Lehrstuhl für Bauinformatik
Fakultät für Bauingenieur- und Vermessungswesen
Technische Universität München

Virtual Reality basierte Analyse und interaktive Steuerung von Strömungssimulationen im Bauingenieurwesen

Siegfried Kühner

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. F. Valentin
Prüfer der Dissertation:	
	1. Univ.-Prof. Dr. rer. nat. E. Rank
	2. Univ.-Prof. Dr. rer. nat. Chr. Zenger

Die Dissertation wurde am 22. Mai 2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 04. Dezember 2003 angenommen.

für Cecilia

Vorwort und Dank

Die vorliegende Arbeit ist während meiner Tätigkeit als Mitarbeiter am Lehrstuhl für Bauinformatik der technischen Universität München (Dezember 1999 – März 2003) im Rahmen eines von der Firma Siemens AG gewährten Stipendiums entstanden. An dieser Stelle möchte ich mich recht herzlich bei einer Reihe von Menschen bedanken, die einen positiven Einfluss auf das Gelingen dieser Arbeit hatten.
Ganz besonders möchte ich mich bei meinem Doktorvater Herrn Prof. Dr. Ernst Rank für seine ausgezeichnete fachliche und didaktische Betreuung bedanken.
Ein weiterer Dank geht an Herrn Prof. Dr. Manfred Krafczyk für die hervorragende und engagierte fachliche Betreuung der Arbeit. Seine stete Diskussionsbereitschaft und Unterstützung vor allem im Bereich der Strömungsmechanik waren eine unerlässliche Hilfe.
Herrn Prof. Dr. Christoph Zenger danke ich für die Übernahme des zweiten Gutachtens und für das Interesse an dieser Arbeit.
Bei der Firma Siemens AG bedanke ich mich für die finanzielle Unterstützung in Form eines Stipendiums und das damit entgegengebrachte Vertrauen.
Meinen Kollegen am Lehrstuhl danke ich für das angenehme und motivierende Arbeitsklima. Besonders hervorheben möchte ich Herrn Bernd Crouse, der über drei Jahre mit mir ein Büro geteilt hat. Mit ihm zusammen habe ich eine Vielzahl erfolgreicher Projekte durchgeführt. Herrn Manuel Schulz danke ich besonders für die interessanten Diskussionen im Bereich der parallelen Programmierung. Nicht unerwähnt bleiben sollten die Studenten, die in Diplomarbeiten Teile der Implementierungen dieser Arbeit realisiert haben.
Schließlich möchte ich die Menschen hervorheben, die mir persönlich am meisten bedeuten. Meiner lieben Freundin Cecilia danke ich für den Rückhalt und die Geduld, die sie mir während der Ausarbeitung der Arbeit entgegengebracht hat. Meinen Eltern danke ich von ganzem Herzen für die ständige Unterstützung meiner Ausbildung.

Kurzfassung

In der vorliegenden Arbeit werden neuartige Methoden zur Unterstützung des Planungsprozesses im Bauwesen unter strömungsmechanischen Gesichtspunkten vorgestellt. Auf der Grundlage der Lattice-Boltzmann Methode wird dazu eine Virtual Reality basierte Ergebnisauswertung und anschließend eine vollständig interaktive Strömungssimulation untersucht.
Zunächst werden Verfahren der Visualisierung wissenschaftlich-technischer Daten weiterentwickelt. Die interaktive Analyse großer Datensätze wird durch eine Reduktion der Ergebnisdaten basierend auf hierarchischen Datenstrukturen unterstützt. Weiterhin werden Kriterien zur Bewertung des menschlichen Komfortempfindens in den Prozess der Visualisierung integriert. Virtual Reality Techniken und die kombinierte Darstellung der Ergebnisse der Berechnung mit dem CAD-Modell erleichtern das Verständnis der Strömung.
Die anschließende Implementierung eines prototypischen interaktiven Strömungssimulators verbindet eine Umgebung zur Eingabe und Visualisierung über eine Interprozesskommunikation mit einem Supercomputer oder einem Workstation-Cluster. Dadurch wird es möglich, eine Strömung zur Laufzeit der Simulation durch Verändern der Hindernisse im Strömungsgebiet und der Randbedingungen zu manipulieren und das Ergebnis unmittelbar zu verfolgen.

Abstract

In this thesis, new methods supporting the planning process of a building with respect to fluid mechanical properties are introduced. Based on the Lattice-Boltzmann method, an interactive analysis of the simulation results in Virtual Reality is examined first, followed by the investigation of a complete interactive fluid flow simulation.
First, common techniques of scientific visualization are extended: Hierarchical data structures are used to reduce the huge amount of data generated by the simulation and to map the data. Furthermore, criteria describing human comfort are integrated in the process of visualization. Virtual Reality techniques and the combined representation of the CAD-model with the mapped simulation results facilitate the comprehension of the investigated fluid flow problem.
The following implementation of a computational steering system of a Lattice-Boltzmann based CFD simulation connects a front-end for the user-input and the visualization to a supercomputer or a workstation cluster using inter–process communication. Thus, it is possible to interact with a running simulation, manipulate obstacles in the flow field and boundary conditions while following the evolution of the flow immediately.

Inhaltsverzeichnis

Abbildungsverzeichnis **xiii**

1 Einleitung **1**
1.1 Einführung und Motivation . . . 1
1.2 Aufbau der Arbeit . . . 3

2 Grundlagen verwendeter Methoden aus der Informatik **5**
2.1 Scientific Visualization . . . 5
2.1.1 Allgemeines . . . 5
2.1.2 Pipeline der Datenvisualisierung . . . 6
2.1.3 Abbildungsmethoden in der numerischen Strömungsmechanik . . . 9
2.2 Virtual Reality-Technologien . . . 11
2.2.1 Begriff und Erscheinungsform . . . 12
2.2.2 Aufbau eines VR-Systems . . . 14
2.2.3 Hardware . . . 15
2.2.4 Datenhaltung: VRML und Szenegraph . . . 16
2.2.5 Einsatzbereich und ausgewählte Projekte . . . 18
2.3 Computational Steering . . . 19
2.3.1 Allgemeines und Definition . . . 19
2.3.2 Systemaufbau und Komponenten . . . 19

3 Numerische Strömungssimulationen nach der Lattice-Boltzmann Methode und paralleles Rechnen **21**
3.1 Numerische Simulation der Strömungsmechanik mit der Lattice-Boltzmann Methode 21
3.1.1 Allgemeines zur numerischen Strömungssimulation . . . 21
3.1.2 Die Lattice-Boltzmann Methode . . . 25
3.1.3 Einsatzbereiche . . . 29
3.2 Paralleles Rechnen und Supercomputing . . . 29
3.2.1 Rechnerarchitekturen und Supercomputing . . . 30
3.2.2 Ebenen der Parallelisierung und explizite Parallelisierungskonzepte . . . 30
3.2.3 Effizienzanalysen und Grenzen . . . 32

4 Analyse der Anforderungen an die Steuerung und Auswertung von Strömungssimulation **35**
4.1 Einsatzbereich: Integration in den Planungsprozess im Bauwesen . . . 35
4.2 Funktionsumfang . . . 38

5 Strategisches Lösungskonzept **43**
5.1 Verwandte Arbeiten - Stand der Technik . . . 43
5.2 Integration in ein Gesamtkonzept . . . 45
5.2.1 Vom geometrischen Modell zur Bemessungsgrundlage . . . 46
5.2.2 Wahl eines Verfahrens zur Simulation und eines zugehörigen Gittertypen . 49

5.2.3 Hierarchische Datenstrukturen als Organisationsprinzip 50
5.3 Zwei-Stufen-Strategie 52
5.3.1 Interaktive Simulationen in der Produktauslegung als Ergänzung einer klassischen Simulation 54
5.3.2 Untersuchung zur Genauigkeit von Lattice-Boltzmann Simulationen bei grober Auflösung 55

6 Interaktive Analyse der Simulationsergebnisse 57
6.1 Eine hybride Netz-Baumdatenstruktur 57
6.1.1 Abstraktion Zelle 59
6.1.2 Objektorientierte Infrastruktur 60
6.1.3 Aspekte zur Optimierung der Performance 63
6.1.4 Integration in eine graphisch–interaktive Entwicklungsumgebung 64
6.2 Datenreduktion mit hierarchischen Datenstrukturen 65
6.2.1 Reduktion von knotenassoziierten Daten 65
6.2.2 Strategie der Reduktion des gesamten Baums 67
6.2.3 Parallelisierung der Datenreduktion 69
6.2.4 Beispiele 70
6.3 Datenabbildung mit hierarchischen Datenstrukturen 73
6.3.1 Partikelverfolgung 73
6.3.2 Isoflächen 74
6.4 Darstellung von Kriterien des menschlichen Komforts 75
6.4.1 Komfortkriterien 75
6.4.2 Integration in das Postprocessing einer numerischen Simulation 77
6.5 Kombinierte Darstellung von Simulationsdaten mit CAD-Daten in Virtual Reality 78
6.6 Nachbereitung der Ergebnisse: Multimediale Dokumentation 82
6.7 Beispiel: Simulation in einem Großraumbüro 83

7 Interaktive Steuerung numerischer Strömungssimulationen: Der computergestützte Windkanal VFReal 87
7.1 Systementwurf 87
7.1.1 Analyse der Anwendungsfälle 87
7.1.2 Identifikation von Komponenten 89
7.1.3 Auswahl der Werkzeuge zur Implementierung 91
7.2 Teilsystem Simulation 92
7.2.1 Funktionsumfang 92
7.2.2 Architektur des Systems 94
7.2.3 Datenhaltung 96
7.2.4 Parallelisierung der Lattice-Boltzmann Simulation 98
7.2.5 Erweiterung der parallelen Lattice-Boltzmann Simulation für die interaktive Steuerung und Auswertung 103
7.2.6 Modifikation des diskreten Gitters zur Laufzeit der Simulation 105
7.3 Teilsystem Virtual Reality basierte Umgebung zur Ein- und Ausgabe 106
7.3.1 Der Szenegraph – Inhalte der dreidimensionalen virtuellen Welt 107
7.3.2 Immersives Benutzermenü 107
7.3.3 Geometrische Hindernisse im Strömungsgebiet 109
7.3.4 Abbildung der Ergebnisse der Simulation (Datenvisualisierung) 109
7.3.5 Zusammenspiel der Komponenten 113

7.4 Beispiel zur Anwendung der interaktiven Strömungssimulation 114
7.5 Zusammenfassung und Diskussion 114

8 Zusammenfassung und Ausblick **119**
8.1 Zusammenfassung 119
8.2 Ausblick 120

Literaturverzeichnis **123**

Vorveröffentlichte Teilergebnisse **131**

Abbildungsverzeichnis

2.1 Pipeline der Datenvisualisierung 7
2.2 Ansätze zur Partitionierung der Pipeline der Datenabbildung 9
2.3 Beispiele für Abbildungen von CFD-Daten: Vektoren (links oben), Stromlinien (rechts oben), Isoflächen (links unten), Faltungsintegral (LIC) (rechts unten) . . . 10
2.4 Vereinfachtes Schema eines Virtual Reality Systems 14
2.5 Beispiele für Eingabe-Geräte: Stylus (Wand, links) und Positions-Tracking (rechts) 16
2.6 Display Systeme für Virtual Reality: Panorama-Wand (links oben), CAVE^{TM} (rechts oben), Head Mounted Display (unten links), Holobench (unten rechts) . . 17
2.7 Beispiel eines einfachen Szenegraphen 17
2.8 Computational Steering – Systemaufbau und Datenfluss 20

3.1 CFD als Ergänzung zum Experiment in der Produktauslegung 22
3.2 Ebenen der parallelen Verarbeitung 31

4.1 Informationsaustausch in der Vorplanung 36
4.2 Veranschaulichung der komplexen Geometrie und der Vielfalt von Bauwerken . . . 37
4.3 Schnittstellen innerhalb einer numerischen Simulation 38
4.4 Verschiedene Abbildungen eines CAD-Modells auf ein diskretes Modell 40

5.1 Verwandte Arbeiten: Virtueller Windtunnel (links oben – aus [16]), Strömungsvisualisierung auf hierarchischen Gittern (rechts oben – aus [90]), Interaktive Simulation des Innenraums eines Fahrzeugs im Projekt VISiT (links unten – aus [55]), Interaktive Finite-Element Analyse von Brücken (rechts unten – aus [21]) 46
5.2 Klassischer Ablauf numerischer Simulationen mit vielfältigen Modellen 47
5.3 Gesamtkonzept zur numerischen Simulation basierend auf Produktmodelldaten . . 48
5.4 Der Spacetree (hier: Octree in 3D) als Organisationsprinzip zum Übergang zwischen den Ebenen einer numerischen Simulation 51
5.5 Erzeugung des numerischen Gitters aus einem B-Rep Volumenmodell (links) über ein Facettenmodell der Oberfläche (Mitte) und einem Octree (rechts) 52
5.6 Zwei-Stufen-Strategie zum Einsatz von CFD in der Produktauslegung 54
5.7 Dynamische Strukturen bei der Umströmung eines Zylinders (aus [67]) 56
5.8 Aus [57]: Schnitt durch den rechten Hufeisenwirbel (2) und den beiden symmetrischen Seitenwirbeln (4) (links) – Symmetrischer Tütenwirbel (3) (rechts) 56

6.1 Baumdatenstruktur mit Verknüpfung der Blätter – schneller Zugriff auf die benachbarten Zellen 58
6.2 Unterschiedliche Verzeigerung der Blätter bei benachbarten Zellen mit gleicher Baumtiefe (links) und ungleicher Baumtiefe (rechts) 58
6.3 Knotendaten an den Ecken werden von mehreren Zellen gleichzeitig referenziert . 59
6.4 Zelle einer hierarchischen Diskretisierung: Nummerierung der Kindzellen, Eckknoten, Flächen und Kanten 60

6.5 Klassendiagramm der hierarchischen Datenstrukturen in der Bibliothek `VFVis` – wichtige Funktionen und Attribute sind exemplarisch aufgeführt 61
6.6 Integration von `VFVis` in eine Umgebung zur visuellen Programmierung 64
6.7 Datenreduktion durch Subsampling einer Ebene einer Baumdatenstruktur 65
6.8 Die Krümmung des Vektorfeldes als Kriterium zur Datenreduktion 66
6.9 Gesamt-Strategien der Reduktion (Veranschaulichung in 2D) 67
6.10 Einsortieren der sequentiell gespeicherten Zellen in die Baumdatenstruktur 69
6.11 3D Datenreduktion – Schnittebene (37345 Punkte) durch das ursprüngliche Gitter (7469000 Punkte) 71
6.12 3D Datenreduktion – Kanten der Schnittebene (9427 Punkte) durch ein ausgedünntes Gitter (889653 Punkte) 71
6.13 3D Datenreduktion – Schnittebene (9427 Punkte) durch ein ausgedünntes Gitter (889653 Punkte) 71
6.14 3D Datenreduktion – Schnittebene (19656 Punkte) durch das ursprüngliche Gitter (5936112 Punkte) 72
6.15 3D Datenreduktion – Kanten der Schnittebene (5336 Punkte) durch ein ausgedünntes Gitter (643745 Punkte) 72
6.16 3D Datenreduktion – Schnittebene (5336 Punkte) durch ein ausgedünntes Gitter (643745 Punkte) 72
6.17 Ermittlung der Schnittfigur zwischen Zelle und Isofläche beim Algorithmus *Marching Cubes* 74
6.18 Abbildung von Komfortkriterien (hier: PMV-Index) als Bestandteil des Postprocessings 77
6.19 Weitere Beispiele zu kombinierter Visualisierung 78
6.20 Kombinierte Darstellung von CAD-Daten und Ergebnissen der Simulation in Virtual Reality 79
6.21 Kombinierte Visualisierung in Virtual Reality: Szenegraph und angegliederte Werkzeuge 80
6.22 Ablauf der Konvertierung eines IFC-Datensatzes nach VRML 80
6.23 Übergang vom Produktdatenmodell nach VRML – Beispiel 81
6.24 Software-Architektur zur Erstellung einer multimedialen Dokumentation 82
6.25 Ablegen von Informationen in eine Datenbank während einer Visualisierung – hier: in AVS/Express 83
6.26 Automatisch generierte Dokumentation in HTML 84
6.27 Beispiel Großraumbüro: Geometrie als Facettenmodell 85
6.28 Strömungsgebiet im Großraumbüro: Octree-Darstellung 85

7.1 Grundprinzip einer interaktiv zu steuernden Simulation 88
7.2 Anwendungsfälle innerhalb einer interaktiven Strömungssimulation 89
7.3 Komponenten einer interaktiven Strömungssimulation 90
7.4 Aufteilung des Systems – Randbedingungen, Hindernisse und Gebietszerlegung (in diesem Beispiel: 4 Domänen) 93
7.5 Systemarchitektur Alternative *MasterSim*: Kommunikation zwischen der Simulation und der Ein- und Ausgabe über einen Hauptknoten 94
7.6 Systemarchitektur Alternative *MasterVis*: Direkte Kommunikation zwischen den Prozessen der Simulation und der Ein- und Ausgabe 95

7.7 Links: Veranschaulichung des Austauschs der Verteilungsfunktionen in 2D, nur Kommunikation mit dem nächsten Nachbarn – rechts: Auszutauschende Verteilungsfunktionen am Beispiel des Modells D3Q19 99
7.8 Hitachi SR8000 – Leistung der parallelen Lattice-Boltzmann Simulation 100
7.9 Workstation-Cluster Pentium 4, 1GB SDRAM – Leistung der parallelen Lattice-Boltzmann Simulation 100
7.10 Leistung eines Einzelknotens auf der Hitachi SR8000 in Abhängigkeit von der Speicherung der Verteilungsfunktionen 102
7.11 Ablauf einer interaktiven Lattice-Boltzmann Simulation – der Datenaustausch mit der Visualisierung (Hauptknoten) erfolgt parallel zur Berechnung 103
7.12 Leistung der interaktiven Simulation bei einer Kommunikation zwischen der Hitachi SR8000 und der Virtual Reality Anlage im LRZ München – das linke Bild entspricht der Variante *MasterVis* und das rechte Bild repräsentiert die Variante *MasterSim*. 104
7.13 Voxelierung der Hindernisse mit dem Bresenham-Algorithmus – Skizze) 105
7.14 Klassendiagramm der Virtual Reality basierten Komponente zur Steuerung der Simulation 107
7.15 Der Szenegraph von `VFReal` 108
7.16 3D Benutzer-Menü als Teil der Szene 108
7.17 Der Sub-Szenegraph für geometrische Objekte (Hindernisse) in `VFReal` 109
7.18 Möglichkeiten des Startens der Pipeline der Visualisierung von Stromlinien 111
7.19 Beispiele zur Datenabbildung in `VFReal` – Stromlinien (oben links), Isofläche (oben rechts), Schnittebene (unten links), Vektorpfeile (unten rechts) 112
7.20 Anwendung der interaktiven Strömungssimulation – Bildserie 1-3 115
7.21 Anwendung der interaktiven Strömungssimulation – Bildserie 4-6 116

Kapitel 1

Einleitung

1.1 Einführung und Motivation

Computer und Informationstechnologien sind heutzutage allgegenwärtig. Längst geht es dabei nicht nur um isolierte Rechenmaschinen, sondern um Werkzeuge zur Kooperation und zur Interoperabilität. Der IT-Sektor ist für Unternehmen zu einem strategischen Feld geworden, mit dem man sich nachhaltig Vorteile auf dem Markt sichern will.
Betrachtet man nun den technischen Bereich, so ist die digitale Produktentwicklung ins Zentrum der Entwicklung gerückt. Das heute noch lange nicht erreichte Endziel ist es, z.B. im Bauwesen vom Entwurf des Architekten bis hin zur Fertigstellung des Bauwerks vollständig mit einem digitalen 3D Modell zu arbeiten. Der starke Konkurrenzkampf am Weltmarkt führt zu einem immensen Preisdruck und folglich zur Notwendigkeit kürzerer Entwicklungszeiten. Es ist dabei offensichtlich, dass wesentliche Entscheidungen in einer frühen Planungsphase getroffen werden müssen. Dazu werden klassische Methoden mit Prüfständen und physikalischen Prototypen durch die Simulation eines 3D Modells am Computer ergänzt bzw. soweit wie möglich ersetzt. Beispielsweise werden auf dem Gebiet der Crashsimulationen in der Automobilindustrie mehrere Tausend numerische Simulationen zur Entwicklung eines Fahrzeugs durchgeführt [49]. Dem gegenüber steht lediglich eine kleine Reihe von Versuchen zur Bestätigung der Simulationsergebnisse oder sogar nur zur Einhaltung gesetzlicher Vorschriften. Im Automobilbau ist das 3D Modell Stand der Technik und Digital-Mockup Systeme sind etabliert, um Untersuchungen zum Design, zur Kinematik und zur Erkennung von Kollisionen beim Einbau von Bauteilen durchzuführen.
Einer der wichtigsten Beiträge und zugleich aber auch eine der größten Herausforderungen für eine digitale Produktentwicklung ist die Integration und Weiterentwicklung von Verfahren des Scientific Computing. Hier werden physikalische oder chemische Probleme bzw. die zugehörigen mathematischen Modelle mit Methoden der numerischen Simulation und der Informatik gelöst. Oft handelt es sich um ein gekoppeltes, ggf. nichtlineares partielles Differentialgleichungssystem. Die meisten relevanten Systeme der Struktur- und Strömungsmechanik besitzen dazu im allgemeinen keine analytische Lösung. In diesem Kontext hat sich aufgrund der Fortschritte der Numerischen Mathematik, der Informatik und der Computer-Hardware das Feld der numerischen Simulation als dritte Säule der Wissenschaft neben Theorie und Empirie etabliert [50].
In dieser Arbeit sollen Fragestellungen der numerischen Strömungsmechanik im Zentrum stehen. John von Neumann hat diese Entwicklung in einem spektakulären Aufsatz 1946 begründet [75]. Seine Idee des digitalen Windkanals bleibt weiterhin ein starker Aspekt der Forschung im wissenschaftlichen Höchstleistungsrechnen und spielt auch in dieser Arbeit eine zentrale Rolle.
Grundsätzlich ist eine numerische Simulation an sich in die folgenden drei Teile untergliedert:

Preprocessing: Ausgangspunkt ist die Definition eines geometrischen Modells, das in eine diskrete Form überführt und mit Rand- und Anfangsbedingungen eines physikalischen Problems erweitert wird (Problemdefinition).

Computation: Das betrachtete Differentialgleichungssystem wird in einer diskreten Form mit Methoden der Numerischen Mathematik (Finite-Elemente-, Finite-Differenzen- und Finite-Volumen-Verfahren, oder ähnliche) am Rechner gelöst. Dieser Teilbereich hat heutzutage einen entscheidenden Einfluss auf die Entwicklung von modernen Supercomputern.

Postprocessing: Die Ergebnisse der Berechnung werden mit Verfahren der grafischen Datenverarbeitung in Computerdarstellungen überführt und analysiert.

Aufgrund der unterschiedlichen Anforderungen dieser drei Teile ist es möglich, dass jeder Prozess eine andere Datenstruktur verwendet und sogar auf einer unterschiedlichen Plattform ausgeführt wird. Der Gesamtprozess der Simulation wird dann durch Schnittstellenprobleme belastet. Die Notwendigkeit einer engeren Kopplung wurde von der Informatik bereits frühzeitig erkannt und Lösungsvorschläge entwickelt (siehe [39]).
Im Folgenden wird die Problematik im Bereich des Bauwesens genauer beschrieben. In Anbetracht der Tatsache, dass Konrad Zuse – auf den wesentliche Entwicklungen der Rechnertechnik und der Informationsverarbeitung zurück gehen – ein Bauingenieur war, möchte man denken, dass die Verwendung von digitalen 3D Modellen im Bauingenieurwesen einen sehr ausgeprägten, hohen Entwicklungsstand besitzt. Es fällt jedoch auf, dass andere Industriezweige (v.a. die Automobilindustrie) das Bauingenieurwesen überholt haben. Folgende Aspekte können als Gründe angeführt werden:

- Beispielsweise sind in der Strukturmechanik reduzierte Modelle, die 3D Probleme mit 2D Modellen abbilden, historisch gewachsen und haben weiterhin ihre Berechtigung. Für eine Reihe von Problemstellungen werden damit jedoch Einschränkungen gemacht und 3D FEM-Formulierungen sind dann geeigneter [32]. Trotz CAD-Anbindungen [10] wird es noch erhebliche Anlaufzeit benötigen, bis sich diese 3D Verfahren durchsetzen, zumal die reduzierten Modelle in den Normen etabliert sind.

- Die an der Planung eines Bauwerks beteiligten Personen bilden eine sehr heterogen zusammengesetzte Gruppe, in der eine Vielzahl unterschiedlicher Sichten auf das Modell vorliegt. Dies manifestiert sich in einer völlig unterschiedlichen Semantik der verwendeten Modelle, die nicht einfach durch ein gemeinsames, standardisiertes Datenformat im Sinne eines digitalen Produkts zu lösen sind [87]. Die Produktmodellspezifikation IFC [1] definiert dafür zwar anwendungsspezifische Sichten (so genannte *Views*), dennoch verbreitet sich die IFC im Bauingenieurwesen weiterhin recht langsam.

Die numerische Strömungsmechanik auf der Basis der Navier-Stokes-Gleichungen ist im Bauingenieurwesen recht wenig etabliert. Dies ist eigentlich überraschend, wenn man sich die breite Palette möglicher Einsatzgebiete veranschaulicht [27]:

- Bauwerksstabilität: Das gängige Vorgehen, Belastungen durch Windkräfte aus Standard-Tabellenwerken zu entnehmen und Risiken durch erhöhte Sicherheitsbeiwerte zu begegnen ist einerseits unwirtschaftlich und kann anderseits Resonanzphänomene (vor allem bei Brücken und hohen Türmen) infolge der nichtlinearen Dynamik der bidirektionalen Fluid-Struktur-Wechselwirkung nicht abdecken.

- Gebäudebelüftung: Basierend auf der Vorhersage von horizontalen und vertikalen Gebäudedurchströmungen können Lüftungskonzepte entwickelt und die Anordnung von Klimaanlagen bzw. Ventilationen optimiert werden.

[1] Industry Foundation Classes, http://www.iai-ev.de

- Menschliches Komfortempfinden: Mit Hilfe einer numerischen Simulation können in Innenräumen ungünstige Kombinationen von Luftgeschwindigkeit und Temperatur ermittelt und daraus Indikatoren für das menschliche Wohlbefinden abgeleitet werden.

- Schadstoffausbreitung und Stadtbelüftung: Die Entwicklung effizienter Konzepte zur Evakuierung eines Gebäudes kann durch die Simulation der Ausbreitung von Rauch oder von Schadstoffen unterstützt werden. Weiterhin kann bereits in der konzeptionellen Phase der Planung eines Industriegebietes die Ausbreitung entstehender Schadstoffe in die umgebenden Siedlungen erkannt und berücksichtigt werden.

Eine einfache Erklärung für die oben angesprochene schwache Etablierung mag zunächst sein, dass die Ausbildung der Bauingenieure in der Strömungsmechanik im Vergleich zur Strukturmechanik recht kurz kommt. Weiterhin herrscht allgemein noch nicht genug Vertrauen in die numerische Strömungssimulation, vor allem bei der Verwendung von Turbulenzmodellen [58]. Zusätzlich ist der Aufwand der Berechnung enorm, meist nur von spezialisierten Ingenieurbüros zu bewältigen und letztendlich teuer.
Der erfolgreiche Einsatz numerischer Simulationen der Strömungsmechanik (CFD) im Bauwesen muss einem sehr unterschiedlich zusammengesetzten Personenkreis verständlich gemacht werden. Im Sinne des Auftraggebers muss die Simulation möglichst effizient und nach dem Prinzip des minimalen Aufwands und maximalen Gewinns an Erkenntnissen durchzuführen sein. Hier setzt diese Arbeit gemeinsam mit einer eng verzahnten Dissertation [25] an. Als Grundlage dient das Verfahren der Strömungssimulation nach der Lattice-Boltzmann Methode. Die Anwendbarkeit im Bereich des Bauingenieurwesens wurde von Krafczyk [57] gezeigt. Die Lattice-Boltzmann Methode bringt erhebliche Vorteile bei den im Bauwesen auftretenden komplexen Strukturen und eignet sich besonders gut für eine effiziente Kopplung der drei Teilprozesse der Simulation (siehe oben) mit Baumdatenstrukturen. Ein erstrebenswertes Fernziel dieser Bemühungen ist die interaktive Strömungssimulation für Problemstellungen der Ingenieurspraxis, d.h. die Option, die Strömung online zu verfolgen und dabei gleichzeitig in die Problemdefinition eingreifen zu können.
In diesem Umfeld beschäftigt sich die vorliegende Arbeit mit der Analyse und Visualisierung sowie der Zusammenführung der Teilprozesse in ein prototypisches System zur interaktiven Steuerung einer numerischen Strömungssimulation. Dabei wird eine Zweistufen-Lösung vorgeschlagen: Mit der Hilfe von High-End Visualisierungsverfahren und Virtual Reality wird ein Beitrag zur Unterstützung der Kommunikation zwischen den Planungsbeteiligten gegeben. Als zweiter Schritt werden die Prozesse Preprocessing, Berechnung und Auswertung in einer Applikation vereinigt, indem ein Supercomputer bzw. Workstation–Cluster direkt mit einem Visualisierungsrechner verbunden wird. Optimierungen und Lösungsansätze zu den Bereichen Gittergenerierung und numerische Simulation finden sich in der oben angesprochenen Arbeit von Crouse [25].

1.2 Aufbau der Arbeit

In den Kapiteln 2 und 3 werden die Grundlagen der verwendeten Methoden und Technologien knapp umrissen. Eine interaktive Strömungssimulation verwendet Verfahren der Visualisierung wissenschaftlich-technischer Daten, der Virtuellen Realität und des Computational Steerings. Grundlegende Aspekte zur numerischen Strömungssimulation nach der Lattice-Boltzmann Methode und der parallelen Implementierung auf einem Supercomputer werden in Kapitel 3 vorgestellt.

Kapitel 4 beleuchtet zunächst Fragen der Integration einer Strömungssimulation in den Planungsprozess im Bauwesen und berücksichtigt insbesondere das heterogene Umfeld der Beteiligten in der Planung sowie die komplexen Strukturen von Bauwerken. Darauf aufbauend werden der gewünschte Funktionsumfang im Bereich der Steuerung und Auswertung der Berechnung und die Anforderungen an eine Implementierung festgelegt.
Unter Abgrenzung von verwandten Arbeiten (Abschnitt 5.1) wird anschließend in Kapitel 5 eine zweistufige Lösungsstrategie vorgestellt: Zunächst wird die Visualisierung und Auswertung in ein Gesamtkonzept einer numerischen Simulation im Stapelverarbeitungsbetrieb eingegliedert. Dieses Konzept stellt einen Rechenkern nach der Lattice-Boltzmann Methode in den Mittelpunkt, der die Grundlage schafft, den Datenaustausch der Teilprozesse Datenaufbereitung, Berechnung und Auswertung unter Verwendung von raumpartitionierenden Datenstrukturen zu optimieren (in Anlehnung an von Frank [39] formulierten Ideen). Die zweite Stufe sieht vor, das Gesamtkonzept mit Hilfe einer Interprozesskommunikation über heterogene Hardware-Plattformen in einer einzigen Applikation zu vereinen, um die Simulation zur Laufzeit steuern und auswerten zu können (Computational Steering). Dieses System arbeitet vorerst noch mit kartesischen Gittern bei moderaten Genauigkeitsanforderungen. Voruntersuchungen (Abschnitt 5.3) zeigen, dass man damit bei Lattice-Boltzmann Simulationen bereits Ergebnisse erhält, die qualitativ die wesentlichen Eigenschaften einer Strömung wiedergeben.
Die Visualisierung und Analyse der Ergebnisse ist bereits für Simulationen im klassischen Ablauf einer Stapelverarbeitung an die zu verwendenden raumpartitionierenden Datenstrukturen anzupassen. Die in [25] entwickelte Datenstruktur ist eine hybride Netz-Baumdatenstruktur und wird zunächst kurz umrissen (Abschnitt 6.1). Die Eigenschaft der Rekursion wird genutzt, um die zu erwartenden riesigen Datenmengen effizient zu reduzieren (Abschnitt 6.2). Weiterhin wird gezeigt, dass klassische Verfahren der Datenvisualisierung auf diese Datenstruktur übertragbar sind und teilweise optimiert werden können (Abschnitt 6.3). Das Feld der Datenvisualisierung wird im Sinne der besseren Verständigung im Planungsprozess von Bauwerken erweitert: Dazu gehören die kombinierte Darstellung von hochaufgelösten CAD-Daten mit den graphisch abgebildeten Ergebnissen der Simulation (Abschnitt 6.5) und die Darstellung von Kriterien des menschlichen Komfortempfindens (Abschnitt 6.4). Die Ergebnisse werden schließlich systematisch in eine Datenbank übertragen, um automatisch eine multimediale Dokumentation zu erzeugen (Abschnitt 6.6). Aspekte zur Implementierung (Abschnitt 6.1.2) und ein Beispiel einer Simulation eines Grossraumbüros (Abschnitt 6.7) schließen das Kapitel ab.
Der zweite Punkt der zweistufigen Lösungsstrategie resultiert in einer interaktiven Steuerung einer numerischen Strömungssimulation, die in Kapitel 7 ausführlich vorgestellt wird. Es wird ein prototypisches System entworfen, das eine Komponente zur Eingabe und Visualisierung über eine Interprozesskommunikation mit einem Supercomputer bzw. einem Rechen-Cluster verbindet, so dass Problemdefinition, Berechnung und Auswertung in einer Applikation zusammengeführt sind (Abschnitt 7.1). Das wesentliche Ziel dabei ist, die grundsätzliche Machbarkeit einer interaktiven Strömungssimulation sowie deren Nutzen zu zeigen und ein System zu entwickeln, das flexibel erweitert werden kann. Im weiteren Verlauf dieses Kapitels werden die Komponenten Simulation (Abschnitt 7.2) sowie Ein- und Ausgabe Front-End (Abschnitt 7.3) erläutert und die Schnittstellen beschrieben. Schließlich wird das System hinsichtlich Leistung und Stabilität diskutiert (Abschnitt 7.5).
Im letzten Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und Ansatzpunkte für weitere Forschungsarbeiten vorgeschlagen.

Kapitel 2

Grundlagen verwendeter Methoden aus der Informatik

Aus dem Bereich der Informatik spielen in dieser Arbeit Methoden der Visualisierung wissenschaftlich-technischer Daten, des Umfeldes der Virtuellen Realität und des Gebiets des Computational Steering eine zentrale Rolle. Die folgenden Abschnitte geben jeweils einen kurzen Abriss dieser Themengebiete, mit dem Ziel, eine Wissensbasis für die weitere Diskussion zu schaffen.

2.1 Scientific Visualization

2.1.1 Allgemeines

Die Bedeutung der Visualisierung ist vor allem im Zeitalter moderner Computersimulationen unumstritten. Beispiele findet man in den Natur- und Ingenieurwissenschaften, in der Medizin, in der Telekommunikation oder in Handel und Finanzen. Daten unterschiedlicher Herkunft und Natur werden in eine graphische Form übergeführt und damit die am stärksten ausgeprägte Fähigkeit der menschlichen Wahrnehmung angesprochen. Es ist somit nicht verwunderlich, dass die Disziplin der Visualisierung wesentlich älter ist als numerische Computersimulationen – man denke zum Beispiel an alte Landkarten. Experimente wurden stets von abgeleiteten Diagrammen begleitet, um die Versuchsergebnisse in klarer und verständlicher Form aufzubereiten.
Zunächst ist es notwendig, die folgenden, in enger Beziehung zueinander stehenden Begriffe zu definieren (siehe [94]):

Image Processing bzw. Bildverarbeitung beschäftigt sich mit der Bearbeitung von 2D Bildern. Dazu zählen z.B. Transformationen und Betonungen (Kontrast, Schärfe, etc.).

Computergraphik behandelt den Prozess der Abbildung eines geometrischen Modells auf ein vom Computer generiertes Bild. Dazu zählen 2D Pixel–Operationen sowie 3D Rendering-Techniken.

Visualisierung transformiert Daten in geometrische Informationen bzw. Bildinformationen. Im Mittelpunkt steht der Begriff der Einsicht mit dem Hauptziel, Unsichtbares präzise, effizient und nachvollziehbar darzustellen.

Die Visualisierung in ihrer heutigen Form in Verbindung mit Methoden der Computergraphik wurde 1987 als formale Disziplin *Visualization in Scientific Computing* bzw. *Scientific Visualization* [68] eingeführt. Weiterhin muss die Pionierarbeit von Tufte [110] genannt werden, in der die Verwendung von Computerbildern zur Darstellung von Daten eingeführt wird. Erst später hat sich der Bereich *Information Visualization* [52] entwickelt. Information Visualization bezieht sich auf Datensätze, deren Natur nicht zwingend in Verbindung mit einem physikalischen

Problem oder einem Koordinatensystemen steht. Informationsträger sind zumeist riesige Datenbanken etwa aus den Bereichen Handel und Finanzen. Als Techniken werden Graphen, Icons oder geometrische Anordnungen verwendet. Hauptaugenmerk ist das Suchen nach Abhängigkeiten.
Die weiteren Ausführungen beschränken sich nun auf das Feld der ‚Scientific Visualization'[1], das Messdaten oder Simulationsergebnisse von naturwissenschaftlichen Problemen verarbeitet. Zielsetzung ist das Erfassen eines physikalischen Phänomens aus den Rohdaten.
Durch die stets wachsende Leistung der Rechner und die Etablierung von Methoden des parallelen Rechnens, charakterisieren sich die erzeugten Daten durch

- enorme Größe (bis zu mehreren Tera-Byte)
- grosse Informationsdichte
- Multidimensionalität
- Zeitabhängigkeit

Diese wachsenden Ansprüche führten zu einer rasanten Entwicklung auf dem Feld der Visualisierung, ohne die ein Wissenschaftler oder Ingenieur heute keine Chance mehr hätte, Simulationen effizient und verständlich auszuwerten.
Die folgenden Abschnitte beinhalten den Weg von den Rohdaten der Simulation zur Abbildung am Bildschirm (Pipeline der Visualisierung) sowie eine kurze Übersicht häufig verwendeter Methoden der Datenabbildung für numerische Simulationen der Strömungsmechanik. Weiterführende Literatur zum Thema Visualisierung findet man z.B. in [17, 77, 109, 14, 38]. Zu jüngeren Entwicklungen sei auf die Konferenzen *ACM SIGGRAPH* und *IEEE Visualization* verwiesen.

2.1.2 Pipeline der Datenvisualisierung

Ziel der Visualisierung ist es, einen Rohdatensatz in eine Bildform zu überführen. Dabei wird eine Prozesskette durchlaufen, die als Pipeline der Datenvisualisierung bezeichnet wird und in Abbildung 2.1 dargestellt ist.
Entscheidendes Kriterium der Visualisierung wissenschaftlich–technischer Daten ist die Möglichkeit der Interaktion durch den Benutzer in jeder Stufe der Pipeline.

Quelldaten: Diese werden in der Regel durch eine numerischen Simulation oder ein Experiment erzeugt und können auch analytisch gewonnen werden. Die zugehörigen Berechnungsverfahren halten die Daten meist auf diskreten Gittern. Man unterscheidet dabei zwischen strukturierten und unstrukturierten Gittern (Netzen). Im letzteren Fall können die Zellen der Gitter eine beliebige Anordnung und Form aufweisen. Die Quelldaten sind dann

- eine Liste von Knoten mit Koordinaten und
- zugeordneten Daten sowie
- eine Beschreibung der Anordnung der Zellen und
- gegebenenfalls den Zellen zugeordnete Daten.

[1] Scientific Visualization wird im weiteren Verlauf auch mit Visualisierung wissenschaftlich–technischer Daten oder Datenvisualisierung bezeichnet.

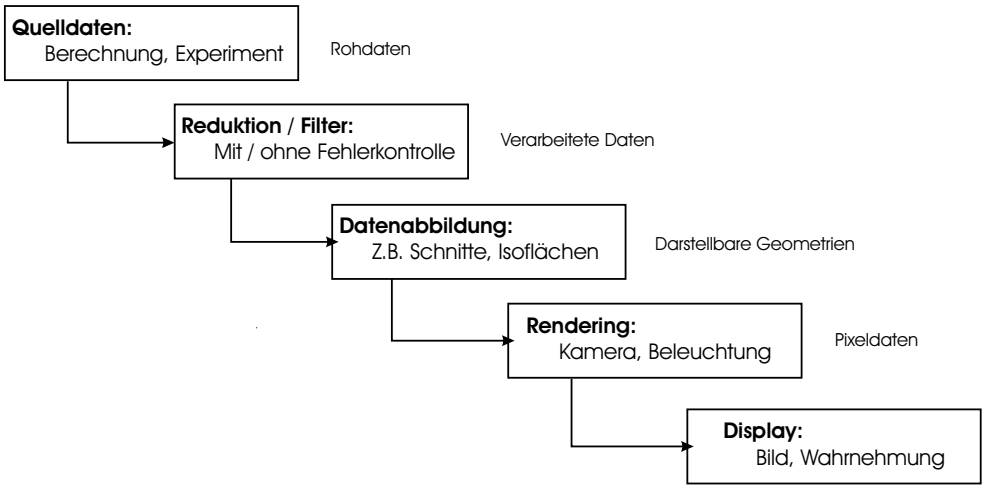

Abbildung 2.1: Pipeline der Datenvisualisierung

Bei strukturierten Gittern entfällt die explizite Angabe der Konnektivität der Zellen, weil diese implizit aus dem Typ des Gitters und der Reihenfolge der Knoten hervorgeht. Eine besondere Rolle spielen hierarchische Formen wie dünne Gitter [117] oder Octrees [39], auf die in Abschnitt 5.2 noch genauer eingegangen wird. Weiterhin finden hybride Formen wie blockstrukturierte Gitter eine breite Anwendung.

Filter, Reduktion: Typische Datensätze einer CFD Simulation im Bereich des Höchstleistungsrechnens beginnen bei einer Größenordnung von $\mathcal{O}(10^6)$ Gitterpunkten (siehe Abschnitt 6.7 oder [59]). Im Sinne einer effizienten Datenabbildung ist es unerlässlich, die Rohdaten zu reduzieren. Dazu gehören zunächst Methoden zur Extraktion von Untermengen des ursprünglichen Datensatzes wie z.B. das Filtern von Daten innerhalb einer wohl definierten geometrischen Region, innerhalb eines vorgegebenen Wertebereichs (*threshold*), oder die Auswahl einer Komponente aus einem multi-dimensionalen Datensatz (etwa Geschwindigkeit in x-Richtung bei Speicherung aller Komponenten des Geschwindigkeitsvektors). Die zweite Gruppe bilden so genannte Subsampling-Techniken, die eine kleine Menge von Punkten innerhalb einer bestimmten Umgebung in einem einzigen zentral liegenden Punkt überführen und diesem schließlich den Mittelwert der Werte an den ursprünglichen Punkten zuordnen.

Abbildung: Der Kernpunkt der Visualisierung liegt in der Überführung von Daten in darstellbare Objekte mit Geometrie, Topologie und Farbgebung. Abschnitt 2.1.3 gibt einen Überblick der wichtigsten Abbildungstechniken im Bereich der Strömungsmechanik. Diese Objekte werden als Satz von Linien oder Polygonen beschrieben. Oftmals findet am Ende der Abbildung noch eine Reduktion der Anzahl der Polygone (z.B. nach [95], [92]) statt. Dieser Schritt diente ursprünglich der Entlastung des Renderings und verliert mittlerweile durch die enorme Weiterentwicklung der Graphik–Hardware[2] an Bedeutung.

Rendering: In dieser Stufe wird die nach der Abbildung erzeugte dreidimensionale Szene in ein zweidimensionales Bild (eine Pixelform) überführt (siehe [17]). Dieser Vorgang wird selbst in der so genannten *Rendering-Pipeline* abgearbeitet. In der Datenvisualisierung werden

[2]Eine Graphikkarte in einem Standard-PC kann heute über 30 Millionen Dreiecke pro Sekunde verarbeiten (http://www.nvidia.com).

hierfür hauptsächlich lokale Beleuchtungsmodelle verwendet. Dazu gehören die Verfahren *Flat Shading*, *Gouraud Shading* und *Phong Shading*. Bei ausreichender Qualität bleibt die Möglichkeit der Interaktion durch Verändern der Benutzerperspektive erhalten. Auf einer Energiebilanz basierende Methoden (*Radiosity*) oder Techniken der Strahlverfolgung (*Ray-Tracing*) sind hier zu rechenintensiv, können aber zur Weiterverarbeitung in Filmsequenzen oder in photorealistischen Bildern verwendet werden.

Display: Vom Benutzer wahrgenommen wird zuletzt das Bild auf einer 2D Projektionsfläche – in der Regel dem Bildschirm. Allerdings fehlt hierbei der Eindruck von räumlicher Tiefe, den ein Benutzer durch seine räumliche Vorstellungskraft kompensieren muss. In Abschnitt 2.2 werden Projektionstechniken gezeigt, um dieser Problematik entgegenzuwirken.

Computer-Programme zur Datenvisualisierung bilden diese Pipeline mit objektorientierten Methoden ab. Es ist jedoch zu beachten, dass nicht das klassische Vorgehen der Objektorientierung – Datenstruktur und Algorithmen in einer Klasse zusammenzufassen – verwendet wird. Die Prozesse (Algorithmen) der Pipeline werden als vollwertige Objekte betrachtet. Das bedeutet, dass zum Beispiel eine Isofläche (siehe unten) eine eigene Klasse ist, die auf verschiedenen Gitterdatentypen operieren kann und nicht eine Methode derselben. Der letzte Fall hätte zur Folge, dass die Algorithmen abhängig von den Daten sind und mehrfach implementiert werden müssten. Andererseits gibt es eine Reihe fundamentaler Funktionen wie z.B. das Bilden von Gradienten, die wiederum sehr stark vom Gittertyp abhängig sind. Beispielsweise umgeht die in dieser Arbeit eingesetzte Programmier-Bibliothek VTK[3] diese Problematik mit einem hybriden Ansatz. Hält man nun mehrere Pipelines nebeneinander und referenziert die Bausteine quer, entsteht ein Netzwerk der Visualisierung. Kommerzielle Programmpakete wie AVS/Express[4] oder OpenDX[5] bieten zur Zusammenstellung der Pipeline bzw. des Netzwerkes Werkzeuge der graphisch–interaktiven Programmierung an.
Zur Implementierung der Pipeline der Visualisierung bieten sich die in Abbildung 2.2 dargestellten Ansätze der Partitionierung an:

- Die am häufigsten verwendete Variante (d.h. der heute verwendete Standard in kommerziellen Programmpaketen) sieht eine starke Graphik–Workstation als lokalen Rechner vor, auf dem das Filtern, die Datenabbildung, das Rendern und die Ausgabe erfolgt. Die Quelldaten werden auf einem Rechen-Cluster (z.B. Linux-Cluster [54]) oder einem Supercomputer (siehe Abschnitt 3.2) erzeugt (in Abbildung 2.2 und in den weiteren Erläuterungen als Rechenserver bezeichnet).

- Die nächste Stufe der Aufteilung sieht vor, das Filtern auf den Rechenserver auszulagern. Man kann oft nicht davon ausgehen, dass der vorhandene lokale Rechner die notwendige Leistungsfähigkeit hat, die unter Umständen riesigen Mengen von Ergebnisdaten einer numerischen Simulation auf Höchstleistungsrechnern effizient zu verarbeiten. Entscheidender Vorteil gegenüber dem ersten Ansatz ist, dass bei geringerer Datenmenge die Möglichkeit der Interaktion mit der Datenabbildung bestehen bleibt. Dieser Ansatz wird später auch in Abschnitt 6.2 vorgeschlagen und umgesetzt.

- Die Integration der Datenaufbereitung auf dem Rechenserver führt zwar dazu, dass die Anforderungen an die Leistungfähigkeit des lokalen Rechners sehr gering werden und die

[3] http://www.kitware.com
[4] http://www.avs.com
[5] http://www.opendx.org

Darstellung sogar in einem Browser denkbar wird. Die Interaktion mit der Datenabbildung ist dann allerdings nur noch möglich, wenn die entsprechende Applikation auf dem lokalen Rechner über eine Form der Interprozess-Kommunikation mit dem Rechenserver Daten austauscht (in [115] vorgeschlagen). Man muss bei dieser Art der Partitionierung noch unterscheiden, ob auf dem Rechenserver eine eigenständige Applikation die Datenaufbereitung übernimmt, oder ob diese direkt in die Simulation eingebunden ist ("Co-Visualisierung" [93]). Der letzte Fall ist sicher eine interessante Lösung für sehr grosse Datensätze, hat aber den Nachteil, dass eine Interaktion des Benutzers mit der Datenabbildung eine neue Simulation des betrachteten Problems zur Folge hat.

- Die letzte Variante sieht nur noch die Ausgabe des gerenderten Bildes auf dem lokalen Rechner vor.

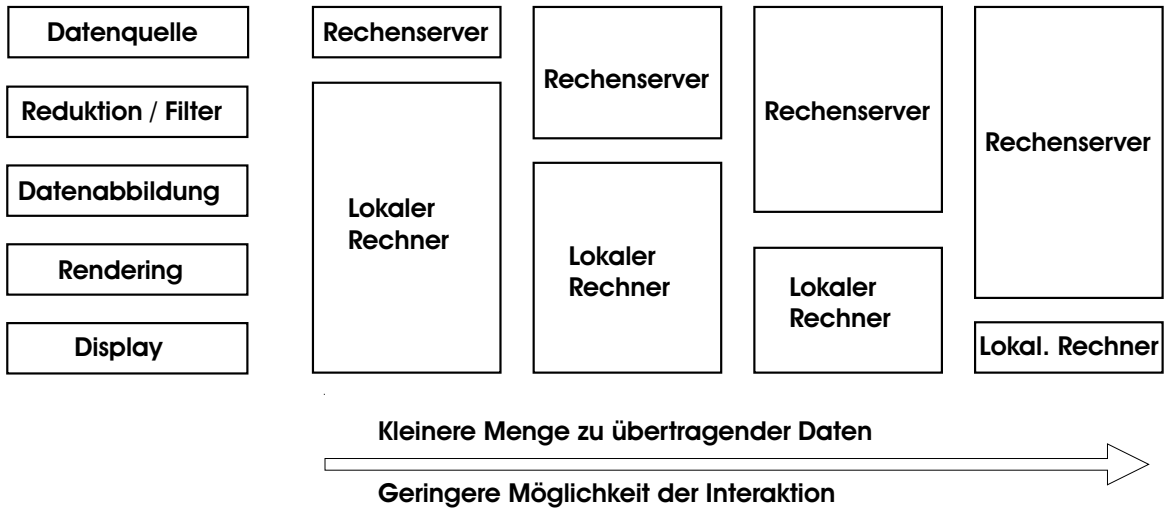

Abbildung 2.2: Ansätze zur Partitionierung der Pipeline der Datenabbildung

Je mehr Abschnitte der Pipeline vom Rechenserver übernommen werden, desto weniger Daten müssen auf dem lokalen Rechner verarbeitet werden. Dieser Aspekt gewinnt vor allem in interaktiv gesteuerten Simulationen (siehe Kapitel 7) an Bedeutung, wenn ständig neue Ergebnisse vom Rechenserver zur Graphik–Workstation versendet werden und die Bandbreite des Netzwerkes ein Flaschenhals werden kann. Weiterhin wird es mit steigender Auslagerung schwieriger, in die Darstellung der Ergebnisse einzugreifen, um eine interaktive Datenanalyse zu betreiben. In [79] findet man eine detailliertere Beschreibung der Vor- und Nachteile der jeweiligen Ansätze im Umfeld des Höchstleistungsrechnens.

2.1.3 Abbildungsmethoden in der numerischen Strömungsmechanik

Die folgende Auflistung beschreibt die wichtigsten Methoden zur Abbildung von Datensätzen strömungsmechanischer Berechnungen und zeigt deren Nutzen auf:

Planare Schnitte mit Farbgebung: Das Extrahieren ebener Teilgebiete ist eine sehr effiziente Technik, weil damit einerseits kaum räumliches Vorstellungsvermögen vom Benutzer gefordert wird und andererseits keine aufwendigen Berechnungsverfahren notwendig sind. Auf der Schnittebene werden die Datenwerte einer skalaren Feldgröße als Farbverteilung

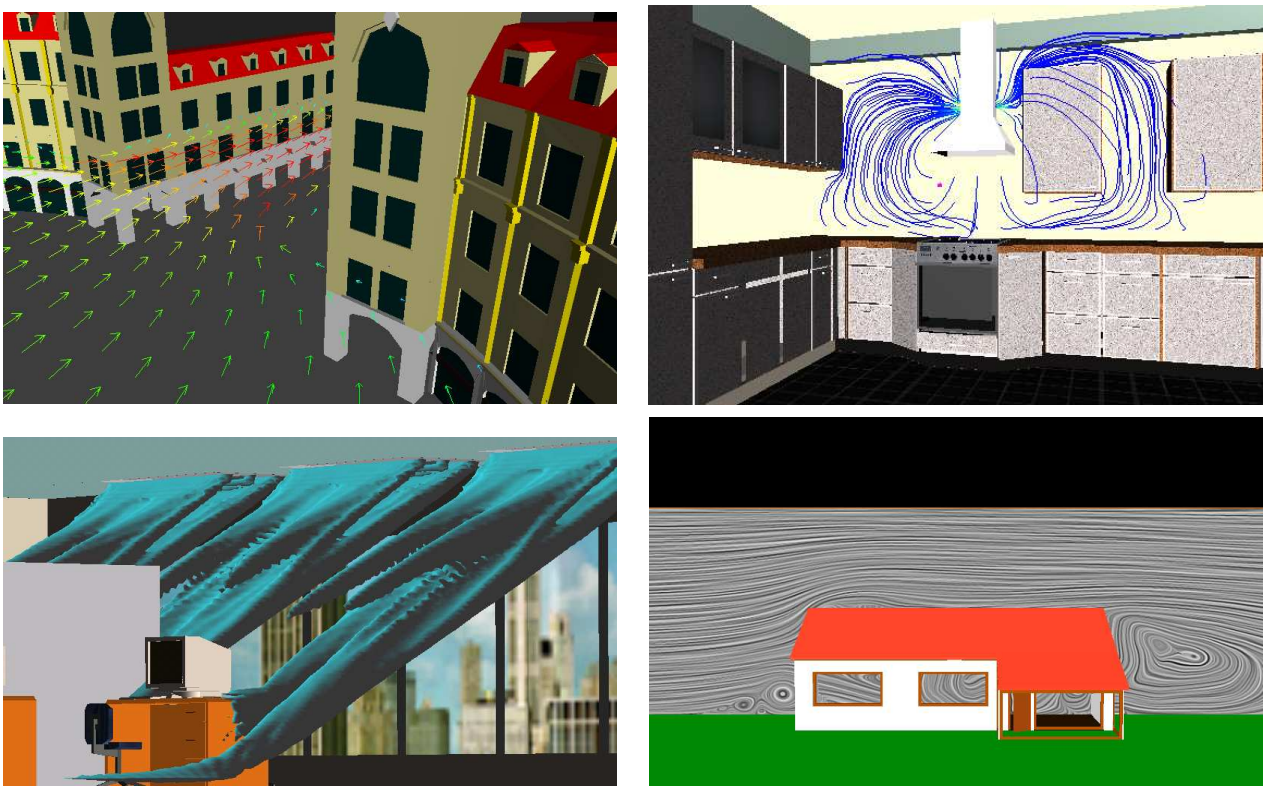

Abbildung 2.3: Beispiele für Abbildungen von CFD-Daten: Vektoren (links oben), Stromlinien (rechts oben), Isoflächen (links unten), Faltungsintegral (LIC) (rechts unten)

dargestellt (siehe z.B. Abbildung 6.11). Werden durch die Ebenen etwa andere Objekte im Strömungsfeld verdeckt, bringt eine transparente Darstellung schnell Abhilfe, um die Korrelation zwischen Strömung und der Umgebung erkenntlich zu machen.

Isoflächen: Eine Isofläche (siehe Abbildung 2.3, links unten) beschreibt alle Orte im Raum, an denen eine betrachtete skalare Größe einem Schwellwert f_{iso} entspricht. Isoflächen eignen sich besonders gut, um die Entwicklung einer Strömung und transiente Eigenschaften aufzuzeigen (siehe Abschnitt 7.3.4 und 7.4). Zur Berechnung der Isofläche wird fast ausschließlich der Algorithmus *Marching Cubes* [65] verwendet. Zusätzlich kann die erzeugte Fläche mit einer Farbinformation versehen werden, um etwa Abhängigkeiten zwischen den Größen der Strömung anzuzeigen (Beispiel: Isofläche des Druckes bei Darstellung des Betrags der Geschwindigkeit als Farbinformation).

Stromlinien: Wirbelstrukturen können mit Stromlinien sehr gut identifiziert werden. Dabei wird der Weg masseloser Partikel durch ein Strömungsfeld verfolgt und zu einer Kurve verbunden, deren Tangente folglich in jedem Punkt parallel zum Ortsvektor liegt. Dargestellt werden entweder die durchgezogene Kurve als Polylinie oder Glyphen (z.B. Kugeln) an den Endpunkten der einzelnen Segmente der Polylinie. Die Startpunkte der Stromlinien (sog. Injektionsorte oder Saatpunkte) werden gewöhnlich entlang von Linien oder Ebenen angeordnet. Die Anordnung und die Anzahl der Saatpunkte erfordert jedoch eine gewisse Sorgfalt, denn es kann leicht passieren, dass sich mehrere Stromlinien überdecken und damit keine vernünftige Aussage über die Eigenschaften der Strömung getroffen werden kann. Ein Beispiel für Stromlinien ist in Abbildung 2.3 (rechts oben) gegeben. Eine Er-

weiterung ist die Darstellung von dünnen Bändern anstelle von Linien, die entsprechend der Rotation im Strömungsfeld entlang der Stromlinie gedreht werden. In [89] wird als Erweiterung die Verfolgung von massebehafteten Partikeln vorgeschlagen, mit dem Ziel, zum Beispiel Aussagen über die Verschmutzung von Fahrzeugen treffen zu können. Dabei ist allerdings einzuschränken, dass der numerische Aufwand bei der Abbildung wesentlich höher ist als bei masselosen Partikeln, weil anstelle einer Integration über ein Vektorfeld nun eine vollständige Bewegungsgleichung inklusive Kollisionserkennung gelöst werden muss.

Bahnlinien (Partikelverfolgung): Ausgehend von wohl definierten Saatpunkten wird hier der Weg der Partikel durch das zeitabhängige Strömungsfeld verfolgt (im Gegensatz zum stationären Vektorfeld bei Stromlinien).

Streichlinien: Hier werden in regelmäßig kurzen Zeitintervallen Partikel von einem festen Startpunkt aus injiziert und die zu einem Zeitpunkt gehörigen Orte der Partikel dargestellt. Streichlinie wie Bahnlinie entsprechen im Falle einer stationären Strömung der Stromlinie.

Vektorpfeile: Diese Methode ist in Abbildung 2.3 links oben dargestellt und dient in erster Linie zur Veranschaulichung der Richtung der Strömung. Weitere Informationen können durch eine Farbgebung der Vektoren oder durch eine Skalierung der Länge bezüglich einer skalaren Feldgröße hinzugefügt werden. Falls der Wertebereich der Feldgröße eine sehr grosse Skala abdeckt, ist die letzte Variante allerdings problematisch. Die Folge sind entweder zu lange Pfeile, die möglicherweise größer als die Ausmaße des Strömungsgebiets sind oder sehr kurze Pfeile, die man nicht mehr wahrnehmen kann. Vektorpfeile werden ähnlich wie die Startpunkte der Stromlinien entlang von Linien oder Flächen (auch gekrümmte Oberflächen) dargestellt.

Faltungsintegraltechnik (Line Integral Convolution, LIC): Diese Technik zeigt die Richtung im Strömungsfeld – ähnlich dem im Versuch angewandten Verfahren, Öl auf eine Oberfläche eines Hindernisses aufzutragen. Die Idee ist, eine Textur entlang der Richtung des Geschwindigkeitsfeldes zu verzerren. Dieses Verfahren wurde erstmals in [18] veröffentlicht und hinsichtlich Effizienz [101] oder Anordnung der Textur auf beliebig gekrümmten Oberflächen [105] kontinuierlich weiterentwickelt. Ein Beispiel für ein Faltungsintegral ist in Abbildung 2.3 (rechts unten) dargestellt.

Die Effizienz dieser Algorithmen hängt stark vom Typ der verwendeten Quelldaten ab. Im Falle von strukturierten Gittern ergibt sich ein erheblicher Vorteil, weil viele dieser Algorithmen stets die einem beliebigen Punkt P_i zugeordnete Zelle Z^* suchen und dann aus den Werten der Eckpunkte von Z^* den Wert der Daten an der Stelle P_i interpolieren müssen.
Selbstverständlich können die oben beschriebenen Techniken durch Beschriftungen und Legenden unterstützt werden. Bei Stromlinien ist es besonders schwierig, deren dreidimensionale Eigenschaften mit einer Projektion auf einen 2D Bildschirm zu erfassen. Hier kommen die Vorteile von Virtual Reality (siehe Abschnitt 2.2) besonders zum Tragen. Eine schnelle und gezielte Analyse der Strömung erfordert trotz dieser Vielfalt an leistungsfähigen Abbildungen weiterhin ein 'Gefühl' für das physikalische Problem.

2.2 Virtual Reality-Technologien

Die im vorherigen Abschnitt beschriebenen Techniken zur Visualisierung sind zunächst isoliert von der Hardware zur Ein- und Ausgabe zu betrachten. Das Ergebnis ist stets eine Pixeldarstel-

lung für eine beliebige Projektionsfläche. Ein Ingenieur hat meist einen Standard-Monitor oder einen Laptop an seinem Arbeitsplatz zur Verfügung. Die Folge ist, dass er ein dreidimensionales Produkt als Projektion auf einem 2D Bildschirm untersuchen muss. Der Eindruck räumlicher Tiefe fehlt. Der Benutzer muss somit eine gedankliche Transformation zwischen 2D und 3D vollziehen. Training und Erfahrung reichen aus, um bekannte Geometrien zu erfassen. Dazu ist allerdings ein gutes räumliches Vorstellungsvermögen erforderlich, das man bei einer Präsentation nicht von allen Teilnehmern voraussetzen kann. Die räumliche Struktur komplexer Wirbel oder Isoflächen sind auch vom Fachmann nur als echte 3D Projektion erkennbar. Hier setzen die Techniken von Virtual Reality (im Folgenden auch mit VR bezeichnet) an. Natürlich steckt – wie unten genauer erläutert wird – etwas mehr dahinter. Das postulierte Ziel von VR ist die vollständig realistische Nachbildung eines Objekts am Computer. Im Idealfall gehören dazu auch audio-visuelle und haptische (den Tastsinn betreffende) Eigenschaften.
Nach Pionierarbeiten Anfang der neunziger Jahre (z.B. "Der virtuelle Windkanal" [16]) wurde Virtual Reality zu einer Art Trend mit Arbeiten in einem breiten wissenschaftlichen Umfeld. Man könnte von einer Experimentierphase sprechen, in der in vielen Gebieten isoliert gezeigt wurde, dass VR einen Nutzen bringen kann. Heute setzen sich die Arbeiten durch, die einen Mehrwert in der Entwicklung eines Produkts bringen. Als Beispiel sei VR in der Architektur genannt: Auf der einen Seite schränkt VR als Werkzeug zum Entwurf die Kreativität des Architekten stark ein und kann sich deshalb nur sehr schwer durchsetzen. Andererseits verwendet man VR aber gerne als Medium, um dem Kunden ein entworfenes Bauwerk anschaulich zu präsentieren (Architektonischer Rundgang [13]).
Der folgende Abriss dient der besseren Erklärung des Einsatzes von Virtual Reality im Bereich der Steuerung und Auswertung strömungsmechanischer Berechnungsdaten in den Kapiteln 6 und 7. Weiterführende Ausführungen zu Virtual Reality findet man unter anderem in [112, 102, 98]. Neueste Trends werden auf den Konferenzen *IEEE Virtual Reality*, *ACM Symposium on Virtual Reality Software and Technology (VRST)* und *Virtual Reality and its Application in Industry (VRAI)* präsentiert. Zu Anwendungen im Bauwesen wird auf die Konferenz *Applied Virtual Reality in Engineering and Construction* verwiesen.

2.2.1 Begriff und Erscheinungsform

Der Begriff Virtual Reality hat eine lange Entwicklung hinter sich und wird in seinen Details noch immer unterschiedlich ausgelegt. Die folgenden Ausführungen stützen sich zu einem großen Teil auf Definitionen von Bryson [15].
Die Idee von VR wird erstmals 1965 in [104] formuliert. Frei übersetzt sagt Sutherland in einem zugehörigen Vortrag:

> Betrachten Sie es nicht als Bildschirm, sondern als Fenster mit Blick in eine virtuelle Welt. Die Herausforderung der Computergraphik ist, dass diese Welt real aussieht, real klingt und in Echtzeit auf Interaktionen reagiert und sich real anfühlt.

Diese Aussage trifft genau den Punkt. Allerdings ist 1989 Jaron Lanier der erste, der seine Arbeiten mit dem Begriff Virtual Reality bezeichnet und deshalb als der Begründer von VR gilt. Er arbeitete mit spezieller Hardware (siehe unten) zur Erzeugung eines drei-dimensionalen Eindrucks. In der Folgezeit hat man die eingesetzte Hardware-Technik direkt mit VR in Verbindung gebracht und VR als Erweiterung von Mensch-Maschine Schnittstellen betrachtet [6].
Es ist zwar offensichtlich, dass sich VR durch anspruchsvolle Hardware auszeichnet. Aus der obigen Definition würde aber folgen, dass selbst ein vorberechneter, computeranimierter Film schon als VR zu bezeichnen ist, solange man diesen nur auf einer entsprechenden Anlage betrachtet.

Nach einer präziseren Definition zeichnet sich VR vielmehr durch das Vorhandensein von diversen Komponenten aus. VR ist die Nachbildung einer echten oder imaginären Welt mit Methoden der Computertechnologie, in der man mit räumlich ausgeprägten Objekten interagieren kann. Im einzelnen sind die folgenden Komponenten unerlässlich:

3D–Projektion: Die räumliche Wahrnehmung ist das entscheidendste Kriterium von VR. Der wichtige Eindruck räumlicher Tiefe entsteht erst durch stereoskopische Darstellung. Dabei wird jeweils ein Bild für das linke und rechte Auge generiert und mit diversen Techniken wie Schutterbrillen (aktives Stereo) oder Polarisationsfilter (passives Stereo) so überlagert, dass das Gehirn einen Eindruck von räumlicher Tiefe erhält.

Tracking: Tracking hat die Wirkung, dass jede Bewegung zu einer Änderung des dargestellten Bildes führt. Besonders suggestiv ist diese Wirkung beim Head-Tracking (Verfolgung der Bewegung des Kopfes). Ohne Tracking wird sich ein Benutzer immer als externer Betrachter der Computersimulation fühlen.

Interaktivität: Dies ist der wesentliche Unterschied zu einer vorberechneten Animation. Man denke beispielsweise an einen Rundgang durch ein Gebäude. Das beginnt damit, dass man selbst entscheidet, an welche Orte innerhalb des Gebäudes man sich bewegt. Je mehr Möglichkeit zur Interaktion man hat (z.B. könnte man Möbel verschieben und so die Innenarchitektur untersuchen), desto realistischer ist das VR System. Allerdings wächst der Aufwand zum Erstellen und Betreiben des Systems mit steigender Interaktivität.

Echtzeitfähigkeit und Latenzfreiheit: Wünschenswert ist in diesem Zusammenhang, die simulierte Welt mit einer Rate von 20–30 Bilder pro Sekunde darzustellen (Stichwort: *Framerate*), andernfalls verliert die Darstellung an Überzeugungskraft und wirkt ermüdend. Prinzipiell sollte nach einer Interaktion durch den Benutzer die Veränderung in der simulierten Welt verzögerungsfrei erfolgen (frei von Latenz). Dennoch gibt es Anwendungen, in denen Zeitraffer oder Zeitlupe von Interesse sind. Letzteres wird in dieser Arbeit noch eine wesentliche Rolle spielen (siehe Kapitel 7).

Die folgenden Techniken werden ebenfalls mit VR in Bezug gebracht, sind aber aus heutiger Sicht noch keine zwingende Voraussetzung:

Immersion: Darunter versteht man, dass man sich voll zur virtuellen Welt zugehörig fühlt und sich vollständig von der realen Welt abschottet. Streng genommen bedeutet das, dass man nur noch die Virtuelle Welt wahrnimmt. Dieser Aspekt wird immer noch sehr strittig diskutiert. Eine Holobench (siehe Abbildung 2.6, rechts unten) ist demnach keine immersive Projektionsanlage, aber mächtig genug für 3D-Projektionen. Erfahrungen an der CAVE^{TM} (siehe Abbildung 2.6, rechts oben) zeigen, dass man bei immersiven Projektionen oft mehr damit beschäftigt ist, die simulierte Welt spielerisch zu erleben, anstatt diese gezielt zu erforschen und Aussagen über reale Fragestellungen zu treffen.

3D–Eingabegerät: Diese Werkzeuge erleichtern lediglich die Interaktion mit dem Modell in 3D und sind nicht zwingend für die räumliche Wahrnehmung notwendig.

Haptik: Der Bereich der Haptik bietet noch grosses Potential für Forschungen. Allerdings ist es sehr fraglich ob es in absehbarer Zeit gelingt, eine Hardware zu entwickeln, die den Tastsinn adäquat ansprechen kann.

Abgeleitet aus diesen Komponenten kann man somit die folgenden drei Erscheinungsformen von Virtual Reality definieren:

Non-Immersive-VR: Hier erfolgt die graphische Darstellung an einem gewöhnlichen Desktop-PC oder an Projektionsanlagen.

Immersive VR: Bei dieser Art von VR wird der Benutzer vollständig von der realen Welt abgeschottet.

Augmented Reality: Diese Form überlagert ein vom Computer generiertes Bild mit der realen Welt. In der Regel schaut der Anwender durch ein transparentes Display auf das reale Produkt.

2.2.2 Aufbau eines VR-Systems

Der klassische Aufbau eines Virtual Reality Systems gleicht dem Aufbau einer numerischen Simulation. Es besteht aus Eingabe, Simulation und Ausgabe. Eine mögliche, zugehörige Hardware-Ausstattung ist in Abbildung 2.4 vereinfacht dargestellt.

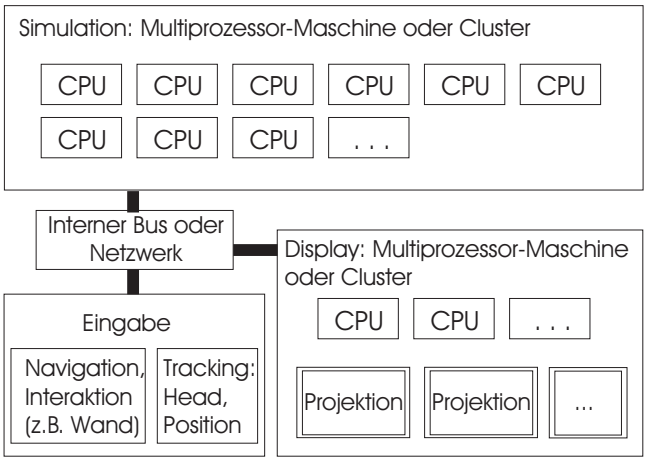

Abbildung 2.4: Vereinfachtes Schema eines Virtual Reality Systems

Eingabe: Diese Komponente nimmt die Eingaben des Benutzers auf. Dazu gehört das Tracking-System, die Navigation und sonstige Interaktionen.

Simulation: Bei komplexen Echtzeitsimulationen braucht man Multiprozessormaschinen oder einen Workstation-Cluster.

Ausgabe: Dieses Feld war bislang die Domäne von Silicon Graphics Workstations. Die Entwicklung geht allerdings ganz klar in die Richtung eines Clusters zur Visualisierung, bestehend aus PCs unter dem Betriebssystem Linux.

Lange Zeit verwendete man eine starke Multiprozessormaschine für das komplette System. Mittlerweile werden die zugehörigen Simulationen aber immer größer und komplexer. Die Konsequenz ist, dass die Simulation nicht mehr mit den für die Visualisierung zuständigen Prozessoren zu bewältigen ist. Der Datenbestand ist somit nur noch selten auf einem gemeinsam genutzten

Speicherbereich vorhanden. Die Kommunikation über Netzwerkprotokolle und ggf. die parallele Programmierung ist ein wesentlicher Bestandteil einer VR Applikation. Damit ist das System flexibel erweiterbar.

2.2.3 Hardware

Eingabe-Geräte

Diese Kategorie umfasst zunächst Geräte zur inkrementellen Eingabe[6]. Typische VR Eingabe-Geräte besitzen aber zusätzliche Freiheitsgrade, um direkt Rotationen oder eine Translation in z-Richtung einzugeben. Die folgende Auflistung beinhaltet die am weitesten verbreiteten Geräte.

Spacemaus: Diese Mausvariante wird auch sehr stark in Verbindung mit CAD-Systemen verwendet. Ein federnd gelagerter Steuerknopf erlaubt die Eingabe von sechs Freiheitsgraden.

Wand: Ein Wand[7] (siehe Abbildung 2.5, rechts oben) ist ein stabartiges Gerät mit einer Reihe von Knöpfen, das man frei im Raum bewegen kann.

Datenhandschuh: Mit Hilfe von Dehnmeßstreifen an den Rücken der Finger wird deren Stellung registriert. Datenhandschuhe verfügen gewöhnlich ebenfalls über ein Tracking-System.

Tracking–Systeme

Die optimale räumliche Wahrnehmung durch den Benutzer ist nur durch die Unterstützung einer räumlichen Projektion mit so genannten Tracking-Systemen möglich. Dabei wird die absolute Bewegung des Anwenders im Raum ermittelt (absolute Eingabe). Man unterscheidet dabei zwischen Position-Tracking und Head-Tracking.
Letzteres hat eine äußerst suggestive Wirkung auf den Benutzer, denn jede Bewegung des Kopfes führt zu einer entsprechenden Anpassung des Bildes. Es existieren magnetische, akustische, optische und mechanische Tracking-Systeme. Zur Bewertung eines Tracking-Systems werden die Kriterien Auflösung, Genauigkeit, Latenzzeit und Aktualisierungsrate herangezogen. Mechanische Tracking-Systeme liefern zwar sehr genaue Ergebnisse bei einer sehr hohen Aktualisierungsrate, schränken jedoch die Bewegungsfreiheit des Benutzers stark ein.

Systeme zur Visualisierung

Typische Ausgabe-Geräte zur Vermittlung des aktuellen Zustands sind:

Großbildprojektionen mit Stereobrille: Diese Systeme (siehe Abbildung 2.6, links oben) werden meistens für Diskussionen mit mehreren Benutzern eingesetzt. Deshalb wird oft auf das Head-Tracking eines Hauptanwenders verzichtet. Wichtig ist es hierbei, den Gegenstand der Diskussion in stereoskopischer Darstellung und gegebenenfalls in Lebensgröße zu betrachten. In jüngster Vergangenheit kommen auch transportable Projektionsanlagen wie das System CYKLOOP von Vircinity[8] zur Anwendung.

[6] Eine Desktop-Maus ist ein Beispiel für ein Eingabegerät zur inkrementellen Eingabe
[7] Wand – engl: Zauberstab
[8] http://www.vircinity.com

Abbildung 2.5: Beispiele für Eingabe-Geräte: Stylus (Wand, links) und Positions-Tracking (rechts)

HMD (Head-Mounted-Displays) und BOOM: Bei einem HMD (siehe Abbildung 2.6, links unten) ist das Display auf einem Helm montiert und direkt vor den Augen des Betrachters positioniert. Ein BOOM kann man sich wie ein Fernglas vorstellen, in das man hineinschaut. Die Eindruck von Immersion ist in beiden Systemen stark ausgeprägt.

CAVE^{TM}: Mit Hilfe von stereoskopischen Projektionen auf bis zu sechs Flächen eines würfelförmigen Raums (siehe Abbildung 2.6, rechts oben und [28]) wird ein hoher Grad an Immersion erzeugt.

Holobench: Mit zwei aufeinander senkrecht stehenden Projektionsflächen (siehe Abbildung 2.6, rechts unten) wird ein sehr guter Eindruck von räumlicher Tiefe erzeugt.

An dieser Stelle muss festgehalten werden, dass diese aufwendigen Anlagen sicher auch weiterhin keinen Massenmarkt abdecken werden. In diesem Umfeld werden sich voraussichtlich vielmehr Desktop Bildschirme mit Stereo-Brillen aus dem Bereich der Computerspiele durchsetzen.

2.2.4 Datenhaltung: VRML und Szenegraph

VRML (Virtual Reality Modelling Language) ist eine plattformunabhängige Sprache zur Beschreibung virtueller Welten und basiert auf einem Szenegraph (siehe unten). Die Entwicklung von VRML geht auf das Dateiformat Open Inventor und dem zugehörigen, gleichnamigen Toolkit von SGI zur Erstellung von Szenegraphen zurück. Die Version VRML 97 ist schließlich als ISO-Standard verabschiedet worden. VRML ist mittlerweile ein sehr weit verbreitetes Datenformat und Plugins für Browser sind frei verfügbar. Gängige CAD-Systeme und Programme zur Modellierung in 3D unterstützen den Import und Export von VRML. Dieser Umstand und der etwas irreführende Name sind ausschlaggebend dafür, dass VRML oft mit Virtual Reality gleichgesetzt wird. Entscheidend aber ist vielmehr die Datenstruktur eines Szenegraphen, der mit VRML beschrieben werden kann.

Der Szenegraph

Ein Szenegraph ist ein gerichteter, azyklischer Graph *(DAG – Directed Acyclic Graph)*, der sich aus die Szene beschreibenden Knoten *(Nodes)* zusammensetzt. Diese können Felder *(Fields)*

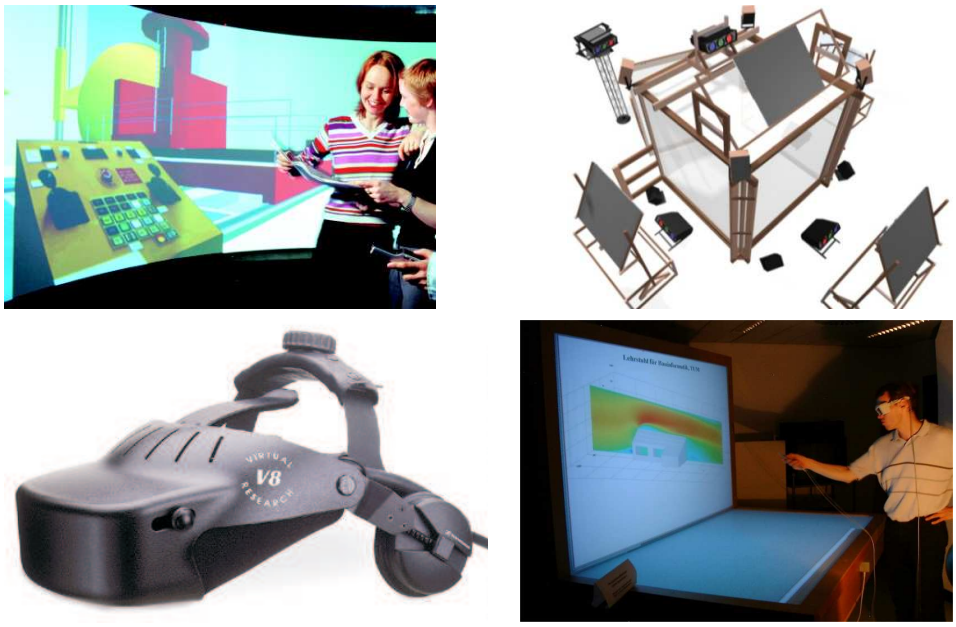

Abbildung 2.6: Display Systeme für Virtual Reality: Panorama-Wand (links oben), CAVE^{TM} (rechts oben), Head Mounted Display (unten links), Holobench (unten rechts)

enthalten, um Eigenschaften zu speichern. Knoten sind selbst Instanzen einer Klasse. So gesehen ist der Szenegraph eine objektorientierte Beschreibung einer 3D-Welt. Der Szenegraph beschreibt immer nur, was gezeichnet werden soll, nicht aber wie. In Abhängigkeit der Leistungsfähigkeit der eingesetzten Maschine werden dabei intern (in den Funktionen der Klassen) Routinen einer prozeduralen Low-Level Graphik-Bibliothek wie OpenGL verwendet.

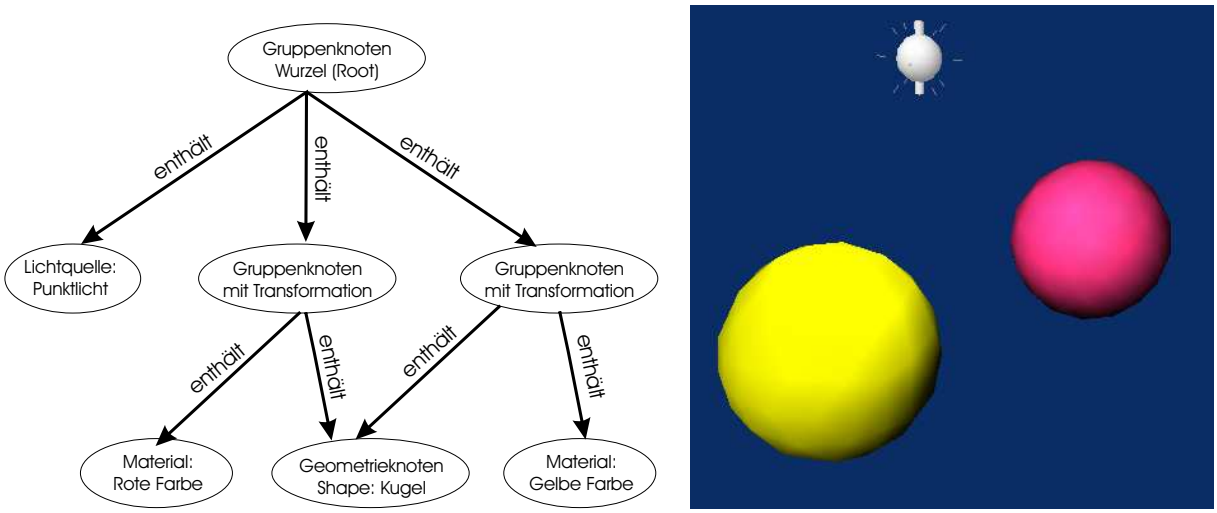

Abbildung 2.7: Beispiel eines einfachen Szenegraphen

Es ist keine zwingende Voraussetzung, dass eine VR-Anwendung einen Szenegraphen als interne Datenstruktur verwendet. Die folgenden Eigenschaften und die Vielfalt der Knoten zeigt allerdings die Flexibilität und Erweiterbarkeit, sowie die Möglichkeiten, Interaktionen des Benutzers zu verwerten:

Geometrie-Knoten: Es existieren eine Reihe von Basis-Objekten wie Zylinder, Kugel oder Hexaeder sowie Formen, die durch eine Menge von Polygonen beschrieben sind. Die letztere Gruppe ist nichts anderes als ein Facetten-Modell.

Gruppenknoten: Mit Hilfe von Gruppenknoten wird eine Hierarchie im Szenegraph aufgebaut. Zusätzlich können die Gruppenknoten noch eine Transformationsvorschrift enthalten. Diese Struktur wird zur Optimierung des Renderings genutzt, weil stets nur die Teile des Szenegraphen neu gezeichnet werden, die sich tatsächlich verändert haben (Render Caching).

Wiederverwendbarkeit durch Querreferenzierung: Dieses Prinzip ist im Beispiel von Abbildung 2.7 erkennbar. Die Geometrie der Kugel wird nur einmal erzeugt, aber in zwei verschiedenen Zweigen des Graphen verwendet. Der Unterschied der Kugeln liegt in der Transformation sowie der zugeordneten Farbe.

Lichtquellen: Typischerweise gehören dazu gerichtete Lichtquellen, Punktlichter und Spot-Lichter.

Interaktionen: Mit Hilfe von *Events* und *Sensoren* werden Interaktionen umgesetzt. Der Programmierer gibt System-Events (z.B. eine Mausbewegung) an den Szenegraphen weiter. Innerhalb des Szenegraphen befinden sich dann Sensoren, die auf korrespondierende Events reagieren und eine zugewiesene Aktion durchführen (Beispiele: Transformation eines Objekts oder Starten einer Animation).

2.2.5 Einsatzbereich und ausgewählte Projekte

Die Weiterentwicklung von VR wird zunehmend durch die Anwendungen und deren Mehrwert geprägt sein [12]. Die folgende Auflistung zeigt eine Reihe von ausgewählten Anwendungsbereichen von VR.

Architektur und Städteplanung: Ein Bauvorhaben kann bereits vor der Erstellung in Form eines Virtuellen Modells anschaulich präsentiert werden.

Militärtechnik: Das Militär ist einer der ersten und größten Anwender von VR. Der "Close Combat Tactical Trainer" der US-Navy ist eine der größten VR Anlagen weltweit.

Virtuelle Prototypen: Durch die immer kürzer werdenden Produktzyklen nutzen vor allem Automobilhersteller die Möglichkeit, Gestaltung und Funktionalität am virtuellen Produkt zu untersuchen.

Training: Dazu gehört u.a. die Ausbildung von Astronauten bei der NASA oder die Schulung von Chirurgen am virtuellen Modell eines Patienten

Scientific Visualization: Der virtuelle Windkanal [16] gilt heute noch als Referenzprojekt für VR und demonstriert eindrucksvoll die Stärke von VR zur Erforschung von Ergebnissen numerischer Simulationen.

Entertainment: Die Spieleindustrie gilt oftmals als eine Art Motor für neue Entwicklungen, insbesondere im Bereich der Hardware.

2.3 Computational Steering

2.3.1 Allgemeines und Definition

Computational Steering ist definiert als *interaktive Kontrolle eines Prozesses am Computer* [73]. Diese Definition hat vorerst noch nichts mit einer numerischen Simulation zu tun. Beispielsweise ist jeder Debugger ein Computational–Steering System: Während der Ausführung eines Programms macht man Beobachtungen, verändert Parameter und bekommt so schnell wie möglich die Veränderungen angezeigt. Auf der anderen Seite gibt es noch die inverse Steuerung eines Programms: Hier stellt man ein gewünschtes Ergebnis ein und erhält vom System die Werte der Parameter des Prozesses.
Wissenschaftliche Berechnungen werden gewöhnlich in einer Art Stapelverarbeitung durchgeführt. Man bereitet Daten vor, startet eine Berechnung und erhält zum Schluss ein Ergebnis. Entspricht dieses nicht den Erwartungen, simuliert man neu. Es ist offensichtlich, dass eine interaktive Steuerung vor allem für aufwendige Berechnungen einen entscheidenden Mehrwert liefert – wie bereits 1988 von Brooks [11] gefordert.
Mulder [73] unterscheidet drei verschiedene Arten des Computational Steering:

- Erforschung eines konkreten Problems durch Verändern von Parametern. Ein Beispiel wäre etwa die Umströmung einer Kugel. Als Parameter möchte man etwa mit der Anströmgeschwindigkeit, der Viskosität oder anderen Stellgrößen experimentieren. Dieses Gebiet wird mit *model exploration* bezeichnet.

- Verändern des verwendeten Algorithmus in der Simulation. Ein Beispiel ist, unterschiedliche Lösungsverfahren miteinander zu vergleichen.

- Optimierung der Performance und Fehlersuche (Debugging). In parallelen Applikationen könnte man mit der Optimierung der Lastverteilung auf die Prozesse experimentieren.

Der wesentliche Mehrwert einer interaktiven Simulation kann wie folgt zusammengefasst werden:

- Erhöhung der Produktivität, weil die Zeit zwischen der Veränderung von Parametern und der Analyse des Ergebnisses verkürzt wird.

- Ein effizientes Mittel der Überzeugung ist die Strategie, die Auswirkung einer Idee bzw. eines Vorschlags unmittelbar zu demonstrieren (so genannte *what-if-analysis* [41]).

- Die Zusammenhänge zwischen einen Effekt und der Ursache werden schneller sichtbar.

2.3.2 Systemaufbau und Komponenten

Es existieren mittlerweile eine Reihe von Systemen zur interaktiven Steuerung von Prozessen (im Folgenden wird statt Prozess von einer numerischen Simulation gesprochen). Dabei kann man zwischen speziell auf eine Anwendung zugeschnittenen Systemen und allgemeiner anwendbaren Systemen unterscheiden. Anhand des typischen Aufbaus (siehe Abbildung 2.8) der letztgenannten Systeme kann man sich sehr gut die Vorgänge und den Datenfluss in einer interaktiven Steuerung veranschaulichen.
Der Anwender auf der linken Seite interagiert mit einer Benutzerschnittstelle zur Aufnahme der Eingaben und Darstellung der Ergebnisse. Diese Schnittstelle muss bei weitem keine Virtual Reality Umgebung sein, dennoch ist eine qualitativ hochwertige Visualisierung für eine effiziente Steuerung unerlässlich. Ganz rechts steht die Applikation, die sich meist auf einem anderen

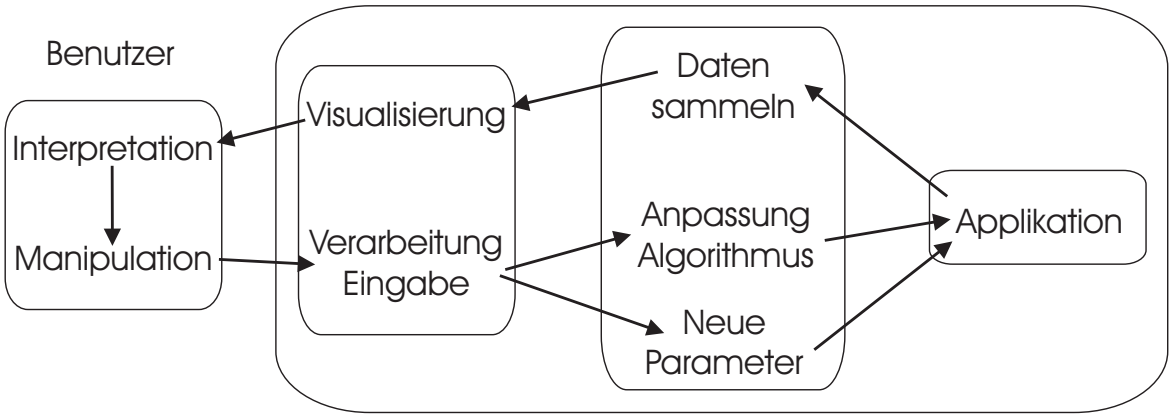

Abbildung 2.8: Computational Steering – Systemaufbau und Datenfluss

Rechner befindet. Das kann z.B. ein Supercomputer sein und die Anwendung selbst kann auch verteilt arbeiten. Hier steckt die größte Herausforderung der Kommunikationsschicht (zweite Komponente von rechts) in interaktiven Simulationen: Verteilte Prozesse müssen mit zufällig vom Benutzer ausgelösten Ereignissen synchronisiert werden. In numerischen Simulationen sind die zu transferierenden Datenmengen sehr gross und letztlich auch ein zentrales Problem. In [64] wird nämlich die Bandbreite (vor allem in Netzwerken) als der Flaschenhals für interaktive Steuerungen identifiziert. Die Entwicklung im Bereich der Rechenleistung und der Computergraphik enteilt zunehmend dem Fortschritt bei den Übertragungstechniken.
Unter den allgemeinen Systemen zum Computational Steering existieren eine Reihe von kommerziellen und akademischen Entwicklungen (z.B. COVISE[9] oder SCIRun[10]), die sich aber im Umfang stark unterscheiden. Man wird zunächst versuchen wollen, die eigene Simulation in ein solches System zu integrieren. Um festzustellen, ob dies möglich ist, sollte man sich über die folgenden Fragen Gedanken machen:

- Welche Ansprüche hat man an die Benutzeroberfläche und die Visualisierung. Ist eine Anbindung an Virtual Reality Umgebungen möglich?
- Wie groß ist der Programmieraufwand, um eine vorhandene Simulation für eine interaktive Steuerung zugänglich zu machen?
- Ist das System zugänglich und welche Plattformen werden unterstützt?
- Wird ein Eingriff in eine verteilte Simulation unterstützt?
- Welche Datenstrukturen werden unterstützt? Viele der meisten existierenden Systeme arbeiten nur mit Gittern in der Form von vollen Matrizen.
- Welche zusätzlichen Features bietet das System? Existiert eine Umgebung zur visuellen Programmierung oder gibt es einen Code-Generator?

Weiterführende Literatur zu Computational Steering kann z.B. [111] entnommen werden.

[9] http://www.vircinity.com
[10] www.sci.utah.edu

Kapitel 3

Numerische Strömungssimulationen nach der Lattice-Boltzmann Methode und paralleles Rechnen

In diesem Kapitel werden numerische Methoden der Strömungsmechanik (CFD) vorgestellt. Der Schwerpunkt wird dabei auf die in dieser Arbeit verwendete Lattice-Boltzmann Methode gelegt. Um den hohen Ansprüchen von CFD an die Rechenleistung gezielt entgegenzuwirken, verwendet man Methoden des parallelen Rechnens in Verbindung mit Supercomputing, die am Ende dieses Kapitels kurz umrissen werden.

3.1 Numerische Simulation der Strömungsmechanik mit der Lattice-Boltzmann Methode

3.1.1 Allgemeines zur numerischen Strömungssimulation

Die numerische Simulationen strömungsmechanischer Probleme stellt einen wichtigen Schwerpunkt der Forschung dar. Obwohl verschiedenste kommerzielle Programmpakete (z.B. CFX[1] oder STAR-CD[2]) im produktiven Einsatz sind, ist noch großer Entwicklungsbedarf vorhanden, der nur durch interdisziplinäre Forschung zwischen Naturwissenschaftlern, Ingenieuren, Mathematikern und Informatikern bewältigt werden kann. Die folgenden Ausführungen wollen einen kurzen Überblick über dieses Gebiet geben, dessen Komplexität veranschaulichen und verdeutlichen, warum es nicht bzw. nur sehr schwer mit der Forderung einer Simulation in Echtzeit (siehe Abschnitt 2.2) gemäß den Kriterien von Virtual Reality zu vereinbaren ist. Anschließend wird die Lattice-Boltzmann Methode als neuartiges Lösungsverfahren vorgestellt, wobei vor allem die Motivation der Verwendung dieser Methode im Umfeld einer interaktiven Strömungssimulation (siehe Kapitel 7) erklärt wird. Eine ausführliche Abhandlung zu CFD findet man z.B. in [34, 43] oder im Kurzlehrgang NUMET am Lehrstuhl für Strömungsmechanik[3] der Universität Erlangen. Die Notwendigkeit dieser Simulationen in der industriellen Produktentwicklung ist aufgrund der folgenden Vorteile gegenüber dem Experiment offensichtlich:

Kosten und Aufwand: Windkanalversuche sind oftmals sehr zeitaufwendige und ggf. teure Experimente. In der Simulation kann man die Geometrie durch Modifikation des CAD-Modells verändern, während man im Experiment meistens ein neues Modell bauen müsste. Weiterhin gibt es auch eine Reihe von Vorgängen, die technisch nur äußerst schwierig im

[1] http://www.cfx.aeat.com
[2] http://www.cd-adapco.com
[3] http://www.lstm.uni-erlangen.de

Windkanal zu realisieren sind. Beispielsweise benötigt man zur Bestimmung der Strömungsverhältnisse um ein fahrendes Auto einen beweglichen Untergrund [31] – eine Apparatur, die nur in den wenigsten Windkanälen zu finden ist.

Upscaling-Problematik: Trägheitskräfte, Zähigkeitskräfte und Schwerkräfte treten in jeder Strömung auf. Die Verhältnisse dieser Kräfte untereinander und in Verbindung mit den geometrischen Abmessungen charakterisieren die Strömung und sind durch so genannte Kennzahlen quantifiziert. Es ist jedoch nicht möglich, diese Kennzahlen unabhängig voneinander zu skalieren, was im Klartext bedeutet, dass die Strömung im Windkanal niemals die reale Strömung identisch abbilden kann. Als Beispiel sei die Reynoldszahl Re (Verhältnis der Trägheitskraft zur Zähigkeitskraft) der Froudezahl Fr (Verhältnis der Trägheitskraft zur Gravitationskraft) gegenübergestellt.

$$Re = \frac{u_\infty \cdot l}{\nu} \qquad \text{und} \qquad Fr = \frac{u_\infty}{\sqrt{g \cdot l}}$$

u_∞	Geschwindigkeit
l	charakteristische Länge
g	Erdbeschleunigung
ν	Viskosität

Man erkennt leicht, dass man bei einer Verkürzung der Länge die Geschwindigkeit erhöhen muss, um die gleiche Reynoldszahl zu erhalten, während man diese jedoch verringern müsste, um die Froudezahl konstant zu halten.

Messungen: Trotz weit entwickelter Messverfahren wird man immer nur eine begrenzte Anzahl von Sonden im Windkanal unterbringen können und in vielen Fällen sind Messungen erst gar nicht möglich.

In [31] wird dies klar bestätigt, und gleichzeitig darauf hingewiesen, dass die Simulation als eine Unterstützung des Experiments anzusehen ist. Darauf aufbauend ist die in Abbildung 3.1 dargestellte Strategie der Auslegung eines Produkts im Hinblick auf die strömungsmechanischen Eigenschaften motiviert:

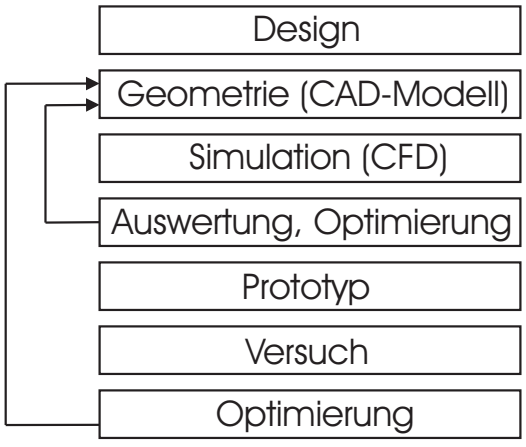

Abbildung 3.1: CFD als Ergänzung zum Experiment in der Produktauslegung

Ausgehend vom Design und dem CAD Modell werden mehrere Simulationszyklen durchgeführt, wobei eine Optimierung zu einer Änderung des CAD-Modells führt. Konnte man ein hinsichtlich Funktionalität und Effizienz zufrieden stellendes Ergebnis erzielen, wird ein Modell im Windkanal experimentell untersucht. Stimmen die Windkanalversuche nun mit den Simulationen hinreichend gut überein, ist der Auslegungszyklus abgeschlossen, ansonsten wird das Modell optimiert und neue Simulationen werden durchgeführt. Der strategische Ansatz dieser Arbeit (siehe Abschnitt 5.3) wird diese Methodik erweitern.
Die bisher diskutierten Aspekte erscheinen zunächst sehr überzeugend, dennoch muss man die Schwierigkeiten einer numerischen Simulation hinterfragen, indem man das Vorgehen genauer betrachtet.
Die reale Physik einer Strömung wird mit einem System partieller Differentialgleichungen, den Erhaltungssätzen bzw. Transportgleichungen beschrieben.

$$\underbrace{\frac{\partial \rho \Phi}{\partial t}}_{\text{instationärer Term}} + \underbrace{\xi_\Phi \nabla \cdot (\rho \vec{u} \Phi)}_{\text{konvektiver Term}} = \underbrace{\nabla \cdot (\Gamma_\Phi \nabla \Phi)}_{\text{diffusiver Term}} + \underbrace{Q_\Phi}_{\text{Quellterm}} \tag{3.1}$$

Die Transportgröße Φ kann zum Beispiel ein Stoff, der Impuls oder die Temperatur sein. Inkompressible Strömungen mit konstanter Dichte

$$\rho(T) = \rho_0 = const.$$

lassen sich durch die Transportgleichungen des Impulses (den Navier-Stokes Gleichungen (3.2)) sowie der Masse (die Kontinuitätsgleichung (3.3)) beschreiben:

$$\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_j} = -\frac{1}{\rho_0} \cdot \frac{\partial p}{\partial x_i} + \nu \cdot \frac{\partial^2 u_i}{\partial x_j^2} \tag{3.2}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.3}$$

mit

u_k	Geschwindigkeit
t	Zeit
ρ_0	Dichte
ν	Viskosität
p	Druck

Nun wird der kontinuierliche Raum in eine diskrete Form eines Gitters (Finite Differenzen Methode FDM) oder in diskrete Teilvolumina (Finite Volumen Methode FVM) bzw. Elemente (Finite Element Methode FEM) übergeführt. Dabei werden die partiellen Differentialgleichungen durch ein System von algebraischen Gleichungen approximiert. Die Lösung liefert dann Näherungswerte an diskreten Punkten $\vec{x}_i$. Zwischenwerte werden durch Interpolation (FDM, FVM) oder mit Hilfe von Ansatzfunktionen (FEM) ermittelt. Hierbei entstehen Fehler als Folge der Approximation durch ein diskretes Modell, Rundungsfehler und Ungenauigkeiten bei der iterativen Lösung des Gleichungssystems.
Die gängigen Simulationsprogramme verwenden die FVM. Beliebig geformte Ränder können damit zwar gut approximiert werden, dennoch ist die Generierung dieser Netze ein sehr zeitraubender Prozess, der oft eine sehr starke Benutzerinteraktion erfordert. Eine effiziente Verwaltung dieser Netze benötigt zusätzlich einen erheblichen Überbau in den zugehörigen Datenstrukturen.

Wesentlich entscheidender allerdings ist die Eigenschaft der Navier-Stokes Gleichungen (3.2) als nichtlineares, gekoppeltes System mit variablen Koeffizienten. Die Berechnung ist sehr aufwendig, weshalb in der Praxis für spezielle Problemstellungen vereinfachte Formen dieser Gleichungen angewendet werden. Für die Gleichungen (3.2 und 3.3) wurde bereits eine derartige Annahme getroffen, denn die Dichte ist im allgemeinen vom Druck und der Temperatur abhängig. Zur Lösung dieser Problemstellung muss dann noch eine Energieerhaltungsgleichung mit einbezogen werden. Als weitere Vereinfachungen seien die Euler-Strömung, die Potentialströmung, die Boussinesq-Approximation oder die Grenzschicht–Approximation genannt.
Schwieriger zu handhaben sind turbulente Strömungen, die zeitabhängig, unregelmäßig, mischungsintensiv, dissipativ, drehbehaftet und dreidimensional [114] ausgeprägt sind. Infolge dieser Eigenschaften sind die charakteristischen Längen der kleinsten Wirbel l_k sehr gering und verhalten sich gemäß Kolmogorov zu den größten Wirbeln L mit

$$L/l_k = Re^{3/4}.$$

Eine direkte Simulation muss diese Wirbel durch das zugrunde liegende Gitter voll auflösen. Die dafür notwendige Anzahl der Gitterpunkte N_g, der Zeitschritte N_t und der Rechenoperationen N_{op} in Abhängigkeit von der Reynoldszahl Re kann wie folgt angeschrieben werden (siehe [114]):

$$N_g = Re^{9/4}; \qquad N_t = Re^{3/4}; \qquad N_{op} = Re^{11/4} \tag{3.4}$$

Viele Strömungen von praktischem Interesse bewegen sich im Bereich von $Re = \mathcal{O}(10^4)$ (Strömungen in Innenräumen) und höher (z.B. Umströmung eines Autos – $Re = \mathcal{O}(10^6)$). Die direkte numerische Simulation einer Strömung mit $Re = 5 \cdot 10^4$ erfordert nach Gleichung (3.4) bereits theoretisch ca. $3,74 \cdot 10^{10}$ Gitterpunkte und selbst auf den schnellsten Rechnern[4] der Welt mehrere Wochen Rechenzeit. Außerdem müsste man in diesen Überlegungen noch die Frage des notwendigen Speicherplatzes erörtern. Damit scheidet eine direkte numerische Simulation (DNS) in vielen Fällen aus.
An dieser Stelle setzt die Turbulenzmodellierung an, die heutzutage unerlässlicher Bestandteil der Simulation von Strömungen mit hohen Reynolds-Zahlen ist. Alle Größen in der Transportgleichung (3.1) werden in einen Hauptanteil $\bar{u}_i$ und kleine Schwankungen u_i' aufgeteilt, so dass z.B. für die Geschwindigkeit u_i gilt

$$u_i = \bar{u}_i + u_i'.$$

Der Hauptanteil kann nun durch statistische, zeitliche oder räumliche Mittelung gebildet werden. Das bedeutet, dass die kleinen Skalen durch Filterung unterdrückt werden und nur deren Einfluss auf die größeren Skalen modelliert wird. Wendet man einen derartigen Filter formal auf die Navier–Stokes Gleichungen an (siehe [43], S.159f), erhält man die so genannten *Reynoldsgleichungen*

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial(\bar{u}_i \bar{u}_j)}{\partial x_j} = -\frac{1}{\rho_0} \cdot \frac{\partial \bar{p}}{\partial x_i} + \nu \cdot \frac{\partial^2 \bar{u}_i}{\partial x_j^2} \underbrace{- \frac{\partial(\overline{u_i' u_j'})}{\partial x_j}}_{\text{I}} \tag{3.5}$$

für den Hauptanteil $\bar{u}_i$, in der nun der zusätzliche Term I mit den unbekannten *Reynoldsspannungen* $R(u_i') = -\rho_0(\overline{u_i' u_j'})$ auftaucht. Die neuen unbekannten Größen benötigen jetzt weitere

[4] Die Hitachi SR 8000 am LRZ München (http://www.lrz.de) hat eine Spitzenleistung von ca. $2 \cdot 10^9$ Operationen pro Sekunde.

Gleichungen. Diese basieren auf Näherungen und Hypothesen aufgrund empirischer Informationen und bilden das *Turbulenzmodell*. Es existiert eine Vielzahl von Turbulenzmodellen, die man z.B in [88] und der dort angegebenen Literatur entnehmen kann. Hier wird die Large–Eddy Methode kurz angerissen, weil diese sich sehr gut für die Lattice-Boltzmann Methode eignet (siehe nächster Abschnitt). Die Large-Eddy Methode ist ein volumengemittelter Ansatz, wobei die Hauptspannung als *Grobstruktur* mit großen energiereichen Wirbeln interpretiert wird. Die nicht aufgelösten Anteile bilden die *Feinstruktur* der kleinen Wirbel, die dissipativ wirken. Dieser Effekt des Entziehen von Energie wird als Scheinzähigkeit interpretiert. Dieser Ansatz wird später im Rahmen der Lattice-Boltzmann Methode vorgestellt.

3.1.2 Die Lattice-Boltzmann Methode

Trotz der großen Fortschritte in der Entwicklung numerischer Diskretisierungsverfahren stellt die Lösung der Navier–Stokes Gleichungen weiterhin eine der größten Herausforderungen im wissenschaftlichen Rechnen dar. Neben Methoden zu deren direkter Lösung wurden in den letzten zwei Jahrzehnten alternative Verfahren entwickelt (siehe Literatur in [1]), welche die Charakteristika eines Vielteilchensystem direkt durch ein künstliches, im Computer existierendes System diskreter Partikel abbilden. Diese Teilchen bewegen sich zwischen den Knoten eines regelmäßigen Gitters und kollidieren an dessen Knoten. Studien der Molekulardynamik haben gezeigt, dass bereits $10^4 - 10^8$ Teilchen eine sehr gute Annäherung an die hydrodynamischen Grundgleichungen liefern. Aus dieser Idee der so genannten Gittergase entstand schließlich die Lattice-Boltzmann Methode, die in diesem Abschnitt kurz beschrieben wird. Darauf folgt eine Zusammenfassung der Vorteile des Verfahrens im Rahmen der Anwendung in der Planung von Bauwerken sowie innerhalb von interaktiven Simulationen. Außerdem wird die Basis geschaffen, um die in Abschnitt 7.2 beschriebene Implementierung nachzuvollziehen. Eine detaillierte Beschreibung des Verfahrens findet man in [103, 57] und der dort referenzierten Literatur.
Wesentlicher Nachteil der Verwendung explizit diskreter Partikel bei der Lattice-Gas Methode ist das Rauschen, insbesondere des Druckfeldes. Deshalb ging man dazu über, Teilchenaufenthaltswahrscheinlichkeiten zu verwenden [69], deren Dynamik mit der Boltzmann–Gleichung

$$\underbrace{\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{x}}}_{\text{I}} + \underbrace{\vec{F} \cdot \frac{\partial f}{\partial \vec{\xi}}}_{\text{II}} = \Omega(f) \tag{3.6}$$

aus der statistischen Physik beschrieben werden kann.

$\vec{\xi}$	(kontinuierliche) mikroskopische Geschwindigkeit
$f = f(t, \vec{x}, \vec{\xi})$	Wahrscheinlichkeit, ein Teilchen zum Zeitpunkt t, am Ort $\vec{x}$ mit der Geschwindigkeit $\vec{\xi}$ anzutreffen
$\Omega(f)$	Kollisionsoperator (Modellierung der Interaktion der Teilchen)
I	advektiver Teilchentransport mit der Geschwindigkeit $\vec{\xi}$
II	Einfluss der äußeren Kräfte

Bedingt durch den Kollisionsoperator ist Gleichung 3.6 allgemein eine komplizierte Integro–Differentialgleichung und somit aufwändiger zu lösen als die Navier–Stokes Gleichungen. Der Ansatz von Bhatnagar, Gross und Krook [7] geht jedoch davon aus, dass sich das Gas als Kontinuum verhält und die Abweichungen vom Gleichgewichtszustand sehr gering sind. Der Kollisionsoperator vereinfacht sich zu

$$\Omega = -\frac{1}{\tau} \cdot (f - f^{(0)}). \tag{3.7}$$

$f^{(0)}$ ist die sich einstellende Gleichgewichtsverteilung, eine lokale Maxwellverteilung

τ ist die Relaxationszeit, mit der die Annäherung an den Gleichgewichtszustand beschrieben wird

Nun folgt die Diskretisierung von (3.6) im mikroskopischen Geschwindigkeitsraum. Hierzu wählt man einen Satz bestimmter Verteilungen aus, die sich mit diskreten Geschwindigkeiten $\vec{e}_i$ im Raum bewegen. Die Verteilungsfunktion $f = f(t, \vec{x}, \vec{\xi})$ wird durch die Werte $f(t, \vec{x}, \vec{e}_i) = f_i(t, \vec{x})$ an N ausgewählten Kollokationspunkten dargestellt, wodurch man die diskrete Gitter–Boltzmann Gleichung

$$\frac{\partial f_i}{\partial t} + \vec{e}_i \cdot \frac{\partial f_i}{\partial \vec{x}} = -\frac{1}{\tau} \cdot (f - f^{(0)}) \qquad i = 0, \ldots, N-1 \tag{3.8}$$

erhält. Der Satz der diskreten Geschwindigkeiten $\{\vec{e_i}\}$ spannt eine Einheitszelle eines kartesischen Gitters auf. Hierbei existieren mehrere Modelle für die Anordnung der Kollokationspunkte, die von Qian [82] mit der Notation *DkQb* beschrieben werden. k entspricht der Dimension des Raumes und b der Anzahl der Kollokationspunkte. Die Richtungen der Geschwindigkeiten $\{\vec{e_i}\}$ sind für die in Abschnitt 7.2 verwendeten Modelle *D3Q15*

$$\{\vec{e_i}\} = c \begin{pmatrix} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{pmatrix} \tag{3.9}$$

und *D3Q19*

$$\{\vec{e_i}\} = c \begin{pmatrix} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{pmatrix} \tag{3.10}$$

Die Gleichungen 3.8 werden nun in Raum und Zeit mit finiten Differenzen diskretisiert (typischerweise setzt man $c = \Delta x = \Delta t = 1$) und es ergibt sich die Lattice–Boltzmann Gleichung

$$f_i(t + \Delta t, \vec{x} + \vec{e}_i \Delta t) = f_i(t, \vec{x}) - \frac{\Delta t}{\tau}(f_i(t, \vec{x}) - f_i^{(0)}(t, \vec{x})). \tag{3.11}$$

Die Maxwell'sche Gleichgewichtsverteilung $f_i^{(0)}$ wird als Taylor-Reihe bis zur zweiten Ordnung entwickelt [82]:

$$f_i^{(0)}(t, \vec{x}) \cong t_i \rho \left(1 + \frac{e_{i\alpha} u_\alpha}{c_s^2} + \frac{u_\alpha u_\beta}{2c_s^2} \left(\frac{e_{i\alpha} e_{i\beta}}{c_s^2} - \delta_{\alpha\beta}\right)\right) \tag{3.12}$$

Die frei wählbaren Parameter t_i werden mit Hilfe der Chapman-Enskog Analyse [20] so bestimmt, dass die Lattice–Boltzmann Gleichung (3.11) unter Verwendung von Gleichung (3.12) die Physik der Navier–Stokes Gleichungen (3.2 und 3.3) beschreibt. Die Momente der Verteilungsfunktionen f_i geben dabei die makroskopischen Größen Dichte

$$\rho = \sum_i f_i(t, \vec{x}) \tag{3.13}$$

und Geschwindigkeit

$$\vec{u} = \frac{1}{\rho} \sum_i \vec{e_i} f_i(t, \vec{x}) \tag{3.14}$$

wieder. Weiterhin gilt: c_s ist eine Modellkonstante (die Schallgeschwindigkeit) und ist hier:

$$c_s = \frac{1}{\sqrt{3}} c = \frac{1}{\sqrt{3}}$$

Der Parameter t_i ist für die später verwendeten Modelle *D3Q15*

$$t_i = \left(\tfrac{2}{9} \;\; \tfrac{1}{9} \;\; \tfrac{1}{9} \;\; \tfrac{1}{9} \;\; \tfrac{1}{9} \;\; \tfrac{1}{9} \;\; \tfrac{1}{9} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \;\; \tfrac{1}{72} \right) \tag{3.15}$$

und *D3Q19*

$$t_i = \left(\tfrac{1}{3} \;\; \tfrac{1}{18} \;\; \tfrac{1}{18} \;\; \tfrac{1}{18} \;\; \tfrac{1}{18} \;\; \tfrac{1}{18} \;\; \tfrac{1}{18} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \;\; \tfrac{1}{36} \right) \tag{3.16}$$

Eine detaillierte Herleitung dieser Parameter findet man beispielsweise in [108]. Dort wird auch die korrigierte Relaxationszeit

$$\tau = \frac{6\nu + 1}{2} \tag{3.17}$$

angegeben, die sich infolge vernachlässigter Terme in der Chapman-Enskog Analyse ergibt. Die Zustandsgleichung für den Druck p ist angegeben mit

$$p = \frac{1}{3} \rho. \tag{3.18}$$

Damit ist ein numerisches Schema gegeben, das äquivalent zur direkten Simulation der inkompressiblen Navier-Stokes Gleichungen ist. Zur Lösung der Gleichung (3.11) wird ein einfaches kartesisches Gitter verwendet. Die Hindernisse innerhalb des Strömungsfeldes werden mit einer so genannten *marker and cell* Beschreibung festgehalten, d.h. es existiert ein Flag-Feld, mit dem für jeden Knoten identifiziert wird, ob es sich um ein Fluid, eine Struktur oder eine bestimmte Randbedingung handelt. Jeder Zeitschritt wird bei der Implementierung in zwei wesentliche Teile gegliedert:

- Die *Kollision* ist die Ermittlung der neuen Verteilungsfunktionen. Dazu wird die Gleichung (3.11) für $t + \Delta t$, jedoch am gleichen Ort $\vec{x}$ berechnet:

$$f_i(t + \Delta t, \vec{x}) = f_i(t, \vec{x}) + \Omega_i(\vec{x}, t) \tag{3.19}$$

- Die *Propagation* ist das Weiterleiten (bzw. die Advektion) der Verteilungsfunktionen zum nächsten Nachbarn gemäß

$$f_i(t + \Delta t, \vec{x} + \vec{e_i} \Delta t) = f_i(t + \Delta t, \vec{x}). \tag{3.20}$$

Nun sind aber viele praxisrelevante Strömungen stark turbulent und damit durch sehr hohe Reynoldszahlen gekennzeichnet. Es stellen sich die gleichen Probleme bezüglich Rechenzeit und Speicherbedarf wie bei klassischen Navier–Stokes Lösern. Das in Abschnitt 3.1.1 angeschnittene Large-Eddy Turbulenzmodell eignet sich besonders für die Lattice-Boltzmann Methode und

wurde dafür erstmalig von Hou [51] angewendet. Der Einfluss der kleinen, nicht aufgelösten, dissipativen Skalen wird als zusätzliche Wirbelviskosität ν_T modelliert und die gesamte Viskosität ν_0 ergibt:

$$\nu_{total} = \nu_0 + \nu_T = \frac{\tau_0 - 1}{6} + \frac{\tau_T}{3} \tag{3.21}$$

Smagorinsky's Ansatz [99] für die Wirbelviskosität ist

$$\nu_T = c_s^2 \mid \epsilon_{\alpha\beta}^{LB} \mid, \tag{3.22}$$

mit dem Dehnungstensor

$$\epsilon_{\alpha\beta}^{LB} = -\frac{3}{2}\frac{1}{\rho(\tau_0 + \tau_T)} S_{\alpha\beta}^{LB}, \tag{3.23}$$

basierend auf dem Spannungstensor

$$S_{\alpha\beta}^{LB} = (\frac{1}{2\tau_0} - 1) \sum_i e_{i\alpha} e_{i\beta} (f_i - f_i^{(0)}). \tag{3.24}$$

Die lokale Relaxationszeit der Turbulenz τ_T ergibt sich schließlich zu

$$\tau_T = \frac{\sqrt{\tau_0^2 + \frac{18 c_s^2 Q^{1/2}}{\rho}} - \tau_0}{2} \tag{3.25}$$
$$Q = \parallel \epsilon_{\alpha\beta}^{LB} \parallel$$

Bemerkenswert ist hierbei, dass sich der Spannungstensor $S_{\alpha\beta}^{LB}$ als algebraischer Term[5] lokaler Knotengrößen schreiben lässt, womit sich erhebliche algorithmische und numerische Vorteile ergeben. Ein Beispiel für eine Berechnung mit einem Large-Eddy Modell befindet sich in [57] sowie in Abschnitt 5.3.
Der Ansatz des Kollisionsoperators nach Gleichung (3.7) setzt eine kleine Knudsen-Zahl ε voraus, die allerdings bei Simulationen mit wenigen Gitterpunkten bzw. größeren Gitterweiten nicht zwangsläufig gewährleistet ist. Der Kollisionsoperator (3.7) bildet einen Satz von Verteilungen $\{f_i\}$ für jeden Gitterknoten auf sich selbst unter Verwendung eines einzigen Relaxationsparameters τ ab.
In diesem Kontext bringt das so genannte Momentenmodell [62, 30] eine erhebliche Verbesserung der Stabilität. Die grundlegende Idee basiert darauf, die Verteilungen für die Kollision erst in den Momentenraum $\vec{\Xi}$ zu transformieren. Nun werden anstelle der Verteilungen die Momente relaxiert, allerdings mit individuell für die jeweiligen Momente optimierten Relaxationsraten. Einige Momente Ξ_i entsprechen der Dichte und den Flusskomponenten und werden durch die Relaxation im Momentenraum nicht verändert. Anschließend werden die Momente wieder in den Raum der Verteilungen zurücktransformiert.
Die Werte der Transformationsmatrix $\hat{M}$ und der Gleichgewichtsmomente $\Xi_i^{(0)}$ sowie die zugehörigen Modellparameter p_i können [30] entnommen werden.
Die Vorteile der Lattice-Boltzmann Methode werden hier nochmal kurz zusammengefasst:

- Durch die *marker and cell* Beschreibung der Geometrie kann die Gittergenerierung fast vollständig automatisiert werden. Komplexe CAD-Objekte werden durch Voxelierung in eine diskrete Beschreibung überführt.

[5] Bei Navier–Stokes Verfahren müssen dazu Gradienten gebildet werden

- Solange die Verteilungsfunktionen im Speicher auf vollen Matrizen gehalten werden, ist das Verfahren leicht zu implementieren. Die Aufteilung in parallele Prozesse (siehe 3.2) lässt sich sehr effizient umsetzen, weil nur der Vorgang der Propagation Informationen mit lediglich dem nächsten Nachbarn austauscht.
- Ein Large-Eddy Turbulenzmodell kann sehr effizient implementiert werden, weil zur Bestimmung des Spannungstensors keine Gradienten ermittelt werden müssen.
- Der Druck kann direkt als Summe der Primärvariablen ermittelt werden. Es muss also keine Poisson-Gleichung gelöst werden.

Trotz der offensichtlichen Vorteile des expliziten Verfahrens in Raum und Zeit sowie der einfachen kartesischen Gitter beschäftigen sich neuere Entwicklungen mit impliziten Lösungsverfahren [107], Multigrid-Verfahren [108] oder adaptiven Verfahren [25]. Interessant im Umfeld dieser Arbeit sind auch die Entwicklungen im Bereich der Strömungen mit Wärmeübertragung [35].

3.1.3 Einsatzbereiche

Das Anwendungsspektrum der Lattice-Boltzmann Methode ist sehr weitreichend und vielfältig. Im Bereich der kommerziellen Solver findet man den klar dominierenden Marktführer Exa[6] PowerFLOW^{TM}, der u.a. in den Gebieten Automobilbau, Luftfahrt, Öl und Chemie eingesetzt wird. Beachtenswert ist dabei, dass nahezu alle großen deutschen Automobilhersteller PowerFLOW^{TM} zur Simulation von Außenströmungen verwenden. Allerdings bleibt festzuhalten, dass sich der industrielle Einsatz noch auf inkompressible Strömungen beschränkt. Kompressible, temperaturbehaftete Strömungen dagegen sind Gegenstand aktueller Forschungen (siehe oben). Die Habilitationsschrift von Krafczyk [57] zeigt die hervorragende Anwendbarkeit der Lattice-Boltzmann Methode für die Problemstellungen des Bauingenieurwesen. Dazu zählt in erster Linie die Handhabung von sehr komplexen Geometrien (z.B. können sogar poröse Medien bewältigt werden). Weiterhin wird die Eignung von Lattice-Boltzmann für eine effiziente Implementierung mit parallelen Algorithmen und für hierarchische Datenstrukturen verdeutlicht. Numerische Beispiele zeigen die Tauglichkeit für die Simulation turbulenter Strömungen mit Large-Eddy-Modellen. Die dort geschilderten Ideen und Resultate sind eine entscheidende Grundlage und Motivation für die Verwendung der Lattice-Boltzmann Methode in dieser Arbeit.
Sehr gute Ergebnisse im High–Performance Computing erzielten z.B. Nie [76] für das anspruchsvolle Problem der Rayleigh–Taylor Instabilität oder d'Humières für ein dreidimensionales Driven–Cavity Problem mit diagonaler Anströmrichtung [30]. Im letzteren Fall wurde gezeigt, dass bei einer Reynoldszahl von Re=2000 und unter Verwendung der Momentenmethode die Ergebnisse (selbst bei leicht gröberer Auflösung des diskreten Gitters) im Vergleich zu einer mit dem kommerziellen Solver FLUENT[7] ermittelten Referenzlösung gleichwertig waren.

3.2 Paralleles Rechnen und Supercomputing

Im vorhergehenden Abschnitt wurde der enorme Speicherbedarf und Aufwand der Berechnung bei einer CFD Simulation deutlich. Eine Lattice-Boltzmann Simulation mit dem Modell D3Q19 und der Momentenmethode benötigt die folgende Anzahl an Rechenoperationen (ca. 150 Rechenoperationen pro Gitterknoten) pro Zeitschritt bzw. den folgenden Speicherplatz:

[6] http://www.exa.com
[7] http://www.fluent.com

Re	Gitterpunkte	Rechenoperationen	Speicherplatz
10^4	$1,00 \cdot 10^9$	$1,50 \cdot 10^{11}$	111,76 GB
10^5	$1,78 \cdot 10^{11}$	$2,67 \cdot 10^{13}$	19873,82 GB

Tabelle 3.1: Speicherbedarf einer direkten numerischen Lattice-Boltzmann Simulation

Obwohl man natürlich durch die Verwendung von Turbulenzmodellen Abhilfe schaffen kann, ist klar, dass eine gewöhnliche Berechnung trotz der rasanten Entwicklung der Prozessoren schnell an ihre Grenzen stößt. In dieser Arbeit steht demgegenüber noch die Idee der interaktiven Simulation an. Eine mögliche Lösung sind parallele Rechensysteme wie Multiprozessor-Maschinen oder Workstation–Cluster. Die Rechenlast bzw. der Speicherplatz wird also aufgeteilt.

3.2.1 Rechnerarchitekturen und Supercomputing

Die klassische Aufteilung nach Flynn [37] beschreibt vier verschiedene Rechnerarchitekturen, die sich nach der Menge der Instruktionen und der Menge der Datenströme unterteilen. Heute sind nur noch die SISD-Architektur[8] sowie die MIMD-Architektur[9] von Bedeutung. Die schnellsten Rechner der Welt (siehe Top 500 Rangliste[10]) werden als Supercomputer bezeichnet und sind heutzutage fast ausschließlich MIMD-Systeme. Dazu gehören in erster Linie MPP-Systeme (MPP steht für *massively-parallel-processor*) und SMP-Rechner (SMP steht für *symetric-multi-processing* – mit einem gemeinsamen Speicherbereich). Neuerdings gibt es unter den Top 500 verstärkt Cluster-Systeme (verhalten sich wie MPP-Systeme). Der Trend unter den schnellsten Rechnern geht aber in die Richtung von hybriden Systemen wie dem Bundeshöchstleistungsrechner Hitachi SR 8000. Das System ist in der obersten Ebene eine MPP-Architektur mit 168 Knoten. Ein Knoten wiederum ist ein SMP-System mit 8 Prozessoren und einem gemeinsamen Speicherbereich. Der Einzelprozessor wiederum besitzt einen sehr speziellen und eingeschränkten Befehlssatz (RISC-Prozessor) und kann während eines Prozessortakts mehrere Operationen gleichzeitig ausführen (Pseudo-Vektorisierung).

3.2.2 Ebenen der Parallelisierung und explizite Parallelisierungskonzepte

Als Folge der oben beschriebenen Architektur der heutigen Supercomputer kann ein Programm nun in mehreren Ebenen parallel implementiert sein.
Abbildung 3.2 stellt diese Ebenen und deren Beziehung zueinander schematisch dar:

- Die innerste Ebene ist die so genannte *Vektorisierung*. Diese wird vom Programmierer durch Compiler-Anweisungen aktiviert. Die Parallelisierung geschieht auf der Ebene der Instruktionen des Prozessors. Beispielsweise besteht eine Floating-Point Operation aus 6 Teilschritten, die in jedem Takt des Prozessors gleichzeitig ausgeführt werden können. Die Verarbeitung einer Schleife im Quellcode erfolgt sozusagen wie bei einem Fließband. Ein Beispiel ist die Addition zweier Vektoren. Während für das $i - te$ Element im Vektor die Operanden bereitgestellt werden, werden für das Element $i + 1$ gleichzeitig die Exponenten subtrahiert, für das Element $i + 2$ werden die Mantissen angepasst, etc.

[8] single instruction stream, single data stream – typischer von-Neumann-Rechner, d.h. eine Instruktion arbeitet mit einem Datenstrom

[9] multiple instruction stream, multiple data stream – mehrere Instruktionen bearbeiten mehrere Datenströme

[10] http://www.top500.org

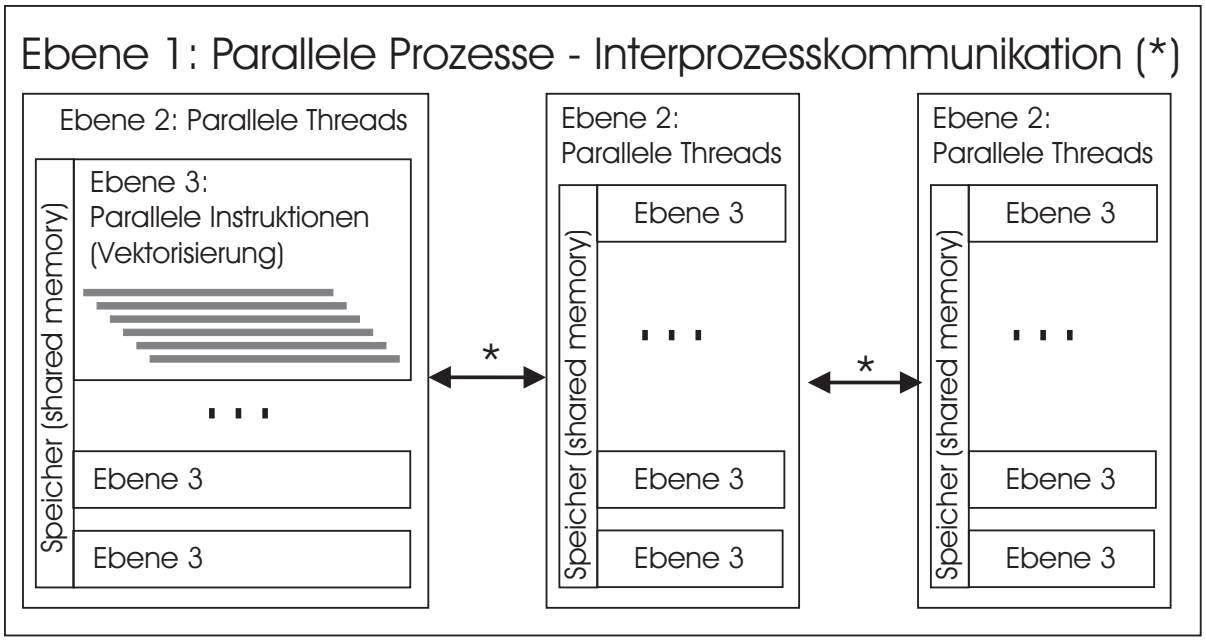

Abbildung 3.2: Ebenen der parallelen Verarbeitung

- Die zweite Ebene ist eine parallele Verarbeitung des Programmes unter Verwendung eines gemeinsamen Speicherbereichs (*shared memory*). Dies kann entweder implizit durch Compiler-Anweisungen erfolgen oder explizit durch die Erzeugung von so genannten *Threads* (Programmfäden), die dann durch die Implementierung aufeinander abgestimmt werden müssen (Synchronisierung).

- Die oberste Ebene ist die Ebene der parallelen Prozesse mit verteiltem Speicher, das wohl bekannteste Konzept der parallelen Programmierung. Die Parallelität wird ausschließlich explizit vom Entwickler beschrieben. Im Gegensatz zu den parallelen Threads müssen hier Daten zwischen den getrennten Adressräumen der Prozesse durch eine Interprozesskommunikation ausgetauscht werden. Dazu verwendet man so genannte Message Passing Systeme wie den am weitesten verbreiteten Standard MPI[11]. Der Anteil der Kommunikation sollte möglichst gering gehalten werden, weil er (besonders auf Workstation-Cluster mit einer Ethernet–Verbindung) schnell zum Flaschenhals des parallelen Programms werden kann.

Die Verteilung der Aufgaben durch explizite Parallelisierung kann auf sehr unterschiedliche Art durchgeführt werden:

- Master-Slave, Farm-Prinzip: Die Gesamtaufgabe wird in unabhängige Teile zerlegt (z.B. Parameterstudien).

- Datenparallelisierung: Jeder Prozessor bearbeitet einen Teilbereich des Datenraums.

- Algorithmische Aufteilung: Jeder Prozessor ist für einen bestimmten Teil der Berechnung zuständig und bearbeitet dabei den kompletten Datensatz. Ein Beispiel findet man in der interaktiven Computergraphik. Während ein Prozessor den Datenbestand verändert, sorgt der zweite für das Rendering.

[11] http://www.mpi-forum.org

Für Simulationen nach der Lattice-Boltzmann Methode wird eine Aufteilung des Datenraums in geometrische Teilgebiete durchgeführt. Diesen Vorgang nennt man Gebietszerlegung. Eine optimale Aufteilung berücksichtigt die folgenden Kriterien:

- Ausgeglichene Verteilung der Aufgaben (*load balancing*)
- Minimierung des Anteils der Kommunikation.

Man unterscheidet zwischen zwei verschiedenen Verfahren der Gebietszerlegung:

- Gebietszerlegungen auf der Grundlage von Graphenmodellen sind algorithmisch aufwendig und benötigen viel Speicher, liefern jedoch die besten Ergebnisse hinsichtlich Lastverteilung und Minimierung der Kommunikation.
- Geometrische Gebietszerleger sind dagegen einfacher aufgebaut und schneller. Nachteilig ist jedoch, dass diese in der Regel nur ein Kriterium der Gebietszerlegung (siehe oben) optimal erfüllen.

Ein detaillierter Überblick zu den Techniken der Gebietszerlegung kann z.B. [19] entnommen werden.

3.2.3 Effizienzanalysen und Grenzen

Die Effizienz eines parallelen Programmes muss generell an einer Reihe von Messgrößen bewertet werden.
Die Beschleunigung der Ausführung ist der *Speed-Up*

$$S(p) = \frac{T(1)}{T(p)} \tag{3.26}$$

p	Anzahl der Prozesse
$T(1)$	Ausführungszeit des Programms mit einem Prozess
$T(p)$	Ausführungszeit des Programms mit p Prozessen

Theoretisch gilt $0 \leq S(p) \leq p$. Oftmals wird jedoch eine Überschreitung des maximalen Wertes beobachtet. Die Ursache dafür sind Sekundäreffekte (z.B. verbesserter Zugriff auf den Cache) aufgrund der Aufteilung des Problems in kleinere Teilbereiche.
Der normalisierte Speed-Up wird als *parallele Effizienz*

$$E(p) = \frac{S(p)}{p} \tag{3.27}$$

bezeichnet. Theoretisch gilt $0 \leq E(p) \leq 1$. Bei diesen Betrachtungen wurde die gleiche algorithmische Struktur für das serielle sowie das parallele Programm vorausgesetzt. Der *absolute Speed-Up* bzw. die *absolute parallele Effizienz* vergleicht dagegen den schnellsten seriellen Algorithmus mit dem schnellsten parallelen Algorithmus. Ein weiteres Gütemaß ist der *Scale-Up*

$$S^*(p) = \frac{\text{T(1) mit (l Freiheitsgrade)}}{\text{T(p) mit (p} \cdot \text{l Freiheitsgrade)}}, \tag{3.28}$$

der unterschiedliche Problemgrößen berücksichtigt.

Einschränkend ist festzuhalten, dass man durch die Erhöhung der Anzahl der Prozesse die Beschleunigung nicht bis in das Unendliche treiben kann, selbst wenn man eine optimale Kommunikation und Synchronisierung erreicht. Grundsätzlich setzt sich nämlich jedes Problem aus einem inhärent sequentiellen Anteil a und einem potentiell parallelisierbaren Anteil $(1-a)$ zusammen. Das Amdahlsche Gesetz quantifiziert den damit erreichbaren maximalen Speedup:

$$S_{max}(p,a) = \frac{T(1)}{T(p)} = \frac{T(1)}{a \cdot T(1) + \frac{1-a}{p} \cdot T(1)} = \frac{p}{pa-(1-a)} \tag{3.29}$$

$$\lim_{p \to \infty} S_{max}(p,a) = \frac{1}{a} \tag{3.30}$$

Das bedeutet, dass man bei einem seriellen Anteil von nur 1% und 100 Prozessen einen maximalen Speed-Up von 50 erreicht.
Wenn man zwischen den einzelnen Prozessen der parallelen Applikation noch Daten austauschen muss, dann wird man ohnehin nie in der Lage sein, die optimalen Werte der Leistungssteigerung zu erzielen. Für Probleme des wissenschaftlich–technischen Höchstleistungsrechnens ist das Volumen und die Frequenz der zu transferierenden Daten in der Regel ein limitierender Faktor. Die Zeit der Kommunikation zum Austausch von Daten zwischen parallelen Prozessen ist

$$t_{Kommunikation}[s] = t_{Latenz}[s] + \frac{\text{Transfer-Volumen [MBit]}}{\text{Bandbreite [MBit/s]}}. \tag{3.31}$$

Die Bandbreite[12] und die Latenzzeit[13] werden sehr stark von der verwendeten Hardware beeinflusst. Bei der Entwicklung eines parallelen Algorithmus sollte man diesen Umstand stets im Auge behalten, denn gerade bei einer signifikant hohen Latenzzeit sollte man stets bemüht sein, mit geringer Frequenz (dafür aber mit größerem Datenvolumen pro Transfer) zu kommunizieren.

[12] Die Bandbreite ist der Datendurchsatz durch ein Netzwerk.
[13] Die Latenzzeit ist die Zeit für die Initialisierung bevor ein Datentransfer beginnt.

Kapitel 4

Analyse der Anforderungen an die Steuerung und Auswertung von Strömungssimulation

Zur Umsetzung der in Kapitel 1 postulierten Zielsetzung, numerische Simulationen effizient in die Planungsphase von Bauwerken zu integrieren, wurden in den Abschnitten 2 und 3 zunächst die notwendigen Grundlagen der einzusetzenden Methoden und Technologien diskutiert. Im nächsten Schritt werden nun die Anforderungen hinsichtlich der zu implementierenden Software-Komponenten ermittelt.
Zunächst wird der Einsatzbereich, die Planung von Bauwerken beleuchtet und deren Besonderheiten bzw. Schwierigkeiten identifiziert. Dies betrifft die Prozesse und die Beteiligten sowie den Gegenstand der Planung und dessen Charakteristika. Daraus werden die funktionalen Bestandteile an die zu entwickelnden Software-Lösungen definiert. Dazu gehören in erster Linie die Schnittstellen, der Rechenkern, die Visualisierung und die Bedienung (Benutzeroberflächen).

4.1 Einsatzbereich: Integration in den Planungsprozess im Bauwesen

Grundsätzlich sollte die Simulations-Software flexibel hinsichtlich des Anwendungsbereichs sein. Allerdings sind gerade im Umfeld von Bauwerken Besonderheiten zu berücksichtigen, die hier kurz umrissen werden.

Planungsphasen

Bauprojekte werden in Phasen[1] abgewickelt. Diese Phasen sind in Deutschland fest durch eine Rechtsordnung, der *Honorarordnung für Architekten und Ingenieure (HOAI)* festgelegt. Die HOAI beinhaltet insgesamt 9 Phasen von der Projektvorbereitung bis zum Projektabschluss. Die technischen Aspekte der Planung werden fast ausschließlich bereits in der sehr frühen HOAI-Phase 2 (Vorplanung) bearbeitet. Am Ende der Vorplanung sind die folgenden Aspekte festgelegt:

- Gebäudegeometrie und Gestaltungsrahmen
- Funktionale Zusammenhänge
- Konstruktive Systeme
- Haustechnik

[1] Eine Phase ist innerhalb eines Projekts ein zeitlicher Abschnitt, der sachlich gegenüber anderen Abschnitten getrennt ist [4].

In dieser Phase wird somit das Objekt definiert und die Kosten stehen bereits weitgehend fest. Anschließend beginnt die Bauvorbereitung und dann folgt die Ausführung. Fehlentscheidungen in dieser Planungsphase führen zwangsläufig zu gravierenden und damit auch sehr kostspieligen Konsequenzen. Bis einmal eine Nachbesserung in die Wege geleitet ist, sind oft schon weitere unabhängige Planungsphasen angelaufen oder bereits abgelaufen.

Planungsbeteiligte

Während der kompletten Planung, insbesondere bei der Vorplanung sind eine Vielzahl von Entscheidungsträgern beteiligt. Im Gegensatz zu anderen Branchen wie etwa dem Automobilbau ist dieser Personenkreis durch seine heterogene Ausprägung ausgezeichnet. Diese Vielfalt umfasst neben unterschiedlichen Fachrichtungen mit entsprechenden Ansichten auf das bauliche Objekt auch persönliche und wirtschaftliche Interessen. Im Einzelnen sind dies:

- Bauherr – ist oftmals durch ein externes Projektmangement vertreten.
- Architekt / leitender Ingenieur – im Falle von Hochbauten obliegt ihm die fachliche Koordination und die Integration der Beiträge anderer Fachleute in die Gesamtplanung.
- Fachplaner (z.B. Tragwerksplaner, Haustechnik)
- Sonderfachleute (z.B. Spezialisten Bauphysik)
- Berater
- Behörden (z.B. Brandschutzbehörde, Gewerbeaufsicht)

Experten für Strömungssimulationen reihen sich meistens in die Gruppe der Sonderfachleute oder der Berater ein. Es ist aber auch denkbar, dass sie Untersuchungen im Auftrag von Behörden durchführen.
Wesentlich problematischer als nur die Vielfalt der Beteiligten ist die Tatsache, dass sie während der Vorplanung ein hohes Mass an Interaktion untereinander haben. Gleichwohl sind diverse Planer von den Zwischenergebnissen anderer Beteiligter abhängig. Beispielsweise braucht ein Statiker Geometriedaten aus dem Entwurf des Architekten. Somit sind sehr häufig Abstimmungen nötig, die durch den Architekten koordiniert werden. Im Idealfall läuft der Austausch von Informationen gemäß Abbildung 4.1 ab.

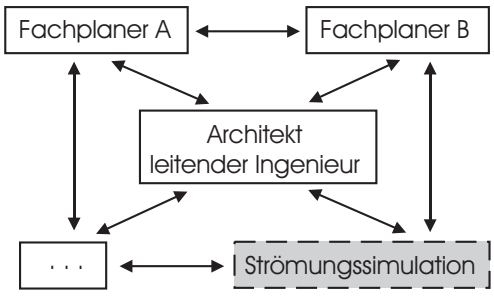

Abbildung 4.1: Informationsaustausch in der Vorplanung

Detailliertere Ausführungen zu den Planungsphasen und den Planungsbeteiligten befinden sich u.a. in [42].

Planungsgegenstand

Den größten Einfluss auf die Gestaltung der numerischen Simulation hat der Gegenstand der Planung. Die geometrischen und physikalischen Verhältnisse sind dabei sehr vielfältig und äußerst komplex.

Geometrische Ausprägung

Die Geometrie eines Bauwerks ist einerseits sehr komplex und andererseits meist ein Unikat. Eine Vereinheitlichung des Planungsgegenstandes scheidet also aus, denn das Spektrum umfasst Hallenbauten, Hochhäuser, Industriebauten (z.B. Kühltürme), Brücken sowie Funk- und Fernsehtürme. Dieser Aspekt wird durch die Gegenüberstellung unterschiedlichster Bauwerke in Abbildung 4.2 illustriert. Das Design reicht von filigranen Strukturen (Mitte) bis zu massiven Bauwerken (rechts).

Abbildung 4.2: Veranschaulichung der komplexen Geometrie und der Vielfalt von Bauwerken

Die Folge ist, dass man nur sehr schwer in der Lage sein wird, vollautomatisch – sozusagen per Knopfdruck – aus einem CAD-Modell ein allgemeines Netz für eine numerische Berechnung zu erzeugen. Der Idealfall wäre, eine diskrete Beschreibung in Form eines vollständig aus Hexaedern zusammengesetzten Netzes zu erhalten. Eine Software, die diese Aufgabe möglichst schnell erledigt, existiert bisher noch nicht [86]. Weiterhin kann im allgemeinen nicht davon ausgegangen werden, dass man im Verlauf der Planung ein vollständig korrektes CAD-Modell des Bauwerkes mit geschlossenen Oberflächen erhält. Diese Fragestellung wird in [25, 91] behandelt und soll hier nicht weiter verfolgt werden.

Physikalische Problemstellungen

Die physikalischen Eigenschaften von Strömungen im Bauingenieurwesen sind ebenfalls sehr unterschiedlich. Dazu zählen:

- Strömungen um Bauwerke infolge von Windlasten. Diese Strömungen sind hoch turbulent und können nur mit ausgereiften Turbulenzmodellen (siehe Abschnitt 3.1.1) effizient numerisch berechnet werden.

- Strömungen in Innenräumen als Folge von Windlasten oder durch Temperatur bzw. Strahlung getrieben. Häufiger technischer Hintergrund ist die Auslegung von Belüftungs- bzw. Klimaanlagen oder Schutzmaßnahmen gegen Sonneneinstrahlung.

- Strömungen im Baugrund. Diese Strömungen sind in der Regel Multi–Phasen–Strömungen [96].

- Spezielle Probleme im Wasserbau. Hierzu zählen Strömungen über Wehranlagen oder Turbinenströmungen in Wasserkraftwerken. Diese Strömungen sind meist hoch turbulent.

Aus der komplexen Geometrie ergibt sich zusätzlich eine breite Palette von Strömungsprofilen. Das Auftreten von aerodynamisch günstigen Körpern, gemeinsam mit stumpfen Körpern bzw. scharfen Abrisskanten ist keine Ausnahme, sondern der Regelfall.
Die weiteren Ausführungen werden sich auf Windströmungen in und um Bauwerke beschränken. Die technisch relevanten Strömungen sind dann in jedem Fall dreidimensional ausgeprägt, spielen sich im turbulenten Bereich ab und umströmen komplexe geometrische Strukturen. Die typischen Reynoldszahlen [81] befinden sich in den folgenden Bereichen:

Strömungen in Innenräumen	$Re = \mathcal{O}(10^4)$
Außenströmungen	$Re = \mathcal{O}(10^6 - 10^8)$

Beispielsweise ist die Reynoldszahl für ein Gebäude mit sieben Stockwerken bei einer Windgeschwindigkeit von 20 m/s gleich $Re = 2.6 \cdot 10^7$. Eine ausführliche Behandlung der Thematik der Strömungen in und um Bauwerke findet man in [100, 81] und der dort referenzierten Literatur.

4.2 Funktionsumfang

Aus der Beschreibung des Planungsablaufs können jetzt eine Reihe von Anforderungen definiert werden. Oberste Priorität ist es dabei, die Produktivität in der Planung zu erhöhen. Dies impliziert, dass ein Berechnungsingenieur mit Hilfe seiner Simulation schnell auf Veränderungen des Planungsgegenstandes reagieren können muss und dabei möglichst viele Alternativen vorschlagen sollte. Es wurde weiterhin die Vielfalt der Beteiligten an der Planung angeschnitten, die man in diesem Kontext allesamt überzeugen muss.

Durchgängigkeit – Schnittstellen

Die drei Teile einer Simulation (Preprocessing, Berechnung und Postprocessing) wurden bereits in Abschnitt 1.1 angesprochen. Zwischen diesen Komponenten müssen Daten transportiert und entsprechend transformiert werden. Diese beiden Aspekte sind die Kernprobleme von Schnittstellen.

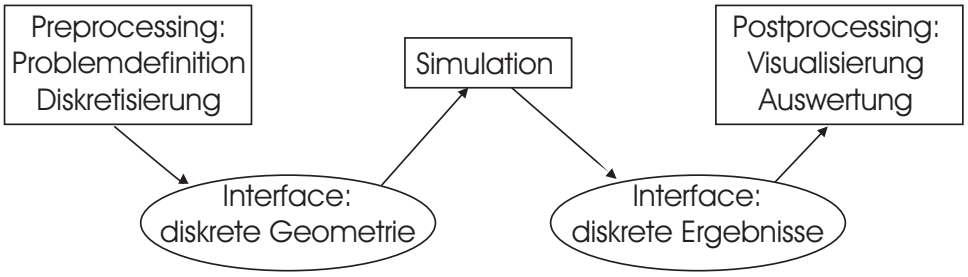

Abbildung 4.3: Schnittstellen innerhalb einer numerischen Simulation

Abbildung 4.3 skizziert die Schnittstellen innerhalb einer numerischen Simulation. Die erste Schnittstelle entsteht zwischen dem Preprocessing und der Simulation. Die Probleme beschränken sich hier nur auf Fehler bei der Übertragung der Daten, denn die elementare Funktion der Gittergenerierung ist die Erzeugung von angepassten Eingabedaten für die Simulation.
Auf der anderen Seite kann es notwendig sein, die Ergebnisdaten der Simulation für die Visualisierung nachzubearbeiten. Notwendig wird dies aufgrund der Zielsetzung einer interaktiven Simulation: Dazu ist es wichtig, einen möglichst kleinen Datensatz für die Visualisierung bereitzustellen. Die relevanten Eigenschaften der berechneten Strömung müssen jedoch erhalten bleiben. Abhilfe schafft hier eine Datenreduktion mit Qualitätskontrolle. Die in Abschnitt 2.1 vorgestellten Algorithmen zur Visualisierung arbeiten am effizientesten mit auf strukturierten Gittern vorliegenden Daten. Liegt bei der Simulation eine andere Art der Diskretisierung vor und möchte man diesen Vorteil nutzen, so ist eine Konvertierung nötig, die gegebenenfalls einen Verlust der Qualität der Ergebnisdaten zur Folge hat.
Der günstigste Fall liegt natürlich vor, wenn Preprocessing, Simulation und Postprocessing mit den gleichen Datenstrukturen arbeiten. Werden diese Komponenten in einer einzigen Applikation zur interaktiven Steuerung einer Simulation vereinigt, so müssen keine Daten mehr manuell (d.h. durch Kopieren innerhalb oder zwischen Dateisystemen) ausgetauscht werden, denn dieser Vorgang ist dann Bestandteil des Programms selbst. Die einzelnen Komponenten der modular aufgebauten interaktiven Simulation befinden sich dann auf verschiedenen Rechnern und Daten werden zwar innerhalb der Applikation, aber letztlich über eine Netzwerkverbindung ausgetauscht. Dadurch gewinnt die bereits angesprochene Reduktion der Daten wieder an Bedeutung.

Gittergenerierung und Problemdefinition

Die Überführung der als Volumen modellierten Geometrie in eine diskrete Form wird als der Flaschenhals in der Kette der numerischen Simulation angesehen [39]. Die vollständige Automatisierung dieses Prozesses ist vor allem für unstrukturierte Netze aus Hexaedern bisher nicht gelungen. Eine interaktive Simulation muss hier zumindest Lösungen enthalten, mit denen der manuelle Eingriff durch einen Benutzer bis auf ein Minimum reduziert wird. Auf jeden Fall, muss der gewählte Ansatz in Verbindung mit dem eingesetzten Berechnungskern entwickelt und auf diesen abgestimmt werden. Abbildung 4.4 stellt die möglichen Gittertypen gegenüber. Die Schwierigkeit, das Gitter automatisch zu erzeugen, nimmt von links (strukturiertes Gitter) nach rechts (unstrukturiertes Netz) zu.
Einen ausführlichen Überblick zum aktuellen Stand der Gittergenerierung findet man in [106]. Zur Behandlung der Gittergenerierung im Umfeld dieser Arbeit wird auf [25] verwiesen.

Berechnungskern

Im vorhergehenden Abschnitt wurde die Mannigfaltigkeit der Problemstellungen im Bauwesen gezeigt. Es genügt also kein Berechnungskern, der auf spezielle Problemtypen zugeschnitten ist, wie z.B. ein Solver für Potentialströmungen oder Euler-Strömungen. An dieser Stelle sollen die spezifischen Fragestellungen von Turbinenströmungen aus dem Bereich des Wasserbaus ausgeschlossen werden. Gekoppelte Probleme mit Fluid-Struktur Interaktion werden hier ebenfalls nicht berücksichtigt. Das gewählte Verfahren soll folglich inkompressible und mit Temperatur behaftete, schwach kompressible Strömungen berechnen können. Dazu kommt entweder ein Solver auf der Basis der direkten Diskretisierung der Navier–Stokes Gleichungen in Frage oder alternativ ein dazu äquivalentes Verfahren wie die Lattice-Boltzmann Methode (siehe Abschnitt 3.1.2).

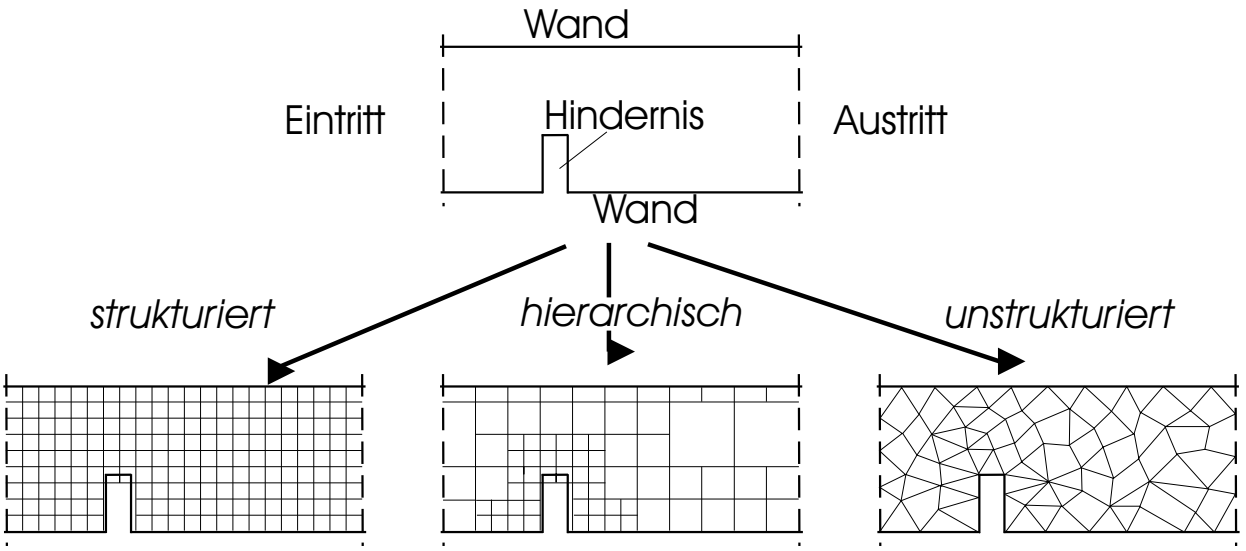

Abbildung 4.4: Verschiedene Abbildungen eines CAD-Modells auf ein diskretes Modell

Das Ziel einer interaktiven Simulation ist, während der Berechnung einzugreifen, Parameter zu verändern und anschließend das Programm an der selben Stelle fortzusetzen. Diese Forderung kann mit einem Verfahren erfüllt werden, das die zeitliche Entwicklung der Strömung numerisch durch eine explizite oder implizite Zeitschleife abbildet.

Visualisierung

Die Visualisierung soll idealerweise mit den Techniken der Virtuellen Welt umgesetzt werden, dennoch werden gleichzeitig Standard-Desktop Lösungen unterstützt. In Abschnitt 2.2 wurden bereits die Vorteile der dreidimensionalen Darstellung für den Experten beschrieben, zusätzlich wird im Umfeld der Planung noch die äußerst suggestive Wirkung von Virtual Reality betont. Die Visualisierung der Daten soll gemeinsam mit einer möglichst photorealistischen Darstellung der umgebenden Geometrie erfolgen, um jedem Anwender einen maximalen Effekt der Wiedererkennung zu vermitteln.
Der Vorgang der Visualisierung und Auswertung ist vor allem für Laien ein kreativer Prozess. So muss dieser in der Lage sein, intuitiv und vor allem interaktiv unterschiedliche Darstellungen der Ergebnisdaten zu betrachten. Interaktiv bedeutet auf jeden Fall, dass der Benutzer bei verzögerungsfreiem Bildaufbau seine Perspektive verändern kann bzw. das Modell transformieren kann. Gleichermaßen sollte es aber möglich sein, mit den Datenabbildungen in Echtzeit zu interagieren, also z.B. beim Verschieben eines Schnittes sofort die Darstellung der neuen Ebene zu erhalten.
In diesem Kontext ist es wichtig, dass die Benutzerdialoge zur Einstellung von Algorithmen zur Datenabbildung einfach erlernt werden können. Die ideale Lösung liegt sicher darin, dem Benutzer selbst die Möglichkeit zu geben, die Dialoge zu konfigurieren (*customizing*). Diese Option ist sehr aufwendig zu implementieren und wird hier nicht berücksichtigt. Dennoch wird dieser Aspekt nicht gänzlich vernachlässigt und bei den späteren Implementierungen darauf geachtet, die Dialoge möglichst einfach und übersichtlich zu gestalten.
Trotz der ausgereiften Techniken im Bereich der Datenvisualisierung (siehe Abschnitt 2.1) ist es unerlässlich, den Umfang der Daten zu verringern. Eine Datenreduktion wurde bereits bei den Schnittstellen gefordert und bekommt durch den Einsatz von Virtual Reality-Techniken eine

zusätzliche Gewichtung.

Persistente Informationen

Die Phase der Auswertung von numerischen Simulationen erfolgt zunächst mit Hilfe von Technologien der Computergraphik (siehe Abschnitt 2.1). Unabhängig davon, ob dies im Rahmen einer interaktiven Simulation oder in Form einer Anwendung zur reinen Visualisierung geschieht, entstehen als Ergebnisse eine Reihe von multimedialen Informationen wie Bilder, Diagramme, Filmsequenzen oder Kommentare und Bemerkungen zu entdeckten Eigenschaften des simulierten Phänomens.
Für den weiteren Verlauf der Planung müssen diese Informationen in Form eines Berichts vorliegen. Die Erfahrung zeigt, dass die oben genannten multimedialen Daten in der Regel nach der Erstellung ungeordnet und oftmals redundant vorliegen. Daraus ergibt sich der Wunsch, während der Visualisierung bereits einen Automatismus zur Verfügung zu haben, mit dem man die Ergebnisse der Auswertung unmittelbar in eine Datenbank ablegt. Liegen diese Daten einmal geordnet vor, kann man die Erstellung eines Berichts weitgehend automatisieren.

Kooperatives Arbeiten

Aufgrund der vielfältigen Zusammensetzung der Konsortien in der Planung von Bauwerken hat das kooperative Arbeiten bereits eine große Bedeutung erfahren (siehe DFG Schwerpunkt 1103: Vernetzt-kooperatives Arbeiten[2]). Die steigende Spezialisierung und Internationalisierung erfordert die verteilte Ingenieurplanung an unterschiedlichen Orten.
Dieses Feld wird hier zwar nicht explizit berücksichtigt, ist aber zur Vervollständigung eines Anforderungskataloges unerlässlich. Es wird später (siehe Abschnitt 8.2) noch darauf eingegangen, dass hier das größte Potential liegt, um an den Ergebnissen dieser Arbeit anzuknüpfen.
Zur Motivation dient folgendes Szenario: Viele kommerzielle CAD-Systeme (z.B. CATIA[3] oder Pro/ENGINEER[4]) unterstützen bereits das gemeinsame Bearbeiten eines CAD-Modells von verschiedenen Arbeitsplätzen. Gleichermaßen vorteilhaft ist es, gemeinsam – unterstützt durch eine Videokonferenz – eine laufende Strömungssimulation zu beobachten, Randbedingungen und Geometrien zu manipulieren und unmittelbar die Ergebnisse wiederum interaktiv auszuwerten.

[2] http://www.iib.bauing.tu-darmstadt.de/dfg-spp1103
[3] http://www.catia.com
[4] http://www.ptc.com

Kapitel 5

Strategisches Lösungskonzept

In Kapitel 4 wurden die Anforderungen an die interaktive Steuerung und Auswertung numerischer Simulationen in Zusammenhang mit dem Planungsprozess vorgestellt. Daraus wird jetzt eine machbare Lösungsstrategie entwickelt. Dazu werden zuerst kurz verwandte Projekte vorgestellt, um einerseits eigene Ideen abzuleiten, aber auch zur Abgrenzung der eigenen Strategie. Der zu entwickelnde Lösungsansatz wird an ein Gesamtkonzept zur Integration der Komponenten einer numerischen Simulation angelehnt. Dieses Konzept stellt einen Berechnungskern nach der Lattice-Boltzmann Methode in den Mittelpunkt. Die zugehörige numerische Diskretisierung ist ein kartesisches Gitter, das mit Hilfe von hierarchischen Datenstrukturen automatisch erzeugt wird. Die Implementierung dieses Konzepts erfolgt in zwei Versionen: Zunächst wird sicher gestellt, dass eine interaktive Visualisierung und Auswertung von vorberechneten Simulationsdaten effizient abläuft (*Offline-Version*). Anschließend sollen die Gittergenerierung, die Simulation und die Visualisierung für kleinere Probleme in einer einzigen Applikation vereinigt werden, um die Simulation interaktiv zu steuern und auszuwerten (*Online-Version*).

5.1 Verwandte Arbeiten - Stand der Technik

Die Verbindung von Strömungssimulationen mit Virtual Reality und Computational Steering hat bereits viele anspruchsvolle Projekte hervorgebracht. Der entscheidende Anstoß dieser Entwicklung ist de facto der virtuelle Windkanal von Bryson [16]. Das dort vorgestellte System visualisiert vorberechnete CFD-Datensätze mit einem Head-Mounted-Display. Ein Datenhandschuh dient dazu, die grafischen Abbildungen der Ergebnisse interaktiv zu manipulieren. Das Projekt führt das Prinzip ein, mit den Techniken virtueller Welten in eine Strömung quasi einzutauchen. Die vorberechneten Datensätze umfassten zu diesem Zeitpunkt (1992) bereits Gitter der Größenordnung $\mathcal{O}(10^4)$. Die aufwendigste Komponente ist die Integration von Partikelpfaden bzw. Stromlinien, wodurch die Leistungsfähigkeit des Systems bei ungefähr 10 Bildern pro Sekunde liegt.
Nachfolgend werden kurz drei aktuelle Projekte in diesem Bereich vorgestellt, von denen sich diese Arbeit zwar abgrenzt, sich aber auf einer konzeptionellen Ebene inspirieren lässt. Die Ergebnisse und Erfolge dieser Projekte begründen die Machbarkeit der hier vorgestellten Überlegungen.

Interaktive Visualisierung von Berechnungen in der Fahrzeugaerodynamik

Dieses Projekt wurde unter der maßgeblichen Beteiligung des "Instituts für Visualisierung und Interaktive Systeme" der Universität Stuttgart[1] von der BMW AG initiiert. Weitergehende Informationen zu dieser Arbeit findet man in [89, 90].
Der Schwerpunkt des Projekts ist die Entwicklung eines Visualisierungssystems für den produktiven Einsatz in der Analyse der Fahrzeugaerodynamik. Ausgangspunkt sind vorberechnete

[1] http://wwwvis.informatik.uni-stuttgart.de/~roettger/html/Main/flow.html

Strömungsberechnungen mit dem Programm PowerFLOW^{TM}. Die Ergebnisdaten liegen auf kartesischen Gittern mit unterschiedlichen Auflösungen (hierarchische Gitter) vor. Diese werden direkt eingelesen und die Algorithmen zur Datenvisualisierung (z.B. Partikelverfolgungen, siehe Abschnitt 2.1) werden speziell für diese Gitter angepasst. Somit konnte vermieden werden, die Daten vor der Visualisierung zu konvertieren bzw. zu reduzieren. Die Geometrie des Fahrzeugs und die Abbildungen der Visualisierungsergebnisse werden in einem Szenegraph vorgehalten, womit Optimierungen unter der Ausnutzung der räumlichen Hierarchie möglich sind. Die Analyse kann wie bei Brysons Windkanal mit den intuitiven Eingabemechanismen von Virtual Reality durchgeführt werden. Abbildung 5.1, rechts oben zeigt einen Bildschirmschnappschuss. Links befindet sich ein graphisches Objekt zum intuitiven Verändern der Startpunkte von Partikelbahnen, das z.B. mit einer Space-Maus versetzt werden kann. Eine besondere Eigenschaft ist die Verfolgung von massebehafteten Partikeln, um eine Simulation der Verschmutzung in die Visualisierung zu integrieren.
Für die eigene Arbeit werden die folgenden Ideen weiterverfolgt bzw. angepasst:

- Um Konvertierungen (mit Interpolationen) für die Visualisierung zu vermeiden, soll dazu das Gitter der Simulation direkt verwendet werden. Mit hierarchischen Gittern kann die Datenmenge für die Visualisierung geringer gehalten werden.
- Ergebnisdaten werden intuitiv mit Virtual–Reality Techniken ausgewertet. Mit der Benutzung eines Szenegraphen zur Datenhaltung ist man sehr flexibel hinsichtlich Optimierungen und Erweiterungen.

EU-Projekt Virtual Intuitive Simulation Testbed – VISiT

Dieses europäische Projekt wurde von der Daimler Chrysler AG gestartet, mit hauptsächlicher Beteiligung der Firma Vircinity und dem Rechenzentrum der Universität Stuttgart. Detailliertere Beschreibungen zu diesem Projekt können [55] entnommen werden.
Es wurde ein Computational Steering System entwickelt, das kommerzielle Gittergeneratoren (ICEM-Hexa[2]) und kommerzielle CFD-Löser (STAR-CD, FLUENT[3]) in ein Virtual–Reality System einbettet. Basis des Systems ist die Integrationsplattform COVISE (siehe Abschnitt 2.3). Der Anwender hat die Möglichkeit, die Randbedingungen der Simulation oder Teile des CAD-Modells zu manipulieren. Dazu wurden spezielle, intuitive Benutzermenüs entwickelt. Als Folge werden Parameter einer laufenden Simulation verändert und gegebenenfalls das Gitter neu generiert. Die Simulation wird dann sofort fortgesetzt und falls möglich, wird der Status der Berechnung vor der Änderung als Ausgangssituation für die weitere Berechnung verwendet. Die Auswirkungen der Veränderung, sprich die neuen Ergebnisdaten, können unmittelbar in der Virtual–Reality Umgebung betrachtet werden und mit klassischen Abbildungsformen (siehe Abschnitt 2.1) analysiert werden. Um möglichst kurze Antwortzeiten zu erhalten, werden die massiv-parallelen Versionen der CFD-Programme verwendet. Diese sind klassische Navier-Stokes Löser mit blockstrukturierten Finite-Volumen Netzen.
Abbildung 5.1, links unten zeigt die Anwendung des Systems bei Daimler-Chrysler. Die Klimaanlage eines Fahrzeugs wird optimiert, indem man interaktiv die Öffnungen der Anlage positioniert und die Auswirkung auf den Passagier beobachtet. Die Veränderung der Geometrien darf dabei aber nur lokal geschehen, damit sich keine Auswirkungen auf die Blockstruktur des Finite-Volumen Netzes ergeben. Rechts unten im Bild ist ein 3D Benutzermenü zur Kontrolle der Simulation zu sehen.

[2] http://www.icemcfd.com
[3] http://www.fluent.com

Für die eigene Arbeit ergeben sich zwei wesentliche Schlussfolgerung, die im weiteren Verlauf aufgegriffen werden sollen:

- Dieses Projekt zeigt, dass sich die Gittergenerierung und die Visualisierung mit einer parallelen Simulation in einer Applikation verbinden lassen.
- Der Einsatz eines interaktiven Systems beschränkt sich heutzutage noch auf Konzeptstudien [3] und zur Unterstützung des Designs.

Interaktive Finite-Element Analyse

An der Chalmers University of Technology in Göteborg[4] wurde ein System entwickelt, mit dem man strukturmechanische Berechnungen interaktiv bedienen kann. Weiterführende Informationen findet man in [21, 22].
Der Anwendungsbereich umfasst große Strukturen, in erster Linie Brücken. Der Benutzer bekommt zunächst die Spannungen oder Verschiebungen der Brücke unter einer Last angezeigt. Wenn diese Last nun verschoben wird, werden die Veränderungen unmittelbar einem angeschlossenen FEM-Programm mitgeteilt und eine neue Berechnung wird gestartet. Während der Berechnungszeit werden dem Benutzer Näherungslösungen angezeigt. Dazu werden vor der interaktiven Anwendung mehrere Lastfälle für wohldefinierte Punkte entlang der Brücke vorberechnet und die Näherungslösung ergibt sich aus einer Interpolation zwischen diesen Lastfällen. Das Finite-Element Netz besteht aus Tetraedern, so dass die aktuellen Entwicklungen darauf abzielen, auch die Geometrie, respektive das CAD-Modell im Virtual–Reality System zu verändern und automatisch eine Neuvernetzung zu starten.
Abbildung 5.1, rechts unten zeigt einen Bildschirmschnappschuss des Systems im Einsatz. In der Mitte des Bildes befindet sich ein kleiner türkiser Konus, der die aufgebrachte Last repräsentiert und einfach mit den intuitiven Eingabegeräten virtueller Welten verändert werden kann.
Wichtig für diese Arbeit ist die Erkenntnis, dass es im Bauingenieurwesen einen durchaus positiven Trend im Bereich von Werkzeugen zur interaktiven Unterstützung des Designs bzw. der Konstruktion gibt.

5.2 Integration in ein Gesamtkonzept

Eigene Erfahrungen zeigen, dass viele wissenschaftliche, aber auch praxisorientierte Anwendungen sehr häufig isoliert entwickelt werden. Die Folge ist eine Vielfalt von Lösungskonzepten. Erhebliche Probleme entstehen dann, wenn die Zusammenführung im Rahmen eines größeren Projekts ansteht. Die gleichen Probleme treten in der Ingenieurspraxis ganz besonders dann auf, wenn aufgrund der Vielfalt der Planungsbeteiligten (siehe Kapitel 4) eine Vielzahl von Modellen aufeinander prallen und es damit zu Schwierigkeiten in der interdisziplinären Kooperation kommt. Aus diesem Grund muss auch eine interaktive Simulation der Strömungsmechanik mit einem Gesamtkonzept der Anwendung von struktur- und strömungsmechanischen Simulationen vereinbar sein. Ein derartiges Konzept wurde am Lehrstuhl für Bauinformatik der Technischen Universität München bereits entwickelt und in [59, 87] veröffentlicht. Das gemeinsame Ziel dabei ist, sämtliche Entwicklungen numerischer Simulationen konzeptionell so zu gestalten, dass ausgehend von einem Produktdatenmodell ein einheitlicher und durchgängiger Weg bis hin zu einem technischen Bericht (respektive der Bemessungsgrundlage) ermöglicht wird.

[4] http://vrlcb.sm.chalmers.se/ifem.shtml

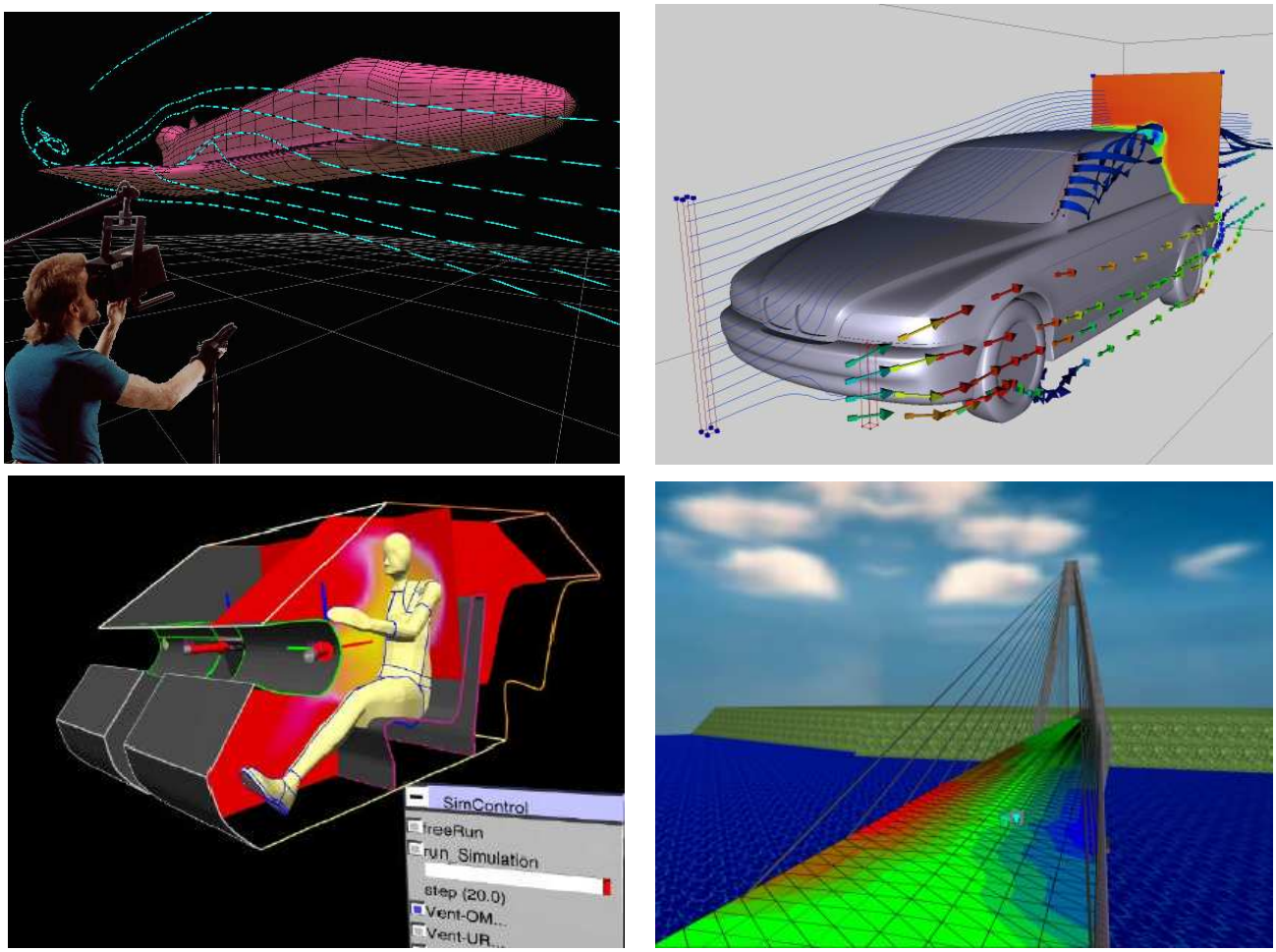

Abbildung 5.1: Verwandte Arbeiten: Virtueller Windtunnel (links oben – aus [16]), Strömungsvisualisierung auf hierarchischen Gittern (rechts oben – aus [90]), Interaktive Simulation des Innenraums eines Fahrzeugs im Projekt VISiT (links unten – aus [55]), Interaktive Finite-Element Analyse von Brücken (rechts unten – aus [21])

5.2.1 Vom geometrischen Modell zur Bemessungsgrundlage

Wenn man von den unterschiedlichen Modellen der Planungsbeteiligten spricht, dann muss man hervorheben, dass die Zusammenführung der Modelle nicht an Fragestellungen wie Dateischnittstellen oder ähnlichem scheitert, sondern an einer völlig unterschiedlichen Semantik der Modelle. Man vergleiche z.B. die Sichtweisen eines Architekten mit denen eines Tragwerksplaners. Es macht wenig Sinn, ein grosses, gemeinsames Supermodell zu entwickeln, welches allen Disziplinen gerecht wird. Pragmatisch sinnvoller ist die Identifikation von gemeinsamen Teilmodellen, die von mehreren Partnern benutzt werden – so auch für den Teilbereich der Simulation.
Numerische Simulationen bestehen zunächst aus mehreren Teilschritten, die mit einer Kette von voneinander abhängigen Modellen verknüpft sind. Alle diese Modelle sind in irgendeiner Form mit der Topologie und der Geometrie einer Struktur verbunden. Das klassische Vorgehen beginnt mit der Definition eines Gitters oder eines Finite-Element-Netzes. Bevor die Berechnung startet, werden diesem Modell Eigenschaften wie Randbedingungen oder Materialparameter zugewiesen. Für die Visualisierung wird dann oft ein anderes Modell verwendet. Die geometrische Beschreibung ist somit sehr stark an den numerischen Rechenkern gebunden. Durch die vielen unterschiedlichen

Disziplinen im Bauwesen mit Berechnungen der Strukturmechanik, der Strömungsmechanik oder sogar der Fluid-Struktur-Interaktion entsteht eine große Palette von Modelltypen. Man vergleiche einfach ein Problem der Wärmeleitung mit der Fragestellung der Luftströmung in einem Raum. Im ersten Fall rechnet man mit einem Finite-Element Verfahren und verwendet Tetraeder- oder Hexaeder-Netze. Im zweiten Fall dagegen hat man ein Navier-Stokes Verfahren mit der Finite-Volumen Methode, die meist mit block-strukturierten Gittern rechnet. Dieser klassische Ablauf einer numerischen Simulation ist in Abbildung 5.2 veranschaulicht.

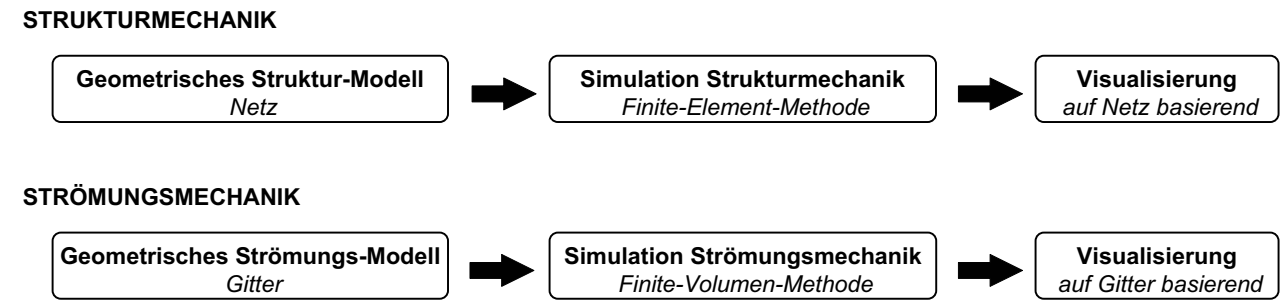

Abbildung 5.2: Klassischer Ablauf numerischer Simulationen mit vielfältigen Modellen

Wenn also das geometrische Modell eines Bauwerks mit den zugehörigen Attributen einer Simulation einzig auf der Basis des verwendeten numerischen Verfahrens definiert wird, dann wird damit die interdisziplinäre Kooperation erheblich erschwert. Eine Änderung der geometrischen Eigenschaften im Modell erfordert automatisch Modifikationen in allen abhängigen Modellen anderer Kooperationspartner. Die logische Folgerung ist, das geometrische und topologische Modell eines Bauwerks auf jeden Fall unabhängig von einem speziellen Typ einer numerischen Simulation zu halten. Diese Forderung ist der wesentliche Stützpfeiler einer effizienten Kooperation zwischen mehreren Disziplinen, vor allem im Bereich numerischer Simulationen.
Abbildung 5.3 illustriert ein logisches Gerüst für den Ablauf einer numerischen Simulation auf der Basis eines eigenständigen geometrischen Modells.
Der Bereich der Strömungsmechanik ist in dieser Abbildung hervorgehoben auf der linken Seite dargestellt. Die rechte Seite illustriert die gleiche Kette für den Ablauf der Strukturmechanik, um einerseits die Analogie zu zeigen und andererseits Querverbindungen unter diesen Ketten zu verdeutlichen. Eine detailliertere Beschreibung der Anwendung dieser Methodik für den Bereich der Strukturmechanik kann [85] entnommen werden.
Die grundlegende Idee dieses Schemas ist, die Bereiche

- geometrisches Modell
- physikalische Modellierung
- numerische Simulation
- Visualisierung, Auswertung und Dokumentation

in semantisch strikt voneinander getrennte Ebenen aufzuteilen. Natürlich stehen diese Ebenen aber in enger Beziehung zueinander.
Ausgangspunkt ist die *geometrische Ebene*. Alle geometrischen Informationen werden einem Produktmodell für Gebäude entnommen. Im Bauingenieurwesen gibt es klare Tendenzen [2] dafür, dass sich auch auf internationaler Ebene langfristig das Modell der Industry Foundation Classes

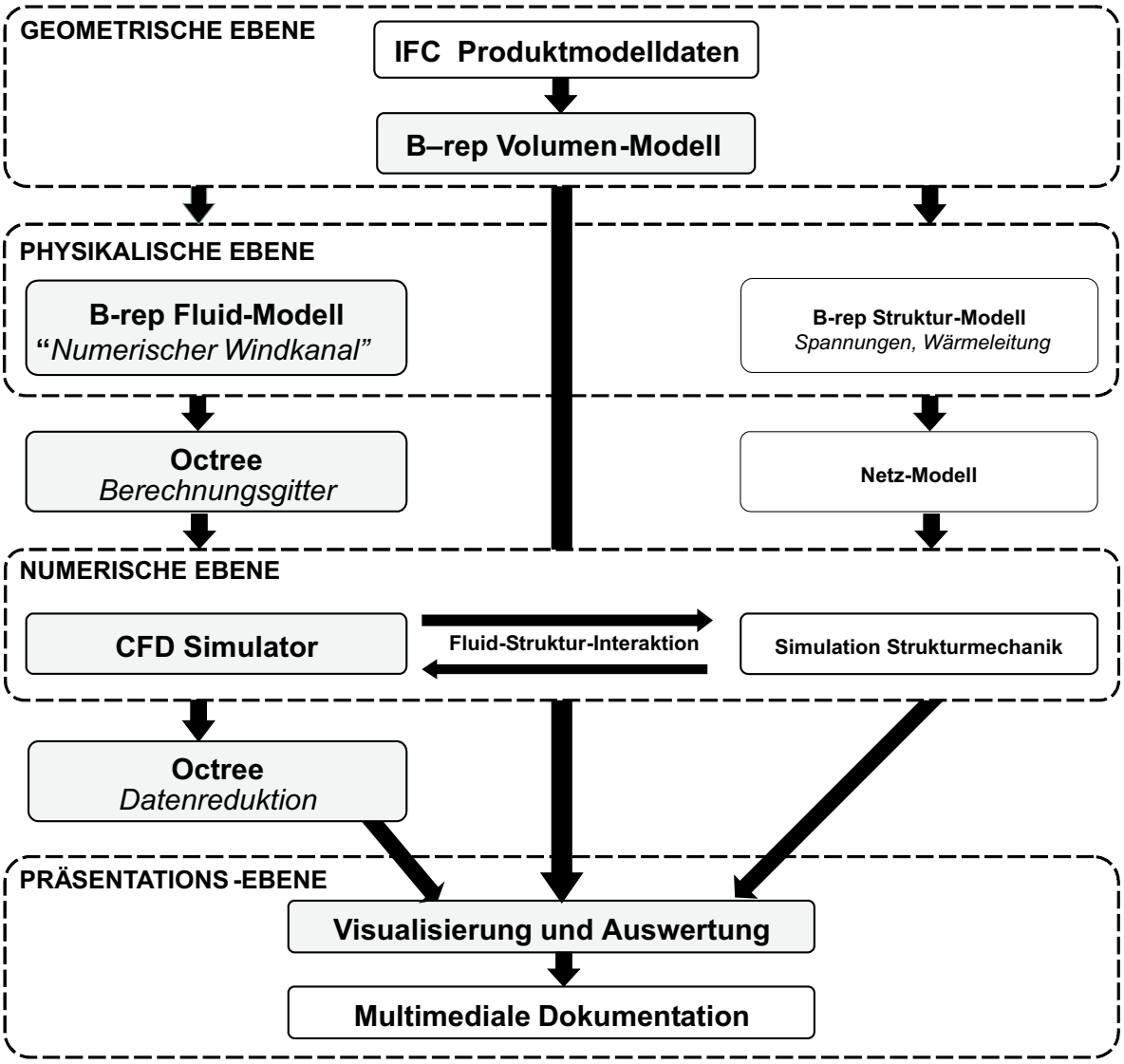

Abbildung 5.3: Gesamtkonzept zur numerischen Simulation basierend auf Produktmodelldaten

(IFC, in der aktuellen Version 2x) der Industrieallianz für Interoperabilität[5] durchsetzen wird. Das Produktmodell dient als gemeinsamer Standard für den Datenaustausch. Es umfasst alle Daten, Prozessregeln und Verknüpfungen, die während der Planung, der Herstellung, dem Betrieb bis zur Entsorgung entstehen. Die Geometrie wird mit Hilfe eines CAD-Systems wie AutoCAD[6] oder direkt über eine Programmierschnittstelle für IFC, wie der Eurostep IFC Toolbox[7] von dem Produktmodell abgeleitet. Am Ende der geometrischen Ebene steht dann ein B-Rep Volumenmodell.

Auf der physikalischen Ebene werden alle weiteren Attribute erzeugt, die für die numerische Simulation nötig sind. Dazu gehören z.B. die Beschreibung des Materials (z.B. Wandrauhigkeiten, Dichte, Viskosität), die Randbedingungen der Strömung (z.B. Geschwindigkeitsprofil und Geometrie einer Öffnung zur Einströmung) oder Lastfälle für die strukturmechanische Simulation. Erst

[5] http://www.iai-ev.de
[6] http://www.autodesk.de
[7] http://www.eurostep.com

nach der vollständigen Beschreibung dieser Ebene wird ein numerisches Modell erzeugt. In [46] wurde ein Modul entwickelt, mit dem die Parameter dieser Ebene bequem in einem CAD-System definiert werden können.
Zur Erzeugung des numerischen Modells ist nun ein sehr aufwändiger Prozess erforderlich, die Gittergenerierung. Dieser Prozess wird in [25] ausführlich diskutiert. Man erkennt, dass das numerische Modell leicht austauschbar ist. In diesem Fall müsste man zusätzlich noch den Vorgang der Gittergenerierung anpassen. Die Ergebnisse der numerischen Ebene werden anschließend an die Ebene der Präsentation weitergegeben. In der Anforderungsliste von Abschnitt 4.2 wurde bereits festgelegt, dass hier eine weitere Abbildung notwendig ist, weil die umfangreichen Ergebnisdaten der numerischen Simulation reduziert werden müssen. Weiter unten wird gezeigt, dass unter Verwendung eines geeigneten numerischen Verfahrens (hier: die Lattice-Boltzmann Methode – siehe Abschnitt 3.1.2) der Übergang von der physikalischen Ebene zur numerischen Ebene und von dieser zur Präsentationsebene mit hierarchischen Datenstrukturen vereinfacht wird. Weitere, komplizierte Abbildungen werden erforderlich, falls man Informationen zwischen verschiedenen Komponenten auf der numerischen Ebene austauschen muss, wie es bei einer Simulation der Fluid-Struktur-Interaktion nötig ist. Ein Ansatz dazu ist in [45] vorgestellt.
Am Ende der Kette steht die Ebene der Präsentation. In der Visualisierung können dann die Ergebnisdaten der Simulation gemeinsam mit dem volumenorientierten B-Rep Modell der Geometrie analysiert werden. Abschließend wird ein Bericht als multimediales Dokument erzeugt.
Es wird betont, dass die Ebene der Präsentation und die Abbildung der numerischen Ebene auf die Präsentations-Ebene in dieser Arbeit zu implementieren sind.

5.2.2 Wahl eines Verfahrens zur Simulation und eines zugehörigen Gittertypen

Die numerische Ebene ist innerhalb des obigen Modells unabhängig und austauschbar im Sinne der Modularität. Das numerische Verfahren kann somit frei gewählt werden, jedoch sind in erster Linie die Auswirkungen auf die Abbildungen, insbesondere von der physikalischen Ebene zu berücksichtigen. Dieser Vorgang wurde bereits in Abschnitt 4.2 als schwer zu automatisieren identifiziert und deshalb wurde gefordert, die Erzeugung des numerischen Modells auf das Verfahren der Berechnung abzustimmen. Man ist zunächst geneigt, die Geometrie auf der numerischen Ebene mit kartesischen Gittern zu beschreiben. Diese Idee wäre aber wertlos, hätte man kein Berechnungsverfahren zur Verfügung, das auf der Basis dieser Diskretisierung die gestellten Anforderungen an Funktion und Effizienz (siehe Abschnitt 4.2) erfüllen kann.
Die in Abschnitt 3.1.2 vorgestellte Lattice-Boltzmann Methode verwendet jedoch in der Regel kartesische Gitter, weshalb dieser Ansatz nun genauer diskutiert wird:

- Die Anforderungen an den Berechnungskern werden von der Lattice-Boltzmann Methode erfüllt. Es liegt ein Verfahren vor, das äquivalent zur inkompressiblen Navier-Stokes Gleichung ist. Turbulente Strömungen können mit einem Large-Eddy Ansatz sehr effizient berechnet werden. Fragestellungen in Verbindung mit Wärmetransport können mit dieser Methode ebenso gelöst werden. Weiterhin bildet die Zeitschleife die Evolution der transienten Strömung ab. Erfolgt in einer interaktiven Simulation eine Modifikation durch den Anwender, so kann die Entwicklung bis zu einer erneuten Konvergenz bzgl. des stationären Zustandes stets angezeigt und verfolgt werden. Man wartet also nicht erst auf ein neues Ergebnis, sondern kann bereits vor dem Erreichen der Konvergenz Tendenzen erkennen und entsprechende Maßnahmen treffen.

- Kartesische Gitter bringen einen erheblichen Vorteil, wenn sich die Geometrie im Verlauf einer Simulation ständig verändert. Dieser Fall wird bei interaktiven Simulationen (siehe Kapitel 7) eine gewichtige Rolle spielen, weil durch den Eingriff des Benutzers Änderungen der Geometrie beliebig oft und vielfältig auftreten werden. Im Falle von randangepassten Gittern bzw. Netzen muss hingegen das Simulationsgebiet ständig neu vernetzt werden. Im Gegensatz dazu ist ein kartesisches Gitter selbst unabhängig von der Geometrie. Unter Verwendung der Lattice-Boltzmann Methode findet man den denkbar einfachsten Fall vor: Die Beschreibung der Geometrie kann automatisch geändert werden, indem man für jede Zelle des Gitters lediglich einen Parameter ändert, der festlegt, ob die Zelle ein Fluid, eine Struktur oder eine bestimmte Randbedingung beschreibt (so genannte *marker and cell* Beschreibung, siehe Abschnitt 3.1.2).

- Allgemein haben kartesische Gitter den Nachteil, dass der Rand nur stückweise konstant angenähert wird. Die Folge ist, dass die Randbedingung nur mit einer Approximationsgüte erster Ordnung beschrieben wird, im Gegensatz zu linear angepassten Rändern mit einer Güte zweiter Ordnung. Wird jedoch zusätzlich der wahre Abstand eines Gitterpunktes zum Rand gespeichert, so kann für die Lattice-Boltzmann Methode gemäß [72] ebenso eine Randbedingung der Ordnung $\mathcal{O}(h^2)$ erreicht werden (h ist der Abstand benachbarter Gitterpunkte).

- Trotz der Verwendung von Turbulenzmodellen ist es offensichtlich, dass man praxisrelevante Beispiele nicht mit uniformen kartesischen Gittern lösen kann. Bei der Umströmung eines Körpers erfordern die physikalisch bedingten hohen Gradienten eine sehr feine Auflösung des Gitters am Rand dieser Körper. Die daraus resultierende kleine Gitterweite ist dann maßgebend für das restliche Strömungsgebiet, obwohl damit in weiten Teilen kein erheblicher Informationsgewinn erreicht wird. Aus diesem Grund ist man langfristig bestrebt, für die Simulation hierarchische, unterschiedlich fein aufgelöste, aber kartesische Gitter (siehe Abbildung 5.4) zu verwenden. In [36, 71] ist die Methodik der Lattice-Boltzmann Methode für unterschiedlich fein diskretisierte, blockstrukturierte Gitter hergeleitet. Die Erweiterung dieses Ansatzes für hierarchische Gitter in 2D wurde in [26] veröffentlicht.

- Die anschließende Visualisierung kann ebenfalls die geordnete Struktur kartesischer Gitter ausnutzen. Die meisten Algorithmen zur Abbildung der Simulationsdaten profitieren damit von einer effizienten Suche eines Punktes in einer Gitter- bzw. Netzzelle (siehe Abschnitt 2.1).

Die Verwendung der Lattice-Boltzmann Methode mit einer Diskretisierung auf der Basis von kartesischen Gittern genügt also den Anforderungen von Kapitel 4 für interaktive Simulationen und lässt sich gut in das oben beschriebene Ablaufschema integrieren.

5.2.3 Hierarchische Datenstrukturen als Organisationsprinzip

Der Übergang von der geometrischen Ebene zum numerischen Modell kann durch die Verwendung von kartesischen Gittern vereinfacht werden. Dazu werden, in Anlehnung an die Dissertation von Frank [39], raumpartitionierende Baumdatenstrukturen (Spacetrees) verwendet.
Abbildung 5.4 zeigt das Grundprinzip einer raumpartitionierenden Baumdatenstruktur in der dreidimensionalen Erscheinungsform, dem Octree. Ausführliche Beschreibungen zu Baumdatenstrukturen sind der Fachliteratur [80, 44] zu entnehmen. Diverse Anwendungen im Bereich der

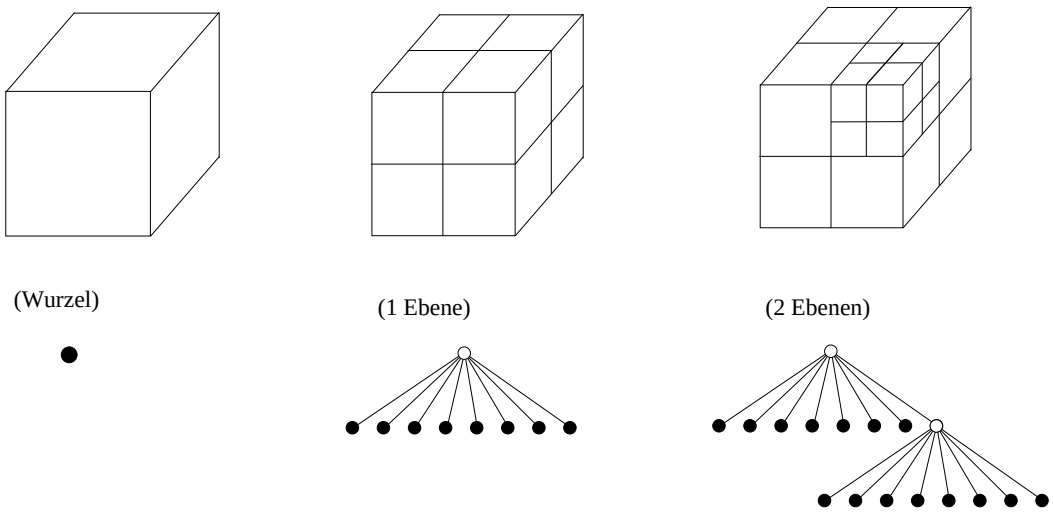

Abbildung 5.4: Der Spacetree (hier: Octree in 3D) als Organisationsprinzip zum Übergang zwischen den Ebenen einer numerischen Simulation

numerischen Simulation sind in [39, 25] beschrieben. An dieser Stelle ist es jedoch notwendig, einige Begriffe zu klären, die vor allem in Kapitel 6 verwendet werden.
In einem Spacetree hat jedes Element (im Kontext dieser Arbeit wird ein Element mit Zelle bezeichnet) einen Vorgänger (Elternzelle bzw. *Parent*) und 2^d Nachfolger (Kindzellen bzw. *Children*), wobei d der Anzahl der Raumdimensionen entspricht. Für d=2 spricht man von einem Quadtree, für d=3 von einem Octree. Das Verhältnis der Kantenlänge einer Kindzelle zur Elternzelle ist dabei stets 1:2. Die Wurzelzelle (*Root*) ist die oberste Zelle in der Hierarchie des Baumes und hat kein Elternelement. Sie hat die Baumtiefe 0. Traversiert man nun von der Wurzelzelle über die Kindzellen nach unten, kommt man über die einzelnen Ebenen (*Levels*) zu den Blättern des Baumes (Blatt-Zellen bzw. *Leaves*), welche per Definition keine Kindzellen besitzen. Die Menge aller Leaves bildet meist das Berechnungsgitter. Betrachtet man nun eine Zelle (jedoch keine Leave-Zelle) an einer beliebigen Position im Baum isoliert mit ihrer kompletten nachfolgenden Struktur, so erhält man wieder einen Spacetree. Diese Eigenschaft erlaubt zur Generierung und Modifikation desselben die Verwendung rekursiver Algorithmen. Im Kontext numerischer Simulationen kommen oft balancierte (geglättete) Bäume zum Einsatz, bei denen eine maximale Differenz der Level benachbarter Zellen vorgeschrieben ist. Meistens wird ein maximaler Unterschied der Levels von 1 vorgeschrieben. Im weiteren Verlauf wird allgemein nur noch vom dreidimensionalen Fall, dem Octree gesprochen.
Mit dieser Datenstruktur kann dann nach folgendem dreistufigem Verfahren aus dem B-Rep Volumenmodell der physikalischen Ebene das Berechnungsgitter erzeugt werden:

1. Mit Hilfe eines CAD-Systems wird ein Facetten-Modell der Oberfläche des B-Rep Modells extrahiert.

2. Die Voxelierung des CAD-Modells erfolgt jetzt durch eine rekursive Verfeinerung eines Octrees. Für eine Zelle wird getestet, ob diese von einer Facette geschnitten wird. Ist dies nicht der Fall, kann abgebrochen werden und die Zelle wird nicht weiter verfeinert. Besteht eine Schnittmenge, wird die Zelle in acht Kindzellen aufgespalten und der Vorgang wird solange rekursiv fortgesetzt, bis entweder eine vorgegebene maximale Baumtiefe erreicht wird, oder keine Schnittmenge zwischen Zelle und Facettenmodell existiert. Alle Blatt-Zellen

des Baumes, die eine nicht-leere Schnittmenge mit der Geometrie besitzen, beschreiben den Rand der Geometrie. Die restlichen Zellen werden in einem Nachlaufschritt noch dem Strömungsgebiet oder den festen Körpern zugeordnet.

3. Der resultierende Octree ist gewöhnlich räumlich unterschiedlich fein aufgelöst. Steht ein Simulationskern zur Verfügung, der auf Baumdatenstrukturen rechnet, so kann der Octree direkt übernommen werden. In der gegenwärtigen Situation wird jedoch weiter mit vollen Gittern gerechnet, so dass die gröberen Zellen noch rekursiv bis auf die Ebene der feinsten Zellen aufgefüllt werden müssen.

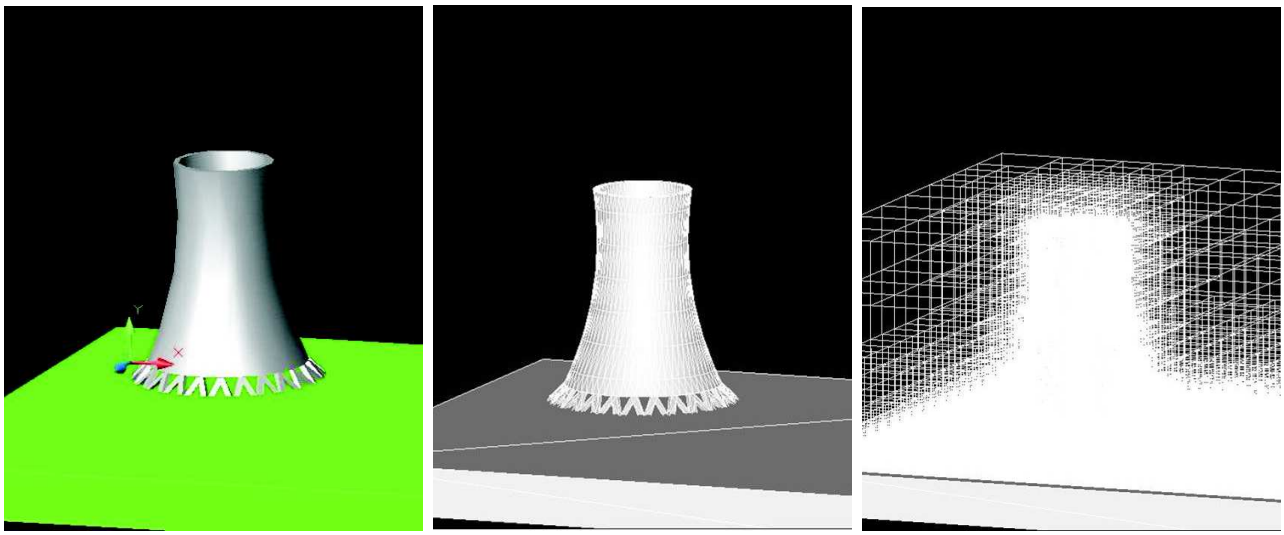

Abbildung 5.5: Erzeugung des numerischen Gitters aus einem B-Rep Volumenmodell (links) über ein Facettenmodell der Oberfläche (Mitte) und einem Octree (rechts)

Abbildung 5.5 zeigt die ersten beiden Schritte vom CAD-Modell über das Facettenmodell zum Octree. Details zu diesem Verfahren und der Implementierung können [53, 25] entnommen werden. Dort wird jedoch eine Erweiterung der Octree-Datenstruktur (siehe Abschnitt 6.1) eingeführt. Diese wird hier für den Übergang zur Ebene der Präsentation verwendet. Das betrifft einerseits eine Reduktion der Ergebnisdaten der Simulation (siehe Abschnitt 6.2) sowie die Anpassung gängiger Methoden zur Datenabbildung auf graphische Darstellungen (siehe Abschnitt 6.3).

5.3 Zwei-Stufen-Strategie

Die Umsetzung des soeben beschriebenen Konzepts auf der Basis von Octree-Datenstrukturen (für die Gittergenerierung, Berechnung und Auswertung) als ein Computational–Steering System (siehe Abbildung 2.8) ist sicherlich ein erstrebenswertes, aber äußerst anspruchsvolles Ziel.
Dieser Schritt wäre allerdings zum heutigen Zeitpunkt noch viel zu groß, zumal die Berechnung und die Visualisierung mit Octrees noch gar nicht vorhanden sind. Der Berechnungskern eines Lattice-Boltzmann Verfahrens für hierarchische Gitter wird in einer komplementären Arbeit [25] erst für 2D entwickelt. Auf die Visualisierung wird hier noch angegangen. Zu Beginn dieses Kapitels wurde bereits von machbaren Strategien gesprochen, die hier zu formulieren sind.

Dazu muss man sich zunächst Gedanken machen über die Größe des simulierten Problems und den zugehörigen Rechenzeiten machen. In diesem Zuge wurden erste Studien für Außenströmungen [23, 87] mit einer Auflösung von $400 \times 100 \times 200 = 8 \cdot 10^6$ Gitterpunkten und einer Reynoldszahl von $Re = 75.000$ durchgeführt (die reale Strömung ist sogar noch wesentlich stärker turbulent). In [59] wurden Innenströmungen mit $302 \times 78 \times 252 \approx 5.94 \cdot 10^6$ Gitterpunkten bei einer Reynoldszahl von $Re = 50.000$ simuliert (dies entspricht den realen Zuständen). Die Erfahrungen aus diesen Studien zeigen, dass man ungefähr 1000 Zeitschritte benötigt, um nach einer lokalen Veränderung der Geometrie ein daraus resultierendes, neues stationäres Strömungsbild zu ermitteln. Dabei wird davon ausgegangen, dass der Ausgangszustand der Berechnung ein stationärer Zustand ist, der vor der Veränderung vorlag.

Für interaktive Simulationen hat dies den folgenden Einfluss: Auf dem Supercomputer Hitachi SR 8000 kann man mit bis zu 100 Millionen Gitterknoten-Erneuerungen pro Sekunde rechnen [96]. Auf einem Workstation-Cluster mit acht Pentium 4 PCs kann man von ca. 5 Millionen Gitterknoten-Erneuerungen [54] pro Sekunde ausgehen. Für die obigen Beispiele ergeben sich für 1000 Zeitschritte die in Tabelle 5.1 aufgelisteten Rechenzeiten.

	Innenströmung aus [59]	Außenströmung aus [23]
Gitterpunkte	$302 \times 78 \times 252 \approx 5.94 \cdot 10^6$	$400 \times 100 \times 200 = 8 \cdot 10^6$
Reynoldszahl	≈ 50000	≈ 75000
Hitachi SR 8000	59,4 sec.	80 sec.
8xP4 Linux-Cluster	1188 sec. = 19,8 min.	1600 sec. = 26,67 min.

Tabelle 5.1: Abschätzung der Rechenzeiten für 1000 Zeitschritte in einer interaktiven Simulation

Weiterhin dauert die Erzeugung eines zugehörigen Octrees (für 8 Millionen Zellen benötigt man eine Baumtiefe von 8) gemäß [53] auf einem PC mindestens 20 Minuten. Der dann benötigte Speicherbedarf ist ein zusätzliches Problem. Natürlich kann und sinnvollerweise muss man die Gittergenerierung auf dem Workstation-Cluster oder dem Supercomputer durchführen, womit man den zeitlichen Aufwand bei der Generierung in den Griff bekommt. Im letzteren Fall wird man aber schnell mit der Problematik von C++ Compilern auf Supercomputern in Konflikt geraten. Im Klartext bedeutet das, dass es dafür kaum volle und effiziente ANSI C++ Compiler gibt.

Man sieht also, dass interaktive Simulationen für die obigen Problemgrößen derzeit nicht besonders attraktiv sind. Für moderne Supercomputer hat man gerade noch akzeptable Antwortzeiten, die man durchaus mit Diskussionen in einem Projektteam überbrücken wird. Bei einem in der Praxis eher anzutreffenden Linux-Cluster sind die Zeiten mit Sicherheit zu lange und klassische Simulationen im Betrieb der Stapelverarbeitung wird man folglich weiterhin benötigen.

Diese Betrachtungen führen zu folgender Hypothese, die anschließend weiterverfolgt werden soll: Eine interaktive Simulation soll bei moderaten Genauigkeitsanforderungen arbeiten und ein qualitativ gutes Ergebnis liefern (das dann zusätzlich noch durch eine genauere Simulation bestätigt werden muss). Geht man von einer Gittergröße von ca. 500.000 Knoten aus, so erhält man die in Tabelle 5.2 aufgelisteten Simulationszeiten. Dabei erkennt man, dass man selbst auf einem Linux-Cluster das Ergebnis in wenigen Sekunden erhält. Sogar die doppelten Werte bei einem Gitter von 1 Millionen Punkten wären noch akzeptabel. Weiterhin wirkt es sich vorteilhaft aus, dass man für diese Gittergröße dann weniger als die oben genannten 1000 Zeitschritte benötigt, bis man nach einer Modifikation eines umströmten Hindernisses wieder einen konvergenten Zustand erhält.

Zeitschritte	1	100	500	1000
Hitachi SR 8000	0.005 sec.	0.5 sec.	2.5 sec.	5.0 sec.
8xP4 Linux-Cluster	0.1 sec.	10 sec.	50.0 sec.	100.0 sec.

Tabelle 5.2: Abschätzung der Rechenzeiten für 500.000 Gitterpunkte bei einer interaktiven Simulation

5.3.1 Interaktive Simulationen in der Produktauslegung als Ergänzung einer klassischen Simulation

In Abschnitt 3.1.1 (siehe Abbildung 3.1) wurde die Rolle von CFD als Ergänzung zum Experiment vorgestellt. So wie die klassische Simulation eine Unterstützung des Versuchs ist, wird hier ein erweiterter Zyklus für die Praxis vorgeschlagen.

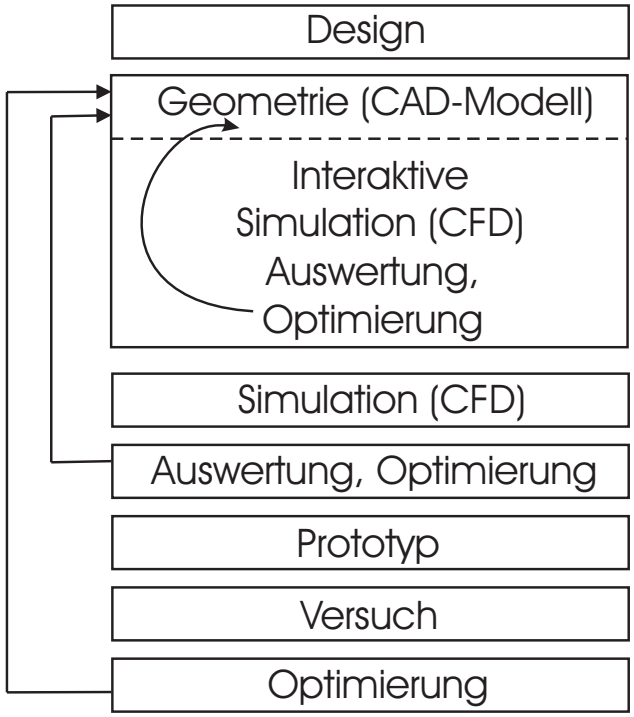

Abbildung 5.6: Zwei-Stufen-Strategie zum Einsatz von CFD in der Produktauslegung

Dieser in Abbildung 5.6 dargestellte Zyklus repräsentiert ein zweistufiges Konzept zum Einsatz numerischer Simulationen:
Die erste Stufe umfasst die interaktive Simulation (*Online-Version* der Datenauswertung). So können Fachplaner (ggf. mit Fachfremden) sehr schnell mehrere Varianten des Entwurfs hinsichtlich der strömungsmechanischen Eigenschaften durchspielen und diskutieren. Als Ergebnis erhält man möglichst wenige Varianten des Designs. Diese werden dann in der zweiten Stufe, respektive einer Simulation im Modus der Stapelverarbeitung verifiziert. Die Ergebnisse werden jeweils mit Hilfe von Virtual-Reality Techniken analysiert (*Offline-Version* der Datenauswertung), um die komplexen dreidimensionalen Phänomene möglichst effizient zu erfassen. Erhält man am Ende der zweiten Stufe noch keine zufriedenstellenden Ergebnisse, dann ist das Design grundlegend zu überdenken und neue interaktive Simulationen schließen sich an. Im anderen Fall, fährt man wie bereits in Abbildung 3.1 mit Windkanalversuchen fort.

Die Entwicklung, bzw. die Implementierung dieses Konzepts gliedert sich in die folgenden zwei Teile:
Zunächst wird die Ebene der Präsentation, respektive die interaktive Analyse unter Einbeziehung von Virtual-Reality Technologien optimiert und implementiert. Dies wird in Kapitel 6 beschrieben. Dazu werden raumpartitionierende Datenstrukturen verwendet, um eine Reduktion der Ergebnisse der Simulation durchzuführen und klassische Verfahren der Datenvisualisierung darauf angepasst. Unter Einbeziehung von hochaufgelösten CAD-Daten zur Darstellung der geometrischen Objekte wird die Erkennung des Produktes verbessert. Für Innenraumströmungen wird als zusätzliche Option die Darstellung von abgeleiteten Größen, wie den Kriterien des menschlichen Komforts, aufgenommen. Schließlich erfolgt noch eine Anbindung der Visualisierung an eine Datenbank, um beispielsweise Bilder oder Kommentare persistent zu halten.
Im Rahmen der zweiten Stufe ist dann ein prototypartiges System zu entwerfen, das die Eingabe und die Visualisierung über eine Interprozesskommunikation mit der Berechnungskomponente verbindet (siehe Kapitel 7). Die Berechnung wird auf einem Supercomputer bzw. einem Workstation-Cluster ausgeführt. Es wird hier allerdings ein Schritt zurückgegangen und es werden volle und nicht hierarchische numerische Gitter verwendet.

5.3.2 Untersuchung zur Genauigkeit von Lattice-Boltzmann Simulationen bei grober Auflösung

Unabhängig von den Rechenzeiten der Tabelle 5.2 muss aber noch geklärt werden, ob man bei einer groben Auflösung überhaupt eine qualitativ gute Lösung erwarten kann. Dazu wird die folgende Simulation von Krafczyk [57] betrachtet, die auf einem uniformen Gitter bei einer Auflösung von $301 \times 61 \times 211$ diskreten Punkten durchgeführt wurde:
Das Problem ist als dreidimensionale Kanalströmung mit einem zeitlich konstanten, logarithmischen Einlaufprofil am linken Rand definiert, wobei am rechten Rand als Austrittsrandbedingung ein Referenzdruck mit verschwindender Normalenableitung der Geschwindigkeit vorliegt. Im ersten Drittel des Kanals befindet sich am Boden aufliegend ein hexaederförmiges Hindernis mit halber Kanalhöhe. Die Decken-, Boden-, und Würfelflächen sind mit einer Haftrandbedingung versehen. Aus experimentellen Untersuchungen [67] kennt man die dabei auftretende komplexe, räumliche Struktur aus Abbildung 5.7.
Die Berechnung wurde zwar mit einem geeigneten Turbulenzmodell durchgeführt, aber die äußerst wichtige Grenzschichtdicke ist um eine Größenordnung zu gering aufgelöst. Trotz dieses erheblichen Mangels zeigen die Ergebnisse der Berechnung, dass alle wesentlichen Merkmale gut getroffen wurden:

1. Der vordere Staupunkt ist vorhanden.

2. Abbildung 5.8 (linkes Bild) zeigt den Hufeisenwirbel stromabwärts des Würfels.

3. Abbildung 5.8 (rechtes Bild) zeigt den Tütenwirbel, der sich durch das ganze Nachlaufgebiet zieht.

4. Das symmetrische Wirbelpaar an den Seiten des Würfels ist in Abbildung 5.8 (linkes Bild) sehr gut zu erkennen.

5. Der hintere Staupunkt ist gut getroffen.

6. Der Wirbel an der Würfeldecke ist wie im Experiment zu beobachten.

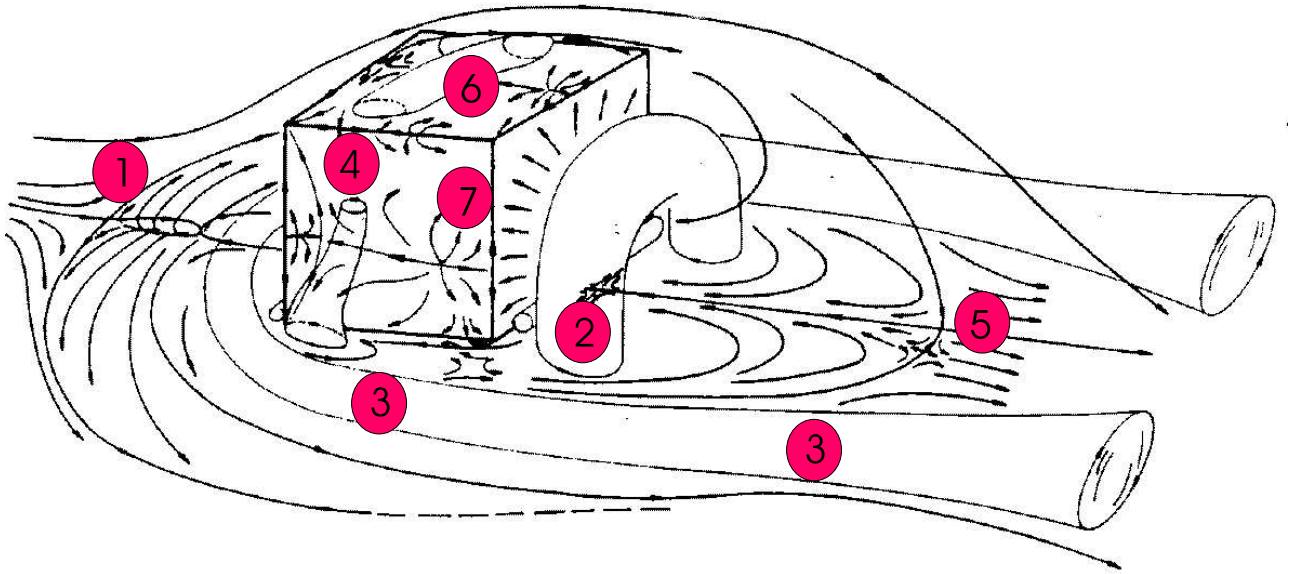

Abbildung 5.7: Dynamische Strukturen bei der Umströmung eines Zylinders (aus [67])

7. Die Verzweigung entlang der Würfelhöhe an den Seiten kann z.B. in Abbildung 5.8 (rechtes Bild) in Form eines symmetrischen Wirbelpaares erkannt werden.

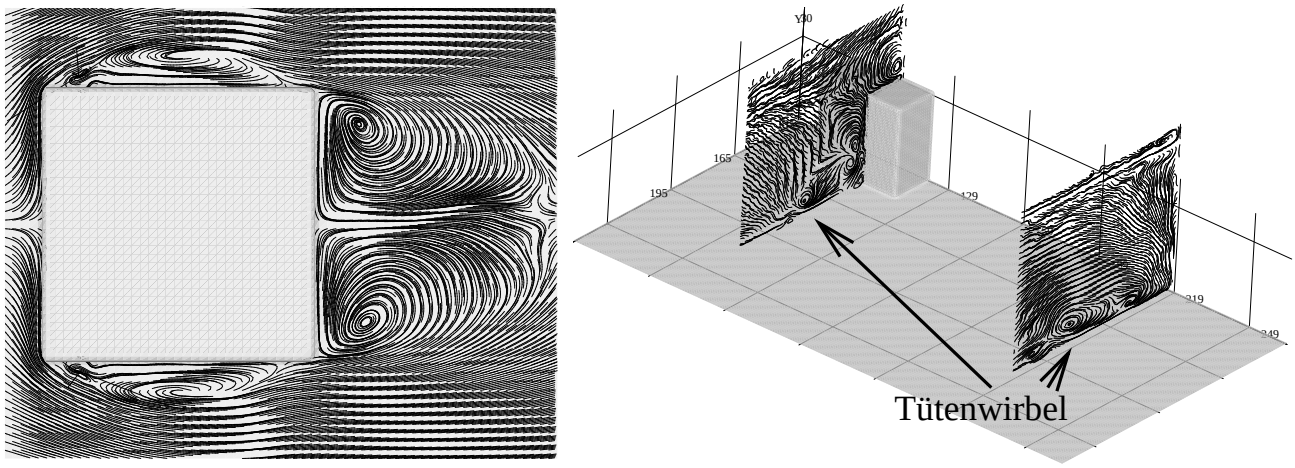

Abbildung 5.8: Aus [57]: Schnitt durch den rechten Hufeisenwirbel (2) und den beiden symmetrischen Seitenwirbeln (4) (links) – Symmetrischer Tütenwirbel (3) (rechts)

Diese Argumentation ist selbstverständlich kein Beweis dafür, dass selbst grobe Auflösungen tendenziell zu qualitativ richtigen Ergebnissen einer Strömungssimulation führen. Dennoch liegt ein Indiz dafür vor, dass man damit sehr häufig auf dem richtigen Weg liegt.

Kapitel 6

Interaktive Analyse der Simulationsergebnisse

In diesem Kapitel werden Entwicklungen vorgestellt, die den Begriff *Virtual Reality basierte Analyse* aus dem Titel dieser Arbeit repräsentieren (*Offline-Version* der Datenauswertung). Im Rahmen des in Abschnitt 5.2 vorgestellten Gesamtkonzepts wurde in [25] eine hybride Netz-Baumdatenstruktur entwickelt, die auf die Generierung des numerischen Gitters durch Voxelierung des B-Rep-Modells der Geometrie zugeschnitten ist. Diese Datenstruktur wird kurz vorgestellt und auf Aspekte der Implementierung eingegangen (Abschnitt 6.1). Der Übergang von der numerischen Ebene zur Präsentation wird durch eine Reduktion der Datenmenge (Abschnitt 6.2) unter Ausnutzung der rekursiven Eigenschaften dieser Datenstruktur unterstützt und Algorithmen zur Datenvisualisierung werden anschließend auf diese Datenstruktur angepasst (Abschnitt 6.3). Mit Hilfe der kombinierten Darstellung von Ergebnisdaten und einem hochaufgelöstem CAD-Modell (Abschnitt 6.5) in einer Virtual Reality-Umgebung wird eine bessere Wiedererkennung im Sinne der Immersion (siehe 2.2) erreicht. Die Geometrien können dabei auch direkt aus einem Produktmodell übernommen werden. In Abschnitt 6.4 wird eine Erweiterung der Auswertung für die Simulation von Innenräumen vorgestellt: In der Praxis relevante, abgeleitete Größen zur Darstellung des menschlichen Komforts werden direkt in die Visualisierung integriert. Um die Ergebnisse der Visualisierung persistent halten zu können, wird ein Modul entwickelt, welches Ergebnisse einer Visualisierungssitzung (d.h. Bilder, Kommentare, etc.) unmittelbar in eine Datenbank speichert, aus der schließlich eine multimediale Dokumentation erstellt werden kann (Abschnitt 6.6). Zuletzt wird die Anwendung an einem Beispiel einer Simulation eines Großraumbüros (Abschnitt 6.7) demonstriert.

6.1 Eine hybride Netz-Baumdatenstruktur

In Abschnitt 5.2 wurden Baumdatenstrukturen als Organisationsprinzip zur Integration der Teilprozesse einer numerischen Strömungssimulation besprochen. Durch die hierarchischen Eigenschaften kann mit Hilfe einer rekursiven Verfeinerung ein allgemeines CAD-Modell nahezu vollautomatisch in ein diskretes Gitter überführt werden [25, 53]. In den Abschnitten 6.2 und 6.3 wird der Vorteil der Hierarchie im Umfeld des Postprocessings deutlich gemacht. Die Verwendung von Baumdatenstrukturen bereitet jedoch Probleme, wenn es darum geht, auf eine bestimmte Zelle im Gitter zuzugreifen. In Abbildung 6.1 ist skizziert, wie man bei der Ermittlung des Weges masseloser Partikel durch das Strömungsgebiet im anschließendem Zeitschritt von Punkt x_i nach x_{i+1} gelangt. Ein Zugriff auf die benachbarte Zelle erfordert eine Traversierung des Baums nach oben und anschließend wieder abwärts. Im ungünstigsten Fall wandert man dabei durch ein Geflecht von Zeigern bis zur obersten Ebene und dann wieder abwärts.
Die Operation des Zugriffs auf eine benachbarte Zelle wird in numerischen Simulationen ständig benötigt, vor allem wenn es darum geht, Gradienten im diskreten Vektorfeld zu ermitteln. In

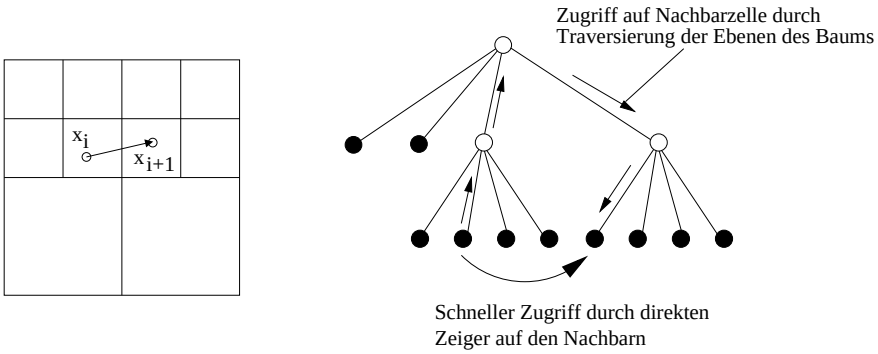

Abbildung 6.1: Baumdatenstruktur mit Verknüpfung der Blätter – schneller Zugriff auf die benachbarten Zellen

[24] wird für den Bereich der Gittergenerierung eine hybride Datenstruktur[1], welche eine Kombination aus einer Netz- und einer Baumdatenstruktur ist. Dabei werden die Blätter der Baumdatenstrukturen zusätzlich mit Zeigern auf die benachbarten Zellen erweitert. Im Falle der oben angesprochenen Partikelverfolgung kann nun anstelle der Traversierung des Baums direkt von Punkt x_i nach x_{i+1} gegangen werden. Mit dieser Datenstruktur werden die Vorteile der Hierarchie und der Rekursion von Baumdatenstrukturen mit den Vorzügen der flachen Verknüpfungen im Sinne eines Netzes (siehe Abbildung 6.1) kombiniert.

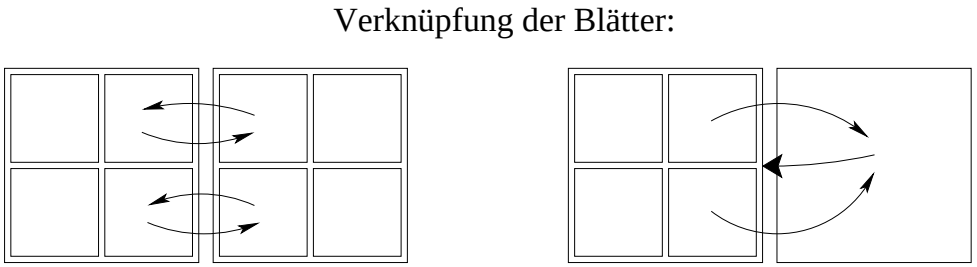

Abbildung 6.2: Unterschiedliche Verzeigerung der Blätter bei benachbarten Zellen mit gleicher Baumtiefe (links) und ungleicher Baumtiefe (rechts)

Einschränkend wird jedoch darauf hingewiesen, dass diese Flexibilität und die damit verbundene Verbesserung der Performance durch einen zusätzlichen Speicherbedarf erreicht wird. Die Verbindung zu den benachbarten Zellen wird dabei nur für die Zellen gesetzt, die mit der benachbarten Zelle eine gemeinsame Fläche teilen (so genannte *Flächennachbarn*). Wenn nun an jeder der sechs Seitenflächen eine stärkere Verfeinerung vorliegt, wären das insgesamt 24 Flächennachbarn. Hier wird die in Abbildung 6.2 dargestellte Verknüpfung der Nachbarn verwendet. Die Verknüpfung von Zellen mit gleicher Baumtiefe (Abbildung 6.2, links) ist unproblematisch, genauso wie der Übergang von einer feinen zu einer gröberen Zelle (Abbildung 6.2, rechts - Verzeigerung von links nach rechts). Liegen jedoch in der Nachbarschaft feinere Zellen vor, so wird der Zeiger auf die Mutterzelle gesetzt, welche die gleiche Baumtiefe wie die betrachtete Zelle hat (Abbildung 6.2, rechts - Verzeigerung von rechts nach links). Um auf den gesuchten Nachbarn zuzugreifen, muss noch zusätzlich eine Traversierung um eine Ebene nach unten erfolgen.

[1]Zur Bezeichnung dieser Datenstruktur wird im Folgenden ebenfalls der Begriff Octree verwendet.

6.1.1 Abstraktion Zelle

Es wurde bereits öfters der Begriff der Zelle verwendet, die ein Baustein der Baumdatenstruktur ist. Unabhängig vom Typ des Gitters kann eine Zelle als Grundbaustein einer Diskretisierung des Raumes abstrahiert werden. Der Unterschied bei den verschiedenen Partitionierungen des Raums (strukturierte und hierarchische Gitter, unstrukturierte Netze – siehe Abbildung 4.4) liegt lediglich in der Anordnung der Zellen zueinander. Diese Arten der Diskretisierung speichern die gesuchten mehrdimensionalen Primärvariablen der Simulation an diskreten Knoten. Man spricht dabei von *knotenassoziierten Daten*. Diese können in Abhängigkeit vom eingesetzten numerischen Verfahren in der Mitte der Zellen (*collocated grids*), versetzt in den Mitten der Kanten bzw. Flächen der Zellen (*staggered grids*), an Orten innerhalb der Zellen (z.B. an den Gausspunkten bei der FEM) oder wie hier unter Verwendung der Lattice-Boltzmann Methode an den Eckpunkten der Gitterzellen liegen. Jede Zelle erhält folglich 8 Zeiger auf die Eckpunkte der Zellen mit den Primärvariablen (diese werden im weiteren Verlauf auch als Eckknoten oder Vertex bezeichnet). Abbildung 6.3 verdeutlicht, dass mehrere Zellen einen Eckknoten teilen. Bei Octrees können das bis zu $8 \times (Anzahl\_der\_Ebenen - 1)$ Zellen sein. Die Knoten werden selbstverständlich nur einmal in einem sequentiell angelegten Feld gehalten. Diese Datenhaltung ist in Abbildung 6.3 exemplarisch für den Eckknoten P skizziert, der von den Zellen 4,5,6,7 und 44 referenziert wird.

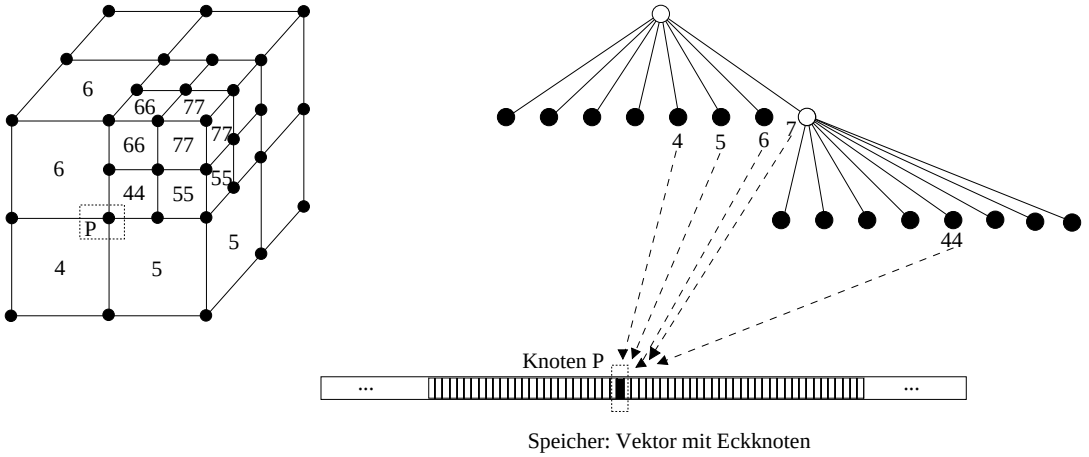

Abbildung 6.3: Knotendaten an den Ecken werden von mehreren Zellen gleichzeitig referenziert

Neben den Zeigern auf die Kinder, die Nachbarn und die Knoten werden für jede Zelle noch eine Reihe weiterer Attribute gehalten. Die wichtigsten Attribute sind im Klassendiagramm der Implementierung in Abbildung 6.5 mit aufgenommen. Dazu gehören der Parameter `classification`, mit dem festgestellt werden kann, ob die Zelle im Bereich des Strömungsgebiets liegt, oder ein festes Objekt beschreibt. Der Parameter `mnWhoAmI` gibt den Index an, den eine betrachtete Zelle als Kind innerhalb der Mutterzelle besitzt. Auf die vollständige Erläuterung dieser Attribute wird im Rahmen dieser Arbeit nicht explizit eingegangen. Sobald es in den weiteren Ausführungen zur Datenreduktion (Abschnitt 6.2) und zur Abbildung der Daten (Abschnitt 6.3) für das Verständnis notwendig ist, werden die betroffenen Elemente der Datenstrukturen erläutert.
Die Abbildung 6.4 zeigt die Nummerierung der Kindzellen, Eckknoten, Flächen und Kanten einer Zelle im Oktalbaum an. Diese Nummerierung wird in den Datenstrukturen zur Beschreibung der Zelle (Klasse `otCell`, siehe Abbildung 6.5) entsprechend abgebildet. Für die Flächen und Kanten gibt es dabei keine spezielle Logik. Deren Anordnung ist beliebig gewählt und die entsprechenden

Nummern sind in Abbildung 6.4 für diejenigen Leser aufgenommen, die sich mit dem zugehörigen Quellcode näher befassen wollen.

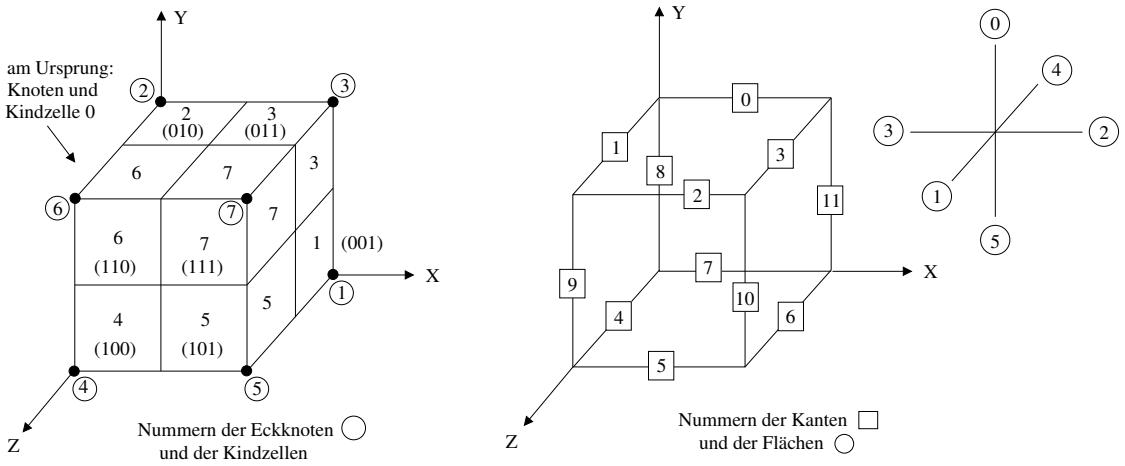

Abbildung 6.4: Zelle einer hierarchischen Diskretisierung: Nummerierung der Kindzellen, Eckknoten, Flächen und Kanten

Ganz anders ist dies für die Nummerierung der Eckpunkte der Zellen, welche mit der Nummerierung der Kindzellen identisch ist. Hierbei handelt es sich um eine Bit-Codierung der Indizes [70]. Zur Erklärung wird der Punkt 6 herangezogen, der einer Bit-Darstellung von 110 entspricht. Das größte Bit wird dann zu 1 gesetzt, falls die z-Koordinate des Punktes dem maximalen z-Wert der Zelle entspricht. Analoges gilt für das mittlere Bit, welches die y-Koordinate repräsentiert. Der x-Wert von Punkt 6 entspricht dem minimalen x-Wert der Zelle und wird damit zu 0. Mit Hilfe dieser Nummerierung können viele Algorithmen sehr effizient und elegant geschrieben werden, indem man teure und umständliche `if`-Abfragen durch Bit-Shifting `<<` und/oder dem binären `und &` sowie `oder |` ersetzt. Exemplarisch sei die Suche des Index der Kindzelle aufgeführt, die den Punkt $P(xp, yp, zp)$ enthält. Ist der Mittelpunkt der Elternzelle mit $P_c(xc, yc, zc)$ gegeben, lässt sich der Algorithmus in einer Zeile wie folgt schreiben:

```
int childIndex = (xc<xp) | ((yc<yp)<<1) | ((zc<zp)<<2);
```

6.1.2 Objektorientierte Infrastruktur

Die oben beschriebenen hierarchischen Datenstrukturen sind in der Bibliothek `VFVis` implementiert. Diese modulare Bibliothek ist objektorientiert entwickelt, und kann somit gängige Anforderungen an eine Implementierung hinsichtlich Wiederverwendbarkeit, Erweiterbarkeit und Wartbarkeit erfüllen.
Abbildung 6.5 zeigt den zugehörigen Klassenbaum in der Notation nach der Unified Modelling Language [78]. Zur Beschreibung der Baumdatenstruktur wird neben der Klasse für eine Zelle (für Octrees ist dies `vfv_otCell`) noch eine Klasse zur Verwaltung eingeführt (`vfv_octree` für Octrees). Weiter unten wird darauf ausführlicher eingegangen. Die hierarchischen Datenstrukturen sind auch in der 2D Version, dem Quadtree implementiert. Die Inhalte der Klassen der 2D Version sind analog zur 3D Version und deshalb in Abbildung 6.5 nicht mehr im Detail aufgelistet.

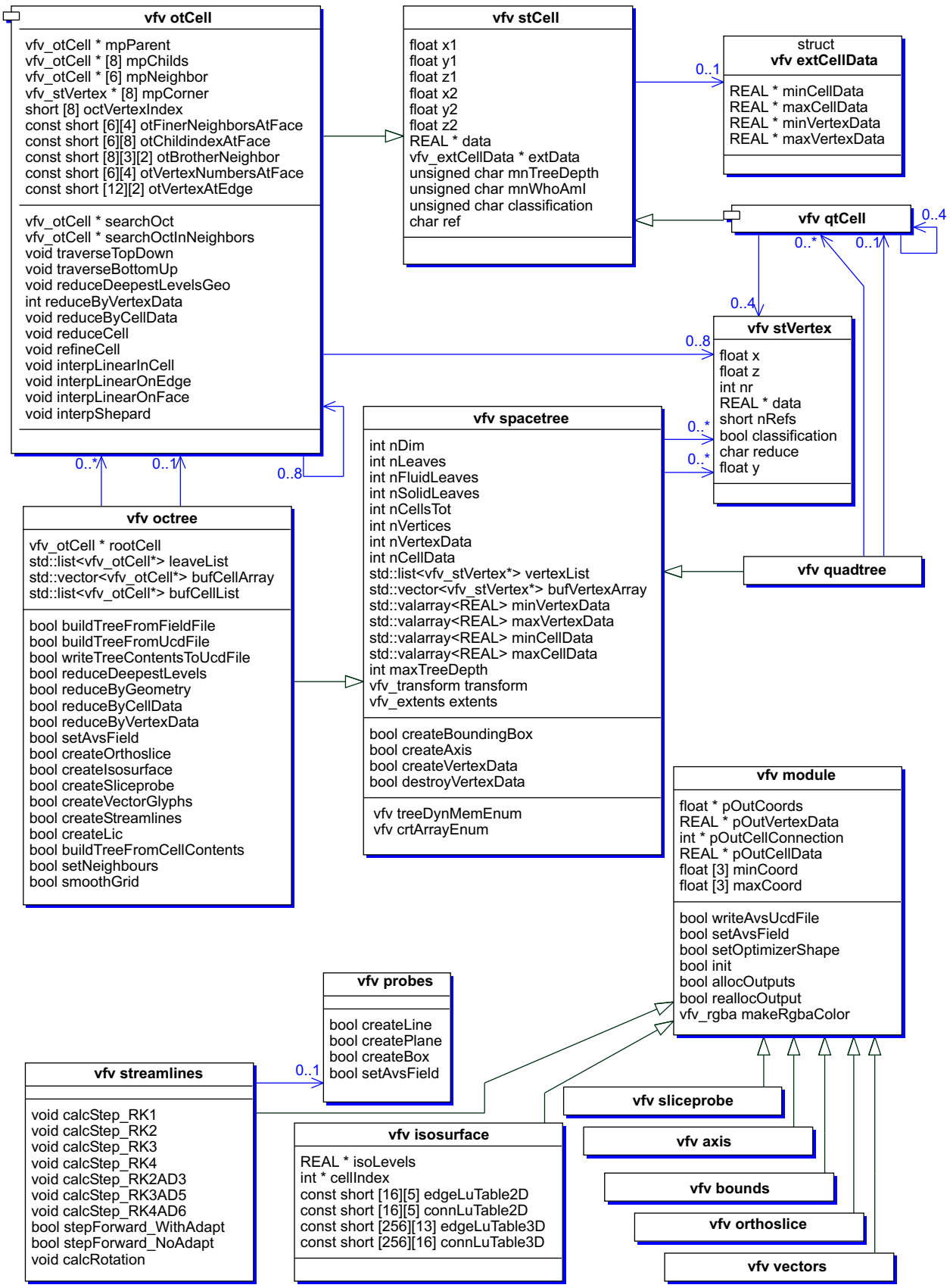

Abbildung 6.5: Klassendiagramm der hierarchischen Datenstrukturen in der Bibliothek `VFVis` – wichtige Funktionen und Attribute sind exemplarisch aufgeführt

Im unteren Teil des Klassenbaums befinden sich die Klassen zur Visualisierung der in den Baumdatenstrukturen gespeicherten Daten. Diese sind weitestgehend von den Baumdatenstrukturen gekapselt. Näheres dazu folgt in Abschnitt 6.3.
Die folgende Auflistung enthält eine kurze Beschreibung der einzelnen Klassen der Implementierung des Octrees:

vfv_spacetree Diese Klasse ist die Basisklasse zur Verwaltung eines Spacetrees. Hier wird auch die Liste aller Eckknoten gehalten (`vertexList`, siehe Abbildung 6.3). Der Member `bufVertexArray` ist dagegen ein Feld, in dem manche Algorithmen die Eckknoten sortiert zwischenspeichern. Weiterhin findet man hier Zählvariablen, um z.B. stets einen Überblick über die Gesamtzahl der Zellen oder der Knoten zu haben. Ebenso ist es wichtig, die minimalen und maximalen Werte der gespeicherten Knotendaten vorzuhalten.

vfv_octree Diese Klasse dient der speziellen Verwaltung eines Octrees. Der wichtigste Member ist selbstverständlich die Wurzelzelle (`rootCell`), welche die rekursive Struktur des Octrees hält.
Oftmals ist die rekursive Natur der Baumdatenstrukturen ein Hindernis für schnelle Algorithmen. In solchen Fällen wird der Baum entrekursiviert [39] und die Zellen in Form einer Liste (`leaveList` für die Blätter, `bufCellList` für den gesamten Baum) oder eines sequentiellen Feldes (`bufCellArray`) gespeichert.
Ausgehend von dieser Klasse kann weiterhin die Datenreduktion (z.B. `reduceByVertexData`) von Abschnitt 6.2 angesteuert werden oder die Berechnung der Datenabbildungen (z.B. `createOrthoslice`) von Abschnitt 6.3 gestartet werden.
Außerdem sind noch eine Reihe von Schnittstellenfunktionen enthalten, um beispielsweise eine unmittelbare Anbindung an eine Programmierschnittstelle eines Visualisierungs-Programms wie AVS/Express (`setAvsField`) zu realisieren.

vfv_stCell Dies ist die Basisklasse zur Beschreibung einer Zelle der oben beschriebenen hybriden Netz-Baumdatenstruktur. Neben den Statusvariablen wie der Baumtiefe (`mnTreeDepth`) – `mnWhoAmI` und `classification` sind weiter oben erklärt – findet man einerseits einen Vektor zur Speicherung von den Zellen zugeordneten Daten (`data`, so genannte *zellassoziierte Daten*) und eine Struktur mit erweiterten Zelldaten (`extData`), die später für die Datenabbildung (siehe Abschnitt 6.3) benötigt werden.

vfv_otCell Hier befindet sich die Implementierung der Zelle für den Octree. Dazu gehören natürlich die Zeiger auf die Kinder (`mpChilds`), die Elternzelle (`mpParent`), die Eckknoten (`mpCorner`) und die Nachbarzellen (`mpNeighbor`).
Neben klassischen Funktionen von hierarchischen Datenstrukturen wie der Traversierung, der Vergröberung (`reduceCell`) und der Verfeinerung (`refineCell`) stellt diese Klasse vor allem Funktionen zur Interpolation bereit, mit denen beispielsweise der Datenwert eines Punktes innerhalb der Zelle nach der Methode von Shepard [97] (`interpShepard`) ermittelt werden kann.
Alle Member vom Typ `const short` sind so genannte Lookup-Tables, auf die im folgenden Abschnitt eingegangen wird.

vfv_stVertex Diese Klasse definiert die Eckknoten. Dazu gehören zunächst die Koordinaten des Knotens (`x,y,z`) und ein Feld mit den vektoriellen Knotendaten (`data`). Die Statusvariable (`classification`) beschreibt, ob der Knoten innerhalb oder außerhalb des Strömungsgebiets liegt. Der Member `reduce` wird bei der Datenreduktion benötigt und `nRefs` speichert

die Anzahl der Zellen, die diesen Knoten referenzieren. Ein Knoten kann somit erst dann gelöscht werden, wenn `nRefs` mit 0 belegt ist.

Es soll hier noch angemerkt werden, dass ausschließlich Listen und Vektoren aus der Standard Template Library (STL) des C++ Standards verwendet werden. Damit spart man erheblich an Arbeitsaufwand und erreicht gleichzeitig eine optimierte und plattformunabhängige Implementierung.

6.1.3 Aspekte zur Optimierung der Performance

Die vorgestellte hybride Netz- Baumdatenstruktur wird für sehr zeitkritische Anwendungen eingesetzt. Eine optimale Rechenleistung erzielt man natürlich hauptsächlich durch eine Verringerung der Komplexität der zugehörigen Algorithmen. Dieser Aspekt ist bereits implizit durch die Gestaltung der Datenstruktur verwirklicht.
Auf der anderen Seite bietet eine sorgfältige und findige Codierung sehr viel Spielraum zur Beschleunigung und zur Optimierung der Speicherung. Bekanntlich steckt der Teufel im Detail, jedoch wird selten darüber geschrieben. Daher werden im Folgenden zwei Gesichtspunkte exemplarisch herausgegriffen:

Index-Tabellen Mit Index-Tabellen (Lookup-Tables) kann der Quellcode wesentlich vereinfacht und effizienter gestaltet werden. Ein Beispiel ist das Feld `otFinerNeighborAtFace[6][4]`, welches für die 6 Seitenflächen einer Zelle die Indizes der 4 angrenzenden feineren Nachbarn hält. Ein Blick auf Algorithmus A-2 (siehe Abschnitt 6.2) zeigt, dass dieser Zugriff z.B. in der Anweisung 24 nötig ist, wenn für die vier Kinder der benachbarten Zelle die Seitennachbarn neu gesetzt werden. Ohne die obige Index-Tabelle müsste man eine Fallunterscheidung (`Switch`-Anweisung) für die sechs Seiten durchführen. Als weiteres Beispiel für Index-Tabellen sei die Speicherung der Indizes der Eckknoten entlang einer Fläche (`otVertexNumbersAtFace[6][4]`) genannt, die beispielsweise für lineare Interpolationen genutzt werden kann.

Eigenes Speicher-Management Bei der Erzeugung von Octrees wird durch die Verfeinerung stets inkrementell neuer Speicher angelegt. Bei der Reduktion hat man den inversen Vorgang des Löschens von Speicher vorliegen. In adaptiven Simulationen treten beide Probleme gleichzeitig auf. Jedes neue Allokieren von Speicher ist intern ein komplexer Vorgang. Dabei wird Speicherplatz dynamisch vom Betriebssystem angefordert. Somit können schnell Speicherfragmente entstehen, die lähmend auf die Baumdatenstruktur wirken. Aus diesen Grund wurde in `VFVis` ein eigenes Speicher-Management implementiert: Zunächst wird ein grosses, sequentiell angelegtes Feld von Zellen erzeugt. Der Speicher wird dann sukzessive aus diesem Feld zugewiesen. Werden zwischenzeitlich Zellen gelöscht, dann werden diese zu einem temporären Puffer hinzugefügt. Bei jeder weiteren Allokierung wird zuerst versucht, den temporären Puffer zu leeren, bevor wieder auf das ursprünglichen Feld zugegriffen wird. Reicht dieses Feld nicht mehr aus, so wird es reallokiert und vergrößert. Im Gegensatz zu ähnlichen Strategien der Programmierung in Fortran77 gibt es einen feinen Unterschied: Die Speicherverwaltung ist vollständig in einem überladenen `new`-Operator von `vfv_otCell` versteckt und hat somit keine Auswirkung auf die Codierung einer Anwendung der Bibliothek `VFVis`.

6.1.4 Integration in eine graphisch–interaktive Entwicklungsumgebung

Durch die modulare Struktur von `VFVis` kann diese recht einfach in eine Umgebung zur grafisch–interaktiven (visuellen) Programmierung (siehe auch Abschnitt 2.1.2) integriert werden. Die Klasse `vfv_octree` hat dazu bereits entsprechende Funktionen implementiert (siehe oben). Abbildung 6.6 zeigt exemplarisch die Integration in das Programmpaket AVS/Express.

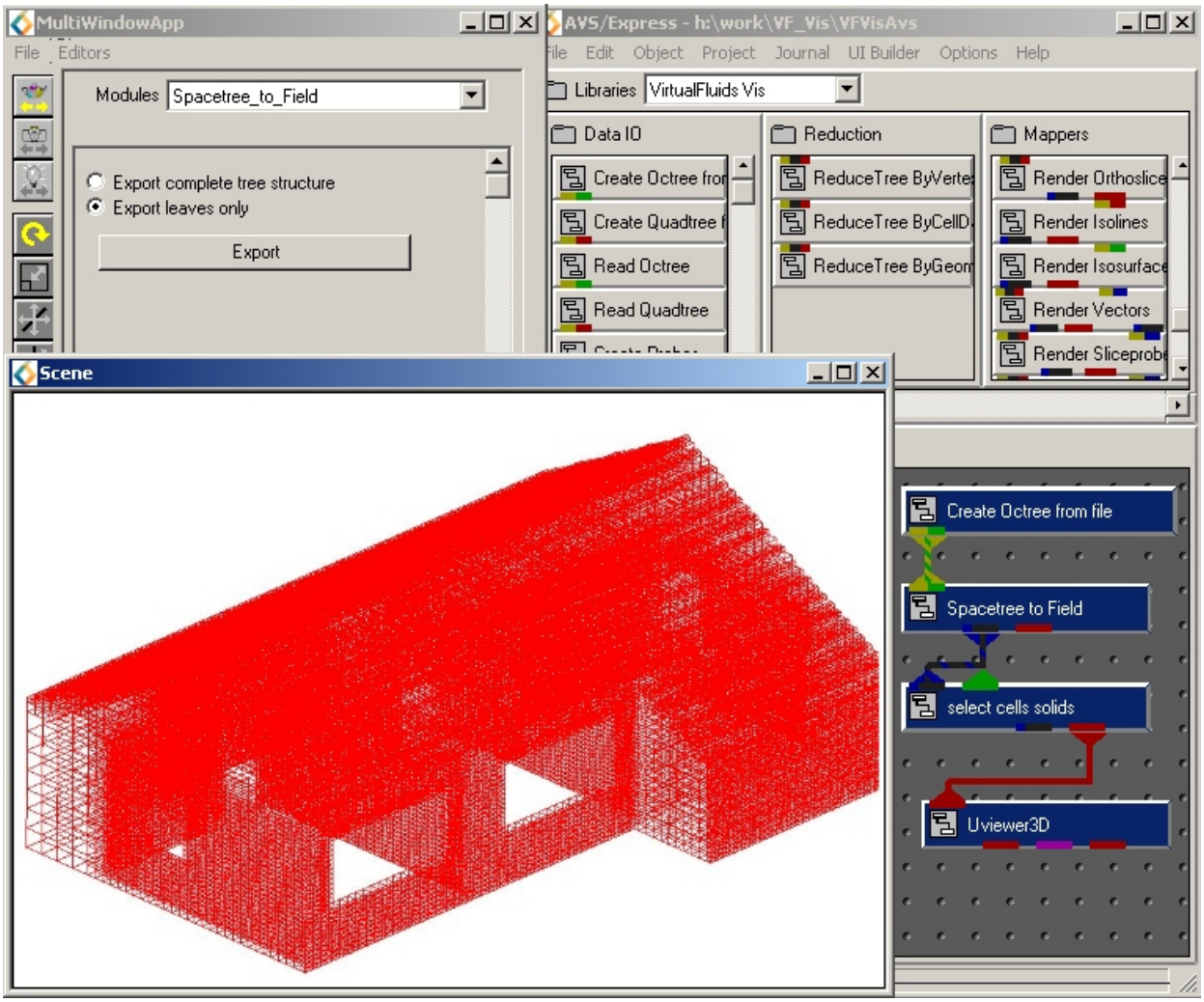

Abbildung 6.6: Integration von `VFVis` in eine Umgebung zur visuellen Programmierung

Die Baumdatenstrukturen sind im Container `Data_IO` integriert. Nun kann man alle Vorzüge der visuellen Programmierung nutzen und einfach einen Octree als grafisches Objekt in den so genannten Netzwerk-Editor (Abbildung 6.6, unten rechts) instantiieren. Die Zellen des Octrees werden mit dem Modul `Spacetree_to_Field` in ein internes Datenformat von AVS/Express exportiert, womit anschließend die volle Breite der Komponenten-Bibliothek von AVS/Express genutzt werden kann.

6.2 Datenreduktion mit hierarchischen Datenstrukturen

Die effiziente Analyse der Ergebnisdaten einer numerischen Simulation erfordert die Auswertung einer Vielzahl unterschiedlicher Abbildungen der Ergebnisdaten in möglichst kurzer Zeit. Dazu werden die unter Umständen riesigen Datenmengen der Simulation mit Hilfe der Octree-Datenstruktur von Abschnitt 6.1 ausgedünnt (siehe auch [61]). Dieser Vorgang ist unabhängig von der Simulation und der Visualisierung. Je nach den verfügbaren Ressourcen der Graphik-Workstation können die hier vorgestellten Verfahren in ein Software-Paket zur Visualisierung integriert werden (siehe Abbildung 6.6, Container `Reduction`) oder direkt nach der Berechnung – und auf deren Plattform – durchgeführt werden. Die grundlegende Idee ist, die hierarchische Natur der Baumdatenstruktur auszunutzen und Zellen lokal und in Abhängigkeit eines durch den Benutzer kontrollierten Kriteriums rekursiv zu vergröbern.

6.2.1 Reduktion von knotenassoziierten Daten

Abbildung 6.7 illustriert den Vorgang der Vergröberung einer Zelle durch lokales Subsampling einer Ebene, wobei die Kindzellen Blätter sein müssen.

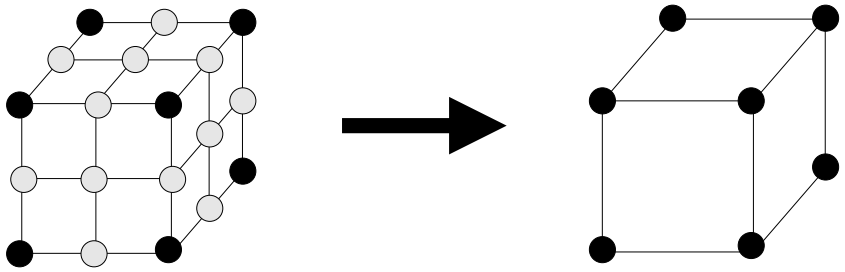

Abbildung 6.7: Datenreduktion durch Subsampling einer Ebene einer Baumdatenstruktur

Die acht Kindzellen einer Zelle referenzieren dabei insgesamt 27 Eckknoten. Nach der Vergröberung der Zelle bleiben lediglich die 8 Eckknoten (schwarze Knoten in Abbildung 6.7) der Zelle übrig, während die restlichen 19 Eckknoten (graue Knoten in Abbildung 6.7 – 1 Knoten in der Zellmitte, 6 Flächenmitten und 12 Knoten in den Mittelpunkten der Kanten) eliminiert werden. Natürlich muss dabei berücksichtigt werden, dass die Mittelpunkte der Flächen und der Kanten von angrenzenden Zellen referenziert werden können (siehe dazu Algorithmus A-2). Ein noch zu definierendes Reduktionskriterium muss also für alle diese 19 Knoten angewendet werden.
Zum Test einer Zelle bzgl. der Möglichkeit der Vergröberung wird der Algorithmus A-1 vorgeschlagen. Wurde ein Knoten bereits in einer benachbarten Zelle getestet, so ist dies durch den Member `reduce` erkennbar (siehe Anweisung 2 in A-1). Sobald der erste nicht reduzierbare Knoten auftritt, wird die Schleife über die Kindknoten (Anweisung 1, A-1) vorzeitig abgebrochen und die Zelle als nicht reduzierbar (Anweisung 16, A-1) markiert. Sonst wird die Zelle als reduzierbar (Anweisung 14, A-1) markiert. Im Kontext der Reduktionsstrategie des gesamten Baums (siehe Algorithmus A-2) wird ersichtlich, dass die Kindzellen nicht sofort gelöscht, sondern nur markiert werden können.
Als Reduktionskriterium wird die Krümmung einer physikalischen Größe des betrachteten Datensatzes gewählt (wie z.B. in [113] vorgeschlagen). Dieses Kriterium hat den Vorteil, dass es unabhängig von der Natur des untersuchten Datensatzes ist. Durch die objektorientierte Implementierung von Abschnitt 6.1.2 können andere Kriterien ohne großen Aufwand integriert wer-

Algorithmus A-1 Reduktion einer Zelle nach den Knotendaten

```
for all 19 Knoten der 8 Kinder gemäß Abbildung 6.7 do
  if Member reduce des Knotens gleich -1 then
    Schleife verlassen (Knoten ist bereits als nicht reduzierbar erkannt worden)
  else
    Reduktionskriterium (z.B. Test auf Krümmung, siehe unten) anwenden
    if Knoten nicht reduzierbar then
      Member reduce des Knotens gleich -1 und Schleife verlassen
    else
      Member reduce des Knotens gleich 1 (d.h. der Knoten ist reduzierbar)
    end if
  end if
end for
if Vorherige Schleife nicht vorzeitig abgebrochen then
  Member ref der aktuellen Zelle gleich 3 (die Zelle wird reduziert)
else
  Member ref der aktuellen Zelle gleich 2 (die Zelle wird nicht reduziert)
end if
```

den. Möglich wären z.B. vom strömungsmechanischen Problem abhängige Kriterien (so genannte phänomenbasierte Sensoren wie der Stoßtensor, siehe [47]).

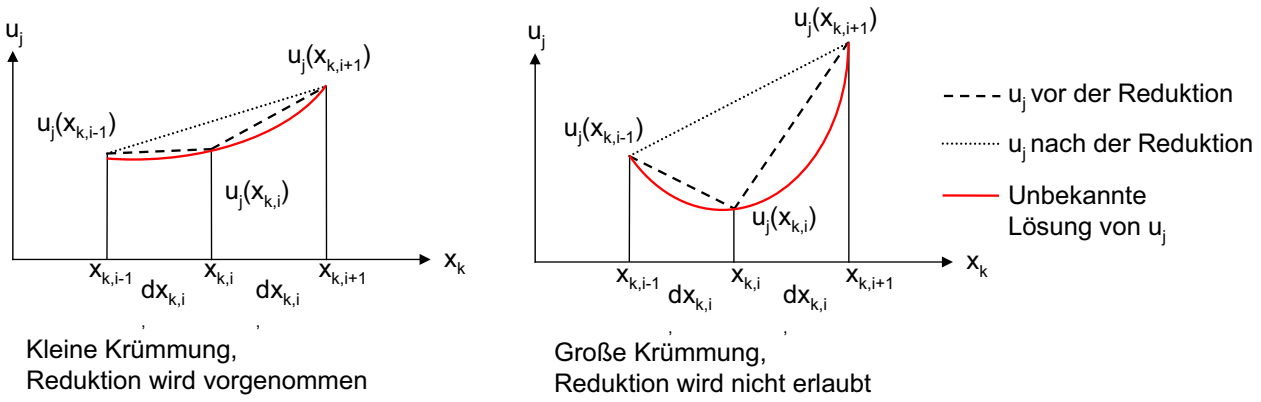

Abbildung 6.8: Die Krümmung des Vektorfeldes als Kriterium zur Datenreduktion

Der Einfluss der Krümmung auf die Reduktion der Zelle ist in Abbildung 6.8 anschaulich dargestellt. Der unbekannte Verlauf der Größe u_j ist als durchgezogene Linie dargestellt. Dieser wird in der ursprünglichen Situation von der gestrichelten Linie approximiert und nach der Reduktion der Zelle von der punktierten Linie angenähert. Bei einer großen Krümmung (Abbildung 6.8, rechts) unterscheidet sich der Verlauf der punktierten Linie stark von der gestrichelten Linie.
Zur Bestimmung des Krümmungsverhaltens der Komponente j des Vektorfeldes u_j entlang der Raumdimension k wird die Ableitung (u_j'') mit finiten Differenzen der Approximationsgüte 2. Ordnung ermittelt. Dazu werden Informationen aus den benachbarten Zellen benötigt. Hier profitiert man von der Verzeigerung der Blätter der hybriden Netz-Baumdatenstruktur (siehe Abschnitt 6.1).
Die 2. Ableitung $u_j''(x_k)$ wird nun gegen den maximalen Wert der Komponente j des Vektor-

feldes $u_{j,max}$ skaliert und dann mit einem vom Anwender zu spezifizierenden Schwellwert ε_{max} in Beziehung gebracht. Ein betrachteter Knoten x_i kann also eliminiert werden, falls für alle Raumdimensionen k und für alle Komponenten j des Vektorfeldes gilt:

$$\frac{u''_j(x_{i,k})}{u_{j,max}} < \varepsilon_{max}; \tag{6.1}$$

Der Benutzer kann somit selbst entscheiden, ob er eine sehr starke Ausdünnung vornimmt, um bei verminderter Genauigkeit eine schnellere Aufbereitung der Abbildungen zu erhalten, oder den Schwellwert aufgrund einer starken Hardware niedrig lässt.

6.2.2 Strategie der Reduktion des gesamten Baums

Bei der Wahl eines Vorgehens zur Reduktion des gesamten Baums muss man sich Gedanken machen, ob man den Baum in einem geglätteten (balancierten) Zustand behält oder auf die Balancierung verzichtet (siehe Abbildung 6.9).

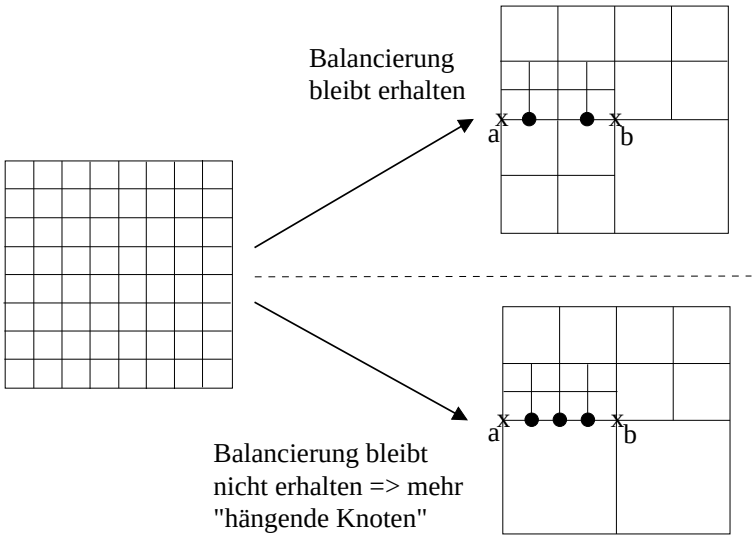

Abbildung 6.9: Gesamt-Strategien der Reduktion (Veranschaulichung in 2D)

Zunächst wurde versucht, die Reduktion des Baums ohne Berücksichtigung eines geglätteten Zustands durchzuführen. Dieses Vorgehen ist zwar einfach zu implementieren und führt zu einer stärkeren Reduktion, hat aber zwei wesentliche Nachteile:

- Hängende Knoten (siehe Abbildung 6.9, entlang von a–b) treten bei Baumdatenstrukturen zwar grundsätzlich auf, deren Einfluss wird aber gerade wegen der Glättung minimiert. In der Visualisierung ist das z.B. in Form eines unstetigen Farbverlaufs oder Löcher in einer Isofläche zu erkennen.

- Die Algorithmen zur Datenabbildung basierend auf Baumdatenstrukturen von Abschnitt 6.3 setzen einen geglätteten Zustand des Baums voraus.

Aufgrund dieser Nachteile wird bei der Reduktion des gesamten Baums stets darauf geachtet, dass der Baum geglättet bleibt. Algorithmus A-2 beschreibt diesen Vorgang.

Bei diesem Vorgehen geht man von der tiefsten Ebene nach oben (Anweisung 3, A-2). In der aktuellen Baumtiefe werden dann nur diejenigen Zellen betrachtet, die selbst zum Gebiet der Fluid-Zellen gehören und Blätter als Kinder haben (Anweisung 5, A-2). Nun wird für alle 6 Seitenflächen der Zelle sichergestellt, dass der Unterschied zur Baumtiefe der Nachbarzellen nicht größer als 1 wird (Anweisungen 6-10, A-2).

Algorithmus A-2 Reduktion des Octrees unter Beibehaltung eines geglätteten Zustandes

```
1: Markiere alle Zellen als nicht reduziert (Variable ref=1)
2: Markiere alle Knoten als nicht reduzierbar (Variable reduce=0)
3: for all Ebenen des Baums (von unten nach oben) do
4:   for all Zellen des Baums do
5:     if Zelle befindet sich im betrachteten Level, ist ein Fluid und hat Blätter als Kinder
       then
6:       for alle sechs Seiten der Zelle do
7:         if Ein angrenzender Nachbar hat noch zwei Ebenen unter sich then
8:           Abbruch der Schleife
9:         end if
10:      end for
11:      if Schleife über die Flächen wurde nicht vorzeitig beendet then
12:        Füge die aktuelle Zelle einer temporären Liste tempList von Zellen hinzu
13:      end if
14:    end if
15:  end for
16:  for all Zellen aus tempList do
17:    Reduktion einer Zelle nach den Knotendaten testen (siehe Algorithmus A-1)
18:  end for
19:  for all Zellen aus tempList do
20:    if Member ref der aktuelle Zelle gleich 2 then
21:      hier wird keine Reduktion durchgeführt, nächste Iteration der Schleife 19
22:    else
23:      for alle sechs Seiten der Zelle do
24:        Nachbarn neu verzeigern
25:      end for
26:    end if
27:  end for
28:  for all Zellen aus tempList do
29:    Zelle reduzieren (Kinder entfernen), falls Member ref gleich 3
30:  end for
31:  tempList leeren
32:  Schwellenwert für nächstes, gröberes Level zur Sicherheit verschärfen
33: end for
34: Rekursive Reduktion für alle Zellen mit Status Solid und nicht mehr benötigte Knoten löschen
```

Damit wird implizit ein weiteres Kriterium zur Reduktion eingeführt. Wenn dieser Test bestanden wird, dann wird die Zelle zu einer temporären Liste von Zellen hinzugefügt (Anweisung 12, A-2), für die nun gemäß Algorithmus A-1 geprüft wird, ob eine Vergröberung möglich ist (Anweisung 17, A-2). Im nächsten Schritt werden für alle reduzierbaren Zellen die Zeiger zu den Flächennachbarn neu gesetzt (Anweisungen 23-25, A-2) und die Kinder dieser Zellen endgültig

entfernt (Anweisungen 28-30, A-2). Die anschließende Iteration in der nächst höheren Ebene wird gegebenenfalls Zellen ausdünnen, die bereits reduziert wurden. Die Berechnung der Gradienten zur Ermittlung der Krümmung nach Gleichung 6.1 basiert dann auf Daten, die bereits vergröbert wurden. Aus diesem Grund wird nun der Schwellwert der zulässigen Krümmung zur Sicherheit verringert (Anweisung 32, A-2). Anschließend können alle Zellen außerhalb des Strömungsgebiets rekursiv vergröbert werden (Anweisung 34, A-2), weil in diesen Zellen keine Daten vorliegen und damit auch das Problem der hängenden Knoten nicht auftritt. Zuletzt werden noch alle nicht mehr benötigten Knoten entfernt (Anweisung 34, A-2).

6.2.3 Parallelisierung der Datenreduktion

Aufgrund ihrer rekursiven Eigenschaften eignen sich Baumdatenstrukturen besonders gut für die Implementierung von verteilten Algorithmen. Das obige Verfahren liegt ebenfalls in einer parallelen Version vor. Die dafür wesentlichen Aspekte werden im Folgenden beschrieben. Details zur zugehörigen Implementierung können [54] entnommen werden.
Es ist vorgesehen, dass im Anschluss an eine parallele Berechnung jeder Prozess für sich einen eigenen Octree vorhält und eine Datenreduktion lokal innerhalb der Domäne vornimmt. Hierbei ist zu beachten, dass man an den Gebietsrändern eigentlich Informationen mit den benachbarten Prozessen austauschen müsste, um die Gradienten zur Berechnung der Krümmung zu ermitteln. Dies wäre grundsätzlich machbar, jedoch wäre dabei ein erheblicher und komplexer Anteil an Kommunikation mit benachbarten Teilprozessen notwendig. Deshalb wird folgendes Vorgehen bevorzugt:

1. In der lokalen Domäne werden die Zellen an den Gebietsrändern zunächst von der Reduktion ausgeschlossen. Die Zellen im Inneren des Gebiets werden wie in Algorithmus A-2 ausgedünnt.

2. Die übrig gebliebenen Blätter und die Knoten werden zu einem Hauptprozess gesendet.

3. Auf dem Hauptprozess werden die Zellen und Knoten der parallelen Teilprozesse gesammelt und zu einem gesamten Octree zusammengefügt (siehe Algorithmus A-3).

4. Der Gesamt-Octree wird nun nochmals gemäß Algorithmus A-2 ausgedünnt.

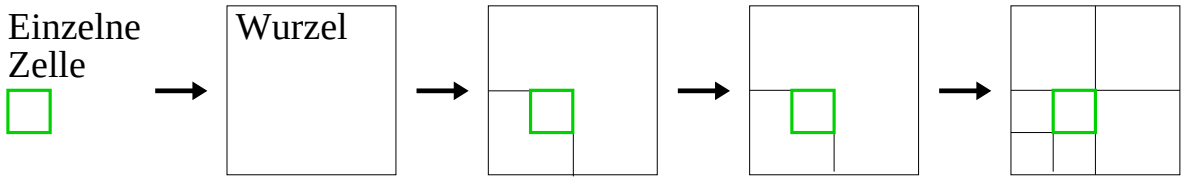

Abbildung 6.10: Einsortieren der sequentiell gespeicherten Zellen in die Baumdatenstruktur

In Punkt 2 tritt das Problem auf, dass keine Zeigerstruktur versendet werden kann. Um die Knoten zu den Zellen zuordnen zu können, werden mit den Zellen noch die Nummern der Knoten versendet. Es werden nur die Blätter versendet, weil eine möglichst geringe Datenmenge über das Netzwerk transportiert werden soll. Auf dem Hauptprozess werden schließlich die eingesammelten Knoten und Zellen gemäß Algorithmus A-3 (siehe auch Abbildung 6.10) zu einen globalen Octree des gesamten Strömungsgebiets zusammengefügt.

Bei diesem Algorithmus (A-3) muss die Schleife von Anweisung 10 bis 21 etwas genauer erklärt werden. Hierbei handelt es sich um einen rekursiven Vorgang: Man befindet sich in der Zelle *pCurrent* und die Zelle *pLeave* soll in den Baum eingefügt werden. Man ermittelt also die Kindzelle *tempChild*, in der sich *pLeave* befinden muss (Anweisung 11, A-3). Wenn *tempChild* bereits existiert, geht man rekursiv in diese Zelle hinein, indem man *pCurrent* auf *tempChild* setzt (Anweisung 19, A-3) und mit der nächsten Iteration der Schleife von Anweisung 10, A-3 fortfährt. Existiert *tempChild* noch nicht, wird geprüft, ob diese der einzufügenden Zelle *pLeave* entspricht (Anweisung 13, A-3). Ist dies der Fall, wird *pLeave* eingefügt und abgebrochen (Anweisung 14, A-3), sonst wird die Zelle *tempChild* allokiert und rekursiv fortgefahren (Anweisung 16, A-3).

Algorithmus A-3 Aufbau des Octrees aus den Zellen und Knoten der parallelen Domänen

```
 1: for alle parallelen Domänen do
 2:    Empfange Liste der Knoten
 3:    Transformiere die lokalen Koordinaten der Knoten in das globale Koordinatensystem
 4:    Feld der Knoten zu einer Liste der gesamten Knoten hinzufügen.
 5:    Empfangen der Liste der Blätter und Hinzufügen zur Liste aller Blätter.
 6:    Erzeugen der Wurzelzelle rootCell
 7:    for all Blätter do
 8:       Aktuelles Blatt ist pLeave
 9:       Aktuelle Zelle currentCell = rootCell
10:       loop
11:          Ermittle die Kindzelle tempChild von currentCell, in der pLeave liegt.
12:          if tempChild existiert noch nicht then
13:             if tempChild entspricht pLeave then
14:                füge pLeave an der Stelle von tempChild ein und verlasse die Schleife 10
15:             else
16:                allokiere tempChild und setze currentCell auf tempChild
17:             end if
18:          else
19:             setze currentCell auf tempChild
20:          end if
21:       end loop
22:    end for
23: end for
24: Fülle noch fehlende Kindzellen rekursiv von der Wurzel mit Solid-Zellen auf
25: Setze in den Zellen die Zeiger auf die Knoten
```

6.2.4 Beispiele

Das zuvor beschriebene Verfahren wird nun an zwei Beispielen demonstriert. Dabei wurden jeweils alle drei Komponenten der Geschwindigkeit mit einer maximalen Krümmung von 0.05 reduziert. Die Abbildungen 6.11, 6.12 und 6.13 zeigen eine Schnittebene durch das Strömungsfeld, auf der der Betrag der Geschwindigkeit abgebildet ist. Der ausgedünnte Datensatz unterscheidet sich optisch kaum vom ursprünglichen Datensatz. Die Anzahl der Gitterpunkte konnte von 7469000 auf 889653 reduziert werden. Der ausgedünnte Octree hat demzufolge nur noch 11,91 % der Eckknoten des originalen Baums.

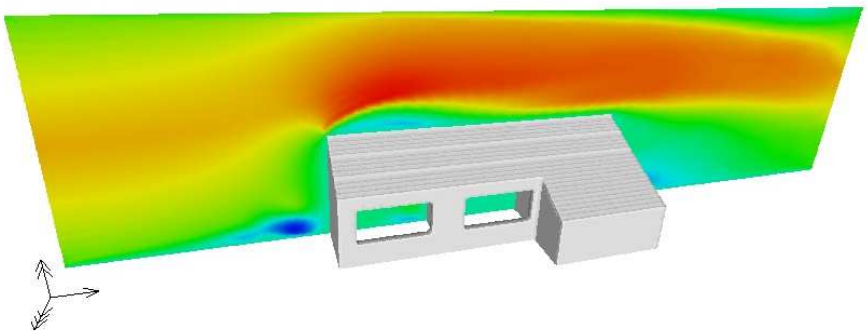

Abbildung 6.11: 3D Datenreduktion – Schnittebene (37345 Punkte) durch das ursprüngliche Gitter (7469000 Punkte)

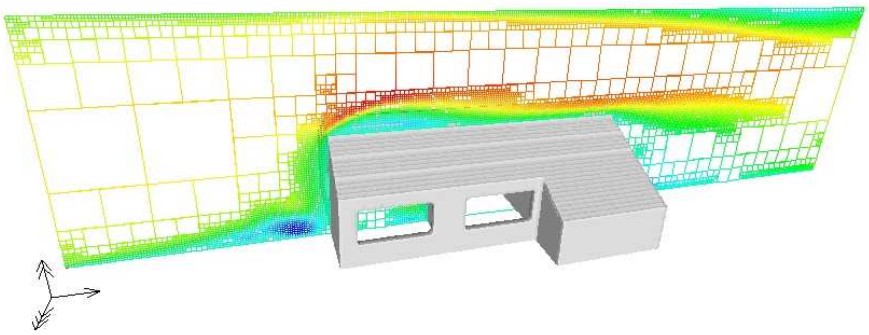

Abbildung 6.12: 3D Datenreduktion – Kanten der Schnittebene (9427 Punkte) durch ein ausgedünntes Gitter (889653 Punkte)

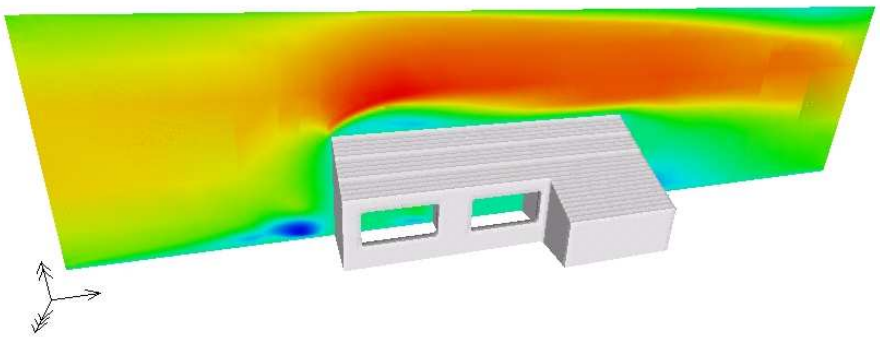

Abbildung 6.13: 3D Datenreduktion – Schnittebene (9427 Punkte) durch ein ausgedünntes Gitter (889653 Punkte)

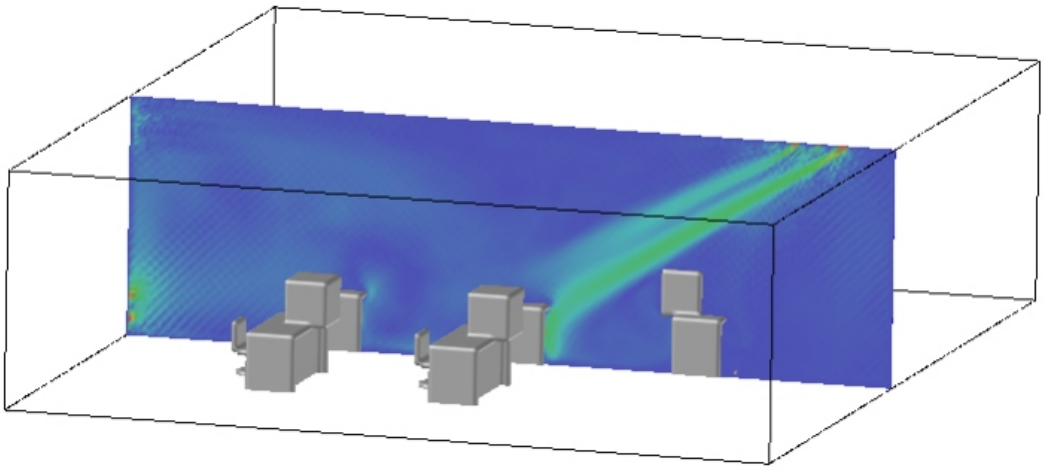

Abbildung 6.14: 3D Datenreduktion – Schnittebene (19656 Punkte) durch das ursprüngliche Gitter (5936112 Punkte)

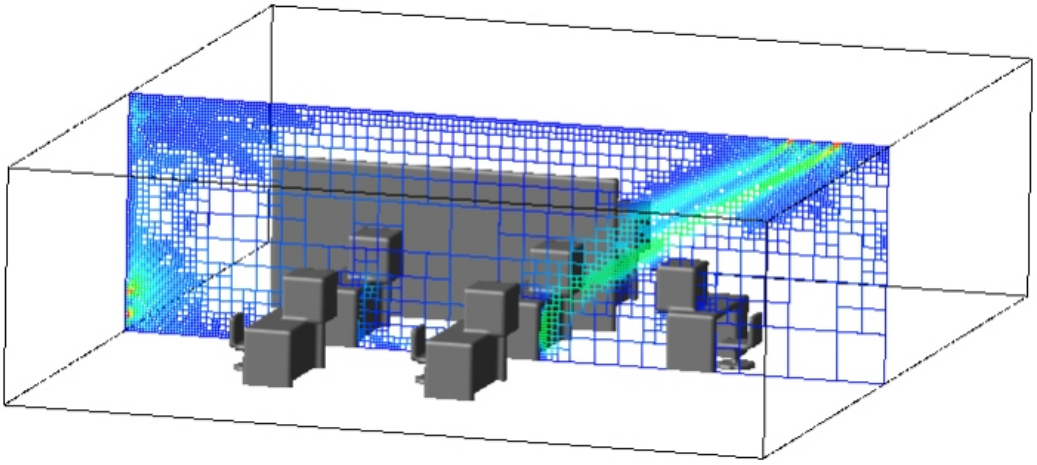

Abbildung 6.15: 3D Datenreduktion – Kanten der Schnittebene (5336 Punkte) durch ein ausgedünntes Gitter (643745 Punkte)

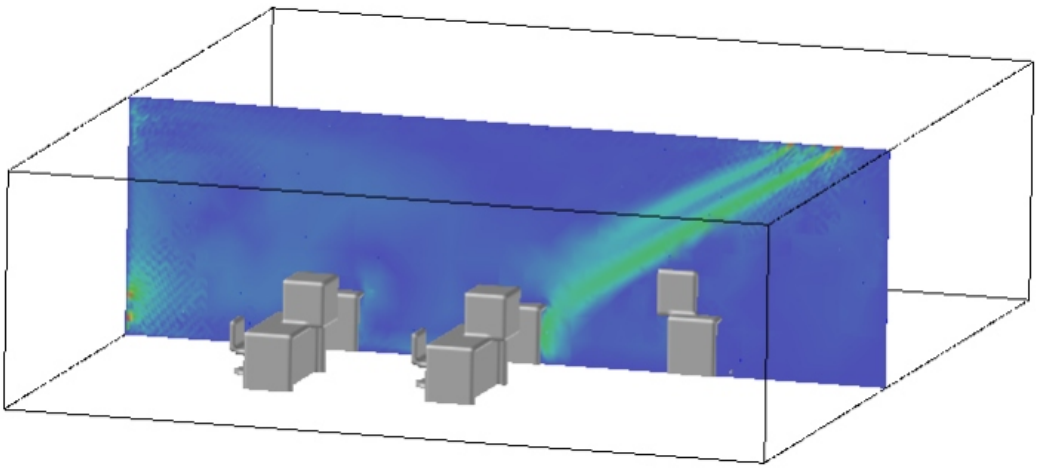

Abbildung 6.16: 3D Datenreduktion – Schnittebene (5336 Punkte) durch ein ausgedünntes Gitter (643745 Punkte)

Das zweite Beispiel zeigt die Simulation einer Belüftungsanlage eines Großraumbüros. Die Abbildungen 6.14, 6.15 und 6.16 zeigen ebenfalls eine Schnittebene durch das Strömungsfeld, auf der der Betrag der Geschwindigkeit abgebildet ist. Der originale Datensatz konnte von 5936112 auf 643745 Gitterpunkten (10,84 %) ausgedünnt werden.

6.3 Datenabbildung mit hierarchischen Datenstrukturen

Ein objektorientiertes Framework zur Visualisierung hierarchisch adaptiver Datenstrukturen wurde in [8] vorgestellt. Die folgenden Abschnitte grenzen sich von dieser Arbeit dadurch ab, dass die vorgestellten Methoden zur Abbildung der Daten sich speziell auf die hybride Netz-Baumdatenstruktur von Abschnitt 6.1 beziehen.
Sämtliche Module zur Datenabbildung sind in das objektorientierte Gerüst von `VFVis` eingebettet. In Abbildung 6.5 sind die zugehörigen Klassen der Datenabbildung im unteren Teil des Klassenbaums zu sehen.
Die Mutterklasse `vfv_module` hält die Variablen zur Speicherung der Ergebnisse. Das sind ein Feld von diskreten Punkten (`pOutCoords`), eine Indexliste zur Beschreibung der Zellverbindungen[2] (`pOutCellConnection`) und ein Feld zur Speicherung der Knotendaten (`pOutVertexData`). Weiterhin enthält `vfv_module` Schnittstellen zur Anwendung innerhalb einer Szenegraph-Bibliothek (`setOptimizerShape`) oder zum Einsatz in Verbindung mit einer Programmierschnittstelle eines Visualisierungs-Programms wie AVS/Express (`setAvsField`). Mit der zuletzt genannten Funktion können diese Algorithmen ohne großen Aufwand in die Komponentenbibliothek von AVS/Express integriert werden (siehe Abbildung 6.6, Container `Mappers`) und die Vorteile der visuellen Programmierung genutzt werden.
Die Algorithmen zur Datenabbildung haben im Prinzip ähnliche Eigenschaften wie die Gegenstücke auf der Basis von uniformen Gittern. Der Unterschied wird am Beispiel der Berechnung von Stromlinien erklärt. Die Erzeugung von Isoflächen lässt sich durch die hierarchische Struktur des Octrees optimieren.
Als weitere Algorithmen sind die Erzeugung von Vektorpfeilen (`vfv_vectors`) und die Generierung von Schnittebenen (`vfv_orthoslice` und `vfv_sliceprobe`) implementiert.

6.3.1 Partikelverfolgung

Bei der Partikelverfolgung ist zwischen Bahnlinien (Weg eines masselosen Partikels durch ein zeitabhängiges Strömungsfeld) sowie Stromlinien (Kurve, deren Tangenten dem Geschwindigkeitsvektor zu einem gegebenen Zeitpunkt entsprechen, bzw. der Weg eines masselosen Partikels durch ein stationäres Strömungsfeld) zu unterscheiden. Hier wurde nur die Berechnung von Stromlinien implementiert. Stromlinien werden im einfachsten Fall mathematisch durch die gewöhnliche Differentialgleichung 1. Ordnung

$$\frac{d\vec{x}}{dt} = \vec{v}(\vec{x},t); \qquad \Rightarrow \qquad \vec{x}_{i+1} = \vec{x}_i + \int_{t_i}^{t_{i+1}} \vec{v}(\vec{x},t)dx; \tag{6.2}$$

beschrieben. Zu deren Lösung sind in der Klasse `vfv_streamline` Runge-Kutta Verfahren bis zur 4. Ordnung mit adaptiver Steuerung der Zeitschrittweite [29] implementiert. Die Interpolation

[2] Eine Zelle ist hier ein Teil der grafischen Objekte, auf denen die Daten abgebildet werden – also z.B. ein Dreieck im Fall von Isoflächen

der Geschwindigkeit $\vec{v}(\vec{x_i}, t)$ am Punkt $\vec{x_i}$ ist eine Operation, die vom Typ des diskreten Gitters abhängig ist und daher von der Klasse des Octrees übernommen wird.
Im nächsten Zeitschritt muss dann festgestellt werden, zu welcher Zelle der Punkt $\vec{x}_{i+1}$ gehört. Die Suche der Zelle, in der ein gegebener Punkt liegt, wird in der Visualisierung sehr häufig verwendet. Diese Operation ist ebenfalls vom verwendeten Gitterdatentyp abhängig und wird entsprechend durch die Klasse `vfv_octree` übernommen. An dieser Stelle profitiert man wiederum von der Verzeigerung der Blattzellen. Befindet sich der gesuchte Punkt in einer Nachbarzelle, so kann diese direkt erreicht werden, während man bei klassischen Octrees im ungünstigsten Fall mehrere Ebenen auf- und abwärts traversieren müsste (vergleiche Abbildung 6.1).

6.3.2 Isoflächen

Die räumliche Organisation des Octrees kann zur Verringerung des Berechnungsaufwands bei der Generierung von Isoflächen genutzt werden. Der Algorithmus *Marching Cubes* [65] generiert Isoflächen als eine Menge von Dreiecken. Um diese Facetten zu ermitteln, wird jede Zelle hinsichtlich einer Schnittmenge mit der Isofläche getestet. Dabei existieren 256 (8 bit) mögliche Schnittfiguren einer Isofläche mit einer Hexaederzelle, wobei eine Schnittfigur aus maximal 4 Dreiecken besteht. Diese Schnittfiguren sind so in eine Lookup-Tabelle (`connLookupTable3D` in der Klasse `vfv_isosurface`) einsortiert, dass der entsprechende Index mit schnellen Bit-Operationen ermittelt werden kann. Dabei wird wie in Abbildung 6.17 der Wert der Isofläche (`isoLevel`) mit den Werten der acht Eckknoten verglichen. Jeder Eckknoten liefert dabei jeweils ein Bit zur Bestimmung des Index der Isofläche innerhalb der Lookup-Tabelle.

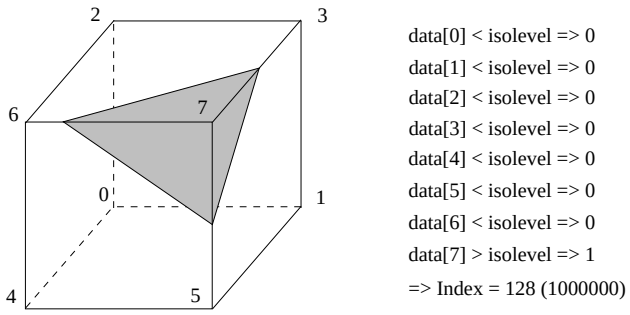

Abbildung 6.17: Ermittlung der Schnittfigur zwischen Zelle und Isofläche beim Algorithmus *Marching Cubes*

Um diesen Vorgang zu beschleunigen, werden für jede Zelle des Octrees erweiterte Zelldaten gehalten (siehe die Struktur `vfv_extCellData` im Klassenbaum von Abbildung 6.5). Dort werden jeweils die maximalen und minimalen Werte aller Knotendaten gespeichert, die sich in der Hierarchie des Baums unterhalb der betrachteten Zelle befinden.
Das modifizierte Vorgehen ist in Algorithmus A-4 zusammengefasst. In einer rekursiv auszuführenden Vorauswahl, werden alle Zweige des Baums gefiltert, für die gilt: Das untersuchte Isolevel ist größer als das Maximum aller tiefer liegenden Knotendaten des Zweiges bzw. kleiner als das Minimum der tiefer liegenden Knotendaten (siehe Anweisung 7 und 8, A-4). Ist dies nicht der Fall, so wird dieser Test rekursiv fortgesetzt. Ist die betrachtete Zelle schließlich ein Blatt, so wird dieses zu einer temporären Liste hinzugefügt (Anweisung 5, A-4). Schließlich wird der Algorithmus *Marching Cubes* für alle vorselektierten Zellen angewendet (Anweisung 15-17, A-4).

Algorithmus A-4 Ablauf der optimierten Erzeugung von Isoflächen für Spacetrees

```
1: Temporäre Liste mit Zeiger auf Zellen: tempList
2: Vorauswahl der Zellen zum Test nach dem Marching Cubes Algorithmus. Rekursives Verfahren beginnend mit der Wurzel-Zelle:
3: aktuelle Zelle ist pC
4: if pC ist eine Blatt-Zelle then
5:    füge pC zu tempList hinzu
6: else
7:    if (IsoWert < Minimum aller tiefer liegenden Zellen) und (IsoWert > Maximum aller tiefer liegenden Zellen) then
8:       Zelle und damit alle tiefer liegenden Zellen werden nicht von der Isofläche geschnitten
9:    else
10:      for all Kind-Zellen von pC do
11:         Vorauswahl rekursiv fortsetzen mit Anweisung 2
12:      end for
13:   end if
14: end if
15: for all Zellen aus tempList do
16:    Marching Cubes Algorithmus anwenden
17: end for
```

6.4 Darstellung von Kriterien des menschlichen Komforts

Postprocessing-Programme bilden typischerweise die Primärvariablen der Strömung wie den Druck oder die Geschwindigkeit dar. Diese Größen selbst sind z.B. für einen Klimaingenieur nur zweitrangig. Sie dienen vielmehr als Basis zur Ermittlung der für eine Bemessung relevanten Kriterien zur Beurteilung des menschlichen Komfortempfindens.
In [74] wird vorgeschlagen, die Berechnung der Komfortkriterien in die Phase der numerischen Berechnung zu integrieren. Diese ist jedoch von einer Reihe von Parametern (z.B. ein Parameter für die Kleidung) abhängig, die nichts mit der eigentlichen numerischen Berechnung zu tun haben. Eine Studie dieser Parameter würde stets eine neue Berechnung zur Folge haben.
Hier wird die Auswertung des menschlichen Komfortempfindens in den Teilschritt des Postprocessings aufgenommen und ein Anwender kann dann den Einfluss der oben angesprochenen Parameter innerhalb der Visualisierung studieren.

6.4.1 Komfortkriterien

Die Bewertung von menschlichen Komfortkriterien wurde zuerst von Fanger [33] quantifiziert und 1994 als ISO-Standard (ISO 7730) verabschiedet.
Im Folgenden wird ein kurzer Überblick über die implementierten Kriterien und deren Einflussgrößen gegeben.

PMV-Index (predicted mean vote): Dieser Index ist ein Kriterium zur Bewertung des thermischen Komforts des Menschen und basiert auf der 'menschlichen Behaglichkeitsgleichung'. Diese beschreibt eine Kombination der Lufttemperatur, der mittleren Strahlungstemperatur, der Luftgeschwindigkeit und der relativen Luftfeuchtigkeit, in der sich ein Mensch unter einer gegebenen Aktivität und Kleidung wohl fühlt. Der PMV-Index bildet die Abweichung von diesem optimalen Zustand auf einer 7-stufigen Beurteilungsskala ab:

(+3) zu warm
(+2) warm
(+1) etwas warm
(0) neutral
(-1) etwas kalt
(-2) kalt
(-3) zu kalt

Der PMV-Index wurde experimentell abgeleitet, indem statistisch eine Beziehung des subjektiven Empfindens des menschlichen Komforts zur Thermoregulierung des Körpers ermittelt wurde. Der PMV-Index ist definiert mit:

$$PMV = (0.303 \cdot e^{-0.036M} + 0.028) \cdot L \quad (6.3)$$
$$\text{und} \qquad L = L(M, W, f_{cl}, t_a, \bar{t}_r, v_{ar}, p_a)$$

M Aktivitätsrate (*metabolic rate).* Dies ist der Energieumsatz bezogen auf den menschlichen Körper. $[W/m^2]$
W abgegebene mechanische Leistung (meistens 0) $[W/m^2]$
f_{cl} Verhältnis der Oberfläche des bekleideten Körpers zur unbekleideten Fläche $f_{cl} = f_{cl}(I_{cl})$
I_{cl} Isolationswert der Bekleidung $[W/m^2]$
t_a Lufttemperatur $[^oC]$
$\bar{t}_r$ mittlere Strahlungstemperatur $[^oC]$
v_{ar} Luftgeschwindigkeit $[m/s]$ (relativ zum Körper)
p_a partieller Wasserdampfdruck $[Pa]$

PPD-Index (predicted percentage of dissatisfied): Dieser Index gibt den Anteil der Personen an, die sich bei einem gegebenen Raumklima unwohl fühlen. Der PPD-Index ist direkt vom PMV-Index abgeleitet:

$$PPD = 100 - 95 \cdot e^{-(0.03353 \cdot PMV^4 + 0.2179 \cdot PMV^2)} \quad (6.4)$$

Zugrisiko DR: Ein Zug ist die Folge einer lokalen Abkühlung und beeinflusst das menschliche Wohlbefinden entscheidend.

Das Zugrisiko sagt den Prozentsatz von Personen voraus, die sich unter den gegebenen Strömungsverhältnissen unwohl fühlen. Die entscheidenden Parameter sind:

t_a Lokale Lufttemperatur $[C]$
v_a Lokale Luftgeschwindigkeit $[m/s]$
Tu Turbulenzgrad [%] (Fluktuationen der transienten Strömung)

Das Zugrisiko berechnet sich zu:

$$DR = (34 - t_a)(v - 0.05)^{0.62}(0.37 \cdot v_a \cdot Tu + 3.14) \quad (6.5)$$

6.4.2 Integration in das Postprocessing einer numerischen Simulation

Diese Kriterien werden nun zu einem Bestandteil einer Postprocessing Software gemacht. Abbildung 6.18 zeigt dies am Beispiel von AVS/Express. Diese Module können nun innerhalb einer Umgebung zur visuellen Programmierung (Abbildung 6.18, unten rechts) verwendet werden.

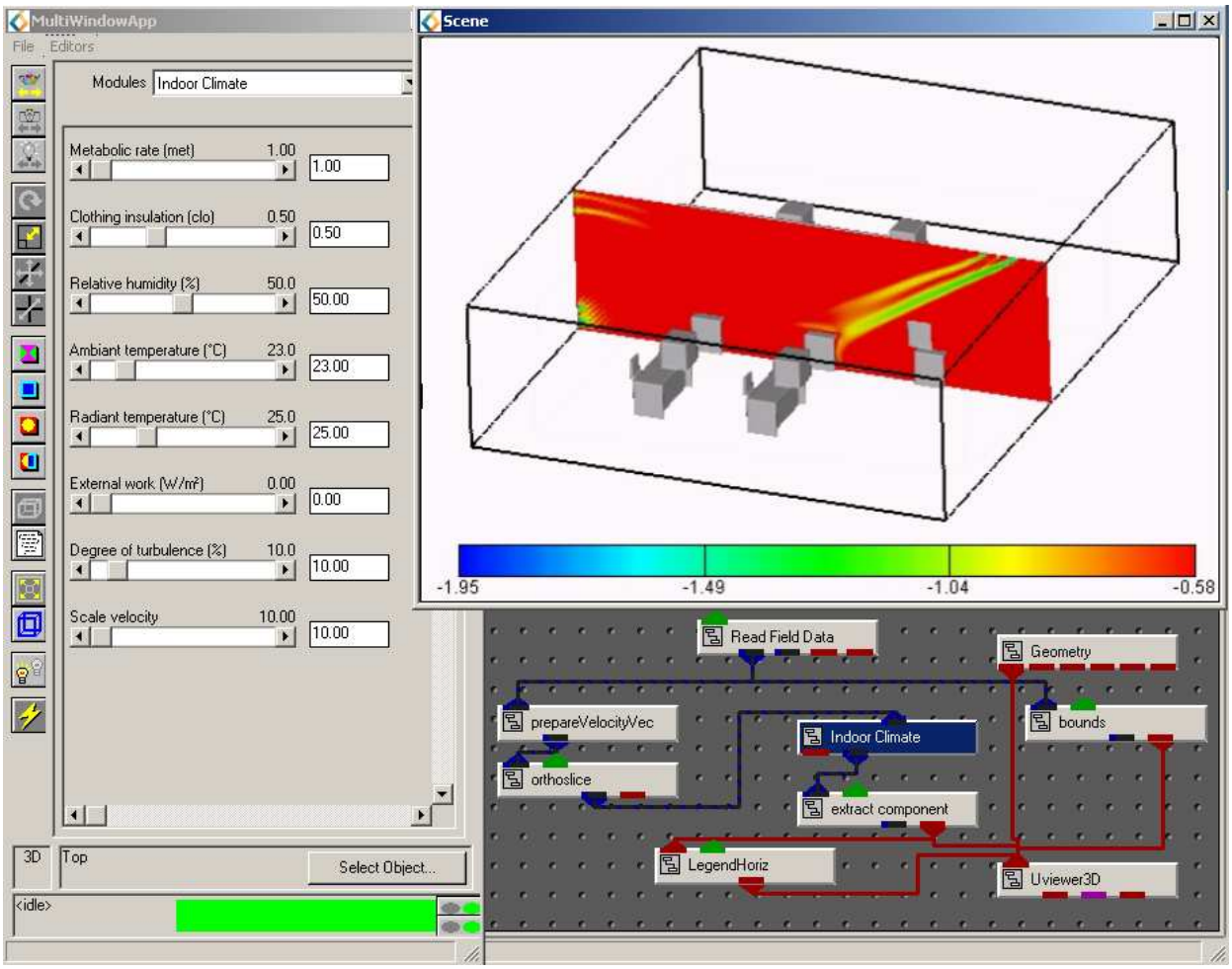

Abbildung 6.18: Abbildung von Komfortkriterien (hier: PMV-Index) als Bestandteil des Postprocessings

Die Luftgeschwindigkeit wird aus der numerischen Simulation übernommen und die Komfortkriterien werden direkt auf den diskreten Punkten des Gitters der Simulation abgebildet (in Abbildung 6.18, rechts oben ist der PMV-Index als Farbplott auf einer Schnittebene dargestellt). Links in Abbildung 6.18 ist ein Benutzerdialog dargestellt, mit dem interaktiv Parameterstudien zur Bewertung des menschlichen Komforts durchgeführt werden können. Dabei werden Größen wie die Kleidung oder die Aktivität einer Person untersucht. Die Lufttemperatur und der Turbulenzgrad werden hier als Mittelwert für den ganzen Raum angegeben, könnten aber (genau wie vorhin die Luftgeschwindigkeit) direkt aus den Ergebnissen einer transienten Simulation übernommen werden.

6.5 Kombinierte Darstellung von Simulationsdaten mit CAD-Daten in Virtual Reality

Die in den Abschnitten 6.1 bis 6.4 beschriebenen Methoden wurden bisher stets in Verbindung mit einer Standard-Visualisierungssoftware wie AVS/Express vorgestellt. Derartige Programm-Pakete sind die typischen Werkzeuge im produktivem Einsatz eines Berechnungsingenieurs. In Abschnitt 5.3 wurde gefordert, Virtual Reality als Grundlage der Visualisierung und Auswertung einzusetzen.

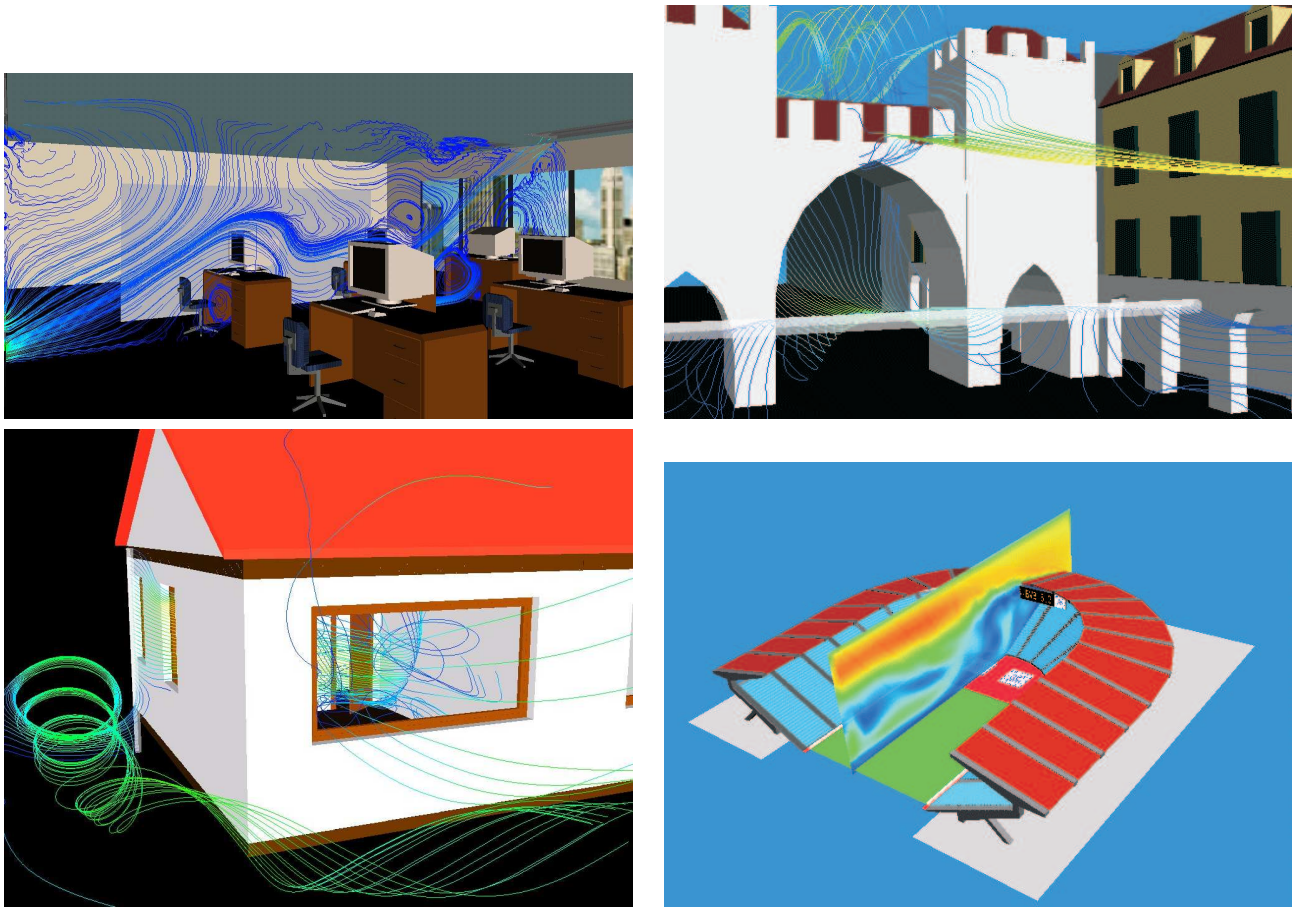

Abbildung 6.19: Weitere Beispiele zu kombinierter Visualisierung

Dieser Schritt wurde durch den Software-Prototyp *VirtualFluidsVis* (VFV) [60] realisiert. Abbildung 6.20 zeigt die Anwendung von VFV bei der Studie der Strömungsverhältnisse infolge einer Belüftungsanlage in einem Großraumbüro.
Die wesentlichen Eigenschaften von VFV können wie folgt zusammengefasst werden:

Kombinierte Darstellung von CAD-Daten mit den Ergebnissen der Simulation: Die geometrischen Objekte werden hierbei als texturierte Oberflächen dargestellt, womit eine bessere Erkennung des computergenerierten Modells gewährleistet wird. Damit wird es leichter, fachfremden Personen die Korrelation zwischen den geometrischen Eigenschaften eines Gebäudes, den hydrodynamischen Randbedingungen und dem resultierenden Strömungsfeld zu erläutern. Bildschirmabbildungen von VFV und damit Beispiele zur kombinierten Visualisierung findet man in den Abbildungen 2.3 und 6.19.

Abbildung 6.20: Kombinierte Darstellung von CAD-Daten und Ergebnissen der Simulation in Virtual Reality

Objektorientierte Entwicklung und Datenhaltung mit einem Szenegraph: Durch diese beiden Merkmale ist die Applikation sehr gut erweiterbar. Abbildung 6.21 zeigt den Szenegraph von VFV. Auf der linken Seite wird die Geometrie eingebunden. Als Schnittstelle zur internen Struktur des Szenegraphen wurde VRML verwendet, das als ISO-Standard zur Beschreibung virtueller Welten von einer sehr breiten Palette von CAD-Systemen exportiert werden kann. Ebenso gibt es eine Reihe von Werkzeugen die Geometrien in Form von Produktmodelldaten nach VRML exportieren. Weiter unten wird ein eigener Prototyp (mit zusätzlichen Eigenschaften) eines Konvertierers von Produktmodelldaten nach VRML vorgestellt. Der Zweig auf der rechten Seite des Szenegraph von Abbildung 6.21 umfasst die Darstellung der Ergebnisse der Berechnung. Die Daten werden dabei als so genannte `IndexedGeometry` dargestellt – eine Datenstruktur zur Beschreibung eines Facettenmodells. Die Erzeugung dieser Abbildungen wird dann in einer vom Szenegraphen gekapselten Komponente wie die Klassenbibliothek `VFVis` (siehe Abbildung 6.21) durchgeführt.

Intuitive Benutzerschnittstelle: Die Abbildungen der Simulationsdaten können sehr einfach manipuliert werden: In Abbildung 6.19, rechts oben ist ein weißer Balken zu erkennen, der die Startpunkte einer Partikelverfolgung repräsentiert. Dieser Balken kann nun z.B. mit einer Maus gegriffen und versetzt werden. Die Ermittlung der neuen Partikelpfade erfolgt unmittelbar. Analoges geschieht beim Verändern der Position einer Schnittebene. Diese ist mit so genannten Griffen versehen, wie man sie z.B. in einem CAD-System zum interaktiven Transformieren von Objekten vorfindet. Diese Griffe bzw. der Balken von oben sind grafische Objekte der Szene und im Szenegraph von Abbildung 6.21 allgemein als

Controls bzw. Dragger bezeichnet. Auf den Einsatz von komplizierten Dialogen wird so weit es geht verzichtet. Der Fokus liegt hier vielmehr auf einem leicht zu erlernenden System.

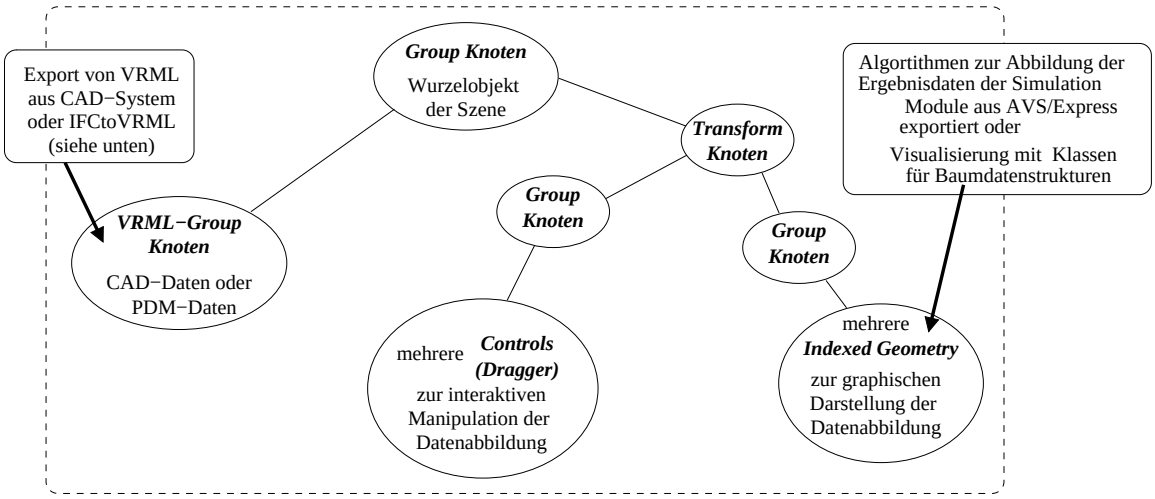

Abbildung 6.21: Kombinierte Visualisierung in Virtual Reality: Szenegraph und angegliederte Werkzeuge

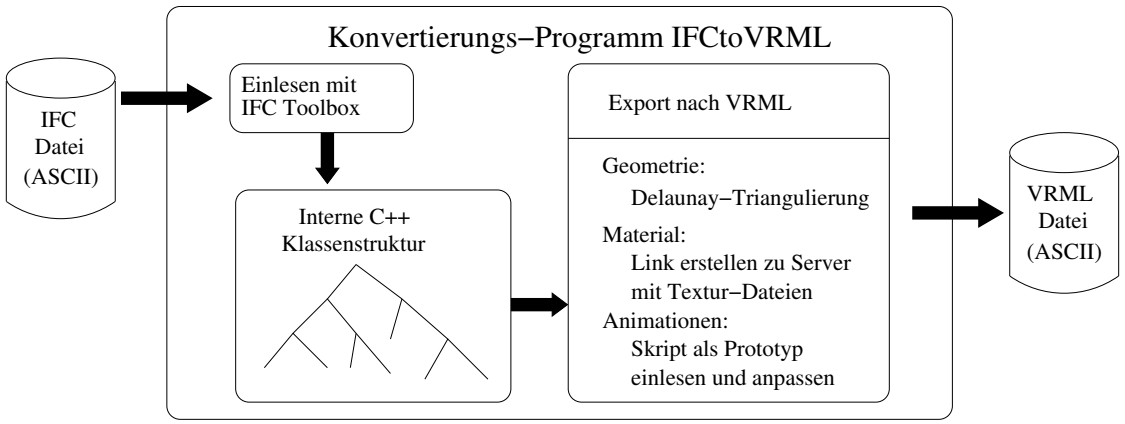

Abbildung 6.22: Ablauf der Konvertierung eines IFC-Datensatzes nach VRML

Anbindung von Produktmodelldaten an Virtual Reality

Ein entscheidendes Charakteristikum von Virtual Reality ist die Dynamik der nachgebildeten Szenen. Das bedeutet, dass man z.B. Objekte bewegen kann oder Animationen starten kann. Könnte man nun durch ein Gebäude wandern und Türen öffnen und schließen, so ist dies natürlich voll und ganz im Sinne von Virtual Reality. Architekturrundgänge haben diese Möglichkeit in der Tat. Dazu muss aber vorher ein CAD-Modell mühevoll nachbearbeitet werden.
Eine mögliche Alternative der Abbildung des geometrischen Modells in eine virtuelle Welt ist, dieses direkt aus einem Produktmodell zu übernehmen. Kommerzielle Programme können diesen Vorgang natürlich leisten, beschränken sich aber meist auf einen reinen Export der Geometrie.

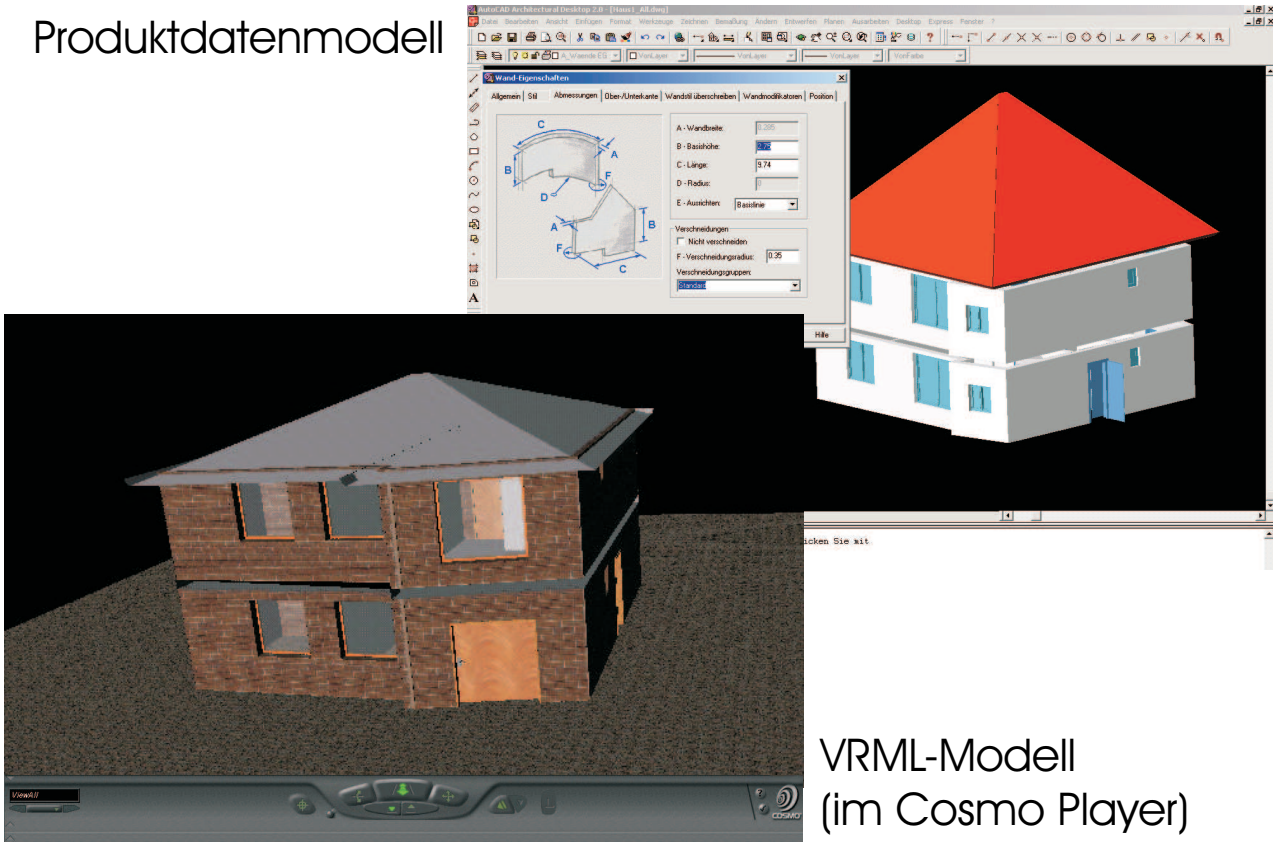

Abbildung 6.23: Übergang vom Produktdatenmodell nach VRML – Beispiel

Nachfolgend wird kurz ein Prototyp eines Moduls zur Konvertierung von Produktmodelldaten nach VRML vorgestellt. Ein Beispiel dazu ist in Abbildung 6.23 dargestellt. Neben den geometrischen Objekten im Produktmodell werden noch exemplarisch folgende Daten übernommen:

- Das Öffnen und Schließen von Türen und Fenstern wird als Animation in VRML generiert. Die Griffe werden mit einem Sensor versehen, der wiederum ein Skript zur Animation des Öffnungs- und Schließvorgangs startet.
- Die Materialien (z.B. Holz) der geometrischen Objekte im Produktmodell sind jeweils mit einer vordefinierten Textur verknüpft, die in VRML den Oberflächen der exportierten Geometrien zugewiesen werden. Für Präsentationen kann hier erheblich an Modellierungsaufwand gespart werden, weil die Zuweisung von Texturen zu den geometrischen Objekten in der Regel in einer aufwendigen Nachbearbeitung (siehe oben) durchgeführt wird.

Weiterhin wäre auch der Export von Beleuchtungsquellen im Raum möglich, die dann in der Virtuellen Welt näher beschrieben werden und zusätzlich mit einem Schalter versehen werden. Ebenso könnte man den Szenegraph der VRML-Szene entsprechend der Hierarchie der Raumstruktur im Produktmodell strukturieren.
Der Ablauf bei der Konvertierung ist in Abbildung 6.22 skizziert. Details zur Implementierung sind in [56] nachzulesen. Ein IFC-Modell befindet sich in einer ASCII-Datei in einem für Produktmodelle spezifischen Format, das mit der Programmier-Bibliothek Eurostep IFC Toolbox[3] eingelesen wird. Geometrische Objekte sind im Produktmodell in einer abstrakten Form wie *'Objekt*

[3]http://www.eurostep.com

Wand und zugeordnete Parameter' vorgehalten. Zur Visualisierung wird jedoch eine triangulierte Darstellung benötigt, die durch einen angeschlossenen Delaunay-Triangulierer erzeugt wird. An dieser Stelle muss bemerkt werden, dass noch wesentlich mehr geometrische Operationen durchgeführt werden müssten, womit eigentlich eine Bibliothek mit Operationen zur geometrischen Modellierung bzw. Manipulation eingesetzt werden müsste. Als Beispiel sei die Verschneidung einer Wand mit einer Dachschräge genannt: In diesem Prototypen würde man Wände erhalten, die über das Dach hinausragen.

6.6 Nachbereitung der Ergebnisse: Multimediale Dokumentation

Eine Visualisierung bzw. eine Analyse der Simulationsergebnisse ist unabhängig von der Umgebung (Desktop Visualisierungssystem wie AVS oder Virtual Reality Umgebung), stets ein interaktiver Prozess. Mit Hilfe verschiedener Aufbereitungstechniken (z.B. Reduktion der Dimension der Ergebnisse in Form von Schnittebenen) bei unterschiedlichen Beobachtungsperspektiven wird versucht, wesentliche Charakteristika des simulierten Problems zu extrahieren. Eigene Erfahrungen haben gezeigt, dass am Ende einer solchen Visualisierungssitzung bestenfalls eine Serie von unkommentierten Bildern bzw. Filmen entsteht und der Großteil der erfassten Ergebnisse verloren geht. Am Ende der numerischen Simulation bzw. der Auswertung der Ergebnisse muss aber in der Praxis stets ein Bericht für die weiteren Prozesse im Verlauf der Planung folgen.

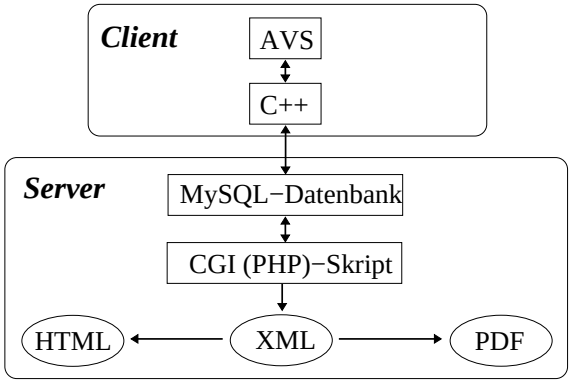

Abbildung 6.24: Software-Architektur zur Erstellung einer multimedialen Dokumentation

Daraus entsteht die Notwendigkeit, diese Ergebnisse sofort persistent zu halten. Der grundsätzliche Ansatz besteht darin, eine Software zu entwickeln, mit der man im Verlauf einer interaktiven Analyse von Ergebnissen numerischer Computersimulationen relevante Charakteristika des untersuchten Problems in einem multimedialen Dokument (z.B. im HTML-Format) ablegt. Ein Beispiel: Der Ingenieur findet einen Bereich von sehr großen Schubspannungen. Nun könnte er einen Schaltknopf drücken, mit dem er einen aktuellen Bildschirmschnappschuss in das Dokument mit einbinden lässt. Er wird gleichzeitig gefragt, welche Relevanz er diesem Problem zuordnet (z.B. "besonders kritisch") und aufgefordert, einen Kommentar hinzuzufügen.
Diese Idee wurde in [5] als Client-Server Architektur gemäß Abbildung 6.24 implementiert. Der Client kann in eine typische Applikation zur Datenvisualisierung eingebunden werden. Dies ist exemplarisch für AVS/Express implementiert (siehe Abbildung 6.25). Hier kann nun ein Anwender beispielsweise ein Bild mit der Darstellung der Stromlinien speichern und dieses in der

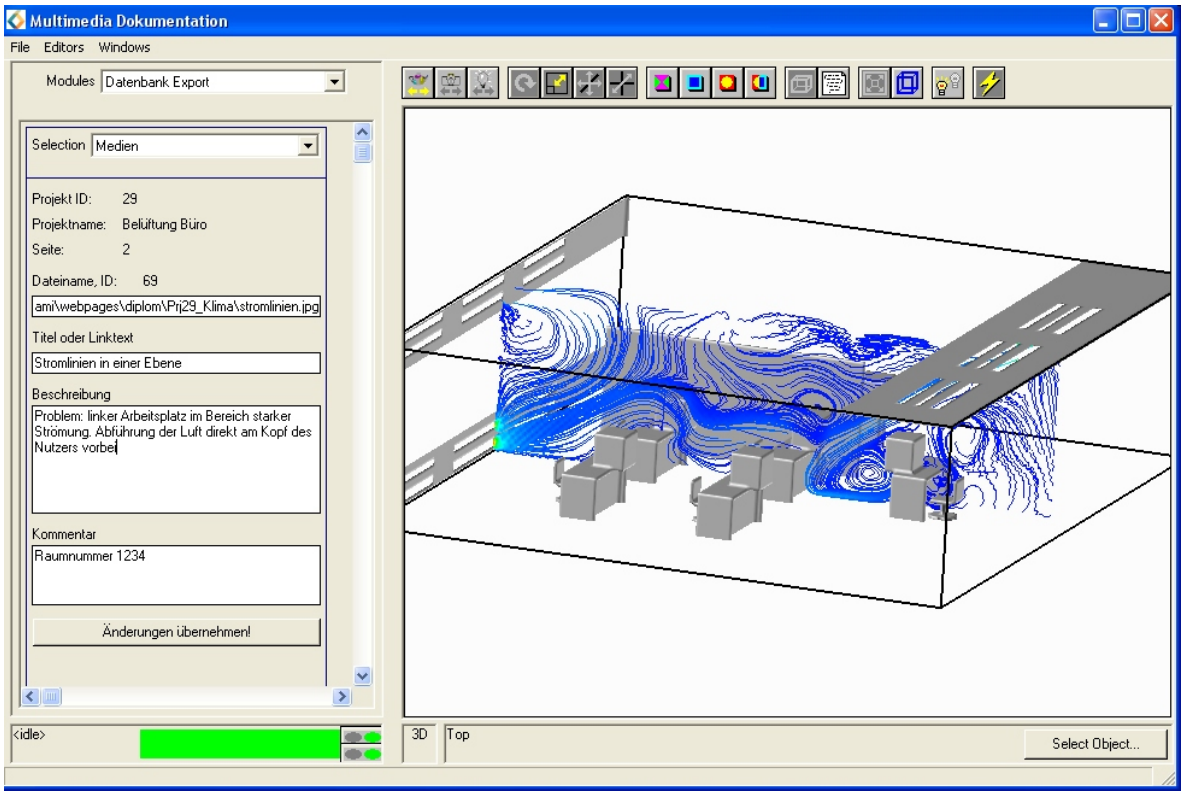

Abbildung 6.25: Ablegen von Informationen in eine Datenbank während einer Visualisierung – hier: in AVS/Express

Eingabemaske links (Abbildung 6.25) mit einem Kommentar versehen (im Beispiel ist dies die Bemerkung, dass der linke Arbeitsplatz im Bereich starker Strömung liegt). Dabei wird bereits während der Visualisierung die Struktur des zu generierenden Dokuments festgelegt. Die Daten können jederzeit in die angeschlossene Datenbank (hier: mySQL[4]) gespeichert werden und sind dann sofort verfügbar. Auf der Seite des Servers kann nun mit Hilfe eines CGI-Skripts die Dokumentation in den Formaten HTML oder PDF generiert werden. Abbildung 6.26 zeigt einen Ausschnitt des erzeugten HTML-Dokuments des Beispiels von Abbildung 6.25.
Die Ausgabe im PDF-Format ist selbstverständlich keine multimediale Dokumentation. Dieses Format ist in erster Linie als Druckversion vorgesehen.
Mit diesem Werkzeug entsteht der folgende Mehrwert: Erzeugte Bilder und Filme liegen nun geordnet vor und sind mit Bemerkungen und Kommentaren versehen, die das Verständnis dieser Medien verbessern. Weiterhin ist der Aufwand, eine derartige Rohversion zum fertigen Bericht zu vervollständigen, wesentlich geringer als den Bericht völlig von vorne zu erstellen.

6.7 Beispiel: Simulation in einem Großraumbüro

Abschließend wird der Ablauf der Simulation gemäß des Konzeptes von Abschnitt 5.2 (basierend auf Simulationen im Modus der Stapelverarbeitung) an einem komplexen Beispiel nochmal veranschaulicht.

[4]http://www.mysql.com

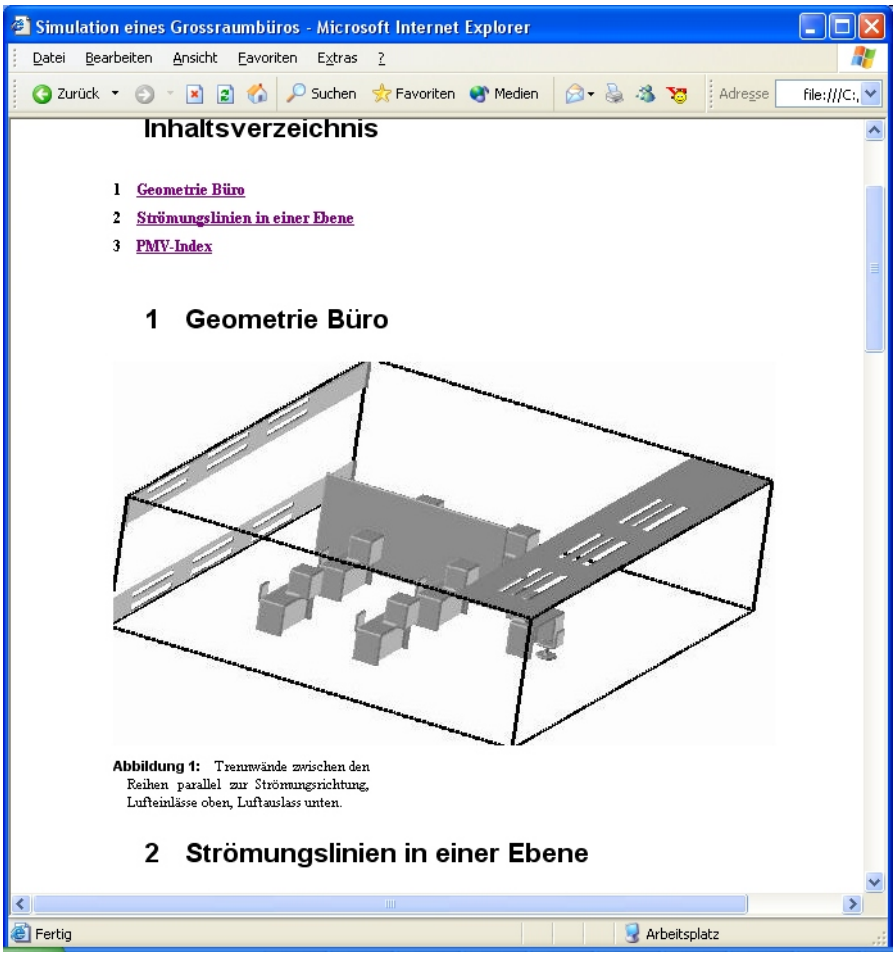

Abbildung 6.26: Automatisch generierte Dokumentation in HTML

Zuerst wird aus dem CAD-Modell ein Facettenmodell (Abbildung 6.27) extrahiert und daraus ein Octree (Abbildung 6.28) erzeugt, der das numerische Gitter abbildet.
Die Definition der Randbedingungen der Simulation ist in Abbildung 6.26 ersichtlich. Die Öffnungen an der Decke und an der linken Wand oben sind Einlassöffnungen während die Öffnungen an der linken Wand unten Absaugöffnungen sind.
Die Berechnung wurde nach der Lattice-Boltzmann Methode auf der Hitachi SR8000-F1 durchgeführt. Bei einer Diskretisierung des Strömungsgebiets von $302 \times 78 \times 252$ Gitterknoten erfolgte die Berechnung mit einem Large-Eddy-Turbulenzmodell. Die Reynoldszahl der Strömung war im Bereich von 50.000, also einem für Innenraumströmungen realistischen Bereich.
Die Ergebnisdaten der Simulation wurden auf etwa 11 % des ursprünglichen Gitters ausgedünnt, ohne einen sichtbaren Verlust an Informationen zu erhalten (siehe Abbildung 6.14, 6.15 und 6.16).
Anschließend wurden Kriterien des menschlichen Komfortempfindens betrachtet. Abbildung 6.18 zeigt hier ein Bild von einer Parameterstudie des PMV-Index.
Natürlich lassen sich die Ergebnisse der Simulation in einer VR Umgebung in Form einer kombinierten Darstellung mit den CAD-Daten analysieren (siehe Abbildung 6.20).
In einem abschließenden Schritt wurde noch ein Dokument einer multimedialen Präsentation erzeugt (siehe Abbildung 6.25 und 6.26).

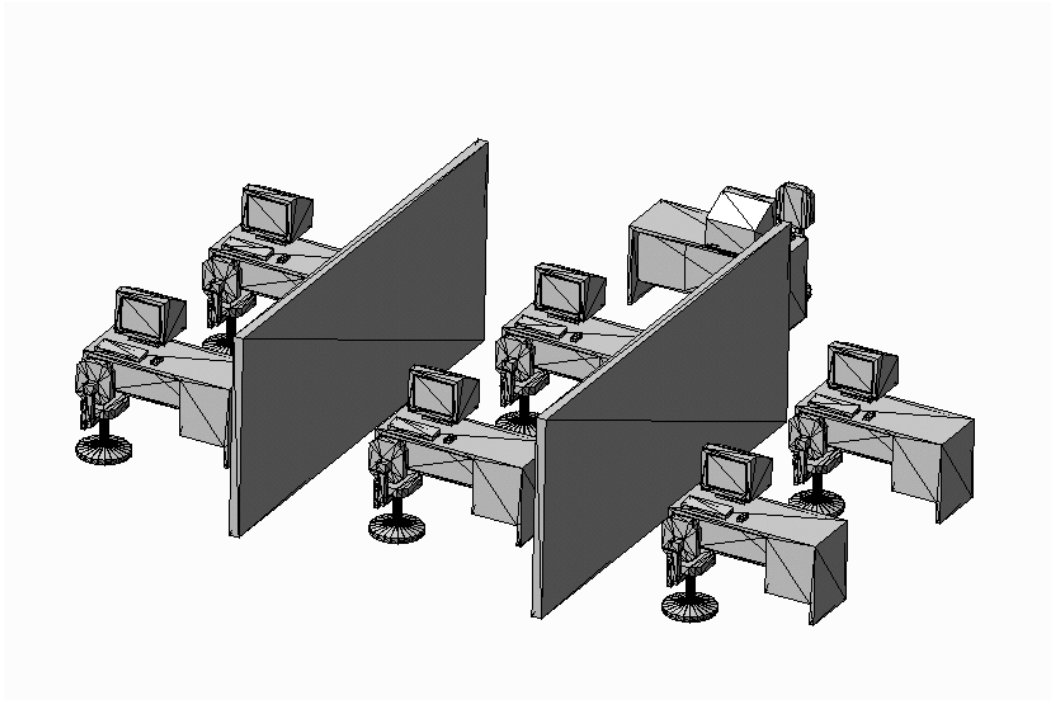

Abbildung 6.27: Beispiel Großraumbüro: Geometrie als Facettenmodell

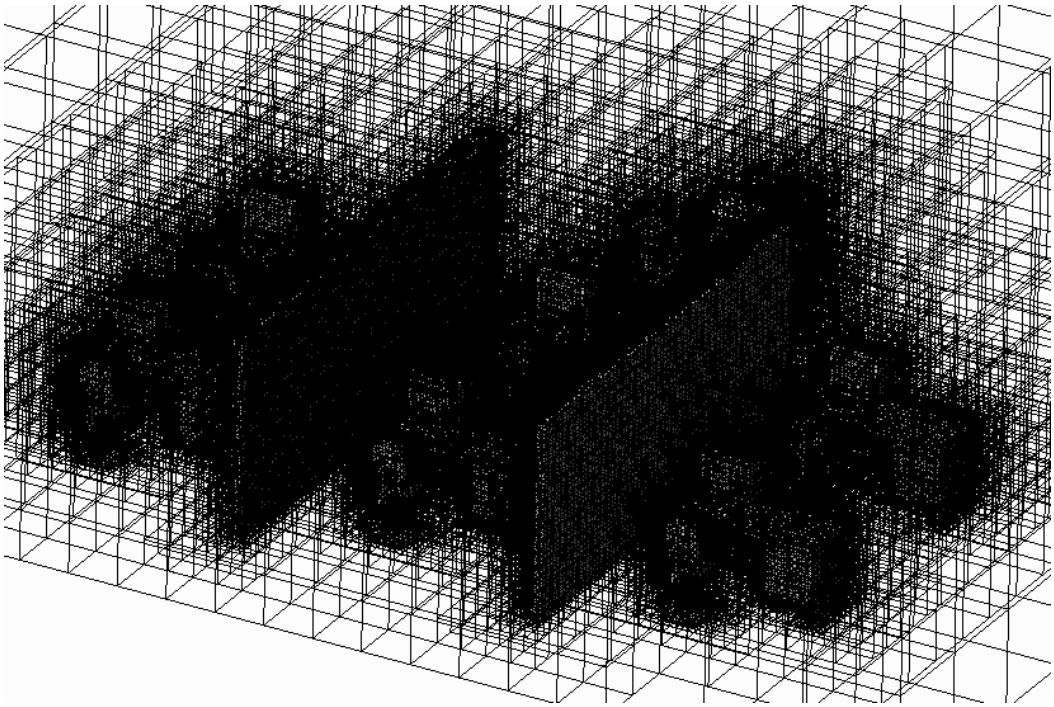

Abbildung 6.28: Strömungsgebiet im Großraumbüro: Octree-Darstellung

Kapitel 7

Interaktive Steuerung numerischer Strömungssimulationen: Der computergestützte Windkanal VFReal

In diesem Kapitel wird ein auf der Lattice-Boltzmann Methode basierender Prototyp einer interaktiven Strömungssimulation (*Online-Version* der Datenauswertung) vorgestellt. Nimmt man wiederum Bezug zum Titel dieser Arbeit, so findet man hier die Aspekte, die der Begriff *interaktive Steuerung* impliziert. Es wird jedoch betont, dass es sich hierbei noch nicht um ein System handelt, das produktiv im Sinne der zweistufigen Lösungsstrategie von Abschnitt 5.3 einzusetzen ist. Vielmehr geht es darum, die Machbarkeit dieses Systems zu zeigen und ein Grundgerüst zu schaffen, das in anschließenden Projekten flexibel bis zu einem produktivem Status erweitert werden kann.
Zunächst erfolgt der Entwurf eines System, das eine Komponente zur Eingabe und Visualisierung über eine Interprozesskommunikation mit einem Supercomputer bzw. einen Workstation-Cluster verbindet. Damit werden die Problemdefinition (physikalische Ebene gemäß Abschnitt 5.2), die Berechnung (numerische Ebene) und die Auswertung (Ebene der Präsentation) in einer Applikation zusammengeführt (Abschnitt 7.1). Anschließend werden die einzelnen Komponenten (Abschnitte 7.2 für die Simulation und 7.3 für die Ein- und Ausgabe) sowie die Schnittstellen detailliert erläutert. Das Kapitel wird mit Beispielen und einer Bewertung des Systems abgeschlossen (Abschnitt 7.5).

7.1 Systementwurf

7.1.1 Analyse der Anwendungsfälle

Aus der Sicht des Anwenders kann man sich das Grundprinzip einer interaktiven Strömungssimulation, wie in Abbildung 7.1 skizziert, folgendermaßen vor Augen führen:
Auf der einen Seite hat man ein numerisches Lösungsverfahren, das typischerweise höchste Ansprüche an die Rechenleistung erfordert. Aus diesem Grund erfolgt die Berechnung – wie in Abbildung 7.1 (rechts) dargestellt – auf einem massiv parallelen Supercomputer (siehe Abschnitt 3.2). Dem gegenüber steht ein leistungsfähiges System zur Datenauswertung. In dieser Arbeit soll eine Virtual Reality Umgebung – wie in Abbildung 7.1 (links) dargestellt – zum Einsatz kommen. Diese beiden, im klassischen Verständnis einer numerischen Simulation klar getrennten Subsystem, werden jetzt durch eine bidirektionale Kommunikation miteinander verbunden. Die Art und der Umfang der zu übertragenden Daten werden im weiteren Verlauf dieses Kapitels noch herausgearbeitet, dennoch ist bereits hier klar, dass vom Teilsystem der Datenauswertung zum Teilsystem der Berechnung Parameter zur Steuerung und umgekehrt Ergebnisdaten transferiert werden.

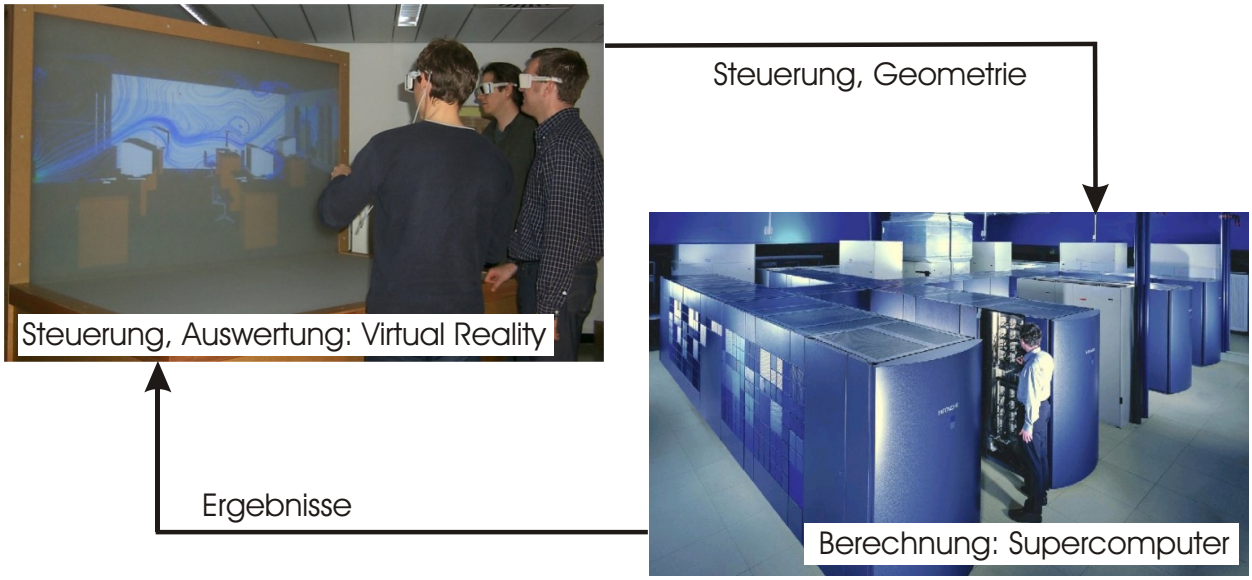

Abbildung 7.1: Grundprinzip einer interaktiv zu steuernden Simulation

Nun wird das System methodisch entwickelt. Die Analyse umfasst neben der Zielsetzung (wurde in Abschnitt 5.3 behandelt) die Untersuchung möglicher Anwendungsfälle, die letztlich zur Bildung der Komponenten (siehe Abschnitt 7.1.2) des Systems führt.
Im Rahmen eines interaktiven Eingriffs während einer laufenden Strömungssimulation kann man die in Abbildung 7.2 dargestellten Anwendungsfälle, inklusive der damit ausgelösten Prozessketten wie folgt identifizieren:

1. Der Benutzer ändert einen Parameter zur Steuerung der Simulation: Ein solcher Parameter kann z.B. eine physikalische Größe wie die Viskosität sein oder das Ändern des Schwellwertes der Konvergenz. Diese Parameter werden dann dem Simulationskern mitgeteilt, der entsprechend neue Ergebnisse liefern muss. Gegebenenfalls werden die Datenabbildungen sowie der zugehörige Szenegraph erneuert und schließlich neu gerendert.

2. Der Benutzer navigiert durch die Szene: In diesem Fall wird nach der Erfassung der Eingabe lediglich eine neue Perspektive ermittelt und erneut graphisch dargestellt. Der Datenbestand der Applikation ändert sich nicht.

3. Die geometrischen Objekte im Strömungsgebiet werden manipuliert: Hier tritt nach der Erfassung der graphisch-interaktiven Eingabe eine Verzweigung auf. Zunächst ist jedes geometrische Hindernis natürlich ein Objekt der 3D-Welt, womit der Szenegraph erneuert wird. Gleichzeitig führt dieser Eingriff zu einer Änderung des diskreten Gitters der numerischen Simulation. Nach der Erneuerung des Gitters werden neue Daten der Simulation berechnet und anschließend als graphische Objekte abgebildet. Die Verzweigung des Ablaufs endet mit dem abschließenden Rendering.

4. Der Benutzer verändert Randbedingungen: Nach der Erfassung dieser Interaktion wird zuerst das numerische Gitter erneuert. Für die betroffenen Punkte wird eine Markierung zur Beschreibung des Typus der Randbedingung (*marker and cell*, siehe Abschnitt 3.1.2) verändert. Die Folge ist eine neue Berechnung der Simulation und damit eine Neuberechnung der Datenabbildung und des Szenegraphen.

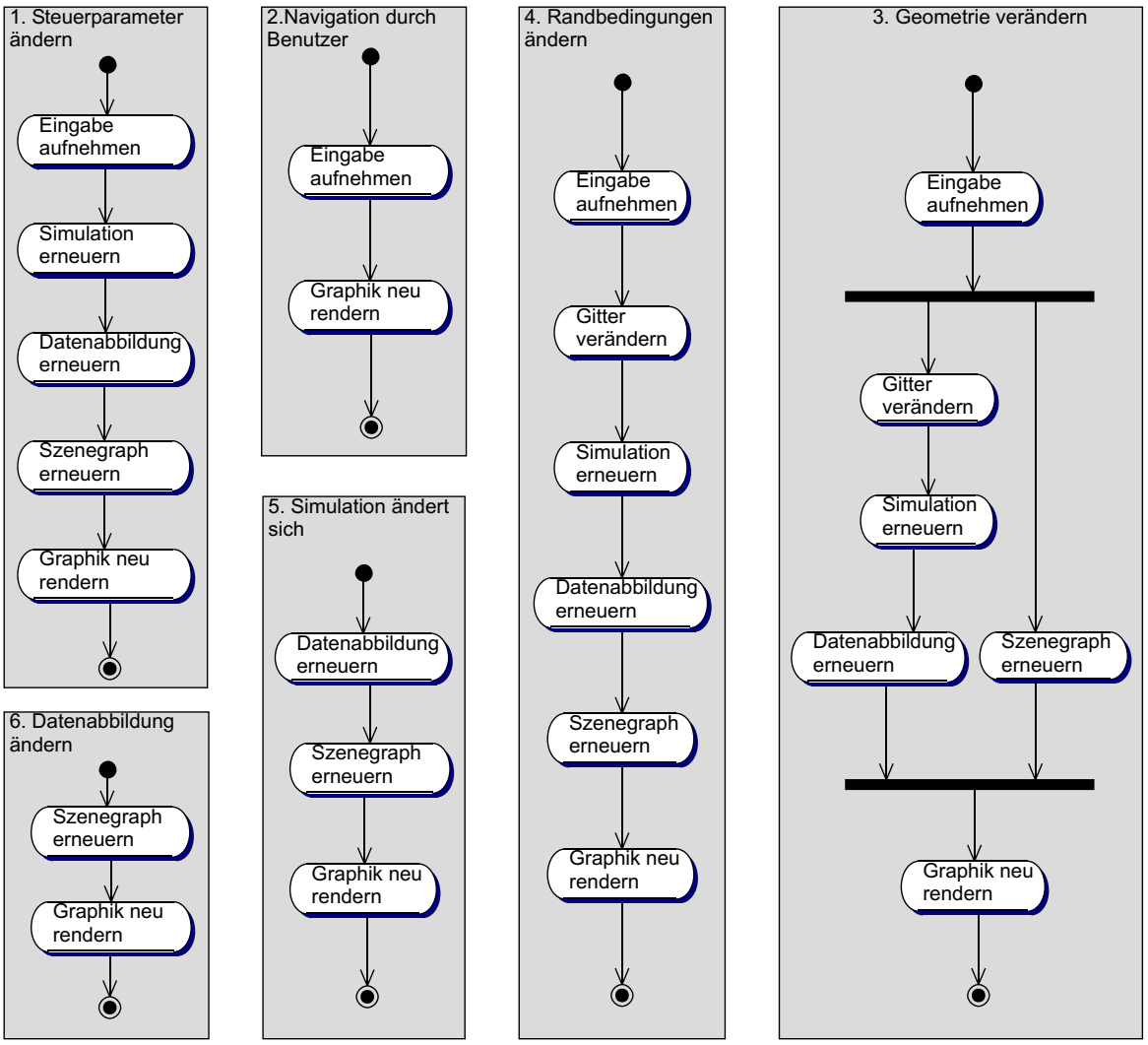

Abbildung 7.2: Anwendungsfälle innerhalb einer interaktiven Strömungssimulation

5. Die Simulation liefert neue Ergebnisse: Dieser Vorgang läuft ständig und ohne Eingriff des Benutzers ab, weil der Simulationskern stets explizit neue Zeitschritte berechnet. Die Folgeprozesse von der Erneuerung der Datenabbildung und des Szenegraphen bis hin zum Rendering wurden bereits beschrieben.

6. Der Benutzer ändert die Abbildung der Daten: In diesem Fall möchte der Anwender beispielsweise eine Schnittebene an eine andere Position setzen oder eine Isofläche neu erzeugen. Nach der Berechnung dieser Abbildungen folgt nur noch die Erneuerung des Szenegraphen und das Rendering.

7.1.2 Identifikation von Komponenten

Aus den im vorhergehenden Abschnitt erfassten Szenarios der Anwendung einer interaktiven Strömungssimulation können die in Abbildung 7.3 skizzierten Komponenten, inklusive der gegenseitigen Beziehungen, festgehalten werden.
Die folgenden Ausführungen erläutern diese Komponenten und begründen die gewählte Zuordnung zum Teilsystem der Visualisierung bzw. der Simulation:

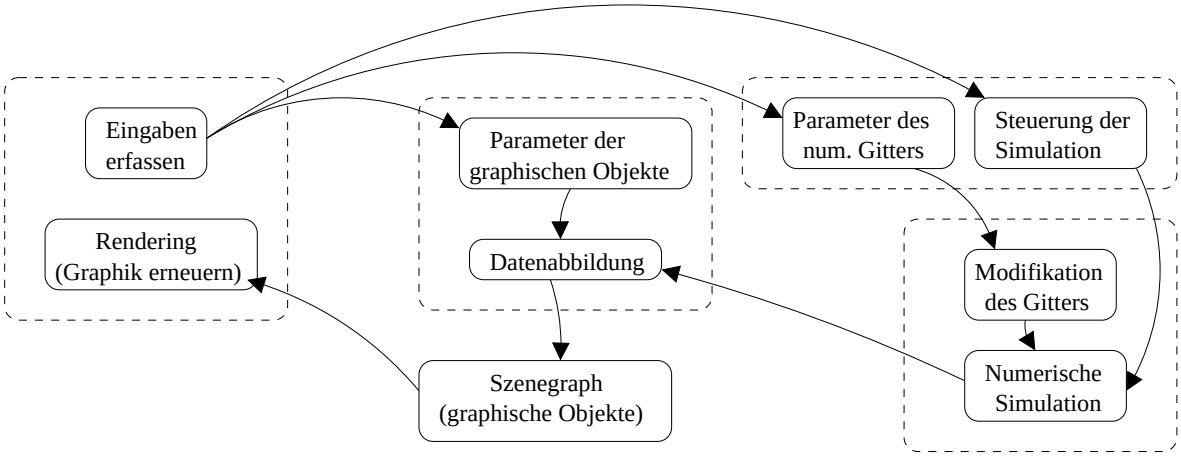

Abbildung 7.3: Komponenten einer interaktiven Strömungssimulation

Eingabe und Rendering: In jedem vom Benutzer getriebenen Anwendungsfall steht die Erfassung der Eingabe, also die Schnittstelle zwischen Mensch und Maschine. Im Fall von Virtual Reality ist diese Erfassung nicht trivial, vor allem wenn man bestrebt ist, ein System zu schaffen, das ohne großen Aufwand auf andere Systeme zu portieren ist. Anschaulich ausgedrückt: Was muss man tun, um gleichermaßen einen Datenhandschuh wie eine 3D-Maus zu unterstützen? Ebenso problematisch ist das Rendering am Ende eines Anwendungsfalls. Auch hier kann die Mensch-Maschine Schnittstelle sehr unterschiedlich konfiguriert sein (z.B. Desktop – Holobench – CAVE^{TM}). Es sei an dieser Stelle vorweg genommen, dass die beschriebene Problematik mit der Wahl einer geeigneten Programmier–Bibliothek (siehe Abschnitt 7.1.3) sehr leicht bewältigt werden kann. Die Komponenten *Eingabe* und *Rendering* werden zur Komponente *Virtual Reality-Schnittstelle* zusammengefasst und gehören auf jeden Fall zum Subsystem der Visualisierung.

Szenegraph: Eine Virtual Reality basierte Datenhaltung geschieht typischerweise mit Hilfe eines Szenegraphen (siehe Abschnitt 2.2.4). Dieser wird lokal für das Rendering vorgehalten.

Simulation: Die numerische Berechnung erfolgt selbstverständlich innerhalb des dafür vorgesehenen Teilsystems unter Umständen mit einer Vielzahl parallel laufender Prozesse. In Abschnitt 7.2.2 folgen Überlegungen, wie dieses Teilsystem aufgebaut sein soll (siehe dazu auch die beiden alternativen Architekturen in den Abbildungen 7.5 und 7.6).

Gittergenerierung: Diese Komponente koppelt sich sehr eng an die Simulation, so dass sie auch in das Teilsystem der Berechnung ausgelagert wird. Dies hat den Vorteil, dass nur Parameter von geometrischen Objekten vom Teilsystem der Ein- und Ausgabe versendet werden müssen und nicht das komplette Gitter. In diesem Prototypen handelt es sich noch um eine recht einfache, auf Ideen aus der Rastergrafik (siehe Abschnitt 7.2.6) basierende Gittergenerierung, die aber für einfache geometrische Grundobjekte (z.B. Konus, etc., siehe Abschnitt 7.2.1) sehr effizient arbeitet. Wird dieser Teilprozess später mit einem allgemein anwendbaren Verfahren (z.B. mit Oktalbäumen, siehe [25, 53]) durchgeführt, dann sind dafür sehr hohe Rechenleistungen erforderlich, die auf dem Teilsystem der Berechnung zur Verfügung gestellt werden können.

Datenabbildung: Hier stellt sich die Frage einer geeigneten Partitionierung gemäß der Ansätze

von Abbildung 2.2, Abschnitt 2.1.2. Erfolgt die Datenabbildung auf der Seite der Berechnung, so ist zunächst einmal der Vorteil der schnellen Berechnung durchaus gewichtig. Mit diesem Ansatz ist es dann auch nicht mehr nötig, die gesamten Ergebnisdaten der Simulation zum System der Visualisierung zu senden. Stattdessen werden nur noch die abgebildeten Daten zur Weiterverarbeitung mit dem Szenegraphen gesendet. Bei interaktiven Simulationen kann allerdings nicht davon ausgegangen werden, dass sich das Volumen der zu transferierenden Daten damit verringert (einfaches Beispiel: der Anwender hat gerade 10 Isoflächen mit verschiedenen Schwellwerten instantiiert). Zusätzlich ist damit zu rechnen, dass bei einer Anfrage einer neuen Datenabbildung eine Latenz durch den Transfer über eine Netzwerkverbindung entsteht.

Demgegenüber ist es sehr wichtig, dass ein Benutzer die Auswertung der Daten möglichst interaktiv und intuitiv durchführen kann. Weiterhin basieren die Datensätze einer interaktiven Simulation gemäß Abschnitt 5.3 auf einer Gittergröße von maximal 10^6 diskreten Punkten. In diesem Bereich ist eine annähernd echtzeitfähige Abbildung der Daten möglich, so dass die Komponente der Datenabbildung in das Teilsystem der Ein- und Ausgabe integriert wird.

7.1.3 Auswahl der Werkzeuge zur Implementierung

Zur Implementierung des oben skizzierten Systems werden nun geeignete Werkzeuge ausgewählt, um die Funktionalität des Systems zu ermöglichen:

- Die numerische Simulation läuft parallel auf einem Supercomputer oder alternativ auf einem Workstation-Cluster. Die Interprozesskommunikation wird mit dem Standard MPI[1] umgesetzt.

- Der Datenaustausch zwischen dem Teilsystem der Ein- und Ausgabe mit dem Teilsystem der Berechnung kann gewöhnlicherweise nur dann mit MPI erfolgen, falls beide Systeme mit dem selben Betriebssystem arbeiten. Diese Einschränkung wird durch die Verwendung der dafür ausgerichteten, speziellen MPI-Implementation PACX-MPI[2] umgangen.

- Die Seite der Virtual Reality basierten Steuerung könnte idealerweise mit einem bereits vorhandenen System zur Integration von Applikationen im Bereich Computational Steering entwickelt werden. Akademischen Entwicklungen wie SCI-Run fehlt dabei meistens die Anbindung an Virtual Reality und die Plattformunabhängigkeit. Beim kommerzielle System COVISE fehlt lediglich der zuletzt genannte Aspekt.

  Stattdessen wird eine Kombination aus den Programmier-Bibliotheken TGS Open Inventor[3], VRJuggler[4] und gegebenenfalls VTK[5] gewählt.

- Der Szenegraph wird mit der Bibliothek Open Inventor implementiert, die sich sehr gut zur schnellen Entwicklung von Applikationen (*Rapid Application Development*) eignet. Dabei wird die kommerzielle Version von TGS verwendet, weil diese eine wesentlich bessere Performance als die originale Version von SGI zeigt und vor allem die hier unerlässliche

[1] http://www.mpi-forum.org
[2] http://www.hlrs.de/organization/pds/projects/pacx-mpi
[3] http://www.tgs.com
[4] http://www.vrjuggler.org
[5] http://www.kitware.com

Erweiterung der Manipulation des Szenegraphen durch mehrere konkurrierende Threads (siehe Abschnitt 7.3.5) enthält. Ein interessanter Baustein von Open Inventor sind die so genannten *Dragger* (siehe Abschnitt 7.3.4). Das sind intuitiv bedienbare, dreidimensionale Steuer-Objekte zum Transformieren geometrischer Objekte.

Zusätzlich enthält der Open Inventor von TGS die Erweiterung *DataViz* zur Abbildung der Ergebnisse der Simulation. Vergleichend soll hierbei die Bibliothek Visualization Toolkit (VTK) untersucht werden. VTK hat eine wesentlich breitere Palette an Algorithmen zur Datenabbildung, gilt jedoch als relativ langsam.

- VRJuggler ist eine Bibliothek zur Entwicklung von portablen Virtual Reality basierten Applikationen, die hier zur Erfassung der Eingaben und der Ansteuerung der Projektionsflächen verwendet wird. Ein Beispiel: Ein Anwender bewegt einen Zeigestift im physikalischen Raum. Der zugehörige Treiber liefert die Position und die Lage des Stifts, die jetzt noch mit den Koordinaten des eigenen Szenegraphen in Verbindung gebracht werden muss. Diese Transformation wird auf jeder VR-Hardware anders ausfallen. An dieser Stelle setzt VRJuggler ein, der alle geräteabhängigen Komponenten einer Applikation kapselt. Die Portierung auf eine andere Virtual Reality Anlage erfolgt einfach durch Anpassen einer Konfigurations-Datei. Analog stellt VRJuggler die nötige Zwischenschicht zur Ansteuerung von mehreren Projektionsflächen dar. In diesem Kontext kann VRJuggler auch für den Einsatz auf einem Desktop-Monitor konfiguriert werden.

7.2 Teilsystem Simulation

7.2.1 Funktionsumfang

Die Grobspezifikation des Funktionsumfangs der Simulation ergibt sich aus den Anforderungen von Abschnitt 5.3. Die folgenden Punkte präzisieren die umgesetzte Implementierung:

Methode der Simulation: Der Berechnungskern basiert auf der Lattice-Boltzmann Methode (siehe Abschnitt 3.1.2). In der aktuellen Implementierung ist noch kein Modell eingebaut, mit dem durch Temperatur getriebene Strömungen berechnet werden können. Gegenwärtig kann man zwischen dem Modell D3Q15 und D3Q19, jeweils mit dem LBGK-Verfahren und der Momentenmethode [62, 30] wählen. Die Simulation läuft parallel mit der in Abbildung 7.4 dargestellten und in Abschnitt 7.2.4 erklärten Gebietsaufteilung.

Turbulenzmodelle sind gegenwärtig nicht integriert. Es wäre nicht sonderlich aufwendig, den Kollisionsoperator entsprechend zu modifizieren. Das Problem tritt allerdings bei der Ermittlung der Ergebnisse auf. Diese sind nämlich in einer Mittelung über mehrere Zeitschritte zu berechnen. In einer interaktiven Simulation kann es aber vorkommen, dass innerhalb dieses Zeitabschnitts geometrische Objekte verändert wurden, also eine völlig neue Problemstellung vorliegt. Hierzu sind noch weitere Überlegungen nötig, die über den Umfang dieser Arbeit hinausgehen.

Struktur des numerischen Gitters: Das numerische Gitter ist in Form einer dreidimensionalen, vollbesetzten Matrix implementiert. Damit sind vor allem Modifikationen des Gitters leicht durchzuführen und die Implementierung ist einfacher und schneller zu realisieren.

Geometrische Objekte: Als Hindernisse im Strömungsgebiet können die dreidimensionalen geometrischen Grundobjekte

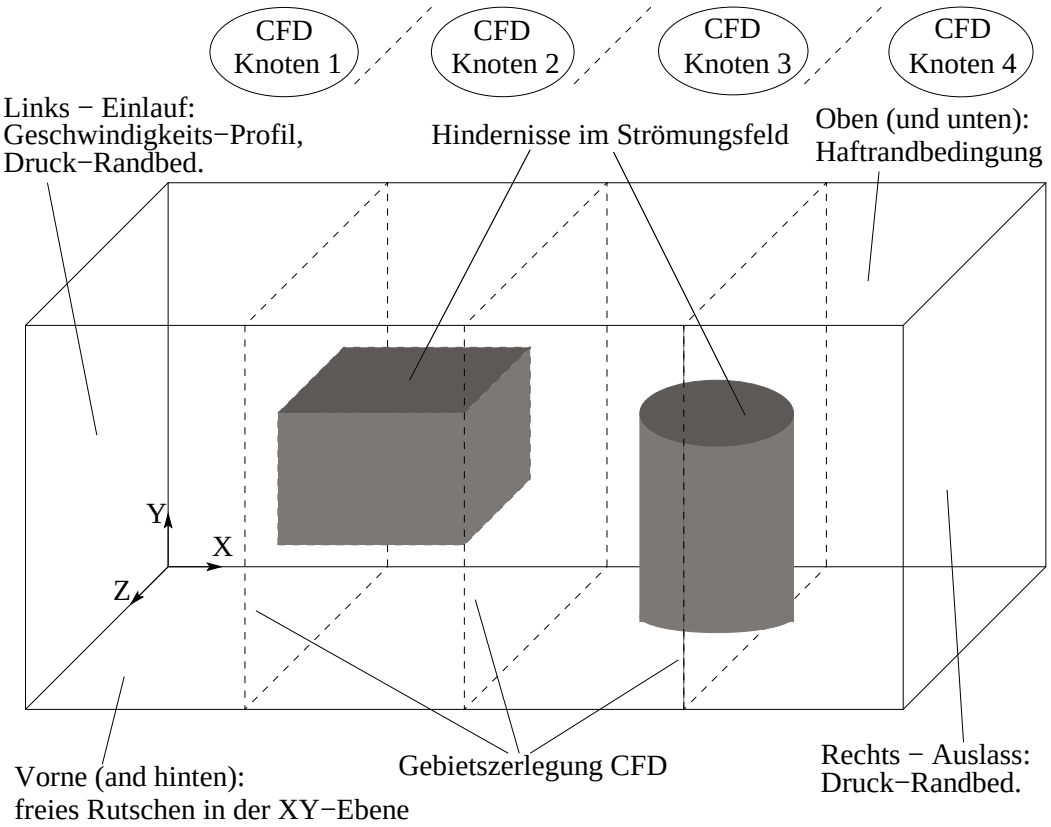

Abbildung 7.4: Aufteilung des Systems – Randbedingungen, Hindernisse und Gebietszerlegung (in diesem Beispiel: 4 Domänen)

Hexaeder – Ellipsoid – Zylinder – Konus – Pyramide

eingesetzt werden. Diese können dann beliebig miteinander zu komplexen Körpern zusammengesetzt werden. Auch hier soll zunächst die prinzipielle Machbarkeit gezeigt werden. In zukünftigen Programm-Versionen werden dann B-Rep Beschreibungen beliebiger Körper verwendet.

Randbedingungen: In der ersten Version der interaktiven Simulation wurde größter Wert auf eine möglichst stabile Simulation gelegt. Dazu wird ein Windkanal gemäß Abbildung 7.4 nachgebildet. An der linken Öffnung ist ein parabolisches Einströmprofil der Geschwindigkeit vorgegeben. Die maximale Geschwindigkeit dieses Profils kann während der Laufzeit der Simulation vom Benutzer geändert werden. Ebenso wird der Druck an der linken und der rechten Öffnung vorgegeben. Der damit modellierte Druckgradient kann ebenso während der Berechnung verändert werden. Der Boden und die Decke sowie die geometrischen Objekte im Strömungsgebiet sind mit einer Haftrandbedingung versehen. Die vordere und die hintere Seite des Kanals sind als freie Rutschbedingungen in der XY-Ebene modelliert, damit aus den besagten Gründen der Stabilität das Profil im Einlauf vorgegeben werden kann.

Ausgabe der Ergebnisse der Simulation: Das wesentliche Merkmal einer interaktiven Simulation ist die Übertragung der Ergebnisse an eine Komponente zur Visualisierung während der Laufzeit. Gleichzeitig kann das Feld der Geschwindigkeit und des Druckes in eine Datei

geschrieben werden (im Format von AVS/Express). Optional ist es noch möglich, die Verteilungsfunktionen der Lattice-Boltzmann Simulation in regelmäßigen Abständen in einer Datei zu sichern. Nachdem dies die Primärvariablen der Simulation sind, kann man damit vollständige Zustände für bestimmte Zeitpunkte wieder herstellen. Weiterhin gibt die Komponente der Simulation stets Informationen bezüglich der Stationärität der Strömung sowie der Anzahl der Gitterknoten-Erneuerungen (als Maß für die Leistungsfähigkeit der Simulation) zurück.

Steuerung der Simulation: Während der Laufzeit können geometrische Hindernisse neu erzeugt, modifiziert und wieder gelöscht werden. Weiterhin können die Viskosität, sowie die Parameter des Einströmprofils und des Druckgradienten (siehe oben) verändert werden. Zusätzlich können jeweils die Intervalle eingestellt werden, innerhalb derer die Ergebnisse an die Grafik gesendet bzw. in eine Datei geschrieben werden.

7.2.2 Architektur des Systems

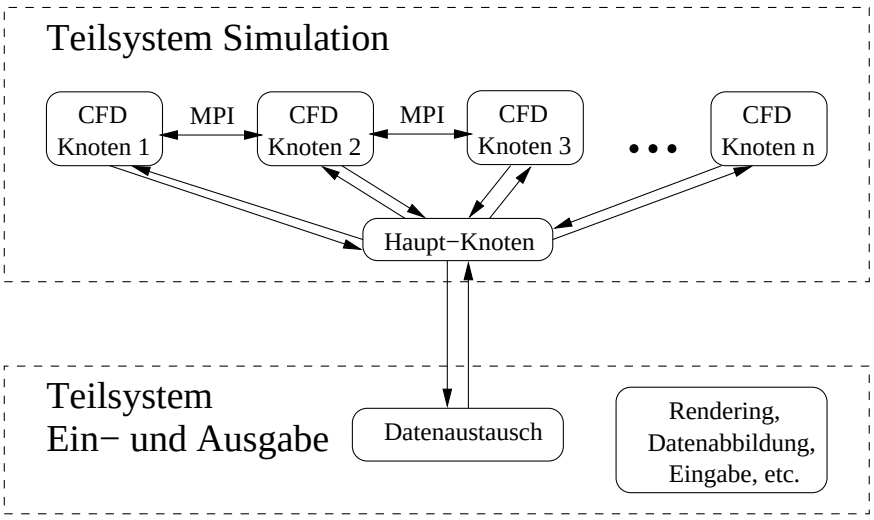

Abbildung 7.5: Systemarchitektur Alternative *MasterSim*: Kommunikation zwischen der Simulation und der Ein- und Ausgabe über einen Hauptknoten

Bei der parallelen Lattice-Boltzmann Simulation stellt sich nun die Frage, ob man innerhalb des Subsystems Simulation noch einen Hauptprozess – wie in Abbildung 7.5 skizziert – vorhalten sollte, um die Ergebnisse der einzelnen Prozesse zu sammeln und erst dann an die Komponente der Visualisierung zu senden.
Die zweite Variante wäre, die Ergebnisse direkt von den einzelnen Prozessen an die Komponente der Visualisierung zu senden (siehe Abbildung 7.6). In diesem Fall kann man ebenso von einem Hauptprozess sprechen, nur ist dieser dann voll in die Komponente der Visualisierung integriert. Im weiteren Verlauf werden nun diese beiden Architekturen gegenübergestellt. Das System mit dem Hauptprozess auf der Seite der Simulation wird als Alternative *MasterSim* bezeichnet, die andere Variante wird Alternative *MasterVis* genannt.

- Die Schnittstelle zwischen der Simulation und der Ein- und Ausgabe ist im Fall *MasterSim* relativ schlank, womit die Implementierung einfach zu modularisieren ist. Im anderen Fall

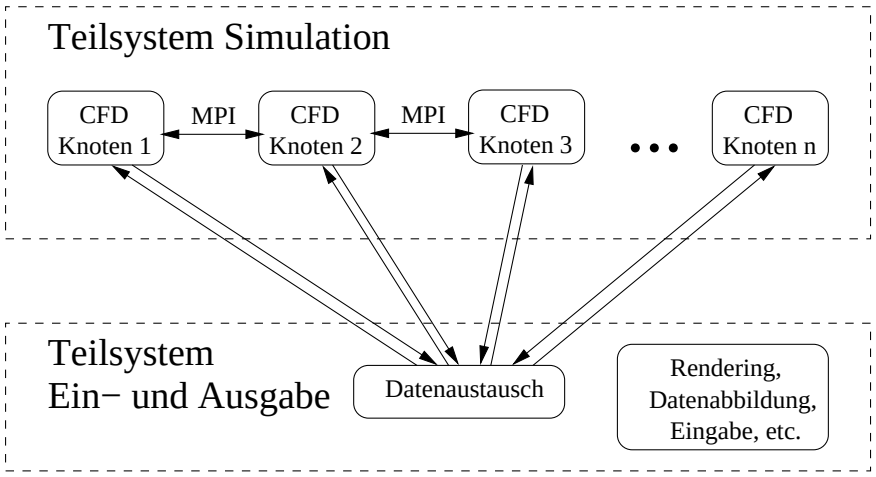

Abbildung 7.6: Systemarchitektur Alternative *MasterVis*: Direkte Kommunikation zwischen den Prozessen der Simulation und der Ein- und Ausgabe

ist der Steuerprozess der parallelen Simulation ein Teil der Ein- und Ausgabe. Dadurch entsteht ein monolithisches System, das zunächst zwar leichter und schneller implementiert werden kann, aber langfristig schwer zu warten und wenig flexibel ist.

- Ein Hauptprozess auf der Seite der Simulation hat den Nachteil, dass noch eine zusätzliche Schicht der Datenübertragung vorhanden ist. An dieser Stelle muss man genau die vorhandene Hardware berücksichtigen:

  Auf einem Supercomputer findet die Übertragung von den Rechenprozessen zum Hauptprozess im internen Bus-System statt. Dieser Vorgang der Kommunikation ist sehr effizient. Für den Fall, dass die Simulation auf einem Workstation-Cluster stattfindet, kann man den Hauptprozess und den Prozess der Ein- und Ausgabe auf der selben Workstation starten. Der zusätzliche Datenaustausch wird nun ebenfalls innerhalb des schnellen Bus-Systems einer Maschine ablaufen. In beiden Fällen entsteht damit zumindest kein gravierender Overhead.

- Die Variante *MasterVis* ist im Falle der Verwendung eines Supercomputers kritisch zu untersuchen, weil mehrere Übertragungsvorgänge vom Supercomputer zu einen externen Rechner bzw. umgekehrt stattfinden. Jeder dieser Vorgänge ist durch eine signifikante Latenzzeit charakterisiert, womit gemäß Gleichung (3.31) ein potentieller Nachteil gegenüber der Variante *MasterSim* entsteht.

Im Sinne der Flexibilität ist die Alternative *MasterSim* zu bevorzugen. Dennoch wurden vorerst beide Alternativen implementiert, um eine endgültige Wahl der Architektur von der Messung der Leistung des Gesamtsystems abhängig zu machen. Die Ergebnisse dieser Messung werden in Abschnitt 7.2.5 präsentiert. Dort wird eine Algorithmik vorgeschlagen, mit der man die Daten zwischen der Simulation und der Visualisierung möglichst effizient austauscht. Sofern nicht explizit etwas anderes angegeben ist, wird in den weiteren Ausführungen allgemein von einem Hauptprozess gesprochen, unabhängig davon, ob dieser ein Teil der Simulation oder der Visualisierung ist.

7.2.3 Datenhaltung

Es liegt natürlich nahe, eine komplexe Software wie hier vorgestellt in C++ zu implementieren. Die Anwendung der Objektorientierung muss jedoch sehr bedacht geschehen, weil ein sehr zeitkritisches System vorliegt. Nachdem aber auf dem verwendeten Supercomputer Hitachi SR8000 gegenwärtig (Februar 2003) weiterhin nur eine Beta-Version des für die Hardware optimierten C++ Compiler vorliegt, ist die Komponente der Simulation in C implementiert. Dennoch sind die Datenstrukturen selbst objektorientiert gehalten.

Die Struktur `lbSimSystem` – allgemeine Informationen zur Simulation

In dieser Datenstruktur werden alle Informationen zur Simulation gehalten, die sowohl auf dem Hauptprozess und auf den Teilprozessen (im weiteren Verlauf auch als *Slaves* bezeichnet) gehalten werden und demzufolge auch untereinander ausgetauscht werden. Die nachfolgende Tabelle gibt eine kurze Beschreibung dieser Daten:

zu empfangende Daten	Bedeutung
`finishSim`	Steuerparameter zum Anhalten der Simulation
`model`	Beschreibung des verwendeten Lattice-Boltzmann Modells
`resetFi`	Steuerparameter zum Zurücksetzen der Simulation
`sceneUpdateIntervall`	Intervall bis zum Datentransfer mit der Ein- und Ausgabe
`checkConvergenceIntervall`	Intervall bis zum Prüfen der Konvergenz der Simulation
`safeFiIntervall`	Intervall bis zum Schreiben der Verteilungsfunktionen
`safeAvsIntervall`	Intervall bis zum Schreiben von Druck und Geschwindigk.
`size_x`, `size_y`, `size_z`	Anzahl der Gitterpunkte des numerischen Gitters
`numGeoObstacles`	Anzahl der Hindernisse im Strömungsgebiet
`geoObstacles`	Feld mit Beschreibung der geometrischen Hindernisse
`u_0`	Geschwindigkeit im Einlaufprofil
`dp`	Druckgradient in der Kanalströmung
`nu`	Viskosität des Fluids
zu sendende Daten	**Bedeutung**
`isStationary`	Zeigt den stationären Zustand des Systems an
`nodalUpdates`	Anzahl der Gitterknoten-Erneuerungen

Tabelle 7.1: Auszug aus der Struktur `lbSimSystem`

Die Struktur `vfrGeoObstacles` – Beschreibung der Hindernisse im Strömungsgebiet

In dieser Struktur sind die geometrischen Hindernisse im Strömungsgebiet codiert. Dabei ist zunächst eine Variable `status` gehalten, um sicherzustellen, dass nur diejenigen Hindernisse bei der Erneuerung des Gitters berücksichtigt werden, die auch wirklich verändert wurden. Die Variable `type` gibt die Art des Hindernisses an (z.B. Ellipsoid). Anschließend folgt das Feld `newGeoParams`, mit dem die geometrischen Objekte in Form von maximal 9 Parametern beschrieben werden. Diese Beschreibung ist abhängig vom Typ des Objektes. Bei einem Ellipsoid sind dies der Mittelpunkt, die Größe der 3 Hauptachsen und die Rotation im Raum. Diese Parameter müssen ein zweites mal im Feld `oldGeoParams` gehalten werden, um die Gitterpunkte die nach einer Veränderung durch den Benutzer nicht mehr durch das Hindernis überdeckt werden

auf den Status eines Knotens im Strömungsgebiet zurückzuführen. Diese doppelte Datenhaltung ist nötig, weil beim Empfang der Daten vom System der Visualisierung das Feld `newGeoParams` bereits überschrieben wird.

Die Struktur `lbSimSlave` – Rechenprozesse der Simulation

In dieser Datenstruktur sind Informationen gehalten, die nur für die Rechenprozesse gehalten werden müssen. Hier ist zunächst eine Instanz der oben erläuterten Struktur `lbSimSystem` zu nennen.
Der wichtigste Teil dieser Struktur ist eine 4D-Matrix, in der die Verteilungsfunktionen der Lattice-Boltzmann Simulation gespeichert sind. Diese Matrix wird insgesamt drei mal (`f`, `f_buf` und `f_file`) allokiert. Nach jeden Zeitschritt werden diese Zeiger zyklisch gewechselt. Die einzelnen Zeiger haben dabei die folgende Bedeutung:

f Dieser Zeiger zeigt am Ende eines Zeitschrittes auf die gültigen Verteilungsfunktionen. Die Berechnung des Druckes sowie der Geschwindigkeit bezieht sich somit auf dieses Feld. Im anschließenden Zeitschritt wird zunächst die Kollision mit `f` berechnet. Ebenso wird damit die Übertragung der Verteilungsfunktionen an die benachbarten Prozesse vorgenommen.

f_buf Die Ergebnisse der Propagation werden hier gespeichert, weil die Übertragung der Verteilungen am Gebietsrand aus dem Feld `f` durch die Möglichkeiten von MPI gerade noch parallel stattfinden könnte (siehe Abschnitt 7.2.4) und somit zu diesem Zeitpunkt nicht auf `f` zugegriffen werden sollte. Ebenso werden hier die ankommenden Verteilungen aus den benachbarten Prozessen gespeichert. Anschließend wird der Zeiger von `f` auf `f_buf` gesetzt und folglich zeigt `f` im nächsten Zeitschritt wieder auf die aktuellen Verteilungsfunktionen.

f_file Hierbei handelt es sich wie bei `f_buf` um einen temporären Speicherplatz für die Verteilungsfunktionen, mit dem Zweck, diese sicher in eine Datei zu schreiben. `f_file` wird am Ende eines Zeitschrittes auf `f_buf` gesetzt.

Informationen zu den Knoten des numerischen Gitters hinsichtlich der *marker and cell* Beschreibung sind in dem Member `bcType` gespeichert und werden in Abschnitt 7.2.6 genauer erläutert. Zuletzt werden noch Variablen für den Datentransfer mit MPI gehalten. Erwähnenswert sind hier die Parameter vom Typ *MPI_Comm* sowie *MPI_Datatype*. Zur genauen Erklärung dieser Datenstrukturen und deren Rolle in MPI wird auf [83, 66] verwiesen und hier nur kurz das Prinzip angerissen: Jeder Sendevorgang mit MPI erfolgt in einer durch die Programmierung definierten Umgebung von Prozessen. So gibt es jeweils ein Objekt vom Typ *MPI_Comm* für den Datenaustausch der Rechenprozesse untereinander (`slaveComm`) sowie ein weiteres Objekt für den Datenaustausch mit dem Hauptprozess (`masterSlaveComm`). Außerdem muss für jeden Sendevorgang bekannt sein, wie der zu übertragende Speicherbereich verteilt ist. Dies kann man mit einem Objekt vom Typ *MPI_Datatype* beschreiben. Ein einfaches Beispiel dazu ist das Versenden einer Spalte in einer Matrix, wobei die Einträge in den Zeilen der Matrix sequentiell im Speicher liegen. Die zu versendenden Verteilungsfunktionen liegen ebenfalls nicht sequentiell im Speicher (siehe Abschnitt 7.2.4) und die Anordnung wird folglich durch eine zugeordnete Instanz von *MPI_Datatype* beschrieben.

Die Struktur `lbSimMaster` – Steuerprozess der Simulation

In dieser Struktur werden alle Daten gehalten, die auf dem Hauptprozess benötigt werden. Dazu gehört wieder eine Instanz vom Typ `lbSimSystem`.

Die wichtigsten Felder sind hier die 4D-Matrizen `primVars` und `primVars_wr`, in denen die Ergebnisse der Simulation (Druck, Geschwindigkeit) gespeichert werden. Die Bezeichnung "primVars" bezieht sich auf die Variablen, die für die Visualisierung von primären Interesse sind und nicht auf die Primärvariablen der Simulation. Auch hier werden diese Felder wiederum doppelt gehalten, weil ein konkurrierender Zugriff vorliegt: Während die aktuell gültigen Ergebnisse in `primVars` zu Zwecken der Visualisierung gebraucht werden, werden bereits neue Ergebnisse gesendet, die nun einfach im zweiten Speicherbereich `primVars_wr` abgelegt werden. Ist der Datenaustausch beendet, werden die Zeiger getauscht und `primVars` zeigt dann wiederum auf den aktuellen Datensatz.
Weiterhin sind noch eine Reihe von Membern zur Verwaltung des Datenaustauschs mit MPI enthalten. Dazu zählt in erster Linie die Umgebung der Kommunikation zwischen dem Hauptprozess und den Teilprozessen der Berechnung.

7.2.4 Parallelisierung der Lattice-Boltzmann Simulation

Der hier implementierte Simulationskern setzt konsequent die in Abschnitt 3.2, Abbildung 3.2 skizzierten Ebenen der Parallelisierung um.

Ebene 1: Interprozesskommunikation

Die Gebietszerlegung für die parallel arbeitenden Prozesse erfolgt wie in Abschnitt 7.2.1, Abbildung 7.4 skizziert, entlang der Länge (hier: x-Achse) der Kanalströmung. Das Strömungsfeld wird somit in gleich große Teile aufgeteilt um eine ausgeglichene Lastverteilung der parallelen Prozesse herzustellen. Die Aufteilung in Blöcke führt dann dazu, dass innerhalb der Teilprozesse wiederum mit vollen Matrizen gerechnet werden kann. Es wird hier bewusst auf intelligentere Verfahren der Gebietszerlegung verzichtet, weil diese in der Regel rechenintensiv sind und nach jeder Änderung der Geometrie durch den Benutzer neu ausgeführt werden müssten. Weiterhin wäre ein ständiges neues Allokieren des Speichers nötig, weil sich die Größe der Teilgebiete stets ändern könnte. Auf der anderen Seite ist natürlich klar, dass mit der gleichmäßigen Zerlegung des Gebietes in Blöcke die ausgeglichene Lastverteilung nur annähernd hergestellt werden kann: Im Extremfall wäre es möglich, dass der Benutzer sämtliche geometrischen Hindernisse gerade in einem einzigen Teilgebiet platziert hat. Die Berechnung der Gitterknoten im Strömungsgebiet ist wesentlich aufwendiger als die Ermittlung der Ergebnisse für Knoten mit geometrischen Objekten. Durch die Aufteilung der Blöcke entlang der Längsachse der Kanalströmung wird ebenfalls angestrebt, das Volumen der zu übertragenden Daten zwischen den Teilprozessen zu minimieren, weil der Querschnitt senkrecht zur Kanalachse stets eine kleinere Fläche hat, als die zu den anderen beiden Achsen senkrecht stehenden Querschnitte.
Nun stellt sich die Frage, welche Verteilungsfunktionen zwischen den Prozessen ausgetauscht werden müssen. Ein Blick auf die beiden Teilschritte der Lattice-Boltzmann Simulation zeigt, dass die Kollision (Gleichung (3.19)) vollständig lokal zu berechnen ist, während die Propagation (Gleichung (3.20)) lediglich einen Zugriff auf den nächsten Nachbarn im numerischen Gitter benötigt. Dieser Vorgang ist im linken Teilbild von Abbildung 7.7 in 2D skizziert.
Durch die Gebietszerlegung entlang der x-Achse ist klar, dass alle Verteilungen, die gemäß den Gleichungen (3.9) und (3.10) einen Anteil in der x-Koordinate besitzen, zwischen den parallelen Prozessen ausgetauscht werden müssen. Diese Verteilungen sind exemplarisch für das Modell D3Q19 in der rechten Skizze von Abbildung 7.7 aufgezeichnet.
Bei der Übertragung der Daten zu den benachbarten Prozessen nutzt man die Möglichkeit, dass MPI Daten versenden kann und man gleichzeitig im Programmlauf fortfahren kann ohne auf die

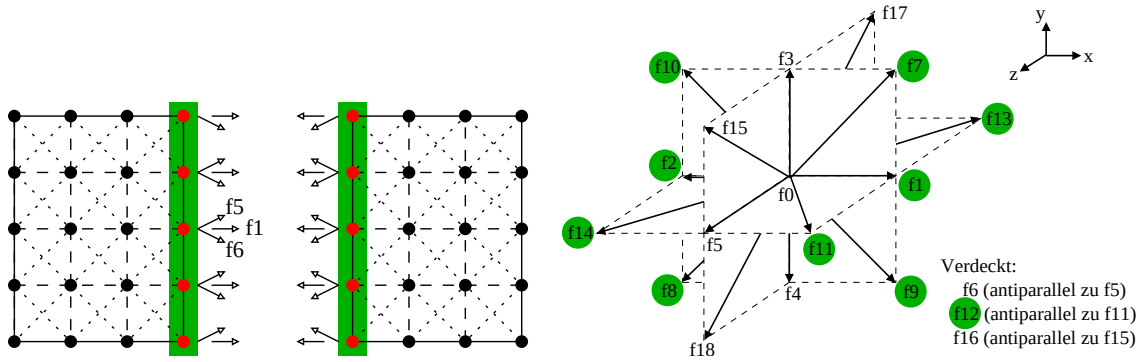

Abbildung 7.7: Links: Veranschaulichung des Austauschs der Verteilungsfunktionen in 2D, nur Kommunikation mit dem nächsten Nachbarn – rechts: Auszutauschende Verteilungsfunktionen am Beispiel des Modells D3Q19

Beendigung des Übertragungsvorgangs warten zu müssen (so genannter *Non-Blocking Modus*). Mit anderen Worten wird der Vorgang des Datenaustauschs versteckt (so genanntes *Message-Hiding*). Ein dafür optimierter Algorithmus (siehe den Pseudocode A-5) für einen Zeitschritt einer Lattice-Boltzmann Simulation ist in [96] vorgestellt und hier angepasst.

Algorithmus A-5 Ablauf einer parallelen Simulation mit *Message-Hiding*

1: **for all** Zeitschritte **do**
2: Berechne die Kollision der Randknoten
3: Sende die Randknoten im Modus *Non-Blocking*
4: Initialisiere das Empfangen benachbarter Randknoten im Modus *Non-Blocking*
5: Berechne die Kollision der inneren Knoten
6: Führe die Propagation der inneren Knoten durch
7: Warte auf die ankommenden Randknoten der benachbarten Gebiete
8: Erfolgreiches Senden aus Anweisung 3 sicherstellen
9: **end for**

Zuerst wird die Kollision für die Knoten an den Gebietsrändern berechnet (A-5, Anweisung 2). Nun sind diese zum Versenden an die benachbarten Prozesse bereit. Während diese Knoten sozusagen unterwegs sind, wird für die Knoten im Inneren des Gebietes die Kollision (A-5, Anweisung 5) und die Propagation (A-5, Anweisung 6) berechnet. Zuletzt muss noch sichergestellt werden, dass die gesendeten Daten das jeweilige Ziel korrekt erreicht haben (A-5, Anweisung 7 und 8).

Parallele Prozesse	1	2	3	4	5	6	7
Hitachi SR8000 – Speed-Up	1,00	1,95	2,88	3,85	4,72	5,53	
Hitachi SR8000 – Effizienz	1,00	0,97	0,96	0,96	0,94	0,92	
Workstation-Cluster – Speed-Up	1,00	2,06	3,03	4,03	4,93	6,01	7,10
Workstation-Cluster – Effizienz	1,00	1,03	1,01	1,01	0,98	1,00	1,01

Tabelle 7.2: Speed-Up und Effizienz der parallelen Lattice-Boltzmann Simulation bei 1 Millionen Gitterknoten

Die Abbildungen 7.8 und 7.9 sowie die Tabelle 7.2 sind ein Auszug umfangreicher Messungen zur Effizienz dieser Methodik der Parallelisierung einer Lattice-Boltzmann Simulation. Auf der

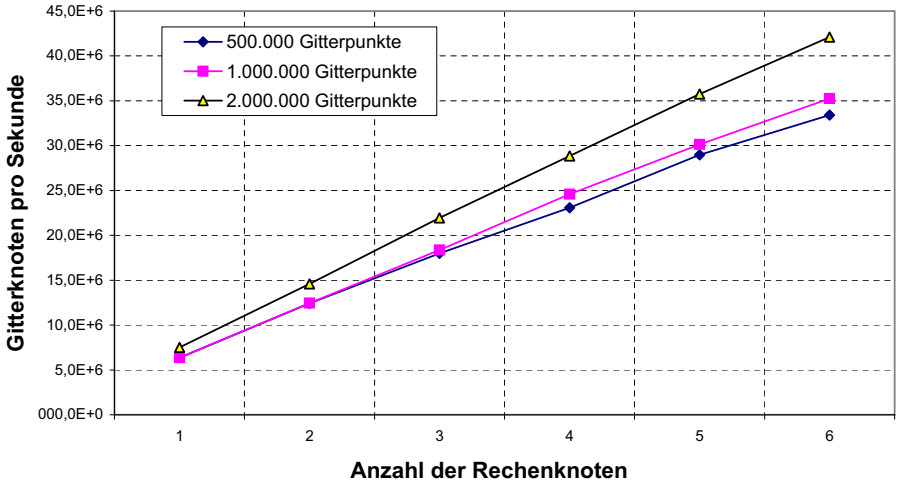

Abbildung 7.8: Hitachi SR8000 – Leistung der parallelen Lattice-Boltzmann Simulation

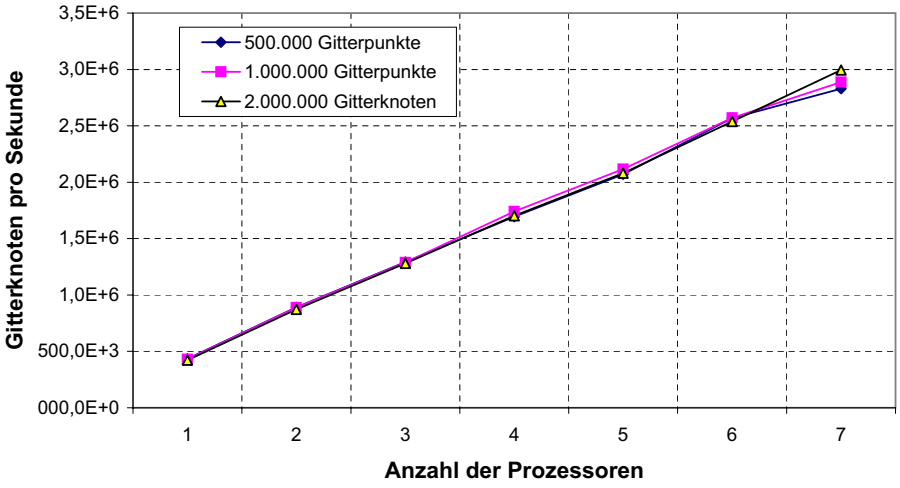

Abbildung 7.9: Workstation-Cluster Pentium 4, 1GB SDRAM – Leistung der parallelen Lattice-Boltzmann Simulation

Hitachi SR8000 verläuft der Datenaustausch zwischen den einzelnen Knoten über ein schnelles Bus-System, so dass die hervorragende Effizienz von 0,92 im Falle von 6 Knoten keinesfalls überraschend ist. Die Werte für den Workstation–Cluster überschreiten den maximalen theoretischen Wert von 1,0 leicht. Dieser Umstand ist auf den schwachen SDRAM Speicher der verwendeten Pentium 4 Rechner zurückzuführen. Durch die parallele Bearbeitung verkleinert sich das Problem für den einzelnen Prozess. In diesem Fall verbessert sich die Effizienz des einzelnen Knotens, was durch die Messungen von Tabelle 7.3 bestätigt wird.

Ebene 2: Auto-Parallelisierung mit gemeinsamem Speicher

Eine automatische Parallelisierung ist nur möglich, wenn dies von der Hardware und vom Compiler unterstützt wird.
Auf der Hitachi SR8000 (siehe Abschnitt 3.2) werden 8 Prozessoren zu einem Knoten mit einem gemeinsamen Speicherbereich zusammengefasst. Durch eine wohl definierte Markierung ei-

ner Schleife im Quellcode erzeugt der Compiler mehrere parallele Threads zur Bearbeitung dieser Schleife. Als Entwickler muss man dafür Sorge tragen, dass innerhalb dieser Schleife keine Abhängigkeiten unter den einzelnen Iterationen vorhanden sind. Nur dann ist eine Aufteilung der Schleife für mehrere parallele Threads möglich. Im günstigen Fall merkt der Compiler derartige Umstände, es kann aber auch passieren, dass fehlerhafte Programme erzeugt werden.
In der hier beschriebenen Applikation wird die Bearbeitung der y-Richtung der Matrix des numerischen Gitters für die auto-parallele Bearbeitung verwendet.

Ebene 3: Parallelisierung auf Instruktionsebene (Vektorisierung)

Dieser Vorgang wird ebenfalls durch speziell formatierte Compiler-Anweisungen auf dafür geeigneten Hardware-Plattformen (i.d.R. Supercomputer) umgesetzt. Tiefer gehende Fragen zur Vektorisierung können in [63, 48] nachgelesen werden. Hier werden kurz grundsätzliche Gedanken der Vektorisierung beschrieben, um deren Einfluss auf die Gestaltung des Quellcodes verständlich zu machen.
Die Vektorisierung wird erreicht, indem Speicherinhalte bereits lange vor dem Gebrauch in die Register geladen werden. Beim so genannten *Prefetch*–Mechanismus wird der verwendete Speicher direkt aus dem Cache in die Register geladen. Für die Bearbeitung einer Schleife bedeutet dies, dass während das Element `i` bearbeitet wird, das Element `i+1` in die Register geladen wird. Zuvor wird ein Segment des Cache (*Cacheline*) aus dem Hauptspeicher geladen. Nur wenn die zu bearbeitenden Elemente einer Schleife im Hauptspeicher nahe zusammen liegen, kann dieser Mechanismus effizient sein. Andernfalls werden beim Laden einer *Cacheline* zu viele nicht benötigte Speicherbereiche mitgeladen, oder anders betrachtet muss der Cache zu oft neu geladen werden. Bei einer Lattice-Boltzmann Simulation tritt hier jedoch ein Dilemma auf: Möchte man eine optimale Leistung hinsichtlich der Kollision erzielen, dann müssen die Verteilungsfunktionen `f[i]` sequentiell im Speicher angeordnet[6] sein. Somit müsste man die 4D-Matrix der Verteilungsfunktion in der Form `f[x][y][z][i]` halten. Betrachtet man andererseits die Bestimmungsgleichung (3.20) der Propagation, dann erkennt man, dass man für eine optimale Implementierung den Zugriff auf die Verteilung des nächsten Nachbarn möglichst effizient gestalten muss. Die Matrix der Verteilungsfunktionen müsste demzufolge in der Form `f[i][x][y][z]` aufgebaut sein.
Es ist a priori nicht zu entscheiden, welche Anordnung des Speichers für die eingesetzte Hardware (z.B. Größe des Caches) die besseren Ergebnisse liefert [40]. Deshalb wurden im Rahmen dieser Arbeit zunächst beide Speichermodelle implementiert, um die endgültige Festlegung anhand der in Abbildung 7.10 und Tabelle 7.3 zusammengefassten Messungen vorzunehmen:
Bei Verwendung einer 4D-Matrix ist es auf der Hitachi SR8000 günstiger, die Verteilungsfunktionen in der Form `f[i][x][y][z]` zu halten. Die Unterschiede wirken sich aber erst dann aus, wenn die Längen der zu durchlaufenden Schleifen einen bestimmten Wert erreichen, weil erst dann die Vektorisierung richtig greift. In den Messungen von Abbildung 7.10 ist dieses Phänomen ab einer Länge der innersten Schleife von 100 Gitterpunkten erkennbar.
Zum Vergleich sind in der Tabelle 7.3 Werte für heutige Standard PCs aufgelistet. Die oben gewählte Speicherverwaltung liefert für diese Rechnerarchitekturen ebenfalls die bessere Leistung. Auffällig ist der äußerst positive Einfluss des schnellen DDRAM Speichers im Falle des Pentium 4 mit 2,1 GHz. Der in dieser Arbeit verwendete Workstation-Cluster besteht jedoch aus Rechnern mit dem langsameren SDRAM Speicher.
Eine entscheidende Auswirkung auf die Vektorisierung hat in erster Linie der Kollisionsoperator, der im Folgenden noch etwas genauer betrachtet wird.

[6] Dies ist bedingt durch die Berechnung der Gleichgewichtsverteilung gemäß der Gleichung (3.12) in Verbindung mit ρ nach Gleichung (3.13.) und $\vec{u}$ nach Gleichung (3.14).

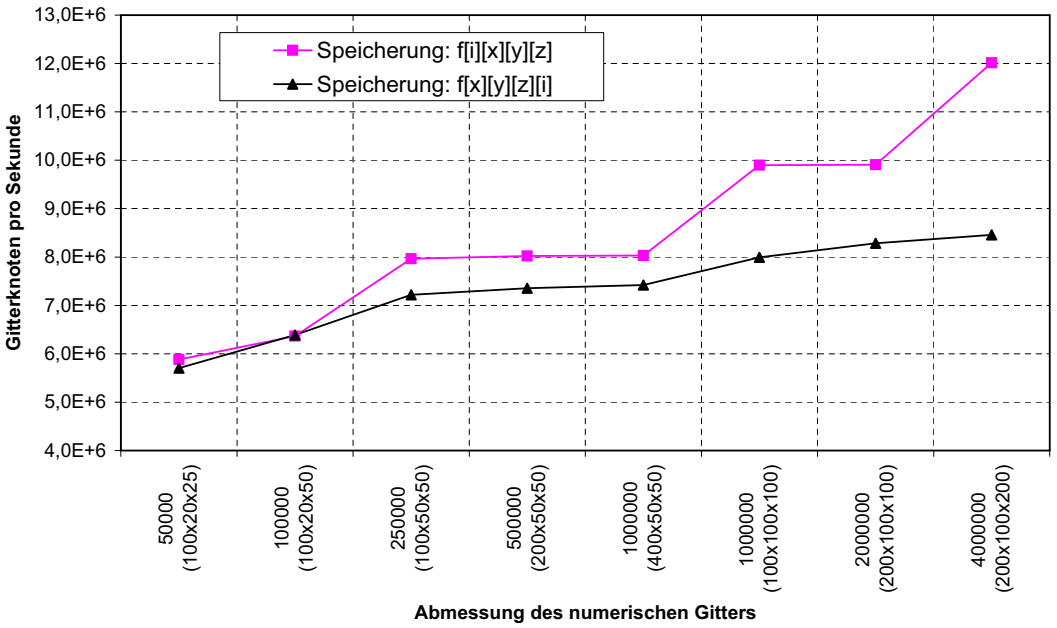

Abbildung 7.10: Leistung eines Einzelknotens auf der Hitachi SR8000 in Abhängigkeit von der Speicherung der Verteilungsfunktionen

Der Kollisions-Operator

Für eine effiziente Vektorisierung ist es weiterhin von entscheidender Bedeutung, dass die Anweisungen innerhalb einer Schleife für jeden Durchlauf gleichartig sind. Die Kollision bereitet in diesem Zusammenhang erhebliche Schwierigkeiten, weil in Abhängigkeit der Art (d.h. Fluidknoten, Druckrandbedingung, etc.) eines Gitterknotens andere Abläufe ausgeführt werden müssen. Dies führt dazu, dass in der zu vektorisierenden Schleife eine Fallunterscheidung durchgeführt werden muss. Die einzelnen Fälle unterscheiden sich dabei sehr stark:

- Ein Fluid–Knoten (der Standard) wird gemäß der Formeln (3.19) für die Kollision in Verbindung mit dem Kollisionsoperator (3.7), der Gleichgewichtsverteilung (3.12), der Dichte (3.13) und der Geschwindigkeit (3.14) ermittelt.

Gitterpunkte	Pentium 4, 2,1GHz Speicher: DDRAM		Pentium 4, 1,7GHz Speicher: SDRAM		Hitachi SR8000 8 Threads, autopar.	
	f[i][x][y][z]	f[x][y][z][i]	f[i][x][y][z]	f[x][y][z][i]	f[i][x][y][z]	f[x][y][z][i]
50000 (100x20x25)	1250979	1074231	585141	435070	5883932	5705118
100000 (100x20x50)	1245486	992818	556936	409868	6364554	6387254
250000 (100x50x50)	1234598	982220	530772	399193	7963256	7217773
500000 (200x50x50)	1222001	974144	513226	390351	8023572	7356204
1000000 (400x100x100)	1218408	972976	504999	382604	8032371	7418770
1000000 (100x100x100)					9899928	7992851
2000000 (200x100x100)					9908408	8285544
4000000 (200x100x200)					12016646	8459413

Tabelle 7.3: Leistung der Lattice-Boltzmann Simulation in Abhängigkeit von der Speicherverwaltung, der Rechnerarchitektur und der Größe des simulierten Problems

- Bei der Vorgabe des Drucks (bzw. der Dichte) oder der Geschwindigkeit kann der Rest wie bei der Berechnung der Fluid-Knoten ablaufen.

- Hindernisse im Strömungsfeld werden mit einer Haftrandbedingung versehen, die mit dem so genannten Prinzip *Bounce-Back* bestimmt werden: Diese Idee basiert auf der physikalischen Tatsache, dass ein Partikel, welches gegen eine Wand prallt, sich nach dem Aufprall in der entgegengesetzten Richtung fortbewegt. Erreicht wird dies, indem man lediglich die antiparallelen Verteilungsfunktionen tauscht (siehe [108, 57]).

In [96] wird vorgeschlagen, die Knoten zu sortieren, so dass zuerst alle Fluid-Knoten berechnet werden, anschließend folgen alle Knoten mit einer Haftrandbedingung, usw. Mit diesem Verfahren wird für einen Knoten der Hitachi SR8000 eine Leistung von ca. 15–20 Millionen Gitterknoten pro Sekunde erzielt.
Es ist jedoch problematisch, diese Methodik in eine Applikation der interaktiven Simulation einzubauen, weil diese Sortierung nach jedem Eingriff durch den Benutzer neu durchgeführt werden muss und somit eine Art Umbau des allokierten Speichers erfolgen muss. Mit Hilfe von wohl bedachten Feinabstimmungen des Quellcodes auf der Basis einer vollbesetzten Matrix, konnte für einen Knoten der Hitachi SR8000 eine Leistungsfähigkeit von 6–12 Millionen Gitterknoten pro Sekunde erreicht werden. Das sind bis zu 60% der Werte des oben genannten Verfahrens.

7.2.5 Erweiterung der parallelen Lattice-Boltzmann Simulation für die interaktive Steuerung und Auswertung

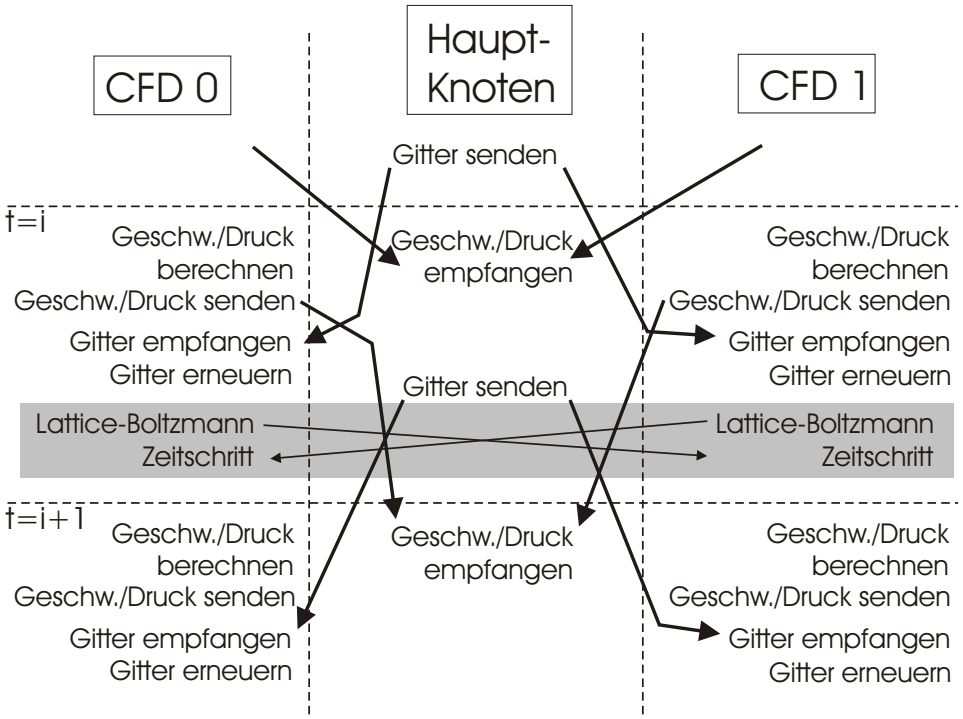

Abbildung 7.11: Ablauf einer interaktiven Lattice-Boltzmann Simulation – der Datenaustausch mit der Visualisierung (Hauptknoten) erfolgt parallel zur Berechnung

Die Erweiterung der Lattice-Boltzmann Simulation für den Zweck der interaktiven Steuerung ist in Abbildung 7.11 für den vereinfachten Fall von 2 Rechenknoten skizziert. Von der Seite der Visualisierung werden die Steuerparameter und die Informationen für die Modifikation des Gitters in einem Zug versendet (`Gitter versenden` in Abbildung 7.11). Gleichzeitig werden die Ergebnisse der Simulation an die Komponente der Visualisierung gesendet (`Geschw./Druck empfangen` in Abbildung 7.11).
In diesen Zusammenhang muss besonders auf das in Abschnitt 7.2.4 angewendete Prinzip des *Message Hiding* hingewiesen werden. Es wurde bereits angedeutet, dass der Datenaustausch zwischen der Komponente der Ein- und Ausgabe und der Komponente der Berechnung kritisch sein kann. Die Pfeile in Abbildung 7.11 kennzeichnen, in welchen Zeitabschnitten eine Nachricht unterwegs ist. Beispielsweise sieht man, dass während des Vorgangs der Übertragung der Ergebnisse an den Hauptknoten ein vollständiger Zeitschritt der Lattice-Boltzmann Simulation durchgeführt wird. Auf die gleiche Art und Weise wird der Transfer der geänderten Geometrien optimiert.
Einzig die Erneuerung des Gitters selbst sowie die Berechnung des Drucks bzw. der Geschwindigkeit stellen damit einen wirklichen zeitlichen Mehraufwand dar. Dieser Umstand ist allerdings deshalb nicht besonders gewichtig, weil entgegen der Skizze von Abbildung 7.11 die zusätzlichen Operationen für die interaktive Steuerung nicht in jedem Zeitschritt aufgerufen werden. Der Benutzer kann ein Intervall von Zeitschritten angeben, innerhalb dem der Datenaustausch zwischen der Berechnung und der Komponente der Ein- und Ausgabe durchgeführt wird.
Um die Synchronisation des Ablaufs zu erleichtern, wird die Übertragung der Ergebnisdaten quasi parallel zur Übertragung der Informationen des modifizierten Gitters durchgeführt.

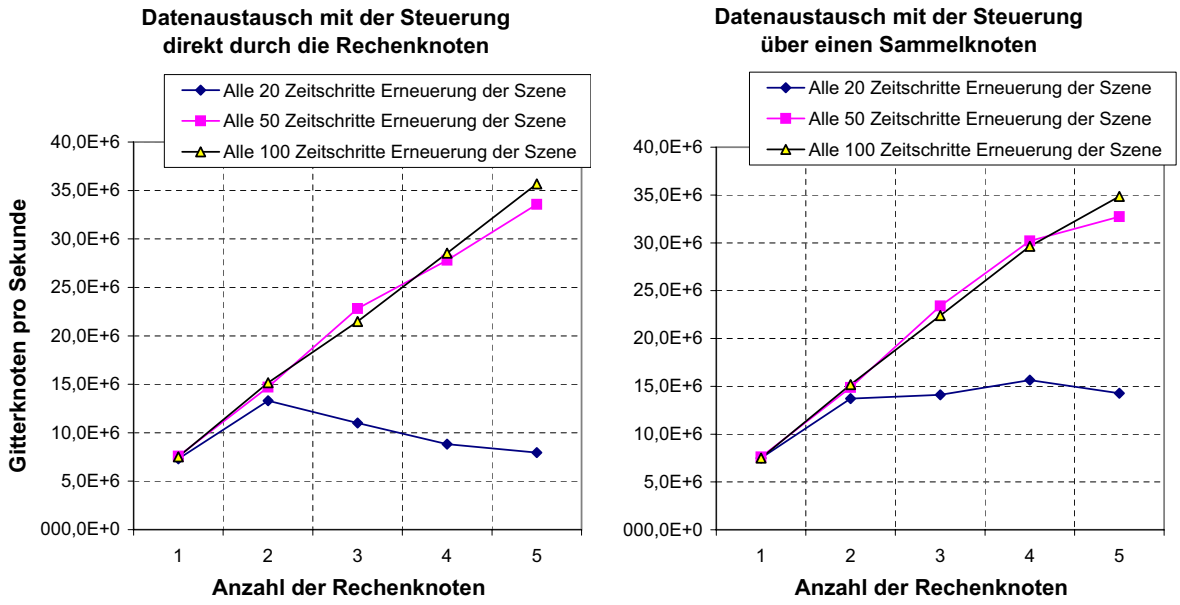

Abbildung 7.12: Leistung der interaktiven Simulation bei einer Kommunikation zwischen der Hitachi SR8000 und der Virtual Reality Anlage im LRZ München – das linke Bild entspricht der Variante *MasterVis* und das rechte Bild repräsentiert die Variante *MasterSim*.

In Abbildung 7.12 ist die Leistungssteigerung der interaktiven Simulation für verschiedene Intervalle des Datenaustauschs zwischen der Hitachi SR8000 und der Steuerung durch die VR Anlage aufgetragen. Die Messung wurde für ein Gitter mit 1,0 Millionen Gitterknoten durchgeführt. Es ist offensichtlich, dass das Intervall des Datenaustauschs von 20 Zeitschritten zu klein ist. Weitere Messungen in diesem Kontext haben gezeigt, dass die Größe des Gitters dabei keine Rolle spielt.

Bei kleineren Gittern wäre die Zeit für den Transfer der Ergebnisdaten zwar kürzer, die parallel laufende Berechnung weiterer Zeitschritte der Strömung wäre dann aber schneller abgeschlossen. Wird das Intervall des Datenaustauschs erhöht, so bleibt während der Berechnung genügend Zeit, um die Kommunikation mit der Komponente der Ein- und Ausgabe abzuschließen. Hierbei kann man auch bei 5 Rechen-Knoten der Hitachi SR8000 eine annähernd optimale Steigerung der Leistung des Systems bis zu 35,6 Millionen Gitterknoten-Erneuerungen pro Sekunde beobachten. Die in Abschnitt 7.2.2 aufgeworfene Frage nach der Architektur des Systems kann nun wie folgt beantwortet werden: Ist das Intervall des Datenaustauschs mit der Komponente der Ein- und Ausgabe groß genug, dann wird die Leistung des Gesamtsystems nicht davon beeinflusst, ob die Rechenprozesse direkt (siehe Abbildung 7.6) oder über einen Sammelknoten (siehe Abbildung 7.5) mit der Komponente der Steuerung kommunizieren. Ist das Intervall zu klein, so verhält sich die Alternative *MasterSim*[7] (siehe Abbildung 7.12, rechts) etwas besser als die Alternative *MasterVis*, allerdings ist dabei auch ein immenser Einbruch der Leistung festzustellen. Dies bedeutet letztlich, dass die Entscheidung für eine der beiden möglichen Systemarchitekturen unabhängig von der Leistung des Gesamtsystems getroffen werden kann.
Insgesamt wurden auch hervorragende Werte für den Speed-Up erzielt. Die Tabelle 7.4 enthält exemplarisch eine Messung für das System der Hitachi SR8000 mit der VR Anlage und für einen Workstation-Cluster. Der Datenaustausch mit der Steuerung wurde dabei in Intervallen von 40 Zeitschritten durchgeführt.

Parallele Prozesse	1	2	3	4	5	6
Gitterknoten	200000	400000	600000	800000	1000000	1200000
Hitachi SR8000 ↔ Holobench	1,00	0,95	0,94	0,93	0,93	
Workstation-Cluster	1,00	0,99	0,96	0,94	0,93	0,90

Tabelle 7.4: Scale-Up der interaktiven Lattice-Boltzmann Simulation – Datenaustausch mit der Steuerung: alle 40 Zeitschritte

7.2.6 Modifikation des diskreten Gitters zur Laufzeit der Simulation

Nach jeder Änderung der Geometrie ist das Erneuern des Gitters mit dem *marker and cell* Schema aus Abschnitt 3.1.2 durchzuführen. Die aktuelle Implementierung erlaubt die Definition von geometrischen Körpern in Form einer Kombination von parametrisierten Grundkörpern (siehe Abschnitt 7.2.1), die nun möglichst effizient in eine Voxel-Information überführt werden müssen.

Abbildung 7.13: Voxelierung der Hindernisse mit dem Bresenham-Algorithmus – Skizze)

Dieser Vorgang ist für Körper vom Typ Hexaeder und Pyramide trivial. Für die Grundkörper vom Typ Ellipsoid, Zylinder und Konus wird ein einfaches Vorgehen gewählt, das hier kurz am

[7] Zur Erklärung des Begriffs siehe Abschnitt 7.2.2

Beispiel eines Zylinders erläutert wird: Man reduziert das Problem um eine Dimension, indem man den Zylinder entlang seiner Hauptachse in eine Serie von Ellipsen zerlegt. Nun kann man jede Ellipse gemäß Abbildung 7.13 mit einem Algorithmus von Bresenham [9] in eine Raster-Information überführen. Der Vorteil dieses Algorithmus liegt darin, dass er nur mit Integer-Operationen auskommt und damit sehr effizient ist. Genauere Beschreibungen und erklärende Skizzen zu diesem Vorgehen sind [116] zu entnehmen.
Zur Bedeutung der Werte des Feldes `bcType` aus der Struktur `lbSimSlave` (siehe Abschnitt 7.2.3) ist noch folgende Anmerkung nötig: Liegt eine ausgezeichnete Randbedingung vor, wird diese mit einem Wert kleiner 0 markiert (z.B. -2 für eine Druck-Randbedingung). Für einen Gitterknoten im Strömungsgebiet wird der Wert 0 zugewiesen, während alle von Hindernissen überdeckten Gitterknoten einen Wert von `bctype = n > 0` erhalten. Im letzten Fall entspricht `n` der Anzahl der geometrischen Objekte, die einen Gitterknoten überdecken.
Wird nun ein geometrisches Objekt *GOb* bearbeitet und man stellt für einen Punkt $P(xp, yp, zp)$ fest, dass dieser von *GOb* überdeckt wird, dann wird der zugehörige Wert `bcType[xp][yp][zp]` inkrementiert. Umgekehrt wird der Wert von `bcType[xp][yp][zp]` dekrementiert, sobald ein geometrisches Objekt transformiert wird und dabei nicht mehr den Punkt $P(xp, yp, zp)$ überlagert. Damit wird `bcType[xp][yp][zp]` automatisch zu 0, sobald der Punkt $P(xp, yp, zp)$ wieder zum Strömungsgebiet gehört.

7.3 Teilsystem Virtual Reality basierte Umgebung zur Ein- und Ausgabe

Im Folgenden wird eine Virtual Reality basierte Komponente zur Eingabe und zur Visualisierung vorgestellt. Natürlich könnte man dieses Teilsystem jederzeit mit einem Standard-Werkzeug zur Visualisierung realisieren, jedoch wurde in Abschnitt 4.2 ein möglichst suggestives Medium zur Präsentation und zur Interaktion – also Virtual Reality – gefordert.
Es soll hier in erster Linie dargestellt werden, welche Komponenten in diesem Teilsystem voneinander abhängen und wo letztlich die Probleme liegen, wenn gleichzeitig die Vorgänge

- Erfassung der Interaktionen des Benutzers
- Rendering
- Datenaustausch mit der Komponente der Simulation
- Datenabbildung

durchgeführt werden müssen. Zunächst werden die Inhalte der Szene beschrieben und anschliessend wird auf das Zusammenspiel der Komponenten eingegangen.
Die wichtigsten Elemente der Klassenstruktur der Teilkomponente Ein- und Ausgabe sind in Abbildung 7.14 dargestellt. Diese Abbildung wird im weiteren Verlauf zur Unterstützung der Erklärungen verwendet.
In der Klassenstruktur sind die Inhalte der Szene bereits erkennbar. Die Klasse `vfRealApp` besitzt eine zentrale Stellung und speichert den Szenegraph des computergestützten Windkanals `VFReal` (siehe Abschnitt 7.3.1). Weiterhin wird dort das Zusammenspiel der obigen Vorgänge gesteuert (siehe Abschnitt 7.3.5).

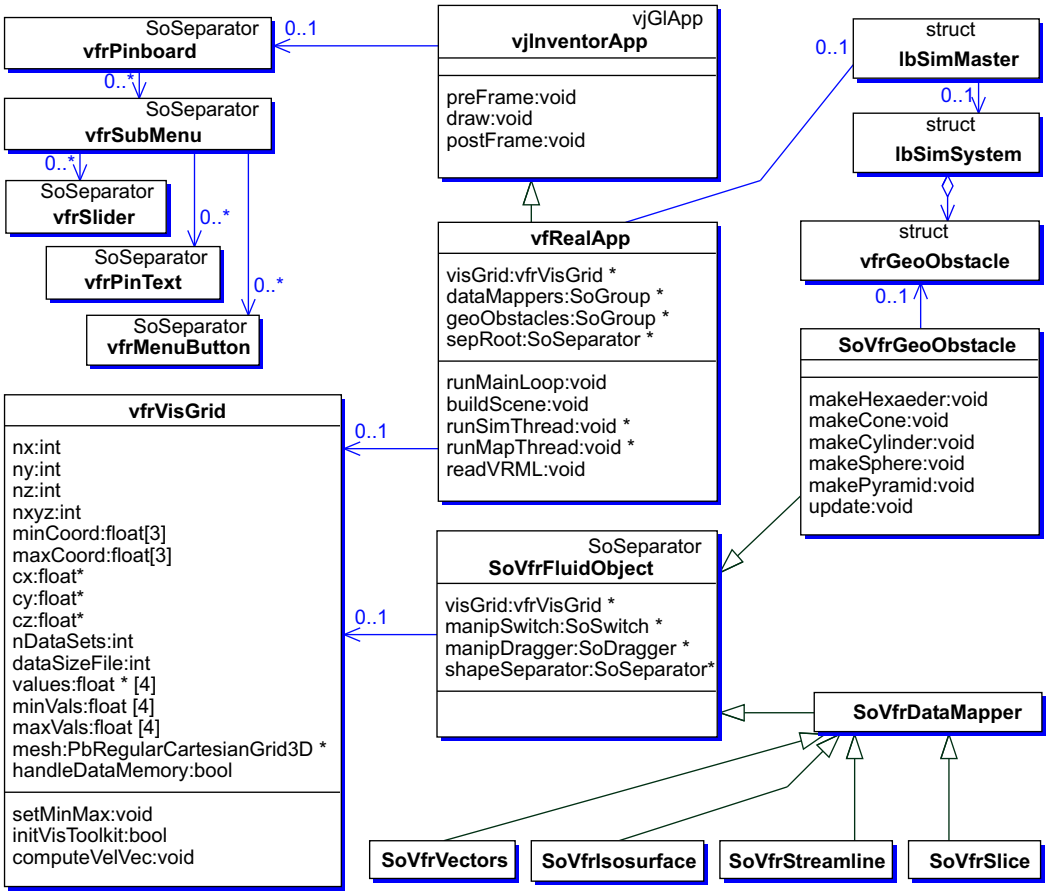

Abbildung 7.14: Klassendiagramm der Virtual Reality basierten Komponente zur Steuerung der Simulation

7.3.1 Der Szenegraph – Inhalte der dreidimensionalen virtuellen Welt

Der Szenegraph von `VFReal` ist in Abbildung 7.15 dargestellt. Neben einem Transformations-Knoten für die Navigation und diversen Lichtquellen gibt es ein Menü zur Erfassung der Eingaben des Benutzers. Dieses wird in Abschnitt 7.3.2 genauer erläutert. Rechts befindet sich der gruppierende Knoten *sepRoot*, unter dem alle Inhalte der vom Anwender zu beschreibenden Szene eingehängt sind. Im Fall von `VFReal` setzt sich diese Szene wie folgt zusammen:

- In einem so genannten VRML-Knoten können Inhalte aus einer VRML-Datei hinzugefügt werden. Dieser Knoten wird in der Regel zum Laden eines qualitativ hochwertig modellierten Modells der umgebenden Geometrien im Sinne der in Abschnitt 6.5 vorgestellten kombinierten Visualisierung verwendet.
- Die Darstellung der Hindernisse innerhalb des Strömungsgebietes wird in Abschnitt 7.3.3 genauer beschrieben.
- Die Abbildung der Ergebnisdaten wird in Abschnitt 7.3.4 ausführlich erklärt.

7.3.2 Immersives Benutzermenü

Die Eingaben des Benutzers werden vollständig mit graphischen Objekten der 3D-Welt erfasst. Dazu wurde in [116] ein Menü implementiert, das sich aus Schiebereglern (Klasse `vfrSlider`),

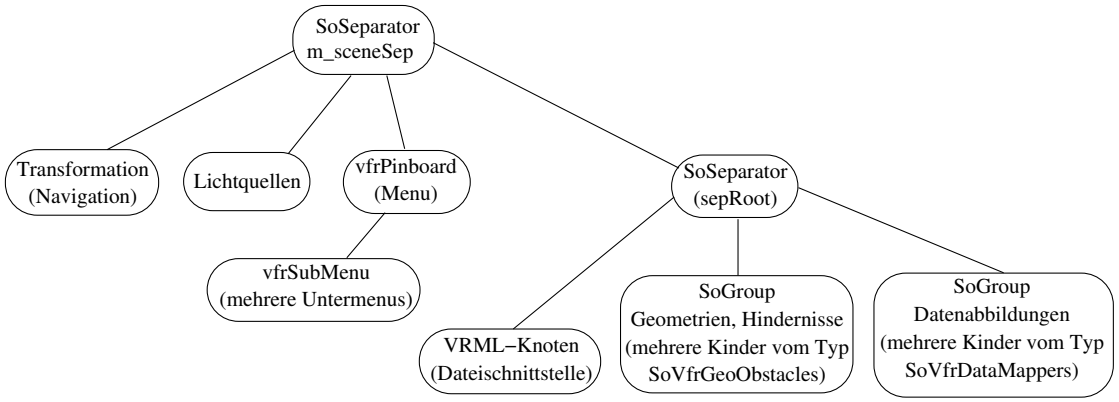

Abbildung 7.15: Der Szenegraph von `VFReal`

Textfeldern (Klasse `vfrPinText`) und Knöpfen (Klasse `vfrMenuButton`) zusammensetzt. Komplexere Elemente wurden bewusst nicht implementiert, weil jede Benutzerschnittstelle einer Virtual Reality Applikation so einfach wie möglich gehalten werden sollte, damit diese intuitiv bedienbar ist.
Abbildung 7.16 zeigt eine Instanz dieses Menüs während der Anwendung.

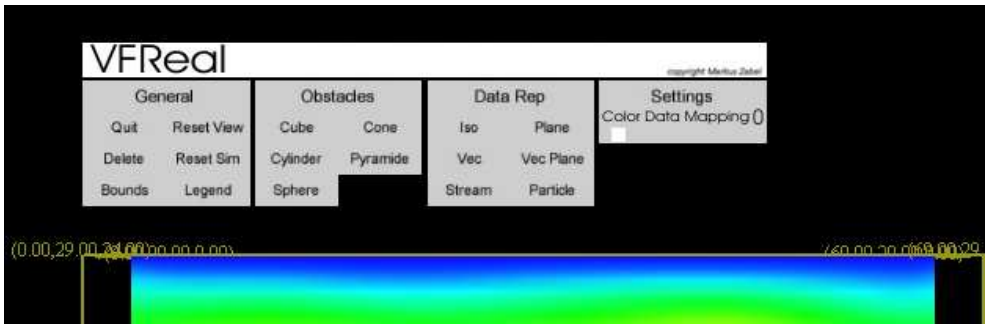

Abbildung 7.16: 3D Benutzer-Menü als Teil der Szene

Es stellt sich die Frage, warum man für derartige Zwecke nicht auf die Funktionalität von GUI-Bibliotheken wie z.B. Qt[8] zurückgreift. Die pragmatische Erklärung ist, dass Qt in der Regel nur mit einer Maus bedienbar ist. Ein Benutzer in Virtual Reality hingegen könnte aber beispielsweise einen Datenhandschuh verwenden.
Das Menu (in `VFReal` *Pinboard* genannt – siehe Klasse `vfrPinboard` von Abbildung 7.14) ist in funktionale Untermenüs (Klasse `vfrSubMenu`) aufgeteilt. Der linke Bereich von Abbildung 7.16 bezieht sich auf allgemeine Einstellungen zur Simulation (z.B. Verlassen des Programms). Die beiden mittleren Untermenüs ermöglichen das Erzeugen von geometrischen Hindernissen (Menü `Obstacles`) bzw. von Datenabbildungen (Menü `Data Rep`). Eine Besonderheit ist das Menü rechts (`Settings`), das sich stets dem aktuellen Kontext anpasst. Zur Erklärung dient folgendes Beispiel: In Abbildung 7.16 ist am unteren Bildrand noch eine Schnittebene sichtbar, welche gerade aktiviert ist. Dadurch wird das Menü `Settings` automatisch geändert und ein Schieberegler (`ColorDataMapping`) erzeugt, mit dem der Datensatz ausgewählt werden kann, auf den sich die Schnittebene bezieht.

[8]http://www.trolltech.com

7.3.3 Geometrische Hindernisse im Strömungsgebiet

Der Sub-Szenegraph für die geometrischen Objekte im Strömungsgebiet ist in Abbildung 7.17 dargestellt und in der Klasse `SoVfrGeoObstacle` implementiert.

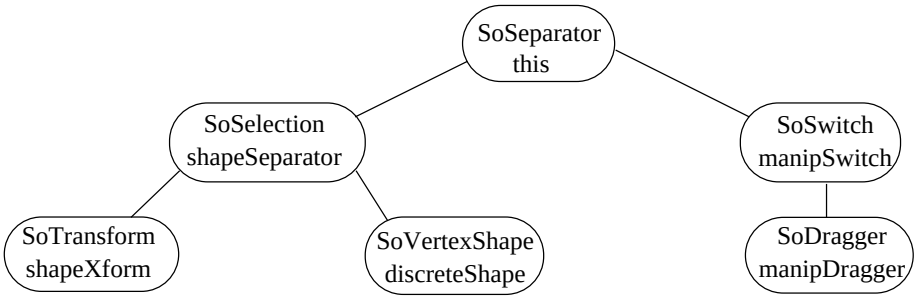

Abbildung 7.17: Der Sub-Szenegraph für geometrische Objekte (Hindernisse) in `VFReal`

Links in Abbildung 7.17 ist die Geometrie der Hindernisse als so genannter *VertexShape*-Knoten gehalten. Gleichzeitig besteht die Möglichkeit, diesen zu transformieren. Hinter einem Knoten vom Typ *VertexShape* steckt nichts anderes als ein Facetten-Modell des geometrischen Objekts. Obwohl die Hindernisse im Strömungsfeld in der aktuellen Version noch einfache Grundkörper wie z.B. Zylinder sind – für die es im Szenegraph eigene Knoten gäbe – erfolgt deren graphische Beschreibung gleich als Facettenmodell. Mit Hilfe des *VertexShape*-Knoten können in künftigen Erweiterungen beliebig komplexe B-Rep Darstellungen von geometrischen Objekten in den Szenegraph eingehängt werden.
Rechts in Abbildung 7.17 befindet sich ein so genannter *Dragger* – ein graphisches Werkzeug zur Aufnahme von Interaktionen des Anwenders (in Abbildung 7.19, links oben ist beispielsweise ein *Dragger* zum Versetzen der Startpunkte einer Partikelverfolgung zu sehen). Die Klasse `vfRealApp` als zentrale Klasse der VR-Umgebung von `VFReal` gibt die mit den typischen Eingabegeräten von VR erfassten Interaktionen des Anwenders an diese *Dragger* weiter. Der Dragger kann auf diese Weise transformiert werden. Nachdem er hier mit der Transformation (Element `shapeXform` in Abbildung 7.17) des geometrischen Objekts in Verbindung gebracht ist, wird dieses analog transformiert. Weiterhin können einem *Dragger* Funktionen zugewiesen werden (so genannte *Callbacks*), die bei einem bestimmten Zustand des *Draggers* ausgeführt werden. Mit diesem Mechanismus wird folgendermaßen dafür Sorge getragen, dass eine Änderung der Geometrie durch den Benutzer an das Teilsystem der Simulation weitergegeben wird:
Sämtliche Informationen zu den geometrischen Objekten im Strömungsgebiet sind innerhalb der Struktur `lbSimSystem` (siehe Abschnitt 7.2.3) im Feld `geoObstacles` (Typ `vfrGeoObstacles`) gehalten. Die Klasse `SoVfrGeoObstacle` besitzt einen Zeiger auf ein zugeordnetes Element innerhalb dieses Feldes. Sobald der *Dragger* losgelassen wird, wird das korrespondierende Element innerhalb von `lbSimSystem::geoObstacles` modifiziert. Beim nächsten Datenaustausch gemäß dem Mechanismus von Abschnitt 7.3.5 wird die hier beschriebene Modifikation des Benutzers im Teilsystem der Berechnung schließlich bekannt sein.

7.3.4 Abbildung der Ergebnisse der Simulation (Datenvisualisierung)

Die Abbildung der Ergebnisdaten ist in einem Sub-Szenegraphen integriert und in den Ableitungen der Klasse `SoVfrDataMapper` (siehe Abbildung 7.14) implementiert. Dazu gehören die

Darstellung der Daten als Schnittebenen (`SoVfrSlice`), Vektorpfeile (`SoVfrVectors`), Stromlinien (`SoVfrStreamline`) und Isoflächen (`SoVfrIsosurface`).
Der Aufbau des Szenegraphen ist dem der geometrischen Hindernisse (siehe Abschnitt 7.3.3) identisch. Zur Darstellung der abgebildeten Ergebnisse werden wieder Knoten vom Typ *VertexShape* verwendet. Ebenso ist stets ein *Dragger* vorhanden, mit dem interaktiv die Datenabbildungen geändert werden können (z.B. Verschieben einer Schnittebene durch Verschieben des zugehörigen *Draggers*).

Verwaltung des Gitters für die Visualisierung

Im Klassendiagramm von Abbildung 7.14 sieht man auf der linken Seite die Klasse `vfrVisGrid`, mit der das numerische Gitter für die Zwecke der Visualisierung beschrieben wird. Die Klassen zur Abbildung der Daten der Simulation erhalten über die Basisklasse `SoVfrFluidObject` Zugriff auf `vfrVisGrid`. Das Feld `values` hält die Ergebnisse der Simulation. Sobald neue Ergebnisse der Simulation empfangen werden, werden gemäß dem Mechanismus von Abschnitt 7.3.5 das Gitter in der Klasse `vfrVisGrid` sowie die erzeugten Instanzen der Datenabbildungen erneuert. Die Klasse `vfrVisGrid` ist grundsätzlich vom Vorgang der Übertragung der Simulationsdaten getrennt und kann die zugeordneten diskreten Datenwerte auch aus einer Datei lesen, so dass die hier beschriebene Applikation recht schnell zu einer reinen Anwendung für Zwecke der Visualisierung in Virtual Reality umgebaut werden könnte.

Bibliotheken zur Abbildung der Simulationsdaten

Für die Abbildung der Daten wurden in Abschnitt 7.1.3 die im Open Inventor von TGS enthaltene Erweiterung *DataViz* sowie die Bibliothek VTK ausgewählt.
Der Vorteil von *DataViz* ist, dass deren Klassen die abgebildeten Ergebnisse der Simulation direkt in den Szenegraphen schreiben. Für VTK ist am Ende der Pipeline der Datenabbildung noch eine Konvertierung der Datenstrukturen von VTK auf den Szenegraph von Open Inventor erforderlich. Das Prinzip, diese Konvertierung durchzuführen und damit die Mächtigkeit von VTK für das Gebiet der virtuellen Welten zugänglich zu machen, wurde erstmals in [84] vorgestellt. Dabei wurde die Konvertierung von VTK auf den Szenegraphen von IRIS Performer implementiert. Eine analoge Konvertierung wurde in dieser Arbeit für Open Inventor entwickelt. Diese ist Bestandteil der Klasse `SoVfrDataMapper` und kann somit von allen Klassen zur Datenabbildung am Ende der Pipeline der Visualisierung gemäß Abbildung 7.18 verwendet werden.
Der Quellcode von `VFReal` kann in zwei unterschiedlichen Konfigurationen für beide Bibliotheken übersetzt werden. Die Klassen von *DataViz* bzw. VTK sind in den Klassen `vfrVisGrid` sowie `SoVfrDataMapper` und in deren Ableitungen gekapselt. Eine kurze Bewertung zu diesen beiden Toolkits wird in Abschnitt 7.5 gegeben.

Abbildung von Stromlinien

Exemplarisch wird nun die Klasse zur Abbildung von Stromlinien ausführlicher beschrieben. Weitere Details zur Implementierung befinden sich in [116].
Stromlinien bzw. Partikelpfade werden von so genannten Saatpunkten aus verfolgt, die der Benutzer mit einem *Dragger* interaktiv verändern kann. Zu diesem Zweck wurde eigens ein spezieller *Dragger* entwickelt (dieser wird *StreamlineDragger* genannt – siehe Abbildung 7.19, links oben), der einerseits frei transformierbar ist und gleichzeitig stets eine Ebene beschreibt, auf der die Saatpunkte einer Partikelverfolgung entlang eines Rasters erzeugt werden. Skaliert man den *StreamlineDragger*, so wird die Anzahl der Saatpunkte verändert. Sobald der *StreamlineDragger*

aktiviert wird, erscheint im Benutzermenü von `VFReal` (siehe Abschnitt 7.3.2) ein Schieberegler, mit dem man den Abstand des Rasters verändern kann.

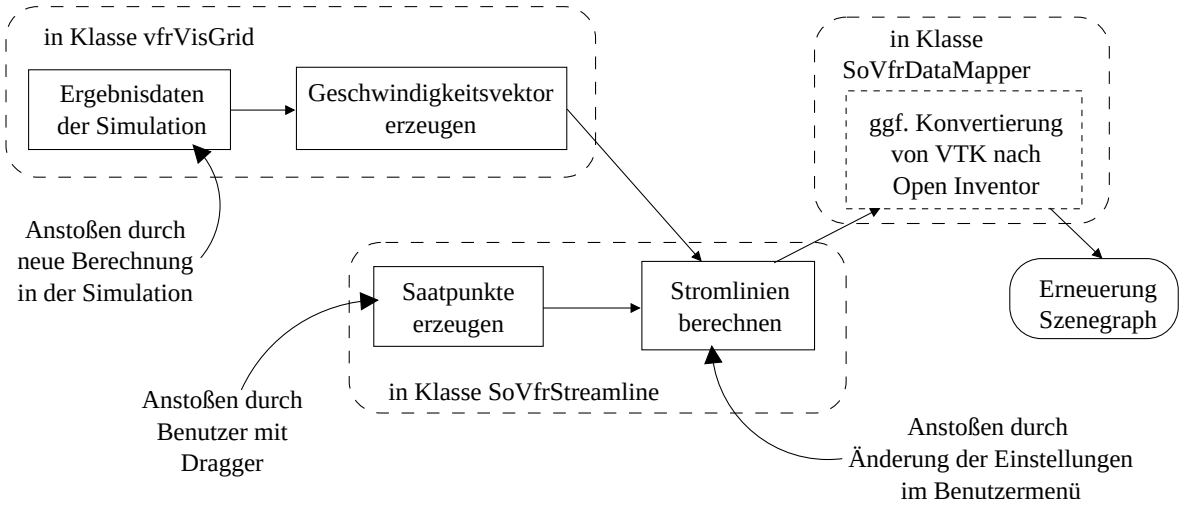

Abbildung 7.18: Möglichkeiten des Startens der Pipeline der Visualisierung von Stromlinien

Bei der Erzeugung bzw. der Veränderung von Stromlinien wird stets die in Abbildung 7.18 skizzierte Pipeline der Visualisierung durchlaufen. In dieser Abbildung ist auch markiert, in welchen Klassen die einzelnen Bestandteile der Pipeline implementiert (bzw. gekapselt) sind. Man beachte weiterhin die verschiedenen Möglichkeiten der Einflussnahme auf den Ablauf der Pipeline. Sobald die Ergebnisse der Berechnung erneuert werden, erfolgt automatisch eine Aktualisierung. Der Benutzer selbst kann die Erzeugung der Stromlinien über den *StreamlineDragger* sowie über das Benutzermenü anstoßen.

Abbildung von Isoflächen

Die zugehörige Algorithmik bei der Generierung von Isoflächen ist in der Klasse `SoVfrIsosurface` gekapselt. Isoflächen eignen sich in interaktiven Simulationen besonders gut, um die zeitliche Entwicklung bzw. die Ausbreitung einer transienten Strömung darzustellen (siehe dazu die Abbildungen in Abschnitt 7.4).

Abbildung von Schnittebenen

Schnittebenen können wiederum sehr schnell und interaktiv mit einem *Dragger* beliebig orientiert im Raum platziert werden. Dies ist insofern wichtig, weil Schnittebenen von den meisten Anwendern als erste Option bei der Analyse der Ergebnisse einer Strömungssimulation verwendet werden, um sich sehr schnell einen qualitativen Eindruck von den Eigenschaften der Strömung zu verschaffen. Schnittebenen sind wie Isoflächen recht gut geeignet, um den zeitlichen Verlauf einer Strömung sichtbar zu machen (siehe dazu die Abbildungen in Abschnitt 7.4).

Abbildung von Vektorpfeilen

Die Abbildung der Strömung mit Vektorpfeilen ist in Abbildung 7.19 (rechts unten) dargestellt. Dazu wird mit einem *Dragger* eine Ebene definiert, auf der die Vektoren in einem festen Raster dargestellt werden.

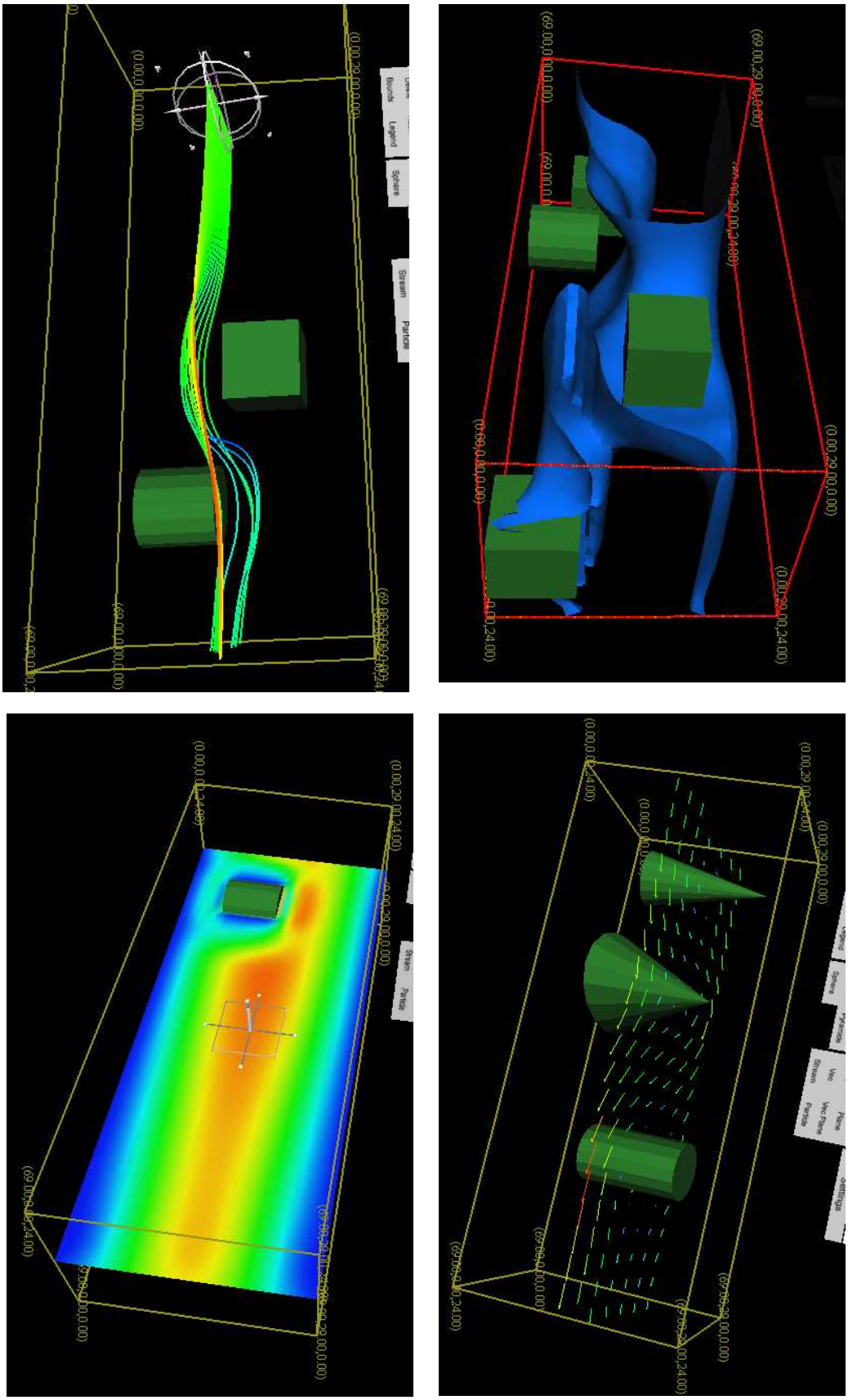

Abbildung 7.19: Beispiele zur Datenabbildung in `VFReal` – Stromlinien (oben links), Isofläche (oben rechts), Schnittebene (unten links), Vektorpfeile (unten rechts)

7.3.5 Zusammenspiel der Komponenten

In den Abschnitten 7.1.2 und 7.3 wurde bereits klar, dass innerhalb des Virtual Reality basierten Teilsystems der Ein- und Ausgabe mehrere Aufgaben parallel ablaufen. Die einzelnen Tasks sind:

1. Erfassung der Eingaben des Benutzers
2. Zeichnen des Szenegraphen auf mehreren Projektionsflächen
3. Austausch der Daten mit der Simulation
4. Neuberechnung der Datenabbildungen

Zur Verdeutlichung der Problematik betrachte man folgendes Szenario: Wartet man auf die Daten vom Teilsystem der Berechnung, dann darf das Rendering auf keinen Fall still stehen. Nach dem Empfang der neuen Daten müssen auf der anderen Seite alle bereits instantiierten Abbildungen der Daten neu berechnet werden. Dies ist ein Vorgang, der durchaus Zeit in Anspruch nehmen kann. Es muss also dafür Sorge getragen werden, dass der Benutzer während derartiger Vorgänge nicht warten muss, sondern sich zumindest weiter durch die Szene bewegen kann. Zu diesem Zweck werden mehrere parallel laufende Threads erzeugt.

Parallele Threads für den Datenaustausch und die Datenabbildung

Durch die Verwendung der Programmier-Bibliothek VRJuggler werden je nach Konfiguration des Systems bereits mehrere Threads erzeugt. Die elementare Funktionalität von VRJuggler befindet sich in der Klasse `vjGlApp` (siehe Abbildung 7.14). Für jede Projektionsfläche werden jeweils zwei Threads (für das linke und das rechte Auge) zum Rendern erzeugt. Für jeden Thread und jedes erneuerte Bild werden die Funktionen

`preFrame` – `draw` – `postFrame`

aufgerufen. Die abgeleitete Klasse `vjInventorApp` (siehe Abbildung 7.14) sorgt nun dafür, dass durch das Überladen der Funktion `draw` auf jeder Projektionsfläche ein Szenegraph von Open Inventor gezeichnet wird. Ein weiterer Thread in der Klasse `vjGlApp` erfasst die Eingaben der angeschlossenen Eingabe-Hardware. Diese Informationen werden dann in der Klasse `vjInventorApp` innerhalb der dafür überladenen Funktion `preFrame` an den Szenegraphen weitergegeben.
Neben den Threads von VRJuggler wird zusätzlich noch ein weiterer Thread erzeugt, um die Kommunikation mit der Simulation durchzuführen. An dieser Stelle ist darauf hinzuweisen, dass zur Übertragung der Ergebnisse der Simulation zwei Speicherbereiche vorhanden sein müssen. Ein Bereich wird von der Klasse `vfrVisGrid` für die Abbildung der Daten verwendet. Der zweite Speicherbereich empfängt zur selben Zeit die neuen Ergebnisse. Sobald diese vollständig angekommen sind, können diese zwei Speicherbereiche durch Tauschen von Zeigern gewechselt werden. Die Datenabbildung wird dann automatisch mit den gerade empfangenen Daten fortgesetzt.
Im Moment der Erneuerung der Ergebnisse der Simulation liegt auf der Seite des Teilsystems der Ein- und Ausgabe eine problematische Situation vor. Es wäre grundsätzlich möglich, dass in einem anderen Thread gerade eine Isofläche berechnet wird. Das Wechseln der Zeiger könnte nun dazu führen, dass sich während des Ablaufs der Generierung der Isofläche der zugehörige Datensatz ändert. Aus diesem Grund wird der Zugriff auf die Felder der Ergebnisdaten mit einem Sperrmechanismus (*Mutex*) geschützt. Damit wird sichergestellt, dass zwei Threads nicht gleichzeitig auf eine Variable, respektive einen Speicherbereich zugreifen können.

Die Abbildung der Daten wird bereits während der Bearbeitung des Szenegraphen nach Veränderungen durch einen `Dragger` oder durch Eingaben mit dem immersiven Benutzermenü (Abschnitt 7.3.2) durchgeführt. Nach der Erneuerung der Ergebnisdaten der Simulation müssen aber alle erzeugten Datenabbildungen neu berechnet werden. Dieser Vorgang kann in der Regel nicht mehr während der Bearbeitung des Szenegraphen durchgeführt werden. Folglich wird ein weiterer Thread erzeugt, der nach dem Empfang der Ergebnisdaten benachrichtigt wird und dann die Abbildungen der Daten erneuert.

7.4 Beispiel zur Anwendung der interaktiven Strömungssimulation

An dieser Stelle wird lediglich eine Bildserie während einer Sitzung einer interaktiven Simulation dargestellt, um zu verdeutlichen, dass man zur Laufzeit der numerischen Simulation neue Geometrien in beliebiger Reihenfolge einfügen, verändern und löschen kann und gleichzeitig die Abbildungen der Daten beliebig manipulieren kann.
Abbildung 7.20, oben zeigt eine Kanalströmung, in die der Benutzer zwei Körpern hineinsetzt. Die neue Strömung wird unmittelbar berechnet und angezeigt (siehe Abbildung 7.20, mitte). Der Benutzer hat jederzeit die Möglichkeit, andere Abbildungsformen zu wählen, wie etwa die Vektorpfeile in Abbildung 7.20, unten oder die Isofläche in Abbildung 7.21, unten. Wird ein geometrisches Objekt versetzt, kann man den Verlauf bis zum neuen stationären Zustand stets mitverfolgen. Abbildung 7.21, oben ist unmittelbar aufgenommen worden, nachdem der Benutzer den Konus von links nach rechts versetzt hat.
Diese Bildserie kann naturgemäß den Charakter einer interaktiven Simulation nur unzureichend widergeben. Einen besseren Eindruck liefert ein im Internet[9] veröffentlichter Film zu `VFReal`.

7.5 Zusammenfassung und Diskussion

In diesem Kapitel wurde ein System zur interaktiven Steuerung und Auswertung einer Lattice-Boltzmann Simulation vorgestellt. Der Schwerpunkt galt dabei der Demonstration der grundsätzlichen Machbarkeit des Eingriffs in eine laufende numerische Simulation der Strömungsmechanik nach der Lattice-Boltzmann Methode, mit den Zielen, weder die Leistungsfähigkeit, noch die Stabilität des Simulationskerns zu gefährden.
Die bestehende Implementierung kann als Gerüst für weitere Entwicklungen verwendet werden und ist in zwei funktionell voneinander getrennte Komponenten aufgeteilt:

Komponente der Berechnung

- Die Leistungsmessungen der Abschnitte 7.2.4 und 7.2.5 zeigen eine sehr hohe parallele Effizienz der Simulation. Durch eine geschickte Implementierung wurde dafür Sorge getragen, dass die Effizienz des Berechnungskerns kaum durch die Komponente der Steuerung und Auswertung beeinträchtigt wird. Auf 5 Rechenknoten der HitachiSR 8000 konnte bereits eine Leistungsfähigkeit von über 35 Millionen Gitterknoten-Erneuerungen pro Sekunde erreicht werden.

[9] http://www.inf.bv.tum.de/personen/petra/iFluids_hardt_tum.exe – Die Gültigkeit dieser Url wurde zuletzt am Tag der Einreichung dieser Arbeit geprüft.

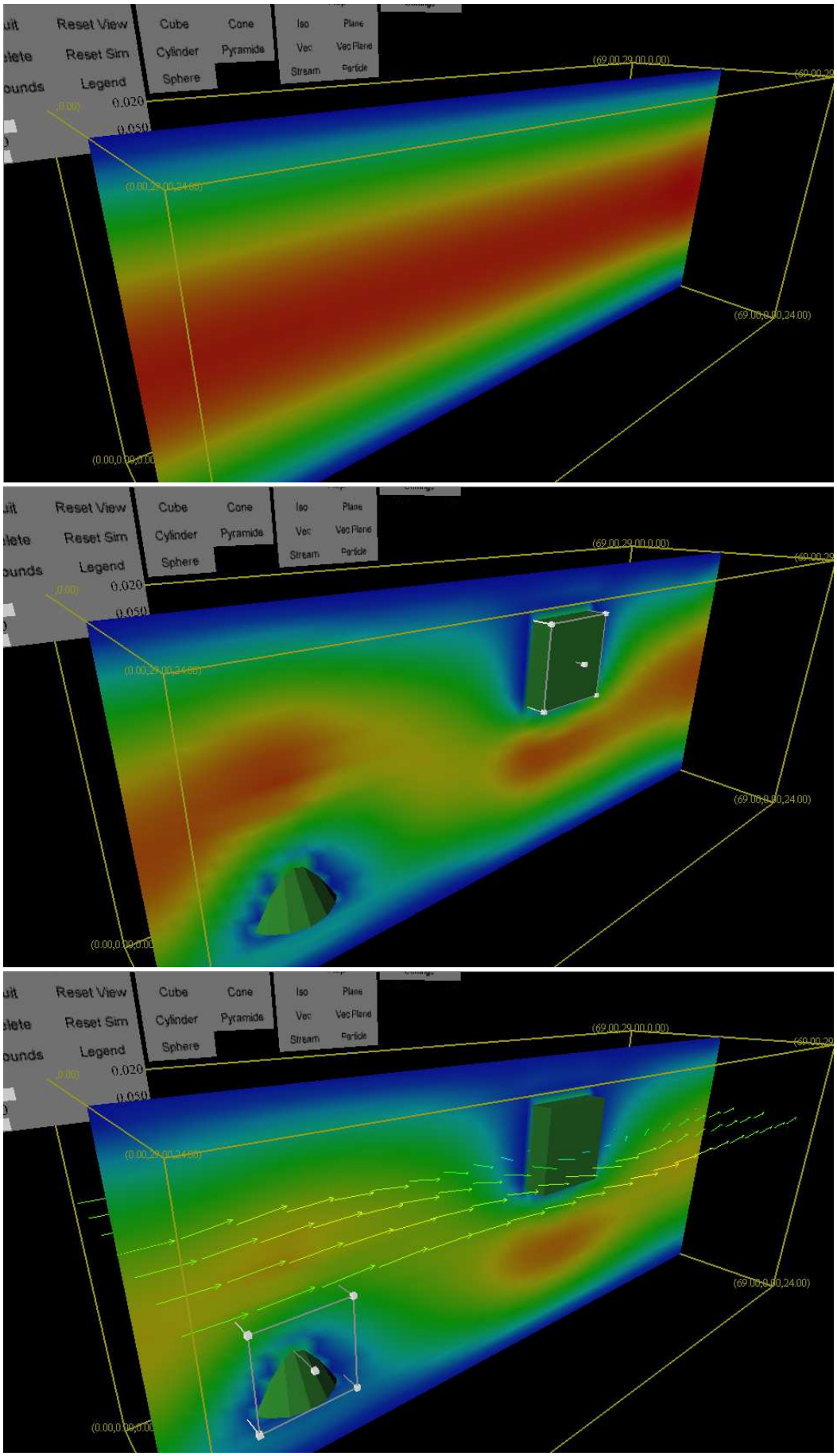

Abbildung 7.20: Anwendung der interaktiven Strömungssimulation – Bildserie 1-3

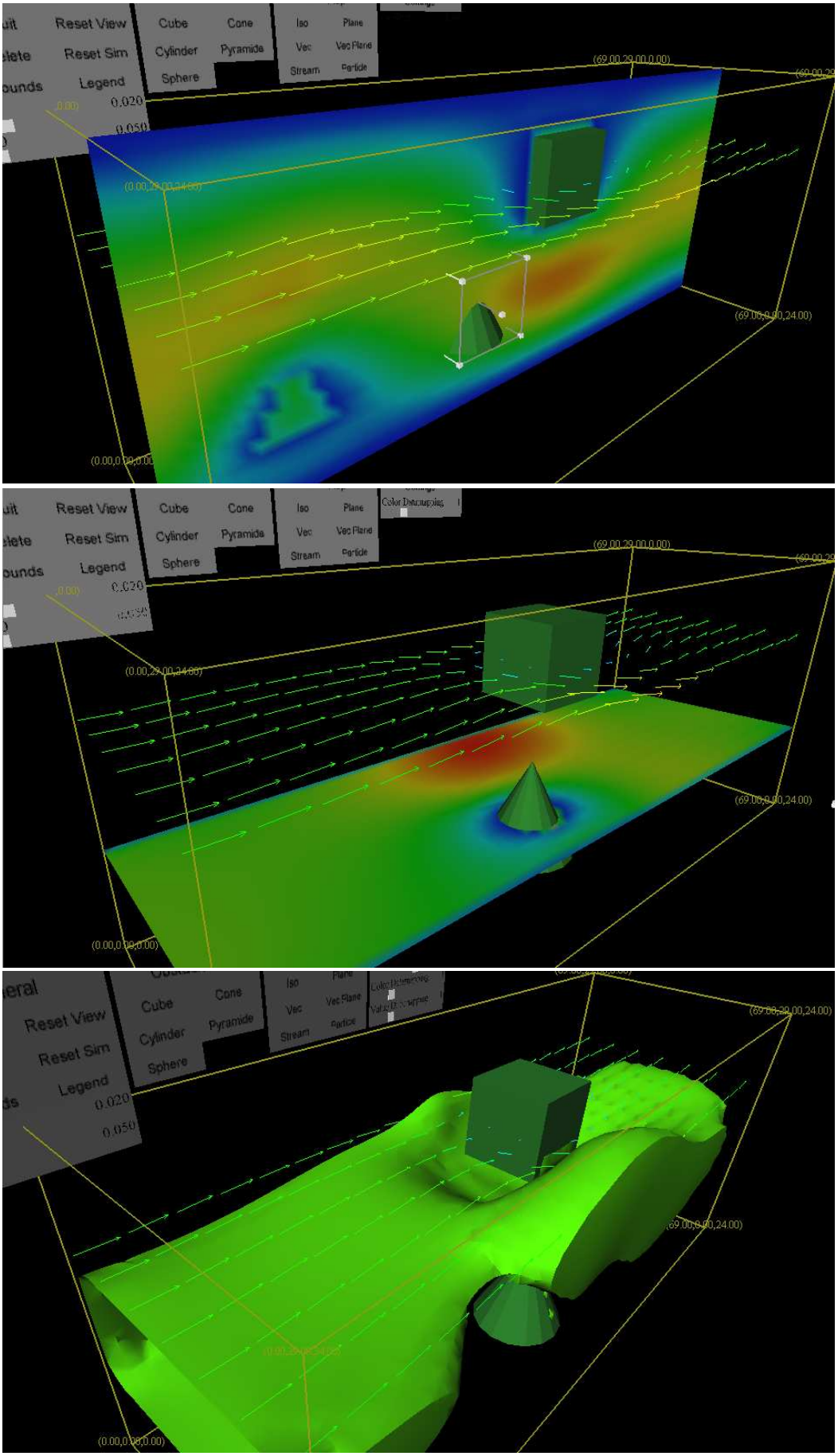

Abbildung 7.21: Anwendung der interaktiven Strömungssimulation – Bildserie 4-6

- Die Gesamtleistung des System ist stark von der Leistungsfähigkeit der zugrunde liegenden, seriell arbeitenden Version des Berechnungskerns abhängig. Bei den hier relevanten Problemgrößen von bis zu 1 Millionen Gitterknoten konnte auf der Hitachi SR8000 eine Leistungsfähigkeit von 7-7,5 Millionen Gitterknoten-Erneuerungen pro Sekunde erzielt werden. Eine weitere Leistungssteigerung könnte man durch die Anwendung der in [96] vorgeschlagenen Methodik erreichen. Dabei werden die Primärvariablen der Simulation nicht mit vollbesetzten Matrizen gespeichert, sondern in einem langen, eindimensionalen Vektor angelegt. Damit wird eine wesentlich bessere Vektorisierung ermöglicht. In diesem Fall erschwert sich aber die Modifikation des Gitters innerhalb der interaktiven Simulation.

- Der wichtigste Schritt bei der Weiterentwicklung dieses Prototyps zu einem produktiv nutzbaren System liegt im Bereich der geometrischen Modellierung. Anstelle von CSG-Grundkörpern werden dann die geometrischen Hindernisse im Strömungsgebiet mit einer B-Rep Oberflächenbeschreibung modelliert und mit Hilfe von Oktalbaum-Datenstrukturen in ein diskretes Gitter überführt.

- Im Berechnungskern ist die Momentenmethode des Lattice-Boltzmann Verfahrens eingebaut. Damit erhält man eine bessere Stabilität sowie eine schnellere Konvergenz der Simulation. Zukünftig müssen noch Turbulenzmodelle integriert werden. In diesem Kontext entsteht aber das Problem, dass bei der Auswertung der makroskopischen Größen Druck und Geschwindigkeit, ein Mittelwert über einen längeren Zeitraum gebildet werden muss, innerhalb dem aber die geometrischen Objekte im Strömungsgebiet verändert werden könnten.

Komponente der Steuerung und Auswertung

- Die Komponente der Steuerung und Auswertung ist hier in eine Virtual Reality Umgebung eingebettet. VR bietet Vorteile bei der Interaktion mit dreidimensionalen Objekten und hat eine äußerst suggestive Wirkung. Durch die Verwendung der Zwischenschicht VRJuggler ist diese Komponente leicht auf verschiedenste VR-Anlagen portierbar.

- Mit Schnittebenen, Stromlinien, Isoflächen und Vektorpfeilen sind die wichtigsten Methoden der Abbildung von Datensätzen einer Strömungssimulation enthalten (siehe Abschnitt 7.3.4). Dabei wurde die Erweiterung *DataViz* von TGS Open Inventor eingesetzt, denn es hat sich gezeigt, dass die Bibliothek VTK für die Zwecke der interaktiven Steuerung und Auswertung zu langsam ist.

- Die Berechnung der Datenabbildung sowie der Datentransfer von und zur Komponente der Berechnung wird mit parallel arbeitenden Threads ("Programmfäden") durchgeführt, um ein flüssiges Rendering zu ermöglichen.

In Zukunft kann auch der Aspekt des kooperativen Arbeitens mit einer interaktiven Strömungssimulation eine wichtige Rolle spielen. Mehrere Anwender sollen sich dann von verschiedenen Standorten aus mit der Simulation verbinden können und in diese eingreifen können.

Kapitel 8

Zusammenfassung und Ausblick

8.1 Zusammenfassung

Zur Optimierung des Zyklus der Produktauslegung von Bauwerken hinsichtlich strömungsmechanischer Eigenschaften wurden in dieser Arbeit die Virtual Reality basierte Auswertung von Ergebnisdaten und die interaktive Steuerung einer numerischen Simulationen nach der Lattice-Boltzmann Methode präsentiert. Im Mittelpunkt steht ein Prototyp eines Computational Steering Systems, der die grundsätzliche Machbarkeit von interaktiven Strömungssimulationen demonstriert und als Basis für weitere Entwicklungen dient.

Maßgebliche Entscheidungen innerhalb des Planungsprozesses im Bauwesen werden in einer frühen Phase der Planung gefällt und sind oftmals zwischen einer Vielzahl von Beteiligten aus unterschiedlichsten Disziplinen abzustimmen. Schnelle Entscheidungsprozesse hinsichtlich fluidmechanischer Berechnungen sind einerseits durch eine qualitativ hochwertige Datenauswertung und anderseits durch strukturelle Optimierungen des Gesamtablaufs der numerischen Strömungssimulation zu unterstützen. Die zuletzt genannte Optimierung zeichnet sich durch die Durchgängigkeit der Schnittstellen zwischen der Datenaufbereitung, der Berechnung und der Visualisierung aus. Dies legt den Wunsch nahe, den Gesamtprozess einer numerischen Strömungsberechnung in einer Applikation zu verbinden, also eine interaktive Simulation bereitzustellen.

Die wesentliche Schwierigkeit im obigen Kontext liegt im Übergang vom geometrischen Modell zum numerischen Gitter, insbesondere im Umfeld der komplexen und vielfältigen geometrischen Ausprägung von Bauwerken. Mit der Lattice-Boltzmann Methode wurde ein Verfahren zur numerischen Strömungssimulation gewählt, für das man Geometrien aus einem CAD-Modell in ein diskretes Voxelmodell übertragen kann, respektive diesen Prozess für interaktive Simulationen weitgehend automatisieren kann. In der eng verzahnten Dissertation von Crouse [25] wurden für diese Voxelierung hierarchische Datenstrukturen als Organisationsprinzip verwendet.

Eine Abschätzung der Rechenzeiten zeigte jedoch, dass interaktive Simulationen der Strömungsmechanik für praxisrelevante Anwendungen dann noch nicht möglich sind, wenn hohe Anforderungen an die Genauigkeit gestellt werden. Daher wurde die folgende zweistufige Strategie vorgeschlagen: Interaktive Simulationen (Online-Version, erste Stufe) werden zur effizienten Vorstudie eingesetzt. Bei moderater Genauigkeit können verschiedene Konzepte schnell diskutiert werden. Anschließend ist der Vorentwurf aus der interaktiven Simulation in der zweiten Stufe (Offline-Version) bei höherer Genauigkeit nochmals nachzurechnen.

Für diese Offline-Version wurden in Kapitel 6 Entwicklungen im Bereich der interaktiven Analyse der Ergebnisse präsentiert. Dazu wurde eine Octree-Datenstruktur mit zusätzlicher Vernetzung vorgestellt, die in [25] für die Voxelierung des CAD-Modells diente. Damit können die Vorteile der hierarchischen Datenstrukturen mit denen von Netzen kombiniert werden. Diese Datenstrukturen wurden verwendet, um nach der Berechnung die Gitter mit den diskreten Ergebnissen durch Subsampling-Techniken bis auf 10% der ursprünglichen Zellenzahl zurückzuführen, ohne einen sichtbaren Qualitätsverlust zu erhalten. Gängige Algorithmen zur Datenvisualisierung

wurden ebenfalls für diese Datenstruktur angepasst. Dies führt bei komplexen Problemen dann zu einer Beschleunigung, wenn die resultierende Diskretisierung eine wesentlich geringere Anzahl an Freiheitsgraden mit sich bringt. Es wurde anschließend gezeigt, dass die Darstellung von abgeleiteten Größen zur Darstellung des menschlichen Komfortempfindens innerhalb klassischer Visualisierungstools einen Mehrwert für den untersuchenden Ingenieur liefert. Durch die kombinierte Darstellung eines hochaufgelösten CAD-Modells zusammen mit den Abbildungen der Ergebnisse der Simulation wird die Beziehung der umgebenden Geometrie zu den physikalischen Eigenschaften der Strömung gut verständlich gemacht. Die Anwendbarkeit dieser Werkzeuge wurde an einem komplexen Beispiel, der Simulation eines Großraumbüros demonstriert.
In Kapitel 7 wurde schließlich ein Prototyp zur interaktiven Steuerung einer numerischen Strömungssimulation (Online-Version, siehe oben) beschrieben. Als Machbarkeitsstudie ging diese Version vorerst noch zurück auf eine Diskretisierung mit uniformen kartesischen Gittern auf der Basis von vollbesetzten Matrizen. Der Entwurf des Systems sah vor, die Berechnung auf einem Supercomputer oder gegebenenfalls auf einem Workstation-Cluster massiv parallel durchzuführen. Der Datenaustausch erfolgte über eine Interprozesskommunikation. Das zu simulierende Gebiet wurde als Kanal modelliert, in den beliebig viele Körper gesetzt und interaktiv verändert werden können. Aktuell handelt es sich dabei noch um die Grundkörper Hexaeder, Kugel, Konus, Zylinder und Pyramide, die im Sinne eines CSG[1]-Systems zu komplexeren Hindernissen kombiniert werden können. Die Gittererneuerung erfolgt durch Modifikation eines Markierungsfeldes mit Hilfe von Algorithmen aus dem Bereich der Rastergraphik. Die Gebietszerlegung wird in Blöcken entlang des Kanals durchgeführt. Die zweite Komponente vereinigt die Eingabe der Steuerparameter und die Visualisierung der Ergebnisse. Sie ist in eine Virtual Reality Umgebung integriert, kann aber ohne zusätzlichen Aufwand für einen Einzelplatzrechner konfiguriert werden. Die Komponente zur Ein- und Ausgabe ist in mehrere Threads unterteilt, so dass der Datentransfer von und zur Simulation vom Rendering unabhängig ist und dem Benutzer nach außen hin verborgen bleibt. Die Messungen der Leistung des Systems zeigten eine sehr gute parallele Effizienz, während die Vektorisierung für den Supercomputer Hitachi SR 8000 zwar hinreichend gut ist, aber noch Spielraum für Optimierungen offen lässt. Der Datentransfer von der Rechenkomponente zum Teilsystem der Ein- und Ausgabe stellt zwar einen Flaschenhals dar, wird aber parallel zur Berechnung von neuen Zeitschritten durchgeführt. In der Anwendung verhält sich das System sehr stabil gegenüber häufigen Interaktionen eines Benutzers.

8.2 Ausblick

Das hier beschriebene Projekt stellt erst eine Basis für weitere Forschungs- und Entwicklungsarbeiten da. Die interaktive Analyse der Daten (siehe Kapitel 6) selbst bietet natürlich immer noch Potential für weitere Verbesserungen, doch ist sie langfristig als Teil eines Computational–Steering Systems zur interaktiven Steuerung von Strömungssimulationen zu betrachten. Auf jeden Fall sollte die volle Breite der Entwicklungen von Kapitel 6 in die Komponente der Visualisierung einer interaktiven Simulation integriert werden.
Das Fernziel wurde bereits in Abschnitt 5.3 formuliert: Alle Komponenten des Gesamtkonzepts von Abschnitt 5.2 auf der Basis von hierarchischen Datenstrukturen sind in ein System zur interaktiven Steuerung einer numerischen Strömungssimulation zu integrieren.
Als erste Erweiterung sollte dies für die Gittergenerierung geschehen, um die geometrischen Hindernisse nicht weiter mit CSG-Körpern, sondern durch ein B-Rep Volumenmodell zu beschreiben. Die vollständige Erzeugung des Gitters mit einem neuen Aufbau eines Oktalbaums macht hierbei

[1] CSG: Constructive Solid Geometry

wenig Sinn und ist zu aufwendig. Ein pragmatischer Ansatz wäre, die geometrischen Objekte im Strömungssfeld aufzuteilen: Dabei gibt es nicht-modifizierbare Objekte (z.B. eine Wand in einem Raum), für die vorweg ein Gitter (*fixe Objekte*) mit Hilfe eines Octree-Generators erzeugt wird und fest im System gespeichert wird. Zusätzlich gibt es modifizierbare Objekte (z.B. ein Tisch oder ein Stuhl), die während der Simulation interaktiv transformiert werden können (*modifizierbare Objekte*). Für alle diese Objekte gibt es in einer Bibliothek ein vorberechnetes lokales Gitter und einen zugehörigen Oktalbaum. Das gesamte Gitter der Simulation wird dann zur Laufzeit der Simulation folgendermaßen modifiziert: Auf eine Transformation durch den Benutzer folgt eine entsprechende lokale Transformation des Octrees des zu manipulierenden geometrischen Objekts. Anschließend wird dieser lokale Octree des modifizierbaren Objekts mit dem Octree der fixen Objekte durch eine boolsche Operation zu einem Gesamt–Octree vereinigt und davon das numerische Gitter abgeleitet.
Danach können die hierarchischen Datenstrukturen in der Visualisierung eingesetzt werden (siehe oben). In diesem Zusammenhang ist der Prozess der Datenreduktion sehr wichtig. In Abschnitt 7.5 wurde die Kommunikation zwischen der Simulation und der Visualisierung als problematisch bewertet, weil das komplette Vektorfeld versendet werden muss. Bei 500.000 Gitterpunkten benötigt man zur Übertragung des Geschwindigkeitsfeldes und des Druckes bereits insgesamt 8 Mega–Byte. Eine effiziente Datenreduktion kann diese Menge auf ca. 10 % reduzieren ohne dabei wesentlich an Qualität zu verlieren.
Im Bereich der Strömungssimulation wird die Berechnung mit Octree-Datenstrukturen erst mittelfristig möglich sein. Vorher müsste man bei dieser Komponente neuere Entwicklungen der Lattice-Boltzmann Methode einbauen. Dazu gehört in erster Linie ein Modell zur Simulation von temperaturbehafteten Strömungen, weil der Einsatz der interaktiven Simulation sich aufgrund der erheblich größeren Rechenzeiten von Aussenströmungen vorerst verstärkt auf Strömungen in Innenräumen konzentrieren wird. Diese Strömungen sind sehr stark durch Konvektion aufgrund von Heizquellen getrieben. Weiterhin gilt es, Turbulenzmodelle in den Simulationskern zu integrieren. Das ist bei Lattice-Boltzmann Simulationen zunächst recht einfach zu implementieren, allerdings muss bei der Auswertung ein Mittelwert der Primärvariablen über die Zeit gebildet werden, was zusätzliche Überlegungen hinsichtlich einer effizienten Datenhaltung erfordert. Bei der Verwendung von allgemeinen geometrischen Objekten (siehe oben) als Hindernisse im Strömungsgebiet ist es sinnvoll, den wahren Abstand eines diskreten Punktes zu einer benachbarten Oberfläche einer Geometrie zu speichern. Auf diese Art kann man eine spezielle Haftrandbedingung mit einer Approximationsgüte zweiter Ordnung erzielen (siehe [72]).
In Abschnitt 4.2 wurde bereits das kooperative Arbeiten mehrerer Teilnehmer an einer interaktiven Simulation gefordert. Hier sind räumlich voneinander getrennte Personen gemeint, wobei jeder von seinem Rechner aus in die Simulation eingreifen und Vorschläge machen kann. Die Interaktionen und die zugehörige Visualisierung müssen dann für alle Teilnehmer synchronisiert werden. Es soll eine (formal) unbegrenzte Anzahl von Teilnehmern zugelassen werden, die sich jederzeit in eine laufende Simulation ein- und ausloggen können.
Zu guter Letzt ist der Einsatz in der Praxis an einem konkreten Beispiel zu testen. Dazu ist eine Kooperation zwischen Hochschule und (mehreren) Industriepartnern unerlässlich, damit die Demonstration der Machbarkeit nicht nur anhand von akademischen Beispielen erfolgt. So wäre es von größtem Interesse, die Effizienz der Produktauslegung auf der Basis der zweistufigen Strategie von Abbildung 5.6 mit dem klassischen Vorgehen von Abbildung 3.1 zu vergleichen. Als konkreter Anwendungsfall könnte dabei die Auslegung einer Klimaanlage zum Einsatz kommen.

Literaturverzeichnis

[1] Bibliography for NATO Workshops on LG–methods. In: *Physica D* 47 (1991), S. 299–337

[2] *Gemeinsame Sprache des Bauwesens setzt sich durch.* Pressemitteilung der 'International Alliance for Interoperability' vom 07.11.2001. 2001

[3] Visit - Wege zum Interagieren in VR. In: *Digital Engineering Magazin 5/01* (2001), S. 68–69

[4] *DIN 69900 - Netzplantechnik.* DIN - Deutsches Institut für Normung. 2002

[5] Alt, F. : *Automatische Erstellung einer multimedialen Dokumentation von Ergebnissen numerischer Simulationen*, Diplomarbeit, Lehrstuhl für Bauinformatik, Technische Universität München, Diplomarbeit, 2002

[6] Aukstakalnis, S. ; Blatner, D. : *The Silicon Mirage, The Art and Science of Virtual Reality.* Peachpit Press, Inc., 1992

[7] Bhatnagar, P. ; Gross, E. ; Krook, M. : A model for collision processes in gases. In: *Phys. Rev.* 94 (1954), S. 511–525

[8] Breitling, P. : *Entwurf und Implementierung eines objektorientierten Frameworks als Schnittstelle zur Visualisierung hierarchisch adaptiver Datenstrukturen*, Institut für Informatik, TU München, Diplomarbeit, 1998

[9] Bresenham, J. E.: Algorithm for Computer Control of Digital Plotter. In: *IBM System Journal* 4 (1965), Nr. 1

[10] Bröker, H. : *Integration von geometrischer Modellierung und Berechnung nach der p-Version der FEM*, Lehrstuhl für Bauinformatik, Technische Universität München, Dissertation, 2001

[11] Brooks, F. P.: Grasping Reality Through Illusion - Interactive Graphics Serving Science. In: ACM (Hrsg.): *CHI'88 Proceedings*, 1988, S. 1–11

[12] Brooks, F. P.: What's Real about Virtual Reality? In: *IEEE Computer Graphics and Applications* 19 (1999), Nr. 6, S. 16–27

[13] Brooks Jr, F. P. ; Airey, J. ; Alspaugh, J. ; Bell, A. ; Brown, R. ; Hill, C. ; Nimscheck, U. ; Rheingans, P. ; Rohlf, J. ; Smith, D. ; Turner, D. ; Varshney, A. ; Wang, Y. ; Weber, H. ; Yuan, X. : Six Generations of Building Walkthrough: Final Technical Report to the National Science Foundation / Department of Computer Science, University of North Carolina. 1992 (TR92-026). – Forschungsbericht

[14] Brown, J. R. ; Ernshaw, R. ; Jern, M. ; Vince, J. : *Visualization - Using Computer Graphics to Explore Data and Present Information.* New York : Wiley & Sons, 1995

[15] Bryson, S. : Virtual Reality: A Definition History, 2001. – Persönliche Stellungnahme, http://www.fourthwavegroup.com/fwg/lexicon/1725w1.htm

[16] Bryson, S. ; Levit, C. : The Virtual Windtunnel: An environment for the exploration of three-dimensional unsteady fluid flow. In: *IEEE Computer Graphics and Applications* 12 (1992), Nr. 4, S. 25–34

[17] Bungartz, H.-J. ; Griebel, M. ; Zenger, C. : *Einführung in die Computergraphik.* Vieweg-Verlag, 2002

[18] Cabral, B. ; Leedom, L. C.: Imaging vector fields using line integral convolution. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM Press, 1993, S. 263–270

[19] Chan, T. ; Mathew, T. : Domain Decomposition Algorithms. In: *Acta Numerica* (1994), S. 61–143

[20] Chapman, S. ; Cowling, T. : *The Mathematical Theory of Non-Uniform Gases.* Cambridge University Press, 1990

[21] Connell, M. ; Tullberg, O. : A Framework for the Interactive Investigation of Finite Element Simulations within Virtual Environments. In: *Engineering Computational Technology*, 2000

[22] Connell, M. ; Tullberg, O. ; Kettil, P. ; Wiberg, N.-E. : Interactive Design and Investigation of Physical Bridges using Virtual Models. In: *First MIT Conference on Computational Fluid and Solid Mechanics*, 2001, S. 608–611

[23] Crouse, B. ; Krafczyk, M. ; Kühner, S. ; Rank, E. ; van Treeck, C. : Indoor air flow analysis based on Lattice Boltzmann methods. In: *Int. Journal of Energy and Buildings* 34 (2002), Nr. 9, S. 941–949

[24] Crouse, B. : Baumdatenstrukturen in strömungsmechanischen Simulationen. In: *Forum Bauinformatik 2000.* Berlin : VDI–Verlag, 2000

[25] Crouse, B. : *Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen*, Lehrstuhl für Bauinformatik, Technische Universität München, Dissertation, 2003

[26] Crouse, B. ; Krafczyk, M. ; Tölke, J. ; Rank, E. : A LB-based approach for Adaptive Flow Simulations. In: *International Journal of Modern Physics B* (2002)

[27] Crouse, B. ; Kühner, S. : Simulation und Visualisierung von Windströmungen um Bauwerke. In: *Forum Bauinformatik 1999.* Darmstadt : VDI–Verlag, 1999

[28] Cruz-Neira, C. ; Sandin, D. J. ; DeFanti, T. A.: Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In: *Proceedings of SIGGRAPH-93: Computer Graphics.* Anaheim, CA, 1993, S. 135–142

[29] Deuflhard, P. ; Bornemann, F. : *Numerische Mathematik II, Integration gewöhnlicher Differentialgleichungen.* de Gruyter, 1994

[30] d'Humières, D. ; Ginzburg, I. ; Krafczyk, M. ; Lallemand, P. ; Luo, L. : 3D Multiple-Relaxation-Time LBE Models. In: *Philosophical Transactions of the Royal Society of London A* 360 (2002), S. 437–451

[31] Durst, F. : Necessity for Modern Numerical Methods in Fluid Mechanics, 2001. – Vorlesung am 'DFG und KONWIHR Workshop zu Lattice–Boltzmann Methoden'

[32] Düster, A. : *High order finite elements for three-dimensional, thin-walled nonlinear continua*, Lehrstuhl für Bauinformatik, Technische Universität München, Dissertation, 2002

[33] Fanger, P. : *Thermal Comfort. Analysis and Application in Environmental engineering.* McGraw-Hill, 1970

[34] Ferziger, J. ; Peric, M. : *Computational Methods for Fluid Dynamics.* Berlin : Springer Verlag, 2002

[35] Filipova, O. ; Hänel, D. : A novel Lattice BGK approach for low mach number combustion. In: *Computational Physics* 2 (2000), Nr. 158, S. 139–160

[36] Filippova, O. ; Hänel, D. : Grid refinement for Lattice-BGK models. In: *Journal of Computational Physics* 147 (1998), S. 219–228

[37] Flynn, M. J. ; Rudd, K. W.: Parallel architectures. In: *ACM Computing Surveys* 28 (1996), Nr. 1, S. 67–70

[38] Foley, J. D.: *Introduction to Computer Graphics.* Addison-Wesley, 1997

[39] Frank, A. : *Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung*, Institut für Informatik, Technische Universität München, Dissertation, 2000

[40] Freudiger, S. : *Effiziente Datenstrukturen für Lattice-Boltzmann-Simulationen in der computergestützten Strömungsmechanik*, Lehrstuhl für Bauinformatik, Technische Universität München, Diplomarbeit, 2001

[41] Geist, G. ; Kohl, J. ; Papadopoulos, P. : CUMULVS: Providing fault-tolerance, visualization and steering of parallel applications. In: *The International Journal of Supercomputer Applications and High Performance Computing* 11 (1997), Nr. 3, S. 224–235

[42] Greiner, P. ; Mayer, P. ; Stark, K. : *Baubetriebslehre, Projektmanagement.* Vieweg Verlagsgesellschaft, 2000

[43] Griebel, M. ; Dornseifer, T. ; Neunhoeffer, T. : *Numerische Simulation in der Strömungsmechanik, eine praxisorientierte Einführung.* Braunschweig : Vieweg, 1995

[44] Güting, R. H.: *Datenstrukturen und Algorithmen.* Stuttgart : B. G. Teubner, 1992

[45] Halfmann, A. : *Ein geometrisches Modell zur numerischen Simulation der Fluid-Struktur-Interaktion windbelasteter, leichter Flächentragwerke*, Lehrstuhl für Bauinformatik, Technische Universität München, Dissertation, 2002

[46] Hardt, P. : *Entwicklung eines Moduls zur Definition strömungsmechanischer Randbedingungen als Attribute von 3D CAD-Geometrien*, Lehrstuhl für Bauinformatik, Technische Universität München, Diplomarbeit, 2001

[47] Hentschel, R. : *Entwicklung und Anwendung eines dreidimensionalen selbstadaptiven Verfahrens zur Simulation viskoser Strömungen auf der Basis strukturierter Gitter*, Institut für Aerodynamik und Gasdynamik, Universität Stuttgart, Dissertation, 1996

[48] Hitachi, Ltd.: *Super Technical Server, Hitachi SR8000 tuning manual (C language version)*

[49] Hoppe, A. : Persönliche Mitteilung, 2002. – Funktionsauslegung Crashsimulation, Audi AG, Ingolstadt

[50] Hossfeld, F. : Komplexität und Berechenbarkeit: Über die Möglichkeit und Grenzen des Computers, 1999. – Vortrag an der Nordrhein-Westfälischen Akademie der Wissenschaften am 06.10.1999

[51] Hou, S. ; Sterling, J. ; Chen, S. ; Doolen, G. : A lattice subgrid model for high Reynolds number flows. In: *Fields Institute Communications* 6 (1996), S. 151–166

[52] IEEE (Veranst.): *The first Information Visualization.* Los Alamitos : IEEE Computer Society Press, 1995

[53] Jaksch, S. : *Facettierung dreidimensionaler Gebiete und Gittergenerierung unter Verwendung von Octree-Datenstrukturen*, Lehrstuhl für Bauinformatik, Technische Universität München, Diplomarbeit, 2001

[54] Karl, M. : *Grundlegende Untersuchungen zu Workstation-Clustern und deren Verwendung bei Strömungssimulationen sowie Implementierung einer LAM-MPI Applikation zur Datenausdünnung*, Lehrstuhl für Bauinformatik, Technische Universität München, Diplomarbeit, 2001

[55] Klimetzek, F. : Virtual Intuitive Simulation Testbed VISiT, 2001. – Final Report Esprit-Projekt 28247

[56] Koeppl, S. ; Seidenfad, T. ; Zabel, M. : *Visualisierung von Produktmodelldaten mit VRML auf einer Holobench*, Lehrstuhl für Bauinformatik, Technische Universität München, Studienarbeit, 2001

[57] Krafczyk, M. : *Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung*, Lehrstuhl für Bauinformatik, Technische Universität München, Habilitationsschrift, 2001

[58] Krafczyk, M. : Persönliche Mitteilung, 2002. – Institut für Computeranwendungen im Bauwesen, Technische Universität Braunschweig

[59] Kühner, S. ; Crouse, B. ; Tölke, J. ; Krafczyk, M. ; Rank, E. : From a product model to visualization: Simulation of indoor flows with Lattice-Boltzmann-Methods. In: *Journal of Computer–Aided and Infrastructure Engineering (zur Veröffentlichung eingereicht)* (2002)

[60] Kühner, S. ; Krafczyk, M. : VirtualFluids - An environment for integral visualization and analysis of CAD and simulation data. In: Girod, B. (Hrsg.) ; Greiner, G. (Hrsg.) ; Niemann, H. (Hrsg.) ; Seidel, H. (Hrsg.): *Vision, Modelling and Visualization 2000.* Saarbrücken, 2000, S. 311–318

[61] Kühner, S. ; Rank, E. ; Krafczyk, M. : Efficient reduction of 3D simulation results based on spacetree data structures for data analysis in Virtual Reality environments. In: *Applied Virtual Reality in Engineering and Construction.* Göteburg, 2001, S. 128–135

[62] Lallemand, P. ; Luo, L. : Theory of the Lattice Boltzmann Method: Dispersion, dissipation, isotropy, Galilean invariance, and stability. In: *Physical Review E* 61 (2000), S. 6546–6562

[63] Lanfer, T. : Concepts, Programming, Optimization and Tuning, 2000. – Tutorial von Hitachi Europe GmbH am LRZ München, http://www.lrz.de

[64] van Liere, R. ; Mulder, J. : Ubiquitous Computational Steering. In: *Workshop on PC Graphics, IEEE Visualization '97*, 1997

[65] Lorensen, W. ; Cline, H. : Marching Cubes: A High Resolution Surface Reconstruction Algorithm. In: *ACM Computer Graphics* 21 (1987), Nr. 4, S. 163–169

[66] MacDonald, N. ; Minty, E. ; Malard, J. ; Harding, T. ; Brown, S. ; Antonioletti, M. : Writing message passing parallel programs with MPI, 2002. – Kursunterlagen – EPCC, Univerity of Edinbourgh, http://www.epcc.ed.ac.uk

[67] Martinuzzi, R. : *Experimentelle Untersuchung der Umströmung wandgebundener, rechteckiger, prismatischer Hindernisse*, Lehrstuhl für Strömungsmechanik, Universität Erlangen, Dissertation, 1992

[68] McCormick, B. H. ; DeFanti, T. A. ; Brown, M. D.: Visualization in Scientific Computing. In: *Computer Graphics* 21 (1987), Nr. 6, S. 1–14

[69] McNamara, G. ; Zanetti, G. : Use of the Boltzmann Equation to simulate Lattice-Gas Automata. In: *Phys. Rev. Lett.* 61 (1988), S. 2332–2335

[70] Meagher, D. : Geometric Modelling using Octree Encoding. In: *Computer Graphics and Image Processing* 19 (1982), Nr. 2, S. 129–147

[71] Mei, R. ; Luo, L.-S. ; Shyy, W. : A multi-block Lattice-Boltzmann method for fluid flows. In: *Proceedings of AIAA Fluids 2000 Meeting in Denver*, 2000

[72] Mei, R. ; Shyy, W. ; Yu, D. ; Luo, L. : Lattice Boltzmann Method for 3-D Flows with Curved Boundary. In: *Journal of Computational Physics* 161 (2000)

[73] Mulder, J. D. ; van Wijk, J. J. ; van Liere, R. : A Survey of Computational Steering Environments. In: *Future generation computer systems* 15 (1999), Nr. 2, S. 119–129

[74] Negrao, C. ; Carvalho, C. ; Melo, C. : Numerical analysis of human thermal comfort inside occupied spaces. In: *Building Simulation 1999, 6th IBPSA Conference*, 1999

[75] von Neumann, J. : The Principles of Large-Scale Computing Machines. In: *Ann. Hist. Comp (reprinted)* 3 (1946), Nr. 3, S. 263–273

[76] Nie, X. ; Qian, Y. ; Doolen, G. ; Chen, S. : Lattice Boltzmann simulation of the two-dimensional Rayleigh-Taylor instability. In: *Physical Review E* 58 (1998), November, Nr. 5, S. 6861–6864

[77] Nielson, G. M. ; Hagen, H. ; Müller, H. : *Scientific Visualization – Overview, Methodologies, Techniques*. Los Alamitos : IEEE Computer Society Press, 1997

[78] Oestereich, B. : *Objektorientierte Softwareentwicklung: Analyse und Design mit der UML*. Bd. 5. München : Verlag R. Oldenbourg, 2001

[79] Olbrich, S. ; Pralle, H. : Verteilte Visualisierung von ingenieurwissenschaftlichen Simulationsergebnissen. In: Wriggers, P. (Hrsg.): *ICCES-Kolloquium '01, Hannover - Geometrische Modellierung / Visualisierung / Software*, 2001

[80] Ottmann, T. ; Widmayer, P. : *Algorithmen und Datenstrukturen*. 3. Heidelberg : Spektrum Akad. Verlag, 1996

[81] Pernpeintner, A. : *Gebäudeaerodynamik und Schadstoffausbreitung*, Lehrstuhl für Fluidmechanik, Technische Universität München, Skript zur Vorlesung, 2002

[82] Qian, Y. ; d'Humières, D. ; Lallemand, P. : Lattice BGK for Navier-Stokes equation. In: *Europhysics Letters* 17 (1992), Nr. 6, S. 479–484

[83] Rabenseifner, R. : Introduction to the message passing interface (MPI), 2002. – Kursunterlagen – HLRS Stuttgart, http://www.hlrs.de

[84] Rajlich, P. : *An object oriented approach to developing Visualization tools portable across desktop and virtual environments*, University of Illinois at Urbana-Champaign, Masters Thesis, 1998

[85] Rank, E. ; Bröker, H. ; Düster, A. ; Rücker, M. : Modellierungs- und Berechnungssoftware für den konstruktiven Ingenieurbau: Die p-Version und geometrische Elemente. In: *Bauingenieur* (2000)

[86] Rank, E. : Persönliche Mitteilung, 2002. – Lehrstuhl für Bauinformatik, Technische Universität München

[87] Rank, E. ; Crouse, B. ; van Treeck, C. : Numerical Simulation of Air Flow for Civil Engineering Constructions on the basis of a product data model. In: *The Ninth International Conference on Computing in Civil and Building Engineering*. Taipei, Taiwan, 2002

[88] Rodi, W. : *Turbulence models and their application in hydraulics*. Brookfield : Balkema, 1993

[89] Roettger, S. ; Schulz, M. ; Bartelheimer, W. ; Ertl, T. : Automotive Soiling Simulation based on Massive Particle Tracing. In: *Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '01*, 2001, S. 309–317,363

[90] Roettger, S. ; Schulz, M. ; Bartelheimer, W. ; Ertl, T. : Flow Visualization on Hierarchical Cartesian Grids. In: *Lecture Notes in Computational Science and Engineering – Proceedings of 3rd International FORTWIHR Conference on HPSEC* Bd. 21, Springer Verlag, 2002, S. 139–146

[91] Romberg, R. ; Düster, A. ; Rank, E. : Solid modeling as a basis of co-operative planning in structural engineering. In: *eWork and eBusiness in Architecture, Engineering, Construction, proceedings of the ECPPM 2002*. Portoroz, Slovenia, 2002

[92] Rossignac, J. ; Borrel, P. : Multi-Resolution 3D Approximations for Rendering Complex Scenes. In: Falcidieno, B. (Hrsg.) ; Kunii, T. (Hrsg.): *Modeling in Computer Graphics*, Springer-Verlag, 1993, S. 455–465

[93] Schäfer, F. ; Breuer, M. : Integrated Particle Tracing within a Parallel Multiblock Flow Simulation Program: Validation and Application. In: Girod, B. (Hrsg.) ; Niemann, H. (Hrsg.) ; Seidel, H.-P. (Hrsg.): *Vision, Modelling and Visualization 1999*. Erlangen, 1999

[94] Schroeder, W. ; Martin, K. ; Lorensen, B. : *The Visualization Toolkit*. New Jerey : Prentice Hall PTR, 1996

[95] Schroeder, W. ; Zarge, J. ; Lorenson, W. : Decimation of Triangle Meshes. In: *Computer Graphics* 26 (1992), Nr. 2, S. 65–70

[96] Schulz, M. ; Krafczyk, M. ; Toelke, J. ; Rank, E. : Parallelization strategies and efficiency of CFD computations in complex geometries using Lattice Boltzmann methods on high-performance Computers. In: *High-Performance Scientific and Engineering Computing, Proceedings of the 3rd International FORTWIHR Conference on HPSEC*, 2001

[97] Shepard, D. : A two-dimensional interpolation function for irregularly-spaced data. In: *Proceedings of the 1968 23rd ACM national conference*, 1968, S. 517–524

[98] Sherman, W. R. ; Craig, A. B.: *Understanding Virtual Reality: Interface, Application, and Design*. Morgan Kaufmann Publishers, 2002

[99] Smagorinsky, J. : General Circulation Experiments with the Primitive Equations, I, The Basic Experiment. In: *Mon. Weather Review* 91 (1963), S. 99–165

[100] Sockel, H. : *Aerodynamik der Bauwerke*. Vieweg-Verlag, 1984

[101] Stalling, D. ; Hege, H.-C. : Fast and Resolution Independent Line Integral Convolution. In: *Computer Graphics* 29 (1995), S. 249–256. – Annual Conference Series

[102] Stanney, K. M.: *Handbook of Virtual Environments*. Lawrence Erlbaum Associates, 2002

[103] Succi, S. : *The Lattice Boltzmann equation for fluid dynamics and beyond*. Oxford : Clarendon Press, 2001

[104] Sutherland, I. : The Ultimate Display. In: *Information Processing 1965: Proceedings of IFIP Congress 65* Bd. 2. New York, 1965

[105] Teitzel, C. ; Grosso, R. ; Ertl, T. : Line Integral Convolution on Triangulated Surfaces. In: Thalmann, N. (Hrsg.) ; Skala, V. (Hrsg.): *WSCG '97 – The Fifth International Conference in Central Europe on Computer Graphics and Visualization* Bd. III University of West Bohemia, Pilzen, 1997, S. 572–581

[106] Thompson, J. F. ; Soni, B. ; Weatherill, N. : *Handbook of Grid Generation*. CRC Press, 1999

[107] Tölke, J. ; Krafczyk, M. ; Schulz, M. ; Rank, E. ; Berrios, R. : Implicit discretization and non-uniform mesh refinement approaches for FD discretizations of LBGK Models. In: *International Journal of Modern Physics C* 9 (1998), Nr. 8, S. 1143–1157

[108] Tölke, J. : *Die Lattice–Boltzmann Methode für Mehrphasenströmungen*, Lehrstuhl für Bauinformatik, Technische Universität München, Dissertation, 2001

[109] Tufte, E. R.: *Visual Statistical Thinking: Displays of Evidence for Making Decisions.* Graphics Press, 1997

[110] Tufte, E. R.: *The Visual Display of Quantitative Information.* Cheshire : Graphics Press, 1983

[111] Vetter, J. : Computational steering annotated Bibliography. 32 (1997), Nr. 6, S. 40–44

[112] Vince, J. : *Essential Virtual Reality Fast.* Springer-Verlag, 1998

[113] Wagner, E. : *Adaptive Gitterverfeinerung für eine Strömungssimulation in drei Dimensionen*, Institut für Informatik, TU München, Diplomarbeit, 1996

[114] Wengle, H. : *Numerische Berechnung turbulenter Strömungen*, Lehrstuhl für Fluidmechanik, Technische Universität München, Skript zur Vorlesung, 2001

[115] Wesche, G. : Three-dimensional Visualization of fluid dynamics on the Responsive Workbench. In: *Future generation computer systems* 15 (1999), S. 469–475

[116] Zabel, M. : *Erweiterung der Datenabbildung und Gittermodifikation einer interaktiven Strömungssimulations-Applikation*, Lehrstuhl für Bauinformatik, Technische Universität München, Diplomarbeit, 2002

[117] Zenger, C. : Sparse Grids. In: Hackbusch, W. (Hrsg.): *Parallel Algorithms for Partial Differential Equations: Proceedings of the* 6^{th} *GAMM-Seminar, Notes on Numerical Fluid Mechanics 32.* Braunschweig, 1991

Vorveröffentlichte Teilergebnisse

Die folgende Auflistung enthält die eigenen Publikationen, in denen bereits Teile dieser Arbeit vorveröffentlicht wurden. Die Erlaubnis zur Vorveröffentlichung wurde von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München erteilt.

1. **S. Kühner**, B. Crouse, M. Krafcyzk, J. Tölke, E. Rank: *From a building product model to the visualization of simulation data: Computation of indoor flow based on Lattice-Boltzmann-Methods*, Eingereicht in: Journal of Computer–Aided and Infrastructure Engineering, 2002

2. **S. Kühner**, M. Krafczyk, E. Rank: *Some Aspects of Virtual Reality based Steering of Indoor Air Flow Computation*, In: Roomvent 2002, Kopenhagen, 2002

3. B. Crouse, M. Krafczyk, **S. Kühner**, E. Rank, C. van Treeck: *Indoor air flow analysis based on lattice Boltzmann methods*, In: International Journal of Energy and Buildings 34(9), 2002

4. **S. Kühner**, M. Krafczyk, J. Tölke: *Towards interactive comfort optimization of Indoor Flows using Virtual Reality based analysis of Large-Eddy simulation results*, In: ICCCBE-IX, 9th International Conference on Computing in Civil and Building Engineering, Taipei, 2002.

5. **S. Kühner**, B. Crouse: *Visualisierung von Ergebnissen numerischer Simulationen in der Strömungsmechanik unter Verwendung von hierarchischen Datenstrukturen*, In: 13. Forum Bauinformatik, München 2001, Fortschritt-Berichte VDI, VDI-Verlag, Düsseldorf, 2001

6. **S. Kühner**, E. Rank, M. Krafczyk: *Efficient reduction of 3D simulation results based on spacetree data structures for data analysis in Virtual Reality environments*, In: Applied Virtual Reality in Engineering and Construction, Göteburg, 2001

7. **S. Kühner**, M. Krafczyk: *VirtualFluids - An environment for integral visualization and analysis of CAD and simulation data*, In: Vision, Modeling and Visualization 2000, Saarbrücken, 2000