeweils einem Öffnungspolygon, die beide einander zugerichtet sind, entgegengestellt. Dies ermöglicht eine Reaktion auf das Erkennen der benannten Falschannahmen unter einer geringer asymptotischer Speicherkomplexität. Der Zeitpunkt, zu dem diese Korrektur zu erfolgen hat, ist allerdings schlecht abschätzbar, weswegen das gleiche Verfahren im Nachgang wiederholt durchgeführt werden muss. Dies führt zu einer Erhöhung der Laufzeit.

Als Alternative zum Zusammenfügen zweier geöffneter Meshes wurde ebenfalls das **selektive Ein- und Ausblenden von Meshes** vorgestellt. Um die korrekte Entscheidung, welche Meshes- ein- und ausgeblendet werden müssen, treffen zu können, wurde das Konzept des Konturbaums durch eine zusätzliche **Markierung** erweitert. Dies ermöglichte Reaktionen auf Fehlannahmen in linearer Zeit während der Simulation. Die zeitaufwendigen Berechnungen sind in der Initialisierungsphase außerhalb der Simulation durchführbar. Dies hat einen starken Anstieg der Speicherkomplexität zur Folge. Ein Beweis, dass die Generierung beispielsweise durch lazy evaluation „on the fly“ erfolgen kann und somit sogar die initiale Berechnung entfallen könnte, ist noch nicht gelungen.

Eine **invertierbare lokale Subdivision** mit konstanter Speicherkomplexität und Laufzeit wurde ebenfalls vorgestellt. Durch sie ist eine Reaktion auf Inhomogenität in einem Mesh möglich, das durch lineares Verschieben der Vertices auftreten kann.

Ein gegen **kollineare Fälle stabiler Punkt-Polygon-Beinhaltungstest** wurde entwickelt, der ohne Erhöhung der Laufzeitkomplexität läuft. Dieser kann in zweidimensionalen Shrink-Wrap-Simulationen Verwendung finden. Für dreidimensionale Anwendungen muss allerdings auf einen Polyeder-Beinhaltungstest zurückgegriffen werden, für den noch kein Verfahren gefunden ist, das mit linearer Laufzeit abgeschlossen werden kann.

Schließlich beinhaltet die Arbeit **Heuristiken zur Abschätzung der Cluster**, die geeignet sind, die Äquivalenzklassen für makroskopische Probleme zu berechnen bzw. diese für alle anderen abzuschätzen. Sie können genutzt werden, um in diesen Fällen eine Startgeometrie zu berechnen, in denen keine bzw. sehr wenige Fehlannahmen über die Äquivalenzverhältnisse der Punktladungen zu erwarten sind. In nicht makroskopischen Fällen ist es auf Grundlage dieser Abschätzung dennoch notwendig, die Erkennung auf Fehlannahmen durchzuführen, wobei der Aufwand der Behebung ihrer Folgen seltener aufgebracht werden muss.

Verschiedene Verfahren zur **Erkennung von falschen Annahmen bezüglich Nicht-Äquivalenz und Äquivalenz** wurden vorgestellt. Besonders hervorzuheben ist die schnelle Erkennung falsch angenommener Äquivalenz, die zumindest während der Durchführung des Shrink-Wrap-Verfahrens in konstanter Zeit ausgeführt werden kann. Eine Schwäche besteht im erforderlichen zusätzlichen Zeitaufwand bei der Aufteilung der neuen Äquivalenz. Schließlich wurde im Kontext der Makro-kontrollierten Erstellung von Geometrie für Videospiele die Möglichkeit zur Verwendung der schrittweisen Suche erkannt. Durch die Anordnung der Punktladung und mehrfaches Wählen von Startpunkten für die schrittweise Suche durch Makro-Kontrolle kann der Design-Prozess der Spielgeometrie durch prozedurale Generierung erleichtert werden, während die Ergebnisse kontrollierbar bleiben. In der makro-kontrollierten Erstellung von Spielgeometrie wurde der Sonderfall makroskopischer Probleme erkannt, weswegen hier die Heuristik zum Clustering zur Anwendung kommen kann.

KAPITEL 9

Fazit und Ausblick

Methoden herauszuarbeiten, durch die Isoflächen mit niedrigerer Laufzeit bei qualitativ höherwertigen Ergebnissen als zum derzeitigen Stand der Forschung durch Meshes angenähert werden können, war das Ziel dieser Arbeit.

Kapitel 1 ordnet die Arbeit in den derzeitigen Forschungsstand ein und analysiert verschiedene bestehende Verfahren und Softwarelösungen, die dieses Problem bereits lösen. Als bekanntester Vertreter wird der Marching Cubes-Algorithmus herausgearbeitet.

Kapitel 2 lieferte die notwendige Theorie der Skalar- und Vektorfelder und wichtige Konzepte der Informatik sowie Computergraphik.

In Kapitel 3 wurden Voraussetzungen für Meshes dargelegt, die Isoflächen repräsentieren sollen und als Grundlage für neue Verfahren lineare Trajektorien für Vertices vorgestellt. Die Versuche in diesem Kapitel, die Positionen dieser Vertices algebraisch oder durch einen Greedy-Algorithmus zu bestimmen schlugen fehl: Die Berechnung der Vertex-Positionen als Schnittpunkt der Trajektorie mit einer Fläche konnte ab einer bestimmten Anzahl von Punktladungen aufgrund der Höhe des Grades des Polynoms nicht fortgesetzt werden. Der Iterationsschritt eines Greedy-Algorithmus bot keinen mathematischen Lösungsraum für eine Vertex-Position, die nach Anwenden des Superpositionsgesetzes bestimmt werden könnte. Außerdem wird dargelegt, dass eine gefundene Startgeometrie die disjunkten Teile der Isofläche möglicherweise nicht korrekt abbildet. Es wurden Methoden besprochen, diese Startgeometrie so zu verändern, so dass diese im folgenden Verlauf der Simulation die Charakteristik der Isofläche besser abbildet. Zur Bewertung dieser Abbildung wurde der markierte Konturbaum eingeführt. Außerdem wurden verschiedene Maßnahmen zur Erkennung und Behebung falsch angenommener Äquivalenzverhältnisse von Punktladungen gezeigt.

In Kapitel 4 wurden die schrittweise Suche und die Shrink-Wrap-Simulation eingeführt, und mit dem Marching Cubes-Algorithmus hinsichtlich der Zeitkomplexität verglichen. Auch

wenn das Verfahren auf Spezialfälle reduziert bleibt, unterschreitet die schrittweise Suche die asymptotische Laufzeit des Marching Cubes-Algorithmus. Das Shrink-Wrap-Verfahren kann während der Simulation durch eine Graphik-Engine ausgeführt werden, wobei nachgewiesen werden konnte, dass dieses dabei die gleiche asymptotische Laufzeit wie bei einem dauerhaft durchgeführten Marching Cubes-Algorithmus aufweist. Für beide Verfahren hat sich herausgestellt, dass ihre Ergebnisse die Anforderungen an ein sauberes Mesh erfüllen, im Gegensatz zum Marching Cubes-Algorithmus, der dafür Nachbearbeitungsschritte benötigt. Außerdem konnte das Ergebnis-Mesh des Marching Cubes-Algorithmus die Isofläche durch seine Vertices nur interpoliert wiedergeben, während die beiden anderen Verfahren sich der Fläche sukzessive annähern und mit längerer Laufzeit immer genauere Ergebnisse liefern. In Kapitel 5 wird die Parallelisierbarkeit der vorgestellten neuen Verfahren hergeleitet.

In Kapitel 6 wurde das Verfahren der schrittweisen Suche verwendet, um eine makrokontrollierte Generierung von Spielgeometrie zu ermöglichen. Es konnte gezeigt werden, dass durch Extrusion der Ergebnisgeometrie mit wenig Aufwand komplexe Spiellandschaften erstellt werden können. Die Einfachheit dieses Konzepts ermöglichte die Erweiterung durch automatische Texturierung und die Ableitung einer für die Spieleentwicklung wichtigen logischen Konzeption der Begriffe „innen“ und „außen“, eines automatischen Pfadsuche- und Kollisionsvermeidungsverfahrens sowie weiterer Anwendung auf Konstruktionsbäume in der konstruktiven Festkörpergeometrie.

In Kapitel 7 wurden wichtige Elemente der Umsetzung der in den vorangegangenen Kapiteln entwickelten Verfahren durch Implementierung validiert. Die in diesem Rahmen erstellten Klassen wurden durch UML-Diagramme wiedergegeben und die Funktionalitäten der einzelnen Attribute und Methoden, soweit nicht selbstverständlich, erläutert.

Zusammenfassend konnte in dieser Arbeit aufgezeigt werden, dass der Einsatz von Isopotentialflächen in der virtuellen Realität sinnvoll ist und dass es möglich ist, ihre Annäherung durch Flächen schneller und mit besseren Ergebnissen als durch den Marching Cubes-Algorithmus zu generieren. Wir haben auch gezeigt, dass der in Kapitel 4 vorgestellte Algorithmus der schrittweisen Suche im Bereich der Videospielentwicklung nutzbringend ist. Unsere Arbeit bietet somit ebenfalls ein Werkzeug zur Generierung von Spielgeometrie sowie zur Nutzung elektrischer Felder im Bereich der Spieleentwicklung.

9.1 Ausblick

Einige der in dieser Arbeit entwickelten Verfahren wie das Generieren eines Konturbaums oder die der Startgeometrie in der Initialisierungsphase laufen auf einen Time-Memory-Tradeoff hinaus. Weiterführende Forschung könnte zutage fördern, dass eine Reduktion der Speicherkomplexität möglich ist. Besonders im Hinblick auf die Subdivision der Icospheres kann eine

Untersuchung, ob die Eigenschaften eines Goldberg-Polyeders für einen Zugewinn an Geschwindigkeit nutzbar gemacht werden können. Obwohl in den Grundlagenkapiteln weitere prominente Arten der Ladungsverteilungen vorgestellt wurden, die entweder auf Punktladungsanordnungen zurückgeführt oder durch Integration über Kurven oder Hüllflächen mit den dargestellten Methoden durch Meshes angenähert werden können, sind auch andere Arten von Feldern denkbar, die keine Reminiszenz an elektrische Felder erkennen lassen. Als einfachstes Beispiel könnte statt der euklidischen die Manhatten-Distanz in der Berechnung des Feldes zugrunde gelegt werden. Dies könnte die runden Formen der makro-kontrolliert generierten Spielewelten auflösen und in einem eher eckigen Erscheinungsbild der Wände resultieren. Weiterhin kann der Einsicht, dass zwei orthogonale, zweidimensionale Felder isomorph zu einem dreidimensionalen Feld sind, dazu verholfen werden, die Nutzbarkeit des Verfahrens der schrittweisen Suche auf dreidimensionale Felder auszuweiten.

Das Forschungsprojekt hat neue Wege aufgezeigt, Isoflächen durch Meshes anzunähern, die weiter verfolgt werden sollten, um die Nutzbarkeit der Ansätze in Anwendungen der virtuellen und erweiterten Realität zu verbessern.

Anhang

Betreute Thesen

[Kat17]
Nadezda Katraeva
Erfassung von Kopfbewegungen für Augmented- und Virtual Reality-Anwendungen
2017

[Bel21]
Timon Belau
Gegenüberstellung von Strahlensuchverfahren für die Darstellung von Äquipotentialflächen
2021

[Wie18]
Jan-Philipp Wiese
Eingabe geometrischer Transformationsdaten in virtueller Realität
2018

ANHANG

Literatur

[Ada19] S. Adams. *Foundations of Physics*. Mercury Learning und Information, 2019. ISBN: 9781683921455. URL: https://books.google.de/books?id=KieUDwAAQBAJ (siehe S. 8).

[Aff15] R. Affeldt. *Proving Properties on Programs*. https://coq.inria.fr/files/coq-itp-2015/course-5.pdf. [Accessed: July 12, 2023]. 2015 (siehe S. 103).

[Ale01] M. Alexa. „Mesh Morphing“. In: *Eurographics 2001-STAR-State of The Art Report*. Citeseer. 2001 (siehe S. 112).

[And14] C. Andújar. *Marching cubes algorithm*. Presentation. 2014 (siehe S. 105).

[AK19] D. Ashlock und S. Krantz. *Fast Start Differential Calculus*. Synthesis Lectures on Mathematics and Statistics. Morgan & Claypool Publishers, 2019. ISBN: 9781681736426 (siehe S. 31).

[ABL03] D. Attali, J.-D. Boissonnat und A. Lieutier. „Complexity of the Delaunay Triangulation of Points on Surfaces the Smooth Case“. In: *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*. SCG '03. San Diego, California, USA: Association for Computing Machinery, 2003, 201–210. ISBN: 1581136633. DOI: 10.1145/777792.777823. URL: https://doi.org/10.1145/777792.777823 (siehe S. 75).

[Bab16] J. Babington. *Basic Electromagnetic Theory*. Essentials of Physics Series. Mercury Learning & Information, 2016. ISBN: 9781944534400. URL: http://merclearning.com/titles/Basic_Electromagnetic_Theory.html (siehe S. 8, 29–32, 36).

[Ban20] M. Bancila. *Modern C++ Programming Cookbook: Master C++ core language and standard library features, with over 100 recipes, updated to C++20, 2nd Edition*. Packt Publishing, 2020. ISBN: 9781800206205. URL: https://www.packtpub.com/product/modern-c-programming-cookbook-second-edition/9781800208988 (siehe S. 125).

[Bec07] B. Beckert. *Theoretische Informatik*. https://formal.kastel.kit.edu/~beckert/teaching/TheoretischeInformatik-SS07/180407b.pdf. Abgerufen am 15. May, 2023. 2007 (siehe S. 126).

[Bel21] T. Belau. *Gegenüberstellung von Strahlensuchverfahren für die Darstellung von Äquipotentialflächen*. Wuppertal: Bachelor-Thesis an der Uni-Wuppertal, 2021 (siehe S. 14, 163).

[Bir86] R. S. Bird. *AN INTRODUCTION TO THE THEORY OF LISTS*. Techn. Ber. PRG-56. 8-11 Keble Road, Oxford OX1 3QD, England: Oxford University Computing Laboratory, Programming Research Group, 1986 (siehe S. 49, 51, 67, 104).

[Bli82] J. Blinn. „A Generalization of Algebraic Surface Drawing“. In: Association for Computing Machinery, Inc., 1982 (siehe S. 11, 14, 29, 34).

[Bli91] J. Blitz. *Electrical and Magnetic Methods of Nondestructive Testing*. Taylor & Francis, 1991. ISBN: 9780750301480 (siehe S. 30).

[Bri08] R. Bridson. *Fluid Simulation*. USA: A. K. Peters, Ltd., 2008. ISBN: 1568813260 (siehe S. 27).

[BW01] K. Brodlie und J. Wood. „Recent Advances in Volume Visualization". In: *Computer Graphics Forum* 20.2 (2001). ISSN 1067-7055, S. 125–148 (siehe S. 11, 16).

[Bun03] P. Bundschuh. *Einführung in die Zahlentheorie*. German. Springer, 2003. ISBN: 978-3540435792 (siehe S. 102).

[Bun94] A. Bunkenburg. „The Boom Hierarchy". In: *Functional Programming, Glasgow 1993: Proceedings of the 1993 Glasgow Workshop on Functional Programming, Ayr, Scotland, 5–7 July 1993*. Hrsg. von J. T. O'Donnell und K. Hammond. London: Springer London, 1994, S. 1–8. ISBN: 978-1-4471-3236-3. DOI: 10.1007/978-1-4471-3236-3_1. URL: https://doi.org/10.1007/978-1-4471-3236-3_1 (siehe S. 49).

[But11] R. Butt. *Applied Linear Algebra and Optimization Using MATLAB*. English. eBook. Mercury Learning und Information, 2011, S. 800. ISBN: 9781936420766. URL: https://doi.org/10.1515/9781936420766 (siehe S. 94).

[Cav66] P. Cavanagh. „Method for Displaying Lines of Force in an Electrostatic Field". In: *American Journal of Physics* 34.11 (1966), S. 1034–1036. DOI: 10.1119/1.1972433. URL: https://doi.org/10.1119/1.1972433 (siehe S. 8, 10).

[CX14] A. N. Chernikov und J. Xu. „A Computer-Assisted Proof of Correctness of a Marching Cubes Algorithm". In: *Proceedings of the 22nd International Meshing Roundtable*. Hrsg. von J. Sarrate und M. Staten. Cham: Springer International Publishing, 2014, S. 505–523. ISBN: 978-3-319-02335-9 (siehe S. 103, 104).

[CB14] P. Chiusano und R. Bjarnason. *Functional Programming in Scala*. Manning Publications, 2014. ISBN: 9781617290657. URL: https://www.manning.com/books/functional-programming-in-scala (siehe S. 26, 126).

[Cig+97] P. Cignoni, P. Marino, C. Montani, E. Puppo und R. Scopigno. „Speeding up isosurface extraction using interval trees". In: *IEEE Transactions on Visualization and Computer Graphics* 3.2 (1997), S. 158–170. DOI: 10.1109/2945.597798 (siehe S. 15).

[CTW88] K. L. Clarkson, R. E. Tarjan und C. J. van Wyk. „A Fast Las Vegas Algorithm for Triangulating a Simple Polygon". In: *Proceedings of the Fourth Annual Symposium on Computational Geometry*. SCG '88. Urbana-Champaign, Illinois, USA: Association for Computing Machinery, 1988, 18–22. ISBN: 0897912705. DOI: 10.1145/73393.73396. URL: https://doi.org/10.1145/73393.73396 (siehe S. 80).

[Cla18] J. Claycomb. *Mathematical Methods for Physics: Using MATLAB & Maple*. Mercury Learning & Information, 2018. ISBN: 9781523120314. URL: https://www.merclearning.com/titles/Mathematical_Methods_for_Physics.html (siehe S. 18).

[Cli21] L. S. Cline. *Fusion 360 for Makers*. Make Community, LLC, 2021. ISBN: 9781680456486. URL: https://books.google.de/books?id=r9hyEAAAQBAJ (siehe S. 143).

[CLR09] T. H. Cormen, C. E. Leiserson und R. L. Rivest. *Introduction to Algorithms*. 3rd. Example 22.1-3, p. 530. MIT Press und McGraw-Hill, 2009 (siehe S. 14, 44, 46, 87, 90).

[CPS19] L. Custodio, S. Pesco und C. Silva. „An extended triangulation to the Marching Cubes 33 algorithm". In: *Journal of the Brazilian Computer Society* 25.6 (2019), S. 6. DOI: 10.1186/s13173-019-0086-6 (siehe S. 106).

[DFMD02] L. De Floriani, M. M. Mesmoudi und E. Danovaro. „Extraction of Critical Points and Nets Based on Discrete Gradient Vector Field“. In: *Eurographics 2002 - Short Presentations*. Eurographics Association, 2002. DOI: 10.2312/egs.20021041 (siehe S. 11).

[DF+99] L. De Floriani, E. Puppo, P. Cignoni und R. Scopigno. „Surface Simplification: Algorithms Overview“. In: *EG99 Tutorial*. Eurographics Association. 1999 (siehe S. 94).

[DFPS00] L. De Floriani, E. Puppo und R. Scopigno. „Level of Detail (LOD) Models Part Two“. In: *Eurographics*. Eurographics Association. 2000 (siehe S. 92).

[Dij80] E. W. Dijkstra. „Some beautiful arguments using mathematical induction“. In: *Acta Informatica* 13 (1980). Accessed on July 6, 2023, S. 1–8 (siehe S. 42).

[DSK17] B. D'mello, M. Satheesh und J. Krol. *Web Development with MongoDB and Node: Build fast web applications for handling any kind of data*. Packt Publishing, 2017. ISBN: 9781788394772 (siehe S. 125).

[DR04] M. Dubson und A. Rouinfar. *Charges and Fields*. https://phet.colorado.edu/sims/html/charges-and-fields/latest/charges-and-fields_en.html. 2004 (siehe S. 17).

[DP11] F. Dunn und I. Parberry. *3D Math Primer for Graphics and Game Development, 2nd Edition*. Taylor & Francis, 2011. ISBN: 9781568817231. DOI: 10.1201/b11152. URL: https://archive.org/details/3d-math-primer-for-graphics-and-game-development-2e/page/n9/mode/2up (siehe S. 132).

[Dur13] S. Durant. „Understanding Goal-Based Vector Field Pathfinding“. In: (2013). Zugegriffen: 19. Oktober 2023. URL: https://gamedevelopment.tutsplus.com/tutorials/gamedev-9007 (siehe S. 143).

[Eis+16] E. Eisemann, M. Schwarz, U. Assarsson und M. Wimmer. *Real-Time Shadows*. CRC Press, 2016. ISBN: 9781439867693. DOI: https://doi.org/10.1201/b11030 (siehe S. 53, 69).

[ESC00] J. El-Sana und Y.-J. Chiang. „External Memory View-Dependent Simplification“. In: *Eurographics* 19.3 (2000) (siehe S. 95).

[Eng+06a] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama und D. Weiskopf. *Real-Time Volume Graphics*. Wellesley, MA: A K Peters, Ltd., 2006. ISBN: 978-1-56881-266-3 (siehe S. 27).

[Eng+06b] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama und D. Weiskopf. „Real-time volume graphics“. In: *International Conference on Computer Graphics and Interactive Techniques*. 2006 (siehe S. 15).

[Fal21] P. Falstad. *3-D Electrostatic Field Simulation*. http://www.falstad.com/vector3de/. Online; Abruf: 14. November 2021. 2021 (siehe S. 17).

[Fli14] J. Flick. „Marching Squares, partitioning space“. In: (2014). URL: https://catlikecoding.com/unity/tutorials/marching-squares/ (siehe S. 137).

[Gam21] G. Gambetta. *Computer Graphics from Scratch*. No Starch Press, 2021, S. 248. ISBN: 1098128966, 9781098128968 (siehe S. 53).

[Gha08] S. Ghali. *Introduction to Geometric Computing*. Springer London, 2008. ISBN: 9781848001152. DOI: 10.1007/978-1-84800-115-2. URL: https://link.springer.com/book/10.1007/978-1-84800-115-2#bibliographic-information (siehe S. 74, 143).

[Gib20] J. Gibbons. *The School of Squiggol: A History of the Bird-Meertens Formalism*. Hrsg. von T. Astarte. Bd. 12233. LNCS. Springer, 2020. DOI: 10.1007/978-3-030-54997-8_2 (siehe S. 128).

[Gib+95] S. F. Gibson, J. Marks, D. Feinberg und M. Sosa. „A Heuristic Method for Generating 2D CSG Trees from Bitmaps“. In: *Mitsubishi Electric Research Laboratories* TR95-19 (1995). URL: http://www.merl.com (siehe S. 144).

[GG05] M. Glinz und H. Gall. *Software Engineering: Systematisches Programmieren: Lesbare und änderbare Programme schreiben.* Lecture, Wintersemester 2005/06, Kapitel 6. 2005 (siehe S. 42).

[Glu20] J. Gluckman. *The Science of Syntax.* Pressbooks, 2020. URL: https://pressbooks.pub/syntax/ (siehe S. 82).

[Goo21] R. L. Goodwin. *Linearizing Computing the Power Set with OpenMP.* Presentation at the 11th IEEE Workshop Parallel / Distributed Combinatorics and Optimization. 2021 (siehe S. 90).

[GE04] M. Gopi und D. Eppstein. „Single-Strip Triangulation of Manifolds with Arbitrary Topology". In: *Eurographics* 23.3 (2004) (siehe S. 149).

[Goz+14] Y. Gozudeli, O. Yildiz, H. Karacan, M. Baker, A. Minnet, M. Kalender, O. Ozay und M. Akcayol. „Extraction of Automatic Search Result Records Using Content Density Algorithm Based on Node Similarity". In: 2014-11 (siehe S. 82).

[Gra15] R. Grabowski. *BricsCAD V15 in 12 Lektionen. Inside BricsCAD 15.* 2015 (siehe S. 144).

[Gre+04] B. Gregorski, J. Senecal, M. Duchaineau und K. Joy. „Adaptive extraction of time-varying isosurfaces". In: *IEEE Transactions on Visualization and Computer Graphics* 10.6 (2004), S. 683–694. DOI: 10.1109/TVCG.2004.35 (siehe S. 16, 103).

[Gre19] B. S. Grewal. *Numerical Methods in Engineering and Science 3/E.* Dulles: Mercury Learning und Information, 2019 (siehe S. 63, 108).

[GZ22] R. Grosso und D. Zint. „A parallel dual marching cubes approach to quad only surface reconstruction". In: *The Visual Computer* 38.5 (2022), S. 1301–1316. DOI: 10.1007/s00371-021-02139-w. URL: https://doi.org/10.1007/s00371-021-02139-w (siehe S. 125).

[GMR20] T. Grzejszczak, R. Molle und R. Roth. „Tracking of dynamic gesture fingertips position in video sequence". In: Bd. vol. 30. No 1. Committee of Automatic Control und Robotics PAS, 2020, S. 101–122. DOI: 10.24425/acs.2020.132587. URL: http://journals.pan.pl/Content/115850/PDF/ACS-2020-1-5.pdf (siehe S. 5, 150).

[Hac08] D. Hachenberger. *Mathematik für Informatiker.* Always learning. Pearson Studium, 2008. ISBN: 9783827373205 (siehe S. 84).

[Had+05] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler und M. Gross. „Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces". In: *Comput. Graph. Forum* 24 (2005-09), S. 303–312. DOI: 10.1111/j.1467-8659.2005.00855.x (siehe S. 15).

[Ham17] G. Hamilton. „Generating Loop Invariants for Program Verification by Transformation". In: *Verification and Program Transformation (VPT 2017) EPTCS 253.* 2017, S. 36–53. DOI: 10.4204/EPTCS.253.5 (siehe S. 42).

[Hao18] J.-Q. Hao. „Optimal Reliable Point-in-Polygon Test and Differential Coding Boolean Operations on Polygons". In: *Symmetry* 10 (2018-10), S. 1–26 (siehe S. 67).

[Har16] S. Hartley. *Visualizing molecular isosurfaces (MOs, etc.) in Blender.* https://blog.hartleygroup.org/2016/02/22/visualizing-molecular-isosurfaces-mos-etc-in-blender/. Accessed: August 22, 2023. 2016 (siehe S. 28).

[Hat16] R. van Hattem. *Mastering Python.* Packt Publishing, 2016. ISBN: 9781785289729 (siehe S. 125).

[Hav08] D. Havey. *The icosahedron-based geodesic sphere.* https://web.archive.org/web/20180808214504/http://donhavey.com:80/blog/tutorials/tutorial-3-the-icosahedron-sphere/. Abgerufen am 27. Mai 2023. 2008 (siehe S. 94).

[HZ22] Z. Hu und W. Zhang. „Bird Meertens Formalism (BMF): A Quick Tour". In: *Department of Computer Science and Technology, Peking University.* Peking, 2022 (siehe S. 51).

[IK95] T. Itoh und K. Koyamada. „Automatic isosurface propagation using an extrema graph and sorted boundary cell lists“. In: *IEEE Transactions on Visualization and Computer Graphics* 1.4 (1995), S. 319–327. DOI: 10.1109/2945.485619 (siehe S. 15).

[JH12] J. Johnson und A. Henderson. *Conceptual Models: Core to Good Design*. Morgan & Claypool Publishers, 2012 (siehe S. 107).

[JK00] H. Jones und J. Kaandorp. „The modelling of growing natural forms“. In: *Eurographics 2000 - Tutorials*. Eurographics Association, 2000. DOI: 10.2312/egt.20001031 (siehe S. 28).

[Jor87] C. Jordan. *Cours d'analyse de l'école polytechnique*. Gauthier-Villars, 1887. DOI: 10.1017/cbo9781107300040. URL: https://www.maths.ed.ac.uk/~v1ranick/jordan/jordan.pdf (siehe S. 36).

[JNS17] M Jüttner, Z. N. und G. S. „A Standalone Interface for Web-Based Virtual Reality of Calculated Fields“. In: (2017). URL: https://www.comsol.com/paper/a-standalone-interface-for-web-based-virtual-reality-of-calculated-fields-52031 (siehe S. 18).

[Kah09] A. Kahler. *Creating an icosphere mesh in code*. http://blog.andreaskahler.com/2009/06/creating-icosphere-mesh-in-code.html. Accessed: August 11, 2023. 2009 (siehe S. 94).

[Kat17] N. Katraeva. *Erfassung von Kopfbewegungen für Augumented- und Virtual Reality-Anwendungen*. Wuppertal: Master-Thesis an der Uni-Wuppertal, 2017 (siehe S. 150, 163).

[Kha+22] D. Khan, A. Plopski, Y. Fujimoto, M. Kanbara, G. Jabeen, Y. J. Zhang, X. Zhang und H. Kato. „Surface Remeshing: A Systematic Literature Review of Methods and Research Directions“. In: *IEEE Transactions on Visualization and Computer Graphics* 28.3 (2022-03-01), S. 1680–1713. ISSN: 1077-2626, 1941-0506, 2160-9306. DOI: 10.1109/TVCG.2020.3016645. URL: https://ieeexplore.ieee.org/document/9167456/ (besucht am 2023-03-23) (siehe S. 91).

[Kob+00] L. Kobbelt, J. Vorsatz, U. Labsik und H.-P. Seidel. „A Shrink Wrapping Approach to Remeshing Polygonal Surfaces“. In: *Computer Graphics Forum* 18 (2000-04). DOI: 10.1111/1467-8659.00333 (siehe S. 94, 111, 112, 115).

[KBS00] L. P. Kobbelt, T. Bareuther und H.-P. Seidel. „Multiresolution Shape Deformations for Meshes with Dynamic Vertex Connectivity“. In: *Eurographics* 19.3 (2000) (siehe S. 92).

[Kot+19] D. Kothari, G. Awari, D. Shrimankar und A. Bhende. *Mathematics for Computer Graphics and Game Programming: A Self-teaching Introduction*. Mercury Learning und Information, 2019. ISBN: 9781683923565. URL: http://merclearning.com/titles/Mathematics_for_Computer_Graphics_and_Game_Programming_A_Self-Teaching_Introduction.html (siehe S. 113, 134).

[KMR06] K. Küpfmüller, W. Mathis und A. Reibiger. *Theoretische Elektrotechnik: Eine Einführung*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2006. ISBN: 9783540293736. URL: https://books.google.de/books?id=eKEkBAAAQBAJ (siehe S. 9).

[Lep+16] T. D. Lepich, R. Roth, R. Moller und C. Brandau. „Semi-automated sanitizing of component dependencies after subsystem reconfiguration in multimodal VR and AR frameworks“. In: *2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. IEEE, 2016. DOI: 10.1109/icce-berlin.2016.7684739 (siehe S. 5).

[LG95] L. Lippert und M. H. Gross. „Fast Wavelet Based Volume Rendering by Accumulation of Transparent Texture Maps“. In: *Computer Graphics Forum* 14.3 (1995). Hrsg. von F. Post und M. Göbel. ISSN 1067-7055, S. 431–444 (siehe S. 15).

[LC87] W. E. Lorensen und H. E. Cline. „Marching cubes: A high resolution 3D surface construction algorithm“. In: *Computer Graphics* 21.4 (1987), S. 163–169 (siehe S. 103).

[Luc19] G. Luciano. *Essential Computer Graphics Techniques for Modeling, Animating, and Rendering Biomolecules and Cells: A Guide for the Scientist and Artist.* Boca Raton, FL: CRC Press, 2019. ISBN: 9781498799218 (siehe S. 28).

[Lut09] T. Lutter. *Eine schnelle numerische Lösung der Neutronendiffusionsgleichungen nach dem Wellendigitalprinzip.* German. Published on December 7, 2009. Göttingen, Germany: Cuvillier Verlag, 2009, S. 212. ISBN: 3736931751 (siehe S. 42).

[MO17] M. Ma und A. Oikonomou. *Serious Games and Edutainment Applications: Volume II.* Bd. 2. Springer International Publishing, 2017. ISBN: 9783319516455. URL: https://link.springer.com/book/10.1007/978-1-4471-2161-9 (siehe S. 130).

[Mar+21] A. Marrs, P. Shirley, I. Wald, P. Christensen, D. Hart, T. M"uller, J. Munkberg, A. Pesce, J. Spjut, M. Vance und C. Yuksel. *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX.* Apress, 2021. ISBN: 978-1-4842-7184-1. DOI: 10.1007/978-1-4842-7185-8. URL: https://doi.org/10.1007/978-1-4842-7185-8 (siehe S. 27, 28).

[Mat+13] S. Matsumoto, K. Mitsufuji, Y. Hiasa und S. Noguchi. „Real Time Simulation Method of Magnetic Field for Visualization System With Augmented Reality Technology". In: *IEEE Transactions on Magnetics. 49. 1665-1668. 10.1109/TMAG.2013.2240672.* 2013 (siehe S. 9, 11, 13, 150).

[May19] L. S. Mayboudi. *Geometry Creation and Import With COMSOL Multiphysics.* Multiphysics Modeling Series. Mercury Learning & Information, 2019. ISBN: 9781683922148 (siehe S. 17, 18).

[Men14] A. Mena. *Beginning Haskell: A Project-Based Approach.* Books for professionals by professionals. Apress, 2014. ISBN: 9781430262510 (siehe S. 125).

[Men07] P. D. M. Mendel. *Mathematik C, Vorlesung WS 06/07.* Bergische Universität Wuppertal, 2007 (siehe S. 32, 33).

[MN97] S. Miguet und J.-M. Nicod. „Complexity Analysis of a Parallel Implementation of the Marching-Cubes Algorithm". In: *International Journal of Pattern Recognition and Artificial Intelligence* 11.07 (1997), S. 1141–1156. DOI: 10.1142/S0218001497000536. URL: https://doi.org/10.1142/S0218001497000536 (siehe S. 126).

[Mil14] B. Milewski. *Category Theory for Programmers.* Bartosz Milewski, 2014. URL: https://bartoszmilewski.com/2014/10/28/category-theory-for-programmers-the-preface/ (siehe S. 25, 26).

[Mla+19] D. Mlakar, M. Winter, H.-P. Seidel, M. Steinberger und R. Zayer. *AlSub: Fully Parallel and Modular Subdivision.* 2019 (siehe S. 128).

[Moa99] S. Moaveni. *Finite Element Analysis: Theory and Application with ANSYS.* Prentice Hall, 1999. ISBN: 9780137850983 (siehe S. 18).

[Moe03] P. D. R. Moeller. *Computer Graphics, Vorlesung WS 03/04.* Bergische Universität Wuppertal, 2003 (siehe S. 33, 66, 67).

[MR97] J. Munro und V. Raman. „Succinct representation of balanced parentheses, static trees and planar graphs". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science.* IEEE Comput. Soc, 1997. DOI: 10.1109/sfcs.1997.646100. URL: https://doi.org/10.1109%2Fsfcs.1997.646100 (siehe S. 82).

[MKH13] S. M. Musa, A. Kulkarni und V. Havanur. *Finite Element Analysis: A Primer.* Mercury Learning und Information, 2013. ISBN: 9781938549342. URL: https://www.merclearning.com/titles/Finite_Element_Analysis.html (siehe S. 17, 18).

[Nav16] C. R. Nave. *EyperPhysics - Electricity and Magnetism: Equipotential Lines*. http://hyperphysics.phy-astr.gsu.edu/hbase/electric/equipot.html. 2016 (siehe S. 31).

[NY06] T. S. Newman und H. Yi. „A survey of the marching cubes algorithm“. In: *Computers & Graphics* 30.6 (2006), S. 854–879. DOI: 10.1016/j.cag.2006.07.021 (siehe S. 104).

[Nie04] G. Nielson. „Dual Marching Cubes“. In: 2004-11, S. 489–496. ISBN: 0-7803-8788-0. DOI: 10.1109/VISUAL.2004.28 (siehe S. 125).

[NH91] G. M. Nielson und B. Hamann. „The asymptotic decider: Resolving ambiguities in marching cubes“. In: *IEEE Transactions on Visualization and Computer Graphics* 1.3 (1991), S. 248–258 (siehe S. 105).

[NH92] P. Ning und L. Hesselink. „Octree Pruning for Variable-Resolution Isosurfaces“. In: *Visual Computing*. Hrsg. von T. L. Kunii. Tokyo: Springer Japan, 1992, S. 349–363. ISBN: 978-4-431-68204-2 (siehe S. 15, 36).

[OSu+01] C. OSullivan, J. Dingliana, F. Ganovelli und G. Bradshaw. „Collision Handling for Virtual Environments“. In: Eurographics Association, 2001. DOI: 10.2312/egt.20011055 (siehe S. 94).

[PDP01] P. Pahl, R. Damrath und F. Pahl. *Mathematical Foundations of Computational Engineering: A Handbook*. Engineering online library. Springer Berlin Heidelberg, 2001. ISBN: 9783540679950 (siehe S. 84).

[Par+98] S. Parker, P. Shirley, Y. Livnat, C. Hansen und P.-P. Sloan. „Interactive Ray Tracing for Isosurface Rendering“. In: *Proceedings of the Conference on Visualization '98*. VIS '98. Research Triangle Park, North Carolina, USA: IEEE Computer Society Press, 1998, 233–238. ISBN: 1581131062 (siehe S. 14, 36).

[PDS16] S. Pulleyking, D. Das und J. Schultz. „Simplified robotic thumb inspired by surgical intervention“. In: 2016-06, S. 1200–1206. DOI: 10.1109/BIOROB.2016.7523794 (siehe S. 10).

[Rai18] D. Raine. *Mathematical Physics: An Introduction*. Essentials of Physics Series. Mercury Learning & Information, 2018. ISBN: 9781683922063. URL: https://www.merclearning.com/titles/Mathematical_Physics_An_Introduction.html (siehe S. 31).

[ras14] rasteron. *Rigged Male Base Mesh*. 2014. URL: https://opengameart.org/content/rigged-male-base-mesh (siehe S. 132).

[RTY94] J. H. Reif, J. D. Tygar und A. Yoshida. „Computability and Complexity of Ray Tracing“. In: *Discrete Comput. Geom.* 11.3 (1994), 265–288. ISSN: 0179-5376. DOI: 10.1007/BF02574009. URL: https://doi.org/10.1007/BF02574009 (siehe S. 14).

[Rho+02] P. Rhodes, R. Laramee, R. Bergeron und T. Sparr. „Uncertainty Visualization Methods in Isosurface Volume Rendering“. In: (2002-06) (siehe S. 38, 39).

[Ros23] Rosetta Code. *Power set*. https://rosettacode.org/wiki/Power_set. Abgerufen am 13. April, 2023. 2023 (siehe S. 84).

[RB22] R. Roth und B. Beitz. „Macro-Controlled Generation of Geometry Using Vector Fields and their Application“. In: *Proceedings of GAME-ON 2022 Lisbon, Portugal*. EUROSIS, 2022-09 (siehe S. 5).

[Rot+18] R. Roth, T. Grzejszczak, M. Niezabitowsk und R. Moller. „Adapting Strategies to Display Simulations of Electric Fields Spawned by Point Charges for Augmented Reality Application“. In: 2018-09, S. 1–5. DOI: 10.1109/ICCE-Berlin.2018.8576225 (siehe S. 5, 107).

[RMP20] R. Roth, R. Möller und T. Pursche. „Extensible Augmented Reality Assisted Contact-Free Patient Surveillance in Emergency Context“. In: 2020-11. DOI: 10.1109/ICCE-Berlin50680.2020.9352178 (siehe S. 5).

[San20] S. F. Santonato. *A Complete End-to-End Simulation Flow for Autonomous Driving Frameworks*. Torino: Politecnico di Torino, 2020 (siehe S. 18).

[Sch21] A. G. Schmidt. „Equipotential Lines“. In: (2021) (siehe S. 31).

[Sch+20] M.-P. Schmidt, L. Couret, C. Gout und C. Pedersen. „Structural topology optimization with smoothly varying fiber orientations“. In: *Structural and Multidisciplinary Optimization* 62 (2020-12). DOI: 10.1007/s00158-020-02657-6 (siehe S. 28).

[SLM88] A. Schmitt, W. Leister und H. Müller. „Ray Tracing Algorithms - Theory and Practice“. In: 1988-01, S. 997–1030. ISBN: 978-3-642-83541-4. DOI: 10.1007/978-3-642-83539-1_42 (siehe S. 15).

[Sch16] R. Schwartz. „The Polygonal Jordan Curve Theorem“. In: (2016). URL: https://www.math.brown.edu/reschwar/M123/jc.pdf (siehe S. 109).

[SB18] A. Sen und A. Bhattacharya. *Radar Systems and Radio Aids to Navigation*. Mercury Learning & Information, 2018. ISBN: 9781683921196. URL: https://www.merclearning.com/titles/Radar_Systems_and_Radio_Aids_to_Navigation.html (siehe S. 2).

[Sha02] N. B. Shamlo. *Electrostatics 3D*. https://web.archive.org/web/20170212035805/http://www.electrostatics3d.com/. Offline, erreichbar +ber Internet-Archiv. 2002 (siehe S. 17).

[SV93] V. Shapiro und D. L. Vossler. „Separation for boundary to CSG conversion“. In: *ACM Trans. Graph.* 12.1 (1993), 35–55. ISSN: 0730-0301. DOI: 10.1145/169728.169723. URL: https://doi.org/10.1145/169728.169723 (siehe S. 144).

[She98] H.-W. Shen. „Isosurface Extraction in Time-Varying Fields Using a Temporal Hierarchical Index Tree“. In: *Proceedings of the Conference on Visualization '98*. VIS '98. Research Triangle Park, North Carolina, USA: IEEE Computer Society Press, 1998, 159–166. ISBN: 1581131062 (siehe S. 16, 36, 81).

[SA17] T. X. Short und T. Adams. *Procedural Generation in Game Design*. Games & animation. CRC Press, Taylor & Francis Group, 2017. ISBN: 9781138743311. DOI: 10.1201/9781315156378. URL: https://www.taylorfrancis.com/books/edit/10.1201/9781315156378/procedural-generation-game-design-tanya-short-tarn-adams (siehe S. 129).

[Sid18] R. Siddiqui. „Path Planning Using Potential Field Algorithm“. In: (2018). URL: https://medium.com/@rymshasiddiqui/a30ad12bdb08 (siehe S. 143).

[SBG22] K. Sierra, B. Bates und T. Gee. *Head First Java*. 3. Aufl. O'Reilly Media, Inc., 2022. ISBN: 9781491910771 (siehe S. 125).

[Sil10] L. F. M. S. Silva. *Merging Meshes Using Dynamic Regular Triangulation*. Universidade Federal Do Rio Grande Do Sul, 2010 (siehe S. 74).

[Sim08] L. Simon. *An Introduction to Multivariable Mathematics*. Synthesis lectures on mathematics and statistics. Morgan & Claypool Publishers, 2008. ISBN: 9781598298017 (siehe S. 31).

[Ski09] S. S. Skiena. *The Algorithm Design Manual*. 2nd. Springer Science & Business Media, 2009, S. 730. URL: https://link.springer.com/book/10.1007/978-1-84800-070-4 (siehe S. 114).

[SB06] B.-S. Sohn und C. Bajaj. „Time-varying contour topology“. In: *IEEE Transactions on Visualization and Computer Graphics* 12.1 (2006), S. 14–25. DOI: 10.1109/TVCG.2006.16 (siehe S. 16, 81, 103).

[Son18] C. Song. *The Scaled Boundary Finite Element Method: Introduction to Theory and Implementation*. Wiley Series in Computational Mechanics. Wiley, 2018. ISBN: 9781119388463 (siehe S. 18).

[Spi14] P. Spiess. „Better Know An Algorithm: Marching Squares“. In: (2014). URL: https://web.archive.org/web/20141230044328/http://devblog.phillipspiess.com/better%20know%20an%20algorithm/2010/02/23/better-know-marching-squares.html (siehe S. 137).

[Sta16] J. Stam. *The Art of Fluid Animation.* Boca Raton, FL: CRC Press, 2016 (siehe S. 27, 28).

[Str03] L. Straßburger. *Einführung in die Computationale Logik.* https://www.ps.uni-saarland.de/courses/cl-ss03/skript/. Skript. Universität des Saarlandes, 2003 (siehe S. 82).

[SH99] P. Sutton und C. D. Hansen. „Isosurface Extraction in Time-Varying Fields Using a Temporal Branch-on-Need Tree (T-BON)". In: *Proceedings of the Conference on Visualization '99: Celebrating Ten Years.* VIS '99. San Francisco, California, USA: IEEE Computer Society Press, 1999, 147–153. ISBN: 078035897X (siehe S. 16).

[Sys20] A. Systems. *VRXPERIENCE Perceived Quality.* 2020 (siehe S. 18).

[Tab15] M. Tabatabaian. *COMSOL5 for Engineers.* Dulles, VA, USA: Mercury Learning und Information, 2015. ISBN: 1942270429 (siehe S. 17).

[Tom15] C. Tominski. *Interaction for Visualization.* Synthesis Lectures on Visualization. Springer International Publishing, 2015. ISBN: 9783031026003. DOI: 10.2200/S00651ED1V01Y201506VIS003. URL: https://books.google.de/books?id=Q4FyEAAAQBAJ (siehe S. 107).

[VG97] E. Veach und L. J. Guibas. „Metropolis Light Transport". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., 1997, 65–76. ISBN: 0897918967. DOI: 10.1145/258734.258775. URL: https://doi.org/10.1145/258734.258775 (siehe S. 14, 15).

[Ver19] J. Verschelde. *Algorithmic Cost of Voronoi Diagrams.* MCS 481 Lecture 22, Computational Geometry. Accessed on August 7, 2023 from http://homepages.math.uic.edu/~jan/mcs481/voronoicost.pdf. 2019 (siehe S. 95).

[Vil14] O. Villar. *Learning Blender: A Hands-On Guide to Creating 3D Animated Characters.* Learning. Pearson Education, 2014. ISBN: 9780133886276 (siehe S. 76).

[Vö+08] B. Vöcking, H. Alt, M. Dietzfelbinger, R. Reischuk, C. Scheideler, H. Vollmer und D. Wagner. *Taschenbuch der Algorithmen.* Berlin Heidelberg: Springer, 2008 (siehe S. 63).

[Wat+10] A. I. Watson, J. D. Fournier, T. P. Lericos und E. J. Szoke. „The Use of D3D When Examining Tropical Cyclones". In: *NOAA/National Weather Service* (2010) (siehe S. 28).

[WB98] C. Weigle und D. C. Banks. „Extracting Iso-Valued Features in 4-Dimensional Scalar Fields". In: *Proceedings of the 1998 IEEE Symposium on Volume Visualization.* VVS '98. Research Triangle Park, North Carolina, USA: Association for Computing Machinery, 1998, 103–110. ISBN: 1581131054. DOI: 10.1145/288126.288175. URL: https://doi.org/10.1145/288126.288175 (siehe S. 16).

[Wen13] R. Wenger. *Isosurfaces: Geometry, Topology, and Algorithms.* 1st. New York: A K Peters/CRC Press, 2013 (siehe S. 104, 137).

[Wie18] J.-P. Wiese. *Eingabe geometrischer Transformationsdaten in virtueller Realität.* Wuppertal: Bachelor-Thesis, 2018 (siehe S. 11, 33, 150, 163).

[Wil94] H. S. Wilf. *Algorithms and Compexity.* University of Pennsylvania, 1994 (siehe S. 46).

[Won14] J. Wong. „Metaballs and Marching Squares". In: (2014). URL: http://jamie-wong.com/2014/08/19/metaballs-and-marching-squares/ (siehe S. 137).

[Xu+19] Q. Xu, D. Yan, W. Li und Y. Yang. „Anisotropic Surface Remeshing without Obtuse Angles". In: *Computer Graphics Forum* 38.7 (2019-10), S. 755–763. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.13877. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13877 (besucht am 2023-05-03) (siehe S. 91).