HUMBOLDT-UNIVERSITÄT ZU BERLIN
INSTITUT FÜR INFORMATIK
VISUAL COMPUTING

Diplomarbeit

Modellierung von Reflexionseigenschaften verschiedener Stoffe für interaktives Echtzeit-Rendering

Bojko Heinrich

4. April 2011

Gutachter: Prof. Dr.-Ing. Peter Eisert, Prof. Dr. Wolfgang Coy
Betreuer: Prof. Dr.-Ing. Peter Eisert

Da die realistische Darstellung von Stoffen innerhalb von Echtzeitanwendungen auch in den letzten Jahren nicht an Relevanz verloren hat und immer weitere Anstrengungen unternommen werden, geeignete Reflexionsmodelle insbesondere für Echtzeitanwendungen aufzustellen, soll in dieser Arbeit die Schätzung eines parametrischen Beleuchtungsmodells anhand von BTF-Daten verschiedener Stoffe vorgestellt werden, um die Resultate anschließend mittels Normal Mapping möglichst realistisch und überzeugend rendern zu können. Hierzu werden die Ergebnisse in einer im Rahmen dieser Arbeit entwickelten 3D-Echtzeitumgebung präsentiert und die innerhalb dieser Umgebung implementierte interaktive Stoffsimulation vorgestellt, deren Fokus auf einer effizienten und authentischen Reproduktion physikalischer Kräfte liegt, die von innen und außen auf die Stoffe wirken können.

Inhaltsverzeichnis

1 Einleitung

1.1 Einführung

In der vorliegenden Arbeit werden die Reflexionseigenschaften verschiedener Stoffmaterialien analysiert, um ein Modell zu entwickeln, mit dessen Hilfe die charakteristischen Merkmale geeignet und effizient repräsentiert werden können. Zudem soll dadurch eine Möglichkeit eröffnet werden, die Stoffe möglichst authentisch in einer im Rahmen dieser Arbeit entwickelten 3D-Echtzeitumgebung darzustellen und mit ihnen zu interagieren. Hierzu werden in Kapitel 2 zunächst die 3D-Echtzeitumgebung und ihre für die Simulation und Beleuchtung der Stoffe relevanten Elemente vorgestellt, wobei sowohl Grundlagen als auch tiefergehende Mechanismen des 3D-Renderings betrachtet werden sollen. Im weiteren Verlauf soll dann in Kapitel 3 die interaktive Stoffsimulation innerhalb der 3D-Echtzeitumgebung mit dem Schwerpunkt der Modellierung von physikalischen Kräften besprochen werden, wobei hier insbesondere auf die Unterscheidung von internen und externen Einflüssen auf die Stoffsimulation eingegangen und einige visuelle Eindrücke der Stoffe in der 3D-Echtzeitumgebung präsentiert werden sollen.

Kapitel 4 beschäftigt sich schließlich mit den Möglichkeiten der Modellierung von Reflexionseigenschaften diverser Stoffe und dem zugrundeliegenden Reflexionsverhalten von Oberflächen. Es werden hierbei bisherige Ansätze der Repräsentation mit Hilfe von sogenannten Bidirektionalen Reflektanzverteilungsfunktionen und Bidirektionalen Texturfunktionen (vgl. Abschnitt 4.2) vorgestellt, wobei im Anschluss die Ableitung eines Beleuchtungsmodells und die Schätzung verschiedener – für die Darstellung der Stoffmaterialien wichtiger – Parameter im Vordergrund steht. Das Optimierungsverfahren zur Gewinnung dieser Parameter – mit dem Ziel einer möglichst authentischen Reproduktion der Stoffe – bildet hierbei den Kern der Arbeit, weswegen neben einer ausführlichen Beschreibung des Verfahrens und der mathematischen Zusammenhänge auch Verbesserungsvorschläge im Hinblick auf die Minimierung der quadratischen Abweichung sowie weitere Resultate der Stoffdarstellung innerhalb der 3D-Echtzeitumgebung gezeigt werden.

1.2 Mathematische Notation

In dieser Arbeit wird für mathematische Ausdrücke folgende Notation verwendet:

Typ	**Notation**	**Beispiel**
Skalar	kursiver Kleinbuchstabe	r, s_i, t_{ij}
Vektor	fetter Kleinbuchstabe	$\mathbf{x}$, $\mathbf{y_i}$, $\mathbf{z_{ij}}$, $\mathbf{v^T}$
Punkt	kursiver Großbuchstabe	P, Q
Matrix	fetter Großbuchstabe	$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$
Winkel	griechischer Kleinbuchstabe	α, β, γ
Abstand zwischen Punkten bzw. Länge eines Vektors	$\|\cdot\|_2$	$\|\mathbf{x}\|_2$, $\|\mathbf{x_j} - \mathbf{x_i}\|_2$
Komponentenweise Multiplikation von Vektoren	$\otimes$	$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 \cdot y_1 \\ x_2 \cdot y_2 \\ x_3 \cdot y_3 \end{pmatrix}$

2 Eine interaktive 3D-Echtzeitumgebung

2.1 Einführung

In diesem Kapitel wird auf die grundlegenden Bestandteile der im Rahmen dieser Diplomarbeit entwickelten 3D-Echtzeitumgebung eingegangen, die für die Stoffsimulation relevant sind, wobei die allgemeine Funktionsweise der verwendeten Konstrukte und Mechanismen im Vordergrund stehen soll. Die 3D-Echtzeitumgebung wurde in C++ unter Verwendung von DirectX 9 entwickelt und soll an dieser Stelle lediglich in Auszügen referenziert werden, um den Fokus auf den enthaltenen Konzepten zu belassen und weniger auf Implementierungsdetails in Form von Quelltextzitaten. Neben dem Aspekt der Echtzeit wurde bei der Entwicklung besonderes Augenmerk auf eine ansprechende visuelle Ästhetik und Glaubwürdigkeit gerichtet, um eine überzeugende interaktive Stoffsimulation zu schaffen.

Einige Elemente der 3D-Echtzeitumgebung, die nicht in direktem Zusammenhang mit der Stoffsimulation stehen, werden im Folgenden exemplarisch genannt, um ihr Vorhandensein auf in der Arbeit verwendeten Screenshots zu rechtfertigen. Dazu zählt beispielsweise die implementierte Vegetation, die durch Bäume, Büsche und Gras vertreten wird und bis auf die Möglichkeit des Schattenwurfs keinen direkten Einfluss auf die Stoffsimulation nimmt. Des Weiteren gibt es eine Vielzahl von Partikelsystemen – wie z. B. Wolken – zu denen die Stoffsimulation selbst auch dazuzuzählen ist, was in Kapitel 3 eingehend beschrieben wird. Das implementierte Terrain, auf dem sich die Kamera frei bewegen kann – wobei die Höheninformationen aus einer Textur extrahiert werden und zur Höhenanpassung der Kamera genutzt werden können – spielt hierbei eine Sonderrolle, da es zur Kollisionsbehandlung direkten Einfluss auf die Stoffsimulation nimmt, worauf in Abschnitt 3.2.2 näher eingegangen wird.

Diejenigen Elemente hingegen, die vor allem die äußere Erscheinung beeinflussen, sollen in diesem Kapitel untersucht werden. So wurde beispielsweise ein dynamischer Tag-Nacht-Zyklus implementiert, bei dem sich Sonne und Mond als direktionale Lichtquellen (vgl. Abschnitt

2.4) abwechseln und das Terrain, die Vegetation und alle gerenderten Modelle – einschließlich des Stofftuchs – abhängig von Lichtfarbe und -intensität entsprechend beleuchten. Außerdem wird in Abschnitt 2.6 eine Technik namens Normal Mapping vorgestellt, die insbesondere in Kapitel 4 eine tragende Rolle spielt und mit der es möglich ist, feine Unebenheiten und Oberflächendetails darzustellen. Abschließend wird in Abschnitt 2.7 die Modellierung dynamischer, weicher Schatten mittels Shadow Mapping besprochen und zudem ein kurzer Vergleich zum konkurrierenden Verfahren namens Shadow Volumes gezogen, das sich für die Stoffsimulation als nicht geeignet erwiesen hat.

2.2 HLSL und Shader

Wie bereits erwähnt, wird in der vorliegenden Arbeit auf DirectX 9[1], eine Sammlung von Programmierschnittstellen, die die Funktionalität der Grafikhardware abbildet und zur Erstellung von Computerspielen und Multimedia-Anwendungen genutzt werden kann, zurückgegriffen. DirectX beinhaltet verschiedene Module, die auf 2D (DirectDraw), 3D (Direct3D), Audio (DirectSound) und Nutzereingaben (DirectInput) spezialisiert sind, wobei in diesem Kapitel vor allem die Funktionalitäten der Direct3D-API (engl.: Application Programming Interface) von Interesse sind [Microsoft 09]. Eine Besonderheit ab Version 9 ist die Einführung von HLSL (High Level Shading Language), eine Weiterentwicklung der ab Version 8 etablierten Sprache für die Programmierung moderner 3D-Grafikhardware. HLSL löst den Vorgänger, der dem Programmierer im Stil einer Assemblersprache eine sehr hardwarenahe Programmierung abverlangte, zugunsten der Abstraktionsmöglichkeiten einer Hochsprache wie beispielsweise C ab, und bietet dem Programmierer eine wesentlich einfachere und intuitivere Möglichkeit, die eigenen Anwendungen durch sogenannte Shader zu erweitern, deren Funktion im Folgenden näher erläutert wird [Peeper 04].

Die darzustellenden 3D-Modelle, die im Allgemeinen aus Polygonen bzw. Dreiecken zusammengesetzt sind, durchlaufen bis zur Anzeige auf dem Bildschirm verschiedene Stufen der Transformation und Beleuchtung. Innerhalb dieser Stufen kann dank der flexiblen Rendering-Pipeline[2] von DirectX 9 vom Programmierer mittels Shadern auf die Daten Einfluss genommen

[1] Es wird das DirectX SDK (Software Development Kit) vom August 2009 verwendet.

[2] Eine Rendering-Pipeline ist ein Modell in der Computergrafik, das die Arbeitsschritte zur Darstellung einer 3D-Szene beschreibt, wobei die Schritte selbst in Hardware implementiert wurden. Die Direct3D-API abstrahiert hierbei die zugrundeliegende Hardware und lässt den Programmierer auf die Schritte zugreifen [Microsoft 09].

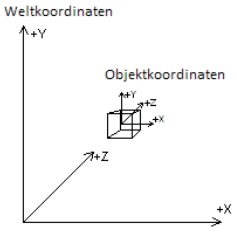

Abbildung 2.1: Relation zwischen lokalen Objekt- und globalen Weltkoordinaten [Microsoft 09].

werden, was in früheren Versionen von DirectX nicht möglich war, da die dort abgebildete Grafikhardware dies zu diesem Zeitpunkt noch nicht zuließ. Shader sind hierbei kleine – aber beliebig komplexe[3] – Programme in HLSL, die in Vertex Shader und Pixel Shader aufgeteilt werden. Um ein besseres Verständnis für den Ablauf des Rendering-Prozesses zu erhalten und um zu verdeutlichen, welche elementare Rolle Vertex und Pixel Shader währenddessen spielen, erfolgt nun ein kurzer Einblick in diejenigen Schritte des Rendering-Prozesses, auf die an späteren Stellen in dieser Arbeit Bezug genommen wird.

2.3 Transformation der Vertices

Zu Beginn liegen die Koordinaten der 3D-Modelle, wie z. B. des zu rendernden Stofftuchs, in ihrem eigenen lokalen Koordinatensystem vor, wobei hier auch oft von Modell- oder Objektkoordinaten die Rede ist. Um festzulegen, wie das 3D-Objekt in der gesamten Szene positioniert und orientiert ist, findet zunächst eine Transformation der Eckpunkte bzw. Vertices (engl.: Vertex, Plural: Vertices) der Dreiecke statt, welche die Objektkoordinaten in das sogenannte Weltkoordinatensystem überführt, in dem alle 3D-Objekte genau denselben – und nicht mehr ihren eigenen – Koordinatenursprung besitzen (vgl. Abbildung 2.1). Bei der Transformation ins Weltkoordinatensystem kann ein Objekt wahlweise skaliert, also vergrößert oder verkleinert, um eine oder mehrere Achsen rotiert und auf den Achsen beliebig verschoben werden. Diese drei Operationen (Skalierung, Rotation und Translation) werden in Form von Matrixmultiplikationen auf die Vertices, die in Form von Vektoren vorliegen, in einem Vertex

[3] In Abhängigkeit von der verwendeten Grafikhardware werden verschiedene Shader-Modelle unterstützt, die wiederum den Befehlssatz, die Syntax, Komplexität und Länge der Shaderprogramme beschränken [Peeper 04].

Shader angewandt [Peeper 04]. Die Transformation der Vertices des Stofftuchs ins Weltkoordinatensystem spielt später für die Kollisionserkennung mit anderen Objekten (vgl. Abschnitt 3.2.2) eine wichtige Rolle, weswegen an dieser Stelle die mathematischen Grundlagen näher ausgeführt werden sollen.

Sei $\mathbf{v} = \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T \in \mathbb{R}^4$ die Position eines Vertex in Objektkoordinaten, seien $s_x \in \mathbb{R}$, $s_y \in \mathbb{R}$ und $s_z \in \mathbb{R}$ die Skalierungsfaktoren für die drei Achsen und $t_x \in \mathbb{R}$, $t_y \in \mathbb{R}$ und $t_z \in \mathbb{R}$ die Verschiebungen auf den Achsen, dann kann mittels

$$\mathbf{v}' = \mathbf{v} \cdot \mathbf{M} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix} = \begin{pmatrix} x \cdot s_x + t_x \\ y \cdot s_y + t_y \\ z \cdot s_z + t_z \\ 1 \end{pmatrix} \tag{2.1}$$

die neue Position $\mathbf{v}' \in \mathbb{R}^4$ in Weltkoordinaten berechnet werden, wobei das Objekt skaliert und verschoben wird und $\mathbf{M} \in \mathbb{R}^{4\times4}$ die entsprechende Matrix darstellt, die diese Operationen ermöglicht. Die Einführung einer vierten Komponente w – für die sonst dreidimensionale Position $\mathbf{v} = \begin{pmatrix} x & y & z & w \end{pmatrix}^T$ eines Vertex – ist hierbei ein Hilfsmittel zur Unterscheidung zwischen Vektoren und Punkten [Kwoon 04]. Setzt man $w = 1$ kann der vorliegende Punkt auch auf den Achsen verschoben werden (vgl. Gleichung 2.1), was im Fall $w = 0$ unterbunden wird und immer dann sinnvoll ist, wenn beispielsweise ein dreidimensionaler Richtungsvektor transformiert werden soll. Der Grund hierfür ist, dass für $w = 0$ die vierte Zeile der Matrix $\mathbf{M}$ ignoriert wird und die dort eingetragenen Verschiebungswerte dadurch nicht ins Gewicht fallen. Koordinaten, die mit einer vierten Komponente ausgestattet sind, werden auch als homogene Koordinaten bezeichnet, sollen an dieser Stelle aber nicht näher beleuchtet werden[4].

Möchte man das Objekt rotieren, kann aus drei verschiedenen, kombinierbaren Matrizen gewählt werden, von denen jede eine Rotation auf der X-, Y- oder Z-Achse widerspiegelt. Als Beispiel soll es an dieser Stelle genügen, die Rotation um einen beliebig, aber festen Winkel θ auf der X-Achse zu zeigen, was mittels

[4] Weitere Implikationen, die durch die vierte Komponente bedingt sind, können in [Kwoon 04] nachgelesen werden und sind u. a. auch für Schattenberechnungen mittels Shadow Volumes von Interesse.

$$\mathbf{v}' = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \\ y \cdot \cos\theta - y \cdot \sin\theta \\ z \cdot \sin\theta + z \cdot \cos\theta \\ 1 \end{pmatrix} \tag{2.2}$$

bewerkstelligt werden kann. Der Vorteil bei der Verwendung von Matrizen ist die Möglichkeit einer Verkettung mehrerer Matrizen, was einer Hintereinanderausführung ihrer entsprechenden Effekte gleichkommt. Möchte man also beispielsweise ein Objekt rotieren und anschließend auf den Achsen verschieben, können die beiden entsprechenden Matrizen einmalig multipliziert und die neue resultierende Matrix anschließend mit allen Vertices des Objekts multipliziert werden, wodurch eine zweimalige Multiplikation jedes Vertex mit den beiden ursprünglichen Matrizen vermieden und Rechenzeit gespart werden kann. Da Matrix-Multiplikationen aber nicht kommutativ sind, also für zwei Matrizen $\mathbf{M_1} \in \mathbb{R}^{n \times n}$ und $\mathbf{M_2} \in \mathbb{R}^{n \times n}$ gilt, dass $\mathbf{M_1} \cdot \mathbf{M_2} \neq \mathbf{M_2} \cdot \mathbf{M_1}$, ist bei der Verkettung unbedingt auf die Reihenfolge der Matrizen zu achten. Betrachtet man das Beispiel einer Frisbee-Scheibe, die zuerst um den eigenen lokalen Objektursprung – die Objektmitte – rotiert und anschließend in der Welt mittels Translation positioniert werden soll, wird deutlich, dass bei einer Vertauschung der beiden Operationen, die Frisbee-Scheibe zuerst in der Welt platziert und anschließend um den Ursprung des Weltkoordinatensystems rotiert werden würde [Microsoft 09].

Eine Transformation in die entgegengesetzte Richtung – d. h. aus dem Koordinatensystem der Welt zurück in das der Objektkoordinaten – ist mit Hilfe der inversen Weltmatrix $\mathbf{M^{-1}}$ möglich und findet später in Abschnitt 3.2.2 Verwendung, wenn es darum geht, erkannte Kollisionen zwischen Objekten in der Welt im lokalen Koordinatensystem des Stofftuchs zu behandeln. Dabei kann die ursprüngliche Position mittels $\mathbf{v} = \mathbf{v}' \cdot \mathbf{M^{-1}}$ berechnet werden. Die weiteren Transformationsschritte ins Koordinatensystem der Kamera und die anschließende Projektion, das Clipping sowie die abschließende Transformation in das zweidimensionale Koordinatensystem der durch den Programmierer festgelegten Bildschirmauflösung können in detaillierter Form in [Möller 08] nachgelesen werden und sind auf Grund ihres Umfangs und fehlenden Relevanz nicht Gegenstand dieser Arbeit.

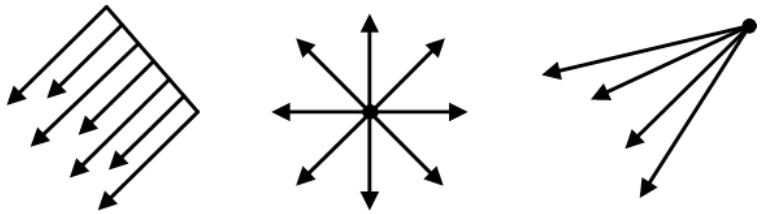

Abbildung 2.2: Lichtquellen: Direktional, Punktlicht und Spotlight.

2.4 Lichtquellen

Ob man Licht als Teilchen oder elektromagnetische Welle behandelt, spielt für das Rendering in der Computergrafik eine untergeordnete Rolle und hängt vor allem vom verwendeten Beleuchtungsmodell ab. Im Allgemeinen unterscheidet man lokale und globale Modelle, wobei erstere nur bestimmte Objekte in die Berechnungen einbeziehen und letztere die gesamte 3D-Szene mit einem einzigen Modell beleuchten, Wechselwirkungen zwischen den Objekten simulieren und die Gesetze der Strahlenoptik sowie der Energieerhaltung berücksichtigen. Da globale Modelle sehr rechenintensiv sind, wird in der vorliegenden Arbeit im Hinblick auf die Wahrung des Echtzeitaspekts ausschließlich von lokalen Beleuchtungsmodellen Gebrauch gemacht. Hier ist vor allem die Vorstellung einer elektromagnetischen Strahlungsenergie von Bedeutung, bei der Licht – je nach Art der Lichtquelle – eine bestimmte Position oder Richtung und Intensität aufweist. Man unterscheidet grundsätzlich drei Formen von künstlichen Lichtquellen (vgl. Abbildung 2.2), die realen Lichtquellen nachempfunden sind und diese möglichst authentisch wiedergeben sollen: Direktionale Lichtquellen bzw. Richtungslichter, Punktlichter und Scheinwerfer bzw. Spotlights [Möller 08].

Direktionale Lichtquellen

Direktionale Lichtquellen besitzen keine Position, da hier eine unendlich weit entfernete Lichtquelle wie die Sonne (vgl. Abbildung 2.3) modelliert wird, bei der die einfallenden Lichtstrahlen parallel zueinander sind und somit für die gesamte Szene derselbe Richtungsvektor verwendet werden kann. Dieser Richtungsvektor liegt in Weltkoordinaten vor und sei als $\mathbf{l} \in \mathbb{R}^3$ definiert. Die Lichtintensität wird – wie bei den anderen beiden Lichttypen – als RGB-Farbwert definiert, so dass Rot-, Grün- und Blauwerte separat festgelegt werden können, wodurch nicht nur die Helligkeit, sondern auch der Farbton in die Berechnungen einfließen kann,

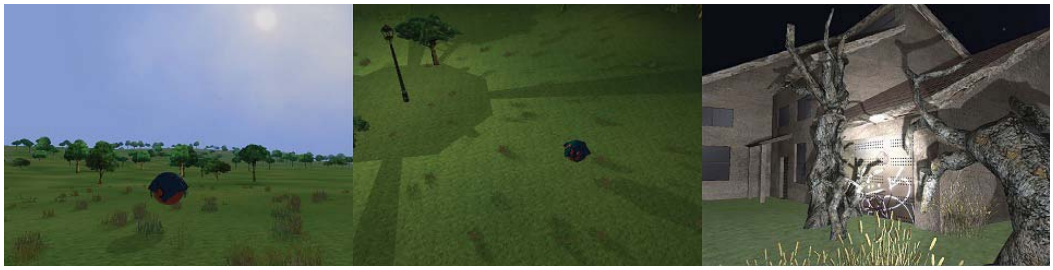

Abbildung 2.3: Die drei Lichttypen in der 3D-Echtzeitumgebung: Direktional in Form von Sonnenlicht, ein Punktlicht in Form einer Laterne und ein Spotlight in Form einer Taschenlampe.

der in der Realität über die Wellenlänge des ausgesendeten Lichts und die damit verbundenen Absorptions- und Reflexionsvorgänge auf den Materialien bestimmt wird [Möller 08].

Punktlichtquellen

Punktlichter besitzen eine Position $L_{Pos} \in \mathbb{R}^3$, die ebenfalls in Weltkoordinaten vorliegt, und die Eigenschaft, dass von ihnen ausgetrahltes Licht bei zunehmender Entfernung zur Lichtquelle an Intensität verliert, was als Attenuation oder Lichtschwächung bezeichnet wird. Die Abnahme der Intensität kann durch einen konstanten Attenuationswert $a_{konstant} \in \mathbb{R}$, einen linearen Attenuationswert $a_{linear} \in \mathbb{R}$ und einen quadratischen Attenuationswert $a_{quadratisch} \in \mathbb{R}$ beschrieben werden, wobei mittels

$$a = \frac{1}{a_{konstant} + a_{linear} \cdot d + a_{quadratisch} \cdot d^2} \tag{2.3}$$

die Attenuation $a \in \mathbb{R}$ berechnet werden kann, die proportional zum Quadrat der Entfernung $d = \|L_{Pos} - \mathbf{p}\|_2$ zwischen der aktuellen Vertexposition $\mathbf{p} \in \mathbb{R}^3$ und der Position der Lichtquelle abnimmt. Alternativ kann nach Festlegung eines Radius $r \in \mathbb{R}$ eine Kugel definiert werden, in der sich das Licht von der Quelle ausbreitet und durch

$$a = 1 - \frac{d^2}{r^2} \tag{2.4}$$

die Abnahme der Intensität berechnet werden, wobei $r > 0$ vorausgesetzt wird [Microsoft 09]. In der Mitte von Abbildung 2.3 ist die Umsetzung einer Punktlichtquelle als Laterne zu sehen,

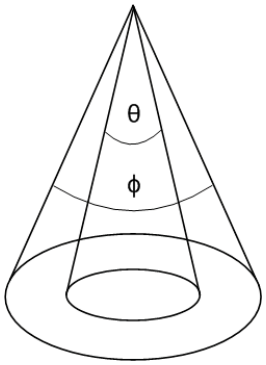

Abbildung 2.4: Ein Spotlight mit innerem und aüßerem Kegel, definiert über zwei Winkel.

wobei die Abschwächung der Lichtstärke bei zunehmender Entfernung zur Quelle besonders auf dem Terrain gut zu erkennen ist und gemäß Gleichung 2.4 berechnet wurde.

Spotlights

Im Gegensatz zu Punktlichtquellen, die Licht in alle Richtungen gleichermaßen abstrahlen, wird bei Spotlights von der Position L_{Pos} ausgehend in Richtung $\mathbf{l}$ ein Lichtkegel definiert, wobei nicht nur eine Attenuation bezüglich der Entfernung zur Lichtquelle stattfindet, sondern zusätzlich eine Abschwächung der Intensität zu den Rändern des Kegels hin zu verzeichnen ist. Dabei existiert ein hellerer, innerer und ein lichtschwächerer, äußerer Kegel, welche über die zwei Winkel θ und ϕ festgelegt werden (vgl. Abbildung 2.4). Zudem besteht bei diesem Typ von Lichtquelle die Möglichkeit, innerhalb des Kegels eine Textur – beispielsweise eines Taschenlampenglases – auf die zu beleuchtenden Flächen zu projizieren, so dass die Form des Lichtkegels moduliert und bestimmte Muster oder Verunreinigungen des Glases simuliert werden können, was in Abbildung 2.3 im rechten Screenshot umgesetzt wurde.

2.5 Beleuchtungsmodell

Die Ausgabe des Vertex Shaders wird direkt als Eingabe für den Pixel Shader verwendet, so dass anhand eines zugrundeliegenden Beleuchtungsmodells die Vertices in einem Vertex Shader oder später als Pixel (Abk.: Picture Element) in einem Pixel Shader eingefärbt werden können. Da die Anzahl und Verteilung der Vertices innerhalb eines 3D-Objekts diskret ist

und vom Grad der Tesselierung[5] der Dreiecke abhängt, werden die Punkte im endgültigen zweidimensionalen Bild der perspektivisch betrachteten 3D-Szene, die zwischen den Vertices liegen, vor der Übergabe an den Pixel Shader in ihrer Position, Farbe, ihren Normalenvektoren und Texturkoordinaten interpoliert [Möller 08]. Je gröber aufgelöst das 3D-Modell ist, desto ungenauer und unrealistischer wirkt demnach die Beleuchtung mit Hilfe eines Vertex Shaders, weswegen in der vorliegenden Arbeit ausschließlich Beleuchtungsberechnungen pro Pixel – also in einem Pixel Shader – vorgenommen wurden. Die verwendete Rendering-Gleichung, welche die finale Pixelfarbe bestimmt, setzt sich aus verschiedenen Komponenten zusammen, die jeweils unterschiedlichen Eigenschaften realen Lichts Rechnung tragen.

2.5.1 Ambienter Anteil

Selbst bei nur einer Lichtquelle werden in der Realität Lichtstrahlen mehrfach von den Oberflächen der Materialien reflektiert, wodurch auch dunklere Bereiche, die nicht auf direktem Weg durch einfallende Lichtstrahlen erreicht werden, erhellt werden. Das hat zur Folge, dass innerhalb der Umgebung indirektes Licht existiert, das nur eine Farbe und Intensität – jedoch keine Quelle – besitzt, und aus der Umgebung selbst stammt, wo es unzählige Male zerstreut und reflektiert wurde [Garstenauer 06]. Dieses auf alle Objekte gleichförmig einfallende Licht fließt als ambiente Beleuchtung in die finale Pixelfarbe ein und sei als $I_A \in \mathbb{R}^3$ definiert.

2.5.2 Diffuser Anteil

Um den diffusen Anteil $I_D \in \mathbb{R}^3$ zu berechnen, wird die Lichtintensität für jeden Pixel anhand des Kosinus des Winkels θ_i zwischen der Oberflächennormalen $\mathbf{n} \in \mathbb{R}^3$ und dem Richtungsvektor des Lichts $\mathbf{l}$ ermittelt (vgl. linke Seite in Abbildung 2.5), was einem Punktprodukt zwischen den beiden Vektoren multipliziert mit der Lichtfarbe $L_{Diffuse} \in \mathbb{R}^3$ entspricht und mittels

$$I_D = \max(\mathbf{n} \cdot \mathbf{l},\, 0) \cdot L_{Diffuse} \tag{2.5}$$

innerhalb eines Pixel Shaders umgesetzt wird. Der diffuse Anteil korrespondiert hierbei mit der noch übrigen Menge an Licht, die an diesem Punkt der Oberfläche nach Absorption, Reflexion und Refraktion – also Lichtbrechung – noch vorhanden ist und gleichförmig in alle

[5] Technik, bei der grob aufgelöste Oberflächen eines 3D-Modells, die in Form von Dreiecken vorliegen, in weitere Dreiecke unterteilt werden, wodurch der Detailgrad erhöht und kantige Konturen runder gestaltet werden können, was insbesondere bei konvexen Oberflächen auffällt [Microsoft 09].

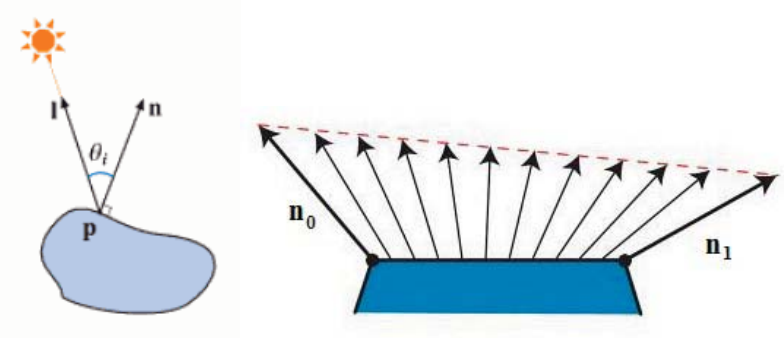

Abbildung 2.5: Links: Der Kosinus des Winkels zwischen **n** und **l** bestimmt den diffusen Anteil bei der Einfärbung des jeweiligen Pixels [Möller 08].
Rechts: Die Interpolation von Normalen vor der Übergabe an den Pixel Shader führt dazu, dass interpolierte Vektoren u. U. nicht mehr Länge 1 haben [Möller 08].

Richtungen abgegeben wird, wobei die Position des Betrachters, wie schon beim ambienten Anteil, keine Rolle spielt. Da ein negativer Kosinuswert implizieren würde, dass Licht von unterhalb der Oberfläche kommt, werden lediglich Werte zugelassen, die größer oder gleich 0 sind. Der diffuse Beleuchtungsanteil beruht auf dem Lambertschen Gesetz, das besagt, dass für ideale diffuse Oberflächen die ausgehende Strahlung proportional zum Kosinus des Winkels θ_i ist [Garstenauer 06, Möller 08].

Wie bereits erwähnt werden sämtliche Ausgabewerte aus einem Vertex Shader bei der Übergabe an den Pixel Shader linear interpoliert, wodurch die Normalen, die lediglich pro Vertex vorliegen, unter Umständen nicht mehr die Länge 1 besitzen und erneut normalisiert werden müssen, was auf der rechten Seite in Abbildung 2.5 dargestellt ist. Wäre das nicht der Fall, würde das Punktprodukt nicht mehr dem Kosinus des Winkels zwischen den beiden Vektoren **n** und **l** entsprechen – die Gleichung

$$\mathbf{n} \cdot \mathbf{l} = \|\mathbf{n}\|_2 \cdot \|\mathbf{l}\|_2 \cdot \cos\theta_i \tag{2.6}$$

würde also verletzt werden. Der Richtungsvektor des Lichts **l** muss also auch normalisiert werden, was einmalig zu Beginn jedes Frames für alle Pixel geschehen kann. Zudem ist es sinnvoll, den Richtungsvektor **l** in umgekehrter Richtung des Lichts zu definieren (vgl. linke Seite in Abbildung 2.5), damit beim Punktprodukt nicht bei jeder Berechnung der Richtungsvektor

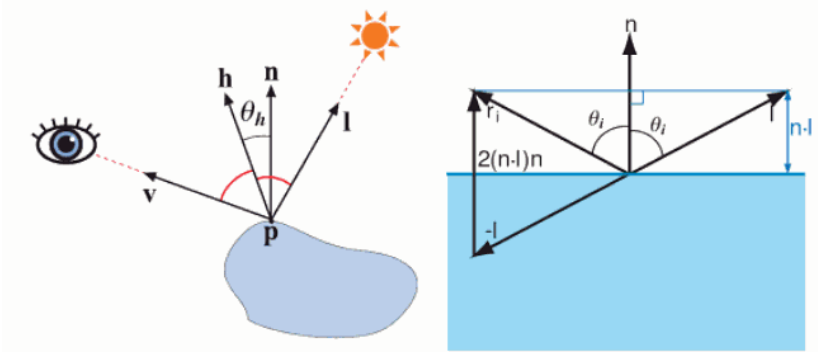

Abbildung 2.6: Links: Der Halbvektor **h** halbiert den Winkel (in rot) zwischen Vektor **v** und Lichtrichtungsvektor **l** [Möller 08].
Rechts: Der Lichtrichtungsvektor **l** mit Einfallswinkel θ_i wird an der Oberflächennormalen **n** reflektiert, so dass der Reflexionsvektor $\mathbf{r_i}$ entsteht [Möller 08].

negiert werden muss [Möller 08].

2.5.3 Specular-Anteil

Der spiegelnde Anteil – in Anlehnung an englische Literatur auch oft im deutschen Sprachgebrauch als Specular-Anteil bezeichnet – repräsentiert insbesondere für glatte und glänzende Oberflächen den Teil des Lichts, der in Abhängigkeit von der Betrachterposition, der Oberflächennormalen und dem Lichteinfallsvektor wie bei einem perfekten Spiegel reflektiert wird. Hierzu wird der sogenannte View-Vektor $\mathbf{v} \in \mathbb{R}^3$ eingeführt, der ausgehend von der aktuellen Vertexposition **p** in Richtung der Betrachter- bzw. Kameraposition $C_{Pos} \in \mathbb{R}^3$ zeigt und als $\mathbf{v} = C_{Pos} - \mathbf{p}$ definiert sei. Nach dem Modell von [Phong 75] ist die Intensität des reflektierten Lichts proportional zu $\cos^n \beta$, wobei β der Winkel zwischen dem View-Vektor **v** und der idealen Reflexionsrichtung

$$\mathbf{r_i} = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l} \tag{2.7}$$

des einfallenden Lichtstrahls (vgl. rechte Seite in Abbildung 2.6) ist und der Exponent n die Rauhigkeit der Oberfläche charakterisiert. Im Fall $n = \infty$ handelt es sich bei der Oberfläche um einen perfekten Spiegel, der das gesamte Licht in Richtung des Reflexionsvektors $\mathbf{r_i}$ reflektiert [Garstenauer 06].

Da die Berechnung des Reflexionsvektors $\mathbf{r_i}$ zur damaligen Zeit noch zu rechenintensiv war, formulierte [Blinn 77] ein äquivalentes Beleuchtungsmodell, das auch heute noch in Echtzeitanwendungen Verwendung findet und als Blinn-Phong-Modell bekannt ist. Der Specular-Anteil $I_S \in \mathbb{R}^3$ berechnet sich dann mittels

$$I_S = \max(\mathbf{n} \cdot \mathbf{h}, 0)^n \cdot L_{Specular} \tag{2.8}$$

wobei der Halbvektor $\mathbf{h} \in \mathbb{R}^3$ den Winkel zwischen $\mathbf{v}$ und $\mathbf{l}$ halbiert (vgl. linke Seite in Abbildung 2.6) und über die Berechnungsvorschrift

$$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{\|\mathbf{l} + \mathbf{v}\|_2} \tag{2.9}$$

in normalisierter Form ermittelt werden kann. Das Punktprodukt korrespondiert in diesem Fall mit dem Kosinus des Winkels θ_h zwischen der Oberflächennormalen und dem Halbvektor, weswegen der Specular-Anteil im Gegensatz zum diffusen und ambienten Anteil abhängig vom Betrachtungswinkel der Kamera ist. Der Farbton des reflektierten Lichtanteils fließt in Form von $L_{Specular} \in \mathbb{R}^3$ in die Gleichung ein und ist im Allgemeinen im Hinblick auf das Verhältnis von Rot-, Grün- und Blauwerten mit der diffusen Lichtfarbe $L_{Diffuse}$ identisch, obgleich Unterschiede in der Intensität sinnvoll sind, um die Stärke des Specular Highlights (engl.: Glanzpunkt) zu regulieren [Garstenauer 06, Möller 08].

2.5.4 Finale Pixelfarbe

Die schon angesprochene Ungenauigkeit bei Beleuchtungsberechnungen in einem Vertex Shader im Hinblick auf die lineare Interpolation sämtlicher Parameter und insbesondere die Nicht-Linearität des Specular-Anteils I_S (vgl. Gleichung 2.8) verlangen eine Beleuchtungsberechnung in einem Pixel Shader, damit die nun pro Pixel vorliegenden Normalen für die Berechnung der Specular Highlights herangezogen und so visuelle Artefakte vermieden werden können. Die finale Pixelfarbe $P_{Color} \in \mathbb{R}^3$ setzt sich dann wiefolgt zusammen

$$P_{Color} = (I_A \cdot K_A + I_D \cdot K_D \cdot a) \otimes T_{Color} + I_S \cdot K_S \cdot a \tag{2.10}$$

wobei die Berechnungen pro Pixel ausgeführt werden und die ambienten und diffusen Anteile zusätzlich mit der Stofftextur multipliziert werden, deren für jeden Pixel individuell gesam-

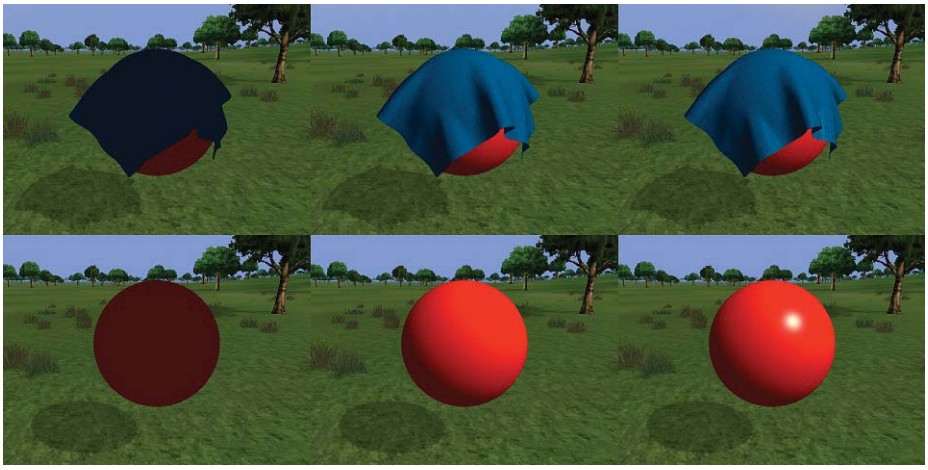

Abbildung 2.7: Sukzessive Hinzunahme von ambienten, diffusen und Specular-Beleuchtungsanteilen (v. l. n. r.) für Kugel und Stofftuch, wobei für die beiden Objekte der Schatten deaktiviert wurde: Nur ambiente Anteile. Ambiente und diffuse Anteile. Ambiente, diffuse und Specular-Anteile.

pelter RGB-Farbwert in Form von $T_{Color} \in \mathbb{R}^3$ in die Pixelfarbe miteinfließt. Die Reflexionskoeffizienten $K_A \in \mathbb{R}$, $K_D \in \mathbb{R}$ und $K_S \in \mathbb{R}$ für die ambienten, diffusen und Specular-Anteile tragen hierbei dem empirisch ermittelten Reflexionsverhalten der darzustellenden Materialien im jeweiligen Anteil Rechnung. Darüber hinaus beschreibt $0 \leq a \leq 1$ für Punktlichter die abstandsabhängige Attenuation der Lichtintensität und wird für direktionale Lichtquellen wie die Sonne gleich 1 gesetzt [Garstenauer 06, Möller 08].

In Abbildung 2.7 wurden die Kugel und das Stofftuch sukzessiv mit den soeben besprochenen Beleuchtungsanteilen gerendert, wobei jeweils beim rechten Bild die in Gleichung 2.10 berechnete Beleuchtung zustande kommt. Für die beiden Screenshots in Abbildung 2.8 kommt zusätzlich eine Technik namens Environment Mapping[6] zum Einsatz, die insbesondere die Kugel optisch aufwertet. Die Echtzeitreflexionen der Umgebung werden hierbei im Screenshot auf der rechten Seite unter Zuhilfenahme des Fresnel-Effekts berechnet, wodurch eine vom Blickwinkel des Beobachters abhängige Reflexion der 3D-Szene auf der Kugel ermöglicht wird. Der

[6] (Kubisches) Environment Mapping ist eine Technik, bei der die Umgebung – z. B. wie im vorliegenden Beispiel aus der Position der Kugel – zuerst in sechs Texturen, die wie die Seiten eines Würfels angeordnet sind, gerendert und anschließend als Echtzeitreflexion der Umgebung auf die Kugel übertragen wird [Microsoft 09, Möller 08].

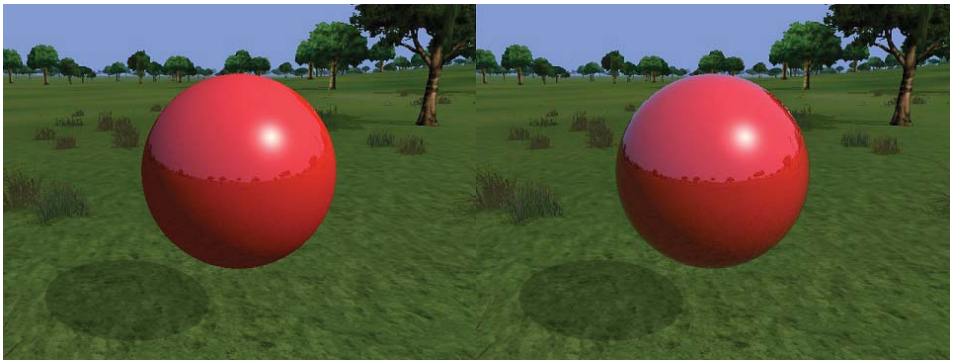

Abbildung 2.8: Links: Environment Mapping ohne Fresnel-Effekt. Rechts: Environment Mapping mit Fresnel-Effekt.

Fresnel-Effekt beschreibt die Interaktion von Lichtstrahlen mit der beleuchteten Oberfläche – die meist den Übergang zwischen zwei verschiedenen Stoffen darstellt – in Abhängigkeit von der Oberflächennormalen **n** und dem View-Vektor **v** und lässt sich am Beispiel einer Wasseroberfläche anschaulich beschreiben, wobei das Verhältnis zwischen gebrochenem Licht von unterhalb der Wasseroberfläche und reflektiertem Licht von oberhalb der Wasseroberfläche berechnet werden kann. Je flacher der Blickwinkel zur beleuchteten Oberfläche, d. h. je größer das Punktprodukt zwischen **n** und **v**, desto stärker ist die Reflexion der Umgebung an diesem Punkt. Schaut man allerdings senkrecht auf die Wasseroberfläche herab – ist das Punktprodukt also sehr klein – wird nur sehr wenig von der Umgebung reflektiert und das Wasser ist umso durchsichtiger [Möller 08]. Im Fall der Kugel lässt sich der Effekt im rechten Bild von Abbildung 2.8 gerade an den Rändern – wo der Blickwinkel sehr flach ist – gut beobachten, wo zunehmend der blaue Himmel bzw. das grünfarbene Terrain reflektiert wird.

2.6 Normal Mapping

Normal Mapping ist eine Technik in der Computergrafik, bei der Oberflächendetails wie beispielsweise feine Risse, Falten, Löcher oder Fugen in verschiedenen Materialien simuliert werden können, ohne die zugrundeliegende Geometrie des 3D-Objekts zu verändern. Dabei wird eine zweidimensionale Textur – die Normal Map – zu Hilfe genommen, welche die gewünschten Unebenheiten in Form von variierenden Normalenvektoren für jeden einzelnen Texel (Abk.: Texture Element) enthält, die dann in einem Pixel Shader anstelle des interpolierten Norma-

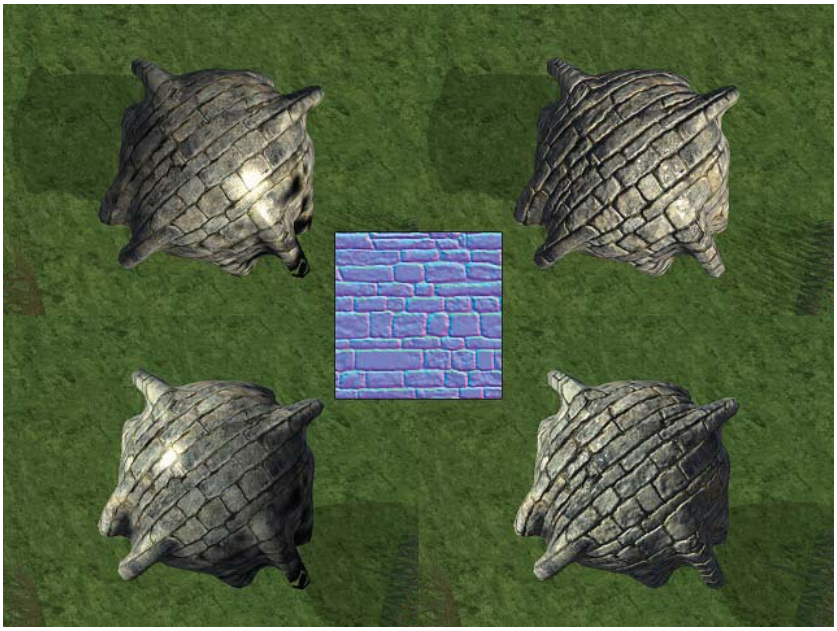

Abbildung 2.9: Einfaches Texture Mapping (links) wird durch die Normal Map (Mitte) optisch aufgewertet und täuscht durch die pro Pixel durchgeführten Beleuchtungsberechnungen des Normal Mappings (rechts) ein detaillierteres 3D-Modell vor.

lenvektors jedes Vertex für die Beleuchtung verwendet werden. Dadurch können dem Betrachter Oberflächendetails suggeriert werden, die im 3D-Gittermodell nicht vorhanden sind und im Fall einer tatsächlichen Modellierung zu einem immensen Rechenaufwand führen würden [Möller 08]. Darüber hinaus verändert sich das Aussehen der Oberfläche und das der Specular Highlights wesentlich, sobald die Position der Lichtquelle verändert wird, wohingegen beim herkömmlichem Texture Mapping das Erscheinungsbild der texturierten Oberfläche und des Specular Highlights unabhängig von der Position der Lichtquelle nahezu gleichbleibt (vgl. Abbildung 2.9).

Ursprünglich wurde das Verfahren 1978 als Bump Mapping vorgestellt [Blinn 78], bei dem eine sogenannte Height Map Verwendung findet, die als zweidimensionale Textur für jeden Texel lediglich individuelle Höheninformationen der Unebenheiten speichert. Da moderne Grafikhardware aber in der Lage ist, das nötige Punktprodukt für jeden Pixel in Echtzeit zu berechnen, wurde dazu übergegangen, die Height Map durch eine Normal Map zu ersetzen

[Daubert 03]. Zudem stellt die Speicherung der drei Koordinaten eines Normalenvektors in einer Textur für heutige Hardware kein Hindernis mehr dar und reduziert außerdem die Anzahl der nötigen Berechnungen pro Pixel im Vergleich zum ursprünglichen Bump Mapping, bei dem anhand der Höhenwerte benachbarter Pixel zunächst mit relativ viel Rechenaufwand Steigungen berechnet werden mussten, um den resultierenden Normalenvektor zu gewinnen [Möller 08]. Normal Mapping spielt für die vorliegende Arbeit insbesondere in Kapitel 4 eine wichtige Rolle, weswegen im Folgenden die grundlegende Funktionsweise näher erläutert werden soll.

Im Allgemeinen wird für die Normal Map ein Texturformat mit 8 Bit pro Farbkomponente gewählt, wobei der Rot-, Grün- und Blaukanal eines Texels mit der X-, Y- und Z-Koordinate des jeweiligen Normalenvektors korrespondiert. Da die Koordinaten einen Wertebereich von -1 bis 1 besitzen, in der Textur aber ausschließlich positive Werte zwischen 0 und 255 pro Kanal gespeichert werden können, muss vor der Verwendung eine Konvertierung durchgeführt werden. Der hellblaue Farbwert $T_{Normal\,Map} = (128, 128, 255)^T$ würde dann einer ebenen Fläche mit dem Normalenvektor $\mathbf{n} = (0, 0, 1)^T$ entsprechen, wobei mittels

$$\mathbf{n} = \frac{T_{Normal\,Map}}{255} \cdot 2 - 1 \tag{2.11}$$

die gesampelte Texelfarbe der Normal Map $T_{Normal\,Map} \in \mathbb{R}^3$ in den Normalenvektor $\mathbf{n} \in \mathbb{R}^3$ für den aktuellen Pixel umgewandelt werden kann [Möller 08].

Für eine korrekte Beleuchtung mittels Normal Mapping spielt das verwendete Koordinatensystem der Vektoren eine entscheidende Rolle. Eine Möglichkeit ist die Definition der Normalen in Weltkoordinaten, so dass das Punktprodukt zwischen dem ebenfalls in Weltkoordinaten vorliegenden Lichtrichtungsvektor ohne Umwege berechnet werden kann. Der Nachteil bei dieser Vorgehensweise ist allerdings, dass bei einer Rotation oder Verformung des 3D-Objekts, welches mittels dieser Normalenvektoren beleuchtet werden soll, die Normal Map nicht mehr gültig ist und neu berechnet werden muss. Außerdem ist es auf diesem Weg nicht möglich, dieselbe Normal Map für zwei Objekte mit unterschiedliche Orientierung zu verwenden, weswegen diese Form der Speicherung keine Verwendung findet. Eine andere Möglichkeit wird durch die Definition in Objektkoordinaten geboten, wobei zwar die Verwendung für Objekte gleicher Größe möglich ist, aber auch hier Verformungen des 3D-Objekts dazu führen, dass dieselbe Normal Map keine Gültigkeit mehr besitzt [Möller 08].

Die übliche Konvention für die Speicherung von heutigen Normal Maps ist daher die Ver-

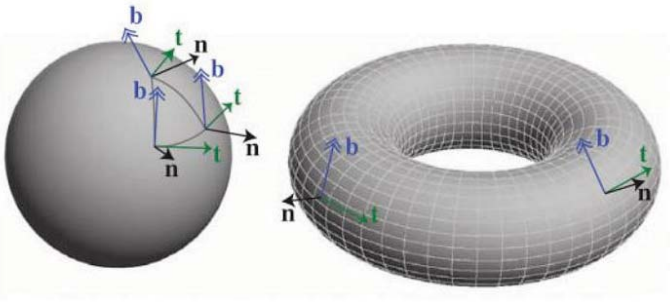

Abbildung 2.10: Tangentenraum bestehend aus Normalen **n**, Tangenten **t** und Bitangenten **b** am Beispiel einer Kugel und eines Torus [Möller 08].

wendung des sogenannten Tangentenraums, bei dem die Vektoren relativ zur Oberfläche definiert werden und dadurch beliebige Transformationen der Objekte gestattet sind und somit auch eine maximale Wiederverwendungsmöglichkeit derselben Normal Map geboten wird. Dabei wird dem Vertex Shader für jeden Vertex neben dem Normalenvektor **n** eine Tangente **t** übergeben, die vorher von der Anwendung mit Hilfe der Position, der Texturkoordinaten und dem Normalenvektor der Vertices des dazugehörigen Dreiecks im 3D-Modell berechnet wurde. Innerhalb des Vertex Shaders kann dann über ein Kreuzprodukt der dritte für die Aufspannung des Tangtenraums benötige Vektor $\mathbf{b} = \mathbf{n} \times \mathbf{t}$ ermittelt werden, der als Bitangente bezeichnet wird (vgl. Abbildung 2.10). Die drei Vektoren, die jeweils orthogonal zueinander stehen, bilden somit das lokale kartesische Koordinatensystem der Normal Map, das für jeden Vertex vorliegt und für die einzelnen Pixel ebenfalls linear interpoliert wird. Um die nötige Beleuchtungsberechnung im Pixel Shader durchzuführen, muss außerdem der Lichtrichtungsvektor und der View-Vektor vom Weltkoordinatensystem in das Koordinatensystem des Tangentenraums transformiert werden, was analog zu den in Abschnitt 2.3 beschriebenen Transformationsschritten mittels einer Matrix geschieht, die als

$$M_{WorldToTangent} = \begin{pmatrix} t_x & t_y & t_z \\ b_x & b_y & b_z \\ n_x & n_y & n_z \end{pmatrix} \tag{2.12}$$

definiert ist [Garstenauer 06, Möller 08]. Danach kann im Pixel Shader gemäß Gleichung 2.10 unter Verwendung des nun im Tangentenraum vorliegenden Normalenvektors $\mathbf{n_{TS}}$, des Lichtrichtungsvektors $\mathbf{l_{TS}}$ und des View-Vektors $\mathbf{v_{TS}}$ die finale Pixelfarbe berechnet werden,

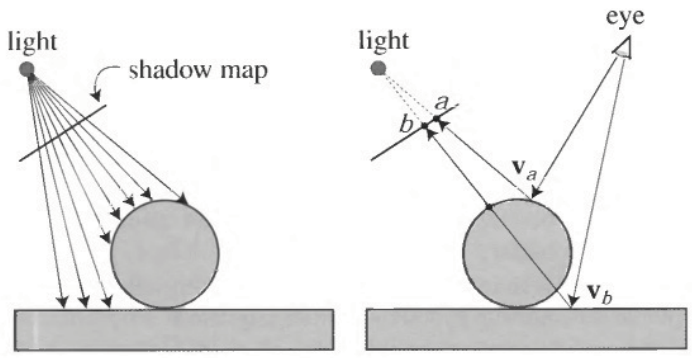

Abbildung 2.11: Links: Die Tiefenwerte der 3D-Szene werden aus der Perspektive der Lichtquelle in die Shadow Map gerendert. Rechts: Bei der Beleuchtung der 3D-Szene kann dann mit Hilfe der Shadow Map entschieden werden, ob sich Pixel im Schatten befinden oder nicht [Möller 08].

wobei sich der Specular-Anteil I_S wiefolgt ergibt:

$$\begin{aligned} \mathbf{r_{TS}} &= 2(\mathbf{v_{TS}} \cdot \mathbf{n_{TS}})\mathbf{n_{TS}} - \mathbf{v_{TS}} \\ I_S &= \max(\mathbf{r_{TS}} \cdot \mathbf{l_{TS}}, 0)^n \cdot L_{Specular} \end{aligned} \tag{2.13}$$

2.7 Schattenmodell

Um die räumliche Beziehung von Objekten in der virtuellen 3D-Szene zueinander zu verdeutlichen und um eine möglichst realistische Darstellung der Stoffsimulation zu erreichen, wurde ein Schattenmodell implementiert, welches in der Lage ist, dynamische, weiche Schatten in Echtzeit zu berechnen. Die hierbei verwendete Technik namens Shadow Mapping ist sehr flexibel und äußerst effizient, wodurch es möglich ist – bei ausreichend zur Verfügung stehender Rechenzeit – beliebig detaillierte Schatten zu produzieren. Daher findet es nicht nur in Computerspielen und anderen vom Echtzeitaspekt abhängigen Programmen Verwendung, sondern auch in modernen Animationsfilmen und professioneller Software für 3D-Modellierung, -Animation und -Rendering [Valient 04].

Konventionelles Shadow Mapping wird üblicherweise in zwei Phasen durchgeführt, wobei diverse Abwandlungen und Modifikationen existieren, die an dieser Stelle aber nicht weiter

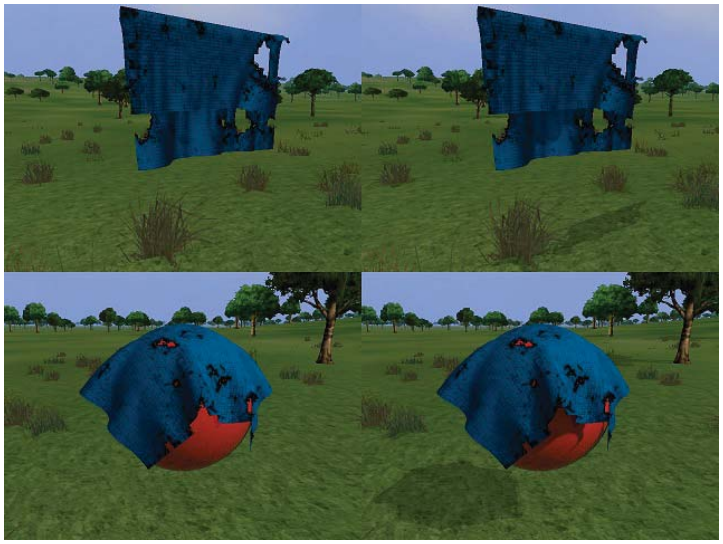

Abbildung 2.12: Links: Ohne Shadow Mapping gerenderte Szene. Rechts: Mit Shadow Mapping gerenderte Szene.

untersucht werden sollen. Dabei wird zunächst die 3D-Szene aus der Position der Lichtquelle gerendert, was im Fall der Sonne mittels einer orthografischen[7] statt perspektivischen Projektion geschieht, um der nahezu unendlich weit entfernten Lichtquelle und dem dadurch parallelen Lichteinfall Rechnung zu tragen. Beim Rendering der Szene aus der Position der Lichtquelle wird in beiden Fällen der Z-Wert bzw. Tiefenwert eines jeden Pixels in einer zweidimensionalen Textur – der Shadow Map – gespeichert, was für ein Spotlight auf der linken Seite von Abbildung 2.11 dargestellt ist. In einem zweiten Durchgang wird jetzt die 3D-Szene ganz normal aus der Position und Perspektive der Kamera gerendert, wobei in einem Pixel Shader, dem die Shadow Map aus dem ersten Durchgang zur Verfügung gestellt wird, überprüft werden kann, ob sich der aktuelle Pixel im Schatten befindet. In der rechten Seite von Abbildung 2.11 ist dieser Vorgang vereinfacht dargestellt, bei dem am Beispiel der Punkte $\mathbf{v_a}$ und $\mathbf{v_b}$ getestet wird, ob der dazugehörige Punkt in der Shadow Map – also $\mathbf{a}$ bzw. $\mathbf{b}$ – einen kleineren Tiefenwert besitzt. Da der Punkt $\mathbf{v_b}$ von der Kugel verdeckt wird – in der Shadow Map für $\mathbf{b}$ also ein kleinerer Tiefenwert gespeichert ist als der soeben für $\mathbf{v_b}$ ermittelte

[7] Bei einer orthografischen Projektion der 3D-Szene bleiben parallele Linien und Strukturen parallel zueinander, was bei der klassischen perspektivischen Projektion nicht der Fall ist, da dort weiter entfernte Objekte zunehmend kleiner abgebildet werden und sich in einem Fluchtpunkt schneiden [Möller 08].

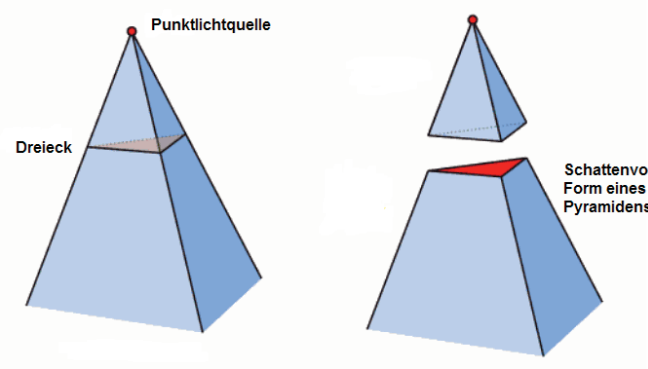

Abbildung 2.13: Links: Ein Dreieck wird von einem Punktlicht angestrahlt. Rechts: Unterhalb des Dreiecks bildet sich das Schattenvolumen in Form eines Pyramidenstumpfes [Möller 08].

Tiefenwert innerhalb des Z-Buffers[8] – kann er etwas dunkler eingefärbt werden als Punkte, die sich nicht im Schatten befinden wie beispielsweise $\mathbf{v_a}$ [Möller 08, Valient 04].

In Abbildung 2.12 ist eine Gegenüberstellung zu sehen, bei der jeweils zwei Szenen mit und ohne Zuhilfenahme von Shadow Mapping gerendert wurden. Die 3D-Objekte werfen dabei nicht nur aufeinander Schatten, sondern es kommt auch zum sogenannten Self Shadowing, also einer Selbstabschattung, was insbesondere beim Faltenwurf im unteren, rechten Screenshot oder beim hängenden Tuch im oberen, rechten Screenshot in Abbildung 2.12 gut zu erkennen ist. Die weichen Schatten werden dabei durch ein Verfahren namens PCF (Percentage-Closer Filtering) erreicht, bei dem nicht nur ein Tiefenwert aus der Shadow Map für jeden im zweiten Durchlauf zu beleuchtenden Pixel ausgelesen wird, sondern mehrere – in bestimmten, variablen Mustern angeordnete – Pixel. Für jeden dieser so gesampelten Werte aus der Shadow Map findet eine gewichtete Vermischung statt, so dass sich an den Schattenrändern, wo sich Pixel entweder innerhalb oder ausserhalb des Schattens befinden, ein weicherer Übergang schaffen lässt, als wenn wie beim herkömmlichen Ansatz nur zwei Möglichkeiten zugelassen werden [Valient 04].

[8] Der Z-Buffer speichert den Z-Wert eines jeden Pixels und wird zu Beginn jedes Frames mit dem größtmöglichen Wert 1 initialisiert. Wenn im Anschluss ein Dreieck gerendert wird, findet ein Vergleich zwischen dem Tiefenwert des zu rendernen Pixels und des schon vorhandenen Tiefenwerts im Z-Buffer statt. Nur falls der neue Z-Wert kleiner ist als der schon Vorhandene, wird er ersetzt und der Pixel weiter verarbeitet [Möller 08].

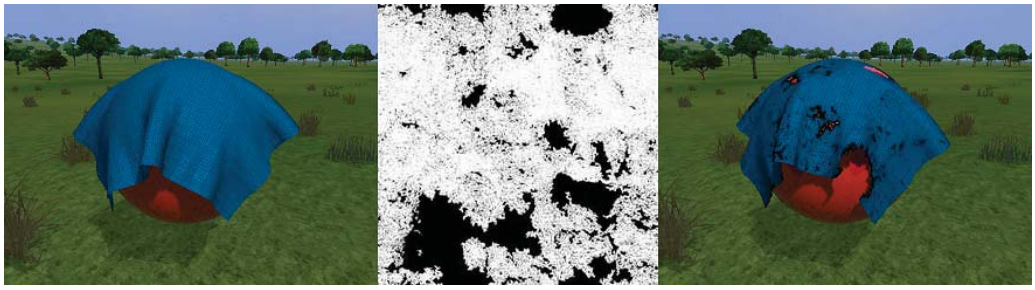

Abbildung 2.14: Links: Stofftuch ohne Löcher oder Risse, Mitte: Textur für die Darstellung von Löchern, Rechts: Stofftuch mit Löchern und Rissen

Eine alternative Möglichkeit zur Modellierung dynamischer Echtzeitschatten wird durch ein Verfahren namens Shadow Volumes (engl.: Schattenvolumen) geboten, das ebenfalls häufig in Computerspielen angewendet wird. Dabei werden vereinfacht ausgedrückt Lichtstrahlen, die von der Lichtquelle ausgehen, durch die Vertices des 3D-Modells gesendet, wodurch die Silhouette des Objekts bis in die Unendlichkeit extrudiert wird und sich das sogenannten Schattenvolumen konstituiert. In Abbildung 2.13 wird dieser Vorgang anhand eines Dreiecks verdeutlicht, das von einer Punktlichtquelle angestrahlt wird. Dabei bildet sich unterhalb des Dreiecks das Schattenvolumen, welches in diesem Fall als Pyramidenstumpf unendlich von der Lichtquelle weg extrudiert wird, wobei sich sämtliche Objekte, die innerhalb des Schattenvolumens liegen, im Schatten befinden [Kwoon 04].

Da für die Extrusion nicht das 3D-Objekt selbst, sondern ein abgeleitetes Silhouetten-Modell verwendet werden muss (vgl. [Kwoon 04]), eignet sich das Verfahren vor allem für Szenarien, in denen die 3D-Objekte keinen ständigen Verformungen unterworfen sind, da andernfalls kontinuierlich das Silhouttenmodell aktualisiert werden muss, was einen immensen Rechenaufwand nach sich ziehen würde. Da sich das in der vorliegenden Arbeit simulierte Stofftuch aber aufgrund von Wind, Gravitation, Kollisionen und Benutzereingaben ununterbrochen dynamisch verändert, verbietet sich eine Implementierung der Schatten mittels Shadow Volumes. Zudem müssten die Risse und Löcher, die beispielsweise in Abbildung 2.12 zu sehen sind, erst aufwändig in das 3D-Objekt bzw. in das resultierende Silhouetten-Modell übertragen werden, was bei der Verwendung von Shadow Mapping relativ einfach mittels Alpha Testing zu realisieren ist. Dabei wird eine vierte Komponente – der Alphawert – von der Farbe eines jeden Pixels interpretiert, mit der die Undurchsichtigkeit eines Pixels beeinflusst werden kann

und die dem Pixel Shader in Form einer Graustufen-Textur mit 8 Bit pro Texel zugeführt wird. Pixel, die einen Alphawert von 1 besitzen, werden in der besagten Textur mit der Farbe weiß repräsentiert und sind komplett undurchsichtig, wohingegen Pixel mit einem Alphawert von 0 der Farbe schwarz entsprechen und komplett durchsichtig sind. Falls ein festgesetzter Referenzwert unterschritten wird, kann die Bearbeitung des Pixels innerhalb der Rendering-Pipeline ausgesetzt werden, was zur Folge hat, dass der Tiefenwert des fraglichen Pixels nicht im Z-Buffer gespeichert wird [Möller 08]. Auf diesem Weg lassen sich z. B. die Löcher im Stoff und die entsprechenden Schattenwürfe produzieren, da im eingangs besprochenen zweiten Durchlauf des Shadow Mappings eben genau dieser Z-Wert als Vergleich herangezogen wird – für einen Riss oder ein Loch aber kein solcher Tiefenwert registriert wurde. Eine Gegenüberstellung eines Stofftuchs mit und ohne Rissen ist zusammen mit der verwendeten Textur für die Alphawerte in Abbildung 2.14 dargestellt.

3 Interaktive Stoffsimulation für eine 3D-Echtzeitumgebung

3.1 Einführung

Im Laufe der letzten Jahrzehnte wurden zahlreiche Modelle entwickelt, die sich mit der Simulation und Visualisierung von Kleidung und Stoffen auseinandergesetzt haben. Dabei wurde in erster Linie Wert auf eine physikalisch korrekte und zuverlässige Arbeitsweise der Algorithmen gelegt, mit dem Ziel eine möglichst authentische Reproduktion des Verhaltens von Stoffmaterialien zu erreichen [Baraff 98, Breen 94, Eberhardt 96]. Diese Herangehensweise, obgleich physikalisch akkurat und präzise in der Nachbildung der Realität, eignet sich allerdings nur bedingt für den Einsatz in interaktiven Anwendungen, wie Virtual Reality und Computerspielen. Auch wenn in diesen Fällen ein gewisser Grad an Realitätsnähe erwünscht und erstrebenswert ist, wird eher Wert auf die Möglichkeit einer schnellen Berechnung der Algorithmen sowie auf die Glaubwürdigkeit der Präsentation gelegt. Gründe hierfür liegen zum einen in der eingeschränkten Zeit pro Frame[1], die für Physikberechnungen zur Verfügung stehen, und zum anderen in der Tatsache, dass Korrektheit, mit dem Ziel der Glaubwürdigkeit, oftmals durch einfache Kniffe angenähert werden kann. Des Weiteren ist auch hier die Stabilität der Simulation eine grundlegende Voraussetzung für eine überzeugende Präsentation [Jakobson 01].

Andere, vor allem neuere, Versuche haben sich eben genau dieser Problematik angenommen und den Schwerpunkt auf Echtzeit und Interaktion gelegt, wobei approximative Rechenansätze verbunden mit performanten Methoden der Geometrieberechnung und -manipulation aus dem Bereich der Computergrafik zu überzeugenden und glaubhaften Animationen von unterschiedlichen Stoffmaterialien geführt haben [Fuhrmann 03, Jakobson 01, Meyer 01]. Auch

[1] In Computerspielen werden pro Sekunde eine gewisse Mindestanzahl von Frames (engl.: Rahmen) berechnet und angezeigt, um dem menschlichen Auge den Eindruck einer flüssigen Präsentation zu vermitteln.

das im Rahmen dieser Diplomarbeit verwendete Modell zählt zur Kategorie der approximativen Methoden, da der Rechenaufwand für physikalisch akkurate Methoden die Möglichkeiten einer interaktiven Echtzeitsimulation in der Regel übersteigt und die visuellen Unterschiede vom Benutzer ohnehin nicht wahrgenommen werden oder vernachlässigt werden können. Selbstverständlich existieren auch unter den neueren Arbeiten zahlreiche Modelle, welche den Echtzeitaspekt vernachlässigen, um dafür die detailgetreue Nachbildung des Stoffverhaltens zu erreichen, was beispielsweise im Hinblick auf Animationsfilme oder professionelle Rendering-Software, welche im Allgemeinen nicht den Anspruch von Echtzeitrendering besitzen, durchaus sinnvoll ist. Ein interessanter Ansatz, der in diese zuletzt genannte Kategorie fällt, konzentriert sich vor allem auf die effiziente und robuste Kollisionserkennung, die daraus resultierende Kollisionsbehandlung, sowie die Haftreibung und Stoffdicke, wodurch eine sehr eindrucksvolle und realitätsnahe Simulation von Stoffen produziert wird [Bridson 02]. Da eine solch aufwändige Kollisionsbehandlung für Echtzeitanwendungen aber mit heutiger Hardware nicht realisierbar ist, verzichten die meisten der erwähnten interaktiven Methoden auf diesen Aspekt der Modellierung.

Fast allen der genannten Arbeiten, ob mit dem Anspruch der Interaktionsmöglichkeit oder der physikalischen Korrektheit, liegt als Modell ein Partikelsystem zugrunde. Dieses einfach zu beschreibende und flexible Konstrukt wird in der Computergrafik für die Simulation von deformierbaren Objekten und Materialien, wie Stoffen oder Flüssigkeiten herangezogen, da es einen ausgewogenen Kompromiss zwischen Berechnungsgenauigkeit und Geschwindigkeit bietet [Fuhrmann 06]. In den nachfolgenden Abschnitten wird nun darauf eingegangen, wie mit Hilfe eines Partikelsystems Stoffmaterialien unter Berücksichtigung des Echtzeitaspekts und der Interaktion effizient modelliert und berechnet werden können, ohne dabei ein physikalisch glaubwürdiges Verhalten zu vernachlässigen. Die im Rahmen dieser Arbeit entwickelte 3D-Echtzeitumgebung wird hierbei exemplarisch visuelle Eindrücke der Simulation und Implementierungsdetails liefern, die an geeigneter Stelle die theoretischen Methoden und Modelle im Praxiseinsatz demonstrieren sollen.

3.2 Modellierung physikalischer Kräfte

Die mittels eines Partikelsystems modellierten Objekte werden durch miteinander verbundene Massepunkte, die Partikel, repräsentiert, welche miteinander interagieren können und auf die externe Kräfte, wie z. B. Gravitation, wirken können. Durch die Krafteinwirkung auf

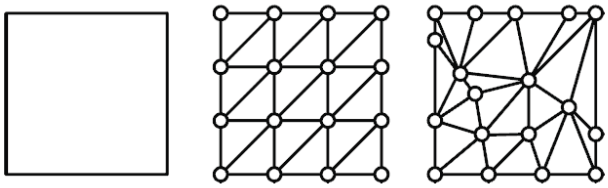

Abbildung 3.1: Gewebe als Partikelsystem: Reguläre und irreguläre Diskretisierung. [Cords 04]

die Partikel, welche den Gesetzen der Dynamik gehorcht, entsteht Bewegung, die wiederum in Form von Differentialgleichungen erfasst werden kann und deren Lösung die Bewegungsvektoren der einzelnen Partikel ergibt [Fuhrmann 06]. Die Interaktion zwischen einzelnen Partikeln wird durch interne Kräfte beschrieben, die je nach relativer Position der Partikel zueinander stark variieren können, was eines der Hauptprobleme bei Stoffsimulationen darstellt, da die hierdurch entstehenden sogenannten steifen Differentialgleichungen[2] effizient gelöst werden müssen. Diese Gleichungen resultieren aus den internen Kräften von Stoffmaterialien, die die Partikel zusammenhalten und dadurch einer zu starken Überdehnung entgegenwirken, gleichzeitig aber ein starkes Verbiegen zulassen [Fuhrmann 03]. Je nach Art der numerischen Integration ergeben sich hierbei neue Probleme bzw. Lösungen, die in den nachfolgenden Abschnitten beschrieben werden sollen. Um die Bewegung eines Partikels zu erfassen, müssen also interne und externe Kräfte berechnet werden, die auf einen Partikel wirken können.

3.2.1 Interne Kräfte und Constraints

Interne Kräfte

Im Allgemeinen wird für die Bestimmung der internen Kräfte ein einfaches Masse-Feder-System herangezogen, da auf diesem Weg die Kräfte zwischen zwei Partikeln durch lineare Federn modelliert werden können. Die darzustellenden Stoffe werden hierzu meist in Dreiecke diskretisiert, um beliebig geformte Kleidungsstücke simulieren zu können und gleichzeitig die Hardware-Beschleunigung moderner Grafikkarten für das Rendern von Dreiecken, also

[2] Der Begriff stammt von der Beschreibung schwingungsfähiger Systeme mit steifen Federn, d. h. Federn mit großer Federkonstante k. Wenn z. B. eine steife Feder mit einer weichen Feder innerhalb desselben System gekoppelt wird, erhält man ein System mit stark unterschiedlichen Zeitskalen, was die numerische Lösung der entsprechenden Differentialgleichungen erschwert [Schwarz 09].

Polygonen, auszunutzen. Die Partikel bzw. Massepunkte werden dann durch die Eckpunkte der Polygone und die Federn durch die Kanten zwischen den Eckpunkten repräsentiert [Fuhrmann 06]. In Abbildung 3.1 sind zwei Möglichkeiten einer solchen Diskretisierung dargestellt. Für die vorliegende Arbeit wurde eine gleichmäßige, reguläre Unterteilung gewählt, wie es bei der Modellierung von 3D-Objekten üblich ist [Cords 04].

Die resultierende Kraft $\mathbf{f_{ij}} \in \mathbb{R}^3$ zwischen zwei benachbarten Partikeln i und j berechnet sich mittels

$$\mathbf{f_{ij}} = k_{ij} \left(\|\mathbf{x_j} - \mathbf{x_i}\|_2 - l_{ij} \right) \cdot \frac{\mathbf{x_j} - \mathbf{x_i}}{\|\mathbf{x_j} - \mathbf{x_i}\|_2} \tag{3.1}$$

wobei $\mathbf{x_i} \in \mathbb{R}^3$ die Position des Partikels, $l_{ij} \in \mathbb{R}$ die Ruhelänge und $k_{ij} \in \mathbb{R}$ die Federkonstante zwischen den zwei Partikeln bezeichnet. In der Praxis ist die Verwendung von linearen Federn eigentlich eine unglückliche Wahl, da die Dehnungseigenschaften der meisten Stoffe nicht linear sind, sich also nach kurzer Zeit der Dehnung sehr steif verhalten. Untersuchungen von [Kawabata 80] zur Scherfestigkeit von Stoffen aus Baumwolle haben gezeigt, dass die Kräfte nur für einen gewissen Zeitraum linear approximierbar sind, abhängig von der Kett- und Schussrichtung[3] unterschiedlich sind und ab einer bestimmten maximalen Krafteinwirkung zum Einreißen der Kleidung führen [Breen 94]. Eine Möglichkeit ist daher die Verwendung von sehr großen Federkonstanten und die Nutzung einer impliziten Integration für das Lösen der Differentialgleichungen [Baraff 98]. Ein anderer Ansatz verwendet kleine Federkonstanten und korrigiert die Position überdehnter Federn in einem Nachbearbeitungsschritt, was beispielsweise von [Meyer 01] in ihrem Verfahren aufgegriffen wurde. Für diese Diplomarbeit wurde die Idee von [Jakobson 01] umgesetzt, bei der die Federn in Form von Stöckchen modelliert werden, also im Prinzip ihre Länge überhaupt nicht ändern. Das hat den entscheidenden Vorteil, dass der Stoff sehr schnell und stabil simuliert werden kann, was im Hinblick auf die vielen Fallstricke bei der Wahl der Integration und Art der Modellierung interner und externer Kräfte in den folgenden Abschnitten näher erläutert wird.

Man unterscheidet prinzipiell drei Formen von internen Kräften, welche direkten Einfluss auf die Steifigkeit, Elastizität und Torsionsspannung des Stoffmaterials haben und bei realen Stoffen neben dem verwendeten Material z. B. durch das Web-, Flecht- oder Strickmuster gegeben sind [Cords 04]:

- Dehnungs- bzw. Zugkräfte

[3] Die verwendeten Fäden bei gewebten Stoffen werden in der Längsrichtung als Kette bzw. Kettfäden und in der Querrichtung als Schuss bzw. Schussfäden bezeichnet [Daubert 03].

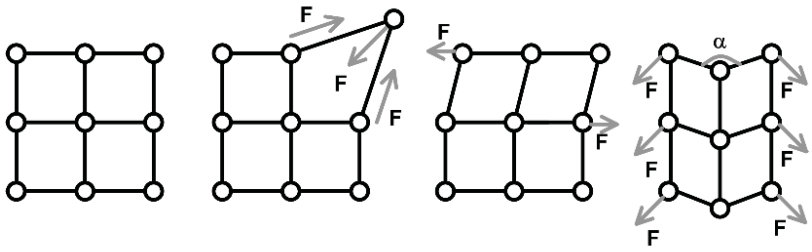

Abbildung 3.2: Wirkende Kräfte: Ausgangsstellung, Dehnungskräfte, Scherkräfte, Biegekräfte (v. l. n. r.). [Cords 04]

- Scherkräfte
- Biege- bzw. Neigungskräfte

Diese drei Kräfte sind charakteristisch für jedes einzelne Stoffmaterial und individuell von den physikalischen Eigenschaften des Gewebes abhängig. Wichtig für die überzeugende Darstellung der Materialien in einer interaktiven 3D-Echtzeitumgebung ist allerdings vor allem das physikalisch korrekte Aussehen und Verhalten und weniger die hundertprozentige physikalische Korrektheit und Übernahme aller Eigenschaften realer Stoffmaterialien. Daher kann auf die aufwendige Modellierung der mikroskopischen Gewebestruktur verzichtet werden und sich ganz auf das makroskopische Verhalten des Stoffes, abhängig von den gewählten internen Kräften, konzentriert werden. In Abbildung 3.2 ist die Wirkungsweise dieser Kräfte dargestellt.

Constraints

Zwischen den Partikeln werden also normalerweise eine Reihe von Federn festgelegt, um das gewünschte Maß an internen Kräften zu modellieren (vgl. Abbildung 3.3). Da es sich in dieser Arbeit in Anlehnung an [Jakobson 01] aber um unendlich steife Federn handelt, bietet es sich an, diese nicht als physikalische Kräfte zu behandeln, sondern als Entfernungs-Constraints[4] der Form

$$\|\mathbf{x_j} - \mathbf{x_i}\|_2 = l_{ij} \tag{3.2}$$

[4] Als Constraint wird üblicherweise eine Randbedingung bei der Lösung eines Optimierungsproblems bezeichnet. In Anlehnung an andere deutschsprachige Arbeiten, wird auch hier der englische Begriff verwendet.

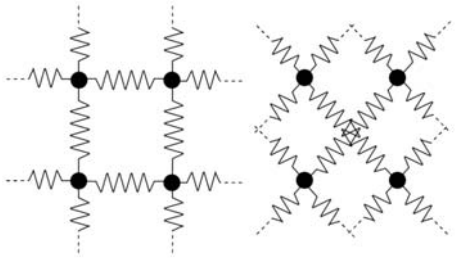

Abbildung 3.3: Partikel werden mit Federn verbunden, um Dehnungskräften (links) und Scher- bzw. Biegekräften (rechts) zu widerstehen. [Zeller 06]

wobei $l_{ij} \in \mathbb{R}$ die Distanz zwischen den beiden Partikeln widerspiegelt, die zu Beginn der Simulation einmalig berechnet wurde und mit der Ruhelänge der Feder korrespondiert. Anschaulich verdeutlichen lässt sich dieser Vorgang, indem man eine unendlich steife Feder zwischen zwei Partikel einbaut, die genau so stark ist und angemessen gedämpft wird, dass sie augenblicklich eine Ruhelänge von l_{ij} erreicht. Die so aufgestellten Constraints bilden für die Nachbarschaft eines jeden Partikels ein Gleichungssystem, welches in jedem Simulationsschritt mittels Relaxierung, also durch sukzessives Anwenden aller Constraints, gelöst wird [Zeller 06]. Üblicherweise wird bei einer Relaxierung gestoppt, sobald ein bestimmtes Konvergenzkriterium erreicht wurde. In unserem Fall genügt es, einige wenige Iterationsschritte durchzuführen. Die lokalen Constraints werden dabei nacheinander angewendet, um eine globale Konfiguration zu erreichen, die alle Constraints gleichermaßen erfüllt. Es zeigt sich, dass bereits eine Iteration ausreicht, um bei laufender Simulation ein visuell ansprechendes und zufriedenstellendes Ergebnis zu erhalten, da die gewählte Verlet-Integration dafür sorgt, dass auch zwischen zwei Simulationsschritten eine Konvergenz eintritt und trotz der Verwendung von Stöckchen bzw. unendlich steifen Federn, ein gewisses Maß an Elastizität erhalten bleibt [Jakobson 01]. Auf die Entscheidung für eine Verlet-Integration und deren Vorzüge wird in Abschnitt 3.3 näher eingegangen.

Für die im Rahmen dieser Arbeit implementierte Simulation eines rechteckigen Stofftuchs hat sich eine Anzahl von insgesamt zwölf Nachbarn pro Partikel als physikalisch überzeugend erwiesen. Das Muster der Nachbarschaft, welches verwendet wurde, ist in Abbildung 3.4 dargestellt. Unter Berücksichtigung der Art der eingesetzten Federn (vgl. Abbildung 3.3) dienen demnach vier Federn der Einschränkung von Dehnungskräften und acht dem Widerstand gegen Scher- und Biegekräfte. Für jeden dieser Nachbarn liegt ein Entfernungs-Constraint wie

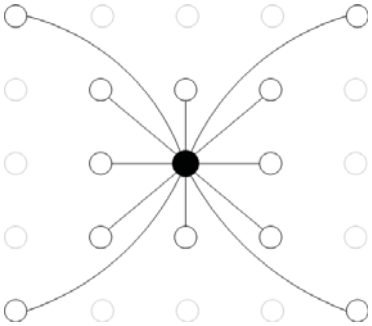

Abbildung 3.4: Jeder Partikel ist mit 12 Nachbarn verbunden.

in Gleichung 3.2 vor, wobei mittels

$$\mathbf{d_{ij}} = (\|\mathbf{x_j} - \mathbf{x_i}\|_2 - l_{ij}) \cdot \frac{\mathbf{x_j} - \mathbf{x_i}}{\|\mathbf{x_j} - \mathbf{x_i}\|_2} \tag{3.3}$$

die Positionsänderung $\mathbf{d_{ij}} \in \mathbb{R}^3$ nach Anwenden des Constraints berechnet wird. Die neuen Positionen $\mathbf{x_i'}$ bzw. $\mathbf{x_j'}$ der beiden Partikel werden dann durch

$$\begin{aligned} \mathbf{x_i'} &= \mathbf{x_i} + \mathbf{d_{ij}} \cdot r_i \\ \mathbf{x_j'} &= \mathbf{x_j} - \mathbf{d_{ij}} \cdot r_j \end{aligned} \tag{3.4}$$

berechnet, wobei $r_i \in \mathbb{R}$ und $r_j \in \mathbb{R}$ die Empfindlichkeit der Partikel zueinander widerspiegeln und mit dem normalisierten Inversen ihrer Massen korrespondieren. Die Einführung des Konzepts der Empfindlichkeit respektive der inversen Masse bietet eine bequeme Möglichkeit bestimmte Partikel zu fixieren, indem deren Empfindlichkeit bzw. inverse Masse auf Null gesetzt wird, was einer unendlich hohen Masse entspricht und gleichzeitig in Anbetracht von Gleichung 3.4 eine Änderung der Partikelposition verhindert. Standardmäßig wird für r_i und r_j der Wert $0,5$ verwendet, was mit einer übereinstimmenden Masse für beide Partikel korrespondiert, da in dieser Arbeit vereinfachend davon ausgegangen wird, dass jeder Partikel dieselbe Masse besitzt.

Nach dem Anwenden eines Entfernungs-Constraints zwischen zwei Partikeln müsste also die Distanz zwischen den Partikeln der Ruhelänge ihrer Feder zu Beginn der Simulation entsprechen. Das ist natürlich nach einem Relaxierungsschritt nicht mehr der Fall, da beide Partikel

Algorithmus 3.1 Pseudocode für den Ablauf der Stoffsimulation.

```
Initialisierung: Abstand benachbarter Partikel berechnen (Ruhelänge)
In jedem Simulationsschritt
    Update der Normalenvektoren
    Beschleunigungsvektor a := 0
    Zeitschritt t wählen
    Für alle Partikel
        Gravitation zum Beschleunigungsvektor a dazuaddieren
        Wind berechnen und zum Beschleunigungsvektor a dazuaddieren
        Für alle Nachbarn
            Entfernungs-Constraints anwenden
        Kollisions-Constraints anwenden
        Verletintegration unter Einbeziehung von a und t
```

inzwischen auch anderen Constraints unterworfen waren. Das hat zur Folge, dass sich der Stoff, trotz der Verwendung von Stöckchen bzw. unendlich steifen Federn wie bereits erwähnt leicht elastisch verhält [Zeller 06].

3.2.2 Externe Kräfte

Eine ganze Reihe von äußeren Einflüssen spielen für das Verhalten des im Rahmen dieser Arbeit implementierten Stofftuchs eine Rolle. Zu den Wichtigsten darunter zählen die Gravitation, Eingaben durch den Benutzer, die Kollision mit anderen Objekten und der Einfluss von Wind. Dabei werden Gravitation und Wind als Beschleunigungsvektor formuliert und über die Zeit integriert. Die Überprüfung auf Kollisionen mit anderen Objekten erfolgt ähnlich den in Abschnitt 3.2.1 beschriebenen Entfernungs-Constraints in Form von Gleichungen und Relaxierung, also nicht unter Einbeziehung der Zeit. Die Nutzerinteraktion mit dem Stofftuch greift wiederum das Konzept der inversen Masse auf und erfolgt ebenfalls unabhängig von der Zeit. Algorithmus 3.1 beschreibt dabei den groben Ablauf der Simulation und bringt die in diesem Abschnitt beschriebenen externen Einflüsse und die im nachfolgenden Abschnitt erläuterte Integration der Bewegungsgleichung in einen zeitlich-kausalen Zusammenhang.

Gravitation

In einem ersten Schritt wird ein Beschleunigungsvektor $\mathbf{a_i} = (0; -9,81; 0)^T \in \mathbb{R}^3$ für jeden Partikel festgelegt, welcher der Fallbeschleunigung $g = 9,81\frac{m}{s^2}$ für einen Körper an der Erdoberfläche Rechnung trägt und auf den Boden der Szene gerichtet ist. Zusammen mit dem Einfluss des Windes bildet die Gravitation eine der beiden globalen Konstanten, die in jedem

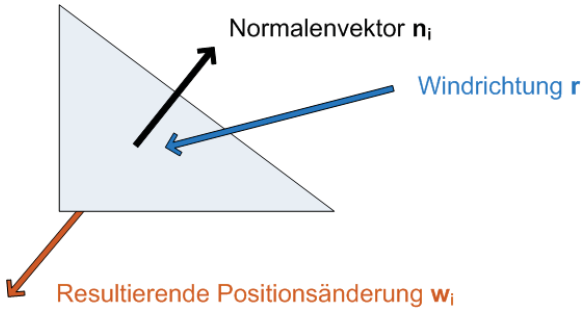

Abbildung 3.5: Einfluss der Windrichtung auf die Positionsänderung eines Partikels.

Simulationsschritt unveränderlich auf die Partikelpositionen einwirken.

Wind

Für die Simulation von Wind werden neben der für alle Partikel einheitlichen Windrichtung $\mathbf{r} \in \mathbb{R}^3$ und einer Windstärke $s \in \mathbb{R}$, die beide in der dynamischen 3D-Echtzeitsimulation jederzeit an die gewünschte Situation angepasst werden können, die Normalenvektoren der Oberfläche des Stofftuchs benötigt, um die neue Bewegungsrichtung unter dem Einfluss des Windes berechnen zu können, was in Abbildung 3.5 anschaulich dargestellt ist. Da sich das Stofftuch unter dem Einfluss des Windes bereits ausreichend dynamisch und überzeugend verhält, wurde bei der Implementation auf eine für jeden Partikel spezifische Windrichtung oder -stärke verzichtet. Die Berechnung der Positionsänderung erfolgt im Programm mittels

$$\mathbf{w_i} = s\,(\mathbf{n_i} \cdot \mathbf{r})\,\mathbf{n_i} \tag{3.5}$$

wobei $\mathbf{n_i} \in \mathbb{R}^3$ den individuellen Normalenvektor und $\mathbf{w_i} \in \mathbb{R}^3$ die resultierende Positionsänderung für jeden Partikel repräsentiert [Meyer 01]. Der hierzu in jedem Simulationsschritt aktualisierte Normalenvektor $\mathbf{n_i}$ wird neben der Windberechnung auch für die pixelbasierte Beleuchtung benötigt (vgl. Kapitel 2) und wird zu Beginn jedes Simulationsschrittes neu berechnet. Dabei wird für jeden Partikel i mit Hilfe der Positionen seiner beiden ersten Nachbarn j und k über die Berechnungsvorschrift

$$\mathbf{n_i} = \frac{(\mathbf{x_j} - \mathbf{x_i}) \times (\mathbf{x_k} - \mathbf{x_i})}{\left\|(\mathbf{x_j} - \mathbf{x_i}) \times (\mathbf{x_k} - \mathbf{x_i})\right\|_2} \tag{3.6}$$

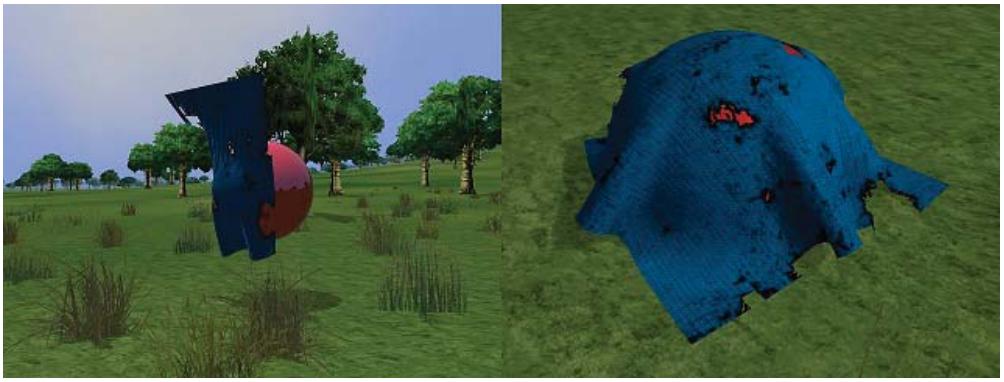

Abbildung 3.6: Links: Wind drückt das Stofftuch gegen die als Kollisionsobjekt definierte Kugel. Rechts: Kugel und Terrain als Kollisionsobjekte.

der neue normalisierte Normalenvektor erzeugt. Wie bereits erwähnt, wird die durch den Wind verursachte Positionsänderung $\mathbf{w_i}$ zum bisherigen Beschleunigungsvektor $\mathbf{a_i}$ dazuaddiert, welcher nun den Einfluss von Gravitation und Wind reflektiert und in Abschnitt 3.3 für die Integration der Bewegungsgleichung benötigt wird.

Kollisionen mit anderen Objekten

Eine Möglichkeit mit Kollisionen umzugehen sind sogenannte strafbasierte Verfahren, die an den Penetrationspunkten Federn verwenden, um die Objekte voneinander fernzuhalten. Dies ist zwar leicht zu implementieren, birgt allerdings auch den Nachteil, dass zunächst eine angemessene Federkonstante gewählt werden muss, um eine Kollision weitestgehend zu vermeiden und gleichzeitig ein stabiles System zu gewährleisten. Andere Ansätze versuchen den exakten Zeitpunkt der Kollision während der Simulation zu bestimmen, um von dort ausgehend die Simulation nach Auflösung der Kollision wiederaufzunehmen, was sich aber hinsichtlich des Anspruchs einer Echtzeitanwendung nicht als praktibal erweist [Jakobson 01].

Daher wird in dieser Arbeit ein sehr schnelles und stabiles Verfahren namens Projektion verwendet, bei dem kollidierende Partikel des Stofftuchs auf die Oberfläche des Hindernisses projiziert werden. Dabei wird der fragliche Partikel entlang der nächstgelegenen Oberflächennormalen des Kollisionsobjektes verschoben, bis keine Durchdringung mehr stattfindet [Jakobson 01]. Analog zu den Entfernungs-Constraints aus dem letzten Abschnitt werden hierzu Kollisions-Constraints aufgestellt und in der Relaxierungsphase angewendet. Auch hier

genügt eine Iteration, um ein physikalisch überzeugendes Resultat zu erhalten und Kollisionen zwischen dem Tuch und anderen Objekten zu vermeiden.

Denkbare Kollisionsobjekte sind beispielsweise Kugeln, Zylinder und Quader, die angelehnt an den englischen Sprachgebrauch auch oft Bounding Spheres, Cylinders und Boxes genannt werden. Die Kollision mit einer Kugel soll an dieser Stelle exemplarisch verdeutlichen, wie ein Kollisions-Constraint im Programm umgesetzt wird. Definiert man $M \in \mathbb{R}^3$ als Mittelpunkt und $r \in \mathbb{R}$ als Radius der Kugel, dann kann mittels

$$\mathbf{x_i'} = \frac{\mathbf{x_i} - M}{\|\mathbf{x_i} - M\|_2} \cdot r + M \tag{3.7}$$

die neue Position $\mathbf{x_i'}$ berechnet werden, die für einen Partikel i nötig ist, um die Kollision mit minimalem Aufwand aufzulösen, falls der Abstand $\|\mathbf{x_i} - M\|_2$ zwischen Partikel und Kugelmittelpunkt kleiner ist als der Kugelradius r.

Neben der Kugel als Kollisionsobjekt wird in der vorliegenden Arbeit ebenfalls das Terrain zur Überprüfung auf Kollisionen herangezogen. Hierzu wird zunächst die aktuelle Partikelposition, die in lokalen Modell-Koordinaten (vgl. Abschnitt 2.3) vorliegt, in das globale System der Weltkoordinaten transformiert. Nun kann eine pixelgenaue Kollisionsabfrage an die Terrainklasse gerichtet werden und die exakte Höhe des Geländes unter dem Partikel ermittelt werden. Die berechnete Höhe wird dann wieder in das lokale System der Modell-Koordinaten des Stofftuchs transformiert und kann als weiteres Kollisions-Constraint behandelt werden. Falls also die Höhe der Partikelposition kleiner ist als die des Terrains, wird die Höhe des Partikels auf die des Terrains gesetzt. Um visuelle Artefakte durch sich überlagernde Terrain- und Stofftuchpolygone zu vermeiden, kann zusätzlich ein Toleranzwert von einem Pixel zur Geländehöhe addiert werden, was in ähnlicher Form auch bei der bereits beschriebenen Kollision mit einer Kugel getan wird. In dem linken Screenshot aus Abbildung 3.6 ist die Auswirkung des Windes auf das Stofftuch zu sehen, der das Tuch gegen die Kugel drückt – auf der rechten Seite das Ergebnis, wenn Terrain und Kugel gleichermaßen als Kollisionsobjekte dienen.

Benutzerinteraktion

Eingaben durch den Benutzer erfolgen mittels der Maus und ermöglichen die freie Bewegung des Tuchs in der 3D-Umgebung. Die Partikel an einer Seite entlang des Tuchs dienen hierzu als Anfasser, was auch im linken Screenshot von Abbildung 3.6 zu sehen ist, wo offenbar die obere

Seite des Tuchs nicht der Gravitation oder anderen Kräften unterliegt. Da eine unmittelbare Umsetzung der Bewegung gewünscht ist und jedwede andere Einflüsse unterbunden werden müssen, werden die Positionen der gegriffenen Partikel in jedem Simulationsschritt direkt an die vom Anwender forcierte Position gesetzt. Das hat zur Folge, dass externe Kräfte wie Wind und Gravitation umgangen werden.

Um die höhere Priorität gegenüber allen internen Kräften, die in Form der Entfernungs-Constraints vorliegen, innerhalb des Systems zu gewährleisten, wird die Masse der fraglichen Partikel als unendlich hoch festgelegt, indem die in Gleichung 3.4 eingeführte Empfindlichkeit r_i, welche wie bereits erwähnt mit der inversen Masse korrespondiert, gleich Null gesetzt wird [Fuhrmann 03]. Dadurch hängen die Positionen der festgehaltenen Partikel nur noch von den Benutzereingaben ab und können ohne Einfluss anderer Kräfte bewegt werden.

3.3 Integration der Bewegungsgleichung

3.3.1 Explizite Verfahren

Die Bewegung eines Partikels wird im Allgemeinen mittels der Newtonschen Bewegungsgleichung

$$\mathbf{f_i} = m_i \cdot \mathbf{a_i} = m_i \cdot \frac{d^2\mathbf{x_i}}{dt^2} \tag{3.8}$$

berechnet, wobei $\mathbf{f_i} \in \mathbb{R}^3$ die wirkende Kraft auf den Partikel, $\mathbf{a_i} \in \mathbb{R}^3$ die Beschleunigung und m_i die Masse des Partikels bezeichnet. Die Kraft $\mathbf{f_i}$ verkörpert hierbei alle internen und externen Kräfte, die auf den Partikel i einwirken. Nach Umstellung von Gleichung 3.8 nach $\mathbf{a_i}$, kann mittels der beiden Gleichungen

$$\mathbf{v_i^{n+1}} = \mathbf{v_i^n} + \frac{\mathbf{f_i^n}}{m_i} dt \tag{3.9}$$

$$\mathbf{x_i^{n+1}} = \mathbf{x_i^n} + \mathbf{v_i^{n+1}} dt \tag{3.10}$$

in jedem Simulationsschritt die neue Position $\mathbf{x_i^{n+1}}$ des Partikels berechnet werden, wobei $\mathbf{v_i} \in \mathbb{R}^3$ die Geschwindigkeit des Partikels repräsentiert. Dieses Integrationsschema, das auch als explizite Euler-Integration bekannt ist, zeichnet sich dadurch aus, dass die neuen Geschwindigkeiten und Positionen der Partikel direkt, also explizit, berechnet werden kön-

nen, wobei die wirkenden Kräfte zum Zeitpunkt n zur Geschwindigkeit zum Zeitpunkt $n+1$ beitragen [Fuhrmann 03].

Obwohl sie einfach zu implementieren ist und einen geringen Rechenaufwand besitzt, hat die explizite Euler-Integration einige Nachteile, die im Folgenden besprochen werden sollen. Zum einen neigt diese Methode zur Instabilität, wenn es darum geht starke Kräfte, wie sie in Form von internen Kräften bei Stoffen vorliegen, bei Verwendung von großen Zeitschritten zu simulieren. Tatsächlich wird sogar ein Integrationszeitschritt dt verlangt, der umgekehrt proportional zur Quadratwurzel der Steifigkeit ist, was in der Physik als Courant-Bedingung bekannt ist [Meyer 01]. Das würde aber bedeuten, dass nur sehr elastische Stoffe bei großen Zeitschritten simuliert werden können, was mit einer starken Einschränkung der Darstellungsmöglichkeiten einhergeht. Steifere Materialien wären auf diesem Weg nur bei sehr kleinen Zeitschritten möglich, was sich unter den Bedingungen einer Echtzeitanwendung wie in der hier vorliegenden Arbeit als Zeitlupeneffekt zeigen würde. Ignoriert man bei der Verwendung von expliziten Integrationsmethoden die Anforderungen an den Zeitschritt, verhält sich das System nicht mehr stabil und kollabiert, was sich in einer unkontrollierten Bewegung der Partikel äußert [Press 92].

3.3.2 Implizite Verfahren

Ein anderes Problem der expliziten Integration zeigt sich bei großen und schnellen Positionsänderungen der Partikel, was im Hinblick auf die Interaktion mit dem Benutzer oder auch bei der Kollisionsbehandlung relevant wird. Auch in diesen Fällen tendiert das System zur Instabilität und kollabiert unter Umständen, so dass ein Zurücksetzen in einen Initialzustand nötig ist. Um das zu vermeiden, wird von [Meyer 01] nach jedem Zeitschritt eine Nachbearbeitung durchgeführt, der zu grosse Positionsänderungen oder überdehnte Federn anhand von vordefinierten Constraints eindämmt und so ein ausreichendes Maß an Stabilität für das gesamte System gewährleistet. Das entspricht einer Änderung vor jedem Zeitschritt mittels

$$\mathbf{v_i^n} = \frac{\mathbf{x_i^n} - \mathbf{x_i^{n-1}}}{dt} \tag{3.11}$$

wobei die Geschwindigkeit zum Zeitpunkt n jetzt nur noch durch die Partikelpositionen zum Zeitpunkt n und $n-1$ ausgedrückt wird [Fuhrmann 03]. Dies erhöht die Stabilität der numerischen Integration in hohem Maße, da die implizit approximierte Geschwindigkeit zu einer gewissen Rückkopplung innerhalb des System führt, die Instabilitäten weitestgehend

verhindert. Das dadurch beschriebene Integrationsschema entspricht der Verlet-Integration [Verlet 97]

$$\mathbf{x_i^{n+1}} = 2\mathbf{x_i^n} - \mathbf{x_i^{n-1}} + \frac{\mathbf{f_i^n}}{m_i} dt^2 \tag{3.12}$$

welches auf die Verwendung von Geschwindigkeiten verzichtet und sich wegen der hohen Stabilität und einfachen Handhabung gerade in interaktiven 3D-Echtzeitumgebungen hoher Beliebtheit erfreut [Fuhrmann 06]. Setzt man Gleichung 3.9 in Gleichung 3.10 ein, erhält man

$$\mathbf{x_i^{n+1}} = \mathbf{x_i^n} + (\mathbf{v_i^n} + \frac{\mathbf{f_i^n}}{m_i} dt) dt \tag{3.13}$$

und nach anschließender Substitution der Geschwindigkeit gemäß Gleichung 3.11

$$\mathbf{x_i^{n+1}} = \mathbf{x_i^n} + \mathbf{x_i^n} - \mathbf{x_i^{n-1}} + \frac{\mathbf{f_i^n}}{m_i} dt^2 \tag{3.14}$$

was genau der Gleichung der Verlet-Integration entspricht. Der in Abschnitt 3.2.2 eingeführte Beschleunigungsvektor $\mathbf{a_i}$, der zum Zeitpunkt der Integration (vgl. Algorithmus 3.1) die Einflüsse von Gravitation und Wind widerspiegelt, dient hierbei als Approximation für die Beschleunigung $\frac{\mathbf{f_i^n}}{m_i}$ und wird in der Implementation über die Zeit integriert. Entfernungs- und Kollisions-Constraints modifizieren die Partikelpositionen hingegen unmittelbar und liefern durch die so vorliegende Positionsänderung implizit eine Geschwindigkeit, die vom Verlet-Integrationsschema im nächsten Simulationsschritt berücksichtigt wird.

Des Weiteren eignet sich die Verlet-Integration besonders für Probleme mit geringer oder keiner Dämpfung [Hauth 02]. Um innerhalb des Systems Dämpfung nachzubilden, kann Gleichung 3.11 mit einem Dämpfungsfaktor multipliziert werden. Falls keine Dämpfung erwünscht ist, wird der Dämpfungsfaktor gleich 1 gesetzt. Für die hier vorgestellte Implementation wurde ein Faktor von $0,99$ gewählt, um einen geringen Luftwiderstand zu simulieren und eine natürlichere Bewegung des Stofftuchs zu erreichen.

Die soeben formulierte implizite Integration bietet also eine enorme numerische Stabilität auch bei der Verwendung von großen Zeitschritten und ist außerdem resistent gegen schnelle Positionsänderungen, wie sie gerade bei der Benutzerinteraktion häufig auftreten können. Zudem können auf diesem Weg durch Constraints und das verwendete Konzept der Relaxierung physikalische Kräfte überzeugend und effizient modelliert werden, um eine schnelle und

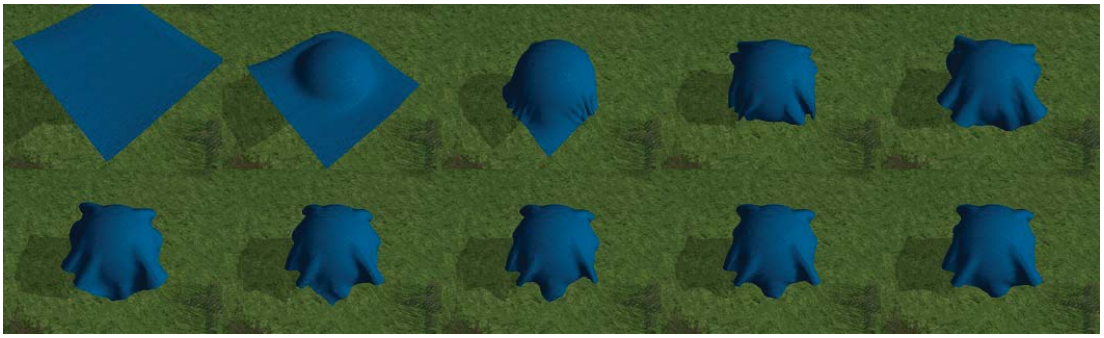

Abbildung 3.7: Ablauf der Stoffsimulation, wenn das Tuch über einer Kugel fallengelassen wird.

robuste Stoffsimulation zu gewährleisten.

3.4 Ergebnisse

Um einen Eindruck zu bekommen, wie sich die Stoffsimulation in Bewegung verhält, wurde in Abbildung 3.7 eine Sequenz von Screenshots angefertigt, die ein Stofftuch zeigt, das über einer als Kollisionsobjekt definierten Kugel fallengelassen wurde. Die Wölbungen an den Rändern des Tuchs, die entstehen, sobald sich das Tuch seiner Ruheposition auf der Kugel nähert, sind Ausdruck einer stabilen Endkonfiguration des Systems, die alle Constraints gleichermaßen erfüllt. Das Tuch steht hierbei zwar weiterhin unter dem Einfluss der Gravitation, der Wind wurde für diese Aufnahmen jedoch deaktiviert. So erhält man einen natürlichen Faltenwurf des Stoffes, wobei die Größe der Falten und Wölbungen an den Rändern abhängig von der Steifigkeit des verwendeten Materials ist.

Steifere Stoffe lassen sich modellieren, indem die Nachbarschaft für jeden Partikel (vgl. Abbildung 3.4 auf Seite 36) – und hier insbesondere die äußeren vier Nachbarn, welche Widerstand gegen Biegekräfte leisten – verändert wird. Standardmäßig wird für die vier äußeren Nachbarn ein diagonaler Abstand von 2 verwendet. Wählt man hier 4 oder 6, lassen sich die in der linken Spalte von Abbildung 3.8 dargestellten Steifigkeiten erreichen. Es fällt auf, dass der Stoff bei höherem Abstand der äußeren Nachbarn zum Ausgangspartikel straffer und fester wirkt, und zum Erfüllen der neuen Constraints die Wölbungen größer ausfallen. Eine noch höhere Steifigkeit lässt sich erreichen, wenn zusätzlich vier neue Nachbarn hinzugefügt wer-

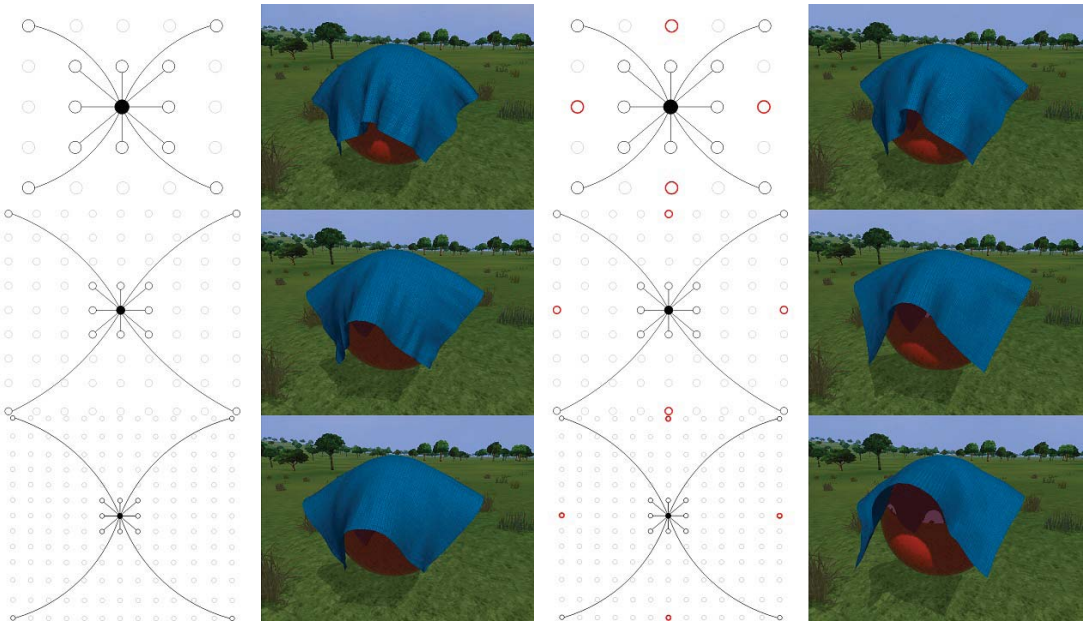

Abbildung 3.8: Links: Ursprüngliches Nachbarschaftsmuster mit variierenden diagonalen Abständen (2, 4 und 6) zu äußeren Nachbarn.
Rechts: Vier weitere Nachbarn werden hinzugefügt und die Abstände auf gleiche Weise variiert.

den, was in der rechten Spalte von Abbildung 3.8 dargestellt ist, wobei die neuen Nachbarn rot markiert wurden. In jedem Fall bleiben die inneren acht Nachbarn mit einem Abstand von 1 unverändert.

Prinzipiell lassen sich beliebig viele zusätzliche Nachbarn einfügen, um noch steifere Stoffe zu modellieren. Da in jedem Simulationsschritt jedoch für jeden Partikel sämtliche Constraints aller Nachbarn angewendet werden, steigt der Rechenaufwand immens an. Das hier gezeigte Stofftuch besteht aus 32x32 Partikeln, wodurch sich eine Gesamtzahl von $n = 1024$ Partikeln ergibt, für die in jedem Schritt alle $m = 12$ Nachbarn behandelt werden müssen, was eine Komplexität der Laufzeit von $\mathcal{O}(n \cdot m)$ ergibt, die sich auch nicht ohne Weiteres verringern lässt. Will man also nicht den Echtzeitaspekt vernachlässigen, verbietet sich eine zu hohe Anzahl von Nachbarn. Da die maximale Anzahl von Nachbarn, die möglich ist, ohne interaktive Bildwiederholraten zu unterschreiten, stark implementierungsabhängig ist und von mehreren Faktoren wie der verwendeten Hardware und den neben der Stoffsimulation zu berechnenden

Aufgaben abhängt, soll an dieser Stelle nicht näher darauf eingegangen werden.

Eine Möglichkeit, den visuellen Eindruck der Stoffsimulation aufzuwerten und gleichzeitig die Komplexität der Berechnungen zu verringern, ist die Modellierung von Rissen (vgl. Abschnitt 2.7). Dabei werden mittels einer Textur, die das sichtbare Muster der Risse beinhaltet und mit der Stofftextur kombiniert wird, zusätzlich Informationen über die Nachbarschaft eines jeden Partikels gewonnen. Insbesondere werden die Partikel, die sich innerhalb eines Loches oder Risses befinden, gesondert markiert und bei der Anwendung der Entfernungs- und Kollisions-Constraints übersprungen. Die so markierten Partikel haben somit keinen Einfluss mehr auf die Stoffsimulation und können als Partikel selbst von n und im Sinne von Nachbarn von m abgezogen werden, wobei nun für jeden Partikel ein individuelles m_i vorliegt, wodurch nicht mehr eine allgemeine Laufzeitkomplexität bestimmt werden kann.

Eine weitere Möglichkeit, die Anzahl der Berechnungen pro Sekunde zu reduzieren, von der auch in der vorliegenden Arbeit Gebrauch gemacht wird, ist die Anzahl der Aktualisierungen der gesamten Stoffsimulation pro Sekunde auf einen konstanten Wert zu beschränken. Um einen flüssigen Bewegtbild-Eindruck zu erhalten, reicht es beispielsweise völlig aus, die Simulation nur 60 oder 100 mal pro Sekunde aufzurufen und die damit verbundenen Berechnungen durchzuführen, wobei das Tuch weiterhin in jedem Frame gerendert wird. Auf diesem Weg kann der Rechenaufwand effizient verringert werden, ohne das visuelle Ergebnis zu verändern. Zudem wird dafür Sorge getragen, dass die Simulation bei einem leistungsfähigeren Prozessor nicht zu schnell abläuft. Andersherum kann mit demselben Mechanismus überprüft werden, ob die Simulationsberechnungen bereits 60 bzw. 100 mal pro Sekunde ausgeführt wurden, um auch auf langsameren Prozessoren die angestrebte Animationsgeschwindigkeit zu erreichen. Falls das zu Beginn eines jeden Simulationsschrittes noch nicht der Fall ist, wird ein Solcher durchgeführt und anschließend erneut geprüft, ob inzwischen die gewünschte Aktualisierungsrate erzielt wurde.

4 Modellierung von Reflexionseigenschaften verschiedener Stoffmaterialien

4.1 Einführung

Nachdem in den bisherigen Kapiteln die Grundlagen der Beleuchtung sowie die physikalische Simulation des Stofftuchs vor allem auf der Makroebene untersucht wurden, soll in diesem Kapitel eine Methode vorgestellt werden, um die charakteristische Oberflächenbeschaffenheit verschiedener Stoffmaterialien aus den Aufnahmen realer Stoffproben – insbesondere der Mesostruktur[1] – zu extrahieren und mit Hilfe des in Abschnitt 2.6 vorgestellten Verfahrens namens Normal Mapping zu visualisieren. Hierbei finden die sogenannten Bidirektionalen Texturfunktionen (BTF) Verwendung, mit deren Hilfe es möglich ist, in Abhängigkeit von Blickwinkel und Lichteinfall die Reflexionseigenschaften verschiedener Stoffmaterialien einzufangen und in Form einer Textur auf beliebige 3D-Objekte aufzutragen [Wang 05]. Da bei den Aufnahmen enorme Datenmengen entstehen, was eine Verwendung für Echtzeitanwendungen nur unter großem zusätzlichen Aufwand ermöglicht, wurden in der Vergangenheit unterschiedliche Ansätze erforscht, die Daten möglichst kompakt zu repräsentieren und effizient zu rendern, weswegen einige dieser Ansätze in Abschnitt 4.2.6 aufgeführt werden sollen [Irawan 08].

Des Weiteren besteht neben den datenbasierten Bidirektionalen Texturfunktionen die Möglichkeit, die Stoffmaterialien durch sogenannte Bidirektionale Reflektanzverteilungsfunktionen (BRDF) analytisch zu modellieren, weswegen im folgenden Abschnitt eine Definition und Gegenüberstellung dieser beiden Funktionsklassen erfolgen soll. Da der Echtzeitaspekt für die vorliegende Arbeit im Vordergrund steht, wurde eine Methode entwickelt, die gesammelten Daten geeignet zu komprimieren – ohne die wesentlichen Charakteristika der Stoffmaterialien zu vernachlässigen – und im Anschluss die Ergebnisse der Reduktion für ein Rendering in

[1] Feine, aber mit dem bloßen Auge sichtbare Struktur eines Materials [Wang 05].

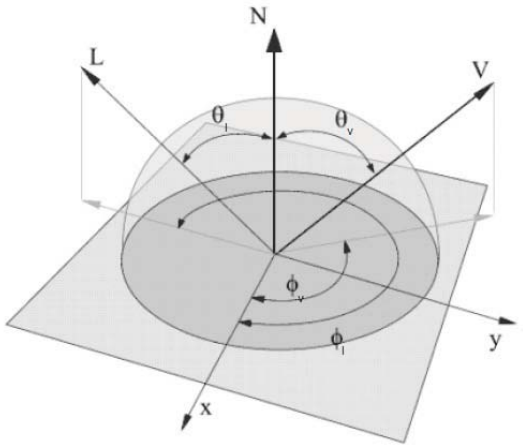

Abbildung 4.1: Die 4 Eingabevariablen einer BRDF: Richtung des Lichteinfalls (θ_l und ϕ_l) und reflektiertes Licht in Richtung des Betrachters (θ_v und ϕ_v) [Gebhardt 03].

Echtzeit zu verwenden, was in Abschnitt 4.3 eingehend erläutert werden soll.

4.2 BRDF und BTF

4.2.1 BRDF und relevante radiometrische Größen

Bidirektionale Reflektanzverteilungsfunktionen bilden eine Klasse von vierdimensionalen Funktionen, die es ermöglichen, die Reflexionseigenschaften von verschiedenen Materialien für Licht einer beliebig, aber festen Wellenlänge zu modellieren und wurden erstmals von [Nicodemus 70] beschrieben. Hierbei dienen der Richtungsvektor $\mathbf{l}$ des eingehenden Lichtstrahls und der Richtungsvektor $\mathbf{v}$ des reflektierten Lichtstrahls in Richtung des Beobachters als Eingabe für die Funktion

$$f(\mathbf{l}, \mathbf{v}) = f(\theta_l, \phi_l, \theta_v, \phi_v) \tag{4.1}$$

wobei die beiden Vektoren in Form ihrer sphärischen Koordinaten mit den Polarwinkeln θ_l bzw. θ_v und den Azimutwinkeln ϕ_l bzw. ϕ_v repräsentiert werden [Gebhardt 03] (vgl. Abbildung 4.1). Präzise definiert wird eine BRDF für eine gegebene Wellenlänge über das Verhält-

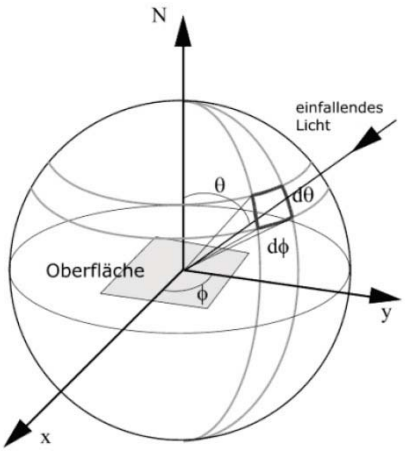

Abbildung 4.2: Der differentielle Raumwinkel entspricht der fett umrahmten Fläche auf der Einheitskugel [Gebhardt 03].

nis von ausgehender differentieller Leuchtdichte dL_v in Betrachtungsrichtung zu eingehender differentieller Beleuchtungsstärke dE_l, die aus der Beleuchtungsrichtung auf die Oberfläche trifft:

$$f(\mathbf{l},\, \mathbf{v}) = \frac{dL_v(\theta_v,\, \phi_v)}{dE_l(\theta_l,\, \phi_l)} \tag{4.2}$$

Gleichung 4.2 lässt sich anschaulich beschreiben, indem man sich vorstellt, dass eine Oberfläche von Licht aus einer kleinen Menge von Richtungen um $\mathbf{l}$ herum beleuchtet wird, was auf der Oberfläche als differentielle Beleuchtungsstärke dE_l gemessen wird. Das Licht wird dann von der Oberfläche in verschiedene Richtungen reflektiert, wobei für jede gegebene Reflexionsrichtung $\mathbf{v}$ die differentielle Leuchtdichte dL_v proportional zur differentiellen Beleuchtungsstärke dE_l ist und dieses – von l und v abhängige Verhältnis – genau der BRDF entspricht [Möller 08]. Die differentiellen Raumwinkel tragen dabei der Tatsache Rechnung, dass die genaue Menge an Licht, die aus einer bestimmten Richtung kommt, nicht notwendigerweise bestimmt werden muss, weil Licht als eine Art Fluss durch den Raum gemessen wird (vgl. Abbildung 4.2). Für einen gegebenen Richtungsvektor in sphärischen Koordinaten (θ, ϕ) und eine kleine Winkelabweichung $(d\theta, d\phi)$ ist der resultierende differentielle Raumwinkel dann als $d\omega$ definiert, wobei der Raumwinkel

$$\omega = \frac{A}{r^2}[sr] \tag{4.3}$$

das Verhältnis einer Kugelfläche A zum Quadrat des Radius r der Kugel beschreibt und in der Einheit sr (Steradiant) angegeben wird. Der volle Raumwinkel erstreckt sich dann über eine Gesamtfläche der Einheitskugel in Höhe von $4\pi sr$ [Gebhardt 03, Wynn 00].

Die Leuchtdichte L als radiometrisch wichtigste Größe gibt demnach also an, wieviel Lichtenergie von einem bestimmten Punkt auf der Oberfläche ausgestrahlt wird. Die verwendete Einheit ist dabei

$$\frac{Watt}{m^2 sr} \tag{4.4}$$

Somit gehört die Leuchtdichte zu den sogenannten raumwinkelabhängigen Größen, die nicht abhängig von der Entfernung zur Lichtquelle sind und pro Einheitsraumwinkel und pro perspektivisch verkleinerter Einheitsfläche angegeben wird. Die Beleuchtungsstärke E beschreibt hingegen, wieviel Strahlungsleistung auf eine bestimmte Fläche auftrifft und gehört im Gegensatz zur Leuchtdichte zu den raumwinkelunabhängigen Größen, wobei

$$\frac{Watt}{m^2} \tag{4.5}$$

als Einheit fungiert und die Beleuchtungsstärke somit gemäß dem photometrischen Entfernungsgesetz mit dem Quadrat der Entfernung zur Strahlungsquelle abnimmt. Nach Gleichung 4.2 ist die Einheit einer BRDF somit sr^{-1}, was intuitiv einer relativen Menge an Energie entspricht, die für eine gegebene eingehende Richtung in ausgehender Richtung reflektiert wird [Gebhardt 03, Möller 08].

Da in dieser Arbeit ausschließlich vereinfachte Punktlichter oder Richtungslichter verwendet werden – also Lichtquellen, die Strahlen nicht aus einer Menge von Richtungen, sondern lediglich aus einer einzigen Richtung aussenden – kann Gleichung 4.2 auch in nicht-differentieller Form betrachtet werden:

$$f(\mathbf{l},\, \mathbf{v}) = \frac{L_v(\theta_v,\, \phi_v)}{E_l(\theta_l,\, \phi_l) \cdot \cos\theta_i} \tag{4.6}$$

Der Kosinus des Winkels θ_i zwischen der Oberflächennormalen $\mathbf{n}$ und dem eingehenden Lichtstrahl $\mathbf{l}$ (vgl. Abschnitt 2.5.2) konvertiert dabei die eingehende Beleuchtungsstärke E_l in einen Wert, der an einem Punkt der Oberfläche gemessen wird, indem die durch den

Einfallswinkel abhängige Flächenverminderung berücksichtigt wird, die durch die orthogonal zur eingehenden Lichtrichtung projizierten Einheitsfläche zustande kommt. Der Grund hierfür ist, dass ein Lichtstrahl, der mit dem Winkel θ bezüglich der Oberflächennormalen eingeht, auf eine Fläche trifft, die um den Faktor $\frac{1}{\cos\theta}$ größer ist, als ein zur Oberflächennormalen paralleler Strahl. Daher nimmt die Lichtintensität der beleuchteten Fläche um den Faktor $\cos\theta_i$ ab [Gebhardt 03].

4.2.2 Eigenschaften von BRDF

Helmholtz-Reziprozität

Es existieren zwei wesentliche, physikalische Eigenschaften, die allen Bidirektionalen Reflektanzverteilungsfunktionen zugrunde liegen. Zum einen die sogenannte Helmholtz-Reziprozität, welche besagt, dass die Ein- und Austrittswinkel eines Lichtstrahls vertauscht werden können, ohne den Wert der BRDF zu verändern:

$$f(\mathbf{l}, \mathbf{v}) = f(\mathbf{v}, \mathbf{l}) \tag{4.7}$$

In der Praxis verletzen allerdings viele Bidirektionale Reflektanzverteilungsfunktionen, die zum Rendern verwendet werden, diese Vorgabe, ohne wahrnehmbare Artefakte zu erzeugen. Die Helmholtz-Reziprozität kann aber dessen ungeachtet dazu verwendet werden, eine gegebene BRDF auf ihre physikalische Korrektheit hin zu überprüfen [Möller 08].

Energieerhaltung

Die zweite Eigenschaft ist die Energieerhaltung, welche besagt, dass die abgestrahlte Energie nicht größer sein kann als die Energie, die auf die Oberfläche auftrifft. Obwohl einige Algorithmen aus dem Bereich des Offline-Rendering[2], wie beispielsweise Raytracing[3], die Einhaltung

[2] Im Gegensatz zum Echtzeitrendering zeichnet sich das Offline-Rendering dadurch aus, dass sämtliche Berechnungen nicht zur Laufzeit durchgeführt werden, sondern in einer Vorberechnungsphase, wobei Performanz eine untergeordnete Rolle spielt [Möller 08].

[3] Raytracing (engl.: Strahlenverfolgung) ist eine Methode aus der Computergrafik, um eine 3D-Szene durch Aussendung von Strahlen von einem Punkt aus zu berechnen, wodurch sich verdeckende Objekte bestimmt und der Weg von Lichtstrahlen in der Szene simuliert werden kann. Raytracing ist ein äußerst rechen- und zeitintensives Verfahren, wird daher meist nur im Bereich des Offline-Renderings verwendet und führt zu außerordentlich realistischen Ergebnissen [Möller 08].

dieses Gesetzes verlangen, ist dies bei Echtzeitalgorithmen nicht unbedingt notwendig, obgleich eine annähernde Energieerhaltung wünschenswert bleibt. Eine mit Hilfe einer BRDF gerenderte Oberfläche, die dieses physikalische Gesetz zu sehr vernachlässigt, erscheint unter Umständen zu hell, was zu einer Verminderung des realistischen Eindrucks führen kann und vermieden werden sollte [Möller 08]. Formal ausgedrückt werden kann die Energieerhaltung mittels

$$\int_{\Omega} f(\mathbf{l}, \mathbf{v}) \cos\theta_v d\omega_v \leq 1 \qquad \forall \omega_v \in \Omega \tag{4.8}$$

wobei für einen gegebenen Lichteinfallsvektor $\mathbf{l}$ die Menge an Energie gemessen wird, die aufsummiert in alle möglichen Reflexionsrichtungen abgestrahlt wird und so der – z. B. durch Absorptionsvorgänge bedingte – auftretende Energieverlust festgestellt werden kann. Dazu wird über die gesamte Hemisphäre Ω – mit der Oberflächennormalen $\mathbf{n}$ als Zentrum – der möglichen Austrittwinkel eines Lichtstrahls integriert, wobei der Kosinus zwischen $\mathbf{n}$ und den Reflexionsvektoren $\mathbf{v}$ multipliziert mit der BRDF $f(\mathbf{l}, \mathbf{v})$ als Integrand fungiert [Daubert 03].

Vereinfachende Annahmen

Um die Handhabung einer BRDF zu vereinfachen und die Anzahl der Dimensionen auf vier beschränken zu können, werden einige Annahmen getroffen, die im Folgenden erwähnt werden sollen [Daubert 03, Gebhardt 03]:

- Da wie bereits eingangs erwähnt eine BRDF nur für eine feste, implizit gegebene Wellenlänge λ des Lichts gültig ist, wird weiterhin davon ausgegangen, dass das reflektierte Licht die gleiche Wellenlänge besitzt wie das eingehende Licht, was dazu führt, dass die BRDF keine fluoreszierenden Effekte einfangen kann.

- Des Weiteren wird angenommen, dass das Licht sofort reflektiert wird und dass insbesondere keine Energie gespeichert und verzögert abgegeben wird, weswegen kein Parameter existiert, der solchen temporalen Effekten Rechnung trägt. Jedwede Form von phosphoreszierenden Effekten kann daher von einer BRDF nicht wiedergegeben werden.

- Als dritte Annahme wird vorausgesetzt, dass das Licht, was an einem Punkt der Oberfläche auftrifft, von eben diesem Punkt auch wieder reflektiert wird. Ein Lichtstrahl

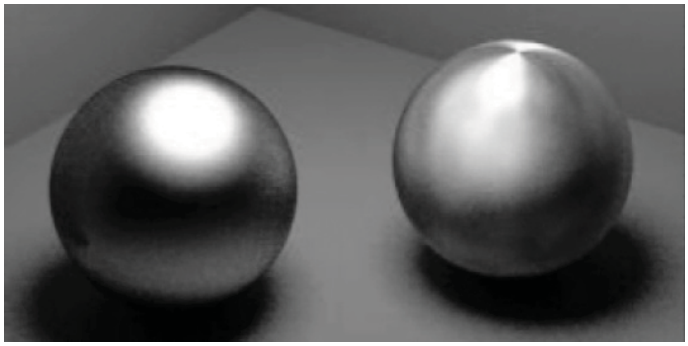

Abbildung 4.3: Links: Isotrope Oberfläche bei normalem Metall. Rechts: Anisotrope Oberfläche bei gebürstetem Metall. [Gebhardt 03]

kann also nicht unterhalb der Oberfläche – beispielsweise durch Subsurface Scattering[4] – abgelenkt werden und die Oberfläche an einem anderen Punkt wieder verlassen. Aus diesem Grund können von der BRDF weder atmosphärische Effekte noch Materialien wie Haut oder komplexe Farbschichten eingefangen werden.

- Schließlich wird davon ausgegangen, dass zwei Lichtstrahlen, die im gleichen Punkt auf der Oberfläche auftreffen, sich nicht gegenseitig beeinflussen. Die von verschiedenen Lichtstrahlen resultierenden Reflexionen haben also keine Auswirkung aufeinander und können linear überlagert werden, was als Superposition bezeichnet wird.

BRDF-Klassen

Bidirektionale Reflektanzverteilungsfunktionen können entsprechend dem isotropen Reflexionsverhalten der zu modellierenden Oberfläche prinzipiell in zwei Klassen unterteilt werden, wobei man isotrope und anisotrope Oberflächen unterscheidet. Eine isotrope Oberfläche reflektiert an einem Punkt auftreffendes Licht bei einer Drehung des Materials um den entsprechenden Normalenvektor immer mit derselben Lichtintensität, wohingegen die BRDF anisotroper Materialien – wie beispielsweise von Samt, Satin oder Kord – in Abhängigkeit der beiden

[4] Wenn Licht auf eine Oberfläche auftrifft und in das Material eindringt, kommt es neben Absorption, welche zur Abnahme der Lichtintensität – nicht jedoch einer Richtungsänderung – führt, bedingt durch Variationen in der Dichte und Struktur oder durch Fremdkörper zu einer Änderung der Ausbreitungsrichtung, was als Subsurface Scattering bezeichnet wird. Im Gegensatz zur Absorption findet hier keine Abschwächung der Lichtintensität statt [Möller 08].

Azimutwinkel ϕ_l und ϕ_v unterschiedliche Werte zurückliefert. Eine BRDF für isotrope Materialien – wie beispielsweise für glattes Leder – hängt demnach nicht von den Azimutwinkeln ab, sondern vielmehr von der Differenz zwischen beiden, was mittels

$$f_{isotrop}(\theta_l,\, \phi_l,\, \theta_v,\, \phi_v) = f_{isotrop}(\theta_l,\, 0,\, \theta_v,\, \phi_v - \phi_l) \tag{4.9}$$

ausgedrückt werden kann, wobei gleichbleibende Polarwinkel θ_l und θ_v vorausgesetzt werden. Die meisten Stoffmaterialien und auch die meisten anderen Materialien besitzen jedoch zumindest eine gewisse untergründige Anisotropie, was für den Fall von gebürstetem und ungebürstetem Metall und der bedingt durch die Anisotropie auftretenden Veränderung im Specular Highlight in Abbildung 4.3 gegenübergestellt wurde [Daubert 03, Garstenauer 06, Gebhardt 03].

4.2.3 BTF

Der Begriff der Bidirektionalen Texturfunktionen wurde erstmals von Dana et al. eingeführt, um ausgehend von einer BRDF, die vor allem das Erscheinungsbild von Materialien bei einer Betrachtung aus mittlerer oder hoher Entfernung akkurat wiedergibt, die Möglichkeit zu schaffen, auch bei sehr naher Betrachtung feine Details und lokale Variationen im Reflexionsverhalten einzufangen. Hierzu wird eine BRDF um zwei zusätzliche Dimensionen erweitert, so dass eine zweidimensionale Textur mit den Texturkoordinaten $(x,\, y)$ als Funktion der eingehenden Beleuchtungsrichtung $\mathbf{l}$ und der ausgehenden Betrachtungsrichtung $\mathbf{v}$ repräsentiert werden kann. Jeder Texel innerhalb der Textur wird also durch $\mathbf{l} = (\theta_l,\, \phi_l)$ und $\mathbf{v} = (\theta_v,\, \phi_v)$ parametrisiert, wodurch sich eine sechsdimensionale Funktion $f(x,\, y,\, \theta_l,\, \phi_l,\, \theta_v,\, \phi_v)$ ergibt [Dana 99, Irawan 08].

In der Praxis werden zur Gewinnung von BTF-Daten rechteckige, flache Materialproben aus unterschiedlichen Blickwinkeln bei variierendem Lichteinfall fotografiert und in einer Datenbank gespeichert (vgl. Abschnitt 4.2.4). Auf diesem Weg ist es möglich zahlreiche Effekte und Eigenschaften wie z. B. Rauhigkeit, Selbstabschattung, Verdeckung, Interreflexionen[5], Volumenstreuung bzw. Subsurface Scattering und sogenanntes Color Bleeding[6] zu erfassen,

[5] Interreflexionen kommen zustande, wenn angestrahlte Objekte Licht reflektieren und dadurch wiederrum Objekte in ihrer näheren Umgebung beleuchten [Möller 08].

[6] Als Color Bleeding wird in der Computergrafik der Effekt bei Interreflexionen bezeichnet, bei dem von Objekten reflektiertes, farbiges Licht an in der Nähe befindliche Objekte weitergegeben wird und diese nicht nur erhellt, sondern zusätzlich auch einfärbt [Möller 08].

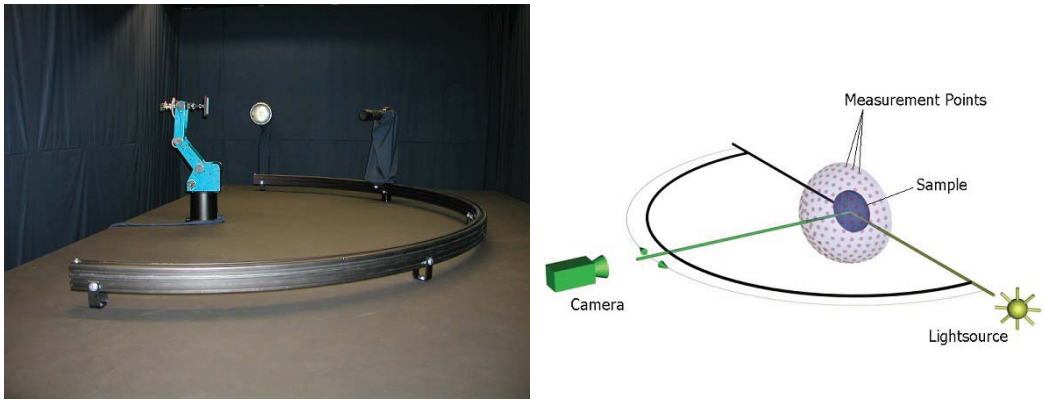

Abbildung 4.4: Links: Labor der Uni Bonn zur Messung von BTF-Daten bestehend aus einer HMI-Lampe, einer CCD-Kamera und einem Roboterarm mit einer Halterung für die Materialprobe [Hauth 02].
Rechts: Schematische Darstellung des Aufbaus bei der Messung an der Uni Bonn inkl. roter Markierungspunkte für Messpositionen [Hauth 02].

die bedingt durch die Mesostruktur des Materials auftreten und sich durch analytische Modelle nur mit großem rechnerischen Aufwand nachbilden lassen würden. Auf der anderen Seite haben analytische Modelle den Vorteil, dass sich künstliche, nicht existierende Stoffmaterialien modellieren und rendern lassen, wobei entsprechende Algorithmen in der Vergangenheit hauptsächlich auf Strickwaren beschränkt und nicht in der Lage waren, bestimmte Materialien wie beispielsweise Kord zu reproduzieren [Sattler 03].

4.2.4 Messung von BTF-Daten

Um einen Eindruck zu erhalten, wie der Ablauf bei der Messung von BTF-Daten beliebiger Stoffmaterialien aussieht, soll an dieser Stelle der Aufbau des Kamerasystems der Rheinischen Friedrich-Wilhelms-Universität Bonn und der dazugehörigen Apparaturen kurz erläutert werden, da für die im Rahmen dieser Diplomarbeit entwickelten Methode (vgl. Abschnitt 4.3) ausschließlich Texturen der BTF-Datenbank Bonn [BTF-DB Bonn] verwendet wurden. Die in der Datenbank befindlichen Aufnahmen diverser Materialien wurden ursprünglich im Laufe der Arbeit von [Hauth 02] erstellt und anschließend frei verfügbar gemacht. Dabei wurden maximal 10×10 cm große Materialproben an einer Halterung eines Roboterarms befestigt, von einer HMI-Lampe (Abk.: Hydrargyrum medium-arc iodide) beleuchtet und von einer auf

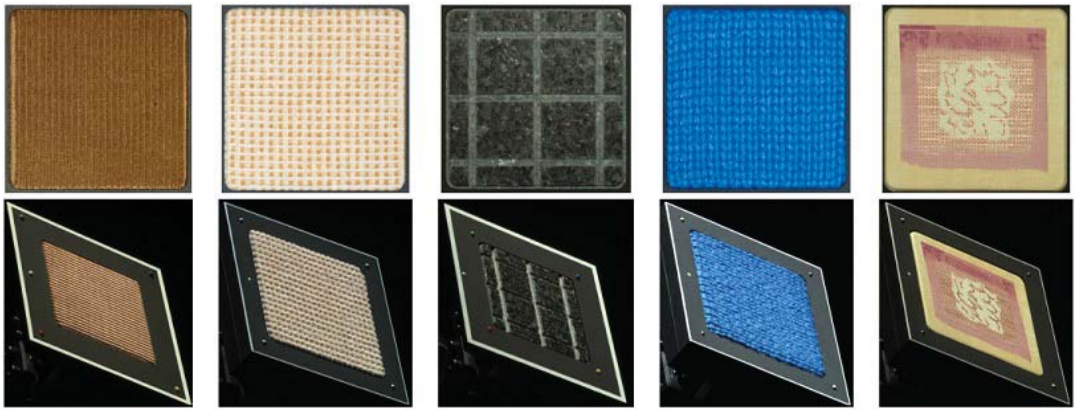

Abbildung 4.5: Oben: Einige der Materialien in der BTF-Datenbank Bonn (v. l. n. r.): Kord, Polster, Granit, Wolle und Tapete [Müller 05].
Unten: Veränderte Mesostruktur und Farbe bei perspektivischem Betrachtungswinkel ($\theta_v = 60°$, $\phi_v = 144°$) auf die Materialprobe und veränderter Beleuchtungsrichtung ($\theta_l = 60°$, $\phi_l = 18°$) [Müller 05].

einer Schiene beweglichen CCD-Kamera fotografiert, wobei alle Teile von einem PC gesteuert werden (vgl. Abbildung 4.4). Trotz der Beschränkung der Größe der Materialprobe konnten zahlreiche Materialien aufgenommen werden, von denen einige in Abbildung 4.5 dargestellt sind [Müller 05].

Die Anzahl der Positionen für die Kamera und die Lichtquelle beläuft sich auf jeweils 81, so dass für jede Materialprobe insgesamt $81 \times 81 = 6561$ Fotos aufgenommen wurden, von denen jedes eine Auflösung von 4500×3000 Pixel bei 12 Bit Farbtiefe und einer verlustfreien Komprimierung[7] besitzt. Nach einer Bearbeitung der Fotos, um die umgebende Halterung mit Hilfe der angebrachten Begrenzungsmarkierungen (vgl. untere Reihe in Abbildung 4.5) aus den Bilddaten eliminieren zu können und die perspektivischen Aufnahmen auf eine normierte Frontalansicht mit $\theta_v = 0°$ und $\phi_v = 0°$ zu projizieren, findet sowohl eine Konvertierung der Bilddaten auf 8 Bit pro Farbkomponente, als auch eine Skalierung auf eine Auflösung von 256×256 Pixel statt. Die nun vorliegenden 6561 Fotos haben eine Gesamtdatenmenge von ca. 1,2 GB pro Materialprobe und werden im JPEG-Format in die Datenbank aufgenommen, wobei der gesamte Vorgang inklusive Messung, Bearbeitung und Speicherung ca. 14 Stunden pro Material in Anspruch nimmt [Müller 05].

[7] Kodak DCR 12-bit RGB-Format

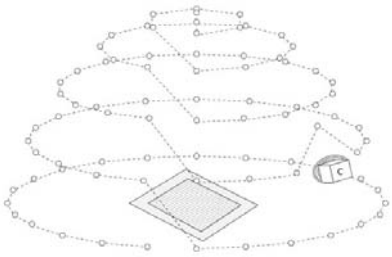

Abbildung 4.6: Schematische Darstellung des Verlaufs der Lichtquelle bei der Messung einer Probe [Filip 05].

4.2.5 Aufbau der BTF-Datenbank Bonn

Für jede der 81 möglichen Kamerapositionen wurden die 81 Positionen der Lichtquelle in Form ihrer sphärischen Koordinaten wiefolgt definiert, so dass sich eine Art Flugbahn der Lichtquelle – bestimmt durch die einzelnen Wegpunkte – ergibt, die in Abbildung 4.6 schematisch dargestellt wurde [BTF-DB Bonn]:

θ_l	ϕ_l	**Anzahl der Messungen pro Polarwinkel θ_l**
0°	0°	1x
15°	0°, 60°, ..., 300°	6x
30°	0°, 30°, ..., 330°	12x
45°	0°, 20°, ..., 340°	18x
60°	0°, 18°, ..., 342°	20x
75°	0°, 15°, ..., 345°	24x

Innerhalb der Datenbank sind die Fotos in gleicher Weise sortiert, so dass in insgesamt 81 Verzeichnissen, welche die Kamerapositionen darstellen, jeweils 81 Dateien für die möglichen Positionen der Lichtquelle enthalten sind. Im Dateinamen sind außerdem die sphärischen Koordinaten für die Position des Lichts und der Kamera zusammen mit einem fortlaufenden Index des betreffenden Fotos kodiert, wobei die Koordinaten für eine weitere Verwendung durch die im Rahmen dieser Arbeit entwickelten Methode zur effizienten Komprimierung der BTF-Daten noch in kartesische Koordinaten umgerechnet werden müssen, worauf in Abschnitt 4.3 eingegangen wird. Der Dateiname

nnnnn **tl***aaa* **pl***bbb* **tv***ccc* **pv***ddd***.jpg**

setzt sich dabei wiefolgt zusammen:

Variable	**Entsprechung**	**Bereich**
nnnnn	Index	00000 - 06560
aaa	θ_l	000 - 345
bbb	ϕ_l	000 - 345
ccc	θ_v	000 - 345
ddd	ϕ_v	000 - 345

Der Maximalwert von nur 345° für den Bereich der Theta- und Phi-Werte ergibt sich hierbei durch die Definition der letzte Stufe der Flugbahn ($\theta_l = 75°$) in 15°-Schritten [BTF-DB Bonn].

4.2.6 Modellierung von BTF

Um die großen Datenmengen, die bei der Messung von BTF-Daten entstehen und die je nach der gewählten Methode im Bereich von mehreren hundert Megabytes bis hin zu mehreren Gigabytes liegen, weiterverarbeiten und rendern zu können, ist eine geeignete Repräsentation und Modellierung der Daten nötig. Eine solche Methode sollte eine kompakte, parametrische Repräsentation bieten und die charakteristischen, visuellen Eigenschaften der Stoffmaterialien möglichst beibehalten, wobei im Fall des Echtzeitrenderings zusätzlich die Möglichkeiten moderner Grafik-Hardware ausgenutzt werden sollten [Filip 05].

Analytische Verfahren

Die Methoden lassen sich dabei prinzipiell in zwei Gruppen aufteilen, wobei die Methoden der ersten Gruppe eine BTF durch eine texelweise Repräsentation einer analytischen BRDF modellieren und die Methoden der zweiten Gruppe Verfahren aus der multivariaten Statistik[8] einsetzen, um die Eingangssignale in Form der BTF-Daten geeignet interpretieren und in einem Modell abbilden zu können. In den Arbeiten von [McAllister 02] und [Daubert 01],

[8] In der multivariaten Statistik werden mehrere statistische Variablen gleichzeitig analysiert, wodurch Zusammenhänge und Abhängigkeiten zwischen den Variablen erkannt und behandelt werden können [Elpelt 07].

die zur ersten Gruppe zu zählen sind, wird hierzu eine räumlich variierende BRDF für jeden Texel mittels des Lafortune Reflexionsmodells [Lafortune 97] repräsentiert, indem in einer zweidimensionalen Textur verschiedene Parameter des Lafortune-Modells gespeichert werden. Wegen der in den BTF-Daten auftretenden räumlichen Inkonsistenz einzelner Texel bei verschiedenen Betrachtungsrichtungen wurde dabei eine separate Modellierung der einzelnen Betrachtungsrichtungen durchgeführt. Ein anderer Ansatz von [Malzbender 01] verwendet sogenannte polynomielle Texturen, in denen die Koeffizienten eines biquadratischen Polynoms[9] pro Texel gespeichert werden, um so die Farbe der Oberfläche an einem Punkt bei variierendem Lichteinfall rekonstruieren zu können. Hierbei wird die Richtung des reflektierten Lichtstrahls konstant gehalten und an die Oberflächennormale angenähert, wodurch zwei Dimensionen eingespart werden können und lediglich die eingehenden Lichtstrahlen variiert werden [Daubert 01, Filip 05, McAllister 02, Sattler 03].

Da die Idee in den Arbeiten von [McAllister 02] und [Daubert 01], die Parameter eines geeigneten Reflexionsmodells so zu wählen, dass die resultierende Pixelfarbe mit möglichst hoher Genauigkeit den BTF-Daten entspricht, der im Rahmen dieser Arbeit entwickelten Methode bezüglich der Schätzung der Parameter ähnelt, soll im Folgenden ein kurzer Einblick in die Vorgehensweise von [McAllister 02] gegeben und ein Vergleich zur in Abschnitt 4.3 vorgestellten Methode gezogen werden. Das von [McAllister 02] verwendete Lafortune-Modell ist eine generalisierte Form des sogenannten cosine-lobe-Modells, was wiederum eine Erweiterung des klassischen Phong-Modells um Eigenschaften der Energieerhaltung und der Helmholtz-Reziprozität darstellt (vgl. Abschnitt 4.2.2). Die Repräsentation nach Lafortune eignet sich, um beliebige BRDF wiederzugeben, wohingegen mit dem Phong-Modell insbesondere glatte Oberflächen mit Specular Highlights in Reflexionsrichtung der eingehenden Lichtstrahlen produziert werden können und daher eine Modellierung komplexer Stoffe nicht möglich ist. Wie beim cosine-lobe-Modell werden auch bei Lafortune verschiedene Loben (lat.: Lappen) aufsummiert, die jeweils verschiedenen Anteilen des Lichts entsprechen, um in der Summe den Wert einer BRDF annehmen zu können, mit der verschiedene Effekte dargestellt werden können, zu denen beispielsweise auch der in Abschnitt 2.5.4 erwähnte Fresnel-Effekt oder die sogenannte Retroreflexion[10] zählen [Gebhardt 03, McAllister 02].

[9] Ein biquadratisches Polynom ist ein Polynom 4. Grades der Form $ax^4 + bx^3 + cx^2 + dx + e$, wobei meist der Spezialfall für $b = d = 0$ und $a \neq 0$ referenziert wird [Rapp 10].

[10] Bei einer Retroreflexion wird Licht unabhängig von der Ausrichtung des Reflektors insbesondere in Richtung der Lichtquelle zurückgesendet, was z. B. auf Landstraßen bei den am Straßenrand angebrachten Leitpfosten oder bei der Herstellung von Warnschutz-Westen Verwendung findet, wobei in beiden Fällen retroreflektie-

Die Repräsentation nach Lafortune setzt sich aus einer Summe verschiedener Terme zusammen

$$f(\mathbf{l},\, \mathbf{v}) = \rho_d + \sum_j \rho_{s,j} \cdot s_j(\mathbf{l},\, \mathbf{v}) \tag{4.10}$$

wobei ρ_d die diffuse Albedo – also das von der Blickrichtung unabhängige, diffuse Reflexionsvermögen – des Lobus darstellt und die Terme innerhalb der Summe den j Loben entsprechen, die gemeinsam den von der Blickrichtung abhängigen Specular-Anteil der BRDF bilden. Hierbei besitzt jeder der j Loben eine individuelle Albedo $\rho_{s,j}$ und eine Form s_j, die als generalisierter Phong-Lobus wiefolgt definiert wird:

$$s_j(\mathbf{l},\, \mathbf{v}) = \left(\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}^T \cdot \begin{pmatrix} c_{x,j} & 0 & 0 \\ 0 & c_{y,j} & 0 \\ 0 & 0 & c_{z,j} \end{pmatrix} \cdot \begin{pmatrix} l_x \\ l_y \\ l_z \end{pmatrix} \right)^n = (v_x c_{x,j} l_x + v_y c_{y,j} l_y + v_z c_{z,j} l_z)^n \tag{4.11}$$

Der Exponent n charakterisiert – wie schon in Abschnitt 2.5.3 beschrieben – die Rauhigkeit der Oberfläche und reguliert somit die Größe des Specular Highlights. Mit Hilfe der drei Parameter c_x, c_y und c_z lassen sich verschiedene Effekte und Oberflächeneigenschaften nachbilden, wobei man das originale cosine-lobe-Model, welches das Phong-Modell beinhaltet, mit $-c_x = -c_y = c_z = \sqrt[n]{\frac{n+2}{2\pi}}$ erhält. Die erhöhte Reflektivität des Fresnel-Effekts bei flachen Betrachtungswinkeln kann durch $c_x = c_y > c_z$ erreicht werden, die angesprochene Retroreflexion mittels $c_x > 0$ und $c_y > 0$. Die in Abschnitt 4.2.2 beschriebene Anisotropie lässt sich modellieren, indem c_x und c_y unterschiedliche Werte zugewiesen bekommen, wohingegen isotrope Oberflächen mit $c_x = c_y$ beschrieben werden können [Gebhardt 03, McAllister 02].

McAllister et al. bestimmen die – als Wurzel der gemittelten Quadrate am besten passendsten – Werte der Lafortune-Parameter, indem die korrespondierenden Reflektanzwerte der von ihnen aufgenommenen BTF-Daten jedes zu bestimmenden Texels als Eingabe für das nichtlineare Optimierungsverfahren nach Levenberg-Marquardt verwendet werden. Sie können so die ursprüngliche Datenmenge von ca. 8 Gigabytes auf weniger als 10 Megabytes reduzieren. Die zu schätzenden Lafortune-Parameter sind zum einen die diffuse Albedo ρ_d, die als

rende Elemente in die Materialoberfläche eingearbeitet werden, die das von beispielsweise Autoscheinwerfern ausgesendete Licht möglichst direkt zum Fahrer zurückwerfen sollen [Möller 08].

RGB-Wert drei Komponenten besitzt, und zum anderen pro Lobus je sechs Werte, bestehend aus der Albedo $\rho_{s,j}$ und den drei Parametern c_x, c_y und c_z. Im Gegensatz zur im Rahmen dieser Diplomarbeit entwickelten Methode, die anhand der BTF-Daten für jeden Texel einen individuellen Normalenvektor und einen ambienten Restwert mit Hilfe eines linearen Optimierungsverfahrens berechnet, wird in der Arbeit von McAllister et al. beim Rendern der Pixel auf die interpolierte, pro Vertex berechnete Oberflächennormale der zugrundeliegenden Geometrie zurückgegriffen, wodurch sich benachbarte Polygone bei einer zu groben Auflösung der 3D-Modelle farblich sehr voneinander unterscheiden können. Des Weiteren ergeben sich laut McAllister et al. Rendering-Artefakte, falls benachbarte Pixel stark unterschiedliche Lafortune-Parameter c_x, c_y und c_z aufweisen, was durch die Verwendung von Normal Mapping ebenfalls vermieden werden könnte und was bei der hier entwickelten Methode wie bereits zu Beginn von Kapitel 4 erwähnt zum Einsatz kommt. Außerdem werden bei McAllister et al. – aufgrund der zu dieser Zeit noch sehr eingeschränkten Möglichkeiten und Rechenkapazitäten der Grafik-Hardware – sämtliche Beleuchtungsberechnungen in einem Vertex Shader durchgeführt, was wie in Abschnitt 2.5 dargelegt zu visuellen Defiziten führt. In ähnlicher Weise tragen die von ihnen verwendeten, vorberechneten Lookup-Tabellen, die aufgrund des – in Echtzeit nicht praktikablen aber wegen Gleichung 4.11 erforderlichen – Potenzierens nötig sind, dazu bei, dass es unweigerlich zu Ungenauigkeiten in der Darstellung zugunsten besserer Performance kommt [McAllister 02].

Statistische Verfahren

Die Verwendung von analytischen BRDF-Modellen zur Repräsentation der einzelnen Texel ist allerdings mit einigen Einschränkungen verbunden, da sie eigentlich entworfen wurden, um je eine bestimmte Klasse von Materialien wiederzugeben und im Wesentlichen nur eine Approximation realer Oberflächenreflexionen sind, die vereinfachte Annahmen über die zugrundeliegenden physikalischen Prozesse treffen. Außerdem besteht die schon in Abschnitt 4.2.3 angesprochene Schwierigkeit vor allem darin, die verschiedenen durch die Mesostruktur auftretenden Effekte mittels dieser Modelle akkurat einzufangen. Daher werden in der zweiten Gruppe der Methoden die gemessenen Daten als multidimensionale Signale interpretiert, so dass allgemeine Signalverarbeitungstechniken wie die Hauptkomponentenanalyse bzw. PCA (engl.: Principal Component Analysis) angewendet werden können. Bei der Hauptkomponentenanalyse wird die Varianz im Restsignal minimiert und eine – der in Abschnitt 4.3 vor-

gestellten Methode der kleinsten Quadrate ähnlichen – Approximierung des Eingangssignals ermittelt. Sie wird daher in einigen Bildkompressionsverfahren zur Reduktion der Bilddaten eingesetzt und so z. B. in der Arbeit von [Nishino 99] dazu verwendet, ein 3D-Modell anhand von zweidimensionalen Aufnahmen aus verschiedenen Perspektiven und unter verschiedenen Beleuchtungsbedingungen zu rekonstruieren, indem die aufgenommenen Bilddaten in zweidimensionalen Koordinatensystemen – definiert auf der Oberfläche des 3D-Modells – komprimiert werden [Müller 05].

Für die Modellierung von BTF-Daten wird in der Arbeit von [Sattler 03] beispielsweise für einen konstanten Blickwinkel der Kamera eine Hauptkomponentenanalyse der aufgenommenen Bilder bei variierenden Beleuchtungsrichtungen dieses jeweils fixen Blickwinkels durchgeführt. So kann der dreidimensionalen Mesostruktur der Materialien und dem sich dadurch stark verändernden Erscheinungbild der Oberfläche – aufgrund variierender Höhe am selben Oberflächenpunkt bei wechselnden Blickwinkeln – begegnet und eine Kohärenz der Texelpositionen gewährleistet werden. Das Ergebnis der Hauptkomponentenanalyse ist dann eine Zerlegung der für jeden Blickwinkel aufgenommenen Bilder in sogenannte Basistexturen, mit deren Hilfe die ursprünglichen Texturen wesentlich platzsparender repräsentiert und zur Laufzeit rekonstruiert werden können, wobei mit dieser Methode eine Kompression von 5:1 erreicht werden konnte [Sattler 03].

Ein anderer Ansatz von [Wong 03] komprimiert die BTF-Daten unter Zuhilfenahme sogenannter Spherical Harmonics bzw. Kugelflächenfunktionen, die eine Menge von orthonormalen Basisfunktionen darstellen, die gewichtet und aufsummiert dazu verwendet werden können, eine gewünschte Zielfunktion auf der Einheitskugel zu beschreiben. Die Gewichte bzw. Koeffizienten skalieren dabei die Basisfunktionen und approximieren in ihrer Summe die Zielfunktion, wobei im Fall von Wong et al. 25 Koeffizienten verwendet werden, um die BTF-Daten texelweise anzunähern. Obwohl die Projektion der BTF-Daten auf die Basisfunktionen recht einfach zu handhaben ist, da z. B. keinerlei Optimierungsprozesse involviert sind, ist die Verwendung in Echtzeitanwendungen nicht praktikabel, da eine relativ hohe Anzahl an Basisfunktionen nötig ist, um zufriedenstellende und realistische Ergebnisse zu erzielen und da vor allem niederfrequente Beleuchtunganteile reproduziert werden können, weswegen eine Anwendung auf die hochfrequenten BTF-Daten von Stoffen – welche vor allem die Mikrostruktur wiedergeben – ungeeignet ist. Daher werden Verfahren dieser Art vor allem in Anwendungen verwendet, die statische Beleuchtungsbedingungen beispielsweise mittels Precomputed

Radiance Transfer vorberechnen, mit deren Hilfe zwar sehr komplexe Beleuchtungsszenarien dargestellt werden können, eine vollständig dynamische Echtzeitumgebung aber nicht möglich ist [Filip 09, Möller 08, Wong 03].

4.3 Schätzung eines parametrischen Beleuchtungsmodells

4.3.1 Idee und Vorgehensweise

In diesem Abschnitt soll eine Methode vorgestellt werden, mit der es möglich ist, die charakteristischen Oberflächendetails verschiedener Stoffmaterialien aus der BTF-Datenbank Bonn durch die Schätzung eines parametrischen Beleuchtungsmodells zu extrahieren, um die Stoffe dann unter Zuhilfenahme der in Abschnitt 2.5.4 vorgestellten Rendering-Gleichung

$$P_{Color} = (I_A \cdot K_A + I_D \cdot K_D \cdot a) \otimes T_{Color} + I_S \cdot K_S \cdot a \tag{4.12}$$

in der im Rahmen dieser Arbeit entwickelten 3D-Echtzeitumgebung zu rendern. Beim Rendering-Prozess wird Gleichung 4.12 noch um den Fresnel-Effekt sowie Environment Mapping und Normal Mapping erweitert (vgl. Abschnitte 2.5.4 und 2.6), was für die Schätzung aber zunächst vernachlässigbar ist. Hier spielen vielmehr die ambiente Komponente I_A und insbesondere die diffuse Komponente $I_D = \max(\mathbf{n} \cdot \mathbf{l}, 0) \cdot L_{Diffuse}$ der Rendering-Gleichung eine zentrale Rolle, da die Schätzung des ambienten Anteils und des Normalenvektors $\mathbf{n}$ mit Hilfe der 81 – für jede feste Betrachtungsrichtung $\mathbf{v}$ zur Verfügung stehenden – Beleuchtungsrichtungen $\mathbf{l_i}$ für die frontale Betrachtungsrichtung mit $\theta_v = 0°$ und $\phi_v = 0°$ den Kern des Verfahrens bildet und im Folgenden eingehend erläutert und diskutiert werden soll. Die Schätzung des Normalenvektors und des ambienten Restanteils erfolgt dabei mittels eines mathematischen Optimierungsverfahrens namens „Methode der kleinsten Quadrate“ für jeden durch seine Texturkoordinaten (x, y) repräsentierten Texel der BTF-Aufnahmen und resultiert in einer Normal Map und einer Ambient Map, die im weiteren Verlauf zum Rendern der Stoffe verwendet werden können. Da der Specular-Anteil $I_S = \max(\mathbf{n} \cdot \mathbf{h}, 0)^n \cdot L_{Specular}$ (vgl. Abschnitt 2.5.3) einen exponentiellen Charakter besitzt und das in dieser Arbeit verwendete Lösungsverfahren für das Optimierungsproblem von einem linearen System von Gleichungen ausgeht (vgl. Abschnitt 4.3.2), wird der Specular-Anteil I_S im Laufe der Optimierung zunächst ignoriert und anhand der Ergebnisse gesondert konstruiert, worauf in Abschnitt 4.4.1 detailliert eingegangen werden soll.

Der Einfachheit halber werden die abstandsabhängige Attenuation a der Lichtintensität gemäß der Verwendung einer direktionalen Lichtquelle wie der Sonne, sowie die Reflexionskoeffizienten K_A und K_D für die ambienten und diffusen Anteile innerhalb der Rendering-Gleichung gleich 1 gesetzt, da sie für die Schätzung vernachlässigbar sind und später beim Echtzeit-Rendering ohnehin angepasst werden können. Die Reflexionskoeffizienten sind – wie schon in Abschnitt 2.5.4 erwähnt – empirischer Natur und werden dazu verwendet, das Reflexionsverhalten der Anteile nachträglich zu justieren, um eine möglichst authentische Reproduktion der ursprünglichen Stoffmaterialien zu erreichen. Innerhalb der diffusen Komponente I_D wird die Lichtfarbe $L_{Diffuse}$ ebenfalls gleich 1 gesetzt, da die für die Aufnahme der BTF-Samples verwendete HMI-Lampe mit einer Farbtemperatur von ca. 6000 K ungefähr dem Licht der Sonne an einem wolkenlosen Tag entspricht (ca. 5500 K) und somit eine korrekte Farbwiedergabe der Stoffe gewährleistet [Müller 05]. Nach der Vereinfachung wird die diffuse Komponente in die modifizierte Rendering-Gleichung

$$P_{Color} = (I_A + I_D) \otimes T_{Color} \tag{4.13}$$

eingesetzt, so dass sich die Gleichung

$$P_{Color} = (I_A + \max(\mathbf{n} \cdot \mathbf{l},\, 0) \cdot L_{Diffuse}) \otimes T_{Color} \tag{4.14}$$

ergibt, die im Folgenden die Grundlage für die Schätzung der Normal Map und der Ambient Map darstellt.

Das hier entwickelte Verfahren zur Ermittlung einer Normal Map, Ambient Map und Specular Map arbeitet in einem Vorverarbeitungsschritt auf den BTF-Daten und muss daher nicht zur Laufzeit der 3D-Echtzeitumgebung ausgeführt werden, wodurch sich für das Echtzeit-Rendering kein zusätzlicher Rechenaufwand ergibt. Die visuellen Ergebnisse unterscheiden sich qualitativ von der konventionellen Methode des Normal Mapping, bei der meist unabhängig von BTF-Daten eine diffuse Textur als Height Map interpretiert und die Normal Map durch Anwendung eines Sobel Filters erstellt wird [Peeper 04]. Eine Gegenüberstellung der Ergebnisse, sowie ein Vergleich zwischen der mittels der hier entwickelten Methode erzielten Resultate und den Original-BTF-Aufnahmen erfolgt zusammen mit einer Analyse der Vor- und Nachteile des Verfahrens abschließend in Abschnitt 4.4.

4.3.2 Methode der kleinsten Quadrate

Vorüberlegungen

Ziel des Verfahrens soll es also sein – mittels der 81 Kameraaufnahmen einer Stoffprobe bei systematisch variiertem Lichteinfall und konstantem Betrachtungswinkel – die Ausrichtung der Normalenvektoren und den Wert eines ambienten Restanteils für jeden Pixel so zu bestimmen, dass bei einer Berechnung der Pixelfarbe mit Hilfe von Gleichung 4.14 – und insbesondere unter Verwendung der geschätzten Normalenvektoren $\mathbf{n}$ und des ambienten Restanteils I_A – mit möglichst hoher Genauigkeit die durch die Kameraaufnahmen bereits bekannte Pixelfarbe P_{Color} zustande kommt. Pro Pixel werden demnach die optimalen Parameter für die vereinfachte Rendering-Gleichung gesucht, welche – bei Verwendung der Methode der kleinsten Quadrate – die Summe der quadratischen Abweichungen der so berechneten Pixelfarbe P^*_{Color} vom gemessenen Datenpunkt in Form der Pixelfarbe P_{Color} minimieren. In der Regel ist die Anzahl der vorgenommenen Messungen bedeutend größer als die Anzahl der zu bestimmenden Parameter, um den unvermeidbaren, zufälligen Messfehlern Rechnung zu tragen, was auch auf den vorliegenden Fall zutrifft, bei dem die 81 Messungen sechs gesuchten Parametern gegenüberstehen, da neben dem aus drei Koordinaten bestehenden Normalenvektor zusätzlich noch der aus drei Farbkomponenten zusammengesetzte ambiente Restanteil geschätzt werden soll [Schwarz 09]. Im Folgenden soll nun zunächst beschrieben werden, wie die Voraussetzungen für eine Anwendung der Minimierung der Fehlerquadrate aussehen, um dann schließlich die konkrete Problemstellung auf das Optimierungsverfahren übertragen zu können.

Allgemeiner Fall einer linearen Ausgleichsfunktion mit mehreren Variablen

Wenn man von einer Größe y ausgeht, die von mehreren voneinander unabhängigen Modellvariablen $a_1, \ldots, a_m$ beeinflusst wird, kann der Zusammenhang zwischen y und den Variablen über eine Modellfunktion f modelliert werden, wobei vorausgesetzt sei, dass f die Form eines mehrdimensionalen Polynoms erster Ordnung aufweist und neben $a_1, \ldots, a_m$ von m Funktionsparametern x_j abhängt:

$$y(a_1, \ldots, a_m) = f(a_1, \ldots, a_m; x_1, \ldots, x_m) = a_1 x_1 + \ldots + a_m x_m \tag{4.15}$$

Die Parameter x_j dienen dabei der Anpassung des gewählten Funktionstyps an die n gemessenen bzw. beobachteten Werte y_i und sollen nun so gewählt werden, dass die Modellfunktion

f die Messwerte möglichst gut approximiert. Nach Verteilungsannahmen, die von Carl Friedrich Gauß – auf den die Methode der kleinsten Quadrate maßgeblich zurückgeht – bezüglich der Messfehler angestellt wurden, sollten diese im Durchschnitt Null ergeben und von jedem anderen Messfehler stochastisch unabhängig sein, was zur Folge hat, dass nur zufällige und keine systematischen Informationen in den Messfehlern enthalten sind. Um diese Eigenschaften und die zusätzlich von Gauß angenommene Normalverteilung der Messfehler zu erreichen, ist es notwendig, dass wesentlich mehr Messwerte als gesuchte Parameter vorhanden sind, was wie bereits erwähnt im Fall der vorliegenden Arbeit erfüllt ist [Hering 04, Schwarz 09].

Für die n vorliegenden Messungen y_i lassen sich nun n Gleichungen aufstellen, die im linearen Gleichungssystem

$$\begin{aligned}
a_{11}x_1 + \ldots + a_{1j}x_j + \ldots + a_{1m}x_m &= y_1 \\
a_{21}x_1 + \ldots + a_{2j}x_j + \ldots + a_{2m}x_m &= y_2 \\
&\vdots \\
a_{i1}x_1 + \ldots + a_{ij}x_j + \ldots + a_{im}x_m &= y_i \\
&\vdots \\
a_{n1}x_1 + \ldots + a_{nj}x_j + \ldots + a_{nm}x_m &= y_n
\end{aligned} \tag{4.16}$$

vereint werden können. Das lineare Gleichungssystem kann jetzt in eine Matrixform überführt werden, indem die Koeffizienten a_{ij} zur Matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, die gesuchten Parameter x_j zum Vektor $\mathbf{x} \in \mathbb{R}^m$ und die gemessenen Beobachtungen y_i zum Vektor $\mathbf{b} \in \mathbb{R}^n$ zusammengefasst werden:

$$\mathbf{Ax} = \begin{pmatrix} a_{11} & \ldots & a_{1j} & \ldots & a_{1m} \\ a_{21} & \ldots & a_{2j} & \ldots & a_{2m} \\ & & \vdots & & \\ a_{i1} & \ldots & a_{ij} & \ldots & a_{im} \\ & & \vdots & & \\ a_{n1} & \ldots & a_{nj} & \ldots & a_{nm} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} = \mathbf{b} \tag{4.17}$$

Um große Abweichungen der Modellfunktion von den gemessenen Daten stärker zu bestrafen

als kleine, wird der Begriff des Residuums $\mathbf{r} \in \mathbb{R}^n$ eingeführt, wodurch ein sogenanntes System von Fehlergleichungen

$$\mathbf{Ax} - \mathbf{b} = \mathbf{r} \tag{4.18}$$

entsteht, deren Komponenten wiefolgt dargestellt werden können

$$\sum_{j=1}^{m} a_{ij}x_j - y_i = r_i, \quad i = 1, 2, \ldots, n, \quad m < n \tag{4.19}$$

und die in quadrierter Form der zu minimierenden Summe der Fehlerquadrate entsprechen. Gemäß dem Gaußschen Ausgleichsprinzip werden dann die Parameter x_j der Fehlergleichungen so bestimmt, dass die Summe der Quadrate der Residuen r_i minimal ist, was äquivalent zur Forderung ist, das Quadrat der euklidischen Norm des Residuenvektors zu minimieren [Schwarz 09]. Die Lösung des Minimierungsproblems

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2 \tag{4.20}$$

erfolgt nach einer Übertragung auf die konkrete Problemstellung und einem Einblick in die praktische Umsetzung im nächsten Abschnitt.

4.3.3 Lösung des Minimierungsproblems

Übertragung auf die Problemstellung

Um die soeben vorgestellte Methode auf das vorliegende Problem anzuwenden, wird zunächst Gleichung 4.14 umgeformt, wobei insbesondere die drei Farbkomponenten Rot, Grün und Blau der Pixelfarbe $P_{Color} = (p_r, p_g, p_b)^T$, des ambienten Restanteils $I_A = (a_r, a_g, a_b)^T$ und der Texturfarbe $T_{Color} = (t_r, t_g, t_b)^T$ und die drei Koordinaten des Lichtvektors $\mathbf{l} = (l_x, l_y, l_z)^T$ sowie des Normalenvektors $\mathbf{n} = (n_x, n_y, n_z)^T$ von Interesse sind. Der Maximum-Operator $\max(\mathbf{n} \cdot \mathbf{l}, 0)$ innerhalb der diffusen Komponente I_D kann an dieser Stelle weggelassen werden, da er beim Rendering dazu verwendet wird, nur einfallende Lichtstrahlen innerhalb der Hemisphäre über einem Punkt der Oberfläche zu beachten, wobei die Oberflächennormale das Zentrum dieser Hemisphäre darstellt. Für den Optimierungsprozess, bei dem die Minimierung der Summe der Fehlerquadrate aller Messungen im Vordergrund steht, und bei dem die Oberflächennormale der gesuchte Wert für gegebene, einfallende, systematisch variierte

Lichtstrahlen ist, können auch theoretisch mögliche, negative Ergebnisse für das Punktprodukt $\mathbf{n} \cdot \mathbf{l}$ zugelassen werden, solange die Summe aller Fehlerquadrate insgesamt minimal wird. Die Umformungen sehen dann wiefolgt aus:

$$\begin{aligned} P_{Color} &= (I_A + \max(\mathbf{n} \cdot \mathbf{l},\, 0) \cdot L_{Diffuse}) \otimes T_{Color} \\ P_{Color} &= I_A \otimes T_{Color} + ((\mathbf{n} \cdot \mathbf{l}) \cdot L_{Diffuse}) \otimes T_{Color} \\ \begin{pmatrix} p_r \\ p_g \\ p_b \end{pmatrix} &= \begin{pmatrix} a_r \\ a_g \\ a_b \end{pmatrix} \otimes \begin{pmatrix} t_r \\ t_g \\ t_b \end{pmatrix} + \left(\left(\left(\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \cdot \begin{pmatrix} l_x \\ l_y \\ l_z \end{pmatrix} \right) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) \otimes \begin{pmatrix} t_r \\ t_g \\ t_b \end{pmatrix} \right. \end{aligned} \quad (4.21)$$

Aus jeder solchen angepassten Rendering-Gleichung, die für jede der n Messwerte $P_{Color,i}$ aufgestellt wurde, lassen sich im Anschluss drei Gleichungen der Form

$$\begin{aligned} l_{x,i} \cdot t_r \cdot n_x + l_{y,i} \cdot t_r \cdot n_y + l_{z,i} \cdot t_r \cdot n_z + t_r \cdot a_r + 0 + 0 &= p_{r,i} \\ l_{x,i} \cdot t_g \cdot n_x + l_{y,i} \cdot t_g \cdot n_y + l_{z,i} \cdot t_g \cdot n_z + 0 + t_g \cdot a_g + 0 &= p_{g,i} \\ l_{x,i} \cdot t_b \cdot n_x + l_{y,i} \cdot t_b \cdot n_y + l_{z,i} \cdot t_b \cdot n_z + 0 + 0 + t_b \cdot a_b &= p_{b,i} \end{aligned} \quad (4.22)$$

ableiten, so dass jeweils nur eine der drei Farbkomponenten Rot, Grün und Blau aus $P_{Color,i}$, I_A und T_{Color} in einer dieser Gleichungen auftaucht. Da $n = 81$ ist, erhält man somit insgesamt 243 Gleichungen für sämtliche vorgenommenen Messungen.

Die gegebenen Werte sind an dieser Stelle die aus den Messungen gewonnene Pixelfarbe $P_{Color,i}$, die diffuse Texturfarbe T_{Color}, die für alle 243 Gleichungen konstant ist und auf deren Berechnung in Kürze eingegangen wird, und der Lichteinfall $\mathbf{l_i}$. Die sechs zu schätzenden Parameter x_j werden als Vektor $\mathbf{x} = (n_x,\, n_y,\, n_z,\, a_r,\, a_g,\, a_b)^T \in \mathbb{R}^6$ zusammengefasst, so dass dann im Anschluss die Überführung der im Gleichungssystem 4.22 festgehaltenen Gleichungen in die Matrixform erfolgen kann:

$$\mathbf{Ax} = \begin{pmatrix} & & \vdots & & & \\ l_{x,i} \cdot t_r & l_{y,i} \cdot t_r & l_{z,i} \cdot t_r & t_r & 0 & 0 \\ l_{x,i} \cdot t_g & l_{y,i} \cdot t_g & l_{z,i} \cdot t_g & 0 & t_g & 0 \\ l_{x,i} \cdot t_b & l_{y,i} \cdot t_b & l_{z,i} \cdot t_b & 0 & 0 & t_b \\ & & \vdots & & & \end{pmatrix} \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \\ a_r \\ a_g \\ a_b \end{pmatrix} = \begin{pmatrix} \vdots \\ p_{r,i} \\ p_{g,i} \\ p_{b,i} \\ \vdots \end{pmatrix} = \mathbf{b} \tag{4.23}$$

Man spricht in diesem Fall auch von einem überbestimmten System von linearen Gleichungen, da $n > m$ ist, also mehr Messwerte als gesuchte Parameter vorhanden sind [Schwarz 09]. Der Übersicht halber soll es an dieser Stelle genügen, einen 3×6-Ausschnitt der 243×6-Matrix $\mathbf{A} \in \mathbb{R}^{243 \times 6}$ und ebenso nur einen 3×1-Ausschnitt des 243×1-Vektors $\mathbf{b} \in \mathbb{R}^{243}$ zu zeigen, welcher einer beliebigen Messung i der $n = 81$ Messungen entspricht.

Lösungsverfahren

Die Lösung des linearen Gleichungssystems

$$\mathbf{Ax} = \mathbf{b} \tag{4.24}$$

ist demnach der gesuchte Parametervektor $\mathbf{x}$ mit $\mathbf{x} = \mathbf{A^{-1}b}$. Da die Matrix $\mathbf{A}$ aber nicht regulär[11], und somit auch nicht invertierbar ist, müssen zur Bestimmung der Lösung zunächst einige Umformungsschritte durchgeführt werden, wobei vorausgesetzt sei, dass $\mathbf{A}$ den Maximalrang $m = 6$ besitzt, d. h. dass ihre Spalten linear unabhängig sind. Hierzu wird die transponierte Matrix[12] $\mathbf{A^T} \in \mathbb{R}^{6 \times 243}$ gebildet und mit beiden Seiten des Gleichungssystems multipliziert, was zu den sogenannten Normalengleichungen

$$\mathbf{A^T Ax} = \mathbf{A^T b} \tag{4.25}$$

führt. Die so entstandene quadratische Matrix $\mathbf{A^T A} \in \mathbb{R}^{6 \times 6}$ ist regulär und kann nun

[11] Eine quadratische Matrix heißt regulär, wenn ihre Determinante einen von Null verschiedenen Wert besitzt. Da die hier vorliegende Matrix $\mathbf{A}$ aber nicht quadratisch und daher auch nicht regulär ist, lässt sich zu $\mathbf{A}$ keine Inverse bilden [Papula 09].

[12] Zu jeder Matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ lässt sich die transponierte Matrix $\mathbf{M^T} \in \mathbb{R}^{m \times n}$ bilden, indem Zeilen und Spalten miteinander vertauscht werden [Papula 09].

Algorithmus 4.1 Pseudocode für die Schätzung der Normal und Ambient Map

```
Einlesen der 81 BTF-Samples für θ_v = 0 und φ_v = 0 (entspricht P_Color)
Generierung einer diffusen Texture Map (entspricht T_Color)
Für alle 256x256 Pixel
    Für alle 81 BTF-Samples
        θ_l und φ_l verwenden, um Lichtvektor l zu rekonstruieren
        Koeffizienten T_Color und l in Matrix A ∈ ℝ^{243×6} eintragen
        Transponierte Matrix A^T ∈ ℝ^{6×243} berechnen
    Minimierungsproblem der Form Ax = b lösen (x beinhaltet n und I_A, mit b = P_Color)
    Probe machen: Geschätztes x mit A multiplizieren und mit b vergleichen
Normal Map, Ambient Map, sowie Differenzbilder der Probe ausgeben
```

mit Hilfe des Gauß-Jordan-Algorithmus invertiert werden, da eine Matrix $\left(\mathbf{A^T A}\right)^{-1} \in \mathbb{R}^{6\times 6}$ existiert, so dass für die Einheitsmatrix $\mathbf{E}$ die Bedingung $\left(\mathbf{A^T A}\right) \cdot \left(\mathbf{A^T A}\right)^{-1} = \mathbf{E}$ erfüllt ist. Des Weiteren ist wegen des Maximalrangs von $\mathbf{A}$ die symmetrische Matrix $\mathbf{A^T A}$ positiv definit[13], was zur Folge hat, dass die Unbekannten x_j durch die Normalengleichungen 4.25 eindeutig bestimmt sind [Schwarz 09].

Als letzter Schritt werden beide Seiten des Gleichungssystems 4.25 mit $\left(\mathbf{A^T A}\right)^{-1}$ multipliziert, so dass sich nach Kürzen der gesuchte Parametervektor wiefolgt ergibt:

$$\mathbf{x} = (\mathbf{A^T A})^{-1}\mathbf{A^T b} \tag{4.26}$$

Damit erhält man zusammenfassend folgende Schritte, die zu einer eindeutigen Lösung führen und die quadratische Abweichung minimieren:

$$\begin{aligned} \mathbf{Ax} &= \mathbf{b} \\ \mathbf{A^T Ax} &= \mathbf{A^T b} \\ \mathbf{x} &= (\mathbf{A^T A})^{-1}\mathbf{A^T b} \end{aligned} \tag{4.27}$$

Ablauf

Der Ablauf der verschiedenen Berechnungsschritte zur Schätzung des Parametervektors $\mathbf{x}$ wurde in Algorithmus 4.1 grob skizziert und verdeutlicht die Reihenfolge und Vorgehensweise

[13] Eine quadratische, symmetrische Matrix $\mathbf{M}$ heißt positiv definit, falls alle ihre Eigenwerte größer als Null sind. Existiert ein Vektor $\mathbf{x}$, der durch $\mathbf{M}$ auf ein Vielfaches von sich selbst abgebildet wird – gilt also die Gleichung $\mathbf{M} \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$ – so ist $\mathbf{x}$ ein Eigenvektor und λ ein Eigenwert der Matrix $\mathbf{M}$ [Papula 09].

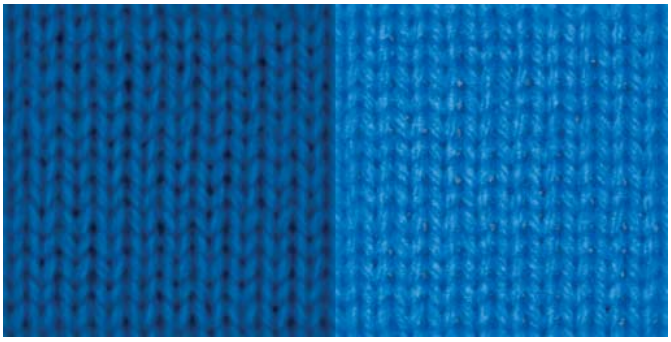

Abbildung 4.7: Links: Diffuse Textur, die als Mittelung aller BTF-Samples berechnet wurde. Rechts: BTF-Sample mit frontaler Beleuchtung und unerwünschten Reflexionen.

zur Bestimmung der pixelweisen Approximation eines Normalenvektors und des Werts eines ambienten Restanteils. Im Folgenden soll nun auf diejenigen Schritte eingegangen werden, die bisher noch nicht besprochen wurden. Nachdem die Messdaten in Form der insgesamt 81 BTF-Samples für die frontale Betrachtungsrichtung mit $\theta_v = 0$ und $\phi_v = 0$ eingelesen und als zweidimensionale Textur mit den pixelweisen Werten $P_{Color} = (p_r, p_g, p_b)^T$ gespeichert wurden, kann mit ihrer Hilfe die für die Rendering-Gleichung

$$P_{Color} = I_A \otimes T_{Color} + ((\mathbf{n} \cdot \mathbf{l}) \cdot L_{Diffuse}) \otimes T_{Color}$$

und somit auch für die Schätzung benötigte diffuse Textur $T_{Color} = (t_r, t_g, t_b)^T$ durch eine pixelweise Anwendung des arithmetischen Mittels auf sämtliche Messwerte berechnet werden. Die resultierende diffuse Textur für das Beispiel des Wollmaterials aus der BTF-Datenbank Bonn, die auf der linken Seite in Abbildung 4.7 zu sehen ist, enthält aufgrund der Mittelung keine unerwünschten Reflexionen mehr, die beispielsweise im BTF-Sample bei frontaler Beleuchtung ($\theta_l = 0$ und $\phi_l = 0$) auf der rechten Seite der Abbildung auftreten und die in entsprechender Form in allen 81 Samples wiederzufinden sind. Um also keine der 81 Beleuchtungsrichtungen zu favorisieren und somit das Rendering-Ergebnis zu verfälschen, das vor allem durch das Normal Mapping – also in Abhängigkeit von Normalenvektor und Lichteinfall – seine charakteristische Erscheinung erhalten soll, wurde als Kompromiss die gemittelte Textur gewählt.

Bevor nun die Koeffizienten T_{Color} und $\mathbf{l}$ in die Matrix $\mathbf{A}$ eingetragen werden können (vgl.

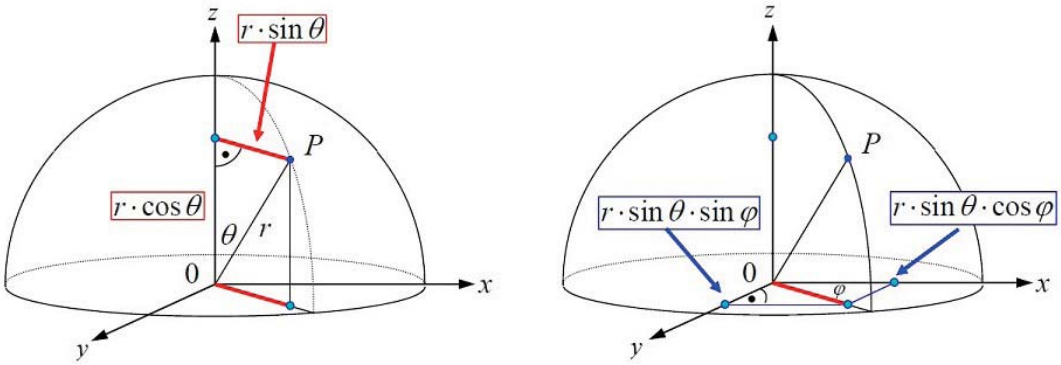

Abbildung 4.8: Ausgehend von den Polarkoordinaten (θ, ϕ, r) können die kartesischen Koordinaten (x, y, z) mit Hilfe von Sinus und Kosinus berechnet werden [Grosch 09].

Gleichung 4.23), muss zunächst noch der Lichtvektor $\mathbf{l}$ mit Hilfe der sphärischen Koordinaten (θ, ϕ), die in den Dateinamen der BTF-Samples in Form von Indizes kodiert wurden (vgl. Abschnitt 4.2.5), rekonstruiert werden. Die Umrechnung in kartesische Koordinaten erfolgt dabei über die Berechnungsvorschrift

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \sin\theta \cos\phi \\ r \sin\theta \sin\phi \\ r \cos\theta \end{pmatrix} \tag{4.28}$$

wobei $r \in \mathbb{R}$ den Abstand des Punktes P zum Koordinatenursprung darstellt und als Radius der Einheitskugel gleich 1 gesetzt werden kann (vgl. Abbildung 4.8).

Anschließend können die Richtung des nun in kartesischen Koordinaten vorliegenden Lichteinfalls und der Wert der diffusen Textur für den aktuellen Pixel in die Matrix $\mathbf{A}$ eingetragen werden, um mit der eingangs beschriebenen Lösung des Minimierungsproblems fortzufahren. Die beiden resultierenden Texturen – die Normal und die Ambient Map – werden dann ausgegeben und außerdem dazu verwendet, eine Probe durchzuführen, um diese mit den Original-BTF-Aufnahmen vergleichen zu können. Auf diesem Weg ist es möglich, sowohl qualitative, visuelle Unterschiede zu bestimmen als auch quantitative Werte für die Abweichungen zu untersuchen, worauf in Abschnitt 4.4 ausführlich eingegangen werden soll.

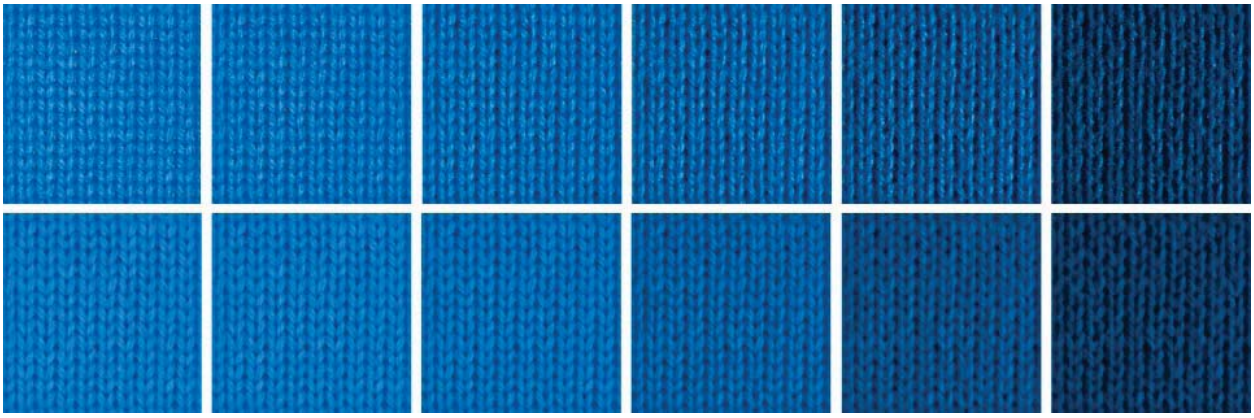

Abbildung 4.9: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Wolle“. Unten: Rekonstruierte Samples, die mittels der geschätzten Parameter berechnet wurden.

4.4 Ergebnisse

Mit Hilfe des geschätzten Normalenvektors $\mathbf{n}$ und des ambienten Restanteils I_A können nun unter Verwendung der Gleichung

$$P^*_{Color,i} = I_A \otimes T_{Color} + ((\mathbf{n} \cdot \mathbf{l_i}) \cdot L_{Diffuse}) \otimes T_{Color}$$

$n = 81$ Texturen generiert werden, deren Pixelfarbe $P^*_{Color,i}$ von den BTF-Samples in Form der Pixelfarbe $P_{Color,i}$ abgezogen wird, um den quadratischen Fehler für jedes der 81 Differenzbilder im einzelnen bzw. die Gesamtabweichung für alle Differenzbilder bestimmen zu können. In Abbildung 4.9 ist zunächst eine Gegenüberstellung der Original-Samples für das Material „Wolle“ aus der BTF-Datenbank Bonn und der rekonstruierten Samples dargestellt, die mittels der geschätzten Parameter berechnet wurden. In den sechs Spalten ist jeweils der erste Repräsentant (d. h. $\phi_l = 0$) aus der Gruppe der Samples für $\theta_l = \{0, 15, 30, 45, 60, 75\}$ der Datenbank zu sehen, deren Aufbau in Abschnitt 4.2.5 bereits erläutert wurde.

Auffallend bei den rekonstruierten Samples ist in jedem Fall der fehlende Specular-Anteil, was insbesondere in den Lücken des Gewebes bei frontalem Lichteinfall in den ersten beiden Spalten ($\theta_l = 0$ bzw. $\theta_l = 15$) zu Tage tritt. Der Specular-Anteil wurde, wie bereits eingangs erwähnt, für den Optimierungsprozess zunächst vernachlässigt und kann demnach auch nicht

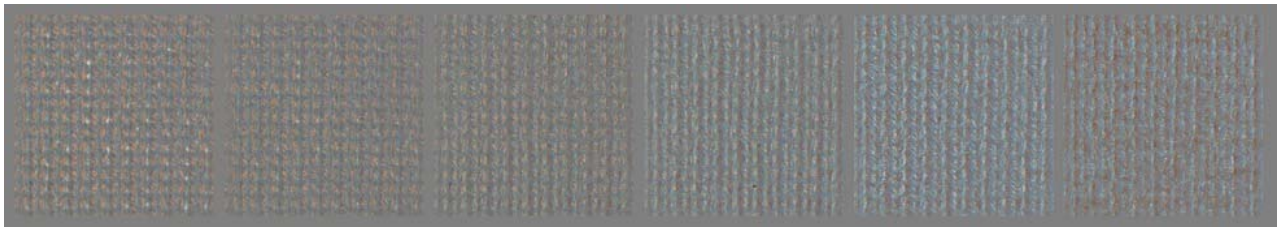

Abbildung 4.10: Die Gesamtabweichung der Schätzung vom Original für alle 81 Samples, die hier auf neutralem Grau dargestellt ist, beträgt für das Material „Wolle“ rund 0,2246%.

unter Verwendung dieses vereinfachten Beleuchtungsmodells wiedergegeben werden. Im nachfolgenden Abschnitt 4.4.1 soll eine Möglichkeit gezeigt werden, eine Annäherung an diesen Anteil zu ermitteln, um den visuellen Gesamteindruck zu verbessern.

Trotz des Fehlens besagter hochfrequenter Beleuchtungsanteile ist die Gesamtabweichung für das Material „Wolle“ mit 0,2246% – im Hinblick auf die anderen Stoffmaterialien, die im Anhang dieser Arbeit aufgelistet wurden – vergleichsweise niedrig. Betrachtet man die Abweichung der sechs Repräsentantengruppen, was in Abbildung 4.10 in Form der Differenzbilder des jeweils ersten Repräsentanten veranschaulicht wurde, ist anzumerken, dass die Bilder der zweiten Spalte für $\theta_l = 15$ mit 0,1814% die geringste und die Bilder der fünften Spalte für $\theta_l = 60$ mit 0,2585% die höchste Abweichung besitzen. Insgesamt lässt sich festhalten, dass die letzten drei Gruppen eine höhere Abweichung als die ersten drei Gruppen aufweisen, was u. a. auf die Anzahl der Samples in den jeweiligen Repräsentantengruppen zurückzuführen ist, da beispielsweise die 18 Samples der vierten Spalte insgesamt 19 Samples der ersten drei Gruppen gegenüberstehen, und die fünfte und sechste Gruppe sogar noch mehr Samples beinhalten (vgl. Abschnitt 4.2.5). Auch in den letzten drei Gruppen gibt es Differenzbilder mit sehr geringen Abweichungen, die z. T. sogar unter denen der ersten drei Gruppen liegen, was aber durch die wesentlich höhere Anzahl an Bildern wieder relativiert wird.

Zum anderen ist der zunehmend flachere Winkel des Lichteinfalls in den letzten Gruppen dafür verantwortlich, dass selbst die Original-Samples aus der BTF-Datenbank eine Ähnlichkeit mit Samples aus den ersten Gruppen vermissen lassen und auf den ersten Blick sogar eine Identifizierung des Stoffmaterials erschweren. Aus diesem Grund ist eine Verbesserung der Gesamtabweichung möglich, indem Samples der letzten Gruppen sukzessiv aus dem Optimierungsprozess ausgeschlossen werden, wodurch sich der quadratische Fehler zunehmend

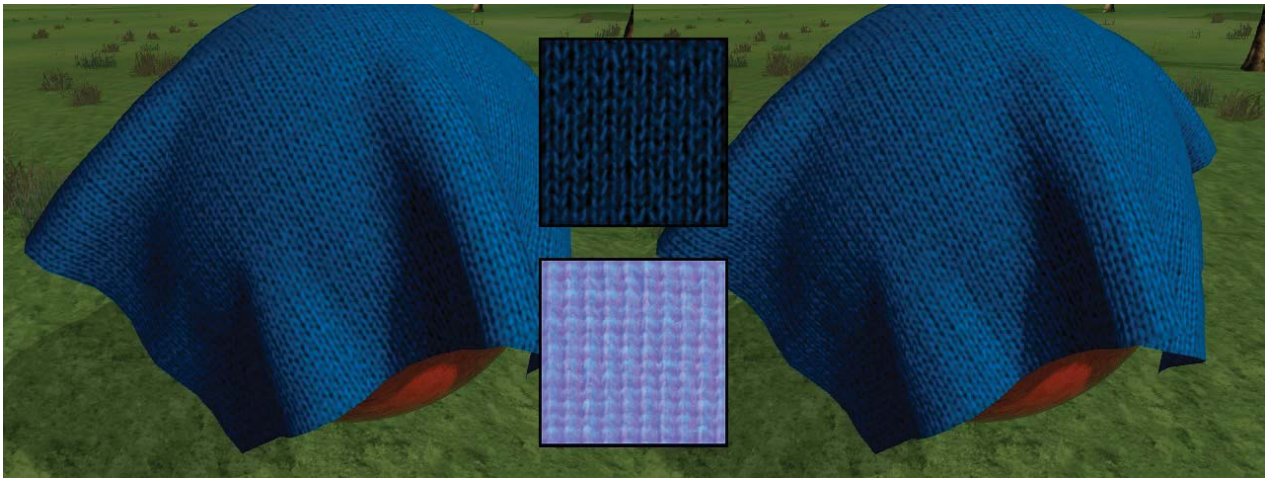

Abbildung 4.11: Links: Rendering mit einfachem Texture Mapping.
Rechts: Rendering mit Normal Mapping unter Verwendung der geschätzten Parameter.
Mitte oben: Ambient Map. Mitte unten: Normal Map

verringern ließe. Da aber im Rahmen dieser Arbeit eine Methode gesucht wurde, die alle 81 Samples in die Schätzung einbezieht und somit gute wie schlechte Kandidaten gleichmaßen berücksichtigt, soll diese Möglichkeit der Verbesserung nur als kurze Randnotiz erwähnt werden.

Die geschätzten Parameter, die in Form der Normal und Ambient Map vorliegen, können dann für das Rendering in der 3D-Echtzeitumgebung verwendet werden (vgl. Abbildung 4.11). Wie schon in Abschnitt 2.6 angesprochen liefert Normal Mapping ein deutlich ansprechenderes Ergebnis als einfaches Texture Mapping, wobei die Unterschiede besonders in Bewegung gut zur Geltung kommen und auf einem Standbild nur z. T. sichtbar sind. Dies liegt vor allem auch daran, dass für die hier gezeigten Szenen der Specular-Anteil vorerst weggelassen wurde, dieser aber gerade für die visuellen Charakteristika der Stoffoberfläche verantwortlich ist. Lässt man die Lichtquelle über dem Stofftuch kreisen oder ändert den Blickwinkel der Kamera, ist die Änderung der Oberfläche bei Verwendung von Normal Mapping – auch ohne Specular-Anteil – deutlich zu erkennen, wohingegen das Erscheinungsbild beim Texture Mapping nahezu unverändert bleibt, sobald die Lichtquelle oder der Betrachter ihre Position verändern (vgl. auch Abbildung 2.9 in Abschnitt 2.6). Bei einigen der anderen Stoffmateria-

Abbildung 4.12: Differenz von BTF-Sample #1 und Probe-Sample #1 wird als Specular Map weiterverwendet.

lien wie beispielsweise „Gewebtes Polyacryl“ und „Kord“ sind die Unterschiede auch auf den Standbildern wesentlich besser zu erkennen, deren Screenshots im Anhang dieser Arbeit zu finden sind.

Die in Abbildung 4.11 dargestellten Szenen wurden – wie bereits in Abschnitt 2.5.4 erwähnt – unter Zuhilfenahme des Fresnel-Effekts gerendert, der dazu führt, dass bei zunehmend flachen Betrachtungswinkeln, ein gewisser Teil der Umgebung reflektiert wird. Dieser Anteil ist bei dem hier gezeigten Material „Wolle“ eher kosmetischer Natur und insbesondere bei Stoffen wie Samt, Satin oder Kord – deren vordergründige, anisotrope Eigenschaften noch stärker wahrnehmbar sind als bei Wolle – dazu geeignet, die charakteristischen Reflexionen bei flachem Betrachtungswinkel wiederzugeben (vgl. Abschnitt 4.2.2).

4.4.1 Konstruktion der Specular Map

Um nun das visuelle Ergebnis bei der Berechnung der rekonstruierten Samples mit Hilfe des geschätzten Normalenvektors $\mathbf{n}$ und des ambienten Restanteils I_A weiter zu verbessern, wird die bereits angesprochene Beobachtung über den fehlenden Specular-Anteil – bei frontalem Lichteinfall und insbesondere in der ersten Spalte ($\theta_l = 0$ und $\phi_l = 0$) – ausgenutzt, indem gerade die fehlenden Specular-Beleuchtungsanteile durch eine einfache Berechnung der Differenz bzw. des Residuums aus dem ersten Original-Sample und dem ersten rekonstruierten Sample als Specular Map herangezogen werden (vgl. Abbildung 4.12). Wie in Abschnitt 2.5.3 beschrieben besitzt der Specular-Anteil $I_S = (\mathbf{n} \cdot \mathbf{h})^n \cdot L_{Specular}$ die Eigenschaft, in Abhängigkeit der Oberflächennormalen, der Kameraposition und des Lichteinfalls mehr Licht zum

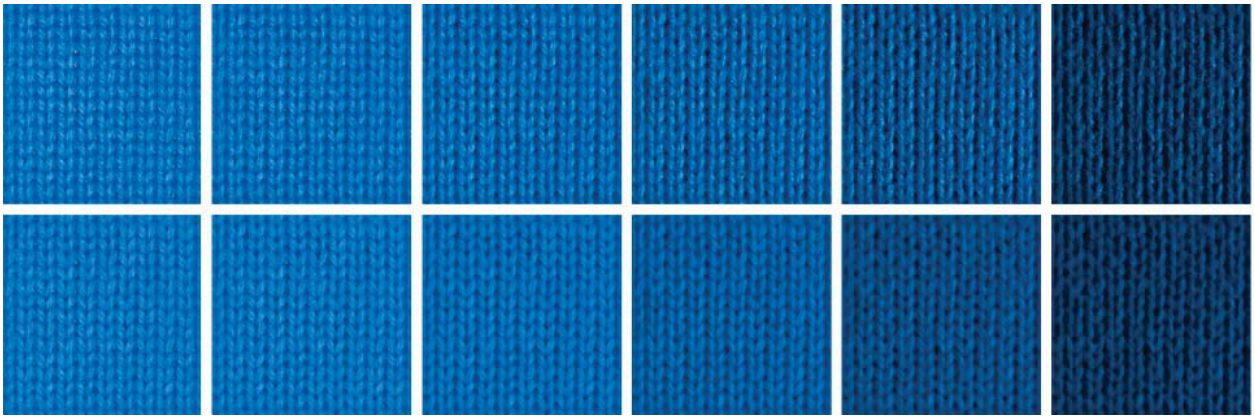

Abbildung 4.13: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Wolle“. Unten: Rekonstruierte Samples, die mittels der geschätzten Parameter berechnet wurden, wobei die Differenz von BTF-Sample #1 und Probe-Sample #1 als Specular Map verwendet wird.

Betrachter zurück zu reflektieren, wenn der Betrag des Winkels θ_h zwischen der Oberflächennormalen und dem Halbvektor kleiner wird, wobei das Punktprodukt $\mathbf{n} \cdot \mathbf{h}$ und damit auch der korrespondierende Kosinus des Winkels θ_h den maximalen Wert 1 erreicht, falls $\theta_h = 0$ ist. Aus diesem Grund eignet sich das erste Differenzbild – bei dem der Betrachtungswinkel ($\theta_v = 0$ und $\phi_v = 0$) und der Lichteinfall ($\theta_l = 0$ und $\phi_l = 0$) der Richtung der Oberflächennormalen entsprechen und somit die Bedingung $\theta_h = 0$ erfüllt ist – besonders gut als Specular Map, da davon ausgegangen werden kann, dass bei dieser Konstellation die Reflexion des Specular-Anteils zum Betrachter hin maximal ist. Bei allen anderen 80 Samples ist zumindest $\theta_l \neq 0$ und dadurch auch der Winkel θ_h zwischen Oberflächennormalen und Halbvektor in jedem Fall größer 0.

Die besagte Differenz der beiden Samples entspricht dann im Specular-Anteil $I_S = (\mathbf{n} \cdot \mathbf{h})^n \cdot L_{Specular}$ der Variablen $L_{Specular}$, welche die Rolle der Farbintensität dieses Beleuchtungsanteils übernimmt und somit als Specular Map fungiert, die den – in Abhängigkeit des Lichteinfalls, der Kameraposition und der Oberflächennormalen stehenden – Anteil $(\mathbf{n} \cdot \mathbf{h})^n$ innerhalb von I_S gewichtet. Die Rekonstruktion mittels der geschätzten Parameter und der nun vorliegenden Specular Map erfolgt dann durch die Gleichung

$$P^*_{Color,i} = I_A \otimes T_{Color} + ((\mathbf{n} \cdot \mathbf{l_i}) \cdot L_{Diffuse}) \otimes T_{Color} + (\mathbf{n} \cdot \mathbf{h_i})^n \cdot L_{Specular} \tag{4.29}$$

n	1	8	16	32	64	128	256	512	1024	optimales n
Wolle	0,3660	0,2640	0,2417	0,2299	0,2251	0,2238	0,2238	0,2241	0,2243	147
Polyacryl	0,3214	0,2542	0,2390	0,2307	0,2271	0,2259	0,2256	0,2255	0,2255	320
Kord	0,5527	0,4070	0,3744	0,3547	0,3450	0,3408	0,3391	0,3385	0,3383	1024
Polster	1,1660	0,8504	0,7659	0,7241	0,7191	0,7271	0,7350	0,7402	0,7430	51
Granit	2,4847	1,3438	1,0645	0,9027	0,8288	0,8124	0,8192	0,8294	0,8368	128

Tabelle 4.1: Gesamtabweichung verschiedener Materialien unter Einbeziehung des Specular-Anteils bei variierendem Specular-Exponenten n.

welche jetzt wieder mit der ursprünglichen Rendering-Gleichung aus Abschnitt 2.5.4 korrespondiert, nachdem ja für die Schätzung von $\mathbf{n}$ und I_A der Specular-Anteil zunächst ausgeklammert wurde.

In Abbildung 4.13 sind in der unteren Reihe die mit Hilfe von Gleichung 4.29 rekonstruierten Samples dargestellt, wobei der nun vorhandene Specular-Anteil insbesondere bei den Repräsentanten der ersten beiden Spalten zu einer höheren sichtbaren Ähnlichkeit mit den Originalen aus der oberen Reihe führt. Die Gesamtabweichung konnte auf diesem Weg von 0,2246% auf 0,2238% verbessert werden, was zwar angesichts des geringen Unterschieds in der Abweichung keine übermäßige Verbesserung vermuten lässt, sich aber dennoch in einer signifikanten, visuellen Aufwertung der rekonstruierten Samples – insbesondere bei frontalem Lichteinfall – niederschlägt. Betrachtet man beispielsweise nur die quadratische Abweichung für das erste Sample-Paar, ist eine Verbesserung der Abweichung von 0,1909% auf 0,0919% zu verzeichnen, was einer Verringerung der Abweichung um mehr als die Hälfte entspricht. Dass beim ersten Sample-Paar nicht eine quadratische Abweichung in Höhe von 0% vorliegt, obwohl gerade die Verwendung der Differenz dieses Paares als Specular Map u. U. zunächst eine hundertprozentige Übereinstimmung suggerieren könnte, liegt an der Verwendung der Differenz als Gewichtungsfaktor $L_{Specular}$ innerhalb des Specular-Anteils – anstatt sie als absoluten Specular-Anteil heranzuziehen. Durch die pixelweise Berechnung, die zwar nun von einem komplexeren Beleuchtungsmodell ausgeht als noch während des Optimierungsprozesses – aber natürlich noch immer nicht einer vollständigen Reproduktion der Realität entspricht – kann daher lediglich eine bessere Annäherung als vorher gewährleistet werden. Auf noch fehlende Elemente und mögliche Verbesserungen des Beleuchtungsmodells insbesondere auch in Bezug auf den Specular-Anteil soll im nachfolgenden Abschnitt 4.4.3 näher eingegangen werden.

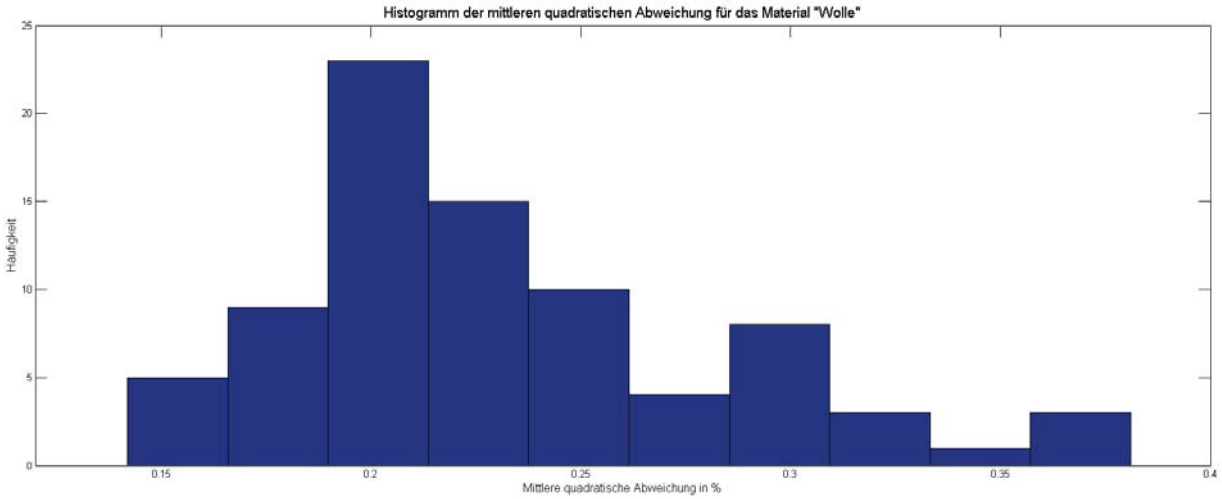

Abbildung 4.14: Histogramm der mittleren quadratischen Fehler für das Material „Wolle“.

Der Specular-Exponent n wurde für die Rekonstruktion der Samples empirisch ermittelt, wobei sich feststellen lässt, dass die quadratische Abweichung für das Material „Wolle“ für $n = 147$ minimal wird. In Tabelle 4.1 sind die Gesamtabweichungen für einige Materialien aus der BTF-Datenbank Bonn bei variierenden Exponenten aufgeführt, die mit der im Rahmen dieser Arbeit vorgestellten Methode rekonstruiert wurden. Diejenigen Exponenten, welche die Gesamtabweichung minimieren, sind hierbei in der letzten Spalte aufgelistet.

Ergänzend zur Analyse der Abweichungen ist in Abbildung 4.14 das Histogramm der mittleren quadratischen Fehler für das Material „Wolle“ dargestellt, wobei hier die Abweichungen zu sehen sind, falls nur die Normal und Ambient Map geschätzt und für die Probe verwendet werden. Demgegenüber steht das Histogramm in Abbildung 4.15, bei dem die Verbesserung in Form der soeben besprochenen Specular Map bereits eingeflossen ist. Vergleicht man die beiden Histogramme, so fällt vor allem die im zweiten Histogramm fehlende letzte Gruppe von leicht erhöhten Abweichungen auf, was auf den nun innerhalb der Berechnungen berücksichtigten Specular-Anteil zurückzuführen ist.

In Tabelle 4.2 sind die Gesamtabweichungen sowie die Teilabweichungen für die Repräsentantengruppen der Materialien aus der BTF-Datenbank Bonn mit und ohne Berücksichtigung des Specular-Anteils zusammengefasst. Für die Berechnung des Specular-Exponenten wurde jeweils der Exponent verwendet, der die Gesamtabweichung gemäß Tabelle 4.1 minimiert. In

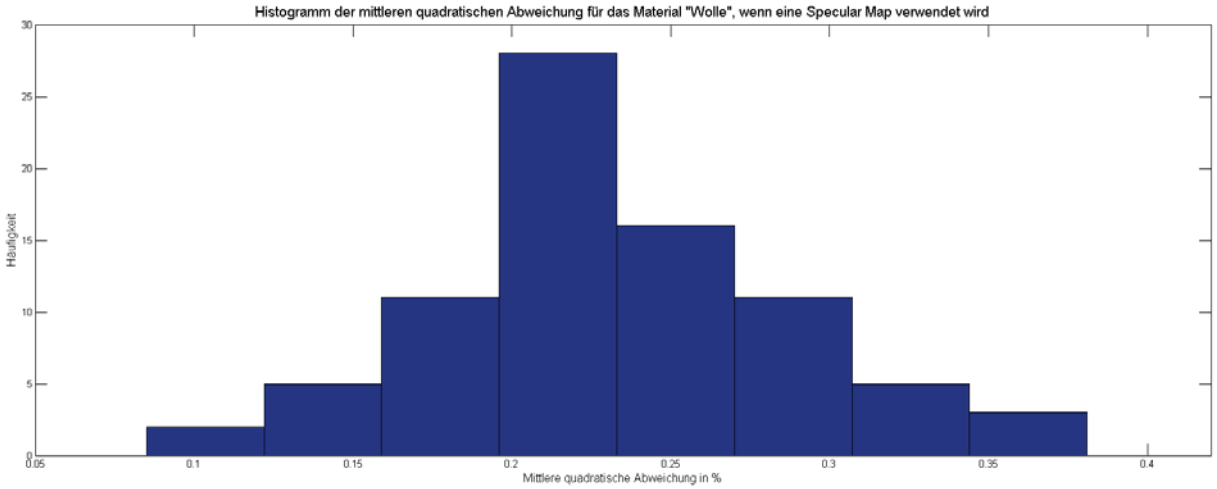

Abbildung 4.15: Histogramm der mittleren quadratischen Fehler für das Material „Wolle“, wobei die Differenz von BTF-Sample #1 und Probe-Sample #1 als Specular Map verwendet wird.

den meisten Fällen ist eine leichte Verbesserung der Abweichung zu verzeichnen, obgleich im Fall vom Material „Polyacryl“ keine Verbesserung in der Gesamtabweichung – und nur eine Änderung der Verteilung bei den einzelnen Repräsentantengruppen – zu beobachten ist. Beispielsweise verringert sich hier die Abweichung in den ersten Gruppen, was aber durch die Verschlechterung der Abweichung und die höhere Anzahl an Samples in den letzten Gruppen wieder aufgehoben wird.

Ähnlich verhält es sich beim Material „Kord“, wobei hier sogar eine leichte Erhöhung der Abweichung bei einer zusätzlichen Berechnung des Specular-Anteils zustande kommt. Zwar verringert sich bei diesem Material die Gesamtabweichung zunehmend, sobald der Exponent n weiter erhöht wird (vgl. Tabelle 4.1), jedoch reicht dies für den verwendeten Bereich von 1 bis 1024 nicht aus, um eine Verbesserung der Abweichung herbeizuführen, was darauf schließen lässt, dass das verwendete Modell nicht in der Lage ist, die komplexen Reflexionseigenschaften von Kord wiederzugeben, worauf ebenfalls im nachfolgenden Abschnitt 4.4.3 eingegangen werden soll. Verwendet man einen Exponenten $n > 4000$, lässt sich die Gesamtabweichung auf höchstens 0,3381% herabsenken, was der Abweichung ohne Berücksichtigung eines Specular-Anteils entspricht. Daraus lässt sich folgern, dass der Term $I_S = (\mathbf{n} \cdot \mathbf{h})^n \cdot L_{Specular}$ die noch fehlende Differenz zum Original in diesem Fall nur unzureichend approximiert, da bei der

Material	Specular	Gesamtabweichung in %	Abweichung in % für Repräsentantengruppe					
			$\theta_l = 0$	$\theta_l = 15$	$\theta_l = 30$	$\theta_l = 45$	$\theta_l = 60$	$\theta_l = 75$
Wolle	ohne	0,2246	0,1909	0,1814	0,1931	0,2253	0,2585	0,2499
	mit $n = 147$	0,2238	0,0919	0,1592	0,1985	0,2296	0,2593	0,2500
Polyacryl	ohne	0,2255	0,1347	0,1295	0,1297	0,1888	0,2430	0,3391
	mit $n = 320$	0,2255	0,1178	0,1242	0,1302	0,1904	0,2438	0,3393
Kord	ohne	0,3381	0,3394	0,2397	0,2384	0,3479	0,3979	0,3942
	mit $n = 1024$	0,3383	0,3018	0,2365	0,2394	0,3499	0,3995	0,3938
Polster	ohne	0,7467	0,5632	0,4548	0,3424	0,5455	0,9986	1,0466
	mit $n = 51$	0,7182	0,0374	0,1972	0,2995	0,5594	1,0012	1,0466
Granit	ohne	0,8473	2,1737	0,6559	0,4841	0,7265	1,0618	1,0345
	mit $n = 128$	0,8124	0,4787	0,4027	0,5043	0,7307	1,0644	1,0348

Tabelle 4.2: Gesamtabweichung verschiedener Materialien mit und ohne Specular-Anteil.

Verwendung eines so hohen Exponenten der Term $(\mathbf{n} \cdot \mathbf{h})^n$ – und somit I_S ebenfalls – für große n und $\mathbf{n} \cdot \mathbf{h} < 1$ gegen Null geht und dadurch auf die Pixelfarbe keinen Einfluss mehr hat.

4.4.2 Alternative Darstellung der Unterschiede zwischen Original und Rekonstruktion

Eine weitere Möglichkeit, die Unterschiede zwischen Original und Rekonstruktion zu visualisieren, ist die Beobachtung und Darstellung eines bestimmten Pixels unter wechselndem Lichteinfall. Hierzu wurde in Abbildung 4.16 für jeweils zwei Pixel die Farbe im Original-Sample auf der linken Seite und desselben Pixels im rekonstruierten Sample auf der rechten Seite in einer Kreisform grafisch dargestellt. Die Kreisform ist eine direkte Ableitung aus den Positionen der Lichtquelle während der Aufnahme, wobei zur Veranschaulichung die bereits aus Abschnitt 4.2.5 bekannte Abbildung der Kamerafahrt bzw. der Bahn der Lichtquelle auf der rechten Seite von Abbildung 4.16 hinzugefügt wurde. Der Polarwinkel θ_l korrespondiert hierbei mit den verschiedenen Ringen eines Kreises, wobei von innen nach außen die Werte zwischen 0° und 75° in 15°-Schritten angenommen werden. Auf den jeweiligen Ringen liegen dann – gemäß der Tabelle in Abschnitt 4.2.5 – die Azimutwinkel ϕ_l, deren Anzahl pro Ring nach außen hin wegen der höheren Anzahl an dort angefertigten Fotos ansteigt. Würde man die schematische Darstellung der Bahn der Lichtquelle aus Abbildung 4.16 von oben herab

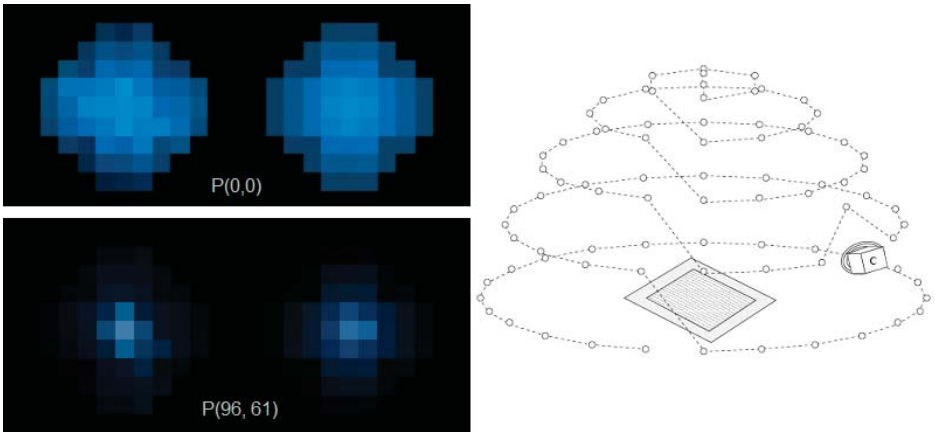

Abbildung 4.16: Links: Plot zweier Pixel in Abhängigkeit von θ_l und ϕ_l, wobei jeweils links das Original-Sample und rechts das rekonstruierte Sample für variierenden Lichteinfall verwendet wurde.
Rechts: Schematische Darstellung des Verlaufs der Lichtquelle bei der Messung einer Probe [Filip 05].

betrachten, würde sich ein solcher Kreis ergeben.

Es wurden exemplarisch zwei verschiedene Bildpunkte ausgewählt, von denen der in Abbildung 4.16 oben dargestellte Pixel mit den Koordinaten (0, 0) einer Stelle im Material „Wolle“ auf der Oberseite einer Faser entspricht (vgl. auch Abbildung 4.12). Auffallend ist hier die fast gleichbleibende Helligkeit bei den verschiedenen Ringen, was mit der guten Erreichbarkeit durch die Lichtstrahlen – selbst bei flachem Lichteinfall – korrespondiert. Dennoch ist beim linken Kreis, der einen Pixel aus dem Original-Sample repräsentiert, eine höhere Helligkeits-Fluktuation zu beobachten, wohingegen der rechte Kreis, der demselben Pixel im geschätzten Sample entspricht, eine gleichförmigere Helligkeitsverteilung aufweist, was auf das vereinfachte Beleuchtungsmodell zurückzuführen ist.

Betrachtet man den unten dargestellten Pixel mit den Koordinaten (96, 61), der genau in einer Lücke zwischen zwei Stofffasern liegt, wo der Specular-Anteil sehr dominant ist, ergibt sich in Bezug auf die höhere Fluktuation im Original ein ähnliches Bild, obgleich hier durch die Lage des Bildpunktes und die dadurch schlechtere Erreichbarkeit durch die Lichtstrahlen bei flachem Lichteinfall – also höheren Werten von θ_l – wesentlich größere Unterschiede in der Helligkeit nach außen hin zu verzeichnen sind. Zusammenfassend lässt sich festhalten, dass

Material	Specular	Gesamtabweichung in %	Abweichung in % für Repräsentantengruppe					
			$\theta_l = 0$	$\theta_l = 15$	$\theta_l = 30$	$\theta_l = 45$	$\theta_l = 60$	$\theta_l = 75$
Wolle	neu mit $n = 87$	0,2186	0,0000	0,1306	0,1908	0,2253	0,2585	0,2499
	alt mit $n = 147$	0,2238	0,0919	0,1592	0,1985	0,2296	0,2593	0,2500
Polyacryl	neu mit $n = 54$	0,2210	0,0000	0,0941	0,1250	0,1897	0,2431	0,3391
	alt mit $n = 320$	0,2255	0,1178	0,1242	0,1302	0,1904	0,2438	0,3393
Kord	neu mit $n = 74$	0,3284	0,0000	0,1750	0,2315	0,3482	0,3979	0,3942
	alt mit $n = 1024$	0,3383	0,3018	0,2365	0,2394	0,3499	0,3995	0,3938
Polster	neu mit $n = 51$	0,7111	0,0000	0,1546	0,2896	0,5513	0,9991	1,0466
	alt mit $n = 51$	0,7182	0,0374	0,1972	0,2995	0,5594	1,0012	1,0466
Granit	neu mit $n = 128$	0,7999	0,0000	0,3603	0,4906	0,7265	1,0618	1,0345
	alt mit $n = 128$	0,8124	0,4787	0,4027	0,5043	0,7307	1,0644	1,0348

Tabelle 4.3: Gesamtabweichung verschiedener Materialien mit optimiertem und ursprünglichem Specular-Anteil.

hochfrequente Änderungen der Farbintensität in den Originalen auf der linken Seite durch das hier verwendete Beleuchtungsmodell nur teilweise angenähert werden und wegen der verwendeten Optimierungsmethode zur Gewinnung der geschätzten Parameter und Minimierung der quadratischen Fehler aller 81 Samples insgesamt gleichmäßiger erscheinen.

4.4.3 Verbesserungen

Verbesserung des Specular-Anteils

Die zu Beginn des letzten Abschnitts angesprochenen Voraussetzungen für eine optimale Bestimmung des Specular-Anteils durch Ermittlung der Differenz des ersten Original-Samples und des ersten rekonstruierten Samples lassen sich weiter verbessern, da der frontale Betrachtungswinkel und Lichteinfall bei dieser Konstellation – um die angesprochene Bedingung $\theta_h = 0$ für den Winkel θ_h zwischen Oberflächennormale und Halbvektor zu erfüllen – auf eine Oberflächennormale angewiesen sind, die parallel zu ihnen ausgerichtet ist. Da für die Berechnung des Specular-Anteils aber die geschätzte Normal Map verwendet wird, deren Normalenvektoren an die Oberfläche des jeweiligen Stoffes angepasst sind, ist in den seltensten Fällen gewährleistet, dass der Betrag des Winkels θ_h für jeden Pixel den Wert 0 annimmt, wenn zumindest die Bedingung des frontalen Lichteinfalls erfüllt ist. Daher bietet es sich an, einen modifizierten Normalenvektor $\mathbf{n_S} = (0, 0, 1)^T$ zu verwenden, der für jeden Pixel ortho-

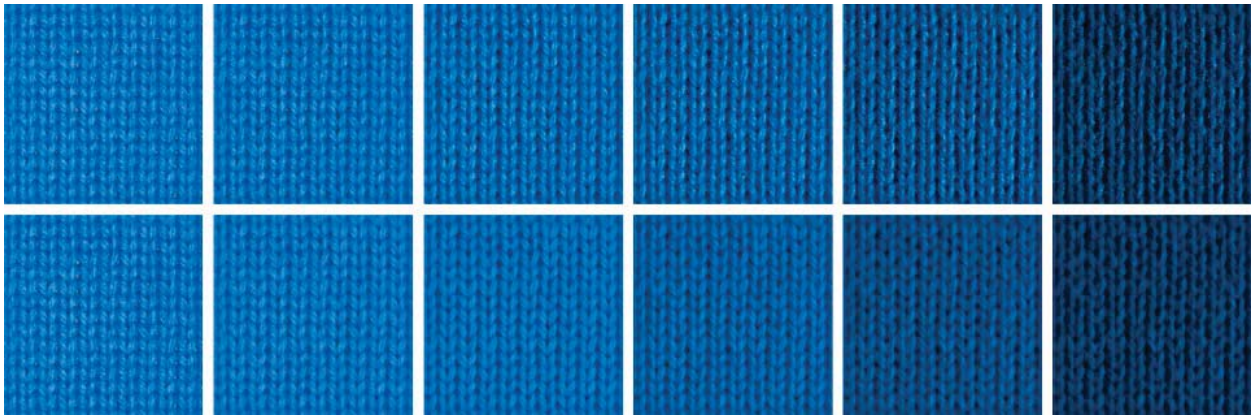

Abbildung 4.17: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Wolle“. Unten: Rekonstruierte Samples mit verbessertem Specular-Anteil.

gonal zur Oberfläche des Polygons – und nicht mehr senkrecht zum Oberflächenpunkt jedes individuellen Pixels – steht. Die in dieser Weise angepasste Gleichung zur Berechnung der rekonstruierten Samples sieht dann wiefolgt aus

$$P^*_{Color,i} = I_A \otimes T_{Color} + ((\mathbf{n} \cdot \mathbf{l_i}) \cdot L_{Diffuse}) \otimes T_{Color} + (\mathbf{n_S} \cdot \mathbf{h_i})^n \cdot L_{Specular} \tag{4.30}$$

wobei jetzt für die Berechnung des diffusen Beleuchtungsanteils wie gehabt die geschätzte Normal Map und für den Specular-Anteil der neue Normalenvektor $\mathbf{n_S}$ verwendet wird.

Durch diese Modifizierung ist eine leichte Verbesserung der Gesamtabweichung bei allen Materialien und zudem eine Abweichung in Höhe von 0% für die jeweils erste Repräsentantengruppe mit $\theta_l = 0$ zu verzeichnen (vgl. Tabelle 4.3 und Abbildung 4.17), da für diesen Fall der Term $\mathbf{n_S} \cdot \mathbf{h}$ innerhalb des Specular-Anteils I_S den maximalen Wert 1 annimmt

$$(\mathbf{n_S} \cdot \mathbf{h})^n = (\mathbf{n_S} \cdot \frac{\mathbf{l}+\mathbf{v}}{\|\mathbf{l}+\mathbf{v}\|_2})^n = (\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^T \cdot \frac{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}{2})^n = (\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^T \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix})^n = 1$$

und somit $I_S = L_{Specular}$ gilt, was genau dem ersten Differenzbild bzw. der Specular Map

entspricht. Durch die Erhöhung des Betrags von $\mathbf{n_S} \cdot \mathbf{h}$ – insbesondere für die ersten Repräsentantengruppen – konnte außerdem für die meisten Materialien ein kleinerer Specular-Exponent n ermittelt werden (vgl. Tabelle 4.3).

Mögliche Erweiterungen des Beleuchtungsmodells

Für das in dieser Arbeit vorgestellte Optimierungsverfahren werden nur die 81 BTF-Samples analysiert und verarbeitet, die bei einem konstanten frontalen Betrachtungswinkel und variierender Position der Lichtquelle aufgenommen wurden, da das Echtzeit-Rendering der Stoffe unter Zuhilfenahme von Normal Mapping umgesetzt wurde, was die zugrundeliegenden Materialien anhand ihrer Oberflächenstruktur – repräsentiert durch die Normalenvektoren – darstellt. Die Materialoberfläche, welche durch die Normal Map kodiert wird, entspricht dabei klassischerweise genau einer frontalen Momentaufnahme, bei der das Licht möglichst gleichmäßig – und ebenfalls von vorn – auf das Material gerichtet ist. Eine Möglichkeit wäre daher die Analyse der restlichen $6561 - 81 = 6480$ Fotos (vgl. Abschnitt 4.2.4) und Extraktion weiterer, charakteristischer Materialeigenschaften, die aus anderen Betrachtungswinkeln – und vor allem aus der Synthese der so neu dazugewonnenen Daten – womöglich überhaupt erst hervorgehen. Das hier verwendete, einfache Beleuchtungsmodell könnte dementsprechend um weitere Elemente erweitert werden, die beispielsweise dem Fresnel-Effekt, der gerade bei unterschiedlichen Betrachtungswinkeln relevant wird, bereits innerhalb des zugehörigen Optimierungsverfahrens Rechnung tragen. Ebenso könnten retroreflektierende Eigenschaften der Stoffe sowie anisotrope Merkmale unmittelbar im Beleuchtungsmodell untergebracht werden, wie es bei einigen der in Abschnitt 4.2.6 erwähnten Modelle der Fall ist. Auf der anderen Seite muss abgewogen werden, inwiefern und zu welchen Teilen der Echtzeitaspekt vernachlässigt werden kann, um die akkuratere Modellierung der Stoffe umzusetzen. Da für die vorliegende Arbeit der Echtzeitaspekt im Vordergrund stand, wurden bestimmte physikalisch korrekte Reflexionseigenschaften zugunsten approximierter Merkmale außer Acht gelassen.

Bestimmte Eigenschaften von Stoffoberflächen lassen sich ungeachtet des verwendeten Modells und dessen Komplexität prinzipiell nur sehr schwer nachbilden und werden daher in Echtzeitanwendungen meist ignoriert. Dazu gehören vor allem feine Härchen, Fusseln, Flaum und Fasern, die – im Vergleich zu den ohnehin schon sehr kleinen Strukturen der Stoffe – z. T. der Mikroebene zuzuordnen sind und bei einer Betrachtung mit bloßem Auge nur aus kurzer Entfernung auffallen. In den meisten Fällen bilden diese Elemente eine neue Ebene auf der

Stoffoberfläche und müssten auch als solche mit einem eigenen Reflexionsmodell nachgebildet werden, was wegen des enormen, zusätzlichen Rechenaufwands und dem geringen visuellen Gewinn in einer Echtzeit-Umgebung – wo die Materialien hauptsächlich aus Entfernungen betrachet werden, aus denen solche Details ohnehin nicht wahrnehmbar sind – meist dazu führt, dass solche Materialeigenschaften weggelassen werden. Anwendungen aus dem Bereich des Offline-Renderings, die nicht an den Echtzeitaspekt gebunden sind, können an dieser Stelle mit komplexen Modellen anknüpfen und auch für Nahaufnahmen sehr authentische und realistische Ergebnisse erzielen, die so in Echtzeit-Anwendungen mit der aktuellen Hardware nicht möglich wären. Eine mögliche Annäherung für Echtzeitanwendungen wäre der Einsatz weiterer Texturschichten, um dieses Elemente anzudeuten, wobei hier eine hohe Auflösung der Texturen nötig wäre, um die feinen Details akkurat wiederzugeben.

Ähnlich verhält es sich mit dem Schattenwurf, der durch die Struktur der Stoffoberfläche entsteht und bei den BTF-Samples insbesondere bei den Materialien „Wolle“, „Gewebtes Polyacryl“ und „Kord“ gut zu sehen ist (vgl. auch Aufnahmen im Anhang). Schattenberechnungen auf der Mikro- und Mesoebene wurden für das in dieser Arbeit verwendete Beleuchtungsmodell zugunsten höherer Performanz ebenfalls nicht umgesetzt, wobei die mit Hilfe von Shadow Mapping generierten Schatten ausschließlich für die Makroeben des Stofftuchs zuständig sind (vgl. Abschnitt 2.7). Eine Möglichkeit wäre daher die Abdunklungen in den Furchen zwischen gröberen Stofffasern mittels Ambient Occlusion umzusetzen, das sich insbesondere für Echtzeit-Anwendungen eignet, um nochmals eine Steigerung der Qualität der Oberflächendarstellung der Stoffe zu erreichen. Ambient Occlusion ist ein Verfahren, bei dem die eingehende Beleuchtungsstärke E_l in der Hemisphäre eines jeden Oberflächenpunkts gemessen wird und entsprechend der Erreichbarkeit durch die Lichtstrahlen eine Abschattung oder Aufhellung stattfindet, wobei dies im Kontext des sonst als konstant festgelegten ambienten Lichtanteils geschieht [Möller 08]. Ambient Occlusion muss aber nicht zwangläufig auf der Mikroebene angewendet werden, sondern wird auch häufig auf die Geometrie der 3D-Objekte angewandt, weswegen der nun nur noch pro Vertex auftretende Rechenaufwand eine Verwendung auch in Echtzeitanwendungen erlaubt. Da dies aber wegen des hier verwendeten Shadow Mappings nur zu einem geringen, sichtbaren Unterschied geführt hätte, wurde auf eine Umsetzung dieses Verfahrens im Rahmen dieser Arbeit verzichtet.

5 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde eine 3D-Echtzeitumgebung vorgestellt, die es ermöglicht, verschiedene Stoffmaterialien mit Hilfe von Normal Mapping und unter dynamischen Lichtverhältnissen authentisch und überzeugend darzustellen, wobei der Benutzer in die Lage versetzt wird, durch Tastatur- und Mauseingaben mit einem Stofftuch zu interagieren. Hierzu wurden interne und externe physikalische Kräfte mittels Constraints modelliert, um diese dann innerhalb eines Masse-Feder-Systems als Partikel behandeln und das Stofftuch als Ganzes durch ein Partikelsystem simulieren zu können. Die internen Kräfte, zu denen Dehnungs-, Scher- und Biegekräfte gehören, wirken dabei zusammen mit den externen Kräften, wie beispielsweise Gravitation, Wind, Benutzerinteraktionen und Kollisionen mit anderen Objekten, mit Hilfe eines impliziten Integrationsschemas nach Verlet auf die Partikelpositionen, um dadurch eine numerische Stabilität auch bei großen Zeitschritten und schnellen Positionsänderungen und so eine schnelle und robuste Stoffsimulation zu gewährleisten.

Das Ergebnis der Simulation sind ein realistischer Faltenwurf der Stoffe, mit der Möglichkeit die Steifigkeit über die Manipulation der Nachbarschaftsrelationen der einzelnen Partikel zu steuern. In derselben Weise können Löcher und Risse im Tuch dargestellt werden, indem bestimmte Nachbarn der Partikel bei den Berechnungen ignoriert werden und zusätzlich eine passende Textur diejenigen Pixel mittels Alpha Testing ausblendet, die sich innerhalb dieser Lücken befinden. Um eine flüssige Simulation zu erhalten, die auf älteren und neuen Computern gleichermaßen überzeugend läuft, wird die Aktualisierungsrate der Simulation zudem automatisch an die jeweilige vorhandene Rechenleistung angepasst.

Schließlich wurden die grundlegenden Reflexionseigenschaften von Stoffmaterialien anhand von Bidirektionalen Reflektanzverteilungsfunktionen und Bidirektionalen Texturfunktionen erläutert, wobei das Datenmaterial aus der BTF-Datenbank Bonn als Ausgangspunkt für die im Rahmen dieser Arbeit entwickelte Methode zur Schätzung eines parametrischen Beleuchtungsmodells diente. Neben den Eigenschaften von BRDF und BTF wurden hierzu zunächst relevante radiometrische Größen sowie die Messung von BTF-Daten beschrieben, bei der je-

der Texel einer zweidimensionalen Textur als Funktion der eingehenden Beleuchtungsrichtung und der ausgehenden Betrachtungsrichtung gespeichert wird. Nachdem bisherige Ansätze der Modellierung von Bidirektionalen Texturfunktionen, zu denen analytische sowie statistische Verfahren zu zählen sind, vorgestellt wurden, folgte eine detaillierte Beschreibung der Methode. Unter der Annahme, dass sich die 81 Kameraufnahmen aus der BTF-Datenbank Bonn, die bei einen frontalen Betrachtungswinkel unter variierendem Lichteinfall angefertigt wurden, über eine aufgestellte Rendering-Gleichung approximieren lassen, wurden mittels der Methode der kleinsten Quadrate die Normalenvektoren und ein ambienter Restanteil, die beide Bestandteil besagter Rendering-Gleichung sind, geschätzt, um die Parameter anschließend für die Darstellung der Stoffe in der 3D-Echtzeitumgebung verwenden zu können.

Um die ermittelten quadratischen Abweichungen vom Original weiter zu minimieren, wurde zudem eine Möglichkeit der Gewinnung des Specular-Anteils vorgestellt – der bedingt durch seine exponentielle Natur aus dem Optimierungsverfahrens ausgeschlossen wurde – wodurch eine Steigerung der Qualität der rekonstruierten Stoffe erreicht werden konnte. Hierzu wurde das erste Differenzbild zwischen Original und Rekonstruktion als Ausgangsbasis für den Specular-Anteil herangezogen und zudem ein modifizierter Normalenvektor für die Berechnung dieses Beleuchtungsanteils verwendet, da auf diesem Weg die Abweichung weiter verringert werden konnte. Die Ergebnisse des Verfahrens wurden sowohl in Form der synthetischen Rekonstruktionen der Stoffe, der aufgestellten Differenzbilder und der Gegenüberstellung eines Pixels als Funktion des für den Lichteinfall zuständigen Polar- und Azimutwinkels präsentiert. Außerdem wurden die verschiedenen Materialien aus der BTF-Datenbank Bonn innerhalb der 3D-Echtzeitumgebung gerendert und die Ergebnisse mit und ohne Normal Mapping verglichen. Aufgewertet wurden die visuellen Ergebnisse hierbei durch das implementierte Shadow Mapping, mit dessen Hilfe für dynamische Punkt-, Richtungs- und Scheinwerferlichter, ein weicher Schattenwurf auf dem Stofftuch selbst und auf anderen Objekten der Umgebung produziert werden konnte.

Neben den bereits in Abschnitt 4.4.3 erwähnten Erweiterungsmöglichkeiten des Beleuchtungsmodells und des Optimierungsverfahrens – zu denen beispielsweise die Einbeziehung weiterer Kameraaufnahmen aus der BTF-Datenbank Bonn, die Berücksichtigung wesentlicher Reflexionseigenschaften, die aufgrund des hohen Rechenaufwands für das hier verwendete Modell ignoriert wurden und die Berechnung von Schatten für Zwischenräume auf der Stoffoberfläche mittels Ambient Occlusion zählen – sind weitere Elemente denkbar, um die die

Stoffsimulation als Ganzes und hier insbesondere die physikalische Korrektheit bei der Kollisionserkennung erweitert werden könnte. Beispielsweise wurden für die hier implementierte Stoffsimulation die Kollisionen einzelner Partikel des Stofftuchs untereinander komplett außer Acht gelassen, wodurch der Faltenwurf oder ein Fallenlassen des Tuchs auf eine Oberfläche wegen der auf diese Weise entstehenden Durchdringung mitunter unrealistisch aussehen. Daher wäre die Erweiterung um ein Verfahren, das diese sogenannten Selbstkollisionen erkennt, abfängt und behandelt, eine entscheidende Verbesserung hinsichtlich der physikalischen Korrektheit. Ein solches Verfahren könnte dabei ähnlich wie in der Arbeit von [Fuhrmann 03] Constraints verwenden, um in gleicher Weise wie die Behandlung sonstiger interner und externer Kräfte, eine Implementierung für Echtzeitanwendungen zu ermöglichen, da ausgeklügelte Verfahren wie im Fall von [Bridson 02] zwar durch komplexere Algorithmen ein äußerst realistisches Ergebnis erzielen, diese aber wegen des immensen Rechenaufwands weit von der Möglichkeit einer Anwendung in Echtzeitumgebungen entfernt sind. In jedem Fall darf bei den Erweiterungen die Wahrung des Echtzeitaspekts nicht vernachlässigt werden, um flüssige Animationen sowohl bei den physikalischen Berechnungen als auch bei der dynamischen Beleuchtung weiterhin gewährleisten zu können.

Literatur

[Baraff 98] David Baraff, Andrew Witkin: *Large steps in cloth simulation*. In ACM SIGGRAPH: SIGGRAPH 98 Conference Proceedings. Orlando, FL, USA. Juli 1998. S. 43–54.

[Blinn 77] James F. Blinn: *Models of Light Reflection for Computer Synthesized Pictures*. In SIGGRAPH 1977: Proceedings of the 4th annual conference on Computer graphics and interactive techniques, ACM Press, 1977, S. 192-198.

[Blinn 78] James F. Blinn: *Simulation of Wrinkled Surfaces*. In Computer Graphics (Proceedings of SIGGRAPH 78), August 1978, S. 286–292.

[Breen 94] David E. Breen, Donald H. House, Michael J. Wozny: *Predicting the drape of woven cloth using interacting particles*. In ACM SIGGRAPH: SIGGRAPH 94 Conference Proceedings. Orlando, FL, USA. Juli 1994. S. 365–372.

[Bridson 02] Robert Bridson, Ronald Fedkiw, John Anderson: *Robust treatment of collisions, contact and friction for cloth animation*. In ACM Transactions on Graphics (SIGGRAPH 2002), Ausgabe 21, 2002.

[BTF-DB Bonn] BTF Database Bonn. Internet: http://btf.cs.uni-bonn.de/ [Letzter Zugriff: 20.10.2010].

[Cords 04] Hilko Cords: *Physikalisch basierte Gewebesimulation in Echtzeit*. Diplomarbeit, Institut für Informatik, Universität Rostock, 2004.

[Dana 99] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar et al.: *Reflectance and Texture of Real-World Surfaces*. In ACM Transactions on Graphics, 1999, 18(1), S. 1–34.

[Daubert 01] K. Daubert, H. P. A. Lensch, W. Heidrich et al.: *Efficient cloth modeling and rendering*. In Proceedings of the 12th Eurographics Workshop on Rendering, 2001.

[Daubert 03] Katja Daubert: *Hardware-Supported Cloth Rendering*. Dissertation, Max-Planck-Institut für Informatik, Universität des Saarlandes, Saarbrücken, 2003.

[Eberhardt 96] Bernhard Eberhardt, Andreas Weber, Wolfgang Strasser: *A fast, flexible particle-system model for cloth draping*. In IEEE Computer Graphics and Applications, September 1996, 16(5), S. 52–59.

[Elpelt 07] Bärbel Elpelt, Joachim Hartung: *Multivariate Statistik: Lehr- und Handbuch der angewandten Statistik*. 7. Auflage, München: Oldenbourg Wissenschaftsverlag GmbH, 2007.

[Filip 05] Jiri Filip, Michal Haindl: *Efficient image-based bidirectional texture function model*. In Texture 2005: Proceedings of the 4th International Workshop on Texture Analysis and Synthesis, 2005, S. 7–12.

[Filip 09] Jiri Filip, Michal Haindl: *Bidirectional Texture Function Modeling: A State of the Art Survey*. In IEEE Transactions on Pattern Analysis and Machine Intelligence, November 2009, S. 1921-1940.

[Fuhrmann 03] Arnulph Fuhrmann, Clemens Groß, Volker Luckas: *Interactive Animation of Cloth including Self Collision Detection*. In Journal of WSCG, Februar 2003, 11(1), S. 141–148.

[Fuhrmann 06] Arnulph Fuhrmann: *Interaktive Animation textiler Materialien*. Dissertation, Technische Universität Darmstadt, 2006.

[Garstenauer 06] Martin Garstenauer: *Character Animation in Real-Time*. Magisterarbeit, Institut für Graphische und Parallele Datenverarbeitung, Johannes Kepler Universität Linz, Linz, 2006.

[Gebhardt 03] Nikolaus Gebhardt: *Einige BRDF Modelle*. Technische Universität von Wien, 2003. Internet: http://www.irrlicht3d.org/papers/BrdfModelle.pdf [Letzter Zugriff: 31.08.2010].

[Grosch 09] Thorsten Grosch: *Reflexion - 3. Vorlesung Photorealistische Computergrafik*. Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg, 2009. Internet: http://isgwww.cs.uni-magdeburg.de/visual/index.php?article_id=77&clang=0 [Letzter Zugriff: 06.01.2011].

[Hauth 02] Michael Hauth, Olaf Etzmuss, Bernd Eberhardt et al.: *Cloth Animation and Rendering*. In Eurographics 2002 Tutorials. Saarbrücken, Deutschland. 2002.

[Hering 04] Ekbert Hering, Rolf Martin, Martin Stohrer: *Physik für Ingenieure*. 9. Auflage, Berlin: Springer-Verlag GmbH, 2004.

[Irawan 08] Piti Irawan: *Appearance of Woven Cloth*. Dissertation, Cornell University, 2008.

[Jacobs 03] Paul Jacobs: *Real Time Cloth Animation Techniques*. EECS 600 Final Project, Case Western Reserve University, Cleveland, USA, 2003. Internet: http://filer.case.edu/pxj18/ [Letzter Zugriff: 04.05.2010].

[Jakobson 01] Thomas Jakobson: *Advanced Character Physics*. In Proceedings of Game Developers Conference, San Jose, USA, März 2001.

[Kawabata 80] Sueo Kawabata: *The standardization and analysis of hand evaluation*. 2. Auflage, Osaka, Japan: Textile Machinery Society of Japan, 1980.

[Kwoon 04] Hun Yen Kwoon: *The Theory of Stencil Shadow Volumes*. In Engel (Hg.): ShaderX2: Introductions & Tutorials with DirectX 9. 1. Auflage, Plano, Texas, USA: Wordware Publishing, Inc., 2004, S. 197-278.

[Lafortune 97] E. Lafortune, S. Foo, K. Torrance et al.: *Non-Linear Approximation of Reflectance Functions*. In SIGGRAPH 1997 Proceedings, August 1997, S. 117–126.

[Malzbender 01] T. Malzbender, D. Gelb, H.Wolters: *Polynomial texture maps*. In ACM SIGGRAPH 2001, ACM Press, Eurographics Association. Schweiz. 2001. S. 519–528.

[McAllister 02] D. McAllister, A. Lastra, W. Heidrich: *Efficient rendering of spatial bidirectional reflectance distribution functions*. In Eurographics / SIGGRAPH Workshop Proceedings, 2002.

[Meyer 01] Mark Meyer, Gilles Debunne, Mathieu Desbrun, Alan H. Barr: *Interactive Animation of Cloth-like Objects in Virtual Reality*. In The Journal of Visualization and Computer Animation, Mai 2001, 12, S. 1–12.

[Microsoft 09] Microsoft Corporation: *Windows DirectX Graphics Documentation*. In DirectX SDK August 2009. August 2009. Internet: http://www.microsoft.com/downloads/details.aspx?FamilyID=b66e14b8-8505-4b17-bf80-edb2df5abad4&displaylang=en [Letzter Zugriff: 13.06.2010].

[Möller 08] Tomas Akenine-Möller, Eric Haines, Naty Hoffman: *Real-Time Rendering*. 3. Auflage. Natick, MA, USA: A K Peters Ltd., 2008.

[Müller 05] G. Müller, J. Meseth, M. Sattler et al.: *Acquisition, Synthesis and Rendering of BTFs*. In Computer Graphics Forum, März 2005, 24(1), S. 83-109.

[Nicodemus 70] F. E. Nicodemus: *Reflectance Nomenclature and Directional Reflectance and Emissivity*. In Applied Optics, 1970, 9(6), S. 1474-1475.

[Nishino 99] K. Nishino, Y. Sato, K. Ikeuchi: *Eigen-texture Method: Appearance Compression based on 3D Model*. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Juni 1999, 1(78), S. 618–624.

[Papula 09] Lothar Papula: *Mathematische Formelsammlung: Für Ingenieure und Naturwissenschaftler*. 10. Auflage, Wiesbaden: Vieweg+Teubner, GWV Fachverlage GmbH, 2009.

[Peeper 04] Craig Peeper, Jason L. Mitchell: *Introduction to the DirectX High Level Shading Language*. In Engel (Hg.): ShaderX2: Introductions & Tutorials with DirectX 9. 1. Auflage, Plano, Texas, USA: Wordware Publishing, Inc., 2004, S. 1-62.

[Phong 75] Bui-Tuong Phong: *Illumination for Computer Generated Pictures*. In Communications of the ACM, 1975, 18(6), S. 311-317.

[Press 92] William Press, Saul Teukolsky, William Vetterling, Brian Flannery: *Numerical Recipes in C. The Art of Scientific Computing, Second Edition*. New York, USA: Cambridge University Press, 1992.

[Rapp 10] Heinz Rapp: *Mathematik für die Fachschule Technik*. 7. Auflage, Wiesbaden: Vieweg+Teubner, GWV Fachverlage GmbH, 2010.

[Sattler 03] Mirko Sattler, Ralf Sarlette, Reinhard Klein: *Efficient and realistic visualization of cloth*. In Proceedings of the 14th Eurographics Workshop on Rendering. Leuven, Belgien. Juni 2003. S. 167-177.

[Schwarz 09] Hans Rudolf Schwarz, Norbert Köckler: *Numerische Mathematik*. 7. Auflage, Wiesbaden: Vieweg+Teubner, GWV Fachverlage GmbH, 2009.

[Valient 04] Michal Valient: *Shadow Mapping with Direct3D 9*. In Engel (Hg.): ShaderX2: Introductions & Tutorials with DirectX 9. 1. Auflage, Plano, Texas, USA: Wordware Publishing, Inc., 2004, S. 181-196.

[Verlet 97] Loup Verlet: *Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*. In Physical Review, 1997, 159(1), S. 98–103.

[Wang 05] Jiaping Wang, Xin Tong, John Snyder et al.: *Capturing and Rendering Geometry Details for BTF-mapped Surfaces*. In The Visual Computer, September 2005, Volume 21.

[Wong 03] T.-T. Wong and C.-S. Leung: *Compression of illumination-adjustable images*. In IEEE Transactions on Circuits and Systems for Video Technology, 2003, 13(11), S. 1107–1118.

[Wynn 00] Chris Wynn: *An Introduction to BRDF-Based Lighting*. NVIDIA Corporation. 2000. Internet: http://developer.nvidia.com/attach/6568 [Letzter Zugriff: 30.09.2010].

[Zeller 06] Cyril Zeller: *Practical Cloth Simulation on Modern GPUs*. In Engel (Hg.): ShaderX4. Advanced Rendering Techniques. 1. Auflage, Hingham, MA, USA: Charles River Media Verlag, 2006, S. 17-27.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Ergebnisse der Schätzung für verschiedene Materialien aus der BTF-Datenbank Bonn

Auf den folgenden Seiten werden die Gegenüberstellungen der rekonstruierten Samples und der Original-Samples, sowie die resultierenden Abweichungen und die Ergebnisse des Minimierungsprozesses in Form der Normal Map und Ambient Map sowie Screenshots aus der 3D-Echtzeitumgebung für weitere Stoffmaterialien und das Material „Granit“ aus der BTF-Datenbank Bonn aufgelistet.

Abbildung 5.1: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Gewebtes Polyacryl“.
Mitte: Rekonstruierte Samples, die mittels geschätzter Parameter ohne Specular-Anteil berechnet wurden.
Unten: Rekonstruierte Samples, die mittels geschätzter Parameter mit Specular-Anteil berechnet wurden.

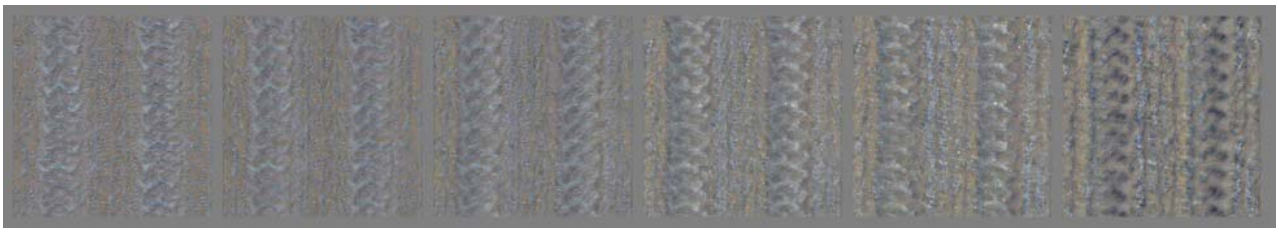

Abbildung 5.2: Die Gesamtabweichung der Schätzung vom Original für alle 81 Samples, die hier auf neutralem Grau dargestellt ist, beträgt für das Material „Gewebtes Polyacryl“ rund 0,2255%.

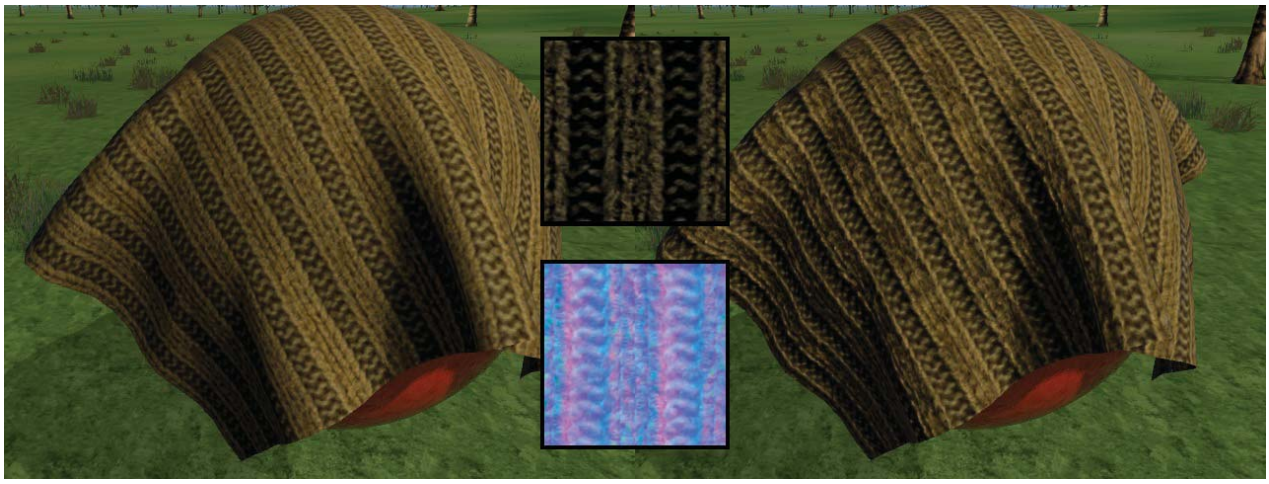

Abbildung 5.3: Links: Rendering mit einfachem Texture Mapping.
Rechts: Rendering mit Normal Mapping unter Verwendung der geschätzten Parameter inkl. Specular-Anteil.
Mitte oben: Ambient Map. Mitte unten: Normal Map.

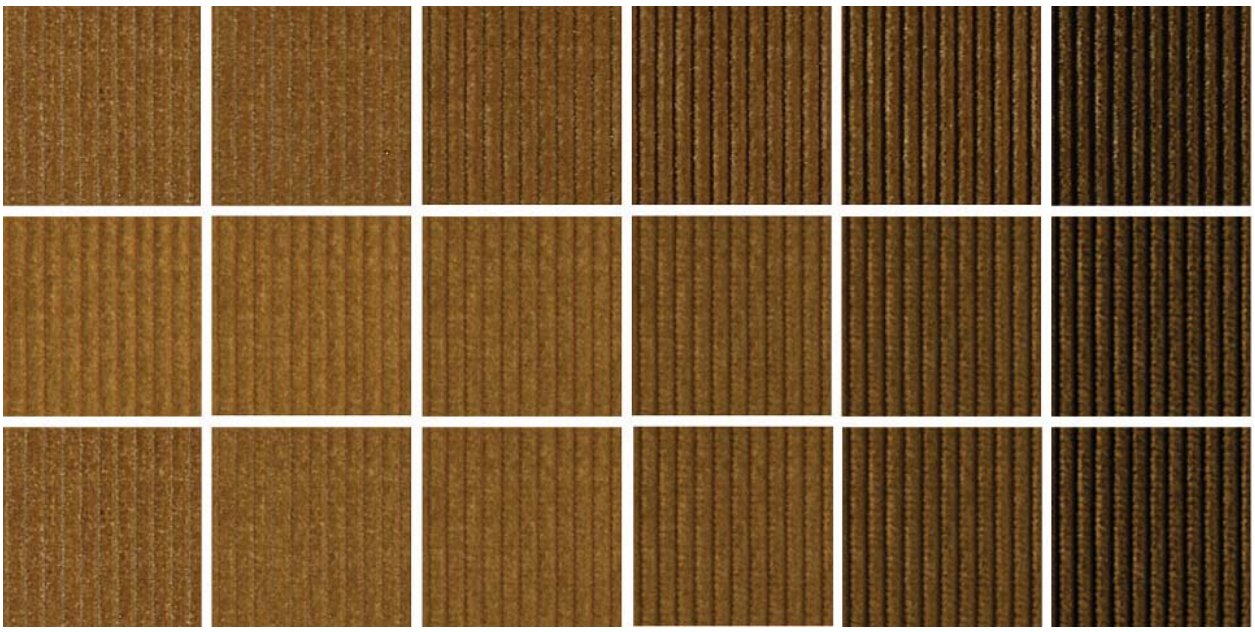

Abbildung 5.4: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Kord“.
Mitte: Rekonstruierte Samples, die mittels geschätzter Parameter ohne Specular-Anteil berechnet wurden.
Unten: Rekonstruierte Samples, die mittels geschätzter Parameter mit Specular-Anteil berechnet wurden.

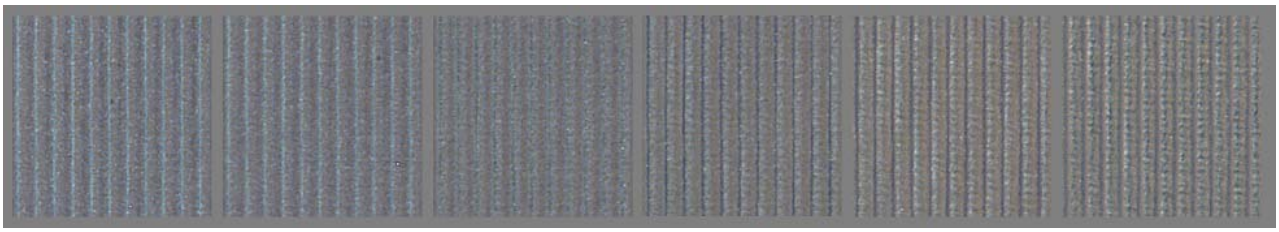

Abbildung 5.5: Die Gesamtabweichung der Schätzung vom Original für alle 81 Samples, die hier auf neutralem Grau dargestellt ist, beträgt für das Material „Kord“ rund 0,3381%.

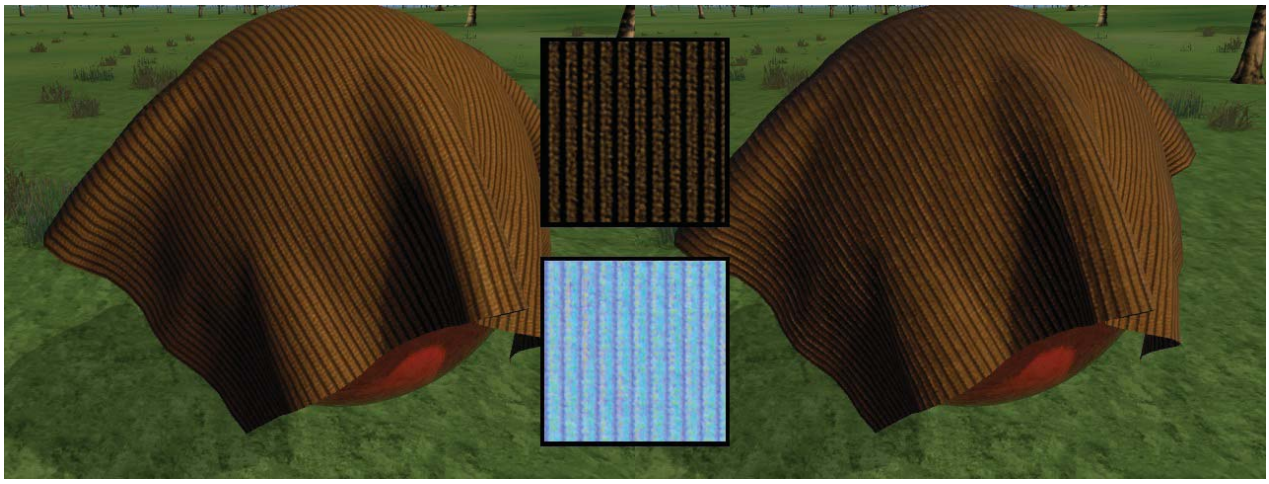

Abbildung 5.6: Links: Rendering mit einfachem Texture Mapping.
Rechts: Rendering mit Normal Mapping unter Verwendung der geschätzten Parameter inkl. Specular-Anteil.
Mitte oben: Ambient Map. Mitte unten: Normal Map

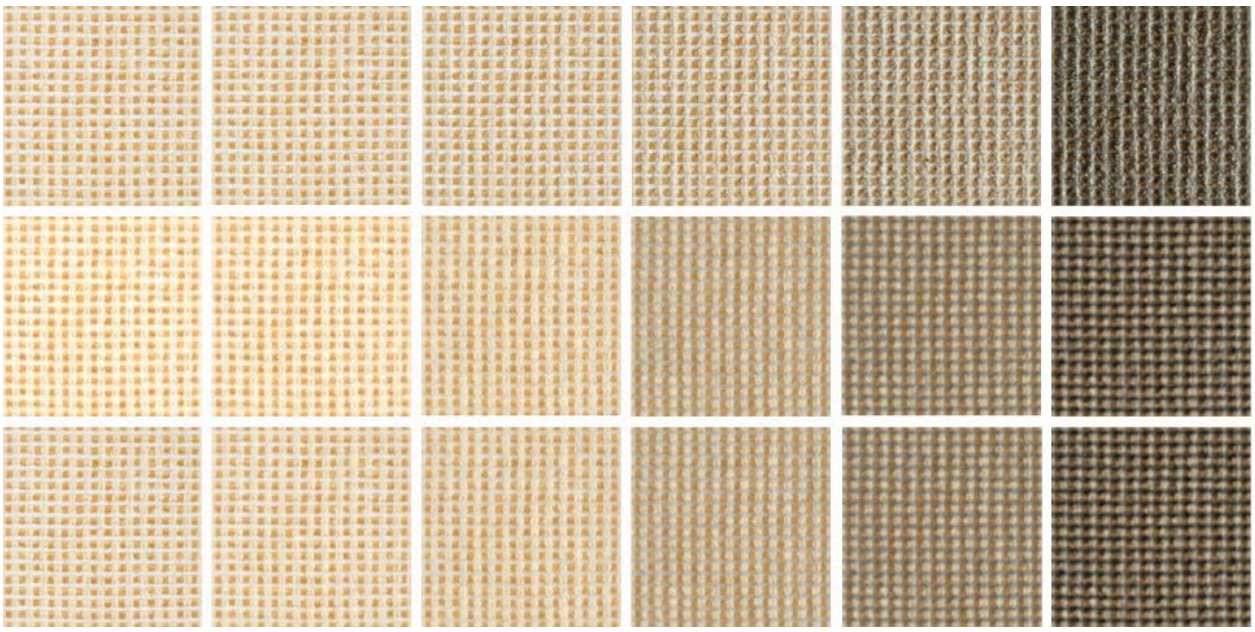

Abbildung 5.7: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Polster“.
Mitte: Rekonstruierte Samples, die mittels geschätzter Parameter ohne Specular-Anteil berechnet wurden.
Unten: Rekonstruierte Samples, die mittels geschätzter Parameter mit Specular-Anteil berechnet wurden.

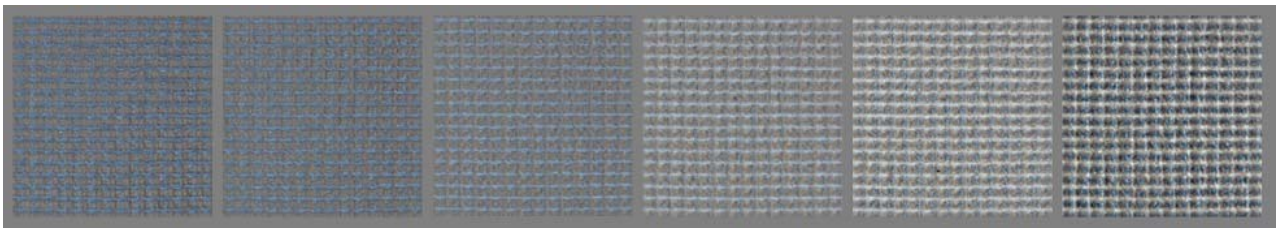

Abbildung 5.8: Die Gesamtabweichung der Schätzung vom Original für alle 81 Samples, die hier auf neutralem Grau dargestellt ist, beträgt für das Material „Polster“ rund 0,7467%.

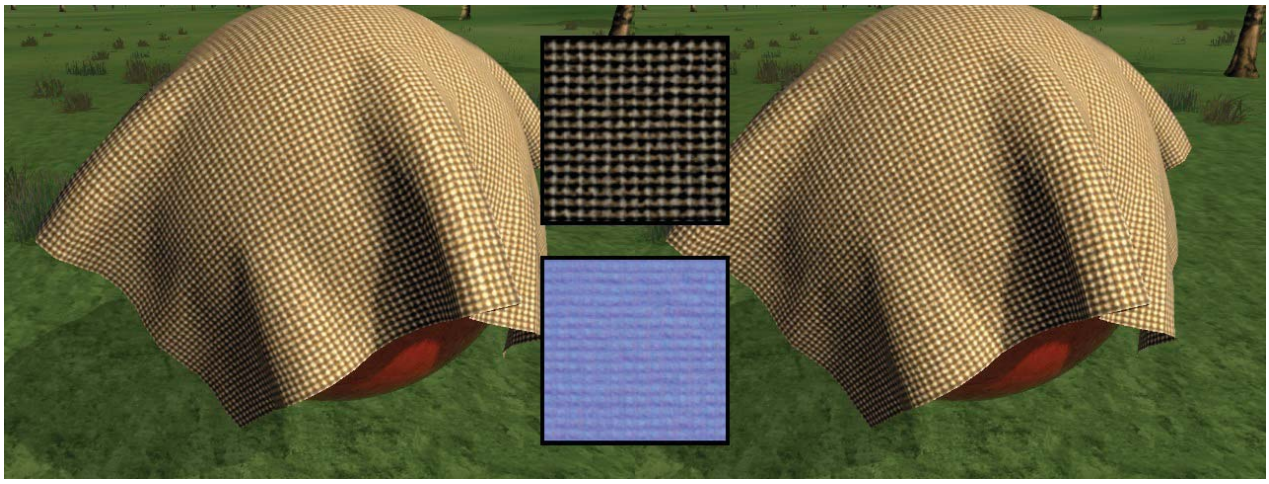

Abbildung 5.9: Links: Rendering mit einfachem Texture Mapping.
Rechts: Rendering mit Normal Mapping unter Verwendung der geschätzten Parameter inkl. Specular-Anteil.
Mitte oben: Ambient Map. Mitte unten: Normal Map

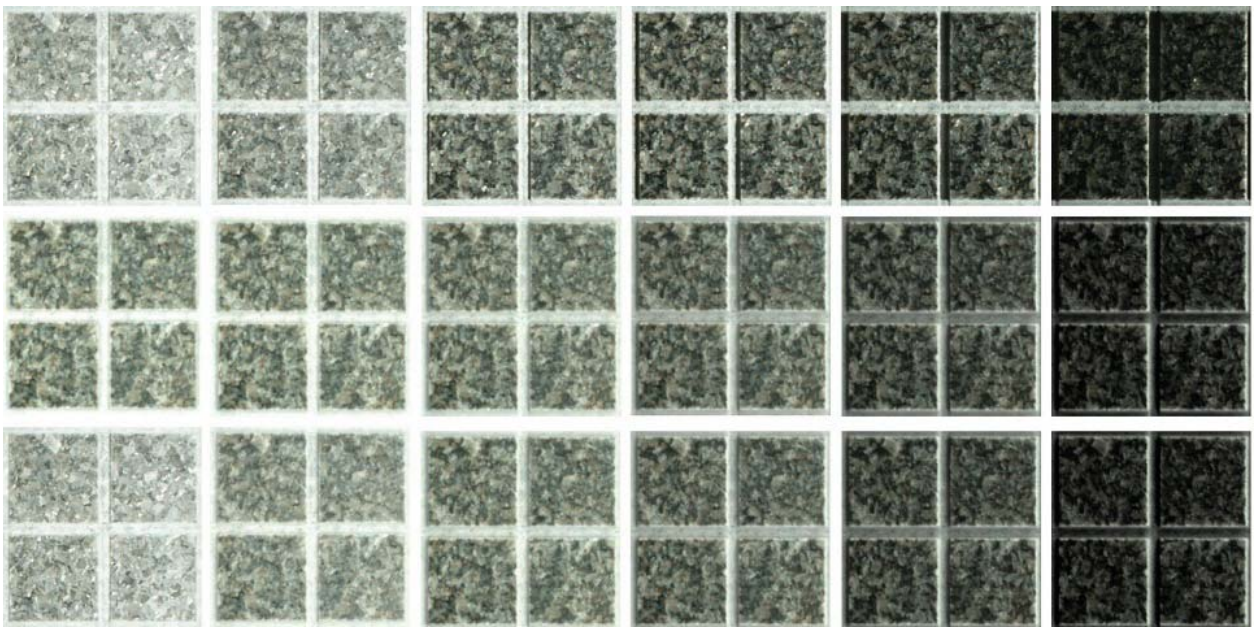

Abbildung 5.10: Oben: Original-Samples aus der BTF-Datenbank Bonn für das Material „Granit“.
Mitte: Rekonstruierte Samples, die mittels geschätzter Parameter ohne Specular-Anteil berechnet wurden.
Unten: Rekonstruierte Samples, die mittels geschätzter Parameter mit Specular-Anteil berechnet wurden.

Abbildung 5.11: Die Gesamtabweichung der Schätzung vom Original für alle 81 Samples, die hier auf neutralem Grau dargestellt ist, beträgt für das Material „Granit“ rund 0,8473%.

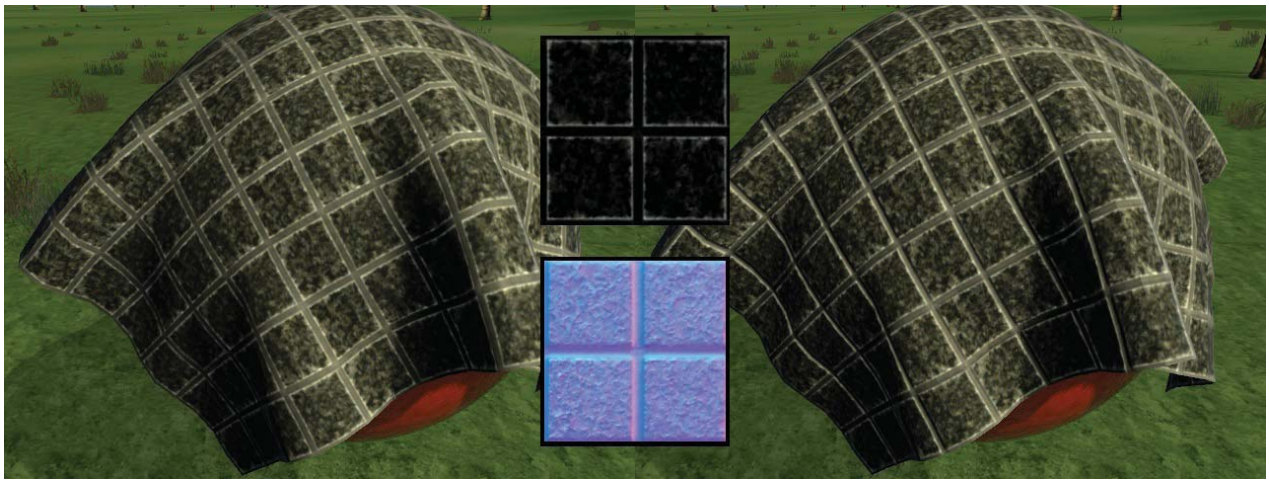

Abbildung 5.12: Links: Rendering mit einfachem Texture Mapping.
Rechts: Rendering mit Normal Mapping unter Verwendung der geschätzten Parameter inkl. Specular-Anteil.
Mitte oben: Ambient Map. Mitte unten: Normal Map

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Berlin, den 4. April 2011 Bojko Heinrich