LightSkin: Globale Echtzeitbeleuchtung für Virtual und Augmented Reality

Dissertation zur Erlangung des Doktorgrades Doktoringenieur (Dr.-Ing.)

von Philipp Lensing, M. Sc.

Eingereicht am 7. Juli 2014 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau.

Gutachter: Prof. Dr. Wolfgang Broll (Betreuer)

Prof. Dr. Beat Brüderlin

Jun.-Prof. Dr. Thorsten Grosch

urn:nbn:de:gbv:ilm1-2014000229

Abstract

In nature, each interaction of light is bound to a global context. Thus, each observable natural light phenomenon is the result of global illumination. It is based on manifold laws of absorption, reflection, and refraction, which are mostly too complex to simulate given the real-time constraints of interactive applications. Therefore, many interactive applications do not support the simulation of those global illumination phenomena yet, which results in unrealistic and synthetic-looking renderings. This unrealistic rendering becomes especially a problem in the context of virtual reality and augmented reality applications, where the user should experience the simulation as realistic as possible.

In this thesis we present a novel approach called LightSkin that calculates global illumination phenomena in real-time. The approach was especially developed for virtual reality and augmented reality applications satisfying several constraints coming along with those applications. As part of the approach we introduce a novel interpolation scheme, which is capable to calculate realistic indirect illumination results based on a few number of supporting points, distributed on model surfaces. Each supporting point creates its own proxy light sources, which are used to represent the whole indirect illumination for this point in a compact manner. These proxy light sources are then linearly interpolated to obtain dense results for the entire visible scene. Due to an efficient implementation on GPU, the method is very fast supporting complex and dynamic scenes. Based on the approach, it is possible to simulate diffuse and glossy indirect reflections, soft shadows, and multiple subsurface scattering phenomena without neglecting filigree surface details. Furthermore, the method can be adapted to augmented reality applications providing mutual global illumination effects between dynamic real and virtual objects using an active RGB-D sensor device. In contrast to existing interactive global illumination approaches, our approach supports all kinds of animations, handling them more efficient, not requiring extra calculations or leading to disturbing temporal artifacts.

This thesis contains all information needed to understand, implement, and evaluate the novel LightSkin approach and also provides a comprehensive overview of the related field of research.

Kurzfassung

In der Natur ist jede Interaktion des Lichts mit Materie in einen globalen Kontext eingebunden, weswegen alle natürlichen Beleuchtungsphänomene in unserer Umwelt das Resultat globaler Beleuchtung sind. Diese basiert auf der Anwendung mannigfaltiger Absorptions-, Reflexions- und Brechungsgesetze, deren Simulation so komplex ist, dass interaktive Anwendungen diese nicht in wenigen Millisekunden berechnen können. Deshalb wurde bisher in vielen interaktiven Systemen auf die Abbildung von solchen globalen Beleuchtungsphänomenen verzichtet, was jedoch zu einer unrealistischen und synthetisch-wirkenden Darstellung führte. Diese unrealistische Darstellung ist besonders für die Anwendungsfelder Virtual Reality und Augmented Reality, bei denen der Nutzer eine möglichst realitätsnahe Simulation erfahren soll, ein gewichtiger Nachteil.

In dieser Arbeit wird das LightSkin-Verfahren vorgestellt, das es erlaubt, globale Beleuchtungsphänomene in einer Echtzeitanwendung darzustellen. Das Verfahren wurde speziell für die Anwendungsfelder Virtual Reality und Augmented Reality entwickelt und erfüllt spezifische Anforderungen, die diese an eine Echtzeitanwendung stellen. Bei dem Verfahren wird das indirekte Licht durch eine geringe Anzahl von Punktlichtquellen (Proxy-Lichtquellen) repräsentiert, die für eine lose Menge von Oberflächenpunkten (Caches) berechnet und anschließend über die komplette sichtbare Szene interpoliert werden. Diese neue Form der Repräsentation der indirekten Beleuchtung erlaubt eine effiziente Berechnung von diffusen und glänzenden indirekten Reflexionen, die Abbildung von weichen Schatten und die Simulation von Multiple-Subsurface-Scattering-Effekten in Echtzeit für komplexe und voll dynamische Szenen. Ferner wird gezeigt, wie das Verfahren modifiziert werden kann, um globale Lichtwechselwirkungen zwischen realen und virtuellen Objekten in einer Augmented-Reality-Anwendung zu simulieren. Im Gegensatz zu den meisten existierenden Echtzeitverfahren zur Simulation von globalen Beleuchtungseffekten benötigt der hier vorgestellte Ansatz keine aufwändigen zusätzlichen Berechnungen bei Animationen und erzeugt darüber hinaus für diese keine visuellen Artefakte.

Diese Arbeit enthält alle Informationen, die zum Verständnis, zur Implementierung und zur Evaluation des LightSkin-Verfahrens benötigt werden und gibt darüber hinaus einen umfassenden Überblick über das Forschungsfeld.

Danksagung

Die Erstellung einer Dissertation ist ein Kraftakt, der ohne die Hilfe besonderer Menschen nicht möglich erscheint. Deshalb möchte ich die Gelegenheit nutzen und den Menschen meinen Dank aussprechen, ohne die diese Arbeit in dieser Form nicht möglich gewesen wäre.

Zuallererst möchte ich meinem Betreuer Wolfgang Broll danken, der mir überhaupt erst die Möglichkeit zur Promotion eröffnet hat. Ich hatte großes Glück, einen Betreuer wie Wolfgang zu haben, der mir viel Freiraum bei der Wahl eines passenden Forschungsfeldes gelassen hat und der darüber hinaus dafür gesorgt hat, dass ausreichend Zeit für die eigenen Forschungen zur Verfügung stand. Wolfgangs unkomplizierte Art und seine Fähigkeit, zu jeder Tages- und Nachtzeit konstruktive Kritik einbringen zu können, waren von unschätzbarem Wert für mich. Durch Wolfgangs Betreuung habe ich viel gelernt und ohne diese wäre diese Arbeit undenkbar gewesen.

Natürlich möchte ich auch meinen beiden Gutachtern Beat Brüderlin und Thorsten Grosch für deren Korrekturvorschläge und die unkomplizierte Hilfe bei der Anfertigung der Dissertation danken. Diese Arbeit entstand unter besonderen Umständen, die eine hohe Flexibilität von den Gutachtern erforderte. Ich bin froh, mit Beat und Thorsten Gutachter gefunden zu haben, die außer ihrer fachlichen Kompetenz ein hohes Maß an Enthusiasmus für das Thema dieser Arbeit aufbringen konnten, so dass ich bei der Anfertigung dieser Dissertation stets ein gutes Gefühl hatte.

Ferner möchte ich Cornelia Kolbeck und meinem Bruder Frederic Lensing für ihre Korrekturen danken. Die Korrekturen von Cornelia und Frederic sind von hohem Wert für mich und diese Arbeit. Die zwei haben viel ihrer Zeit geopfert, um mir schnelle und gründliche Korrekturen geben zu können.

Auch Tobias Franke möchte ich meinen Dank aussprechen, der so freundlich war, mir seine Implementierungen der Light-Propagation-Volumes, sowie des Voxel-Cone-Tracings zu überlassen, damit ich diese zur Evaluation nutzen konnte.

Zu guter Letzt möchte ich noch einer ganz besonderen Person danken: meiner lieben Frau Isabel. Isabel hat wahrscheinlich genau so viel Zeit in diese Dissertation gesteckt, wie ich selbst. Sei es nun wegen der zahlreichen Korrekturen oder wegen der aufmunternden Worte, die mich so manches Mal daran gehindert haben, den Kopf in den Sand zu stecken. Ohne Isabel wäre diese Arbeit nicht möglich gewesen. Ohne Isabel ist eigentlich nichts für mich möglich.

Vielen Dank Euch allen!

Inhaltsverzeichnis

1 Einleitung

1.1 Einführung

Alle natürlichen Beleuchtungsphänomene in unserer Umwelt sind das Resultat globaler Beleuchtung. In der Natur ist jede Interaktion des Lichts mit Materie in einen globalen Kontext eingebunden, weswegen eine Vielzahl von natürlichen optischen Phänomenen nicht durch lokale Beleuchtungsmodelle abgebildet werden kann. Im Vergleich zu lokalen Beleuchtungsmodellen, bei denen nur ein isolierter Oberflächenpunkt und eine Lichtquelle im Vakuum betrachtet werden, wird bei globalen Beleuchtungsmodellen das Wechselspiel des Lichts zwischen allen Elementen einer Szene berücksichtigt. Dies führt zu so wichtigen optischen Phänomenen wie zum Beispiel Schattenwurf, Spiegelungen, Streuung durch Nebel oder indirekte Reflexionen. Aber auch subtilere Phänomene, wie das Erscheinungsbild von menschlicher Haut oder eines Glases voll Milch, sind das Resultat globaler Lichtwechselwirkungen.

Die Simulation von globalen Beleuchtungseffekten ist eines der spannendsten, aber auch eines der komplexesten Themen der Computergrafik. Sie basiert auf der Anwendung mannigfaltiger Absorptions-, Reflexions- und Brechungsgesetze. Die Anwendung dieser Gesetze zeigt, dass natürliches Licht bei der Interaktion mit einem Material oder Medium in verschiedene Richtungen und mit verschiedenen Spektren verteilt wird. Hierdurch entsteht ein Komplexitätsproblem für die Bildsynthese, denn bereits in einer moderat komplexen Szene (einfache Materialien und Oberflächenstrukturen) wird das Licht so viele Interaktionen durchlaufen, dass dessen erschöpfende Simulation unmöglich ist. Aus diesem Grund finden in der Computergrafik approximative Verfahren Anwendung, die die Ausbreitung des Lichts auf strahlenoptische Eigenschaften und Phänomene reduzieren. Dabei erreichen diese Verfahren eine beeindruckende Realitätsnähe, wie aktuelle Computeranimationsfilme, wie zum Beispiel James Cameron‘s „Avatar“, zeigen. Diese Beleuchtungsverfahren benötigen jedoch trotz ihrer approximativen Eigenschaften zur Simulation sehr viel Rechenzeit. In der Regel dauert die Erzeugung eines Bildes eines Animationsfilms mehrere Stunden auf verteilten Systemen.

Für die interaktive Computergrafik, worunter auch Virtual und Augmented Reality (VR/AR) fallen, sind solche Erzeugungszeiten zu langsam. Hier muss ein Bild in weniger als vierzig Millisekunden (25 Bilder pro Sekunde) synthetisiert sein, damit die Interaktion mit dem Anwender nicht ins Stocken gerät. Dies sind schwierige Anforderungen, die über viele Jahre dafür gesorgt haben, dass globale Beleuchtungseffekte in interaktiven Umgebungen nicht anwendbar waren. Stattdessen kamen empirische lokale Beleuchtungsmodelle zur Anwendung [Pho75,Bli77]. Globale Beleuchtungseffekte wurden bestenfalls in Form einfacher Phänomene wie Schlagschatten [Cro77,Wil78] repräsentiert. Dies ist in Bezug auf interaktive Umgebungen ein besonderer Nachteil, denn globale Beleuchtungseffekte dienen dem Anwender nicht nur als optische Finesse, sondern sie sorgen auch dafür, dass geometrische Zusammenhänge korrekt erkannt werden [TSS*98]. Gilchrist [Gil79] hat beispielsweise gezeigt, dass die räumliche Orientierung und die Wahrnehmung von Farben in einer Szene maßgeblich von den Reflexionen zwischen deren Objekten abhängt.

Die schnelle Entwicklung moderner Grafikkarten hat dazu geführt, dass globale Beleuchtungseffekte auch in interaktiven Anwendungen dargestellt werden können. Jedoch blieb die Wechselwirkung des Lichts zwischen den Objekten lange Zeit auf statische Szenen beschränkt. Die Objekte erhalten hier

eine zusätzliche Textur, in die die Schatten und Reflexionen eingezeichnet sind. Die Erzeugung dieser Texturen (Light-Maps) bleibt aufwändigen Offline-Simulationen vorbehalten. Dynamische Szenen oder betrachtungsabhängige Reflexionen können mit diesem Ansatz nicht realisiert werden. Wegen der fehlenden Unterstützung von Animationen handelt es sich im eigentlichen Sinne nicht um ein interaktives Verfahren. Nichtsdestotrotz ist dieses Verfahren ein praktikabler Weg, um globale Beleuchtungseffekte ansatzweise in einer Echtzeitumgebung darzustellen und wird auch heute noch in Computerspielen eingesetzt.

Mit der Einführung von programmierbaren Grafikkarten wurden erstmals Verfahren zur globalen Beleuchtung möglich, die dynamische Veränderungen in der Szene interaktiv widerspiegeln können. Durch die Superpositionseigenschaft des Lichts sind globale Beleuchtungsansätze, wie zum Beispiel Ray-Tracing [Whi80] oder Radiosity [GTG*84], prädestiniert für die Parallelisierung und profitieren unmittelbar von der parallelen Architektur moderner Grafikkarten. Jedoch ist es selbst mit heutzutage üblichen Grafikprozessoren (Graphic-Processing-Units, GPUs) nicht möglich, Verfahren, die im Bereich der Offline-Bildsynthese Verwendung finden, in wenigen Millisekunden auszuführen. Aus diesem Grund entstand eine Vielzahl an approximativen Verfahren, die speziell für interaktive Umgebungen entwickelt wurden. Jedes dieser Verfahren besitzt spezifische Nachteile und Einschränkungen. Allgemein kann man die folgenden typischen Einschränkungen nennen:

- Einschränkung der Lichtausbreitung,
- Einschränkung der Materialkomplexität,
- Einschränkung der Szenenkomplexität,
- Einschränkung von Animationen.

Dabei können die Einschränkungen, je nach Ansatz, verschieden stark ausgeprägt sein. Manche Verfahren erlauben zum Beispiel nur direktes Licht von unendlich weit entfernten Lichtquellen oder beschränken Reflexionen auf niederfrequente Anteile. Andere Verfahren können spiegelnde und transparente Materialien nicht abbilden. Komplexe und große Szenen sind wegen der hohen Speicheranforderungen bei vielen Ansätzen ein Problem. Animationen wiederum können teilweise auf die reine Veränderung der direkten Lichtparameter beschränkt sein oder die Verfahren erzeugen flackernde Artefakte bei Modellanimationen. Es existieren noch weitere Einschränkungen, auf die in den folgenden Kapiteln genauer eingegangen wird.

Bei den meisten interaktiven Verfahren ist die physikalische Korrektheit der Simulation von untergeordneter Bedeutung. In der Anwendung ist es stattdessen häufig wichtiger, dass die berechneten Ergebnisse dem Erwartungsbild des Nutzers entsprechen und sie kohärent präsentiert werden. Dies ist die Grundlage für viele Vereinfachungen, die in interaktiven globalen Beleuchtungsverfahren zur Anwendung kommen.

Gegenwärtig existiert kein interaktives globales Beleuchtungsverfahren, welches man uneingeschränkt für alle Anwendungsfälle empfehlen kann. Insbesondere für das Anwendungsfeld Virtual und Augmented Reality sind die meisten Verfahren eher ungeeignet, so dass es sinnvoll ist, dieses Anwendungsfeld durch einen dedizierten Ansatz abzudecken. Das LightSkin-Verfahren, das in dieser Dissertation vorgestellt wird, ist ein solcher Ansatz.

1.2 Motivation

Durch die Darstellung von globalen Wechselwirkungen des Lichts erscheint eine Szene realistischer. Für Virtual-Reality-Anwendungen kann so die Immersion verbessert werden, da die Szene für den Nutzer plausibler wirkt. Die Immersion beschreibt hier die Identifikation mit dem Avatar in der virtuellen Szene, also wie stark sich ein Nutzer gedanklich von der Realität löst und die virtuelle Szene akzeptiert. Für Augmented-Reality-Anwendungen kann die nahtlose Integration des virtuellen Contents in die reale Szene verbessert werden, denn die photometrische Interaktion zwischen der realen und der virtuellen Szene erlaubt es, dass Objekte aus den verschiedenen Welten Lichtreflexionen austauschen. Hierdurch werden Effekte wie Color-Bleeding und gegenseitige Schattierung möglich.

Beide Anwendungsfelder stellen spezielle Anforderungen an die zu nutzenden Verfahren:

1. **Dynamik und Interaktivität**: Animationen müssen im vollen Umfang unterstützt werden und sollten keine Artefakte hervorrufen. Um Simulator-Sickness [HPT*06] zu vermeiden, sollten die Verfahren keine temporäre Kohärenz durch Bildkompositionen über mehrere konsekutive Frames hinweg erzeugen. Die Anforderung der Dynamik schließt auch Animationen in der Realumgebung bei einer Augmented-Reality-Anwendung (AR) ein. Hier müssen die Verfahren Veränderungen erfassen und direkt im Beleuchtungssystem widerspiegeln.
2. **Komplexität**: Szenen in Virtual-Reality-Anwendungen (VR) können mitunter sehr groß und komplex sein. Das Verfahren sollte solche Szenen unterstützen. Insbesondere sollte der Speicherbedarf des Verfahrens gering sein, um Out-of-Core-Ansätze zur Darstellung [WDS05] zu unterstützen.
3. **Darstellungsrate und Auflösung**: Für die stereoskopische Darstellung werden Bildwiederholungsraten von 60 bis 120 Hz benötigt. Darüber hinaus verwenden VR-Systeme häufig große mehrseitige Projektionen zur Darstellung (CAVE-Systeme [CSD*92]), die hohe Bildauflösungen erfordern.
4. **Plausibilität**: Die berechneten Ergebnisse müssen den Erwartungen des Nutzers entsprechen. Physikalische Korrektheit ist nicht zwingend erforderlich, jedoch wünschenswert.

Es existieren nur wenige Verfahren, die diese Anforderungen erfüllen. Dabei weisen diese wenigen Verfahren spezifische Nachteile in bestimmten Bereichen auf, so dass keines von diesen für das Anwendungsfeld in Gänze in Frage kommt. Deshalb wurde ein neues globales Beleuchtungsverfahren für interaktive Umgebungen entwickelt, das die hier beschriebenen Anforderungen nahezu vollständig erfüllt.

1.3 Wissenschaftlicher Beitrag

Diese Dissertation hat den Anspruch, den gegenwärtigen Stand der Wissenschaft im Bereich der interaktiven globalen Beleuchtung zu erweitern. Dies wird erreicht, indem ein neues globales Echtzeit-Beleuchtungsverfahren namens LightSkin für Virtual-Reality- und Augmented-Reality-Anwendungen vorgestellt wird, welches eine Weiterentwicklung des Instant-Radiosity-Ansatzes (IR) von Keller [Kel97] darstellt. Das LightSkin-Verfahren weist gegenüber existierenden globalen Echtzeit-Beleuchtungsverfahren die folgenden Vorteile bzw. Neuerungen auf:

- Abbildung von physikalisch plausiblen diffusen indirekten Reflexionen,

- Abbildung von optisch plausiblen glänzenden indirekten Reflexionen,
- Abbildung von weichen Schatten,
- Abbildung von Multiple-Subsurface-Scattering-Effekten,
- deutliche Reduzierung von temporären Inkohärenzen bei Animationen,
- Unterstützung von Animationen ohne zusätzlichen Rechenaufwand,
- Reduzierung typischer Singularitäten von Instant-Radiosity-Verfahren,
- geringe Speicheranforderungen,
- geringe Anforderungen an das Shader-Modell (Shader-Model 3.0 ist ausreichend) und
- eine einfache Adaption für Augmented-Reality-Anwendungen zur photometrischen Interaktion zwischen dynamischen realen und virtuellen Szenen unter Verwendung eines RGB-D-Sensors.

Die Grundlagen für diese Errungenschaften bilden ein neues Interpolationsverfahren und dessen effiziente Implementierung auf der Grafikkarte. Das Verfahren erlaubt es, die indirekte Beleuchtung für eine geringe Menge loser Oberflächenpunkte im Modellraum kompakt in Form von analytischen Punktlichtquellen abzubilden, die anschließend für alle sichtbaren Oberflächen interpoliert werden können. Ferner wird ein neuer AR-Ansatz vorgestellt, der einen RGB-D-Sensor nutzt, um die reale Umgebung in Echtzeit in ein geometrisches Modell zu wandeln, welches dann für die globale Beleuchtung mit dem neuen Verfahren verwendet werden kann.

Im Rahmen dieser Dissertation wurden die Teilergebnisse durch die folgenden internationalen Publikationen dokumentiert:

- Philipp Lensing and Wolfgang Broll, 2013, LightSkin: Real-Time Global Illumination for Virtual and Mixed Reality, in: Proceedings of Joint Virtual Reality Conference of EGVE - EuroVR, Eurographics Association, pp. 17-24, DOI=10.2312/EGVE.JVRC13.017-024.
- Philipp Lensing and Wolfgang Broll, 2013, Efficient shading of indirect illumination applying reflective shadow maps, in: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '13), Stephen N. Spencer (Ed.), ACM, New York, NY, USA, pp. 95-102. DOI=10.1145/2448196.2448211.
- Philipp Lensing and Wolfgang Broll, 2012, Instant indirect illumination for dynamic mixed reality scenes, in: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 109-118, 5-8 Nov. 2012, DOI=10.1109/ ISMAR.2012.6402547.
- Philipp Lensing and Wolfgang Broll, 2011, Fusing the real and the virtual: A depth-camera based approach to Mixed Reality, in: Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 261-262, 26-29 Oct. 2011, DOI=10.1109/ ISMAR.2011.6143892.

1.4 Aufbau der Arbeit

Aus der Tatsache heraus, dass diese Arbeit in Deutsch abgefasst wurde, obwohl die meisten wissenschaftlichen Fachartikel in Englisch verfasst sind, folgen einige Schwierigkeiten bei der Eindeutschung etablierter englischer Fachausdrücke. Nach Auffassung des Autors macht es in den meisten Fällen wenig Sinn, eine deutsche Entsprechung englischer Fachausdrücke zu verwenden, sofern diese nicht gebräuchlich ist. Dies hat zur Folge, dass die Texte dieser Arbeit mit nicht zu vermeidenden Anglizismen durchsetzt sind. Auch wenn dies einem Sprachwissenschaftler möglicherweise missfallen

wird, ist es für die fachliche Ausprägung dieser Dissertation von besonderer Bedeutung, da die verwendeten Begriffe beim Leser die richtigen Konnotationen hervorrufen sollen. So wird zum Beispiel bei der indirekten Beleuchtung häufig von Color-Bleeding gesprochen, während das deutsche Pendant Farbbluten in der indirekten Beleuchtung so gut wie nicht verwendet wird, stattdessen aber beim Farbdruck.

Auch bei mathematischen Notationen wurde von deutschen Konventionen Abstand genommen. Dies äußert sich insbesondere in der Verwendung von fettgedruckten Buchstaben für Vektoren, anstatt der Pfeilnotation, wie sie in deutschen Artikeln üblich ist.

Inhaltlich ist die Arbeit wie folgt strukturiert:

In Kapitel 2 werden die wichtigsten Grundlagen und Definitionen zur globalen Beleuchtung und Augmented Reality erläutert. Dabei wird der Fokus auf die Bereiche gelegt, die zum Verständnis dieser Arbeit von Bedeutung sind.

Kapitel 3 liefert einen Überblick über die verschiedenen interaktiven Verfahren, mit denen globale Beleuchtungseffekte realisiert werden können. Dies beinhaltet auch eine Klassifizierung der Verfahren, um eine spätere Abgrenzung zum in dieser Arbeit entwickelten Verfahren zu erleichtern. Es werden ebenfalls Verfahren diskutiert, die bereits in Augmented-Reality-Umgebungen zur Anwendung kommen.

In Kapitel 4 werden einige konzeptionelle Vorüberlegungen zum LightSkin-Verfahren erläutert, während in Kapitel 5 das Verfahren im Detail beschrieben wird. Das Kapitel 5 beinhaltet die Berechnung der indirekten Reflexionen, die Erzeugung von weichen Schatten, sowie die Simulation von Streuungsprozessen unterhalb von Modelloberflächen (Subsurface-Scattering). Im darauf folgenden Kapitel 6 wird die Implementierung des Verfahrens für die Grafikkarte beschrieben und eine Evaluation bezüglich der Qualität und des Laufzeitverhaltens vorgenommen.

Das LightSkin-Verfahren lässt sich nicht unmittelbar in den Kontext einer Augmented-Reality-Anwendung übertragen, sofern die Realumgebung nicht vollständig als dreidimensionales Modell vorliegt. Die Anpassung des Verfahrens, sowie Details zur Modellbildung der realen Szene, basierend auf aktiven Sensoren, sind Gegenstand von Kapitel 7.

Im letzten Kapitel folgen eine kritische Diskussion und eine Bewertung des LightSkin-Verfahrens. Da die Entwicklung des Verfahrens zum Zeitpunkt der Erstellung dieser Arbeit noch nicht abgeschlossen ist, wird darüber hinaus ein Ausblick für zukünftige Erweiterungen diskutiert.

2 Grundlagen

2.1 Globale Beleuchtung

In der Einleitung wurde bereits erwähnt, dass bestimmte optische Phänomene, wie zum Beispiel Schlagschatten, indirekte Reflexionen oder Spiegelungen, ohne die Betrachtung globaler Lichtwechselwirkungen nicht möglich sind. Sehr viele globale Beleuchtungssysteme beschränken sich bei der Simulation auf die Berechnung von Reflexionen im Vakuum. Wenn kein Medium im System berücksichtigt werden muss, reichen die Prinzipien aus Kapitel 2.1.2 und 2.1.3 aus, um globale Reflexionen zu berechnen. Wenn hingegen ein Medium vorhanden ist, zum Beispiel Nebel, sind auch die darauf folgenden Unterkapitel von Bedeutung. Luft wird normalerweise als Vakuum angenommen und erst dann als Medium simuliert, wenn sie zum Beispiel durch hohe Feuchtigkeit zu Nebel wird.

2.1.1 Photometrische Grundgrößen

In dieser Dissertation werden Kenntnisse über photometrische Grundgrößen vorausgesetzt. Da jedoch in der Literatur häufig radio- und photometrische Grundgrößen vermischt werden und somit die Schreibweise häufig inkonsistent ist, sollen kurz die wichtigsten Grundgrößen für diese Arbeit und deren Notationen eingeführt werden. Diese Einführung beinhaltet keine umfangreichen Erläuterungen zu den Zusammenhängen zwischen den Größen, hierfür sei auf [PH10] verwiesen.

Der Lichtstrom wird in dieser Arbeit durch das Zeichen ϕ gekennzeichnet. Er ist die zeitliche Ableitung der Lichtmenge Q: $\phi = dQ/dt$. Die radiometrische Entsprechung dieser Größe ist die Strahlungsleistung (Radiant Flux).

Die Lichtstärke I definiert den differentiellen Lichtstrom $d\phi$ pro differentiellem Raumwinkel $d\boldsymbol{\omega}$: $I = d\phi/d\boldsymbol{\omega}$. Der differentielle Raumwinkel $d\boldsymbol{\omega}$ entspricht einer differentiellen Fläche auf der Einheitskugel, die um den Richtungsvektor $\boldsymbol{\omega}$ liegt, er ist wie folgt definiert: $d\boldsymbol{\omega} = sin\vartheta d\vartheta d\varphi$ (siehe Abbildung 2.2 für die Zuordnung der Winkelgrößen). Die radiometrische Entsprechung zur Lichtstärke ist die Strahlstärke (Radiant Intensity).

Die Beleuchtungsstärke wird in dieser Arbeit durch den Buchstaben E repräsentiert. Sie gibt an, wie viel differentieller Lichtstrom von einem differentiellen Flächenelement empfangen wird: $E = d\phi/dA$. Dagegen kennzeichnet die spezifische Lichtausstrahlung B den gesendeten Lichtstrom pro Flächenelement. Geht man davon aus, dass das Flächenelement nicht senkrecht zur Strahlungsrichtung liegt, gilt für die Beleuchtungsstärke: $E = \frac{d\phi}{dAcos\vartheta}$, wobei ϑ der Winkel zwischen der Oberflächennormalen und der (inversen) Strahlungsrichtung ist. Die radiometrische Entsprechung der Beleuchtungsstärke ist die Bestrahlungsstärke (Irradiance) und die der spezifischen Lichtausstrahlung ist die spezifische Ausstrahlung (Radiosity).

Die Leuchtdichte wird in dieser Dissertation mit dem Buchstaben L notiert. Sie beschreibt den differentiellen Lichtstrom, der von einer differentiellen Fläche in einen differentiellen Raumwinkel entsendet bzw. aus einem differentiellen Raumwinkel empfangen wird: $L = \frac{d^2\phi}{dAd\boldsymbol{\omega}}$. Analog gilt für ein beliebig ausgerichtetes differentielles Flächenelement: $L = \frac{d^2\phi}{dAcos\vartheta d\boldsymbol{\omega}}$. In der Radiometrie steht der Leuchtdichte die Strahldichte (Radiance) gegenüber.

Prinzipiell sind alle hier aufgeführten Grundgrößen von der Wellenlänge des Lichts und somit vom betrachteten Frequenzspektrum des Lichts abhängig. Da photometrische Grundgrößen das Wahrnehmungsempfinden des menschlichen Sehapparats berücksichtigen, müssen radiometrische Größen durch spektrale Gewichtungsfunktionen in photometrische Größen umgerechnet werden:

$$X(\lambda) = K_m X_R(\lambda) V(\lambda), \quad X'(\lambda) = K'_m X_R(\lambda) V'(\lambda). \tag{2.1}$$

X bezeichnet eine photometrische und X_R eine radiometrische Grundgröße. Die Gewichtungsfunktionen $K_m V(\lambda)$ und $K'_m V'(\lambda)$basieren auf dem Helligkeitsempfinden des Sehapparats für hell- und dunkeladaptiertes Sehen. Dieses Wahrnehmungsempfinden reduziert das Spektrum des Lichts bei photometrischen Grundgrößen auf das sichtbare Spektrum zwischen 380nm und 780nm, welches für die Computergrafik von übergeordneter Bedeutung ist.

In dieser Arbeit wird die Wellenlänge des Lichts für die Berechnung der Grundgrößen nicht dediziert berücksichtigt, stattdessen enthält jede Grundgröße einen skalaren Wert, um jeweils die drei Primärvalenzen Rot, Grün und Blau des Monitors anzusteuern. Weitere farbmetrische Zusammenhänge bleiben in dieser Arbeit unberücksichtigt, wie dies bei den meisten interaktiven Beleuchtungsverfahren üblich ist [RDG*12].

In der folgenden Tabelle sind die Grundgrößen zusammengefasst:

Tabelle 2.1: Auflistung der photometrischen Grundgrößen und deren korrespondierenden radiometrischen Größen.

Photometrie			Radiometrie	
Größe	Buchstabe	Einheit	Größe	Einheit
Lichtmenge (Quantity of Light)	Q	Lumensek. ($lm\ s$)	Strahlungsenergie (Radiant Flux)	Joule (J)
Lichtstrom (Luminous flux)	ϕ	Lumen (lm)	Strahlungsleistung (Radiant Flux)	Watt (W)
Lichtstärke (Intensität) (Luminous Intensity)	I	Candela (cd)	Strahlstärke (Intensität) (Radiant Intensity)	Wsr^{-1}
Beleuchtungsstärke (Illuminance)	E	Lux (lmm^{-2})	Bestrahlungsstärke (Irradiance)	Wm^{-2}
Spez. Lichtausstrahlung (Luminosity)	B	Lux (lmm^{-2})	Spez. Ausstrahlung (Radiosity)	Wm^{-2}
Leuchtdichte (Luminance)	L	cdm^{-2}	Strahldichte (Radiance)	$Wsr^{-1}m^{-2}$

2.1.2 Globale Oberflächenreflexionen

Bei der globalen Beleuchtung wird der Weg des Lichts durch die Szene verfolgt. Es gilt, dass Licht nicht nur durch eine primäre Lichtquelle in die Szene abgestrahlt werden kann, sondern dass jede Oberfläche, die Licht empfängt, ebenfalls Licht in die Szene strahlen kann. Dabei kann die Oberfläche das Licht emittieren, reflektieren oder transmittieren. Wenn man davon ausgeht, dass Oberflächen nur als primäre Lichtquellen oder Reflektoren funktionieren, sie also Licht nicht transmittieren, dann lässt sich die Leuchtdichte L eines jeden Oberflächenpunktes $\boldsymbol{x}$ in Richtung $\boldsymbol{\omega}$ mathematisch durch Kajiya's Rendergleichung [Kaj86] beschreiben:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + L_r(\boldsymbol{x}, \boldsymbol{\omega}). \tag{2.2}$$

Die abgestrahlte Leuchtdichte L_o ergibt sich aus der emittierten Leuchtdichte L_e (Selbstleuchteigenschaft) und aus der Reflexion L_r des eingehenden Lichts. Die Stärke der Reflexion L_r ist dabei abhängig vom empfangenen Licht L_i im Punkt $\boldsymbol{x}$ und der bidirektionalen Reflektanzverteilungsfunktion (Bidirectional-Reflectance-Distribution-Function, BRDF) f_r des Oberflächenmaterials:

$$L_r(\boldsymbol{x}, \boldsymbol{\omega}) = \int_{\Omega^+} f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) L_i\,(\boldsymbol{x}, \boldsymbol{\omega_i})\,(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i}. \tag{2.3}$$

Ω^+ kennzeichnet die Hemisphäre, in die die Normale $N(x)$ des Oberflächenpunktes zeigt. Die Leuchtdichte, die aus Richtung $\boldsymbol{\omega_i}$ im Punkt $\boldsymbol{x}$ empfangen wird, ist L_i. Die Operation $(\cdot)^+$ entspricht dem Skalarprodukt, bei dem negative Werte abgeschnitten (auf null gesetzt) werden. Abbildung 2.1 stellt die geometrischen Zusammenhänge grafisch dar:

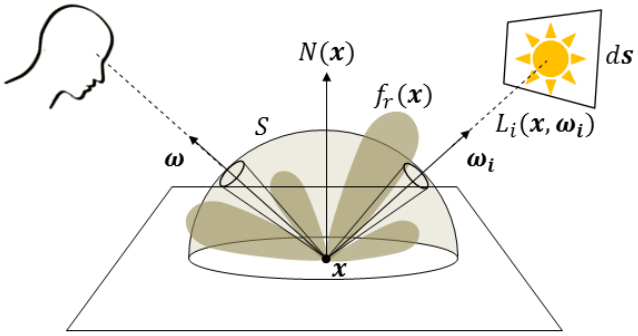

Abbildung 2.1: Geometrische Zusammenhänge zur Rendergleichung.

Die Gleichung (2.3) ist nur rekursiv lösbar, denn es gilt:

$$L_i\,(\boldsymbol{x}, \boldsymbol{\omega_i}) = L_o(r(\boldsymbol{x}, \boldsymbol{\omega_i}), -\boldsymbol{\omega_i}). \tag{2.4}$$

Hier liefert die Funktion r den nächsten Oberflächenpunkt zu $\boldsymbol{x}$, der innerhalb des Strahls in Richtung $\boldsymbol{\omega_i}$ liegt. In rekursiver Schreibweise ergibt sich also für die ausgestrahlte Leuchtdichte eines Oberflächenpunktes die folgende Gleichung:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega} f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) L_o\,(r(\boldsymbol{x}, \boldsymbol{\omega_i}), -\boldsymbol{\omega_i})\,(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i}. \tag{2.5}$$

Alternativ ist es auch möglich, das Integral über alle Oberflächen einer Szene zu bilden, um die resultierende Leuchtdichte im Punkt $\boldsymbol{x}$ zu bestimmen. Hierdurch wird eine Integration über die Hemisphäre nicht mehr nötig. Wenn man davon ausgeht, dass $d\boldsymbol{s}$ die differentielle Fläche eines Oberflächenpunktes $\boldsymbol{s} \neq \boldsymbol{x}$ ist und $\boldsymbol{\omega_i} = \frac{\boldsymbol{s}-\boldsymbol{x}}{\|\boldsymbol{s}-\boldsymbol{x}\|}$, dann gilt für den differentiellen Raumwinkel $d\boldsymbol{\omega_i}$ der folgende Zusammenhang (vgl. Abbildung 2.2):

$$d\boldsymbol{\omega_i} = sin\vartheta d\vartheta d\varphi = \frac{N(\boldsymbol{s}) \cdot -\boldsymbol{\omega_i}}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} d\boldsymbol{s}. \tag{2.6}$$

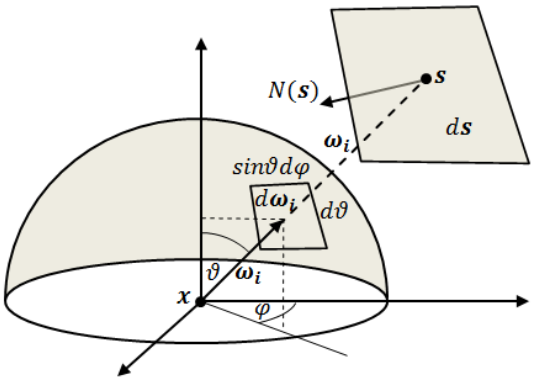

Abbildung 2.2: Zusammenhang zwischen differentiellem Raumwinkel und differentieller Fläche $\boldsymbol{ds}$.

Aus diesem Zusammenhang ergibt sich nun die folgende, alternative Form der Rendergleichung:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = L_e(\boldsymbol{x},\boldsymbol{\omega}) + \int_S f_r(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\omega_i}) L_o\,(\boldsymbol{s},-\boldsymbol{\omega_i}) \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} V(\boldsymbol{x},\boldsymbol{s})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{2.7}$$

Hier ist S die Menge aller Oberflächen. Da nicht für jeden Oberflächenpunkt $\boldsymbol{s}$ feststeht, ob er in Sichtverbindung zu $\boldsymbol{x}$ steht, muss eine zusätzliche Sichtbarkeitsprüfung durchgeführt werden. Diese Prüfung übernimmt die binäre Funktion $V(\boldsymbol{x},\boldsymbol{s})$, die Eins liefert, wenn eine Sichtverbindung besteht und Null, wenn nicht. Die Sichtbarkeit wurde in der vorherigen Form des Integrals implizit durch die Funktion r geprüft. In vielen Publikationen wird der Geometrieterm bzw. die Geometriefunktion G erwähnt, der den folgenden Teil der Gleichung zusammenfasst:

$$G(\boldsymbol{x},\boldsymbol{s}) = \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} V(\boldsymbol{x},\boldsymbol{s}). \tag{2.8}$$

Eine direkte und exakte Lösung der Rendergleichungen (2.5) und (2.7) ist wegen ihrer rekursiven Abhängigkeit nicht möglich. Jedoch kann man die Gleichung durch eine Neumann-Reihe bzw. durch die Neumann-Expansion formulieren. Hier wird die rekursive Eigenschaft des Integrals als unendliche Reihe dargestellt:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = \sum_{j=1}^{\infty} L_j(\boldsymbol{x},\boldsymbol{\omega}). \tag{2.9}$$

Wobei sich L_j wie folgt ergibt:

$$\begin{aligned} L_1(\boldsymbol{x},\boldsymbol{\omega}) &= L_e(\boldsymbol{x},\boldsymbol{\omega}), \\ L_{j+1}(\boldsymbol{x},\boldsymbol{\omega}) &= \int_S f_r(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\omega_i}) L_j\,(\boldsymbol{s},-\boldsymbol{\omega_i})\, G(\boldsymbol{x},\boldsymbol{s})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \end{aligned} \tag{2.10}$$

Die einzelnen Terme der Neumann-Reihe haben eine einfache optische Interpretation. Der Term L_1 stellt nur Oberflächen dar, die direktes Licht in die Szene emittieren. L_2 zeigt die erste (direkte) Reflexion von allen Oberflächen. L_3 beinhaltet die erste indirekte Reflexion und so weiter. Interaktive globale Beleuchtungsverfahren berechnen keine vollständige Lösung für die Neumann-Reihe. Stattdessen wird häufig nur eine Lösung bis zum Term L_3 berechnet ($L_1 + L_2 + L_3$), so dass ein ungewollter Energieverlust stattfindet. Praktisch ist der Energieverlust jedoch vernachlässigbar, denn aus

energetischen Gründen wird in jedem neuen Term die Intensität des Lichts niederfrequenter und schwächer (es gilt $L_j(\boldsymbol{x}, \boldsymbol{\omega}) \geq L_{j+1}(\boldsymbol{x}, \boldsymbol{\omega})$). Wenn die differentiellen Oberflächen $d\boldsymbol{s}$ in der Szene also viel Licht absorbieren (zum Beispiel in Wärme umwandeln), wird die Energie schnell abnehmen. Tabellion und Lamorlette [TL04] haben gezeigt, dass die erste indirekte Reflexion häufig ausreicht, um einen guten subjektiven Eindruck von der Szene zu erhalten.

2.1.3 Bidirektionale Reflektanzverteilungsfunktion (BRDF)

In der Rendergleichung (2.3) wurde bereits die bidirektionale Reflektanzverteilungsfunktion (Bidirectional Reflectance-Distribution-Function, BRDF) verwendet. Diese Funktion beschreibt die Reflexionseigenschaften eines Oberflächenpunktes in einer approximativen Weise. Trifft in der Natur Licht auf ein Objekt, so wird ein Teil des Lichts direkt vom Objekt reflektiert und ein Teil des Lichts dringt in das Objekt ein. Bei der Wanderung des Lichts durch das Objekt werden bestimmte Frequenzen absorbiert und zerstreut und ein Teil des Lichts verlässt die Oberfläche des Objekts wieder. Die wahrgenommene Farbgebung eines Objekts ergibt sich aus der Absorption und Streuung des Lichts unter der Oberfläche. Diese Prozesse können nur dann mit der BRDF abgebildet werden, wenn die Zerstreuung im Objekt auf einen geometrisch kleinen Bereich beschränkt ist. Dies ist zum Beispiel bei Plastik oder Metall der Fall. Wachs oder menschliche Haut lassen sich dagegen nicht hinreichend durch die BRDF beschreiben. Die BRDF bestimmt die Farbe des Objekts, sie ist also von der Wellenlänge des Lichts abhängig. In der interaktiven Computergrafik wird das Frequenzspektrum des Lichts über Farbkomponenten wie z. B. rot, grün und blau dargestellt, weswegen in den folgenden Formeln auf die Darstellung der Abhängigkeit zur Wellenlänge verzichtet wurde. Die BRDF liefert dann für jeden Farbkanal ein separates Ergebnis.

Physikalisch liefert die BRDF das Verhältnis der reflektierten differentiellen Leuchtdichte (dL_o) in Richtung eines Betrachters ($\boldsymbol{\omega}$) im Punkt $\boldsymbol{x}$ zur empfangenen differentiellen Beleuchtungsstärke dE aus Richtung ($\boldsymbol{\omega_i}$):

$$f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = \frac{dL_o(\boldsymbol{x}, \boldsymbol{\omega})}{dE_i(\boldsymbol{x}, \boldsymbol{\omega_i})} = \frac{dL_o(\boldsymbol{x}, \boldsymbol{\omega})}{dL(\boldsymbol{x}, \boldsymbol{\omega_i}) cos\vartheta d\boldsymbol{\omega_i}}. \tag{2.11}$$

ϑ ist hier der Winkel zwischen $N(\boldsymbol{x})$ und $\boldsymbol{\omega_i}$. Für die BRDF gilt Reziprozität ($f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = f_r(\boldsymbol{x}, \boldsymbol{\omega_i}, \boldsymbol{\omega})$) und das Prinzip der Energieerhaltung. Zwar kann das Ergebnis der BRDF zwischen Null und Unendlich liegen, aber bei der Reflexion darf keine zusätzliche Energie in das System gelangen. Es gilt deshalb:

$$\int_{\Omega} f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) cos\vartheta d\boldsymbol{\omega} \leq 1. \tag{2.12}$$

In der interaktiven Computergrafik beinhalten die verwendeten lokalen Beleuchtungsmodelle die BRDF häufig nur implizit. In diesen Modellen kann es vorkommen, dass die physikalischen Eigenschaften der BRDF verletzt werden. Dies geschieht, um die Berechnung der Beleuchtung zu beschleunigen. Bei der Wahrnehmung fällt die nicht stringente Auslegung der physikalischen Grundlagen nicht zwingend ins Gewicht, da die Beleuchtungsmodelle im Wesentlichen die Ideen der BRDF abbilden.

2.1.4 Brechung und Reflexion

Im vorhergehenden Kapitel wurde bereits darauf hingewiesen, dass auf einer Oberfläche einfallendes Licht direkt reflektiert wird oder in das Material eindringen kann. Die direkte Reflexion wird als spiegelnde Reflexion bezeichnet (Specular/Glossy Reflection) und das Eindringen des Lichts bezeichnet man als Brechung bzw. Transmission (Refraction/Transmission). Diffuse Reflexionen sind immer das Ergebnis einer Transmission (siehe vorheriges Unterkapitel). Besteht ein Objekt aus einem (semi)-transparenten Material, so wird das Licht, welches in das Objekt hineingebrochen wurde, nicht im selben Oberflächenpunkt austreten, in dem es eingetreten ist. Diese Objekte können also mit der BRDF nicht abgebildet werden, da diese nur die Reflexion in einem Punkt darstellt.

Möchte man in einem globalen Beleuchtungsverfahren transparente Objekte unterstützen, muss man bestimmen, wie viel empfangenes Licht in das Material transmittiert und wieviel reflektiert wird. Dieses Verhältnis kann durch den Fresnel-Reflexionsterm bestimmt werden. Die Fresnel-Reflexion F_r gibt an, welcher Bruchteil des eintreffenden Lichtstroms reflektiert wird:

$$d\phi_r = F_r(\vartheta_i)d\phi_i. \tag{2.13}$$

Der reflektierte differentielle Lichtstrom $d\phi_r$ ergibt sich aus dem Produkt des Fresnel-Terms mit dem eintreffenden differentiellen Lichtstrom $d\phi_i$. Der Wert der Funktion ist abhängig vom Winkel (ϑ_i) zwischen der Oberflächennormalen und dem Vektor, der in die Richtung des eintreffenden Lichts zeigt. Darüber hinaus ist der Fresnel-Term abhängig von der Polarisation und der Wellenlänge des Lichts. In interaktiven Systemen wird die Polarisation aber ignoriert und die Wellenlänge über Farbkomponenten abgebildet, so dass für diese Arbeit diese Abhängigkeiten von geringer Bedeutung sind.

Da das Prinzip der Energieerhaltung gilt, ergibt sich für den transmittierten Lichtstrom $d\phi_t$ der folgende Zusammenhang (ohne Absorptionsprozesse zu berücksichtigen):

$$d\phi_t = (1 - F_r(\vartheta_i))d\phi_i. \tag{2.14}$$

In der Computergrafik ist die zentrale Größe die Leuchtdichte. Das Verhältnis von eingehender Leuchtdichte L_i zur reflektierten Leuchtdichte L_r lässt sich berechnen, indem man den differentiellen Lichtstrom ersetzt ($L = \frac{d^2\phi}{cos\vartheta dAd\omega}$):

$$L_r \cos(\vartheta_r)\, dAd\omega_r = F_r(\vartheta_i)L_i \cos(\vartheta_i)\, dAd\omega_i. \tag{2.15}$$

Da bei einer Reflexion der Eingangswinkel gleich dem Ausgangswinkel ist ($\vartheta_r = \vartheta_i, d\omega_r = d\omega_i$), ergibt sich die einfache Abhängigkeit:

$$L_r = F_r(\vartheta_i)L_i. \tag{2.16}$$

Der Zusammenhang zur transmittierten Leuchtdichte ist nicht so trivial wie der zur reflektierten Leuchtdichte, denn durch die Brechung des Lichts wird auch der differentielle Raumwinkel verändert. Für die Herleitung wird abermals der Lichtstrom ersetzt:

$$L_t \cos(\vartheta_t)\, dAd\omega_t = (1 - F_r(\vartheta_i))L_i \cos(\vartheta_i)\, dAd\omega_i. \tag{2.17}$$

Da sich der differentielle Raumwinkel durch die Brechung verändert, wird er ausgeschrieben (siehe auch Gleichung (2.6)):

$$L_t \cos(\vartheta_t)\, dA sin\vartheta_t d\vartheta_t d\varphi_t = \big(1 - F_r(\vartheta_i)\big) L_i \cos(\vartheta_i)\, dA sin\vartheta_i d\vartheta_i d\varphi_i. \tag{2.18}$$

Für die Winkel ϑ_t und ϑ_i ergibt sich nach dem Snelliusschen Gesetz das folgende Verhältnis (Brechungsgesetz):

$$n_i sin\vartheta_i = n_t sin\vartheta_t. \tag{2.19}$$

Wobei n_i und n_t die jeweiligen Brechungsindizes der Medien sind. Differenziert man das Snelliussche Gesetz nach ϑ, erhält man:

$$n_i cos\vartheta_i d\vartheta_i = n_t cos\vartheta_t d\vartheta_t \;\rightarrow\; \frac{n_i}{n_t} = \frac{cos\vartheta_t d\vartheta_t}{cos\vartheta_i d\vartheta_i}. \tag{2.20}$$

Schlussendlich werden die Gleichungen (2.19) und (2.20) in (2.18) eingesetzt. Da $\varphi_t = \varphi_i + \pi$ gilt, ist $d\varphi_t = d\varphi_i$, wodurch die transmittierte Leuchtdichte im folgenden Zusammenhang zur eingehenden Leuchtdichte steht:

$$L_t = (1 - F_r(\vartheta_i)) \frac{n_t^2}{n_i^2} L_i. \tag{2.21}$$

Abbildung 2.3 zeigt die Abhängigkeiten der Leuchtdichten grafisch. Der Vektor $\boldsymbol{t}$ lässt sich leicht mit Hilfe des Snelliusschen Gesetzes bestimmen.

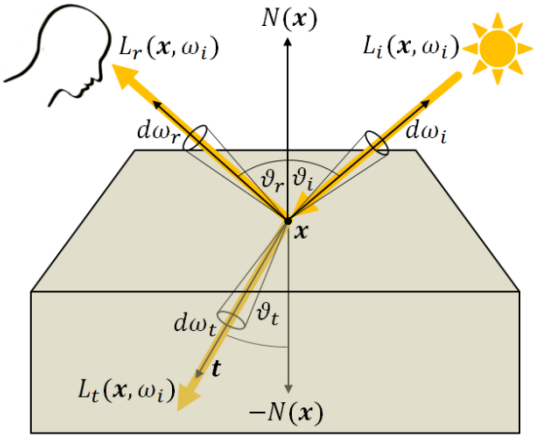

Abbildung 2.3: Zusammenhang zwischen Reflexion und Transmission.

Der Fresnel-Term selbst ist sehr komplex, weswegen er in interaktiven Verfahren approximativ beschrieben wird. Schlick [Sch94] schlägt die folgende Approximation vor:

$$F_r(\vartheta_i) \approx F_r(0°) + \big(1 - F_r(0°)\big)(1 - \cos^+(\vartheta_i))^5. \tag{2.22}$$

Dabei kann $F_r(0°)$ aus einer Look-Up-Textur stammen. Allerdings kann der Wert auch durch die folgende Formel bestimmt werden: $F_r(0°) = \left(\frac{n_t - 1}{n_i + 1}\right)^2$.

Anhand der eingehenden Leuchtdichte die reflektierte und die transmittierte Leuchtdichte zu bestimmen, hat auch für das Subsurface-Scattering eine wichtige Bedeutung. Hierbei durchläuft das transmittierte Licht solange Streuungsprozesse im Medium (scattering in participating media), bis es an einem Oberflächenpunkt des Modells wieder austritt oder komplett absorbiert ist.

2.1.5 Volume-Rendering

Bei den vorherigen Betrachtungen wurde immer davon ausgegangen, dass zwischen den Objektoberflächen einer Szene ein Vakuum besteht. Dies hat den Vorteil, dass man für die Lichtausbreitung nur die Oberflächen betrachten muss, denn die Leuchtdichte verändert sich beim Weg durch ein Vakuum nicht. Natürliche Phänomene, wie zum Beispiel Wolken, Nebel oder trübes Wasser, lassen sich mit diesen Ansätzen jedoch nicht abbilden. Diese volumetrischen Phänomene werden in der Computergrafik als Medium bezeichnet. Ein Medium ist eine volumetrisch ausgeprägte Verteilung von kleinen Materiepartikeln, die mit dem Licht interagieren. Bei der Darstellung eines Mediums ist der Beitrag eines einzelnen Partikels uninteressant, stattdessen geht man von einer gewissen Verteilung bzw. Dichte der Partikel aus und betrachtet, wie das Licht mit dieser Verteilung statistisch interagiert.

Trifft ein Lichtstrahl auf ein Medium, so kann seine Leuchtdichte durch vier verschiedene Arten der Interaktion verändert werden:

1. Absorption: Das Licht wird von den Partikeln in eine andere Energieform gewandelt (z. B. Wärme) und wird so gedämpft.
2. Emission: Hierunter fallen selbstfluoreszierende Partikel. Das Medium emittiert also selbst Licht.
3. Out-Scattering: Auf dem Weg des Lichts durch das Medium wird durch die Partikel das Licht in verschiedene Richtungen gestreut. Hierdurch verringert sich die Leuchtdichte.
4. In-Scattering: Ebenfalls durch die Streuung im Medium kann es passieren, dass Licht in den Lichtstrahl hinein gestreut wird und so die Leuchtdichte verstärkt.

Abbildung 2.4 zeigt die vier Interaktionsmöglichkeiten grafisch.

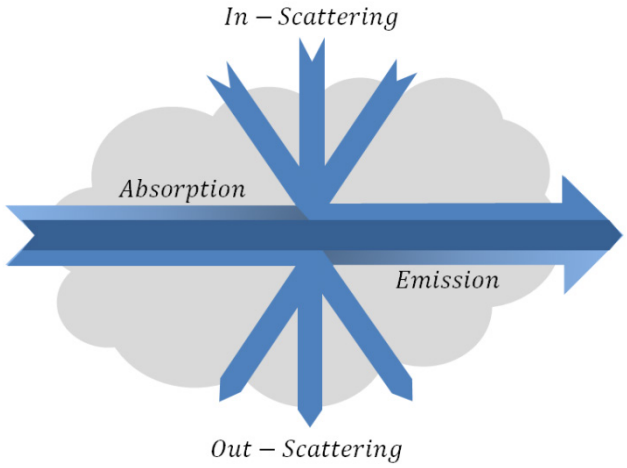

Abbildung 2.4: Interaktionen innerhalb eines Mediums.

Man kann die vier Einflüsse in zwei Kategorien teilen: Absorption und Out-Scattering dämpfen das Licht, während die Emission und das In-Scattering es verstärken. Aus Gründen der Übersichtlichkeit soll der Fokus zuerst auf die Verstärkung des Lichts im Medium gesetzt werden. Das eingehende Licht im Medium lässt sich durch das In-Scattering und die Emission berechnen. Wenn man einen Punkt $\boldsymbol{z}$ im Medium wählt und die resultierende Leuchtdichte in Richtung $\boldsymbol{\omega}$ für diesen Punkt berechnen möchte, kann man die folgende Gleichung verwenden:

$$L_v(\boldsymbol{z},\boldsymbol{\omega}) = L_{ve}(\boldsymbol{z},\boldsymbol{\omega}) + \sigma_s(\boldsymbol{z},\boldsymbol{\omega}) \int_{\Omega} f_p(\boldsymbol{z},\boldsymbol{\omega},\boldsymbol{\omega_i}) L_{vi}(\boldsymbol{z},\boldsymbol{\omega_i}) d\boldsymbol{\omega_i}. \tag{2.23}$$

L_{ve} bezeichnet die Leuchtdichte im Punkt $\boldsymbol{z}$ in Richtung $\boldsymbol{\omega}$, die durch Emission im Medium hervorgerufen wird (Fluoreszenz). L_{vi} ist direktes oder indirektes Licht, welches im Punkt $\boldsymbol{z}$ eintritt (es handelt sich um die gleiche Größe wie $L_i$in Gleichung (2.3)). σ_s ist der Streuungskoeffizient, dieser wird später noch erläutert. Die Funktion f_p ist ähnlich zur BRDF und nennt sich Phasenfunktion (Phase-Function): Sie liefert die Verteilung der Wahrscheinlichkeit, ob Licht aus einem differentiellen Raumwinkel um $\boldsymbol{\omega_i}$ in einen differentiellen Raumwinkel um $\boldsymbol{\omega}$ gestreut wird ($\boldsymbol{\omega_i}$ und $\boldsymbol{\omega}$ sind Richtungsvektoren). Es handelt sich also um eine Wahrscheinlichkeitsdichte. Dabei kann die Verteilung isotrop oder anisotrop sein. Eine isotrope Phasenfunktion verteilt das Licht gleichmäßig in alle Richtungen und ist konstant: $f_p = 1/4\pi$. Anisotrope Phasenfunktionen besitzen eine beliebige Verteilung. Für Phasenfunktionen gilt, wie bereits für die BRDF, Reziprozität und das Prinzip der Energieerhaltung. Zur Approximation natürlicher Medien gibt es verschiedene Modelle für Phasenfunktionen, z. B. das von Henyey und Greenstein [HG41] oder von Blasi et al. [BLS93].

In Gleichung (2.23) sammelt das Integral also das einfallende Licht über die Sphäre Ω des Punktes $\boldsymbol{z}$ und faltet dieses mit der Phasenfunktion für das Medium.

Als nächstes sollen die dämpfenden Einflüsse Absorption und Out-Scattering betrachtet werden. Für die Dämpfung werden zwei Parameter verwendet: der Absorptionsquerschnittskoeffizient $\sigma_a(\boldsymbol{z},\boldsymbol{\omega})$, der die Dämpfung des Lichts pro Meter beschreibt und der Streuungskoeffizient $\sigma_s(\boldsymbol{z},\boldsymbol{\omega})$, der die Wahrscheinlichkeit für ein Streuungsereignis pro Meter bezeichnet. Beide Koeffizienten werden zusammengefasst zum Dämpfungskoeffizienten:

$$\sigma_t(\boldsymbol{z},\boldsymbol{\omega}) = \sigma_a(\boldsymbol{z},\boldsymbol{\omega}) + \sigma_s(\boldsymbol{z},\boldsymbol{\omega}). \tag{2.24}$$

Für die Dämpfung der Leuchtdichte durch den Dämpfungskoeffizienten betrachtet man nun im Medium ein differentielles Streckenstück ds durch den Punkt $\boldsymbol{z}$ in Richtung $\boldsymbol{\omega}$. Dieses Streckenstück soll die Leuchtdichte $\mathrm{L_i}(\boldsymbol{z},-\boldsymbol{\omega})$ empfangen und die Leuchtdichte $\mathrm{L_o}(\boldsymbol{z},\boldsymbol{\omega})$ am Ende aussenden. Die Dämpfung innerhalb des Streckenstücks gehorcht der folgenden Differentialgleichung:

$$\mathrm{L_i}(\boldsymbol{z},-\boldsymbol{\omega}) - \mathrm{L_o}(\boldsymbol{z},\boldsymbol{\omega}) = dL_o(\boldsymbol{z},\boldsymbol{\omega}) = -\sigma_t(\boldsymbol{z},\boldsymbol{\omega})\mathrm{L_i}(\boldsymbol{z},-\boldsymbol{\omega})ds. \tag{2.25}$$

Aus dieser Differentialgleichung kann nun die Integralgleichung für die Dämpfung über eine komplette Strecke $\boldsymbol{s}$ zwischen zwei Punkten $\boldsymbol{x}$ und $\boldsymbol{z}$ bestimmt werden:

$$\mathrm{T_r}(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\int_0^{s=\|\boldsymbol{z}-\boldsymbol{x}\|} \sigma_t(\boldsymbol{x}+s\boldsymbol{\omega},\boldsymbol{\omega})ds\right). \tag{2.26}$$

$\boldsymbol{\omega}$ entspricht hier dem normalisierten Differenzvektor von $\boldsymbol{x}$ nach $\boldsymbol{z}$. T_r wird als Transmissionsgrad bezeichnet. Um zu bestimmen, wie groß die eingehende Leuchtdichte $\mathrm{L_i}(\boldsymbol{x},-\boldsymbol{\omega})$ im Punkt $\boldsymbol{x}$ ist, wenn eine ausgesendete Leuchtdichte $\mathrm{L_o}(\boldsymbol{z},\boldsymbol{\omega})$ im Punkt $\boldsymbol{z}$ bekannt ist, muss man nur den Transmissionsgrad multiplizieren: $\mathrm{L_i}(\boldsymbol{x},-\boldsymbol{\omega}) = T_r(\boldsymbol{x},\boldsymbol{z})\mathrm{L_o}(\boldsymbol{z},\boldsymbol{\omega})$.

Zu guter Letzt werden die vier Einflüsse in einer Gleichung zusammengefasst und man erhält das Volume-Rendering-Integral [Cha50]:

$$\mathrm{L_i}(\boldsymbol{x},\boldsymbol{\omega}) = T_r(\boldsymbol{x},\boldsymbol{x_o})L_o(\boldsymbol{x_o},-\boldsymbol{\omega}) + \int_0^{s=\|\boldsymbol{x}-\boldsymbol{x_0}\|} T_r(\boldsymbol{x_0},\boldsymbol{z}(s))L_v(\boldsymbol{z}(s),-\boldsymbol{\omega})ds. \tag{2.27}$$

Die geometrischen Zusammenhänge der Gleichung (2.27) sind in Abbildung 2.5 dargestellt. Der Punkt $\boldsymbol{x_o}$ ist ein Oberflächenpunkt, der Licht in Richtung des Punktes $\boldsymbol{x}$ reflektiert (entlang $-\boldsymbol{\omega}$). Für $\boldsymbol{x_0}$ gilt $\boldsymbol{x_0} = r(\boldsymbol{x},\boldsymbol{\omega})$ (vgl. Gleichung (2.4)). Bei der Betrachtung der Gleichung (2.27) fällt auf, dass L_i links und L_o rechts vom Gleichheitszeichen stehen. Diese Vertauschung ist beabsichtigt, denn die Gleichung berechnet die *eingehende* Leuchtdichte im Punkt $\boldsymbol{x}$ aus Richtung $\boldsymbol{\omega}$. Der erste Term der Gleichung beschreibt, wie viel Licht von $\boldsymbol{x_o}$ bei $\boldsymbol{x}$ nach der Dämpfung durch das Medium ankommt. Der zweite Term integriert die Leuchtdichte über die Strecke zwischen $\boldsymbol{x_o}$ und $\boldsymbol{x}$ unter Berücksichtigung der Dämpfung T_r (Absorption und Out-Scattering für die jeweilige Teilstrecke) und der Verstärkung L_v im Medium (Emission und In-Scattering). Für den Punkt $\boldsymbol{z}$ gilt $\boldsymbol{z} = \boldsymbol{x_o} - s\boldsymbol{\omega}$.

Die korrekte Berechnung des Volume-Rendering-Integrals ist selbst mit Offline-Verfahren nicht möglich, weswegen approximative Verfahren verwendet werden. Eine gängige Vereinfachung bei interaktiven Verfahren ist die Annahme, dass das Medium homogen ist. Des Weiteren werden zwei Phänomene unterschieden: Single Scattering und Multiple Scattering.

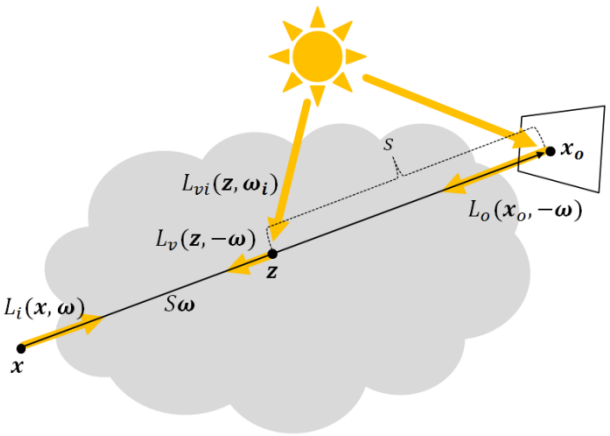

Abbildung 2.5: Geometrische Größen des Volume-Rendering-Integrals (2.27).

Beim Single Scattering wird für den Term L_{vi} aus Gleichung (2.23) nur das direkte Licht betrachtet. Es wird also davon ausgegangen, dass das direkte Licht nur einmalig im betrachteten Punkt gebrochen wird. Sofern das direkte Licht durch ein Objekt blockiert wird, ergeben sich so beim Single Scattering Schattenstrahlen im Medium. Beim Multiple Scattering wird dagegen im Term L_{vi} auch die indirekte Beleuchtung durch wiederholte Streuung im Medium abgebildet. Die Berechnung von Multiple Scattering ist sehr aufwändig und wird nur approximativ durchgeführt. Stam [Sta95] schlägt zur Approximation eine Abbildung als Diffusionsprozess vor, der auch in dieser Arbeit zur Anwendung kommt. Abbildung 2.6 zeigt einen Vergleich zwischen Single Scattering und Multiple Scattering.

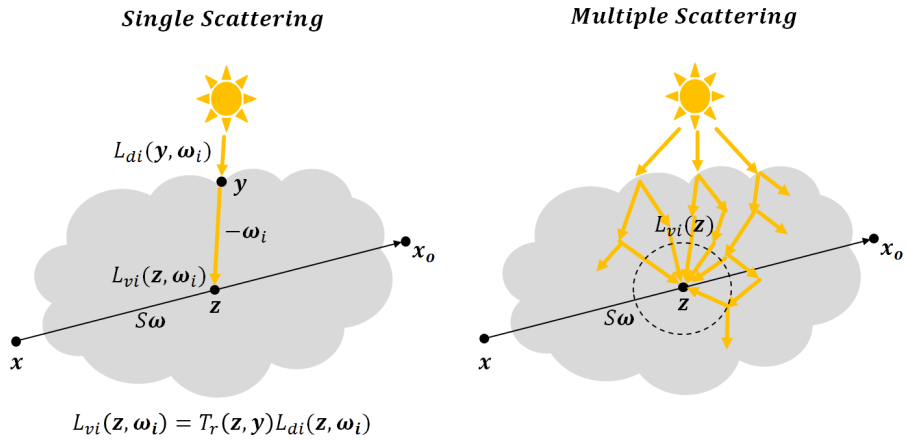

Abbildung 2.6: Vergleich zwischen Single Scattering und Multiple Scattering.

2.1.6 Subsurface-Scattering

Beim Subsurface-Scattering (SSS) finden alle hier vorgestellten Prinzipien zur globalen Beleuchtung in einem Modell Anwendung. Hiermit ist es zum Beispiel möglich, Materialien wie Kerzenwachs, Jade, Marmor, trübe Flüssigkeiten oder menschliche Haut zu synthetisieren. Das Prinzip dahinter lässt sich einfach erklären: Wenn Licht auf die Oberfläche eines Objekts trifft, dann wird ein Teil des Lichts reflektiert und ein Teil in das Objekt gebrochen. Dies geschieht auf Basis des Fresnel-Terms (siehe Kapitel 2.1.4). Für das Licht, das direkt reflektiert wird, kann von einer einfachen Reflexion im Punkt ausgegangen werden, die durch die BRDF gut abgebildet werden kann. Für das Licht, das unter die Oberfläche des Objekts gelangt (Subsurface), findet das Prinzip des Volume-Renderings Anwendung. Unter der Oberfläche wird von einem Medium ausgegangen, welches das Licht auf verschiedene Art und Weise streut. Schlussendlich tritt das Licht wieder aus dem Objekt aus, jedoch unter Umständen an einer anderen Stelle als an der es eingetroffen ist. Je nach Art des Materials und dessen Fähigkeit, Licht zu zerstreuen, kann der Austritt des Lichts geometrisch sehr weit vom Eintrittspunkt entfernt liegen. Für die mathematische Interpretation des Vorgangs wird eine Verallgemeinerung der Rendergleichung verwendet:

$$\mathrm{L_o}(\boldsymbol{x}, \boldsymbol{\omega}) = \int_A \int_{\Omega^+} f_{ss}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{x_i}, \boldsymbol{\omega_i}) L_i(\boldsymbol{x_i}, \boldsymbol{\omega_i}) (N(\boldsymbol{x_i}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i} dA. \tag{2.28}$$

In dieser Gleichung wird nicht nur einmalig das Integral über die Hemisphäre eines Punktes gebildet, sondern es wird für jeden Punkt x_i auf der Oberfläche A des Objekts das Integral über die Hemisphäre gebildet. Prinzipiell kann das eigehende Licht L_i in jedem Oberflächenpunkt $\boldsymbol{x_i}$ zum ausgehenden Licht L_o im Punkt x in Richtung ω beitragen. Wie stark der Beitrag ist, wird durch die Funktion f_{ss} abgebildet, die sich Bidirectional-Scattering-Surface-Reflectance-Distribution-Function (BSSRDF) nennt. Die BSSRDF ist eine Verallgemeinerung der BRDF (siehe Abbildung 2.7). Für die BSSRDF gilt ähnlich zur BRDF der folgende Zusammenhang:

$$f_{ss}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{x_i}, \boldsymbol{\omega_i}) = \frac{dL_o(\boldsymbol{x}, \boldsymbol{\omega})}{d\phi_i(\boldsymbol{x_i}, \boldsymbol{\omega_i})}. \tag{2.29}$$

$d\phi_i$ kennzeichnet den differentiellen Lichtstrom im Punkt $\boldsymbol{x_i}$ aus Richtung $\boldsymbol{\omega_i}$.

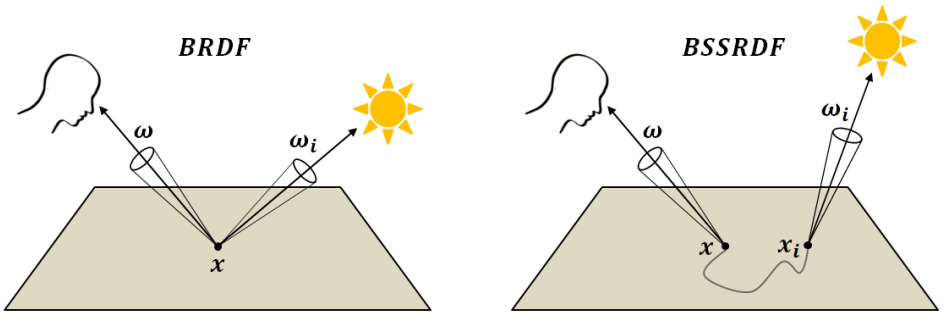

Abbildung 2.7: Vergleich zwischen BRDF und BSSRDF.

Wie schon beim „reinen“ Volume-Rendering ist es beim Subsurface-Scattering sehr teuer, mehrere Streuungsereignisse (Multiple Scattering) in einem Material abzubilden. Sich auf Single-Scattering-Ereignisse zu beschränken, ist beim Subsurface-Scattering häufig keine sinnvolle Alternative, denn viele Materialien, wie zum Beispiel Flüssigkeiten wie Milch, haben einen sehr hohen Albedo („Weißheitsgrad“, diffuser Streuungsgrad), so dass selbst nach hundert Streuungsereignissen noch ca. 87% der ursprünglichen Energie des Lichts vorhanden ist. Solche Materialien können nur überzeugend synthetisiert werden, wenn mehrere Streuungsereignisse abgebildet werden. Jensen [JML*01] schlägt hierfür in Anlehnung an [Sta95] die Dipol-Diffusion vor (diese wird in Kapitel 5.5.2 noch ausführlicher erklärt). Es gibt einige interaktive globale Beleuchtungsverfahren, die sich auf die Darstellung von Subsurface-Scattering auf Basis der Dipol-Diffusion [DS03,KLO07] konzentrieren. Dabei handelt es sich jedoch um separierte Verfahren, die sich ausschließlich auf dieses Phänomen konzentrieren. Für die Simulation menschlicher Haut existieren wiederum mehrere dedizierte Verfahren, die mit der sogenannten Texture-Space-Diffusion [BL03,BL05] arbeiten.

2.1.7 Heckbert's Lichtpfadnotation

Zur Klassifizierung globaler Beleuchtungsverfahren wird in der Regel die Lichtpfadnotation von Heckbert [Hec90] verwendet. Diese zeigt in Form von regulären Ausdrücken an, welche Lichtpfade durch ein Verfahren abgebildet werden können. Ereignisse, die bei der Verfolgung des Lichts unterschieden werden, sind:

- L: Beginn des Lichtpfads in der Lichtquelle,
- D: diffuse Reflexion,
- S: glänzende Reflexion (specular),
- V: Interaktion mit einem Medium und
- E: Ankunft des Lichtpfads im Auge des Betrachters.

Der einfachste Weg, den das Licht nehmen kann, ist LE – das Licht gelangt von einer Lichtquelle direkt zum Betrachter. Durch lokale Beleuchtungsmodelle lassen sich die Lichtpfade $L(D|S)E$ abbilden, sofern die Blockierung des Lichts ignoriert werden kann. Der | Operator wird hier als Oder gelesen. Häufig spricht man bereits von globaler Beleuchtung, wenn für den Lichtpfad $L(D|S)E$ eine korrekte Verdeckungsberechnung durchgeführt wird. Durch diese kommt es zur Schlagschattenbildung. Für diese Arbeit soll ein globales Beleuchtungsverfahren dadurch definiert sein, dass es einen Lichtpfad von $L(D|S|V)^{++}E$ unterstützt. Dabei bedeutet $++$, dass zwei oder mehr Ereignisse vorhanden sein müssen. Gültige Lichtpfade für globale Verfahren wären dann zum Beispiel $LDDE$, $LSSE$, $LDSE$, $LSDE$, $LDSD^{+}VE$, etc. Zur Vervollständigung der Notation soll noch erwähnt werden, dass ein Stern-

chen (*) im regulären Ausdruck bedeutet, dass ein Ereignis keinmal oder beliebig oft auftreten kann und ein Fragezeichen (?) kennzeichnet ein optionales Ereignis.

2.1.8 Optische Phänomene

Kein interaktives globales Beleuchtungsverfahren bietet eine umfassende Lösung aller Phänomene, die durch natürliche Beleuchtung auftreten können. Dies ist der allgemeinen Komplexität der natürlichen Lichtausbreitung geschuldet. Bereits die ersten vorgestellten, nicht-interaktiven globalen Verfahren waren auf bestimmte optische Phänomene fokussiert. Zum Beispiel lassen sich mit Ray-Tracing-Ansätzen sehr elegant und effizient spiegelnde Reflexionen abbilden, während diffuse Reflexionen eine typische Domäne von Radiosity-Verfahren sind. Bei den interaktiven Verfahren hat sich die „Phänomen-fokussierte“ Entwicklung ebenfalls bewährt. Um die Komplexität der Simulation in Grenzen zu halten, konzentrieren sich die Entwickler auf bestimmte Phänomene, die mit dem entwickelten Verfahren besonders effizient simuliert werden können. Hieraus ergibt sich eine Vielzahl von optischen Phänomenen, die man für die Beleuchtungsberechnung unterscheiden kann. Die Trennung eines Phänomens von seinem Kontext hat leider auch zur Folge, dass die korrekten physikalischen Zusammenhänge nicht berücksichtigt werden können. In solchen Fällen konzentriert man sich auf die Wahrnehmung des Betrachters und nimmt diese als subjektive Maßeinheit für die Qualität des Verfahrens. Nachfolgend soll ein grober Überblick über die gängigen optischen Phänomene gegeben werden, die in der globalen Beleuchtungssynthese als fester Wortschatz gelten.

Diffuse Reflexion: Die diffuse Reflexion entsteht durch LD^+E Lichtpfade. Bei der ideal diffusen Reflexion wird eintreffendes Licht gleichmäßig in alle Richtungen verteilt (Lambertscher Strahler). Hierdurch wirkt eine diffuse Oberfläche wie ein Tiefpassfilter, wodurch sich viele Optimierungsmöglichkeiten durch Interpolationsverfahren ergeben. Eine weitere wichtige Eigenschaft von diffusen Reflexionen ist, dass sie unabhängig vom Betrachter sind. Ändert der Betrachter also seine Position oder Orientierung, so müssen die Reflexionen nicht neu berechnet werden.

Glänzende Reflexionen (Glossy/Specular Reflections): Diese Art der Reflexion entsteht bei LD^*S^+E Pfaden. Sie ist auf fast glatten Oberflächen zu beobachten. Im Gegensatz zur diffusen Reflexion produziert die glänzende Reflexion hochfrequente Signale, die vom Betrachtungswinkel abhängig sind.

Spiegelende Reflexionen (Ideal specular Reflections): Durch die Implementierungsdetails verschiedener Verfahren hat sich die Unterscheidung zwischen spiegelnden und glänzenden Reflexionen ergeben. Beide werden durch den gleichen Lichtpfad beschrieben. Der Unterschied besteht nur im Grad der Spiegelung. Lässt sich eine deutliche Spiegelung erkennen, so lassen sich oftmals sehr einfache interaktive Verfahren wie beispielsweise Environment-Mapping [Gre86] oder Reflection-Mapping [BN76] anwenden, um dieses Phänomen plausibel darzustellen. Die Faltung der BRDF mit dem eingehenden Licht (siehe Gleichung (2.3)) ist in diesem Fall eine einfache Multiplikation.

Kaustiken (Caustics): Kaustiken sind beobachtbar, wenn Licht durch eine glänzende oder spiegelnde Oberfläche reflektiert wird und dann auf eine diffuse Oberfläche trifft. Alternativ kann das Licht auch durch Brechung auf die diffuse Oberfläche gelangen. Der Lichtpfad beinhaltet die gleichen Reflexionen wie bei der glänzenden und spiegelnden Reflexion, jedoch ist die Reihenfolge vertauscht. So ergibt sich ein Lichtpfad von LS^+DE. Kaustiken können hochfrequente Reflexionen erzeugen, die jedoch invariant vom Betrachtungspunkt sind. Die effektive Berechnung von Kaustiken in interaktiven

Verfahren ist ein schwieriges Unterfangen, weswegen viele Verfahren dieses Phänomen nicht unterstützen.

Schlagschatten: Die meisten interaktiven Systeme unterstützen die Darstellung eines Schlagschattens. Hierfür wird entweder ein einfaches Shadow-Mapping-Verfahren [Wil78] oder das Shadow-Volume-Verfahren [Cro77] eingesetzt. Die Berechnung der Schattierung basiert nur auf der Blockierung des direkten Lichts. Dabei wird davon ausgegangen, dass die Lichtquelle keine geometrische Ausdehnung hat. So ist die Silhouette des Schattens immer scharf, was in der Natur eigentlich nicht zu beobachten ist.

Weiche Schatten (Soft Shadows): Natürliche Lichtquellen haben eine Ausdehnung, die zur Folge hat, dass Oberflächenpunkte existieren, die nur einen Teil der Lichtquelle „sehen“. Dies führt zu weichen Schattenübergängen im Silhouettenbereich eines Objekts. Durch die Verwendung von Flächenlichtquellen muss das Integral über den sichtbaren Teil der Lichtquelle für jeden Oberflächenpunkt berechnet werden. Dabei wird bei den meisten Verfahren davon ausgegangen, dass die Lichtquelle eine homogene Lichtverteilung besitzt. Die Berechnung des Integrals ist sehr aufwändig, weswegen frühe interaktive Verfahren diese nicht durchführten. Um trotzdem weiche Schatten darstellen zu können, wurden Bildfilterungsverfahren genutzt, die für immer gleich weiche Übergänge bei den Schatten sorgten (Percentage-Closer-Filtering (PCF) Shadow-Maps [RSC87]). Die Berechnung von weichen Schatten ist für die indirekte Beleuchtung ebenfalls von Bedeutung, da indirekt reflektierende Oberflächen wie inhomogene Flächenlichtquellen wirken. Eisemann et al. [EAS*09] liefern einen umfangreichen Überblick über Verfahren zur Erzeugung weicher Schatten.

Ambient Occlusion: Dieses Phänomen könnte eigentlich auch unter „weiche Schatten“ eingeordnet werden, denn es erzeugt ebenfalls Schattierungen. Jedoch spielt beim Ambient Occlusion die eigentliche Lichtquelle keine Rolle. Ambientes Licht ist ein Konstrukt aus der Computergrafik, das keine Entsprechung in der Natur hat. Es soll in gewisser Weise die Berechnung von diffusen indirekten Reflexionen ersetzen, indem davon ausgegangen wird, dass das indirekte Licht vollkommen homogen in der Szene verteilt ist. Wenn man diese Gleichverteilung des Lichts voraussetzt, sind für die Berechnung der Schattierung nur noch die geometrischen Gegebenheiten von Bedeutung. Es muss also nur geprüft werden, wie viel der Hemisphäre eines Oberflächenpunktes durch nahestehende Oberflächen verdeckt wird. Anhand der Prozentzahl der Verdeckung lässt sich dann leicht eine Schattierung berechnen. Durch die Annahme der homogenen Lichtverteilung können beim Ambient Occlusion keine gerichteten Schattierungen, wie zum Beispiel Schlagschatten oder ähnliches, auftreten. Es existieren sehr viele Verfahren zur effizienten Berechnung von Ambient Occlusion, eine aktuelle Zusammenfassung findet man in [MS09].

Volume-Scattering und **Subsurface-Scattering** sind ebenfalls optische Phänomene, die häufig separat betrachtet werden. Diese wurden jedoch bereits in den Kapiteln 2.1.5 und 2.1.6 erläutert.

2.1.9 Physikalisch korrekte vs. optisch plausible Beleuchtung

Im vorherigen Unterkapitel wurde bereits beschrieben, dass sich interaktive globale Beleuchtungsverfahren in der Regel auf bestimmte optische Phänomene konzentrieren und diese möglichst überzeugend darstellen. Hierfür werden in den meisten Fällen vereinfachte, approximative Modelle genutzt, die stark vom physikalisch korrekten Verhalten abweichen können. Deshalb ist es bei den meisten interaktiven Verfahren nicht legitim, von physikalisch korrekten Simulationen zu sprechen

und es wird stattdessen häufig der Begriff der physikalisch plausiblen bzw. der optisch plausiblen Beleuchtung genutzt [RDG*12,Lew94]. Da für die Begriffe keine einheitlichen Definitionen existieren, werden sie manchmal synonym verwendet. In dieser Arbeit soll zwischen physikalisch korrekten, physikalisch plausiblen und optisch plausiblen Beleuchtungsergebnissen unterschieden werden:

Physikalisch korrekt: In dieser Arbeit gilt eine Simulation als physikalisch korrekt, wenn deren Ergebnisse im betrachteten Kontext auf physikalisch korrekten Grundlagen der Energieerhaltung und der Strahlenoptik (geometrische Optik) basieren und sie absolut zu den Eingangsgrößen sind. Der Kontext wird hier durch das Phänomen bestimmt. Demnach ist zum Beispiel das Radiosity-Verfahren [GTG*84] im Kontext des Phänomens der idealen diffusen Reflexionen als physikalisch korrekt anzusehen. Wellenoptische und quantenoptische physikalische Gesetze, sowie farbmetrische Zusammenhänge werden bei dieser Betrachtung außen vor gelassen.

Physikalisch plausibel: In dieser Arbeit soll sich der Begriff der physikalisch plausiblen Beleuchtung aus dem Begriff des physikalisch plausiblen Shadings ableiten. Ritschel et al. [RDG*12] bezeichnen ein Shading-Modell als physikalisch plausibel, wenn dessen BRDF bzw. BSSRDF die Gesetze der Energieerhaltung korrekt abbildet, es also gilt: $\int_\Omega f(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) cos\vartheta d\boldsymbol{\omega} \leq 1$ und $f(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = f(\boldsymbol{x}, \boldsymbol{\omega_i}, \boldsymbol{\omega})$ (siehe Kapitel 2.1.3). Die erzeugten Gradienten der BRDF bzw. BSSRDF dürfen dabei auf empirischen Modellen beruhen, wenn diese optisch plausibel sind. Demnach ist die physikalisch plausible Beleuchtung auch immer optisch plausibel. Im Gegensatz dazu ist die optisch plausible Bleuchtung jedoch nicht zwingend physikalisch plausibel. Für diese Arbeit soll verschärfend für die physikalisch plausible Beleuchtung gelten, dass $\int_\Omega f(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) cos\vartheta d\boldsymbol{\omega} = 1$, wenn das verwendete Material keine Absorbtion durch Reflexionskoeffizienten abbildet. Wird einkommendes Licht also durch die BRDF bzw. BSSRDF stark gebündelt reflektiert, so ist darauf zu achten, dass ein energetisches Gleichgewicht besteht. Ferner soll der Geometrieterm aus der Rendergleichung (Gleichung (2.8)) korrekt abgebildet werden. Da jedoch alle interaktiven globalen Beleuchtungsverfahren die Verdeckungsrechnung und somit die Abbildung von weichen Schatten sehr approximativ vornehmen, soll die Sichtbarkeitsfunktion (V) im Geometrieterm eine untergeordnete Rolle spielen.

Optisch plausibel: Als optisch plausibel sollen die Beleuchtungsergebnisse gelten, die bei dem Betrachter den Eindruck hinterlassen, dass sie konsistent und somit erwartungskonform sind. Dabei müssen diese aber nicht zwingend auf physikalischen Grundlagen basieren. Es handelt sich also um eine subjektive Eigenschaft, bei der nicht klar definiert ist, wie die Erwartungen des Betrachters erfüllt werden können. Leider lässt sich nicht pauschal sagen, dass eine physikalisch korrektere Lösung zwangsläufig zu einer optisch plausibleren Abbildung führt. In diesem Bereich fehlen aussagekräftige Forschungsergebnisse und empirische Untersuchungen. Gegenwärtig existieren nur sehr spezielle Untersuchungen, die sehr eingegrenzte Fälle für bestimmte Beleuchtungsverfahren bzw. Phänomene betrachten (beispielsweise [KFB10,YCK*09]), die sich nicht verallgemeinern lassen. Für diese Arbeit gilt ein Verfahren dann nicht als optisch plausibel (und somit auch nicht als physikalisch plausibel), wenn die Beleuchtungsergebnisse temporär inkohärent sind (zum Beispiel Flackern) oder starke kontrastreiche Fehler entstehen. In solchen Fällen enthält die Abbildung also optische Artefakte.

In der Abbildung 2.8 wird ein Beispiel gezeigt, bei der das Phänomen der direkten diffusen Reflexion physikalisch plausibel, optisch plausibel und nicht plausibel dargestellt wird. Die Abbildung verdeut-

licht, dass der absolut gemessene Fehler einer Darstellung (zum Beispiel in Form eines Differenzbildes) kein ausreichendes Kriterium bietet, um die Qualität eines Verfahrens zu beurteilen.

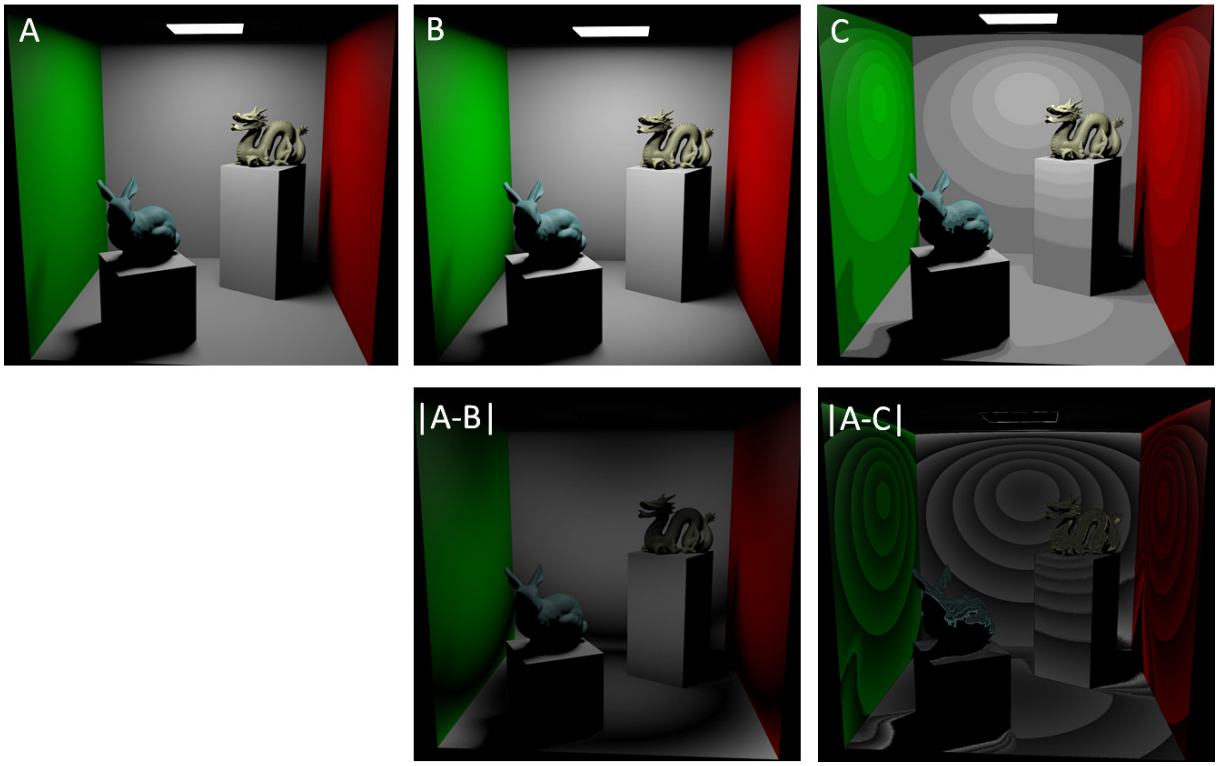

Abbildung 2.8: Vergleich zwischen physikalisch plausiblen, optisch plausiblen und nicht plausiblen Darstellungen. Für die Simulation wurde nur direktes Licht von der Flächenlichtquelle betrachtet. (A) zeigt eine physikalisch plausible Lösung, bei der die Beleuchtungsstärke mit der quadratisch inversen Entfernung abnimmt. In (B) dagegen nimmt die Beleuchtungsstärke linear mit der Entfernung ab, hierbei handelt es sich um keine physikalisch sinnvolle Abbildung, dennoch erscheint das Bild (B) für einen Betrachter optisch plausibel. In (C) nimmt die Beleuchtungsstärke wieder mit der quadratisch inversen Entfernung physikalisch korrekt ab, allerdings wurde eine Quantisierung bei der Reflexion vorgenommen, die den optischen Eindruck der Beleuchtung zerstört und sie somit als optisch nicht plausibel erscheinen lässt (man erkennt kontrastreiche Artefakte). Da die Simulation von (C) optisch nicht plausibel ist, kann sie nach der Definition in diesem Unterkapitel auch nicht physikalisch plausibel sein, obwohl sie das korrekte Distanzgesetz verwendet. Die Differenzbilder in der unteren Reihe zeigen, dass der absolute Fehler über beide Bilder ungefähr gleich ist, dennoch erscheint (B) optisch deutlich plausibler als (C). Die reine Betrachtung des Fehlers ist also nicht ausreichend, um die Qualität der Beleuchtungssimulation zu beurteilen.

2.2 Augmented Reality

Bei einer Augmented-Reality-Anwendung wird die reale Umwelt eines Nutzers um synthetisierte bzw. virtuelle Objekte erweitert. Die Erweiterung kann visueller oder akustischer Natur sein. In dieser Dissertation werden jedoch nur visuelle Systeme betrachtet. Milgram et al. [MTU*94] betrachten Augmented Reality als Teil des Reality-Virtuality-Kontinuums. Dieses Kontinuum hat als Endzustände die reale Umwelt und die virtuelle Umwelt. Zwischen diesen beiden Zuständen bildet sich das Mixed-Reality-Kontinuum, von dem Augmented Reality eine Untermenge bildet (siehe Abbildung 2.9).

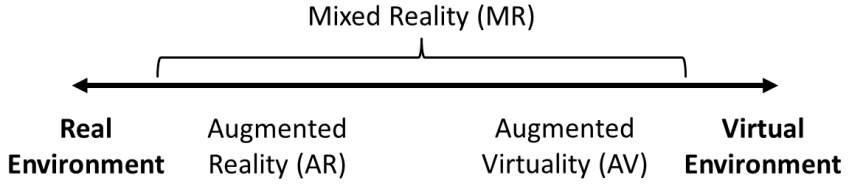

Abbildung 2.9: Reality-Virtuality-Kontinuum nach Milgram et al. [MTU*94].

Der Begriff Augmented Reality wird häufig auf das „Mischen“ von realen und virtuellen Bildern reduziert - ohne dass die beiden Bilder zueinander in Bezug stehen müssen. Demnach wäre z. B. eine Temperatur- oder Zeitanzeige am unteren Rand eines Smartphone-Displays eine AR-Anwendung, wenn denn im Hintergrund das Kamerabild präsentiert würde. Eine solche Technologie entspricht aber eher einem Head-Up-Display (HUD) als einer AR-Anwendung. Um eine klare Trennung zwischen den Technologien zu gewährleisten, soll für diese Arbeit die Definition von Azuma [Azu97] herangezogen werden. Dieser nennt drei elementare Eigenschaften, die eine AR-Anwendung aufweisen muss:

1. Die reale Umgebung wird um virtuelle Objekte erweitert.
2. Die Erweiterung geschieht unmittelbar bzw. in Echtzeit und ist interaktiv.
3. Die virtuellen Objekte stehen in Bezug zur realen Umgebung, sie sind also in der realen Umgebung verankert (registriert).

Die erste Eigenschaft benennt das Mischen der Realität und Virtualität. Die zweite Eigenschaft verlangt, dass das System auf Änderungen in der realen Umgebung unmittelbar reagiert und den virtuellen Inhalt entsprechend anpasst. Die Interaktion mit dem Anwender geschieht also unter anderem über die Realumgebung.

Für visuelle AR-Systeme ist die dritte Eigenschaft von besonderer Bedeutung. Denn sie verlangt, dass virtuelle Objekte in der realen Umgebung verortet sein müssen. Für den Betrachter äußert sich diese Eigenschaft darin, dass virtuelle Objekte scheinbar wie reale Objekte im Raum „stehen“. Man kann sie von mehreren Seiten betrachten, ohne dass sie ihre Position im dreidimensionalen Raum verlieren. Die Verortung bzw. Verankerung der virtuellen Objekte bezeichnet man auch als räumliche Registrierung bzw. Tracking. Das Tracking liegt nicht im Fokus dieser Arbeit, weswegen diesem nur eine oberflächliche Betrachtung gewidmet wird. Neben der räumlichen Registrierung gibt es noch die photometrische Registrierung. Hier wird versucht, die synthetischen Objekte möglichst realistisch bzw. „nahtlos“ in die reale Umgebung zu integrieren. Das beinhaltet, dass virtuelle Objekte auf reale Lichtverhältnisse und dass reale Objekte auf virtuelle Lichtverhältnisse reagieren. Zur Synthese dieser Wechselwirkung werden entsprechende globale Beleuchtungsverfahren verwendet. Um die realen Lichtverhältnisse zu messen, werden verschiedene Sensoren verwendet, die spezifische Vor- und Nachteile besitzen.

In den nachfolgenden Unterkapiteln folgen Erläuterungen zur räumlichen und photometrischen Registrierung, sowie eine kurze Betrachtung üblicher Sensoren für die photometrische Registrierung.

2.2.1 Räumliche Registrierung

Augmented Reality wird nur dann zur überzeugenden Anwendung, wenn man es schafft, die virtuellen Objekte in der Realität fehlerfrei im dreidimensionalen Raum zu registrieren bzw. zu verankern. Da die erweiterten Objekte prinzipiell von allen Seiten betrachtet werden können, müssen sie selbst in einem dreidimensionalen Raum liegen. Zur Darstellung der Objekte aus einem bestimmten Blickwinkel nutzt man dann eine virtuelle Kamera. Diese muss die gleichen Eigenschaften wie die reale Kamera aufweisen. Man unterscheidet hier zwischen den extrinsischen und den intrinsischen Eigenschaften der Kamera. Die extrinsischen Eigenschaften beinhalten die Position und die Ausrichtung der Kamera. Diese können als homogene 4x4-Matrix beschrieben werden:

$$M_{ext} = \begin{bmatrix} R_{00} & R_{01} & R_{02} & t_x \\ R_{10} & R_{11} & R_{12} & t_y \\ R_{20} & R_{21} & R_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.30}$$

In dieser Matrix beinhaltet die 3x3 Submatrix R die Ausrichtung und der Translationsvektor $\boldsymbol{t}$ die Position der Kamera. Die intrinsischen Eigenschaften betreffen die Projektionseigenschaften der Kamera. Diese können auf folgende Parameter reduziert werden (Lochkamera-Modell, siehe Abbildung 2.10):

1. Brennweite der Kamera zur X- und Y-Achse der Bildebene (f_x, f_y).
2. Den Hauptpunkt der Kamera, an dem die optische Achse die Bildebene durchstößt (p_x, p_y).
3. Korrekturkoeffizienten für die radiale und tangentiale Verzerrung durch eine nicht ideale Optik.

Diese Parameter können ebenfalls durch eine Matrix abgebildet werden (die Korrekturkoeffizienten sind nicht inbegriffen):

$$M_{int} = \begin{bmatrix} f_x & 0 & R_{02} & 0 \\ 0 & f_y & R_{12} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.31}$$

Weil die Eigenschaften der realen Kamera auf die virtuelle übernommen werden, müssen die Eigenschaften der realen Kamera bekannt sein. Während die intrinsischen Eigenschaften für eine Digitalkamera immer gleich bleiben und einfach zu bestimmen sind, sind die extrinsischen Parameter einer ständigen Änderung durch den Nutzer ausgesetzt. Die reale Kameraposition und -orientierung muss deshalb durchgehend von der AR-Software neu berechnet werden. Dies ist ein aufwändiges Unterfangen, denn die meisten Ausgabegeräte besitzen keine spezielle Hardware, um ihre eigene Position im Raum zu ermitteln. Deshalb werden in vielen Systemen optische Verfahren für die Bestimmung der extrinsischen Eigenschaften eingesetzt, die direkt mit dem aufgenommenen Bild der Kamera arbeiten [AM04,KM07]. Diese Verfahren setzen außer einer Digitalkamera keine weitere Hardware voraus.

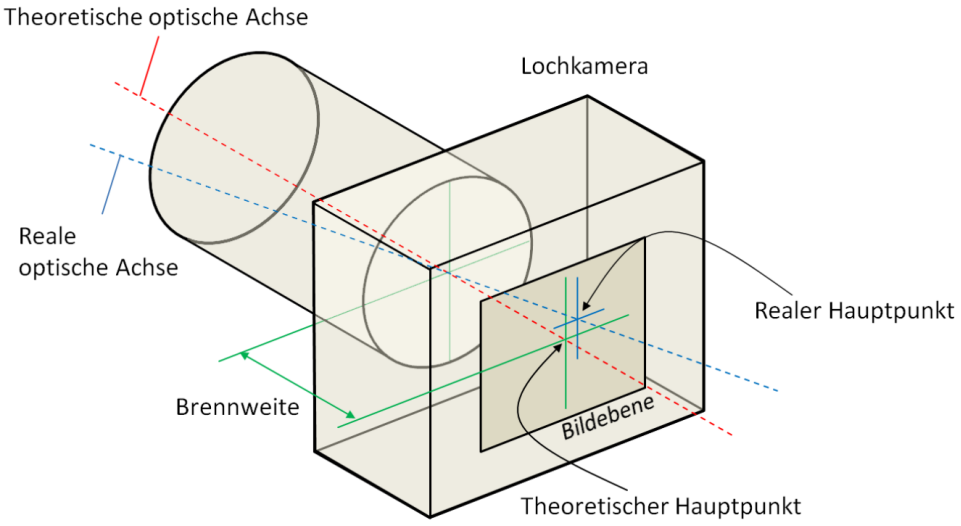

Abbildung 2.10: Intrinsische Eigenschaften einer Lochkamera.

Um die Position und Orientierung der Kamera durch ihre Bilder zu bestimmen, müssen die Bilder einen gewissen Informationsgehalt aufweisen. Prinzipiell ist es so, dass mehrere markante Punkte (sogenannte Features) im Kamerabild gesucht werden. Diese Features können entweder explizit durch den Benutzer in die Szene eingebracht worden sein, indem er zum Beispiel speziell ausgedruckte Bilder in der Szene verteilt (sogenannte Fiducials oder Feature-Maps) oder es wird versucht, die Features aus der Umgebung automatisch zu extrahieren. Wenn man einige solcher Features findet und deren relativen Abstand in der Realität zueinander kennt, kann man mit dem Perspective-n-Point-Verfahren (PnP) die Position und Orientierung der Kamera bestimmen. Beim PnP-Verfahren wird ein Gleichungssystem auf Basis des Cosinussatzes formuliert (siehe auch Abbildung 2.11):

$$x_j^2 + x_{(j+1)\%n}^2 + 2x_j x_{(j+1)\%n} cos\theta_j - c_j^2 = 0. \tag{2.32}$$

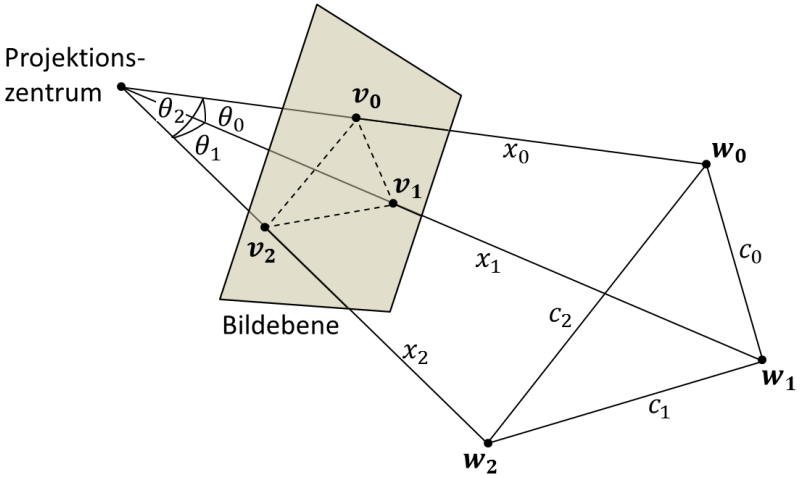

Abbildung 2.11: Größen beim P3P-Verfahren.

Für j gilt $j = 0, \dots, n-1$. Die n Gleichungen müssen nach x_j aufgelöst werden, wobei die relativen Abstände c_j zwischen den Feature-Punkten $\boldsymbol{w_j}$ und die Projektionen $\boldsymbol{v_j}$ auf der Bildebene bekannt sind. Für die Projektionen gilt $\boldsymbol{v_j} = M_{int}\boldsymbol{w_j}$. Die Winkel θ_j sind somit ebenfalls bekannt und können leicht anhand der Projektionen $\boldsymbol{v_j}$ ermittelt werden. Für $n = 3$ (P3P) existieren vier Lösungen für die Lage des Dreiecks und für $n = 4$ (P4P) ist das Gleichungssystem überbestimmt, dies jedoch nur, wenn die Feature-Punkte nicht kollinear liegen. Wurde die Lage des Dreiecks berechnet, kann aus dieser leicht die relative Lage der Kamera bestimmt werden. Zur Lösung des Gleichungssystems können die Verfahren aus [GHT*03] genutzt werden. Praktische Verfahren nutzen jedoch meistens deutlich mehr Feature-Punkte, um die Lage der Kamera zu berechnen.

Sind die relativen Abstände der Features zueinander unbekannt, wie dies bei der automatisierten Feature-Extrahierung der Fall ist, kommen Verfahren zur Anwendung, bei denen versucht wird, identische Features in konsekutiven Kamerabildern zu identifizieren (Feature-Matching). Wenn dies gelingt, lässt sich aus der relativen Verschiebung der Features in den Kamerabildern die Bewegung der Kamera und somit ihre relative Position und Orientierung nachvollziehen. Dies geschieht zum Beispiel beim Parallel-Tracking-and-Mapping-Verfahren (PTAM) von Klein und Murray [KM07].

2.2.2 Photometrische Registrierung

Die photometrische Registrierung ist ein wichtiger Aspekt von Augmented-Reality-Anwendungen, der die Erscheinung eines virtuellen Objekts in einer realen Szene natürlicher wirken lässt (vgl. Abbildung 2.12a & Abbildung 2.12c). Hier geht es darum, dass ein virtuelles Objekt so beleuchtet wird, als würde es tatsächlich in der realen Szene stehen. Geht man davon aus, dass das virtuelle Objekt nur reales Licht empfangen kann, jedoch keines in die Realität senden kann, dann ist die Registrierung relativ einfach (siehe Abbildung 2.12b). Für diesen Fall können so genannte Light-Probes in Form von Radiance-Maps genutzt werden. Eine Radiance-Map speichert die Leuchtdichte über einer (Hemi)-sphäre für einen bestimmten Ausgangspunkt in der Szene (ganz ähnlich zu einer Environment-Map). Die Erzeugung einer solchen Radiance-Map für eine AR-Anwendung kann auf sehr praktischem Wege geschehen: Man stellt eine Weitwinkelkamera (Fish-Eye-Lense) oder eine spiegelnde Kugel an den Ort, an dem das virtuelle Objekt stehen soll. Die eintreffende Leuchtdichte wird so aus Sicht des virtuellen Objekts gemessen. Verwendet man eine spiegelnde Kugel, so muss man anhand der sichtbaren Reflexionen im Kamerabild des Betrachters die Radiance-Map erzeugen. Sofern die Kugel nicht von mehreren Seiten betrachtet wird, ist die Radiance-Map dann schlecht aufgelöst und unvollständig.

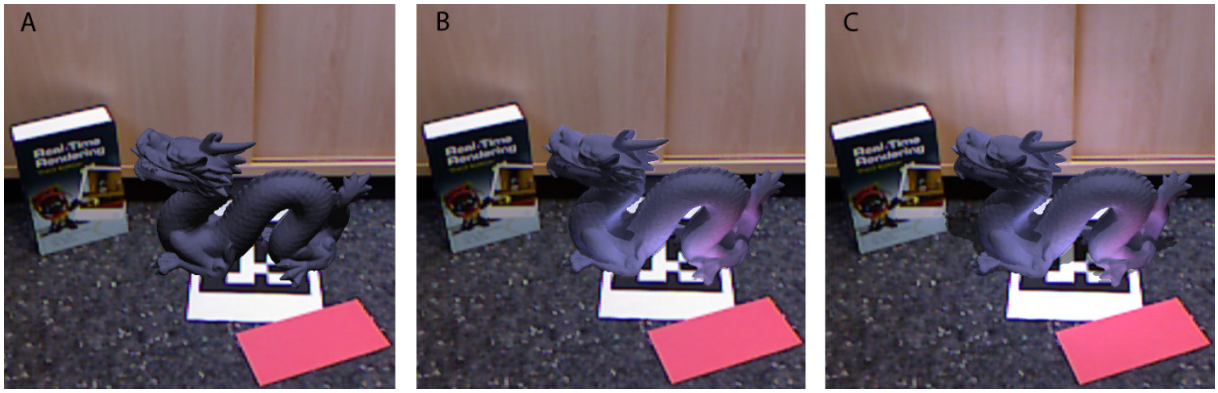

Abbildung 2.12: Verschiedene Arten der photometrischen Registrierung: (A) keine photometrische Registrierung, (B) photometrische Registrierung, bei der nur das virtuelle Modell Licht empfängt, (C) photometrische Registrierung, bei der das virtuelle Modell Licht empfängt und es in die reale Szene reflektiert (man beachte die bläuliche Reflexion an der Schrankwand und den Schattenwurf durch das virtuelle Modell).

Die erzeugte Abbildung in Form einer Radiance-Map beinhaltet direktes und indirektes Licht aus der Szene. Dies führt dazu, dass die Abbildung der Leuchtdichte einen hohen Wertebereich abdecken muss. Eine Speicherung im 24-Bit-RGB-Farbraum ist dann keine adäquate Repräsentation. Hier empfiehlt es sich, stattdessen mit entsprechenden HDR-Kameras zu arbeiten. Die Verwendung von HDR-Kameras schränkt aber die Anwendbarkeit eines Verfahrens stark ein, weswegen in vielen Anwendungen die HDR-Bilder aus mehreren Aufnahmen mit einer normalen Kamera gebildet werden. Dafür müssen die Bilder mit verschiedenen Öffnungszeiten aufgenommen werden. Debevec und Malik [DM97] beschreiben eine Methode, mit der man aus diesen Bildern ein HDR-Bild erzeugen kann.

Durch die Aufhebung der obigen Annahme, dass virtuelle Objekte kein Licht in die reale Szene reflektieren können und ebenfalls kein reales Licht blockieren, wird die photometrische Registrierung deutlich aufwändiger (siehe Abbildung 2.12c). Wenn reale Objekte virtuelles Licht reflektieren sollen, müssen deren Oberflächeneigenschaften bekannt sein. Dies betrifft sowohl die geometrischen Eigen-

schaften der Oberfläche, wie die Position und Normale, als auch die Reflexionseigenschaften in Form der BRDF. Da aber bei AR-Anwendungen üblicherweise von der realen Szene nur eine zweidimensionale Abbildung aus Sicht der Betrachterkamera vorliegt, kann man solche Reflexionen nicht ohne weiteres berechnen. Deshalb wird bei vielen Verfahren vorausgesetzt, dass die reale Szene als approximiertes virtuelles Modell vorhanden ist. Das Modell muss dann in einem aufwändigen Prozess von Hand im Voraus rekonstruiert werden. Dies geschieht üblicherweise mit einem 3D-Modellierungsprogramm. Mit dem Modell ist es dann möglich, Lichtwechselwirkungen zu berechnen, indem man ein globales Beleuchtungsverfahren auf dieses und den erweiterten Content anwendet. Die nötigen direkten Lichtquellen für die Bildsynthese können aus einer Radiance-Map generiert werden, während die Reflexionen auf Basis des virtuellen Modells berechnet werden. Es existieren in diesem Fall zwei Bilder: die Fotografie der realen Szene und die korrekt beleuchtete virtuelle Szene, die die Rekonstruktion der realen Szene enthält. Da die Rekonstruktion der realen Szene nur approximativ ist, kann man diese beiden Bilder nicht einfach miteinander mischen, ohne dass deutliche Übergänge zu erkennen sind (siehe Abbildung 2.13, unten rechts). Durch die Verwendung des Differential Renderings (DR) lassen sich diese Artefakte entfernen. Dieses wurde zuerst von Fournier et al. [FGR92] vorgestellt und später von Debevic [Deb08b] weiterentwickelt. Das Prinzip des Differential Renderings ist relativ intuitiv und soll hier kurz erklärt werden.

Beim Differential Rendering existieren vier Bilder von der Szene:

1. I_r: die Aufnahme der realen Szene ohne Augmentierung,
2. I_v: die virtuelle Rekonstruktion der realen Szene mit globaler Beleuchtung,
3. I_{va}: die virtuelle Rekonstruktion der realen Szene mit Augmentierung durch virtuelle Objekte mit globaler Beleuchtung,
4. I_m: eine binäre Maske für die Trennung von augmentiertem von realem Inhalt.

Das finale Bild für die Darstellung ergibt sich aus der folgenden Berechnung (siehe Abbildung 2.13):

$$I_{Final} = I_m I_r + (I_{va} - I_m I_v) \quad \leftrightarrow \quad I_{Final} = I_{va} + I_m(I_r - I_v). \tag{2.33}$$

Es wird ein Differenzbild $(I_{va} - I_m I_v)$ erzeugt, das den augmentierten Inhalt enthält und sämtliche Beleuchtungsveränderungen, die dieser hervorruft. Dieses Differenzbild muss dann nur noch auf die Aufnahme der realen Szene addiert werden, wobei in dieser der Bereich der Erweiterung schwarz ausmaskiert ist (durch die Multiplikation mit I_m).

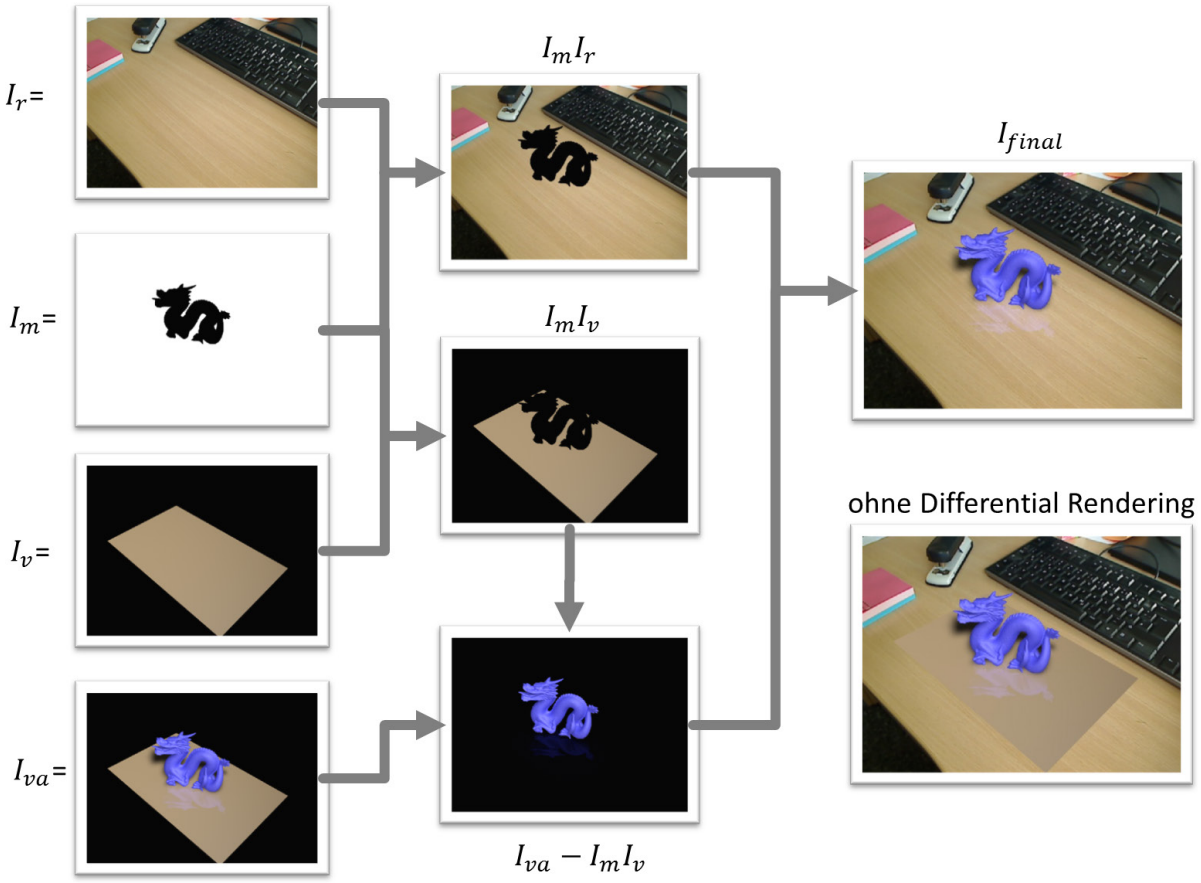

Abbildung 2.13: Die einzelnen Stufen des Differential Renderings. Unten rechts ist ein Vergleichsbild ohne Anwendung des DR, man erkennt deutliche Übergänge auf der Tischplatte.

Die Verwendung des Differential Renderings mit einem a priori rekonstruierten Modell der realen Szene erlaubt es, jedes interaktive globale Beleuchtungsverfahren für die Bildsynthese für Augmented-Reality-Anwendungen einzusetzen. Dann muss aber gelten, dass die reale Szene über die Laufzeit der Anwendung statisch bleibt, denn durch die manuelle Rekonstruktion sind Änderungen zur Laufzeit ausgeschlossen. Dies ist ein elementarer Nachteil von vielen Verfahren, die eine photometrische Registrierung vornehmen. In dieser Arbeit wird ein Verfahren vorgestellt, das auf die a-priori-Rekonstruktion verzichtet, indem es einen aktiven RGB-D-Sensor einsetzt. Hierdurch können globale Beleuchtungsphänomene in Echtzeit für dynamische virtuelle und reale Szenen berechnet werden.

Im folgenden Unterkapitel werden diese Sensoren genauer beschrieben.

2.2.3 Sensoren

Im vorherigen Unterkapitel wurde bereits erwähnt, dass für eine umfassende photometrische Registrierung zusätzliche Informationen zur realen Szene bekannt sein müssen. Diese Informationen können durch die Verwendung zusätzlicher bzw. erweiterter Sensoren bereitgestellt werden. Prinzipiell unterscheidet man zwischen passiven und aktiven Sensoren: Passive Sensoren belassen die reale Szene unverändert, während aktive Sensoren Veränderungen an der Szene vornehmen, um zusätzliche Informationen zu erhalten. Ob es sich um einen aktiven oder passiven Sensor handelt, hängt nicht allein von der Hardware, sondern auch von dessen Verwendungszweck ab. So gilt eine Digital-

kamera als passiver Sensor, wenn sie vom Betrachter direkt verwendet wird. Wird sie jedoch in der Szene platziert, um zum Beispiel eine Radiance-Map zu erzeugen, gilt sie als aktiver Sensor.

Eine umfassende photometrische Registrierung setzt voraus, dass die Geometrie der Realität bekannt ist. Möchte man diese nicht im Vorhinein manuell nachbauen, muss sie durch entsprechende Sensoren eingelesen bzw. erzeugt werden. Prinzipiell werden dann Sensoren verwendet, die zu einem (HDR-)Farbbild ebenfalls eine Tiefenabbildung der Szene liefern. Durch diese Sensoren kann jedem Farbpixel ein korrespondierender Tiefenwert zugeordnet werden. Der Tiefenwert bildet hier den Abstand vom Sensor zum Oberflächenpunkt ab. Für die Erzeugung solcher Tiefenwerte kommen passive und aktive Verfahren zur Anwendung. Um dieses Kapitel kompakt zu halten, werden nur Sensortechniken betrachtet, die es bereits als Produkt für eine breite Masse von Konsumenten zu erwerben gibt.

Bei passiven Verfahren werden oft stereoskopische Kamerasysteme verwendet. Hier besteht der Aufbau aus zwei Kameras, deren Lage zueinander bekannt ist. Darüber hinaus sind ebenfalls ihre intrinsischen Eigenschaften bekannt. Die räumliche Tiefe eines aufgenommenen Oberflächenpunkts kann mit diesem System bestimmt werden, indem man den Punkt in den jeweiligen Projektionen der Kameras eindeutig zuordnet. Die Findung solcher korrespondierenden Projektionspunkte wird in der Literatur oft als Korrespondenzanalyse bezeichnet. Durch die Epipolargeometrie (siehe Abbildung 2.14) ist bekannt, dass ein korrespondierender Punkt im Kamerabild 1 nur innerhalb der Epipolarlinie im Kamerabild 2 gefunden werden kann. Dies reduziert den Suchraum für Korrespondenzen deutlich. Da jedoch die Korrespondenzanalyse auf Basis der Farbinformationen im Kamerabild durchgeführt wird, ist sie ein fehleranfälliger Prozess. Bietet das Farbbild nicht ausreichend markante Punkte zur Orientierung, werden häufig falsche Korrespondenzen erkannt. Ein weiteres Problem entsteht aus der Tatsache, dass ein Oberflächenpunkt im Kamerabild 1 nicht zwingend im Kamerabild 2 zu sehen sein muss, so dass keine Korrespondenz hergestellt werden kann. Hierdurch wird die Tiefenabbildung unvollständig (häufig im Bereich von Objektsilhouetten).

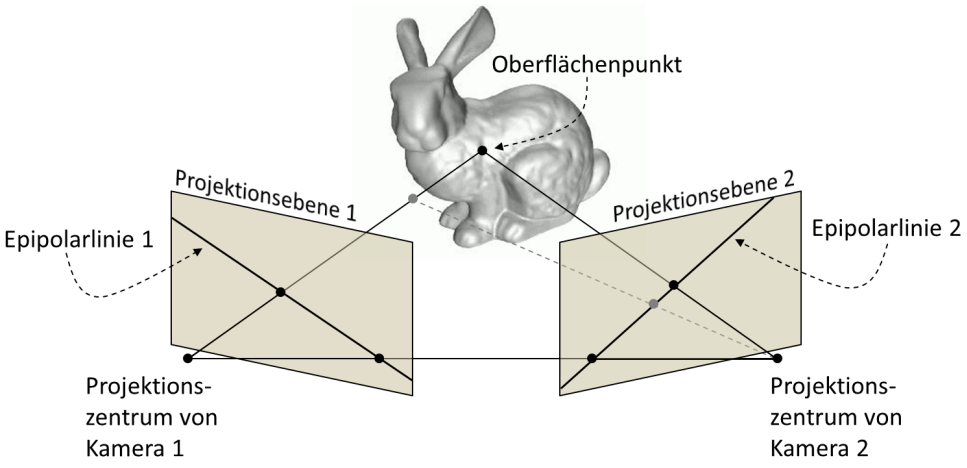

Abbildung 2.14: Epipolargeometrie: korrespondierende Projektionen eines Oberflächenpunkts müssen in den Epipolarlinien liegen.

Ein aktiver Sensor, der zur Tiefengewinnung ein ähnliches Prinzip wie stereoskopische Kamerasysteme anwendet, ist der Microsoft Kinect-Sensor, der auf der Technologie der Firma Primesense ltd. basiert (www.primesense.com). Anstatt mit zwei Kameras wird hier mit einer Infrarotkamera und

einem Infrarotprojektor gearbeitet. Der Projektor sendet ein strukturiertes Punktmuster in die Szene, dessen Projektion von der Infrarotkamera eingelesen wird. Für die ausgesendeten Infrarotstrahlen gilt ebenfalls das Prinzip der Epipolargeometrie, so dass die Projektionen der Reflexionen von diesen Strahlen innerhalb der Epipolarlinien liegen müssen. Der große Vorteil der Primesense-Technologie ist, dass die fehleranfällige Korrespondenzanalyse durch die Struktur des infraroten Punktmusters vereinfacht wird [ZSM*07]. Dieses Muster ist so konstruiert, dass über dessen Punktnachbarschaft der Ursprungsstrahl eines Punktes eindeutig identifiziert werden kann. Hierfür hat der Sensor ein verzerrungsfreies Referenzbild des Musters gespeichert. Wie bereits bei dem stereoskopischen Kamerasystem kann es aber passieren, dass ein infraroter Strahl auf einen Oberflächenpunkt trifft, der nicht im Kamerabild liegt. Zusätzlich kommt es zur Abschattung für die Bereiche, die kein infrarotes Licht empfangen. Es gibt noch weitere Probleme bei der Tiefenbilderzeugung mit der Kinect, die in [LB11] zusammengefasst wurden.

Seit November 2013 sind ebenfalls Time-of-Flight-Systeme (TOF), wie die Microsoft Kinect 2, zur Tiefenbildgenerierung als günstige Endgeräte erhältlich. Bei TOF-Systemen handelt es sich um aktive Sensoren, die Lichtstrahlen in die Szene senden und messen, wie lange die Lichtstrahlen benötigen bis sie als Reflexion zum Sender zurückkehren. Durch die endliche Ausbreitungsgeschwindigkeit des Lichts kann man so auf die zurückgelegte Entfernung schließen. Probleme bei der Messung treten auf, wenn die Oberflächen das Licht nicht zum Sender zurück reflektieren, wie dies zum Beispiel bei spiegelnden und transparenten Oberflächen der Fall ist. Auch wenn der Lichtstrahl sehr flach auf die Oberfläche trifft, kann es sein, dass das reflektierte Licht nicht ausreicht, um eine Messung durchzuführen. So sind die Tiefenabbildungen von TOF-Systemen häufig ebenfalls unvollständig.

Die Tiefenbilder aller hier vorgestellten Sensoren können nicht direkt für die photometrische Registrierung verwendet werden, denn ihre Qualität ist nicht ausreichend. Sie sind oft unvollständig und enthalten sowohl nieder- als auch hochfrequentes Rauschen. In Kapitel 7.2.2 wird ein Verfahren zur Filterung der Tiefenbilder vorgestellt, welches deren Anwendung für die photometrische Registrierung erlaubt. Darüber hinaus existieren Verfahren, die die Tiefenbilder nutzen, um ein vollständiges geometrisches Abbild der realen Szene zu erzeugen. Diese Abbildung wird dann zwar wieder im Voraus erzeugt, wie dies bereits bei der manuellen Rekonstruktion der Fall war, jedoch ist die Erzeugung wesentlich einfacher und nahezu vollautomatisiert. Newcombe et al. [NIH*11] und Izadi et al. [IKH*11] haben diesbezüglich das KinectFusion-Verfahren vorgestellt.

3 Aktueller Forschungsstand

3.1 Interaktive globale Beleuchtung

Die schnelle Entwicklung moderner Grafikkarten hat dazu beigetragen, dass in den letzten Jahren sehr viele interaktive globale Beleuchtungsverfahren vorgestellt wurden, die speziell für die Ausführung durch die GPU optimiert sind. Dabei basieren die Verfahren auf verschiedenen grundlegenden Techniken der globalen Lichtsynthese. 2012 haben Ritschel et al. [RDG*12] in ihrem Artikel „The State of the Art in Interactive Global Illumination“ die wichtigsten Verfahren zusammengefasst und bewertet. Diese Zusammenfassung und Bewertung bildet die Grundlage für dieses Kapitel. So wurden die Bewertungsergebnisse aus [RDG*12] übernommen, um eine relativ objektive Bewertung der Verfahren zu gewährleisten. Da in den letzten zwei Jahren jedoch weitere Verfahren vorgestellt wurden, die in dieser Arbeit ebenfalls berücksichtigt werden müssen, beruhen einige Bewertungen auf den Einschätzungen des Autors dieser Arbeit. Es wurde versucht, die Bewertung aufgrund von analytischen Überlegungen durchzuführen, und es wurde stets darauf geachtet, dass dies im Kontext der anderen Verfahren geschieht, die auf der gleichen Technik basieren.

Die wissenschaftlichen Arbeiten im Bereich der interaktiven globalen Beleuchtung lassen sich nach [RDG*12] in acht essentielle Verfahren (Techniken) aufteilen:

1. Monte-Carlo-Ray-Tracing,
2. Photon-Mapping,
3. Precomputed Illumination,
4. Discrete Ordinate Methods,
5. Finite Elements,
6. Instant Radiosity,
7. Many Light Approaches und
8. Point-Based Global Illumination.

Jedes dieser essentiellen Verfahren bildet ein einzelnes Unterkapitel in dieser Arbeit, jedoch wurden die letzten beiden Verfahren zu einem Unterkapitel zusammengefasst, da sie sich in ihren Ansätzen sehr ähneln. Am Ende eines jeden Kapitels befindet sich eine Bewertungstabelle, in der alle Verfahren aufgeführt sind, die im Kapitel vorgestellt wurden. Die Bewertung basiert auf den folgenden Kategorien, die aus [RDG*12] übernommen wurden:

- **Geschwindigkeit**: klassifiziert die Bildwiederholrate des Verfahrens. Eine Wertung von 5 bedeutet, dass das Verfahren Ergebnisse mit mehr als 100 FPS erzeugt, während eine 1 bedeutet, dass das Verfahren Bilder erst nach mehreren Sekunden (1-5s) liefern kann.
- **Qualität**: klassifiziert die subjektiv wahrgenommene Darstellungsqualität des Verfahrens. Hier geht es vorrangig darum, dass das Verfahren plausible, artefaktfreie und temporär kohärente Bilder erzeugt. Die physikalische Korrektheit spielt eine untergeordnete Rolle, da die meisten Verfahren approximative Lösungen erzeugen, die nur näherungsweise korrekt sind.
- **Dynamik**: klassifiziert, wie dynamisch die Szenen bei den Verfahren sein dürfen. Hier wird unterschieden, ob die Kamera frei bewegt werden kann, ob die Materialien verändert werden können, ob das direkte Licht verändert werden kann und ob rigide Transformationen

oder komplette Verformungen auf Szenenmodelle angewendet werden können. Dabei wird berücksichtigt, wie viel Extraaufwand durch die Animationen im Verfahren entsteht. Ein Wert von 0 bedeutet, dass das Verfahren überhaupt keine Dynamik erlaubt, während 5 ein voll dynamisches Verfahren darstellt.

- **Skalierbarkeit**: klassifiziert, wie gut das Verfahren mit steigender Auflösung und Szenenkomplexität skaliert.
- **Implementierung**: klassifiziert die Komplexität des Verfahrens bei der Implementierung. Hier geht es um die Komplexität bzw. den quantitativen Umfang des Quelltexts, nicht um die mathematischen Grundlagen.
- **GPU**: klassifiziert, in wie weit das Verfahren für die Parallelisierung auf der Grafikkarte geeignet ist, bzw. wie gut es mit zukünftigen Grafikkarten skaliert.

Ritschel et al. [RDG*12] betonen, dass die Bewertung in diesem Kategoriensystem lediglich einen vergleichenden Charakter der Verfahren untereinander hat. Insbesondere die Kategorie Qualität enthält subjektive Bewertungen dieser Experten. Im Rahmen dieser Dissertation war es nicht möglich, eigene Messungen zu allen hier vorgestellten Verfahren durchzuführen, da der Autor dieser Arbeit keinen Zugriff auf die einzelnen Implementierungen hatte.

3.1.1 Monte-Carlo-Ray-Tracing

Ray-Tracing (häufig auch Path-Tracing genannt) ist ein etabliertes Verfahren in der Offline-Bildsynthese [Suf07,PH10]. Das einfache Prinzip und die hohe Parallelisierbarkeit machen es aber ebenfalls für die interaktive Bildsynthese interessant. Beim Ray-Tracing wird ein Strahl vom Auge des Betrachters zum Bildpunkt (Pixel) gebildet. Dieser Strahl wird durch die Szene verfolgt, bis dieser auf eine Lichtquelle trifft oder die verbleibende Energie des Strahls zu gering wird. Trifft der Strahl auf ein Objekt, wird er je nach Material des Objekts gebrochen und reflektiert. Dies führt dazu, dass rekursiv weitere Strahlen durch die Szene verfolgt werden müssen. Die Leuchtdichte eines Oberflächenpunktes ergibt sich, indem man die empfangene Leuchtdichte entlang des Strahls mit der BRDF des Oberflächenpunktes verrechnet.

Klassischerweise lassen sich Szenen mit verspiegelten und transparenten Materialien besonders effizient mit dem Ray-Tracing-Verfahren berechnen. Bei diesen Materialien wird das Licht sehr gerichtet reflektiert, so dass sich nur wenige neue Strahlen für die Verfolgung ergeben. Geht man dagegen von stark diffusen Oberflächen aus, wird das Licht gleichmäßiger gestreut und es müssen sehr viele Strahlen verfolgt werden, um ein zufriedenstellendes Ergebnis zu erzeugen. Dieses Problem wird durch das Monte-Carlo-Ray-Tracing in besonderer Weise adressiert.

Um die reflektierte Leuchtdichte eines Punktes zu berechnen, muss eine Lösung für die Rendergleichung (2.3) gefunden werden. Diese Gleichung enthält ein Integral über die Hemisphäre des betrachteten Oberflächenpunkts. Die Lösung des Integrals mit Hilfe des Ray-Tracings würde erfordern, dass eine unendliche Menge von Strahlen über die Hemisphäre gebildet wird. Bei der Monte-Carlo-Methode geht es darum, eine gute Approximation des Integrals mit einer endlichen Menge von Strahlen zu bilden. Ist jeder Strahl $\boldsymbol{x_i}$ nach einer Dichte p zufällig gewählt, so besagt die Monte-Carlo-Integration, dass

$$\int_{\Omega^+} f(\boldsymbol{x})dx \approx \frac{1}{n}\sum_{i=1}^{n}\frac{f(\boldsymbol{x_i})}{p(\boldsymbol{x_i})}, \quad \boldsymbol{x_i} \in \Omega^+. \tag{3.1}$$

In dieser Gleichung kennzeichnet $p(\boldsymbol{x_i})$ die Wahrscheinlichkeitsdichte des Strahls. Für die Konvergenz der Approximation gilt, dass eine Vervierfachung der verfolgten Strahlen den Fehler der Approximation halbiert. Wie viele Strahlen letztendlich zur Approximation verwendet werden, hängt vom jeweiligen Szenario ab. Um systematische Synthesefehler zu verstecken, wird die Verteilung der zufälligen Strahlen von Oberflächenpunkt zu Oberflächenpunkt variiert. Hierdurch werden die Fehler in Rauschen umgewandelt.

Ray-Tracing ist sehr rechenintensiv, denn es muss eine Vielzahl von Strahlen verfolgt werden. Nichtsdestotrotz existieren Verfahren, die bei moderater Szenenkomplexität interaktive Ergebnisse erzeugen. Wald et al. [WKB*02] haben bereits 2002 ein dezentrales System vorgestellt, dass in der Lage ist, dynamische Szenen interaktiv mit ca. 5 FPS auf 16 handelsüblichen PCs zu berechnen. In ihrem System werden verschiedene Optimierungsmethoden, wie zum Beispiel eine optimierte Quasi-Monte-Carlo-Integration [Owe98] und die Bündelung von kohärenten Strahlen, genutzt. Alle Berechnungen werden auf der CPU durchgeführt, weswegen mehrere PC-Systeme notwendig sind. In späteren Verfahren wird dagegen die parallele Architektur der Grafikkarte genutzt, um interaktive Ergebnisse zu erzeugen. Purcell et al. [PBM*02] waren mit die Ersten, die ein Ray-Tracing-Verfahren auf der Grafikkarte implementiert haben.

Ein großes Problem bei der Implementierung von Ray-Tracing auf der GPU ist der inkohärente Speicherzugriff und die unterschiedliche Lebensdauer der Strahlen. Diese Phänomene haben beide ihren Ursprung in der Inkohärenz der Strahlen. Moderne Grafikkarten sind für den sequentiellen Speicherzugriff optimiert, weswegen stark gestreute Zugriffe zu langen Wartezyklen führen. Die inkonsistente Lebensdauer der Strahlen sorgt wiederum dafür, dass die Arbeitslast sehr ineffizient auf die einzelnen Threads verteilt wird. Novák et al. [NHD10] und später darauf aufbauend van Antwerpen [van11] stellen Verfahren vor, die die Arbeitslast auf der GPU effizienter verteilen. Nießner et al. [NSS10] umgehen das Problem der komplexen Strahlenverfolgung und der damit einhergehenden schlechten Kohärenz, indem sie die Szene in gestapelte parallelprojizierte Tiefenbilder zerlegen, die jeweils für verschiedene Richtungen erzeugt werden. Diese Tiefenbilder werden im Vorhinein erzeugt und belegen relativ viel Speicher. Die Schnittpunktberechnung eines Strahls kann mit Hilfe der Tiefenbilder erfolgen. Ein Strahl wird in den Raum des Tiefenbilds transformiert, in dem dann die Prüfung der Kollision stattfindet. Hierfür müssen nur wenige Pixel des Tiefenbildes betrachtet werden. Durch die approximative Darstellung der Szene ist der gefundene Kollisionspunkt nicht sehr genau, was aber für den subjektiven Eindruck der Szene keine große Rolle zu spielen scheint. Nießner et al. können mit ihrem Verfahren moderat komplexe Szenen mit 15-27 FPS darstellen, wobei nur die erste indirekte Reflexion berechnet wird.

Alle in diesem Kapitel vorgestellten Verfahren benutzen geometrische Beschleunigungsstrukturen, um eine effiziente Strahlenverfolgung zu gewährleisten. Üblicherweise kommen hier KD-Trees [Ben75] zum Einsatz. Diese Beschleunigungsstrukturen müssen bei dynamischen Szenen kontinuierlich aktualisiert werden, was für zusätzlichen Rechenaufwand sorgt. Zwar existieren Methoden, die KD-Trees innerhalb weniger Millisekunden mit Hilfe der GPU erzeugen [ZHW*08], aber bei einer Bildwiederholrate von 30-60 FPS sind diese Konstruktionszeiten bereits ein nicht zu

unterschätzender Einflussfaktor. Bei der Traversierung des KD-Trees muss ebenfalls darauf geachtet werden, dass die Verwendung von Arrays bzw. Stacks nicht effizient durch die Register der GPU abgebildet werden kann. Hierfür existieren spezielle Traversierungsverfahren, die ohne entsprechende Stacks auskommen [PGS*07].

Eine sehr bekannte und häufig genutzte Implementierung eines GPU-Ray-Tracing-Systems ist NVidias OptiX-Engine [PBD*10].

Man kann festhalten, dass die interaktiven Beleuchtungsverfahren im Bereich des Monte-Carlo-Ray-Tracings eine gute Beleuchtungsqualität erreichen können, die aber mit langen Berechnungszeiten einhergehen (vgl. folgende Tabelle). Animationen sind wegen der geometrischen Beschleunigungsstrukturen schwierig zu implementieren. Ein direkter Vergleich eines Ray-Tracing-Verfahrens zum, in dieser Arbeit entwickelten, LightSkin-Verfahren ist demnach nicht sinnvoll.

Tabelle 3.1: Bewertungstabelle nach [RDG*12] für interaktive Monte-Carlo-Ray-Tracing-Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Wald et al.	[WKB*02]	1	4	3	4	3	3	$L(D\|S)^+E$
Novák et al.	[NHD10]	1	4	3	4	4	3	$L(D\|S)^+E$
van Antwerp.	[van11]	2	5	3	4	3	3	$L(D\|S)^+E$
Nießner et al.	[NSS10]	2	2	1	3	4	4	$LD(D\|S)E$

3.1.2 Photon-Mapping

Photon-Mapping (PM) und Ray-Tracing sind vom Prinzip her eng miteinander verwandt. Bei beiden Verfahren werden Strahlen durch die Szene verfolgt. Während jedoch beim Ray-Tracing die Strahlen vom Betrachter zur Lichtquelle verfolgt werden, wird beim Photon-Mapping (im ersten Schritt) der Weg des Lichts von der Lichtquelle zum Auge verfolgt. Dabei versteht man einen verfolgten Strahl als Menge von Photonen, die von Oberflächen absorbiert oder gestreut werden. Jensen [Jen96] hat das Photon-Mapping bereits 1996 vorgestellt und wenn man so will, kann man es als komplementäre Technik zum Monte-Carlo-Ray-Tracing verstehen. Im Gegensatz zum Ray-Tracing ist das Photon-Mapping besonders gut für die Synthese von Kaustiken und diffusen Reflexionen geeignet, während spiegelnde und stark glänzende Reflexionen nicht adäquat dargestellt werden können. Das Bi-Directional Path-Tracing [LW93] kann man als Hybrid der beiden Techniken verstehen, bei dem die Strahlen aus beiden Richtungen verfolgt werden. Ein Problem beim Photon-Mapping ist, dass man nicht im Voraus weiß, wo die Photonen in der Szene landen und ob ausreichend viele Photonen auf eine Oberfläche treffen. Im Normalfall wird nicht jeder sichtbare Oberflächenpunkt Photonen empfangen, so dass man eine stark verrauschte Darstellung erhält. Um dieses Problem zu beheben, werden zwei Schritte beim Photon-Mapping durchgeführt (siehe auch Abbildung 3.1):

1. Ray-Tracing: Die Verfolgung der Photonen aus Sicht der Lichtquelle. Immer wenn ein Photon auf eine Oberfläche trifft, wird es in der Photon-Map gespeichert.
2. Photon-Density-Estimation: Die durchgehende Schattierung auf Basis der Photonendichte in der Region eines sichtbaren Bildpunktes.

Für die effiziente Durchführung des Ray-Tracing-Schrittes gelten die gleichen Überlegungen wie im vorherigen Unterkapitel. Im zweiten Schritt des Verfahrens – der Photon-Density-Estimation – wird

versucht, auf Basis der losen Photonenverteilung die Leuchtdichte für alle Bildpunkte zu bestimmen. Dies kann für jeden Bildpunkt durch die folgende Formel ausgedrückt werden:

$$L(\boldsymbol{x}, \boldsymbol{\omega}) \approx \sum_{p \in Q(\boldsymbol{x})} \frac{\phi_p f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_p}) k(\|\boldsymbol{x_p} - \boldsymbol{x}\|)}{\sum_{p \in Q(\boldsymbol{x})} k(\|\boldsymbol{x_p} - \boldsymbol{x}\|)}. \tag{3.2}$$

L ist die Leuchtdichte des zu schattierenden Punktes $\boldsymbol{x}$ in Richtung $\boldsymbol{\omega}$. Für jedes Photon p wird der Lichtstrom ϕ_p, dessen Position $\boldsymbol{x_p}$ und die Richtung des eintreffenden Lichts $\boldsymbol{\omega_p}$ gespeichert. $Q(\boldsymbol{x})$ kennzeichnet eine Menge von Photonen, die im Bereich um $\boldsymbol{x}$ liegen. Die Funktion k wird als Kernel bezeichnet. Sie gibt an, wie stark ein Photon zum Ergebnis im Punkt $\boldsymbol{x}$ beiträgt, dabei wird diese Funktion normalisiert betrachtet (Teilung durch die Summe über alle k).

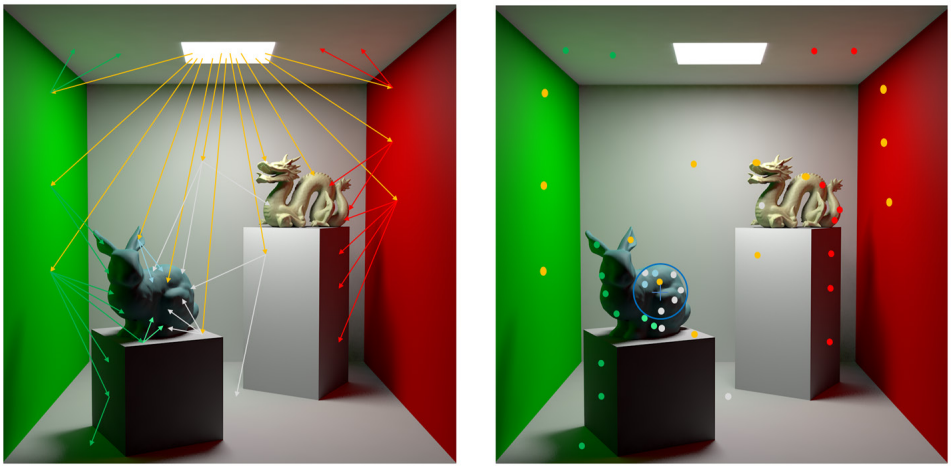

Abbildung 3.1: Die zwei Schritte des Photon-Mappings. Zuerst werden die Photonen aus Sicht der Lichtquelle durch Ray-Tracing verteilt (links) und anschließend wird die Photonendichte bestimmt, um die finale Schattierung zu erhalten (rechts, blauer Kreis).

Die Photon-Density-Estimation setzt voraus, dass die nächstgelegenen Photonen zu einem Punkt schnell gefunden werden können. Es kommen also entsprechende Verfahren zum Einsatz, die das k-Nearest-Neighbors-Problem (kNN) effizient lösen. Die Datenstruktur, in der die Photonen gespeichert werden, bezeichnet man als Photon-Map. Diese besteht in der Regel aus einer geometrischen Beschleunigungsstruktur, die ähnlich zu den Strukturen beim Ray-Tracing ist. Möchte man keine Beschleunigungsstruktur verwenden, bleibt die Möglichkeit, die Photonen nicht in jedem Oberflächenpunkt einzusammeln (Gathering), sondern den Einfluss der Photonen direkt in die Szene zu „splatten“. Das Splatting ist eine Technik, bei der jedes Photon ein Begrenzungsvolumen erhält. Wird eine Kollision eines Photons mit einer Oberfläche registriert, wird das Begrenzungsvolumen genutzt, um die Punkte innerhalb des Volumens zu aktualisieren. Das Splatting ist besonders im Zusammenhang mit dem Deferred Shading [ST90] interessant. Hier kann man einfach die Begrenzungsvolumina in die G-Buffer zeichnen (splatten), um deren Einfluss zu bestimmen. Krüger et al. [KBW06] haben so Kaustiken mit interaktiven Bildwiederholraten berechnen können. Einen guten Überblick über effiziente Methoden zur Photon-Density-Estimation geben Hachisuka et al. [HJB*12], während Mara et al. [MLM13] vor kurzem untersucht haben, welche Verfahren am besten für eine GPU-Implementierung geeignet sind.

Trotz der zweifachen Komplexität des PM, durch das Ray-Tracing und der Density-Estimation, existieren mehrere interaktive Implementierungen:

Bereits 2002 haben Dmitriev et al. [DBM*02] ein Verfahren vorgestellt, welches die temporäre Kohärenz von konsekutiven Bildern mit geringen Animationen bzw. Bewegungen ausnutzt, um effizient globale Beleuchtungseffekte zu berechnen. Hier wird nicht für jedes neue Bild das Photon-Mapping separat berechnet, sondern nur für den Teil des Bildes, der sich verändert hat. Um festzustellen, in welchem Bereich sich Änderungen befinden, wird eine kleine Menge von Pilot-Photonen ausgesandt. Trifft ein solches Pilot-Photon auf eine Veränderung, so werden weitere ähnliche Photonen ausgesandt, um den Bereich wieder konsistent zum Rest des Bildes auszuleuchten. Das Verfahren läuft zum Großteil auf der CPU, jedoch wird die direkte Beleuchtung durch ein Rasterverfahren mit Hilfe der GPU berechnet.

Purcell et al. [PDC*03] waren mit die Ersten, die PM komplett auf der Grafikkarte implementiert haben. Sie konnten durch die Parallelisierung auf der GPU interaktive Ergebnisse erreichen, die sich sukzessiv über mehrere Frames verbessern. Dabei wurde als Suchstruktur für die Photonen ein statisches kubisches Gitter (3D-Grid) verwendet. Durch dieses 3D-Grid sind die Speicheranforderungen des Verfahrens sehr hoch und deshalb werden größere Szenen von diesem nicht unterstützt. Das Verfahren ist jedoch in der Lage, sowohl weiche Schatten und Kaustiken, als auch glänzende und diffuse Oberflächen abzubilden. Allerdings sind die Ergebnisse des Verfahrens relativ verrauscht.

Fabianowski und Dingliana [FD09] und Wang et al. [WZP*09] haben jeweils 2009 PM-Verfahren vorgestellt, die komplett auf der GPU implementierbar sind und Szenen von geringer Komplexität mit 2-5 FPS darstellen können. Fabianowski und Dingliana nutzen eine CUDA-Implementierung, um bis zu zwei indirekte Reflexionen für glänzende und diffuse Oberflächen zu realisieren, dies jedoch nur für statische Szenen mit dynamischem Licht und dynamischer Kamera. Die Interaktivität des Verfahrens wird dadurch erreicht, dass für die Reflexionen der Photonen so genannte Footprints erzeugt werden. Diese Footprints bilden die geometrische Ausbreitung des Photons ab (ähnlich zum Splatting). Diese Ausbreitung wird sehr kompakt durch Photon-Differentials [SFE*07] bestimmt. Basierend auf den Footprints wird dann eine kompakte Beschleunigungsstruktur erstellt, die für die Photon-Density-Estimation genutzt werden kann. Wang et al. dagegen verwenden einen KD-Tree für die Photonen und umgehen die Density-Estimation durch ein Final-Gathering-Verfahren. Beim Final Gathering wird nicht das einkehrende Licht der Photonen aus der unmittelbaren Nähe eines sichtbaren Oberflächenpunkts ermittelt, sondern es wird das austretende Licht von allen Photonen der Szene betrachtet, das im Oberflächenpunkt ankommt. So wirken die Photonen als indirekte Lichtquellen. Da beim Final Gathering mehrere Photonen betrachtet werden müssen, ist es in der Regel teurer als die Density-Estimation. Wang et al. optimieren jedoch die Kosten, indem sie nicht für jeden sichtbaren Oberflächenpunkt der Szene Final Gathering betreiben, sondern nur für wenige Caches. Die Caches sind Zentren von k-Means-Clustern [Mac67], die aus Sicht der Kamera erzeugt werden. Dabei wird beim Clustering ausgenutzt, dass homogene Oberflächen Licht gleichmäßig verteilen, was zu größeren Clustern führt. Schlussendlich wird ein Interpolationsverfahren verwendet, um eine durchgehende Schattierung zu erreichen. Sowohl das Verfahren von Fabianowski und Dingliana, als auch das von Wang et al. produzieren temporäre Artefakte bei Kameraanimationen.

Es existieren wenige PM-Verfahren, die Bildwiederholraten von mehr als 20 FPS bei animierten Szenen unterstützen. McGuire und Luebke [ML09] haben ein Bildraum-PM-Verfahren vorgestellt, welches für moderat animierte Szenen, bei denen nur geringe Veränderungen im Bildraum stattfinden, diese Wiederholraten erreicht. Es handelt sich dabei um einen hybriden Ansatz, bei dem sowohl die GPU als auch die CPU genutzt wird. Die Rastereinheit der GPU wird verwendet, um die Photonenverteilung nach der ersten auftretenden Reflexion zu bestimmen. Dann werden die weiteren Interreflexionen mit der CPU berechnet und in einem KD-Tree gespeichert. Schlussendlich werden alle sich ergebenen Photonen in den Bildraum der Kamera gesplattet, um so die finale Schattierung zu erhalten. Das Verfahren unterstützt dabei diffuse Reflexionen und Kaustiken, aber keine glänzenden Reflexionen. Des Weiteren verursacht die reine Betrachtung des Bildraums ungewollte Schlagschatten am Bildschirmrand. Yao et al. [YWC*10] haben ebenfalls ein schnelles Verfahren für dynamische Szenen vorgestellt, das vollständig für die GPU implementiert werden kann. Es arbeitet auch im Bildraum, jedoch werden hier mehrere Bilder von verschiedenen Standorten innerhalb der Szene genutzt, um eine durchgehende Repräsentation zu erzeugen. Das Verfahren liefert überzeugende Ergebnisse, jedoch können komplexe Szenen damit nur unzureichend beleuchtet werden, da es nicht möglich ist, diese durch eine geringe Menge von Aufnahmen aus verschiedenen Perspektiven zu repräsentieren.

Photon-Mapping ist ein vielversprechender Ansatz, der gegenwärtig von vielen Verfahren verwendet wird, um qualitativ hochwertige globale Beleuchtungseffekte zu erzeugen. Im interaktiven Bereich versagen jedoch die meisten Ansätze bei animierten Szenen: Es werden entweder keine Animationen unterstützt oder man kann entsprechende Artefakte beobachten. Der Grund hierfür liegt in der stochastischen, losen Verteilung der Photonen, die sich durch Animationen verändert. Um für diese Fälle temporär kohärente Bilder zu erzeugen, müssen sehr viele Photonen genutzt werden, was aus Geschwindigkeitsaspekten nicht immer möglich ist. Auch das Ray-Tracing und die Aktualisierung der Beschleunigungsstrukturen machen Animationen kostspielig.

Tabelle 3.2: Bewertungstabelle nach [RDG*12] für interaktive Photon-Mapping-Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Krüger et al.	[KBW06]	3	2	4	3	3	4	LS^+DE
Dmitriev et al.	[DBM*02]	3	3	3	3	3	3	$L(D\|S)^+DE$
Purcell et al.	[PDC*03]	3	3	1	2	2	3	$L(D\|S)^+E$
Fabia. & Ding.	[FD09]	2	3	1	3	3	4	$L(D\|S)^+E$
Wang et al.	[WZP*09]	3	4	3	3	1	4	$L(D\|S)^+E$
McG. & Lueb.	[ML09]	4	2	3	4	3	3	$L(D\|S)^+DE$
Yao et al.	[YWC*10]	4	2	4	3	2	4	$L(D\|S)^+E$

3.1.3 Precomputed Illumination

Einer der ältesten Ansätze zur interaktiven Darstellung globaler Beleuchtungseffekte besteht darin, vorberechnete Beleuchtungsergebnisse zu nutzen. Diese werden im Voraus durch Offline-Verfahren erzeugt und können dann in mehr oder weniger kompakter Form von der Anwendung verwendet werden. Dabei besteht die einfachste Möglichkeit darin, das Ergebnis der Beleuchtungsberechnung in separaten Texturen, so genannten statischen Light- bzw. Shadow-Maps, zu speichern. Dies war viele Jahre der Standard für interaktive Verfahren und das, obwohl keine Dynamik in der Szene abgebildet werden konnte. Um zu erlauben, dass die Kamera in der Szene frei bewegt werden kann, wurden nur diffuse Reflexionen in den Texturen gespeichert. Solche Reflexionen sind vom Betrachtungswinkel unabhängig, da sie in alle Richtungen gleich hell abstrahlen. Glänzende oder spiegelnde Reflexionen können nicht ohne Weiteres mit Light-Maps abgebildet werden (wohl aber approximativ über Environment-Maps). Veränderungen am direkten Licht oder an der Szenengeometrie können ebenfalls nicht vorgenommen werden, da diese aufwändige Neuberechnungen erforderlich machen.

In den letzten zehn Jahren sind relativ viele Verfahren vorgestellt worden, die das Problem der fehlenden Dynamik behandeln, ohne dabei das Prinzip der komplexen Vorberechnung aufzugeben. Typischerweise beschränken diese Verfahren die Dynamik der Szene auf bestimmte Teilaspekte, wie die Veränderung des Lichts, des Materials und der Kamera. Animationen bei Modellen können ebenfalls berücksichtigt werden, dies jedoch nur im eingeschränkten Umfang.

Eines der bekanntesten Verfahren ist der Precomputed Radiance-Transfer (PRT), welcher von Sloan et al. [SKS02] vorgestellt wurde. Er bildete den Beginn für eine Vielzahl an Publikationen in diesem Bereich. Der Ansatz erlaubt es, Veränderungen in der Beleuchtung interaktiv auf eine statische Szene mit diffus reflektierenden Materialien anzuwenden. Hierfür wird das eingehende Licht vom Orts- in den Frequenzraum transformiert und dort in Form von Spherical-Harmonics-Koeffizienten gespeichert. Bei den Spherical Harmonics (SH) handelt es sich um orthonormale Basisfunktionen, die über eine Einheitskugel definiert sind. Die Funktion des eingehenden Lichts über eine Sphäre eines Punktes $\boldsymbol{x}$ lässt sich dann in der folgenden Weise approximieren (siehe auch Gleichung (2.3)):

$$L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \approx \sum_{j=0}^{n^2} c_j y_j(\boldsymbol{\omega_i}). \tag{3.3}$$

In dieser Gleichung sind c_i die Koeffizienten und y_i die Basisfunktionen der Spherical Harmonics. Die Koeffizienten können über die Faltung der eingehenden Lichtfunktion mit den SH-Basisfunktionen bestimmt werden: $c_j = \int_\Omega L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) y_j(\boldsymbol{\omega_i}) d\boldsymbol{\omega_i}$. Dieses Integral lässt sich wiederum mit Hilfe des Monte-Carlo-Estimators (siehe Gleichung (3.1)) durch ein Offline-Verfahren im Vorhinein bestimmen. Auf die gleiche Weise lässt sich die diffuse BRDF in Form von SH-Koeffizienten beschreiben. Wenn man die BRDF direkt mit dem Geometrieterm aus der Rendergleichung (2.3) multipliziert, so erhält man die Koeffizienten der so genannten Transferfunktion: $k_j = \int_\Omega f_r(\boldsymbol{x}, \boldsymbol{\omega_i})(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ y_j(\boldsymbol{\omega_i}) d\boldsymbol{\omega_i}$ (da die BRDF nur diffuse Reflexionen abbilden soll, muss ihr kein differentieller Austrittswinkel übergeben werden). Den großen Vorteil der Darstellung der Funktionen als SH-Koeffizienten beschreibt der folgende Zusammenhang:

$$
\begin{aligned}
L_o(\boldsymbol{x}) &= \int_{\Omega} f_r(\boldsymbol{x}, \boldsymbol{\omega_i})(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^{+} L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) d\boldsymbol{\omega_i} \\
&\approx \int_{\Omega} \left(\sum_{j=0}^{n^2} c_j y_j(\boldsymbol{\omega_i}) \sum_{j=0}^{n^2} k_j y_j(\boldsymbol{\omega_i}) \right) d\boldsymbol{\omega_i} \\
&= \sum_{j=0}^{n^2} c_j k_j \int_{\Omega} y_j^2(\boldsymbol{\omega_i})\, d\boldsymbol{\omega_i} = \sum_{j=0}^{n^2} c_j k_j .
\end{aligned}
\tag{3.4}
$$

Der letzte Schritt der Gleichung resultiert aus der Orthonormalität der SH-Basen, es gilt: $\int_{\Omega} y_j(\boldsymbol{\omega_i}) y_k(\boldsymbol{\omega_i})\, d\boldsymbol{\omega_i} = \begin{cases} 1, wenn\, j = k \\ 0, sonst \end{cases}$.

Die Gleichung (3.4) zeigt, dass die Faltung des eingehenden Lichts mit der Transferfunktion, so wie es in der Rendergleichung formuliert ist, einfach approximiert werden kann, indem man das Skalarprodukt der zugehörigen SH-Koeffizienten bildet. Sloan et al. haben gezeigt, dass bereits 25 Koeffizienten (n=5) ausreichen, um für diffuse Reflexionen eine gute Approximation zu liefern. Das Skalarprodukt aus 25 Komponenten lässt sich einfach innerhalb eines Shader-Programms zur Laufzeit bestimmen. Dabei werden jedoch hochfrequente Anteile des Lichts entfernt. Umso mehr Koeffizienten verwendet werden, desto genauer wird die Approximation. Jedoch kann man aus Gründen der schnellen Berechenbarkeit nur SH-Basen niedriger Ordnung mit wenigen Koeffizienten verwenden. Es lassen sich also keine hochfrequenten Signale in Echtzeit abbilden. Kautz et al. [KSS02] haben den Ansatz jedoch erweitert, um auch niederfrequente glänzende Oberflächen-BRDFs zu unterstützen.

Die globale Beleuchtung auf Basis von Spherical Harmonics beschränkt das Licht nicht nur auf niederfrequente Anteile, sondern auch in der Art der Ausbreitung. So kann man mit dem SH-Ansatz nur unendlich weit entferntes direktes Licht abbilden. Bei der Betrachtung der Gleichung (3.3) lässt sich erkennen, dass das eintreffende Licht L_i sowohl von dem Raumwinkel $\boldsymbol{\omega_i}$ als auch der Position $\boldsymbol{x}$ des Oberflächenpunktes abhängt. Das bedeutet, jeder Oberflächenpunkt müsste eigene Koeffizienten für L_i berechnen bzw. speichern. Dann lässt sich aber die Beleuchtung der Szene nicht mehr interaktiv an einer zentralen Stelle verändern, da für jeden Punkt alle Koeffizienten neu bestimmt werden müssten. Stattdessen wird angenommen, dass das Licht aus unendlich weiter Entfernung kommt, so dass die relative Verschiebung der Punkte zueinander zu vernachlässigen ist. Es wird also das gleiche Prinzip wie bei den Environment-Maps verwendet, weswegen die SH-Koeffizienten für das direkte Licht in der Regel aus diesen gewonnen werden. Annen et al. [AKD*04] haben jedoch ein approximatives Verfahren auf Basis eines analytischen Gradienten der SH-Koeffizienten entwickelt, mit dem es möglich ist, Flächenlichtquellen abzubilden, die nicht im Unendlichen liegen.

Es ist nicht möglich, für jeden erdenklichen Oberflächenpunkt die SH-Koeffizienten zu berechnen. Klassischerweise werden deshalb die SH-Koeffizienten nur für jeden Vertex eines Modells berechnet. Dadurch werden feinere Details der Objektoberfläche nicht berücksichtigt. Sloan [Slo06] hat aber etwas später ein Verfahren vorgeschlagen, welches ebenfalls mit detaillierten Normal Maps funktioniert.

Spherical Harmonics sind nicht die einzigen Basisfunktionen, die geeignet sind, um globale Beleuchtungseffekte abzubilden. Ng et al. [NRH03,NRH04], Liu et al. [LSS*04] und später auch Wang et al. [WTL06] haben gezeigt, dass hierfür auch die Haar-Wavelet-Basis geeignet ist. Alle vier Verfahren können sowohl nieder- als auch hochfrequente Lichtanteile und Reflexionen in hinreichender Qualität abbilden. Jedoch wird auch bei diesen Verfahren davon ausgegangen, dass das Licht von einer unendlich weit entfernten Lichtquelle stammt. Diese Einschränkung wurde durch Sun und Ramamoorthi [SR09] aufgelockert, jedoch ist die Berücksichtigung der Translation des Lichts in den Wavelet-Koeffizienten sehr komplex. Weitere geeignete Basisfunktionen für die kompakte Repräsentation des Lichttransports sind Spherical-Radial-Basen [TS06,XJF*08] oder die Verwendung von nichtlinearen Gauß-Funktionen für glänzende Reflexionen [GKM*06,GKD07].

Zur Laufzeit lässt sich mit den bisher hier vorgestellten Verfahren lediglich die Beleuchtung der Szene ändern. Sun et al. [SZC*07] und Akerlund et al. [AUW07] können mit ihren Verfahren ebenfalls dynamische Materialien, also dynamische BRDFs, abbilden. Loos et al. [LAM*11, LNJ*12] können mit ihrem Modular-Light-Transport-Ansatz animierte Objekte für die indirekte Beleuchtung berücksichtigen. Darüber hinaus benötigt ihr Ansatz keine langen Vorberechnungen, da dieser mit vereinfachten Proxy-Objekten arbeitet. Die Proxy-Objekte bestehen aus zusammengesetzten Grundformen, für die die Lichttransportfunktion einfach zu bestimmen ist. Bereits während der Erzeugung der Proxy-Objekte können die globalen Wechselwirkungen betrachtet werden. Da die Proxy-Objekte die Szene nur sehr grob approximieren, können damit nur diffuse, niederfrequente Reflexionen abgebildet werden. Das Verfahren lässt sich sehr effizient implementieren, so dass es ebenfalls auf mobilen Geräten lauffähig ist (wie zum Beispiel Smartphones etc.).

Kürzlich wurde ein neuer Ansatz von Ren et al. [RWG*13] vorgestellt, der die globale Beleuchtung in Form von Radiance-Regression-Functions abbildet. Diese sind nicht-lineare Regressionsfunktionen, die die Reflexionen auf Oberflächenpunkten abbilden. Die Funktionen erhalten einen Eingabevektor, der die Position, Normale und Reflexionseigenschaften des Oberflächenpunkts beinhaltet, sowie den Betrachtungswinkel und den Einfallswinkel des Lichts. Für diesen Eingabevektor liefern die Funktionen einen Wert für die Reflexion im Oberflächenpunkt zurück. Um die Regressionsfunktionen zu bilden, wird ein neuronales Netzwerk benutzt, welches bei der Vorberechnung der Szene mit verschiedenen Trainingssamples angelernt wird (Zuordnung von Eingabevektor zu Ausgabewerten). Der Ansatz kann sowohl diffuse als auch glänzende Reflexionen abbilden und erzeugt ebenfalls Kaustiken. Das Licht ist nicht auf unendliche Entfernung beschränkt, jedoch lassen sich auch mit diesem Verfahren nur statische Szenen beleuchten.

Die in diesem Kapitel vorgestellten Precomputed-Illumination-Verfahren werden bereits in interaktiven Anwendungen verwendet und können somit als robust angesehen werden. Ein essentieller Nachteil der Verfahren ist die Beschränkung auf statische Szenen. Selbst wenn die Verfahren Veränderungen an den Szenen zulassen, so beschränken sich diese in der Regel auf die Rotation der Objekte (Verschiebungen sind nur bedingt möglich, siehe zum Beispiel [IDY*07]). Dies ist dem Umstand geschuldet, dass die meisten Verfahren nur direktes Licht in unendlich weiter Entfernung approximieren können. Die Verfahren eignen sich deshalb in besonderer Weise für Außenareale bei Tageslicht (Outdoor-Szenen), da hier diese Einschränkung nicht ins Gewicht fällt. Im Vergleich zum LightSkin-Verfahren dieser Arbeit sind prinzipiell die Ansätze von Loos et al. [LAM*11, LNJ*12]

interessant, da diese Verfahren Animationen in der Szene nicht generell ausschließen und sie sehr schnelle Ergebnisse liefern (mit mehr als 60 FPS).

Tabelle 3.3: Bewertungstabelle nach [RDG*12] für interaktive Precomputed-Illumination-Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Sloan et al	[SKS02]	4	2	2	2	4	4	LD^+E
Kautz et al.	[KSS02]	2	2	1	2	3	3	$L(D\|S)^+E$
Ng et al.	[NRH03]	2	5	1	2	3	2	LDE
Ng et al.	[NRH04]	2	3	1	2	3	3	$L(D\|S)^+E$
Annen et al.	[AKD*04]	2	3	1	2	2	3	$L(D\|S)^+E$
Liu et al.	[LSS*04]	3	3	1	2	4	4	$L(D\|S)^+E$
Sloan	[Slo06]	3	3	2	2	3	3	$L(D\|S)^+E$
Tsai, Shih	[TS06]	3	4	1	3	2	2	$L(D\|S)E$
Green et al.	[GKM*06]	2	4	1	2	2	3	$L(D\|S)^+E$
Green et al.	[GKD07]	5	3	1	3	3	4	$L(D\|S)E$
Akerl. et al.	[AUW07]	3	4	1	2	2	3	$L(D\|S)^+E$
Sun et al.	[SZC*07]	3	3	2	1	2	1	$L(D\|S)^+E$
Sun & Rama.	[SR09]	1	4	1	2	1	2	$L(D\|S)^+E$
Xu et al.	[XJF*08]	2	4	3	1	2	3	$L(D\|S)E$
Loos et al.	[LAM*11]	5	2	3	4	3	4	LD^+E
Loos et al.	[LNJ*12]	5	2	3	4	3	4	LD^+E
Ren et al.	[RWG*13]	4	5	1	2	2	3	$L(D\|S)^+E$

3.1.4 Discrete Ordinate Methods

Discrete Ordinate Methods (DOM) finden vor allem bei der Simulation zur Lichtausbreitung in optischen Medien Anwendung. Grundsätzlich versucht man, die Lichtausbreitung in einer diskreten dreidimensionalen Gitterstruktur zu simulieren, indem man den Lichttransport von einer Gitterzelle zu den nächstliegenden Gitterzellen (Nachbarn) berechnet. Dies geschieht dann über mehrere Iterationen, so dass sich das Licht im Medium nach und nach im Gitter verteilt (siehe Abbildung 3.2). Die Gitterzellen sind dann punktuelle Stichproben, für die die Lichtausbreitung bekannt ist. Ein durchgehendes Ergebnis erhält man, indem man die benachbarten Gitterzellen zu einem beliebigen Raumpunkt betrachtet und deren Lichtausbreitung für den Punkt trilinear interpoliert. Da die Lichtausbreitung nur in diskrete Richtungen zu den Nachbarzellen betrachtet wird, spricht man von Discrete Ordinates. Durch die diskrete Ausbreitung kann man zwei ungewollte Effekte bei DOM-Verfahren beobachten: den Ray-Effect und das „Verschmieren“ von Licht. Der Ray-Effect ergibt sich durch die diskreten Hauptrichtungen bei der Ausbreitung des Lichts, die eine gleichmäßige Verteilung des Lichts nicht korrekt abbilden und so eine Art Machband-Effekt erzeugen. Das Verschmieren von Licht entsteht wiederum durch die wiederholte Glättung des Lichts bei der Ausbreitung im Gitter über mehrere Iterationen.

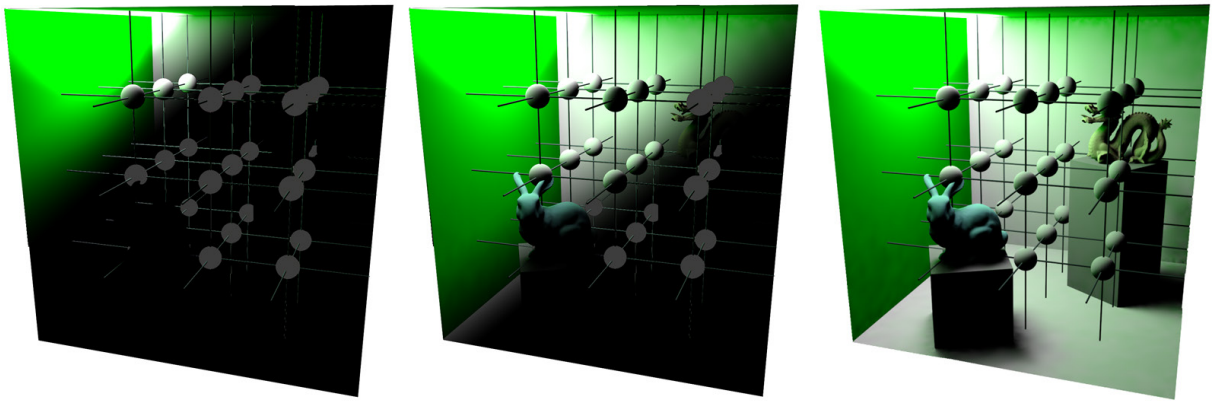

Abbildung 3.2: Prinzip der Lichtausbreitung bei Discrete Ordinate Methods. Im ersten Schritt wird das indirekte Licht aus der oberen linken Ecke in die direkt benachbarten Gitterknoten (Kugeln) injiziert und in den nächsten Schritten (Mitte, rechts) entlang der diskreten Achsen an die Nachbarknoten weiter verteilt.

Wegen der schrittweisen Ausbreitung des Lichts durch eine kubische Struktur sind die meisten DOM-Verfahren nicht interaktiv [GRW*04,Fat09]. Kaplanyan und Dachsbacher [KD10] haben jedoch mit ihren Cascaded Light-Propagation-Volumes (CLPV) eine sehr effiziente Variante des Verfahrens vorgestellt, welche die Berechnung der Lichtausbreitung in Echtzeit erlaubt. Mit diesem Verfahren ist es möglich, die diffuse Ausbreitung des indirekten Lichts abzubilden. Die Ausrichtung des Lichts wird dabei in den Gitterzellen über Spherical-Harmonic-Koeffizenten repräsentiert und so von Zelle zu Zelle weitergetragen und geglättet. Da das Gitter sehr grob ist und die Ausbreitungseigenschaften des Lichts nur durch wenige SH-Koeffizienten repräsentiert werden, lassen sich nur niederfrequente diffuse Reflexionen und Single-Scattering-Phänomene damit abbilden. Um große, komplexe Szenen zu unterstützen, wird mit mehreren Kaskaden der Gitterstruktur gearbeitet: Die Lichtausbreitung in der Nähe des Betrachters wird dabei in einem feinen Gitter und fernes Licht in einem groben Gitter abgespeichert. Damit das Licht nicht durch massive Objekte scheint, wird für die Geometrie der Szene ebenfalls ein solches Gitter mit SH-Koeffizienten erstellt.

Die CLPV von Kaplanyan und Dachsbacher stehen in direkter Konkurrenz zum LightSkin-Verfahren, da sie in etwa die gleichen optischen Phänomene wie dieses abbilden können und darüber hinaus sehr schnelle Ergebnisse liefern. In Kapitel 6.2.3 ist deshalb ein direkter Vergleich zwischen dem LightSkin-Verfahren und den CLPV zu finden.

Tabelle 3.4: Bewertungstabelle nach [RDG*12] für interaktive Discrete-Ordinate-Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Kaplanyan & Dachsbacher	[KD10]	5	2	4	5	3	5	$LD^{+}(D\|V\|S)E$

3.1.5 Finite Elements

Die wohl bekannteste Finite-Elements-Methode ist der klassische Radiosity-Ansatz [GTG*84]. Die grundlegende Idee besteht darin, die Szene in eine diskrete Menge von endlichen Elementen zu zerlegen und den Lichtaustausch nur zwischen diesen Elementen zu berechnen. Dies geschieht unter der Annahme, dass die Reflexionen innerhalb der Elemente als konstant angesehen werden können (siehe Abbildung 3.3). Dabei können die Elemente zum Beispiel Pixel, Texel, Voxel oder so genannte Patches (auch Surfels genannt) sein. Patches sind endliche Oberflächenelemente, die eine Fläche, Ausrichtung und Position besitzen. Klassischerweise lassen sich mit Finite-Elements-Methoden keine

hochfrequenten (spiegelnden) Reflexionen abbilden, da diese vom Betrachtungswinkel abhängig sind. Um diese in ausreichender Form zu approximieren, müsste die Diskretisierung der Szene in Abhängigkeit zur Kamera, also pro Frame, vorgenommen werden, was nur bedingt möglich ist. Wenn man jedoch von rein diffusen Reflexionen ausgeht, so kann die komplette globale Beleuchtung im Voraus berechnet werden. Dabei ist die Lichtausbreitung im Wesentlichen von den geometrischen Eigenschaften der Szene abhängig, welche durch die Formfaktoren beschrieben werden. Ein Formfaktor ist zwischen zwei Patches definiert und gibt an, wie viel ein Patch von dem anderen Patch „sehen“ kann. Diese Angabe steht im Verhältnis zur kompletten sichtbaren Hemisphäre eines Patches. Ein Formfaktor bildet also in gewisser Weise den Geometrieterm aus Gleichung (2.8) ab. Wegen der Annahme, dass die Reflexion im Patch konstant ist, müssen in Bereichen der Szene, in denen die Helligkeit stark variiert, mehrere kleine Patches angelegt werden.

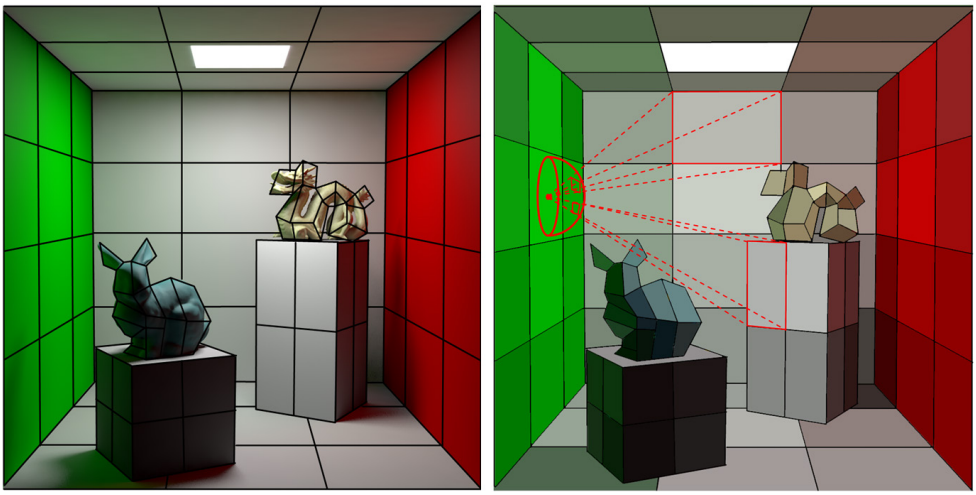

Abbildung 3.3: Das Prinzip von Finite-Elements-Methoden. Zuerst wird die Szene in eine Menge von endlichen Elementen aufgeteilt (links), im zweiten Schritt (rechts) werden die Formfaktoren genutzt, um den Lichtaustausch zwischen den Elementen zu berechnen. Dabei wird von einer konstanten Lichtverteilung pro Element ausgegangen.

Interaktive Finite-Elements-Methoden diskretisieren die Szene auf verschiedene Art und Weise, jedoch ist allen Verfahren gemein, dass sie einen Großteil der Berechnungen auf der Grafikkarte durchführen, um so interaktive Bildwiederholraten zu gewährleisten.

Coombe et al. [CHL04] waren mit die Ersten, die einen Radiosity-Ansatz nahezu komplett für die GPU implementierten. Sie haben hierfür den Progressive-Refinement-Ansatz (PR) von Cohen et al. [CCW*88] modifiziert, so dass dieser effizient die Raster-Hardware der Grafikkarte ausnutzt. Grundsätzlich handelt es sich beim PR-Ansatz um ein iteratives Verfahren. Hier wird das Licht zuerst von einer primären Lichtquelle (selbst ein Patch) anhand der Formfaktoren an die anderen Patches verteilt. Als nächstes wird bestimmt, welcher Patch die meiste „unverteilte“ spezifische Lichtausstrahlung (Radiosity) besitzt. Diese wird dann wiederum von diesem Patch an die anderen Patches verteilt, so dass der verteilende Patch danach keine „unverteilte“ Lichtausstrahlung mehr besitzt. Danach wird wieder geprüft, welcher Patch die meiste unverteilte spezifische Lichtausstrahlung besitzt und das Ganze wiederholt sich. Nach ein paar Iterationen erhält man bereits sehr gute Ergebnisse. Bei Coombe et al. sind die Patches Texel. Um das Licht von einem Texel zu den anderen zu verteilen, wird ein Hemicube [CG85] um den strahlenden Texel gebildet. Dieser beinhaltet alle sichtbaren anderen Patches als Pixel. Eine direkte Verteilung der Lichtausstrahlung an die im Hemicube

befindlichen Patches ist nun nicht sinnvoll, da dies einen verteilten und ineffizienten Schreibzugriff auf die Empfängertexturen zur Folge hätte. Stattdessen werden die Empfängertexturen separat und orthogonal zur Dreiecksnormalen gerastert. Dabei wird in einem Fragment-Shader geprüft, ob die ID des Texels im Hemicube vorhanden ist. Dies geschieht nach dem gleichen Rückprojektionsprinzip, welches auch beim Shadow-Mapping Anwendung findet. Um jeweils den nächsten Patch mit der höchsten unverteilten Lichtausstrahlung zu finden, wird der Z-Buffer effizient ausgenutzt. Es wird ein 1x1 Pixel großer Framebuffer erzeugt, in den alle Patches der Szene gezeichnet werden. Als Farbwert wird die Patch-ID und als Tiefenwert wird die inverse unverteilte Lichtausstrahlung gespeichert. So bleibt durch das Z-Buffering letztendlich nur die Patch-ID übrig, die die höchste unverteilte Lichtausstrahlung besitzt.

Im Gegensatz zu Coombe et al. verwendet Bunnell [Bun05] Dreiecke als Patches, um effizient Ambient Occlusion und diffuse indirekte Beleuchtung zu berechnen. Der Grad der Verdeckung wird für jeden Vertex eines Modells berechnet, indem geprüft wird, wie viel seiner Hemisphäre durch Dreiecke abgedeckt wird. Dies kann effizient durch analytische Formfaktoren bestimmt werden. Um die Formfaktoren möglichst schnell zu berechnen, werden die Dreiecke als Kreisflächen approximiert, für die der Formfaktor besonders einfach bestimmt werden kann. Damit nicht jeder Vertex mit jeder Kreisfläche getestet werden muss, wird eine hierarchische Struktur genutzt, bei der weit entfernte ähnliche Kreisflächen zu einer großen Kreisfläche zusammengefasst werden können. Diese Approximation ist legitim, wenn der Vertex einen ausreichend großen Abstand zum verdeckenden Objekt hat. Da nicht nur die erste sichtbare Kreisfläche als verdeckendes Objekt genutzt wird, sondern ebenfalls jede dahinter liegende Kreisfläche, kann es ungewollt zur doppelten Schattierung kommen (Occluder-Fusion), was sich wiederum in zu stark abgedunkelten Oberflächen äußert. Bunnell empfiehlt einen iterativen Ansatz, um den Effekt der doppelten Schattierung zu verringern.

Die doppelte Schattierung in Bunnell's Ansatz kommt durch die Vernachlässigung der Sichtbarkeitsprüfung zwischen den Patches zustande. Diese ist sehr aufwändig zu berechnen, da prinzipiell jeder Patch durch einen anderen Patch verdeckt werden kann. Man müsste also für jeden Patch eine Sichtbarkeitsprüfung, zum Beispiel in Form des Z-Buffering-Verfahrens, durchführen. Dong et al. [DKT*07], sowie Dachsbacher et al. [DSD*07], schlagen dagegen Methoden vor, bei denen die Sichtbarkeitsprüfung implizit erfolgen kann. Dong et al. benutzen ebenfalls Kreisflächen als endliche Oberflächenelemente. Zwischen den Kreisflächen werden so genannte Visibility-Links gespeichert, die angeben, ob eine Sichtverbindung besteht. Hier wird ganz analog zum Z-Buffer entschieden, dass ein Patch mit einem kürzeren Link näher liegen muss als ein Patch mit einem längeren Link. Um die Sichtbarkeit zwischen den Patches zu bestimmen, wird die Hemisphäre für jeden Patch in so genannte Bins (Behälter) unterteilt (ein Bin hat einen festen Raumwinkel und kann beispielsweise als Pixel eines Hemicubes verstanden werden). Für jeden Bin wird nun der kürzeste Link bestimmt. Alle Patches liegen hierfür in einer Hierarchie, in der jeweils vier kleine Patches zu einem großen Patch zusammengefasst werden. Um den Link für einen Bin zu bestimmen, wird der Raumwinkel des Bins mit dem des (groben) Patches verglichen. Ist der Raumwinkel des Patches größer als der des Bins, werden aus der Hierarchie die nächstkleineren Patches getestet. Ist bereits ein Link für den Bin gespeichert, wird geprüft, ob dieser kürzer ist als der aktuelle; wenn dies nicht der Fall ist, wird er ersetzt. Man erhält so einen sublinearen Zusammenhang zur Anzahl der Patches für die Erstellung der Links. Die Links selber werden dann für die Beleuchtung verwendet und ersetzen die korrekt bestimmten Formfaktoren. Dachsbacher et al. [DSD*07] zeigen, dass die explizite Sichtbarkeits-

prüfung durch die Annahme einer negativen Leuchtdichte (Antiradiance) entfallen kann, wenn man einen iterativen Ansatz verwendet. Dabei ist die grundlegende Idee, dass Licht, welches von Patches empfangen wird, durch diese Patches auch blockiert wird. Dieses blockierte Licht kann man als negatives Licht verstehen, welches den Patch durchwandert und wiederum auf andere Patches trifft. Summiert man das normale, nicht blockierte und das negative Licht in einem Punkt auf, so muss sich die korrekte Schattierung ergeben. Auch Dachsbacher et al. benötigen entsprechende hierarchische Strukturen, um die Links zwischen den Patches zu bilden. Bei diesen Links ist die Reihenfolge der Patches untereinander zwar bedeutungslos, aber trotzdem müssen die Patches innerhalb eines bestimmten Raumwinkels bestimmt werden. Meyer et al. [MES*09] stellen ein verbessertes Verfahren zur Erstellung der Linkstruktur mit der GPU vor.

Verfahren, die im Bildraum arbeiten (Screen- bzw. Image-Space), werden ebenfalls als Finite-Elements-Verfahren bezeichnet. Bei diesen ist das endliche Element ein Pixel des Framebuffers. Üblicherweise nutzen solche Verfahren Deferred Shading [ST90]. Bei diesem werden die geometrischen und photometrischen Größen einer Szene in verschiedenen 2D-Buffern gespeichert, die die Szene aus der Sicht der Kamera repräsentieren. Diese Buffer werden als Deep-Buffer bzw. G-Buffer bezeichnet und beinhalten die Position, Normale und Reflexionseigenschaften der sichtbaren Oberflächen in diskreter Pixelform.

Ritschel et al. [RGS09] benutzen die G-Buffer in Anlehnung an das Screen-Space Ambient Occlusion [Mit07,BSD08], um für kleine Mesostrukturen indirekte diffuse Reflexionen zu berechnen. Dabei sind die Reflexionen auf kleine Bereiche beschränkt, die effektiv innerhalb der G-Buffer abgetastet werden können. Soler et al. [SHR10] haben später ein Verfahren vorgestellt, das ebenfalls ausschließlich die Daten der G-Buffer verwendet, um indirekte diffuse und leicht glänzende Reflexionen zu realisieren. Allerdings ist ihr Verfahren nicht auf kleine Mesostrukturen beschränkt, sondern betrachtet den kompletten Framebuffer für die indirekte Beleuchtung. Dies wird erreicht, indem von den G-Buffern mehrere Mip-Map-Stufen erzeugt werden. Diese Mip-Maps werden dann durch eine Monte-Carlo-Sampling-Methode innerhalb eines festen Pixelradius abgetastet. Da sich in jeder Mip-Map-Stufe die Auflösung des Buffers halbiert, der Radius aber fixiert ist, ist so die Abtastung des kompletten Buffers möglich. Hier findet wieder das Prinzip Anwendung, dass entfernte Objekte entsprechend gröber repräsentiert werden (in Form von Mip-Maps). Das Verfahren ist sehr schnell und wurde bereits in Computerspielen eingesetzt [SHR10], jedoch haben sowohl das Verfahren von Soler et al. als auch das von Ritschel et al. den Nachteil, dass für die indirekte Beleuchtung nur Oberflächen im Sichtfeld des Betrachters genutzt werden können. Dies erzeugt keine korrekten Ergebnisse, denn indirekte Reflexionen entstehen ebenfalls durch nicht sichtbare Oberflächen.

Nichols et al. [NW09,NSW09,NW10] haben ein Bildraum-Verfahren vorgestellt, bei dem die indirekten Reflexionen nicht auf die sichtbaren Oberflächen beschränkt sind. Durch die Verwendung von Reflective Shadow-Maps (RSM) [DS05], auf die in Kapitel 5.2 noch genauer eingegangen wird, werden indirekte Lichtquellen anhand einer primären Lichtquelle auf die Oberflächen der Szene verteilt. Diese indirekten Lichtquellen werden dann auf die Pixel des G-Buffers angewendet, um so die indirekte Beleuchtung zu erhalten. Da in der Regel sehr viele indirekte Lichtquellen durch eine RSM entstehen, ist es nicht möglich, jeden einzelnen Pixel im G-Buffer separat mit ihnen zu beleuchten. Deshalb werden wieder Mip-Maps von den G-Buffern angelegt. Dabei repräsentieren Pixel der groben Mip-Map-Stufen (die eine geringere Auflösung aufweisen) große, geometrisch homogene

Oberflächen und Pixel feiner Mip-Map-Stufen kleine homogene Oberflächen. Für das Shading wird nun ausgenutzt, dass bei diffusen Reflexionen auf homogenen Oberflächen das Licht niederfrequente Farbverläufe erzeugt. So müssen für homogene Oberflächen nur wenige Pixel aus den groben Mip-Map-Stufen beleuchtet werden. Dann werden sukzessiv die nächstfeineren Stufen beleuchtet und schlussendlich werden die Ergebnisse der einzelnen Stufen ineinander gemischt. Also werden homogene Bereiche im G-Buffer gruppiert. Wegen dieser Gruppierung kann man mit dem Verfahren nur diffuse Reflexionen abbilden, die keine hochfrequenten Signale enthalten. Durch das Clustering im Bildraum kann man darüber hinaus Artefakte bei Kamerabewegungen beobachten (durch die Neugruppierung der Cluster).

Die letzte Form der Szenendiskretisierung, die in diesem Kapitel betrachtet werden soll, ist die Repräsentation der Szene als Voxel-Modell (Volumetric Element). Thiedemann et al. [THG*11] haben ein interaktives globales Beleuchtungsverfahren vorgestellt, das auf Basis von dreidimensionalen Texturen eine Voxel-Struktur aus den Dreiecken der Szene erzeugt. Diese Transformation erfolgt vollständig durch Shader-Programme. Durch die approximative Darstellung der Szene als Voxel-Gitter ist eine einfache und schnelle Traversierung der Struktur möglich. Jedoch sind aufgrund der ungenauen Repräsentation der Oberflächen durch die Voxel keine spiegelnden bzw. glänzenden Reflexionen darstellbar. Komplexe und animierte Szenen sind ebenfalls problematisch, da die Voxel- Struktur sehr viel Speicher benötigt und feine Bewegungen wegen der groben Voxel-Struktur flackernde Artefakte verursachen. Crassin et al. [CNS*11] haben mit ihrem Voxel-Cone-Tracing-Ansatz einen Großteil dieser Probleme beheben können. Große Szenen werden hier durch einen Voxel-Octree in Verbindung mit einem Out-Of-Core-Speicherverfahren [CNL*09,Cra11] repräsentiert. In dem Octree werden nur feine Knoten (Voxel) an Orten erzeugt, an denen sich Geometrie in Form von Dreiecken befindet. Ein Voxel repräsentiert ein Oberflächenelement mit einer Position, Ausrichtung und Farbe. Bereiche, die im Vakuum liegen, benötigen keinen Platz in der Octree-Struktur. Die einzelnen Ebenen des Octrees werden als Mip-Map-Ebenen angelegt, bei denen jede Ebene die darunterliegende Ebene in grober Weise approximiert. Zur Berechnung der indirekten Beleuchtung wird durch eine RSM indirektes Licht in die unterste Ebene des Octrees „injiziert“, welches dann in die gröberen Ebenen übernommen wird. Um die indirekte Reflexion für einen sichtbaren Oberflächenpunkt zu erhalten, wird ein Trichter (Kegel, Cone) um den Punkt gebildet (dessen Spitze auf dem Punkt liegt). Entlang der Hauptachse des Trichters werden nun verschiedene Stichproben des indirekten Lichts aus der Octree-Struktur entnommen. Dabei werden Stichproben, die nahe zum Oberflächenpunkt liegen, von feinen, niedrigen Ebenen des Octrees entnommen, während entfernte Stichproben aus den höheren, groben Ebenen stammen. Hier reichen bereits wenige Abtastpunkte aus, um gute Resultate zu erzielen. Damit sich die diskrete Natur des Octrees nicht negativ auf die Abtastung auswirkt, wird eine quadrilineare Filterung innerhalb der Struktur durchgeführt. Für die Position (x,y,z) der Stichprobe wird linear ein Wert aus den acht nächsten Octree-Knoten interpoliert. Durch die Entfernung der Stichprobe vom Oberflächenpunkt wird die Ebene des Octrees bestimmt. Liegt die Probe zwischen den Ebenen, wird ebenfalls eine lineare Filterung zwischen den beiden betreffenden Ebenen durchgeführt.

Das Voxel-Cone-Tracing kann sowohl diffuse als auch glänzende Reflexionen in ansprechender Form abbilden und erzielt hohe Bildwiederholungsraten. Die elementaren Nachteile sind die hohen Speicheranforderungen und die rechenaufwändige Anpassung des Octrees bei Animationen. Auch Schatten können nur in Form von Ambient Occlusion effizient berechnet werden. Nichtsdestotrotz ist

das Verfahren wichtig, weswegen noch ein direkter Vergleich zum in dieser Arbeit entwickelten LightSkin-Verfahren erfolgt (in Kapitel 6.2.4).

Tabelle 3.5: Bewertungstabelle nach [RDG*12] für interaktive Finite-Elements-Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Coombe et al.	[CHL04]	3	3	0	1	4	1	LD^+E
Bunnell	[Bun05]	4	1	3	1	4	4	$LDDE$
Dong et al.	[DKT*07]	3	2	3	1	3	3	LD^+E
Dachsb. et al.	[DSD*07]	3	3	3	2	2	4	$L(D\|S)^+E$
Meyer et al.	[MES*09]	3	3	4	2	2	4	$L(D\|S)^+E$
Ritschel et al.	[RGS09]	5	1	4	4	4	4	$LDDE$
Nich. & Wym.	[NW10]	4	2	4	4	3	4	$LDDE$
Soler et al.	[SHR10]	3	2	4	5	3	4	$LD(S\|D)E$
Thiede. et al.	[THG*11]	4	3	3	2	3	4	$LDDE$
Crassin et al.	[CNS*11]	5	3	4	3	4	5	$LD^+(D\|S)E$

3.1.6 Instant Radiosity

Das Instant Radiosity wurde erstmals 1997 von Keller [Kel97] vorgestellt. Wie beim klassischen Radiosity eignet sich das Verfahren in besonderer Weise zur Berechnung der Lichtausbreitung in diffus reflektierenden Umgebungen, jedoch ist es im Gegensatz zum klassischen Verfahren deutlich schneller. Das grundlegende Prinzip ist relativ einfach und ähnelt in gewissem Maße dem Photon-Mapping (siehe Abbildung 3.4): Aus Sicht einer primären Lichtquelle werden Strahlen durch die Szene verfolgt. Diese Strahlen werden mit Hilfe eines Quasi-Random-Monte-Carlo-Samplings gebildet. Immer wenn ein Strahl auf eine Oberfläche trifft, wird im Schnittpunkt ein hemisphärisches Punktlicht erzeugt (für das sich durch spätere Publikationen der Begriff des Virtual Point-Lights (VPL) etabliert hat). Für jedes so erzeugte VPL wird eine Shadow-Map berechnet. Hat man ausreichend viele Strahlen mit einer ausreichend hohen Rekursionstiefe in die Szene entsendet, so erhält man eine Menge von VPLs, die die indirekte Beleuchtung repräsentieren. Diese VPLs müssen dann nur noch in einem abschließenden Schritt auf die sichtbaren Oberflächenpunkte der Szene angewendet werden, um die indirekte Beleuchtung zu erhalten. Sowohl die Berechnung der Shadow-Maps, als auch die Anwendung der VPLs auf die sichtbaren Oberflächenpunkte, kann durch die Raster-Hardware moderner Grafikkarten beschleunigt werden. Es handelt sich bei dem klassischen Verfahren von Keller jedoch nicht um ein interaktives Verfahren, da sehr viele VPLs erzeugt werden müssen, um ein hinreichend gutes Ergebnis zu produzieren. Segovia et al. [SIM*06] haben das Verfahren auf ein interaktives Niveau beschleunigt, indem sie einen Importance-Sampling-Ansatz für die Verteilung der VPLs vorschlagen, der den sichtbaren Bereich der Szene berücksichtigt. So erreicht man mit weniger VPLs vergleichbar gute Ergebnisse. Die Verteilung der Lichtquellen basiert auf der Beobachtung, dass nur VPLs zum fertigen Bild beitragen, die im direkten Sichtkontakt zu den vom Betrachter wahrgenommenen Oberflächenpunkten stehen. Man kann also aus Sicht des Betrachters Strahlen bilden und deren erste Reflexion an den sichtbaren Oberflächenpunkten berechnen. Dort, wo die Strahlen nach diesen Reflexionen auftreffen, sollte die VPL-Dichte besonders hoch sein.

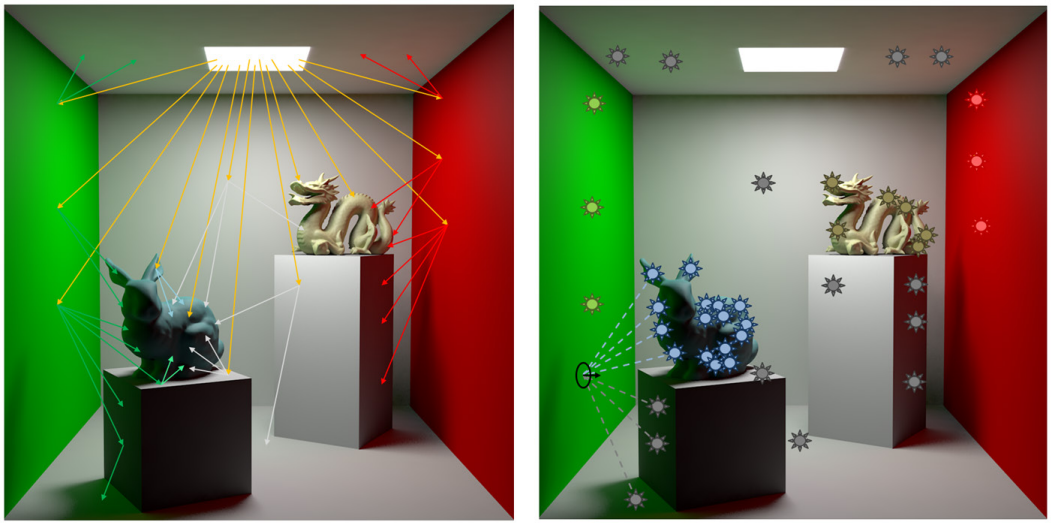

Abbildung 3.4: Das Prinzip des Instant Radiositys. Zuerst werden die virtuellen Lichtquellen durch Strahlenverfolgung in der Szene platziert (links). Danach werden alle sichtbaren Oberflächenpunkte (schwarzer Kreis) mit diesen Lichtquellen beleuchtet (rechts).

Laine et al. [LSK*07] verfolgen mit ihrem Incremental Instant Radiosity einen anderen Ansatz, um interaktive Bildwiederholraten zu gewährleisten: Sie nehmen einmalig zum Startzeitpunkt der Simulation die Verteilung und Anwendung der VPLs auf die Oberflächen vor und aktualisieren diese nur noch inkrementell bei Veränderungen des Lichts von Frame zu Frame. Sofern sich das Licht kontinuierlich verändert, müssen so nur wenige (ein bis zwei Dutzend) VPLs neu verteilt werden, was in Echtzeit möglich ist. Dabei muss jedoch die Szene statisch bleiben, damit keine Artefakte entstehen.

Eine der wichtigsten Techniken zur initialen Verteilung der VPLs aus Sicht einer primären Lichtquelle haben Dachsbacher und Stamminger [DS05] mit ihren Reflective Shadow-Maps (RSM) vorgestellt. Mit diesem Verfahren ist es möglich, sehr effizient und unter Ausnutzung der Grafikkarte die ersten indirekten Lichtquellen in der Szene zu verteilen. Hierbei findet das einfache Prinzip Anwendung, dass jeder Oberflächenpunkt, der von einer direkten Lichtquelle gesehen wird, selbst zu einer indirekten Lichtquelle, also VPL, wird (erste Reflexion). Lichtquellen, die durch weitere Reflexionen entstehen, werden dabei nicht berücksichtigt. Bei der RSM handelt es sich um eine Erweiterung der klassischen Shadow-Map, bei der anstatt eines einfachen Tiefenbuffers aus Sicht der Lichtquelle drei Buffer erzeugt werden. Pro primärer Lichtquelle existieren so ein Pixelbuffer für die Positionen, einer für die Normalen und einer für die diffusen Reflexionseigenschaften der gesehenen Oberflächenpunkte. Ein Pixel aus den drei Buffern bildet dann genau ein VPL ab. Die Anwendung der VPLs kann per Deferred Shading [ST90] direkt auf die Pixel der G-Buffer erfolgen. Bei der Anwendung der VPLs wird jedoch keine Sichtbarkeitsprüfung durchgeführt, so dass keine weichen Schatten durch die indirekte Beleuchtung entstehen können. Ein weiteres Problem ist, dass sehr viele VPLs in einer RSM gespeichert sind. Bereits bei einer RSM-Auflösung von 128x128 Pixeln werden über 16.000 VPLs erzeugt. Wendet man jedes dieser VPLs individuell auf einen Full-HD-G-Buffer an, so führt das zu über 33,9 Milliarden Anwendungen. Dies ist selbst mit modernen Grafikkarten nicht in Echtzeit möglich, weswegen Dachsbacher und Stamminger ein Importance-Sampling vorgeschlagen haben, bei dem ein Kamerapixel in den Raum der RSM projiziert wird. Im Projektionsraum der RSM werden dann vorrangig die VPLs ausgewählt, die nahe zu diesem Empfängerpixel (Punkt) liegen. In einer späteren Publikation schlagen Dachsbacher und Stamminger [DS06] ein Splatting-Verfahren zur

Anwendung der VPLs auf die G-Buffer-Pixel vor. Hier wird jedes VPL mit einer geometrischen Ausbreitung in den G-Buffer gezeichnet (Splatting). Jeder Pixel, der die Geometrie des VPLs schneidet, wird entsprechend schattiert. Da auch hier aus Laufzeitgründen nicht jedes VPL der RSM angewendet werden kann, schlagen die Autoren ein GPU-basiertes Importance-Sampling nach Clarberg et al. [CJA*05] auf Basis der Intensitäten der VPLs vor. Durch die geometrische Einschränkung des Einflusses des VPLs lassen sich mit dem Splatting-Verfahren ebenfalls Kaustiken realisieren.

Bei den Reflective Shadow-Maps wird die Sichtbarkeitsprüfung für die indirekte Reflexion vernachlässigt, dagegen haben Ritschel et al. [RGK*08] mit ihren Imperfect Shadow-Maps (ISMs) ein Konzept vorgestellt, welches eine effiziente, aber grobe Sichtbarkeitsprüfung für die VPLs erlaubt. Damit erzeugen Imperfect Shadow-Maps weiche Schatten für das indirekte Licht. Die Autoren haben erkannt, dass die Berechnung der Shadow-Map für jedes VPL ein signifikanter Engpass bei Instant-Radiosity-Verfahren ist, da die komplette Dreieckstopologie der Szene für jedes VPL gezeichnet werden muss. Also muss bei 1000 VPLs die Szene mindestens 1001 mal gezeichnet werden. Ritschel et al. schlagen deshalb vor, die Szenenrepräsentation zu vereinfachen und nur mit einer vereinfachten losen Punktdarstellung zu arbeiten. Hierfür werden auf den Oberflächen der Szene entsprechende Oberflächenpunkte zufällig verteilt. Für die Erzeugung der Shadow-Map werden dann nur diese Punkte in die Shadow-Map gezeichnet. Das Rastern von Punktprimitiven mit der GPU ist sehr effizient, jedoch sind die Shadow-Maps in diesem Fall nicht vollständig. In der Shadow-Map befinden sich so nur einzelne, lose Tiefenwerte, während der Großteil der Map leer bleibt. Würde man eine solche Map für die Schattierung verwenden, wäre der Schatten perforiert. Um eine vollständige Shadow-Map zu erhalten, wird ein zweidimensionales hierarchisches Pull-Push-Bildverarbeitungsverfahren verwendet [SKE06], um Flächen zwischen den losen Punkten zu bilden. Dieses Bildverarbeitungsverfahren kann für alle ISMs parallel durch die GPU durchgeführt werden, so dass der rechnerische Aufwand sehr gering ist. Die so erzeugten ISMs sind nicht sehr exakt, aber durch die hohe Anzahl fallen die Fehler kaum ins Gewicht. Yu et al. [YCK*09] haben außerdem gezeigt, dass eine approximative Sichtbarkeitsbestimmung bereits ausreicht, um realitätsnahe Bilder zu erzeugen. Probleme bei der Verwendung von ISMs sind die Bildung von Artefakten bei Animationen und dass große Szenen damit nicht effizient dargestellt werden können. Diese Probleme werden von Ritschel et al. [REH*11] in einer späteren Publikation thematisiert, in der sie ein bidirektionales Verfahren vorschlagen, welches den sichtbaren Bereich der Kamera berücksichtigt. Dong et al. [DGR*09] haben ebenfalls ein Verfahren vorgestellt, das die Komplexität der Shadow-Map-Berechnung verringert, indem mehrere VPLs zu wenigen Flächenlichtquellen zusammengefasst werden. Shadow-Maps werden dann nur noch für diese wenigen Flächenlichtquellen berechnet, so dass der Gesamtaufwand zur Berechnung stark reduziert wird. Die Zusammenfassung der VPLs zu einer Flächenlichtquelle basiert hierbei auf den geometrischen Eigenschaften der VPLs (Position und Normale) und wird durch ein k-Means-Verfahren [Mac67] bewerkstelligt.

Schlussendlich kann man zusammenfassen, dass Instant Radiosity eines der überzeugenderen Verfahren für die interaktive Beleuchtung ist, denn es nutzt effizient die Grafik-Hardware, benötigt keine Vorberechnungen oder komplexe Strukturen und erzeugt keinen Extraaufwand für Animationen. Jedoch muss festgehalten werden, dass alle hier vorgestellten Verfahren bei Animationen zur Artefaktbildung neigen. Das in dieser Arbeit entwickelte Verfahren basiert ebenfalls auf dem Instant-Radiosity-Ansatz, erzeugt jedoch deutlich weniger Artefakte. Ein weiteres Problem, das durch die

Verwendung von Punktlichtern entsteht, ist die nicht begrenzte Energie, sobald ein Empfängerpunkt sehr nahe zur Lichtquelle steht (vgl. Gleichung (2.8)). Dieses Problem wird häufig dadurch behoben, dass der Geometrieterm aus Gleichung (2.8) auf ein Maximum beschränkt wird, was wiederum dazu führt, dass das fertige Bild energetische Fehler aufweist (dunkle Ränder in den Ecken). Novak et al. [NED11] haben diesbezüglich ein Verfahren vorgestellt, welches dieses Phänomen im Bildraum reduziert.

Tabelle 3.6: Bewertungstabelle nach [RDG*12] für interaktive Instant-Radiosity –Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Segovia et al.	[SIM*06]	3	3	3	3	3	3	$L(D\|S)^+DE$
Laine et al.	[LSK*07]	4	3	2	2	3	4	LD^+E
Dach. & Stam.	[DS05]	4	1	5	3	4	4	$LDDE$
Dach. & Stam.	[DS06]	5	2	5	3	4	4	$L(D\|S)DE$
Ritschel et al.	[RGK*08]	4	2	4	3	3	4	$LDDE$
Ritschel et al.	[REH*11]	4	3	4	5	1	3	$LDDE$
Dong et al.	[DGR*09]	5	1	3	2	2	4	$LDDE$
Novak et al.	[NED11]	4	3	4	3	2	4	$LD^+(D\|S)E$

3.1.7 Many-Light- und Point-Based-Ansätze

Bei Many-Light- und Point-Based-Beleuchtungsverfahren handelt es sich eigentlich um zwei verschiedene Ansätze. Jedoch haben beide Verfahren essentielle Gemeinsamkeiten, weshalb es sinnvoll ist, beide Verfahren in einem Unterkapitel zu behandeln. Beide Verfahren sind Spezialisierungen des Instant-Radiosity-Ansatzes (siehe vorheriges Unterkapitel), bei denen die indirekte Beleuchtung über eine endliche Menge von virtuellen Lichtquellen simuliert wird.

Bei Many-Light-Ansätzen ist die initiale Erzeugung der virtuellen Lichtquellen nebensächlich, stattdessen wird davon ausgegangen, dass eine große Menge dieser Lichtquellen bereits in der Szene verteilt wurde (Hundertausende bis mehrere Millionen). Die Verfahren konzentrieren sich auf die effiziente Anwendung der Lichtquellen auf die Oberflächenpunkte einer Szene. Dies geschieht in der Regel mit einem sublinearen Berechnungsaufwand zur Anzahl der Lichtquellen. Ein gängiges Prinzip ist es, die Lichter in Octrees zu speichern [WFA*05]. Ein Blatt-Knoten des Octrees repräsentiert ein konkretes virtuelles Licht und die inneren Knoten fassen mehrere Lichter zusammen, indem sie deren Strahlungsverhalten approximieren (siehe Abbildung 3.5). Dabei können die inneren Knoten selbst primitive virtuelle Lichter sein (z. B. VPLs) oder sie bilden die Lichtausbreitung auf komplexere Weise ab (z. B. durch Spherical-Harmonic-Koeffizenten). Um mit diesen Verfahren die Reflexion für einen bestimmten Oberflächenpunkt zu berechnen, wird ein so genannter Light-Cut in der Octree-Struktur bestimmt [WFA*05]. Ein solcher Cut zieht sich durch verschiedene Ebenen des Octrees: Es gilt, dass Lichtquellen, die weit von einem Oberflächenpunkt entfernt liegen, durch hohe innere Knoten des Octrees abgebildet werden können (welche näher an der Wurzel des Baums liegen), während nahe Lichtquellen besser durch tiefe Knoten bzw. Blätter abgebildet werden können. So erhält jeder Oberflächenpunkt auf Basis seiner geometrischen Beschaffenheit einen individuellen Light-Cut im Octree. Bei diesen Verfahren wird für leicht glänzende und diffus reflektierende Oberflächen ausgenutzt, dass ab einer gewissen Entfernung hochfrequente Veränderungen im Licht für die Reflexion keine Bedeutung mehr haben. Sie können also durch innere Knoten zusammengefasst bzw. geglättet

werden. Statt Octrees können aber auch beliebige Bounding-Volume-Hierarchies (BVHs) genutzt werden. Many-Light-Ansätze werden häufig auch als Light-Clustering-Ansätze bezeichnet.

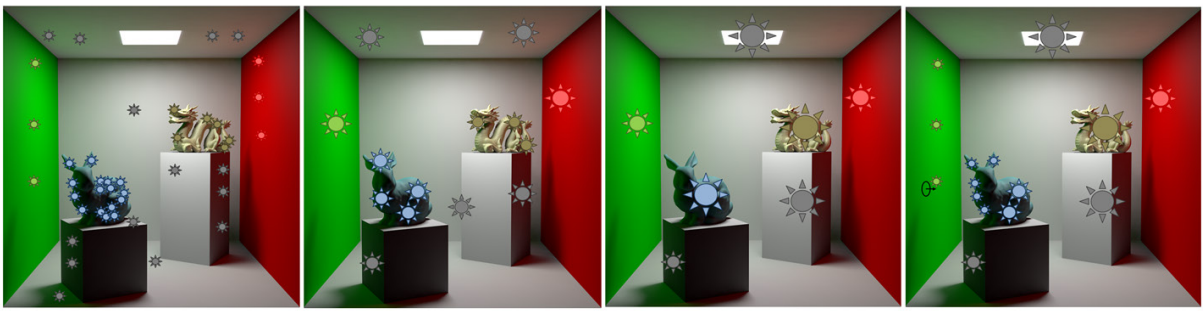

Abbildung 3.5: Many-Light-Ansatz: Die ersten drei Bilder zeigen die Clustering-Stufen der VPLs. Im rechten Bild sieht man, wie die verschiedenen Stufen auf einen Oberflächenpunkt (schwarzer Kreis) angewendet werden. Lichtquellen, die nahe zum Oberflächenpunkt liegen, werden fein abgebildet, während ferne Lichtquellen durch grobe zusammengefasste Lichtquellen repräsentiert werden.

Hasan et al. [HPB07] haben 2007 einen Many-Light-Ansatz veröffentlicht, der die Beleuchtung der Oberflächenpunkte durch die Lichtquellen in Form einer Matrix darstellt. Dabei sind die Zeilen der Matrix Oberflächenpunkte und die Spalten virtuelle Lichtquellen. Ein einzelnes Element der Matrix stellt also einen Punkt dar, der durch eine Lichtquelle beleuchtet wird. Die komplette Beleuchtung auf einen Punkt kann man durch die Summation der Elemente einer Zeile der Matrix berechnen. Aus Effizienzgründen ist es jedoch nicht sinnvoll, jede Zeile der Matrix separat zu berechnen, stattdessen werden wenige Zeilen der Matrix stochastisch ausgewählt. Für eine solche zufällig ausgewählte Zeile werden die einwirkenden Lichtquellen zu Clustern zusammengefasst. Aus diesen wenigen Clustern werden dann repräsentative Lichtquellen ausgesucht und energetisch so angepasst, dass sie dem kompletten Cluster entsprechen. Da die Lichtquellen Spalten in der Matrix sind, hat man so eine Untermenge von Spalten bzw. Lichtquellen, die die Gesamtbeleuchtung gut approximieren. Für diese wenigen Lichtquellen werden als nächstes Shadow-Maps erzeugt, die auf die benachbarten Zeilen (Oberflächenpunkte) der Matrix angewendet werden können. Um gute Ergebnisse mit dem Verfahren zu erzielen, ist die zufällige Auswahl der Zeilen von besonders großer Bedeutung. Da es sich um ein stochastisches Verfahren handelt, sind temporäre Inkohärenzen in konsekutiven Bildern zu erkennen.

Ist der Ursprung der virtuellen Lichtquellen auf eine zweidimensionale Projektion zurückzuführen, wie dies bei Reflective Shadow-Maps beispielsweise der Fall ist, so ist es nicht zwingend nötig, diese in einer dreidimensionalen BVH zu speichern. Stattdessen lassen sich die Lichtquellen direkt im Bildraum der Projektion bearbeiten bzw. zusammenfassen. Dies hat den Vorteil, dass einfache Bildverarbeitungsverfahren für das Clustering genutzt werden können. Ki und Oh [KO08] haben ein solches Light-Clustering für die GPU vorgeschlagen, bei dem ähnliche Lichtquellen in Bildpyramiden bzw. Mip-Maps zusammengefasst werden. Die Ähnlichkeit einer Lichtquelle zu einer anderen ergibt sich aus den Ausbreitungseigenschaften und den Positionen. Es werden immer vier Lichtquellen einer niedrigeren Mip-Map-Ebene in einer Lichtquelle der nächsthöheren Ebene zusammengefasst. Da diese Zusammenfassung nicht immer sinnvoll möglich ist (wegen der Eigenschaften der Lichtquellen), wird ein entsprechendes Flag gesetzt, das die zusammengefasste Lichtquelle als gültig bzw. nicht gültig kennzeichnet. Für die Beleuchtung werden dann die Lichtquellen von der obersten zur untersten Ebene angewendet. Ist eine Lichtquelle auf einer Ebene ungültig, werden die Lichtquellen

der Ebene darunter genutzt und so weiter. Prutkin et al. [PKD12] haben ebenfalls ein zweidimensionales Light-Clustering-Verfahren vorgestellt, welches vollständig auf der GPU implementierbar ist. Anstatt Bildpyramiden werden hier k-Means-Cluster berechnet, deren Zentren die neuen zusammengefassten Lichtquellen bilden. Da beim k-Means-Clustering k Saatpunkte im Vorhinein bestimmt werden müssen, schlagen Prutkin et al. die Verwendung eines Bidirectional Importance-Samplings [SIM*06] vor, um geeignete Positionen für die Saatpunkte zu bestimmen. Um temporäre Inkohärenzen zu vermeiden, werden die Saatpunkte von Frame zu Frame übernommen und entsprechend der Lichtquellenverteilung angepasst.

Bei Point-Based-Ansätzen [Chr08] wird ebenfalls mit BVHs gearbeitet, jedoch werden in diesen nicht nur Lichtquellen gespeichert, sondern auch Oberflächenpunkte, die kein Licht abstrahlen. Hier muss die komplette Szene in Form einer „Punktwolke“ vorhanden sein. Der Begriff Punktwolke ist hier etwas irreführend, denn im eigentlichen Sinn handelt es sich eher um eine „Surfelwolke“ (Surface-Elements). Dabei bezeichnet ein Surfel ein Oberflächenelement mit einer Normalen und einer kleinen Fläche. Üblicherweise spricht man aber trotzdem von Punkten. Um die Reflexion einer Oberfläche der Szene zu bestimmen, wird ein kleiner Projektions-Buffer verwendet, den Ritschel et al. [REG*09] als Microbuffer bezeichnen. Dieser Microbuffer spiegelt in diskretisierter Form das sichtbare, eingehende Licht in einem Oberflächenpunkt wider. Die Point-Based-Beleuchtungsverfahren konzentrieren sich im Wesentlichen darauf, diesen Microbuffer möglichst effizient zu füllen. Um dies zu erreichen, wird das gleiche Prinzip wie bereits bei den Many-Light-Ansätzen genutzt, nämlich dass alle Surfels in einer BVH zusammengefasst werden. Welcher BVH-Knoten dann zur Darstellung verwendet wird, hängt von dessen Abdeckung eines Pixels im Microbuffer ab. Deckt ein Knoten nur die Hälfte eines Pixels ab, wird ein höherer Knoten verwendet. Um die Reflexion eines Punktes anhand seines Microbuffers zu bestimmen, wird dieser noch mit der BRDF des Punktes verrechnet, so dass ein Pixel des Microbuffers den Term $f_r(x, \omega, \omega_i)L_i(x, \omega_i)$ aus der Rendergleichung (2.3) darstellt. Das Integral kann dann durch die Summe der Microbuffer-Pixel approximiert werden, die durch einfache GPU-Reduktionsalgorithmen berechnet werden kann. Ritschel et al. [REG*09] bilden die Gewichtung mit der BRDF in besonders effizienter Weise ab, indem sie den Microbuffer entsprechend der BRDF verzerren. So erhalten Oberflächen (Surfels) mehr Pixel im Microbuffer, wenn sie einen großen Anteil zur Reflexion beitragen. Da potentiell jeder Pixel des fertig synthetisierten Bildes einen Microbuffer benötigt, ist das effiziente Zeichnen von diesem von besonderer Bedeutung. Holländer et al. [HRE*11] beschreiben hierfür mit ihrem ManyLoDs-Ansatz ein Verfahren, um mit der GPU BVHs zu erzeugen, die effizient von mehreren Betrachtungspunkten gezeichnet werden können. Dabei erlaubt die vorgestellte Struktur eine inkrementelle Aktualisierung bei Animationen, so dass bei moderaten Animationen nicht die komplette BVH rekonstruiert werden muss. Maletz und Wang [MW11] stellen einen alternativen Ansatz vor, bei dem die Szene ebenfalls über eine Surfel-BVH repräsentiert wird. Dabei vereinfachen sie die Multiplikation mit der BRDF, indem sie die Surfels entsprechend größer skalieren, wenn sie einen hohen Anteil zur Reflexion beitragen. Um nicht für jeden Pixel des Präsentations-Buffers einen Microbuffer zeichnen zu müssen, verwenden sie außerdem eine bilaterale Filterung im Bildraum. Im Vergleich zu den Verfahren von Ritschel et al. und Holländer et al. ist das Verfahren von Malitz und Wang jedoch rechenintensiver.

Tabelle 3.7: Bewertungstabelle nach [RDG*12] für interaktive Many-Light- und Point-Based-Verfahren.

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Hasan et al.	[HPB07]	1	4	2	3	2	3	$L(D\|S)DE$
Ki & Oh	[KO08]	4	1	4	2	4	5	$L(DD\|V)E$
Prutkin et al.	[PKD12]	4	1	3	3	3	4	$LDDE$
Cristensen	[Chr08]	1	4	3	4	2	0	$LD(D\|S)E$
Ritschel et al.	[REG*09]	3	4	2	3	2	3	$LD(D\|S)E$
Hollän. et al.	[HRE*11]	4	3	2	3	1	4	$LDDE$
Maletz&Wang	[MW11]	2	2	2	3	1	5	$LD(D\|S)E$

3.2 Globale Beleuchtung in Augmented-Reality-Anwendungen

In diesem Kapitel werden aktuelle globale Beleuchtungsansätze im Kontext von Augmented-Reality-Anwendungen betrachtet. Prinzipiell kann jedes der in Kapitel 3.1 vorgestellten Beleuchtungsverfahren in einer AR-Anwendung genutzt werden, jedoch müssen hierfür die Prinzipien des Differential Renderings [FGR92,DM97] aus Kapitel 2.2.2 zur Anwendung kommen. Das Differential Rendering setzt voraus, dass die reale Szene als virtuelle Rekonstruktion vorhanden ist. Ansonsten würden Oberflächen aus der realen Szene nicht berücksichtigt. Aus diesem Grund gehen die hier vorgestellten Ansätze davon aus, dass eine virtuelle Rekonstruktion der realen Szene vorhanden ist. So beschränkt sich die Echtzeitfähigkeit bzw. Interaktivität auf die Veränderung von virtuellen Objekten, sowie auf die Veränderung der Lichtverhältnisse (reale und virtuelle). Veränderungen von realen Oberflächen werden dagegen nicht berücksichtigt, weshalb nur statische reale Szenen simuliert werden können. Die nachfolgenden Ansätze konzentrieren sich deshalb auf die Synthetisierung der realen Lichtverhältnisse (Photometrische Registrierung), sowie auf die Weiterentwicklung interaktiver Beleuchtungsverfahren für das Anwendungsgebiet Augmented Reality.

In diesem Kapitel werden nur aktuelle Ansätze diskutiert, die in der Lage sind, dynamische globale Beleuchtungseffekte, wie indirekte Reflexionen, in Echtzeit zu simulieren. Verfahren, die sich dagegen nur auf die Erzeugung von Schatten konzentrieren, werden in dieser Dissertation nicht betrachtet.

Grosch et al. [GEM07] haben bereits 2007 ein echtzeitfähiges Verfahren zur Berechnung von direkten und indirekten Reflexionen für augmentierte Objekte innerhalb eines Raumes vorgestellt. Für die Simulation wird davon ausgegangen, dass der Raum direktem Tageslicht ausgesetzt wird, das durch Fenster in den Raum gelangt (siehe Abbildung 3.6, links). Virtuelle Objekte können somit direkt durch die Sonne oder indirekt durch reflektierende Oberflächen innerhalb des Raumes beleuchtet werden. Für beide Arten der Beleuchtung schlagen Grosch et al. verschiedene Verfahren zur Berechnung der Reflexionen vor. Zuerst soll die Berechnung des direkten Lichts betrachtet werden. Dieses wird durch eine HDR-Weitwinkelkamera außerhalb des Raumes gemessen, wodurch die Erzeugung einer hemisphärischen Environment-Map ermöglicht wird. Objekte im Raum, die direktes Tageslicht empfangen, werden durch einen Shadow-Mapping-Ansatz direkt beleuchtet. Dieser setzt voraus, dass die Position und die Intensität der primären Lichtquellen bekannt sind. Da das direkte Licht durch eine Environment-Map repräsentiert wird, können die Positionen und Intensitäten der primären Lichtquellen durch ein entsprechendes Importance-Sampling von dieser Map gewonnen werden. Beim Importance-Sampling werden in den Bereichen mehr Stichproben (primäre Lichtquellen) generiert, in denen die Intensität des Lichts besonders hoch ist. Hierbei muss berücksichtigt werden, dass die

Environment-Map das Licht außerhalb des Raumes repräsentiert. Das Objekt befindet sich aber innerhalb des Raumes, weshalb es nur Licht durch ein Fenster im Raum erhalten kann. Für das Importance-Sampling müssen also nur die Pixel der Environment-Map betrachtet werden, die durch das Fenster vom virtuellen Objekt aus zu sehen sind. Hier schlagen Grosch et al. ein einfaches geometrisches Verfahren vor, bei dem vier Ebenen anhand der Fensterkanten konstruiert werden, die ebenfalls Tangentialebenen der Bounding-Sphere des virtuellen Objekts sind. Für jeden unendlich entfernten Pixel der Environment-Map wird dann geprüft, ob dieser innerhalb oder außerhalb dieser vier Ebenen liegt. Liegt der Pixel außerhalb, so wird er schwarz ausmaskiert und erhält demnach für das Importance-Sampling eine Gewichtung von null.

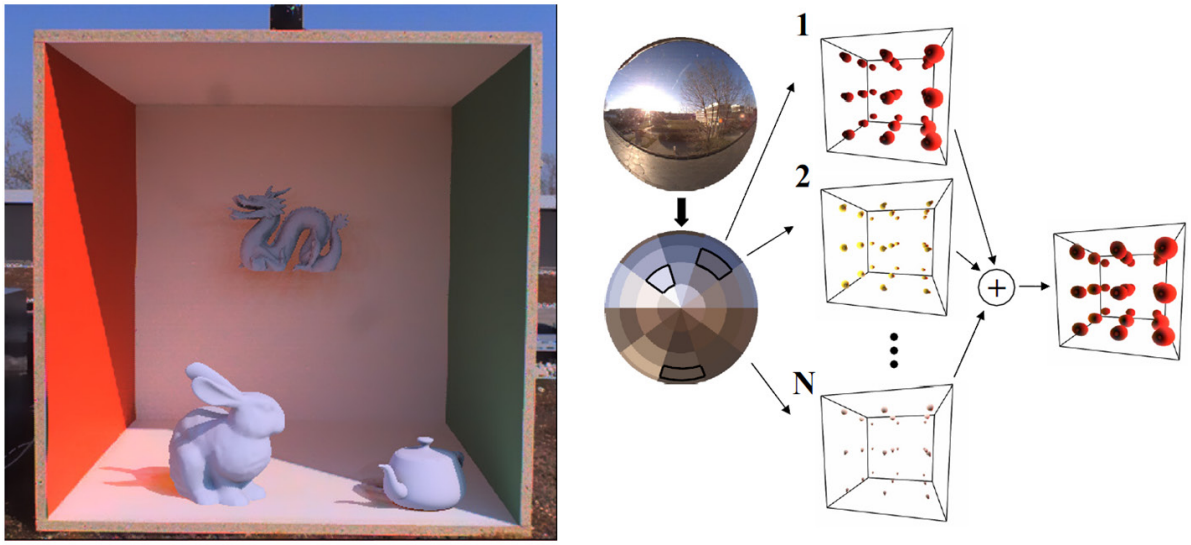

Abbildung 3.6: Globale Beleuchtung nach Grosch et al. [GEM07]. Die Abbildung links zeigt virtuelle Objekte in einem realen Raum (Box). Die virtuelle Beleuchtung wird durch zwei verschiedene Verfahren realisiert: Shadow-Mapping für die direkte Beleuchtung und Irradiance-Volumes für die indirekte Beleuchtung. Um die Beleuchtungsstärke für die Knoten des 3D-Gitters des Irradiance-Volumes zu berechnen, wird die Environment-Map in Flächenlichtquellen unterteilt, für die dann eine Radiosity-Simulation durchgeführt wird (rechts). Die Abbildung wurde aus [GEM07] entnommen.

Zur Simulation der indirekten Reflexionen setzen Grosch et al. Irradiance-Volumes [GSH*98] ein. Bei diesem Ansatz wird das indirekte Licht durch Abtastpunkte innerhalb eines diskreten dreidimensionalen Gitters repräsentiert, ähnlich wie dies bei den Discrete Ordinate Methods gemacht wird (siehe Kapitel 3.1.4). Für jeden Abtastpunkt des Gitters (Knoten) wird die einfallende indirekte Beleuchtungsstärke gespeichert. Um die indirekte Beleuchtungsstärke zu berechnen, wird ein Radiosity-Ansatz genutzt, bei dem die Environment-Map in mehrere homogene Flächenlichtquellen unterteilt wird (typischerweise reichen 64 solcher Flächenlichtquellen aus). Für jede Flächenlichtquelle wird eine eigene Radiosity-Simulation durchgeführt (siehe Abbildung 3.6, rechts). Für die Simulation wird vorausgesetzt, dass die Geometrie, sowie die Reflexionseigenschaften der Materialien des Raumes, im Voraus bekannt sind. Wenn die indirekte Beleuchtung für jeden Knoten des 3D-Gitters berechnet wurde, wird diese in neun Spherical-Harmonics-Koeffizienten abgebildet, um die Laufzeit der Simulation zu verbessern und den Speicherbedarf zu reduzieren. Durch den Einsatz des Radiosity-Verfahrens, sowie durch die Verwendung von nur neun SH-Koeffizienten, kann das Verfahren nur diffuse indirekte Reflexionen ohne weiche Schatten abbilden, diese jedoch mit Bildwiederholraten von 10-64 Bildern pro Sekunde.

Franke [Fra13a,Fra13b] hat kürzlich ein Verfahren vorgestellt, das ebenfalls in der Lage ist, indirekte diffuse Reflexionen durch eine diskrete Abbildung der Beleuchtung innerhalb eines 3D-Gitters zu berechnen (siehe Abbildung 3.7). Im Gegensatz zum Ansatz von Grosch et al. ist es mit diesem Ansatz möglich, weiche Schatten abzubilden. Dabei wird das Prinzip der Light-Propagation-Volumes, die bereits in Kapitel 3.1.4 vorgestellt wurden, genutzt, um die Verbreitung des indirekten Lichts zu berechnen. Die Ausbreitung des Lichts wird hier innerhalb des Gitters von Nachbarknoten zu Nachbarknoten fortberechnet. Da die Knoten das indirekte Licht durch eine geringe Anzahl von SH-Koeffizienten repräsentieren, wird die Lichtausbreitung mit zunehmender Verteilung im Volumen „ausgewaschen“.

Reales Licht kann durch zwei Arten in das Gitter übernommen werden: Entweder sind die Position und die Intensität der Lichtquelle durch die Verwendung von AR-Markern bekannt, oder das Umgebungslicht wird durch eine Weitwinkelkamera aufgenommen. Bei Letzterem werden die Lichtquellen aus dem Bild der Kamera extrahiert, indem ein Median-Cut-Algorithmus [Deb08a] genutzt wird, der es erlaubt, die sphärischen Koordinaten potentieller Lichtquellen zu bestimmen. Sind die sphärischen Koordinaten bestimmt worden, können die dazugehörigen Lichtquellen auf einer Kugel um das Objekt angeordnet und in das Gitter injiziert werden.

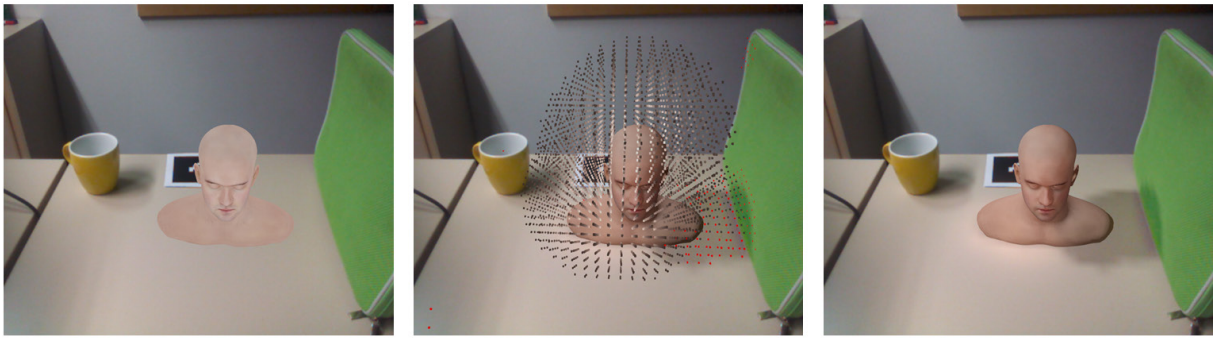

Abbildung 3.7: AR-Beleuchtungssimulation nach Franke [Fra13a]. (Links) Szene ohne globale Beleuchtung, (Mitte) Light-Propagation-Volume um das virtuelle Objekt, (rechts) Szene mit virtueller globaler Beleuchtung. Die Bilder wurden aus [Fra13a] entnommen.

Üblicherweise erzeugt die Verwendung einer fixen volumetrischen Anordnung von Abtastpunkten innerhalb eines Gitters temporäre Artefakte bei Objektanimationen, wenn die Oberflächenpunkte durch das Gitter bewegt bzw. rotiert werden. Franke umgeht dieses Problem, indem er das Gitter gleichförmig mit den Objekten rotiert und verschiebt. Hierdurch werden die temporären Artefakte deutlich verringert, es hat aber auch zur Folge, dass jedes Objekt sein eigenes Propagation-Volume benötigt. Da es sich um kubische Strukturen handelt, sind die Speicheranforderungen bei mehreren Objekten entsprechend hoch und auch die Laufzeit der Simulation erhöht sich deutlich.

Franke liefert in seiner Publikation [Fra13a] einen qualitativen Vergleich zwischen seinem Verfahren und dem Verfahren, das unter anderem für diese Dissertation entwickelt wurde (siehe Abbildung 3.8). In dieser Abbildung ist zu erkennen, dass das Verfahren von Franke weiche Schattierungen erzeugt, die das Verfahren dieser Arbeit für AR-Anwendungen nicht unterstützt. Man erkennt aber auch in der Abbildung, dass die indirekten Reflexionen beim Ansatz von Franke einen starken Energieabfall mit zunehmender Entfernung aufzeigen. Dieser (zu) starke Energieabfall liegt in der gewählten Methode zur Lichtausbreitung begründet. Prinzipiell ist es bei den Light-Propagation-

Volumes nicht sinnvoll, Licht weit in die Szene zu transportieren, da dies entsprechend viele Iterationen im Volume zur Folge hat, die zu einer höheren Laufzeit der Simulation führen (siehe Kapitel 3.1.4). Möchte man das Licht mit wenigen Iterationen dennoch weit in der Szene verteilen, muss man die Auflösung der Gitterstruktur verringern. Hierdurch wird pro Iteration eine größere Entfernung überbrückt. Dies hat aber wiederum zur Folge, dass das indirekte Licht nur sehr grob und unpräzise abgebildet wird.

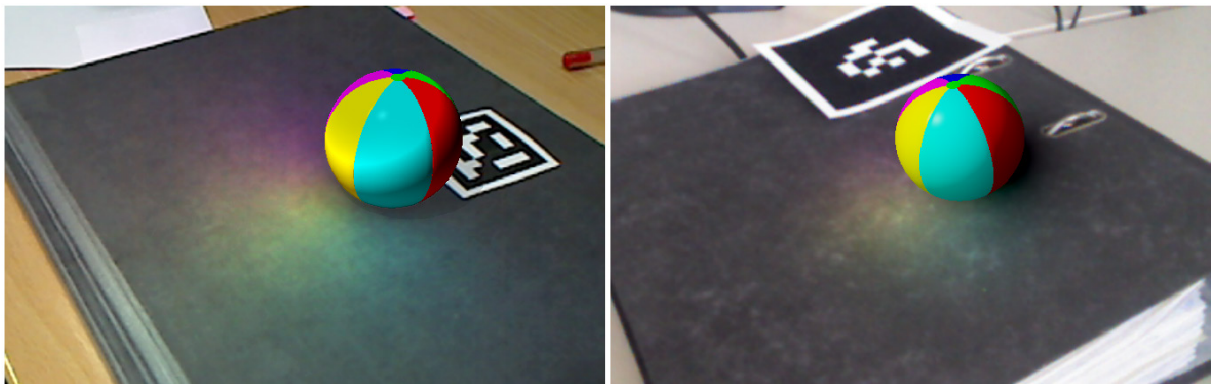

Abbildung 3.8: Vergleich zwischen dem Ansatz von Franke (rechts) und dem Ansatz dieser Arbeit (links). Während der Ansatz von Franke weiche indirekte Schatten für AR-Objekte erzeugt, wird die Ausbreitung des indirekten Lichts stark eingeschränkt. Die Bilder wurden aus [Fra13a] entnommen.

Ein anderes Verfahren zur Ausbreitung des indirekten Lichts in einer AR-Anwendung schlagen Knecht et al. [KTM*10] mit ihrem Differential Instant Radiosity vor. Mit diesem Ansatz können ebenfalls indirekte diffuse Reflexionen und weiche Schatten in Echtzeit berechnet werden. Die Ausbreitung des indirekten Lichts wird hier durch das Imperfect Shadow-Mapping (ISM) von Ritschel et al. [RGK*08] simuliert und mit dem Differential Rendering kombiniert. Es handelt sich also um eine Instant-Radiosity-Methode, bei der, ausgehend von den primären Lichtquellen, virtuelle Lichtquellen in der Szene verteilt werden. Jedes virtuelle Licht erzeugt eine grobe Shadow-Map (die ISM). Da mehrere hundert solcher virtuellen Lichtquellen in der Szene verteilt werden, muss die Berechnung der Shadow-Maps entsprechend beschleunigt werden. Deshalb wird für die Berechnung der ISMs die Szene durch Punkte bzw. Sprites repräsentiert, da diese besonders schnell gerastert werden können. Bei dieser Art der Darstellung kommt es aber häufig zu Approximationsfehlern, da die Sprites immer zum Betrachtungspunkt ausgerichtet werden. Knecht et al. schlagen vor, für die Berechnung der ISMs die Sprites anhand der Normalen der Oberflächen, aus denen die Sprites resultieren, auszurichten. Damit lassen sich Fehler bei der Approximation von steilen Oberflächen reduzieren und die Qualität der ISMs verbessern.

Es können nur relativ wenige ISMs in Echtzeit berechnet werden (mehrere hundert bis tausend), weshalb es zu einer Unterrepräsentation des indirekten Lichts kommt. Die Verteilung der ISM-erzeugenden VPLs in der Szene verändert sich deutlich, wenn Modelle oder das primäre Licht animiert werden. Um dieses Problem der temporären Inkohärenz zu lösen, schlagen Knecht et al. vor, die Ergebnisse der indirekten Beleuchtung über zwei konsekutive Bilder zu mischen (allerdings werden durch die rekursive Anwendung letztendlich nicht nur zwei, sondern mehrere Bilder ineinander gemischt, siehe folgende Formel). Hierdurch sollen störende flackernde Artefakte reduziert werden. Die indirekte Beleuchtung für einen Pixel eines Frames ergibt sich durch die lineare Interpolation von einem alten I_{old} und einem neuen I_{new} Frame:

$$I_{new} = cI_{old} + (1 - c)I_{new}. \quad (3.5)$$

In dieser Gleichung bezeichnet *c* den *Confidence*-Faktor, der sich aus drei Veränderungen im Bild herleitet: der Veränderung der Position, der Veränderung der Normalen und der Veränderung der indirekten Beleuchtung für einen Pixel. Je größer eine Veränderung wird, desto kleiner wird der *Confidence*-Faktor und desto weniger wird von dem alten Bild in das neue übernommen. Diese Art der Interpolation über konsekutive Frames hat den elementaren Nachteil, dass das Licht der Animation „nachläuft". Wird eine kontinuierliche Animation abrupt beendet, kann man beobachten, wie das Licht verspätet auf die Veränderung reagiert. Außerdem kann es zu Geisterbildern bei der indirekten Beleuchtung kommen. Es gilt also einen Kompromiss zu wählen, zwischen flackernden Artefakten und Licht, das nachläuft.

Wie bei allen anderen Ansätzen wird bei Knecht et al. ebenfalls davon ausgegangen, dass die reale Szene im Voraus manuell erzeugt wurde. Später haben Knecht et al. in [KTM*12] die Verwendung eines RGB-D-Sensors zur Rekonstruktion der realen Szene vorgeschlagen. Die temporären Artefakte, die ein solcher Sensor verursacht, werden bei diesem Ansatz wieder durch die Glättung über konsekutive Frames verringert, was jedoch zu den gleichen Problemen wie bei der indirekten Beleuchtung führt. Kürzlich haben Knecht et al. [KTW*13] eine Methode vorgestellt, die ebenfalls in der Lage ist, spiegelnde Reflexionen und Refraktionen abzubilden. Hierfür wird eine Kombination von verschiedenen Ansätzen vorgeschlagen.

Schlussendlich soll noch der Differential-Irradiance-Caching-Ansatz von Kán und Kaufmann [KK13] betrachtet werden, mit dem es ebenfalls möglich ist, den globalen Lichtaustausch zwischen realen und virtuellen Objekten mit interaktiven Bildwiederholraten zu simulieren. Der Ansatz unterstützt die Simulation von diffusen indirekten Reflexionen mit weichen Schatten, sowie die Darstellung von transparenten Objekten, die das Licht ideal brechen. Hierfür nutzen Kán und Kaufmann die Implementierung des GPU-Raytracers OptiX [PBD*10]. Da jedoch die Strahlenverfolgung selbst durch massive Parallelisierung durch die Grafikkarte nicht echtzeitfähig ist, kombinieren die beiden Autoren das Ray-Tracing-Verfahren mit einem Irradiance-Caching-Ansatz. Bei diesem Ansatz wird nur für eine lose Menge von Bildpunkten (Caches) eine Strahlenverfolgung durchgeführt. Wenn für die lose Menge an Bildpunkten die indirekten Beleuchtungsstärken bekannt sind, kann ein Interpolationsverfahren genutzt werden, das für den kompletten Bildraum die indirekten Reflexionen berechnet. Da das Irradiance-Caching in Kapitel 5.3.3 noch genauer erklärt wird, soll hier von einer detaillierten Beschreibung abgesehen werden. Prinzipiell erzeugt der Ansatz von Kán und Kaufmann annährend physikalisch korrekte Ergebnisse, jedoch ist die Ausführungsgeschwindigkeit des Verfahrens nur für simple virtuelle Szenen akzeptabel. Darüber hinaus verursacht der Ray-Tracing-Ansatz zusammen mit der losen Verteilung der Irradiance-Caches temporäre Artefakte bei Animationen (siehe Kapitel 3.1.1 und 4.4.2).

Zusammenfassend lässt sich festhalten, dass alle interaktiven globalen Beleuchtungsansätze für AR-Anwendungen spezifische Vor- und Nachteile aufweisen. Ein systematisches Problem, das für alle in diesem Unterkapitel vorgestellten Ansätze vorhanden ist, wird in dieser Arbeit aber im besonderen Maße adressiert: die Unterstützung von dynamischen realen Szenen und deren Einbettung in das globale Beleuchtungsverfahren in Echtzeit. Diese Unterstützung lässt eine volle Dynamik für virtuelle und reale Szenen zu.

4 Konzeption des LightSkin-Verfahrens

4.1 Anforderungen

Das in dieser Arbeit entwickelte Verfahren ist im Anwendungsgebiet Virtual und Augmented Reality angesiedelt. In Kapitel 1.2 zur Motivation wurde erwähnt, dass kein globales Beleuchtungsverfahren bekannt ist, das die folgenden Anforderungen in Gänze erfüllt:

1. Dynamik und Interaktivität,
2. Komplexität,
3. Darstellungsrate und Auflösung und
4. Plausibilität.

Diese vier Anforderungen werden im nächsten Unterkapitel als Filter verwendet, um existierende Verfahren hervorzuheben, die nach dem Kategoriensystem von Ritschel et al. [RDG*12] (siehe Kapitel 2.1) diese Anforderungen zumindest ansatzweise erfüllen. Die Dynamik und Interaktivität sowie die Darstellungsrate lassen sich dabei direkt auf die Kategorien Dynamik und Geschwindigkeit aus Kapitel 2.1 übertragen. Es wird gezeigt, dass nur sehr wenige Verfahren für das Anwendungsgebiet in Frage kommen, die jedoch allesamt spezifische Nachteile mit sich bringen. Nach der Feststellung, dass kein geeignetes „fertiges“ Verfahren existiert, werden in Kapitel 4.3 Grundtechnologien auf ihre Eignung für ein voll dynamisches Verfahren abgewogen. Es wird sich zeigen, dass der Instant-Radiosity-Ansatz eine geeignete Grundlage bildet. Um jedoch mit diesem Ansatz schnelle Bildwiederholraten zu erzeugen, müssen laufzeitoptimierende Methoden eingesetzt werden, die in bestehenden Ansätzen dazu führen, dass Artefakte bei Animationen auftreten. Deshalb wird in den Unterkapiteln 4.4.1 und 4.4.2 auf diese Optimierungsmethoden eingegangen und ein eigener Optimierungsansatz formuliert. Anschließend wird beschrieben, welche Anforderungen für die Plausibilität bzw. Darstellungsqualität gelten sollen und wie komplexe Szenen unterstützt werden können.

Schlussendlich werden die essentiellen Grundkonzepte des in dieser Arbeit entwickelten Verfahrens im Unterkapitel 4.7 zusammengefasst.

4.2 Betrachtung existierender Verfahren

Nur ein Verfahren aus Kapitel 3.1 hat sowohl für Geschwindigkeit als auch für Dynamik die volle Punktzahl von fünf Punkten erhalten. Es existieren sogar nur zwei Verfahren, die bei der Dynamik überhaupt die volle Punktzahl erreicht haben. Beide Verfahren wurden von Dachsbacher und Stamminger [DS05,DS06] vorgestellt und basieren auf einem Instant-Radiosity-Ansatz und der direkten Anwendung von Reflective Shadow-Maps (RSMs). Dies wird als Indikator gesehen, dass sowohl Instant Radiosity als auch die Verwendung von RSMs eine gute Grundlage bilden, um ein voll dynamisches Verfahren zu entwickeln. Jedoch sind die Verfahren von Dachsbacher und Stamminger bei der direkten Anwendung im Vergleich zu den anderen Verfahren qualitativ schlechter und ziemlich eingeschränkt bei der Abbildung optischer Phänomene. Zum einen können mit diesen Verfahren keine weichen Schatten berechnet werden und zum anderen muss der Einfluss des indirekten Lichts räumlich begrenzt werden, damit eine Anwendung in Echtzeit möglich ist. Beide Einschränkungen sind für das zu konzipierende Verfahren nicht akzeptabel. Viele Ansätze nutzen aber RSMs, um die initiale Verteilung des indirekten Lichts in der Szene vorzunehmen [KD10,CNS*11,RGK*08]. Für diese

Aufgabe ist das Reflective Shadow-Mapping gut geeignet. Sieht man von dem Reflective Shadow-Mapping ab, existiert kein weiteres interaktives Verfahren, das in der Kategorie Dynamik die volle Punktzahl erreicht hat. Reduziert man die Anforderungen für die verbleibenden Verfahren bei der Dynamik um einen Punkt auf vier, kommen noch die folgenden Verfahren für Virtual- und Augmented-Reality-Anwendungen in Frage:

Tabelle 4.1: Auflistung interaktiver Verfahren, die ein sehr gutes Laufzeitverhalten aufweisen und eine relativ hohe Dynamik erlauben.

Methode	Autor		Geschw.	Qualität	Dynam.	Skalierb.	Transport
Cascaded Light-Propagation-Volumes (CLPV)	Kaplanyan & Dachsbacher	[KD10]	5	2	4	5	$LD^{+}(D\|V\|S)E$
Screen-Space Directional Occlusion (SSDO)	Ritschel et al.	[RGS09]	5	1	4	4	$LDDE$
Voxel-Cone-Tracing (VCT)	Crassin et al.	[CNS*11]	5	3	4	3	$LD^{+}(D\|S)E$

Das SSDO-Verfahren beschränkt sich auf Mesostrukturen von der Größe weniger Pixel im Bildraum, weswegen es für das Anwendungsfeld dieser Dissertation keine Lösung bietet. Die CLPV unterstützen sowohl große Szenen als auch einen ausreichend komplexen Lichtpfad. Durch die grobe Szenenapproximation für die indirekte Beleuchtung durch ein volumetrisches Gitter (siehe Kapitel 3.1.4) ist die Qualität der globalen Beleuchtung jedoch stark beschränkt. Dies macht sich besonders bei glänzenden Reflexionen und weichen Schatten bemerkbar. Im Vergleich dazu bietet das VCT (siehe Kapitel 3.1.5) eine bessere Qualität, im Speziellen bei glänzenden Reflexionen. Das VCT-Verfahren ist jedoch speicherintensiv, da ein feiner Voxel-Octree im Grafikkartenspeicher abgelegt werden muss. Dieser Octree kann für große Szenen nicht vollständig im Speicher gehalten werden und muss somit durch entsprechende Out-of-Core-Methoden nachgeladen werden. Dies lässt darauf schließen, dass komplexe Szenen nicht mit dem VCT-Verfahren abgebildet werden können. Zumindest sind Probleme zu erwarten, wenn die Geometriedaten der Szene selbst schon aufgrund ihrer Größe durch Out-of-Core-Methoden in den Speicher geladen werden müssen. Ein weiteres Problem beim VCT ist, dass hohe Auflösungen (z. B. Full-HD) nicht mit einer entsprechenden Bildwiederholfrequenz dargestellt werden können, demnach sind Renderings für hochauflösende projektive Leinwände problematisch (wie sie in VR-Umgebungen häufig genutzt werden). Bei Animationen sind sowohl das CLPV- als auch das VCT-Verfahren nicht optimal. Bei den CLPV verursachen Animationen von der Kamera und von den Modellen Artefakte. Modellanimationen erzeugen hauptsächlich Artefakte bei den weichen Schatten, denn hier wird die Diskretisierung der Szene durch das volumetrische Gitter sichtbar. Beim VCT muss bei der Veränderung der Szene der Voxel-Octree aktualisiert werden, was wiederum rechenintensiv ist und somit Animationen teurer werden lässt. Beide Verfahren werden in Kapitel 6.2 noch genauer dem Verfahren dieser Arbeit gegenübergestellt, um spezielle Vor- und Nachteile abzuwägen.

An dieser Stelle kann festgehalten werden, dass keines der hier diskutierten Verfahren in der Lage ist, die Forderungen bezüglich des Anwendungsfeldes dieser Arbeit zu erfüllen. Demnach kann kein vorhandenes Verfahren genutzt werden und es muss ein neues Verfahren entwickelt werden. Dieses wird in den folgenden Unterkapiteln anhand der oben genannten Anforderungen konzipiert.

4.3 Dynamik und Interaktivität

Die effiziente und die artefaktfreie Behandlung von Animationen ist eines der wichtigsten Anforderungen an das zu entwickelnde Verfahren. Eine VR- oder AR-Simulation, die weder Animationen noch Interaktivität beinhaltet, ist von geringem Nutzen. Precomputed-Illumination-Verfahren (siehe Kapitel 3.1.3) beschränken die Dynamik häufig auf die Steuerung des Lichts, des Materials und der Kamera. Diese Verfahren erscheinen in besonderer Weise ungeeignet, um ein voll dynamisches System zu entwickeln. Andere Verfahren, wie das Monte-Carlo-Ray-Tracing oder das Photon-Mapping, benötigen komplexe Beschleunigungsstrukturen, um eine effiziente Strahlenverfolgung innerhalb der Szene zu gewährleisten. Diese Strukturen sind für die Traversierung entlang von Strahlen optimiert und nicht auf die dynamische Anpassung durch Animationen. Dies erscheint in gewisser Weise sinnvoll, denn die Anpassung der Struktur muss nur einmal pro Frame erfolgen, während dagegen mehrere Millionen Strahlen für ein einzelnes Bild verfolgt werden müssen. Sinken die Kosten für die Strahlenverfolgung, zum Beispiel durch die häufig ausgenutzte temporäre Kohärenz in konsekutiven Frames, fällt die Anpassung der Beschleunigungsstruktur besonders ins Gewicht. Zum einen, weil sie einen verhältnismäßig großen Anteil an den Gesamtkosten veranschlagt und zum anderen, weil die Szene temporär inkohärent wird, was wiederum zu einer höheren Anzahl von verfolgten Strahlen führt. Somit scheiden auch das Ray-Tracing und das Photon-Mapping als Grundlage für ein voll dynamisches Verfahren aus. Discrete Ordinate Methods benötigen ebenfalls eine separierte Struktur zur Abbildung von Beleuchtungseffekten. Diese Struktur wird nicht zur Strahlenverfolgung, sondern für die diskrete Ausbreitung des Lichts über mehrere Iterationen genutzt. Die indirekte Beleuchtung für ein Modell kann hier aus den benachbarten Gitterknoten berechnet werden. Jedoch können die Gitterknoten nur sehr grob die tatsächliche Szene repräsentieren, weswegen bei Bewegungen Artefakte auftreten können. Dies ist für die diffuse Lichtausbreitung unproblematisch, da das Signal niederfrequent ist und entsprechend geglättet erscheint. Bei glänzenden Reflexionen und Schatten kann dies jedoch zu einem Problem werden. Die Auflösung der Gitterstruktur zu erhöhen ist hier keine Alternative, denn für die feinere Struktur werden auch mehr Iterationsschritte zur Berechnung der Lichtausbreitung benötigt und die Speicheranforderungen nehmen kubisch zu. Die Verwendung von Kaskaden der Gitterstruktur mit verschiedenen Auflösungen, wie von Kaplanyan und Dachsbacher [KD10] vorgeschlagen, ist eine valide Möglichkeit, das Problem zu beheben. Jedoch führen diese Kaskaden wiederum zu Artefakten bei Kameraanimationen.

Instant Radiosity (IR), worunter auch die Many-Light-Ansätze fallen, sowie Screen-Space-Finite-Elements-Methoden kommen meist ohne komplexe Zusatzstrukturen aus. Man kann anhand der vorgestellten Verfahren aus Kapitel 3.1.6 erkennen, dass Animationen von diesen Ansätzen besser unterstützt werden. Diese Aussage muss jedoch ins Verhältnis gesetzt werden, denn durch die Diskretisierung der Beleuchtung durch Virtual Point-Lights (VPLs), sowie durch die Diskretisierung der Empfängeroberflächen bei Bildraummethoden entstehen flackernde Artefakte bei Bewegungen. Diese Artefakte können durch eine höhere Anzahl von VPLs und/oder Empfängeroberflächenelementen so stark reduziert werden, dass sie nicht mehr wahrnehmbar sind. Es entsteht so jedoch ein Laufzeitproblem, denn jedes VPL muss auf jedes sichtbare Oberflächenelement angewendet werden. Die Anzahl der Oberflächenelemente und der VPLs kann demnach nicht beliebig erhöht werden, weswegen die meisten schnellen IR-Verfahren deutliche Artefakte aufweisen.

Trotz der Artefaktbildung bei Animationen ist der Instant-Radiosity-Ansatz eine gute Basis für die Entwicklung eines eigenen Verfahrens mit hohen Ansprüchen bei der Dynamik. Er verursacht in der Regel keine Extrakosten bei Animationen und beschränkt diese grundsätzlich nicht. Auch die Ausbreitungseigenschaften des (indirekten) Lichts werden nicht eingeschränkt, so dass sich mit dem Verfahren sowohl nahe, als auch unendlich ferne Lichtquellen realisieren lassen. Des Weiteren profitiert IR von der Raster-Pipeline der Grafikkarte, so dass eine reine GPU-Implementierung möglich ist. Das Problem der Artefaktbildung bei Animationen muss jedoch gelöst werden. Dies kann als wesentlicher Beitrag dieser Arbeit verstanden werden. Da die Artefaktbildung aus Laufzeitoptimierungen resultiert, soll hierauf im nächsten Unterkapitel ein besonderer Fokus gelegt werden.

4.4 Geschwindigkeit

Im vorherigen Unterkapitel wurde erläutert, dass wegen der guten Unterstützung von Animationen das Instant Radiosity eine geeignete Grundlage bildet, um ein neues Verfahren zu entwickeln, das eine hohe Dynamik in der Szene erlaubt. Allerdings führen Laufzeitoptimierungen bei den meisten IR-Ansätzen dazu, dass Artefakte bei Animationen sichtbar werden. In diesem Kapitel wird erklärt, wie IR-Ansätze grundsätzlich beim Laufzeitverhalten optimiert werden können und warum diese Optimierungen zu temporären Inkohärenzen führen. Die Analyse der Ursachen der temporären Inkohärenzen erlaubt wiederum Rückschlüsse zur Konzeptionierung des eigenen Verfahrens.

Instant-Radiosity-Ansätze lassen sich in zwei grobe Teilschritte zerlegen, wovon jeder Teilschritt separat optimiert werden kann:

1. Verteilung der VPLs in der Szene.
2. Anwendung der VPLs auf die sichtbaren Oberflächenelemente.

Die initiale Verteilung der VPLs kann durch Reflective Shadow-Maps erfolgen (siehe Kapitel 3.1.6). Damit lassen sich VPLs auf den Oberflächen erzeugen, die sich im Sichtfeld der primären Lichtquelle befinden. Man erhält so die VPLs, die die erste indirekte Reflexion abbilden. Weitere indirekte Reflexionen können mit der RSM-Methode nicht effizient berechnet werden, da jedes VPL seine eigene RSM erzeugen müsste, was zu einem exponentiellen Laufzeitverhalten bezüglich der Anzahl der indirekten Reflexionen (Bounces) führen würde. In der Regel reicht aber die erste indirekte Reflexion aus, um den optischen Eindruck einer Szene deutlich zu verbessern und insbesondere geometrische Zusammenhänge hervorzuheben. Weitere Reflexionen verursachen meistens nur noch marginale Veränderungen in der Darstellung [TL04].

Die Anwendung der VPLs auf die sichtbaren Oberflächenelemente führt zu einer Laufzeitkomplexität von $O(mn)$, wobei n die Anzahl der VPLs und m die Anzahl der sichtbaren Oberflächenelemente bezeichnet. Erzeugt jedes VPL noch seine eigene Shadow-Map, ergibt sich eine Laufzeitkomplexität von $O(mn + kn)$, wobei k die gemittelte Anzahl der Oberflächenelemente aus Sicht der VPLs ist. Es existieren zwei Möglichkeiten zur Optimierung: man reduziert die Anzahl der VPLs oder man reduziert die Anzahl der Empfängeroberflächen. Beide Ansätze finden bereits Anwendung und sind orthogonal zueinander (siehe folgende Diskussion).

In den nachfolgenden zwei Unterkapiteln soll auf die bestehenden Möglichkeiten zur Reduzierung der VPLs und Oberflächenpunkte eingegangen werden. Aus den damit einhergehenden Einschränkungen und Artefakten hat sich die Idee für diese Arbeit entwickelt. Als Grundlage zur Betrachtung

sollen Reflective Shadow-Maps dienen, da sie das Problem durch ihre projektiven Eigenschaften auf eine zweidimensionale Domäne abbilden, was die Erläuterungen vereinfacht. Bei einer RSM bildet jeder Pixel ein VPL ab. Hat z. B. eine RSM eine Auflösung von 128x128 Pixeln, so ergeben sich bereits 16.384 VPLs, die jeweils auf die sichtbaren Oberflächenpunkte angewendet werden müssen. Geht man von einem Deferred-Shading-Ansatz aus, wird jeder sichtbare Oberflächenpunkt durch einen Pixel im G-Buffer repräsentiert. Bei einer Full-HD-Auflösung von 1920x1080 Bildpunkten müssen also 128x128x1920x1080=33,9 Milliarden Berechnungen durchgeführt werden. So viele Berechnungen sind in Echtzeit (noch) nicht möglich, deshalb müssen Optimierungsmethoden zur Anwendung kommen.

4.4.1 Reduzierung der VPLs

Verfahren zur Reduzierung der VPLs lassen sich als Many-Light-Ansätze klassifizieren und wurden bereits in Kapitel 3.1.7 beschrieben. Im Zusammenhang mit RSMs lassen sich diese Verfahren auf den zweidimensionalen Bildraum reduzieren. Um die Anzahl der VPLs innerhalb einer RSM zu verringern, kann man zwei Möglichkeiten unterscheiden: Importance-Sampling oder Clustering. Das Importance-Sampling zeichnet sich dadurch aus, dass es eine feste Anzahl von VPLs generiert. Dies sorgt für stabile Bildraten. Allerdings besteht die Möglichkeit, dass sich die projizierte Szene nicht mit einer festen Anzahl von VPLs ausreichend approximieren lässt. Dies ist dann der Fall, wenn die Szenenoberflächen sehr komplexe und filigrane Strukturen beinhalten. In diesen Fällen kann es entweder passieren, dass bestimmte, für die Wahrnehmung wichtige Details nicht in der Beleuchtung wiedergegeben werden oder dass in konsekutiven Bildern das Sampling ständig variiert, was sich in Flackern bemerkbar macht. Ein weiteres Problem besteht darin, dass lokale Animationen die komplette VPL-Verteilung verändern können. So kann sich die Beleuchtung in Bereichen verändern, die eigentlich nicht durch die Animation beeinflusst werden. Die Metriken, die die Verteilung der VPLs bestimmen, können abhängig oder unabhängig vom Betrachter gewählt werden. Sind sie unabhängig vom Betrachter, werden besonders viele VPLs auf Oberflächen verteilt, auf denen starke Reflexionen auftreten. Bei betrachtungsabhängigen Ansätzen werden auf den Oberflächen viele VPLs erzeugt, die am fertigen Gesamtbild einen hohen Anteil tragen. Insgesamt lässt sich sagen, dass das Importance-Sampling bessere Ergebnisse liefern kann als eine gleichmäßige Verteilung der VPLs. Im Zusammenhang mit Animationen stellt es aber, wegen der ständig wechselnden Verteilung der VPLs, keine optimale Lösung dar und kommt deshalb als Optimierungsansatz für das Verfahren dieser Arbeit nicht in Frage.

Beim Clustering werden mehrere VPLs zu einem VPL oder zu einer Flächenlichtquelle (VAL - Virtual Area-Light) zusammengefasst. Auch bei diesem Prinzip kann man sicherstellen, dass nur eine feste Menge von Clustern, und somit VPLs, erzeugt wird (zum Beispiel durch das k-Means-Clustering). Für diese Form des Clusterings gelten jedoch die gleichen Überlegungen wie beim Importance-Sampling. Ein adaptives Clustering dagegen erzeugt so viele Lichtquellen, wie dies die Struktur der projizierten Szene vorschreibt. Sind die Oberflächen komplex, werden entsprechend viele VPLs erzeugt, sind sie homogen und einfach, werden sehr wenige erzeugt. Ob Oberflächen zusammengefasst werden können, hängt von ihren Reflexionseigenschaften und ihrer Geometrie ab, die durch Pixel in der RSM repräsentiert werden. Grundsätzlich lassen sich benachbarte Pixel (VPLs) zusammenfassen, wenn der Unterschied zwischen deren Oberflächennormalen und deren Reflexionsspektren einen bestimmten Grenzwert nicht überschreitet. Die Anwendung der Cluster lässt sich ebenfalls adaptiv gestalten:

Lichtquellenferne Empfängeroberflächen können durch große Cluster beleuchtet werden, während nahe Elemente individuelle VPLs verwenden, um hohe Frequenzen abzubilden.

Das adaptive Light-Clustering erscheint als guter Ansatz, um die Komplexität der Beleuchtung zu verringern. Sofern es konservativ durchgeführt wird (also feine Strukturen nicht zusammengefasst werden), verursachen Animationen kaum Artefakte. Jedoch hat die Adaptivität ihren Preis: Die Anzahl der VPLs kann stark schwanken und somit die Bildwiederholrate beeinflussen. Auch der Zugriff auf den Grafikkartenspeicher lässt sich nicht effizient organisieren, da die Cluster beliebig verteilt ausfallen können, was zu einem langsamen verteilten Speicherzugriff führt. Demnach erscheint das Light-Clustering als Optimierungsmethode für ein neues Verfahren nicht unmittelbar geeignet.

4.4.2 Reduzierung der sichtbaren Empfängeroberflächen

Ein zur Reduzierung der VPLs orthogonaler Ansatz ist die Reduzierung der Anzahl der sichtbaren Oberflächen, die Licht von den VPLs reflektieren. Ganz analog zu den RSMs kann man die Empfängeroberflächen ebenfalls im zweidimensionalen Bildraum betrachten. Dies wird durch das Deferred Shading (siehe Kapitel 3.1.5) möglich, bei dem anstatt eines Framebuffers mehrere G-Buffer genutzt werden. Diese G-Buffer beinhalten die Reflexions- und Geometrieeigenschaften (Normale und Position) der sichtbaren Oberflächenelemente in Form von Pixeln. Wenn hier von der Reduzierung der sichtbaren Empfängeroberflächen gesprochen wird, ist damit nicht gemeint, dass nur noch eine bestimmte Anzahl von G-Buffer-Pixeln darstellt wird, sondern dass nur auf eine bestimmte Untermenge von Pixeln alle VPLs angewendet werden. Dies kann auf zwei Arten geschehen: Entweder man reduziert den Einflussbereich eines VPL geometrisch, oder man wählt eine repräsentative lose Untermenge aus allen sichtbaren Oberflächenelementen aus und beleuchtet nur diese mit den VPLs. Der zweite Ansatz benötigt eine weitere Verarbeitungsstufe, bei der die Ergebnisse der gewählten losen Oberflächenpunkte auf den Rest der sichtbaren Oberflächenelemente interpoliert bzw. übertragen werden. Diese Methode bezeichnet man je nach Repräsentation des Lichts in den losen Empfängerpixeln als Irradiance-Caching [WRC88] oder Radiance-Caching [KGP*05]. Ein Cache bezeichnet dabei einen losen Empfängerpunkt. Sowohl das Irradiance-Caching als auch das Radiance-Caching werden hauptsächlich in Offline-Renderverfahren verwendet, um die Bildsynthese zu beschleunigen, es existieren aber auch Caching-Verfahren für Echtzeitanwendungen. Beim Irradiance-Caching wird pro Cache die Beleuchtungsstärke gespeichert, während beim Radiance-Caching die eintreffende Leuchtdichte gespeichert wird. Somit eignet sich das Irradiance-Caching nur für diffuse Reflexionen, während mit dem Radiance-Caching auch betrachtungsabhängige glänzende Reflexionen interpoliert werden können. Das Radiance-Caching ist in der Regel speicherintensiver als das Irradiance-Caching, da die richtungsabhängigen Informationen mehr Daten benötigen. Die Verteilung der Caches wird normalerweise im Bildraum der Kamera vorgenommen, und zwar so, dass in Bereichen, in denen hochfrequente Lichtveränderungen zu erwarten sind, besonders viele, und in niederfrequenten Bereichen besonders wenige Caches verteilt werden. Dies geschieht ganz analog zur Verteilung der VPLs.

Nichols und Wyman [NW09] haben ein hierarchisches Echtzeitverfahren für einen Irradiance-Caching-Ansatz vorgestellt, der sich komplett für die GPU implementieren lässt. Die grundsätzliche Idee besteht darin, mit Hilfe von Bildpyramiden (Mip-Maps) der G-Buffer die Cache-Verteilung vorzunehmen. Dabei wird für je vier benachbarte Oberflächenpixel geprüft, ob deren Normale und Position stark variieren. Ist dies nicht der Fall, können die Oberflächenpixel zu einem Cache

zusammengefasst werden, wenn sie jedoch variieren, werden vier separate Caches erzeugt. Dies wird sukzessiv für alle Mip-Map-Ebenen (Stufen der Bildpyramide) durchgeführt. Schlussendlich erhält man eine Cache-Verteilung wie in Abbildung 4.1 dargestellt. Die Interpolation der diffusen Beleuchtung auf Basis dieser Verteilung kann einfach erfolgen, da nur Farbkomponenten interpoliert werden müssen. Das Problem bei diesem Verfahren, sowie bei allen anderen Irradiance-Caching-Verfahren, besteht darin, dass bei der Nutzung weniger Caches Oberflächendetails vernachlässigt werden, während eine hohe Anzahl von Caches die Bildwiederholrate sinken lässt. Beim Radiance-Caching können die Oberflächendetails erhalten bleiben, da in jedem Cache verschiedene Werte für verschiedene Richtungen gespeichert werden (Leuchtdichte, üblicherweise durch Spherical-Harmonics-Koeffizienten repräsentiert). So können Variationen in den Normalen bei der Interpolation berücksichtigt werden. Scherzer et al. [SNR*12] haben kürzlich mit ihrem Pre-Convolved Radiance-Caching einen interaktiven Ansatz vorgestellt, der solche Caches nutzt, um damit glänzende indirekte Reflexionen abzubilden. Allerdings ist dies in Echtzeit nur für statische Szenen möglich und die Caches benötigen viel Speicher, da die eingehende Leuchtdichte für jeden Cache in einer separaten Textur gespeichert wird.

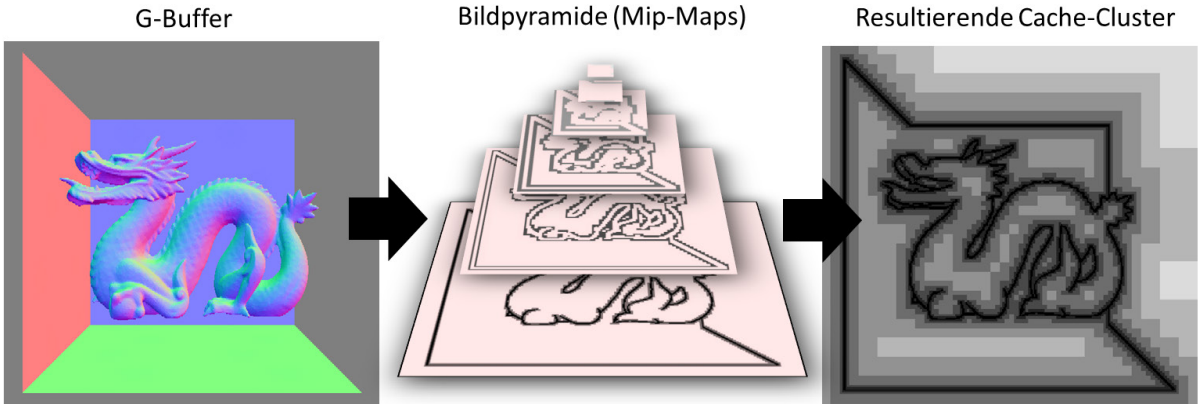

Abbildung 4.1: Prinzip der Cache-Erzeugung nach Nichols und Wyman [NW09]. Auf Basis des G-Buffers werden Flächen zusammengefasst, die homogene Eigenschaften besitzen. Je größer die homogene Fläche ist, desto höher kann diese in der Bildpyramide (Mip-Maps) gespeichert werden. In der Bildpyramide ist jeder gültiger Pixel (die schwarzen und grauen) ein Cache, der beleuchtet werden muss. Für die ungültigen Werte wird eine Interpolation durchgeführt.

Alle bekannten interaktiven echtzeitfähigen Irradiance- und Radiance-Caching-Verfahren erzeugen bei Animationen Artefakte. Diese Artefakte lassen sich nur verringern, indem mehr Caches verwendet werden. Dies führt wiederum dazu, dass die Bildwiederholrate sinkt. Dass Artefakte überhaupt bei Animationen auftreten, hängt damit zusammen, dass die Caches im Bildraum verteilt werden. Verändert sich das Bild durch eine Animation, so führt dies zu einer Neuverteilung der Caches (lokal oder global). Diese Neuverteilung ist inkonsistent zur Animation. Rotiert zum Beispiel ein Objekt, so werden die Caches nicht mit dem Objekt bewegt, sondern anhand der sich ändernden Projektion verteilt: Die Folge ist erkennbares Flackern.

In dieser Arbeit wird der Radiance-Caching-Ansatz aufgegriffen, denn durch diesen kann die Laufzeit der Anwendung deutlich verbessert werden. Mit dem Wissen, dass die Artefakte häufig durch die Neuverteilung der Caches im Bildraum resultieren, lässt sich ein Ansatz entwickeln, der Artefakte bei Animationen so stark reduziert, dass sie kaum noch wahrnehmbar sind. Ferner lässt sich durch eine alternative Repräsentation der eingehenden Leuchtdichte im Cache die Komplexität des Radiance-

Cachings vermindern. Diese Erweiterungen können wie folgt zusammengefasst werden und sind Errungenschaften dieser Dissertation:

1. Die Caches werden im Raum des Modells verteilt und gespeichert und nicht im Bildraum. Hierdurch werden die Caches mit dem Modell transformiert und flackernde Artefakte verschwinden.
2. Die eingehende Leuchtdichte wird mit der BRDF des Caches gewichtet. Anstatt das eingehende Licht als Farbkomponenten zu speichern, wird es in Form einer analytischen Proxy-Lichtquelle gespeichert, die später für die Interpolation verwendet werden kann. Diese Darstellung ist kompakt und erlaubt die Berücksichtigung individueller Oberflächennormalen (filigrane Oberflächen).

Dieser neue Ansatz reduziert viele Nachteile bestehender Verfahren und erlaubt eine effiziente Darstellung des indirekten Lichts durch VPLs. Ferner kann das Light-Clustering, das im vorherigen Unterkapitel beschrieben wurde, als zusätzliche Optimierungsmethode genutzt werden, um das Verfahren noch weiter zu beschleunigen.

4.5 Qualität

Bei der Entwicklung eines Echtzeitverfahrens müssen erfahrungsgemäß optische Qualitätseinbußen hingenommen werden. Kein Verfahren wird in naher Zukunft eine erschöpfende physikalische Simulation gewährleisten können, weshalb die Konzentration auf optische Phänomene sinnvoll ist. Demnach kann man die Qualität unter anderem anhand der Anzahl der unterstützten optischen Phänomene beurteilen. Auch die Länge des unterstützten Lichtpfads kann als objektives Kriterium verstanden werden. Ein Vergleich mit einer physikalisch korrekten Simulation ist nicht immer sinnvoll, weswegen auch die gestellte Anforderung in dieser Arbeit optische Plausibilität und nicht physikalische Korrektheit heißt (siehe Kapitel 2.1.9). Verfahren müssen zwar Ergebnisse liefern, die zu der tatsächlichen Lichtausbreitung passen, jedoch sind kleinere Fehler, wenn sie der Wahrnehmung des Betrachters verborgen bleiben, akzeptabel.

Für den in dieser Arbeit gewählten Ansatz des Instant Radiositys existieren bereits interaktive Verfahren. Diese Verfahren setzen den State-of-the-Art für die Darstellungsqualität fest. Da IR-Ansätze die indirekte Beleuchtung mit Hilfe von hemisphärischen Punktlichtquellen (VPLs) berechnen, kann man bei vielen IR-Implementierungen punktförmige Artefakte beim indirekten Licht beobachten. Diese Artefakte lassen sich vermindern, indem anstatt Punktlichtquellen Flächenlichtquellen verwendet werden. Zur effizienten Anwendung dieser Flächenlichtquellen können Prinzipien von Finite-Elements-Methoden genutzt werden. Diese können also ergänzend vom konzeptionierten Verfahren eingesetzt werden, um die Darstellungsqualität der indirekten Reflexionen zu verbessern.

Typischerweise unterstützen interaktive IR-Ansätze diffuse, jedoch keine glänzenden Reflexionen in einer dynamischen Szene (siehe Kapitel 3.1.6). Lediglich bei dem Verfahren von Novak et al. [NED11] wird ein Lichtpfad angegeben, der glänzende Reflexionen beinhaltet. Diese Reflexionen führen aber bei Animationen zu sichtbaren Artefakten. Das Verfahren dieser Arbeit soll glänzende und diffuse Reflexionen unterstützen können, jedoch ohne Artefakte bei Animationen hervorzurufen. Darüber hinaus soll das Verfahren die Berechnung von weichen Schatten erlauben, da IR-Ansätze wie beispielsweise das Imperfect Shadow-Mapping [RGK*08,REH*11] diese ebenfalls unterstützen. Für die Berechnung von Schatten müssen blockierende Elemente in einer Szene berücksichtigt werden.

Dies verlangt nach einer effizienten approximativen Abbildung der blockierenden Elemente, die durch einfache finite Elemente erfolgen kann.

Grundsätzlich unterstützen einige IR-Ansätze multiple Reflexionen im diffusen Lichtpfad [SIM*06, LSK*07, NED11], allerdings können diese nicht mit Bildwiederholraten berechnet werden, die für eine Echtzeitanwendung erforderlich sind. Die meisten echtzeitfähigen Verfahren konzentrieren sich deshalb auf die effiziente Berechnung der ersten indirekten Reflexionen. Dies soll auch für das Verfahren dieser Arbeit gelten, aber konzeptionell soll es multiple diffuse Reflexionen nicht ausschließen.

4.6 Komplexität

Große, komplexe Szenen sind zwar in AR-Umgebungen eher untypisch, dafür findet man sie sehr häufig in VR-Umgebungen. Diese Szenen sind in der Regel so groß, dass sie nicht vollständig in den Grafikkartenspeicher geladen werden können und somit dynamisch nachgeladen werden müssen. Globale Beleuchtungsverfahren, die für die Simulation selbst sehr viel Speicher belegen, sind für diese Szenarien ungeeignet. Durch den gewählten Ansatz des Instant Radiositys in Verbindung mit dem neuen Radiance-Caching-Ansatz ist sichergestellt, dass das konzipierte Verfahren sehr wenig Speicher benötigen wird. Diese Aussage ist jedoch nur korrekt, wenn es möglich ist, die Anzahl der Caches gering zu halten. Dies muss bei der Entwicklung im besonderen Maße berücksichtigt werden. Ferner muss wegen der Prämisse der Verteilung der Caches im Modellraum aus Kapitel 4.4.2 gelten, dass nur Caches für die indirekte Beleuchtung berücksichtigt werden, die zum fertigen Bild beitragen. Dies muss gelten, weil große Szenen unter Umständen sehr viele solcher Caches beinhalten können. Darüber hinaus sollte es möglich sein, weitere Level-of-Detail-Ansätze in das Verfahren zu integrieren.

4.7 Zusammenfassung

In diesem Unterkapitel sollen die wesentlichen Konzepte des entwickelten Verfahrens den einzelnen Techniken zugeordnet werden, in die sie eingruppiert werden können. Ferner wird eine kurze Zusammenfassung des Konzepts gegeben.

Das Verfahren kombiniert, erweitert und nutzt Elemente von Instant-Radiosity-, Radiance-Caching-, Finite-Elements-, und Many-Light-Methoden:

- Instant Radiosity: Reflective Shadow-Maps werden eingesetzt, um die initiale Verteilung für indirekte Lichtquellen vorzunehmen.
- Radiance-Caching: Das indirekte Licht wird nur für eine lose Untermenge von Oberflächenpunkten (Caches) evaluiert. Anschließend wird es anhand der Ergebnisse der Caches über die komplette sichtbare Szene interpoliert. Hierfür wird ein neues und effizientes Interpolationsverfahren vorgestellt.
- Finite Elements: Sowohl die indirekten Lichtquellen aus der RSM, als auch die Empfängeroberflächen, werden in Form von approximativen endlichen Flächen repräsentiert. Hierfür wird die RSM erweitert und ein neuer Algorithmus zur Berechnung von weichen Schatten vorgestellt.

- Many Light: Dieser Ansatz wird als zusätzliche Optimierungsmöglichkeit genutzt, um die indirekten Lichtquellen zu gruppieren und so einen sublinearen Berechnungsaufwand zu gewährleisten.

Die grundlegende Idee des Verfahrens besteht darin, einen Instant-Radiosity-Ansatz, wie die Reflective Shadow-Maps, so zu erweitern, dass eine durchgehend indirekte Beleuchtung für die komplette Szene in Echtzeit berechnet werden kann, ohne störende Artefakte bei Animationen zu produzieren und ohne den Einflussbereich des indirekten Lichts zu beschränken (z. B. durch Beschränkung der Ausbreitung des indirekten Lichts). Dies wird durch eine neue Radiance-Caching-Methode erreicht. Hierbei werden Caches nicht wie bei anderen Verfahren im Bildraum der Kamera, sondern im Objektraum verteilt. Das hat den Vorteil, dass weniger Caches benötigt werden und dass diese auf natürliche Art und Weise mit dem Objekt mittransformiert werden können, was zu einer besseren temporären Kohärenz führt. Die neuartige Interpolationsmethode erlaubt bereits mit sehr wenigen Radiance-Caches die Berechnung von diffusen und glänzenden Reflexionen.

Bei der Entwicklung der Interpolationsmethode hat sich ergeben, dass diese ebenfalls geeignet ist, um Multiple-Subsurface-Scattering-Phänomene abzubilden. Auch die Erzeugung von weichen Schatten wird durch das neue Verfahren unterstützt, ohne dass jedes virtuelle indirekte Licht eine eigene Shadow-Map berechnen muss, wie dies z. B. bei den Imperfect Shadow-Maps [RGK*08] der Fall ist.

Das konzipierte Verfahren unterstützt bei voll dynamischen Szenen den folgenden Lichtpfad (mit ergänztem Subsurface-Scattering):

$$LD^{+}(S|D|VD)E.$$

Komplexe Szenen werden durch eine geringe Anzahl von Caches sowie durch ein neues Culling-Verfahren unterstützt. Das Culling-Verfahren erlaubt die effiziente Maskierung der Caches, die zum final sichtbaren Bild beitragen.

5 Das LightSkin-Verfahren

5.1 Überblick

Für das Verständnis der weiteren Kapitel ist es vorteilhaft, die grundlegenden Schritte des LightSkin-Verfahrens zu kennen. Prinzipiell lässt sich das Verfahren in fünf grobe Teilschritte zerlegen (siehe auch Abbildung 5.1):

1. **Verteilung von Caches**: Caches sind hier lose verteilte Punkte auf Modelloberflächen, die a priori, also vor der Simulation, verteilt werden müssen. Wie diese Verteilung vorgenommen werden kann und welche Daten ein Cache benötigt, ist Thema von Kapitel 5.6. Caches erlauben es, aufwändige Berechnungen nur für eine geringe Anzahl von Oberflächenpunkten durchzuführen und für diese kompakte Zwischenergebnisse zu speichern. Anschließend werden die Zwischenergebnisse genutzt, um mit Hilfe eines weniger aufwändigen Interpolationsverfahrens ein durchgehendes Ergebnis für alle sichtbaren Oberflächenpunkte zu erzeugen.
2. **Erzeugung von indirekten Lichtquellen**: Das indirekte Licht in der Szene wird durch eine endliche Menge von homogenen Flächenlichtquellen repräsentiert, die mit Hilfe von erweiterten Reflective Shadow-Maps (siehe auch Kapitel 3.1.6) in der Szene verteilt werden. Diese erweiterten Reflective Shadow-Maps werden in Kapitel 5.2 besprochen.
3. **Beleuchtung von Caches**: Jede indirekte Lichtquelle aus Schritt 2 (homogene Flächenlichtquelle) beleuchtet jeden Cache in der Szene. Die Caches speichern das einfallende Licht in kompakter Form ab. Wie diese kompakte Form für Reflexionen auf Basis eines BRDF-Modells abgebildet werden kann, wird in Kapitel 5.3 beschrieben, während in Kapitel 5.5 erläutert wird, wie sie für Streuungsprozesse unterhalb der Modelloberfläche, auf Basis eines BSSRDF-Modells, abgebildet werden kann.
4. **Schattierung von Caches**: Im vorhergehenden Schritt wird davon ausgegangen, dass das indirekte Licht, ausgehend von den homogenen Flächenlichtquellen, nicht durch Objekte in der Szene blockiert werden kann, weswegen keine weichen Schatten entstehen. In diesem Teilschritt wird für jeden Cache geprüft, wie viel des einfallenden Lichts durch andere Objekte blockiert wird. Je nach Grad der Blockierung wird dann das empfangene Licht im Cache gedämpft. Die Details hierzu sind Gegenstand von Kapitel 5.4.
5. **Interpolation der Beleuchtung über alle sichtbaren Oberflächen**: Bei der Interpolation werden die Daten in den Caches, die die einfallende indirekte Beleuchtung kompakt repräsentieren, genutzt, um ein durchgehendes Beleuchtungsergebnis für den kompletten Bildausschnitt zu interpolieren (alle sichtbaren Oberflächenpunkte). Je nach optischem Phänomen (Reflexion oder Scattering) werden verschiedene Modelle zur Interpolation genutzt. Das Prinzip der Interpolation wird in Kapitel 5.3 respektive Kapitel 5.5 erläutert.

Zusätzlich zu diesen fünf Bearbeitungsschritten wird in Kapitel 6.1 auf die Implementierung des Verfahrens für die Grafikkarte eingegangen. Diese beinhaltet ein effizientes Culling-Verfahren, Ansätze für eine adaptive Cache-Verteilung und weitere Optimierungen. Schlussendlich werden in Kapitel 6.2 einige Ergebnisse des Verfahrens vorgestellt und diskutiert.

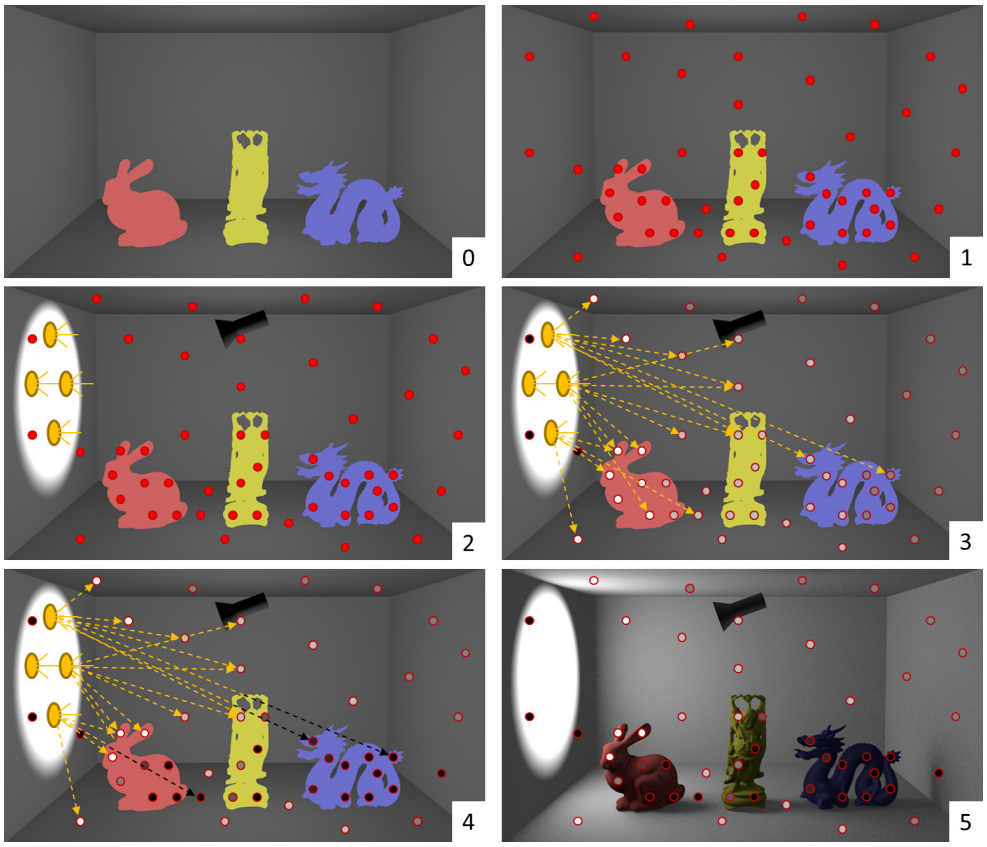

Abbildung 5.1: Die fünf Schritte des LightSkin-Verfahrens. (0) geladene Szene, (1) Verteilung der Caches a priori im Objektraum, (2) Verteilung der indirekten Flächenlichtquellen durch erweiterte RSMs, (3) Beleuchtung der Caches, ohne Verdeckungen zwischen Objekten zu berücksichtigen (es wurden nur einige Pfeile eingezeichnet, jedoch werden alle Caches beleuchtet), (4) Verdeckungsrechnung für jeden Cache, um weiche Schatten zu erzeugen, (5) Interpolation der Cache-Ergebnisse auf den sichtbaren Bildausschnitt. Die Schritte 2 bis 5 werden pro Frame ausgeführt.

5.2 Reflective Shadow-Maps

In diesem Kapitel wird erläutert, wie das indirekte Licht beim LightSkin-Verfahren initial repräsentiert wird. Hierfür werden die Ansätze der Reflective Shadow-Maps [DS05] für indirekte Reflexionen und der Translucent Shadow-Maps [DS03] für Subsurface-Scattering-Effekte in einem Ansatz kombiniert. Beide Verfahren repräsentieren das indirekte Licht in Form von virtuellen Punktlichtquellen, die wegen ihrer nicht vorhandenen geometrischen Ausbreitung in bestimmten Fällen zur Artefaktbildung bei der indirekten Beleuchtung führen. Um diese Artefakte zu vermeiden, wird in dieser Arbeit, nach Vorbild von Radiosity- [GTG*84] und Light-Clustering-Verfahren (z. B. [PKD12]), das indirekte Licht durch virtuelle Flächenlichtquellen repräsentiert.

In Kapitel 5.2.1 wird zuerst erläutert, wie Reflective Shadow-Maps grundsätzlich funktionieren und warum sie in bestimmten Fällen zur Artefaktbildung neigen. Anschließend wird im nächsten Unterkapitel beschrieben, wie sich diese Artefakte durch die Verwendung von homogenen Flächenlichtquellen mit analytischen Formfaktoren für Radiosity-Verfahren reduzieren lassen. Ferner wird in Kapitel 5.2.3 gezeigt, wie sich die Prinzipien der Translucent Shadow-Maps nutzen lassen, um indirektes Licht unterhalb einer Modelloberfläche initial zu repräsentieren. Schlussendlich werden in Kapitel 5.2.4 einige wichtige Implementierungsdetails aufgeführt.

5.2.1 Prinzip

Das LightSkin-Verfahren verwendet Reflective Shadow-Maps zur initialen Verteilung der VPLs für das indirekte Licht. Diese wurden 2005 von Dachsbacher und Stamminger [DS05] vorgestellt. Es handelt sich um eine Erweiterung des klassischen Shadow-Mapping-Prinzips, bei dem aus Sicht der Lichtquelle die Szene gezeichnet wird. Anstatt eines Tiefenbilds werden drei separate Bilder (Buffer) erzeugt. Ein Buffer speichert die Positionen $\boldsymbol{x_l}$, einer die Normalen $\boldsymbol{n_l}$ und einer den reflektierten Lichtstrom ϕ_l zu den vom primären Licht sichtbaren Oberflächenpunkten. So bildet ein Pixel, gelesen von allen drei Buffern, eine hemisphärische virtuelle Punktlichtquelle. Der reflektierte Lichtstrom ϕ_l bildet für jede Farbkomponente des VPL (RGB) die Energie pro Zeiteinheit ab. Die Berechnung des reflektierten Lichtstroms kann erfolgen, indem man den diffusen Reflexionskoeffizienten des Materials mit der Intensität der primären Lichtquelle multipliziert. Die Abbildung 5.2 zeigt die drei Buffer und das grundlegende Prinzip des Reflective Shadow-Mappings.

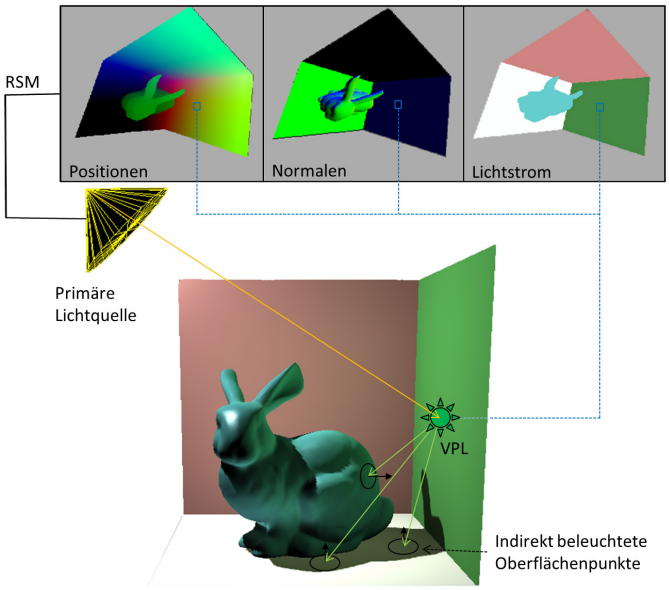

Abbildung 5.2: Prinzip des Reflective Shadow-Mappings. Jede primäre Lichtquelle speichert drei Buffer. Ein Pixel, entnommen aus den drei Buffern, bildet eine hemisphärische Punktlichtquelle (VPL), die für die indirekte Beleuchtung genutzt wird. Dabei wird nicht geprüft, ob das indirekte Licht blockiert wird.

Bei der Beleuchtung eines Oberflächenpunkts anhand eines VPL wird keine Sichtbarkeitsprüfung durchgeführt. Demnach kann das Licht nicht blockiert werden und somit keine weichen Schatten erzeugen. Die Bestimmung der Beleuchtungsstärke in einem Empfängeroberflächenpunkt $\boldsymbol{x}$ (mit der Normalen $\boldsymbol{n_x}$), der durch eine VPL beleuchtet wird, kann durch die folgende Formel berechnet werden:

$$E_l(\boldsymbol{x}, \boldsymbol{n_x}, \boldsymbol{x_l}, \boldsymbol{n_l}, \phi_l) = \frac{\phi_l \left(\boldsymbol{n_l} \cdot \frac{\boldsymbol{x}-\boldsymbol{x_l}}{\|\boldsymbol{x}-\boldsymbol{x_l}\|}\right)^+ \left(\boldsymbol{n_x} \cdot \frac{\boldsymbol{x_l}-\boldsymbol{x}}{\|\boldsymbol{x}-\boldsymbol{x_l}\|}\right)^+}{\pi \|\boldsymbol{x}-\boldsymbol{x_l}\|^2}. \tag{5.1}$$

In dieser Formel stammen $\phi_l, \boldsymbol{n_l}$ und $\boldsymbol{x_l}$ aus der RSM und bezeichnen den Lichtstrom, die Normale und die Position des VPL. Es wird davon ausgegangen, dass das VPL das Licht wie ein Lambert-Strahler verteilt. Es wirkt also wie ein diffuser Reflektor. Die komplette Beleuchtungsstärke ergibt sich dann aus der Summe über alle VPLs:

$$E(\boldsymbol{x}, \boldsymbol{n_x}) = \sum_{i=1}^{m} E_l(\boldsymbol{x}, \boldsymbol{n_x}, \boldsymbol{x_{l,i}}, \boldsymbol{n_{l,i}}, \phi_{l,i}). \tag{5.2}$$

m entspricht hier der Menge der Pixel respektive der VPLs der RSM. Diese Art der Beleuchtung hat einen elementaren Nachteil, der im folgenden Kapitel genauer betrachtet werden soll.

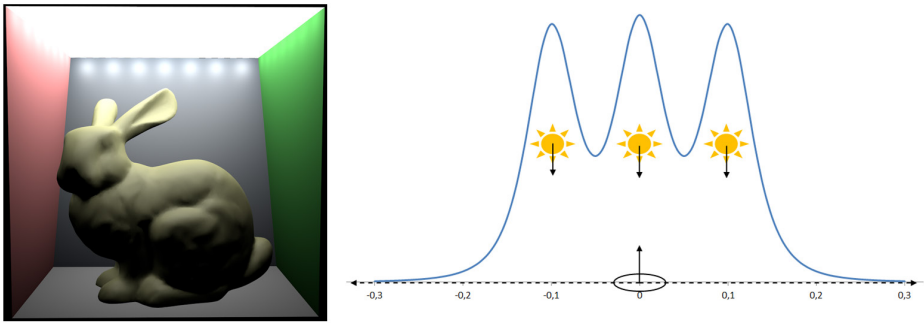

Abbildung 5.3: Artefakte, hervorgerufen durch hemisphärische Punktlichtquellen (VPLs). Im Bild links leuchtet die Deckenfläche. Die Beleuchtung des Raums durch die Fläche wird mit 49 Punktlichtquellen simuliert. Man kann deutlich die Artefakte an der oberen Deckenkante erkennen. Rechts ist der Energieverlauf für einen auf der X-Achse wandernden Oberflächenpunkt dargestellt, der durch drei VPLs beleuchtet wird. Sobald sich der Oberflächenpunkt der Lichtquelle nähert, steigt die empfangene Energie unnatürlich stark an.

5.2.2 Virtual Point-Light vs. Virtual Area-Light

Die Beleuchtung auf Basis von Punktlichtquellen ist nicht optimal. Krivanek et al. [KFB10] haben unter anderem gezeigt, dass nur durch die Verwendung von Punktlichtquellen keine realistische indirekte Beleuchtung möglich ist. Diese Aussage trifft zumindest für eine bestimmte Gruppe von spiegelnden Materialien zu. Das Problem hängt mit der nicht vorhandenen geometrischen Ausbreitung der VPLs zusammen. Bei der Betrachtung der Formel (5.1) fällt auf, dass der Term ungebunden ist. Wandert ein Empfängerpunkt näher zur Lichtquelle, strebt die Beleuchtungsstärke gegen unendlich. Dieser Sachverhalt führt zu punktförmigen Artefakten, die in Abbildung 5.3 dargestellt sind. Um dieses Problem zu beheben, wird typischerweise die resultierende Beleuchtungsstärke begrenzt [NED11]. Dies kann durch einen nicht physikalischen Korrekturwert ε geschehen:

$$E_l(\boldsymbol{x}, \boldsymbol{n_x}, \boldsymbol{x_l}, \boldsymbol{n_l}, \phi_l) = \frac{\phi_l \left(\boldsymbol{n_l} \cdot \frac{\boldsymbol{x} - \boldsymbol{x_l}}{\|\boldsymbol{x} - \boldsymbol{x_l}\|}\right)^+ \left(\boldsymbol{n_x} \cdot \frac{\boldsymbol{x_l} - \boldsymbol{x}}{\|\boldsymbol{x} - \boldsymbol{x_l}\|}\right)^+}{\pi \|\boldsymbol{x} - \boldsymbol{x_l}\|^2 + \varepsilon}. \tag{5.3}$$

Diese Begrenzung führt zu einem ungewünschten Energieverlust bei der indirekten Beleuchtung, der aber in Kauf genommen wird, um die punktförmigen Artefakte zu verringern.

Die Beschränkung auf hemisphärische Punktlichtquellen dient der Vereinfachung bzw. Beschleunigung der Berechnung der indirekten Beleuchtung. Geht man davon aus, dass der Empfängerpunkt $\boldsymbol{x}$ durch eine Flächenlichtquelle bestrahlt wird, dann ergibt sich die eintreffende Beleuchtungsstärke aus der Lösung des Integrals über die Flächenlichtquelle:

$$E_l(\boldsymbol{x}, \boldsymbol{n_x}) = \frac{1}{A} \int_{\boldsymbol{x_l} \in A_l} \frac{\phi(\boldsymbol{x_l})(\boldsymbol{n_l} \cdot (\boldsymbol{x} - \boldsymbol{x_l}))^+ (\boldsymbol{n_x} \cdot (\boldsymbol{x_l} - \boldsymbol{x}))^+}{\pi \|\boldsymbol{x} - \boldsymbol{x_l}\|^4} dA_l. \tag{5.4}$$

Auch bei dieser Gleichung wird davon ausgegangen, dass jeder Punkt auf der Oberfläche der Lichtquelle ein Lambert-Strahler ist. Die Lösung dieses Integrals ist in einer Echtzeitanwendung nicht

möglich. Zudem ist nicht offensichtlich, wie der wechselnde Lichtstrom über die Flächenlichtquelle abgebildet werden soll. Wenn man jedoch davon ausgeht, dass der Lichtstrom über die Fläche konstant ist, dann erhält man eine homogene diffuse Flächenlichtquelle. Durch das Radiosity-Verfahren [GTG*84] ist bekannt, dass sich für eine solche Lichtquelle die einfallende Beleuchtungsstärke in einem Punkt $\boldsymbol{x}$ aus dem folgenden Zusammenhang ergibt:

$$E_l(\boldsymbol{x}, \boldsymbol{n_x}) = B_l F_{x \leftarrow l}. \tag{5.5}$$

Wobei $B_l$die spezifische Lichtausstrahlung (Radiant Exitant, Radiosity) der Flächenlichtquelle bezeichnet ($B_l = \frac{\phi}{A_l}$) und $F_{x \leftarrow l}$ den Formfaktor von der Flächenlichtquelle zum Oberflächenpunkt beschreibt. Für den Formfaktor zwischen einem infinitesimal kleinen Element und einer endlichen Fläche gilt (ohne Sichtbarkeitsprüfung):

$$F_{x \leftarrow l} = \int_{A_l} \frac{cos\vartheta_l cos\vartheta_x}{\pi d^2} dA_l = \int_{\boldsymbol{x_l} \in A_l} \frac{(\boldsymbol{n_l} \cdot (\boldsymbol{x} - \boldsymbol{x_l}))^+ (\boldsymbol{n_x} \cdot (\boldsymbol{x_l} - \boldsymbol{x}))^+}{\pi \|\boldsymbol{x} - \boldsymbol{x_l}\|^4} dA_l. \tag{5.6}$$

Die Gleichungen (5.5) und (5.6) bringen formal keinen direkten Nutzen, denn es muss immer noch das Integral über die Fläche der Lichtquelle gebildet werden. Allerdings existieren für einfache Flächenformen schnelle analytische Lösungen für dieses Integral. Wenn man davon ausgeht, dass die Lichtquelle die Form einer Scheibe (Kreisfläche) hat, kann der Formfaktor durch die folgende Formel approximiert werden:

$$F_{x \leftarrow l,disk} = \frac{r^2 cos\vartheta_l cos\vartheta_x}{r^2 + \|\boldsymbol{x} - \boldsymbol{x_l}\|^2} = \frac{A_{disk} cos\vartheta_l cos\vartheta_x}{A_{disk} + \pi \|\boldsymbol{x} - \boldsymbol{x_l}\|^2}. \tag{5.7}$$

In dieser Formel ist r der Radius bzw. A_{disk} die Fläche der Scheibe. Schlussendlich ergibt sich für die Berechnung der einfallenden Beleuchtungsstärke auf einen Empfängeroberflächenpunkt anhand einer Kreisflächenlichtquelle die folgende, einfache Form:

$$E_{l,disk}(\boldsymbol{x}, \boldsymbol{n_x}, \boldsymbol{x_l}, \boldsymbol{n_l}, \phi_l, A) = \frac{\phi_l}{\pi} \frac{\left(\boldsymbol{n_l} \cdot \frac{\boldsymbol{x} - \boldsymbol{x_l}}{\|\boldsymbol{x} - \boldsymbol{x_l}\|}\right)^+ \left(\boldsymbol{n_x} \cdot \frac{\boldsymbol{x_l} - \boldsymbol{x}}{\|\boldsymbol{x_l} - \boldsymbol{x}\|}\right)^+}{A + \pi \|\boldsymbol{x} - \boldsymbol{x_l}\|^2}. \tag{5.8}$$

Diese Form ist nicht komplexer als die ursprüngliche Form aus Gleichung (5.1) für die hemisphärische Punktlichtquelle und ähnelt der Gleichung (5.3), nur dass der Korrekturwert ε jetzt durch eine konkrete physikalische Größe ersetzt wurde. Abbildung 5.4 zeigt die Beleuchtung eines wandernden Oberflächenpunkts durch Flächenlichtquellen. Es ist deutlich zu erkennen, dass das eintreffende Licht nun gleichmäßiger verteilt wird und weniger Artefakte verursacht werden.

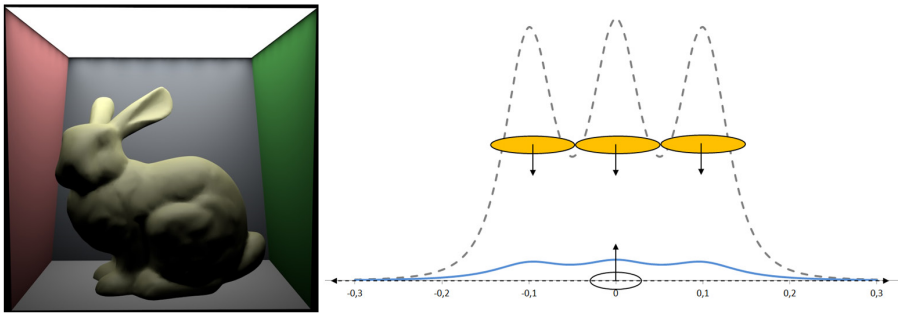

Abbildung 5.4: Beleuchtung mit Kreisflächenlichtquellen. Die Decke des Raums ist eine Flächenlichtquelle, die durch 49 Kreisflächenlichtquellen simuliert wird. Man erkennt, dass die Beleuchtung weniger Artefakte aufweist, als bei der Methode in Abbildung 5.3. Der energetische Verlauf für einen wandernden Oberflächenpunkt ist rechts dargestellt. Die blaue durchgehende Linie ist der Verlauf für drei Flächenlichtquellen und die graue gestrichelte Linie zeigt im Vergleich den Verlauf für drei VPLs.

Für das LightSkin-Verfahren stellt ein Pixel der RSM kein VPL dar, sondern ein Virtual Area-Light (VAL). Dieses VAL hat die Form einer Kreisfläche. Die Bestimmung der Fläche kann anhand der Projektionseigenschaften der primären Lichtquelle geschehen, hierauf wird in Kapitel 5.2.4 noch genauer eingegangen. Doch davor soll eine Erweiterung der RSMs vorgestellt werden, die für die spätere Berechnung des Subsurface-Scatterings wichtig ist.

5.2.3 Erweiterungen für Subsurface-Scattering

Bei dem klassischen Reflective Shadow-Mapping geht man davon aus, dass das primäre Licht direkt in dem Punkt reflektiert wird, in dem es eintrifft. Diese Annahme ist nur dann eine vertretbare Approximation, wenn das Licht auf Materialien trifft, die es nicht tief unter die Oberfläche transportieren, wie zum Beispiel Beton oder Holz. Für diese Materialien kann ein diffuses Farbspektrum bzw. ein spektraler Reflexionskoeffizient die Prozesse unter der Oberfläche des Materials in ausreichender Form abbilden und so eine entsprechende Materialfarbe produzieren. Bei Materialien wie beispielsweise Wachs oder Marmor ist dies nicht möglich, denn die wahrnehmbare Leuchtdichte eines Oberflächenpunkts ist hier zusätzlich vom Lichttransport innerhalb des Materials abhängig. Dieser Transport wurde bereits in Kapitel 2.1.6 beschrieben und wird in Kapitel 5.5 noch detaillierter erläutert. In diesem Unterkapitel ist vorerst nur von Bedeutung, dass Licht von einer primären Lichtquelle, welches auf ein Subsurface-Scattering-Material auftrifft, zum Teil in dieses Material hineingebrochen wird. Dies lässt sich in Anlehnung an Dachsbacher‘s und Stamminger’s Translucent Shadow-Maps [DS03] einfach abbilden, indem man ganz analog zu dem reflektierten Lichtstrom bei den RSMs einen zusätzlichen Buffer für die primäre Lichtquelle erzeugt, der die hineingebrochene (innere) Beleuchtungsstärke bzw. die spezifische Lichtausstrahlung unterhalb der Oberfläche widerspiegelt. Das Prinzip ist in Abbildung 5.5 dargestellt. Beim Multiple Subsurface-Scattering wird das Licht innerhalb des Materials sehr häufig gestreut, dies führt zu einem eher diffusen Transport innerhalb des Materials. Deshalb reicht die Speicherung der spezifischen Lichtausstrahlung unterhalb der Materialoberfläche aus. Diese lässt sich anhand der Daten aus den RSM-Buffern und dem Fresnel-Reflexionsterm bestimmen (siehe Kapitel 2.1.4):

$$B_s(\boldsymbol{x_l}, \boldsymbol{n_l}, \boldsymbol{\omega_i}, n) = \big(1 - F_r(\boldsymbol{n_l}, \boldsymbol{\omega_i}, n)\big)(\boldsymbol{n_l} \cdot \boldsymbol{\omega_i})^{+} E(-\boldsymbol{\omega_i}). \tag{5.9}$$

$E(-\boldsymbol{\omega_i})$ bezeichnet die Beleuchtungsstärke, die beim Oberflächenelement durch die primäre Lichtquelle aus Richtung $-\boldsymbol{\omega_i}$ (Punkt- oder Parallellichtquelle) eingeht und n den Brechungsindex des Materials. $\boldsymbol{\omega_i}$ ist ein normalisierter Vektor, der von $\boldsymbol{x_l}$ zur primären Lichtquelle zeigt.

Die Speicherung der in das Material gebrochenen Beleuchtungsstärke ist nur notwendig, wenn in der Szene entsprechende Materialien vorkommen.

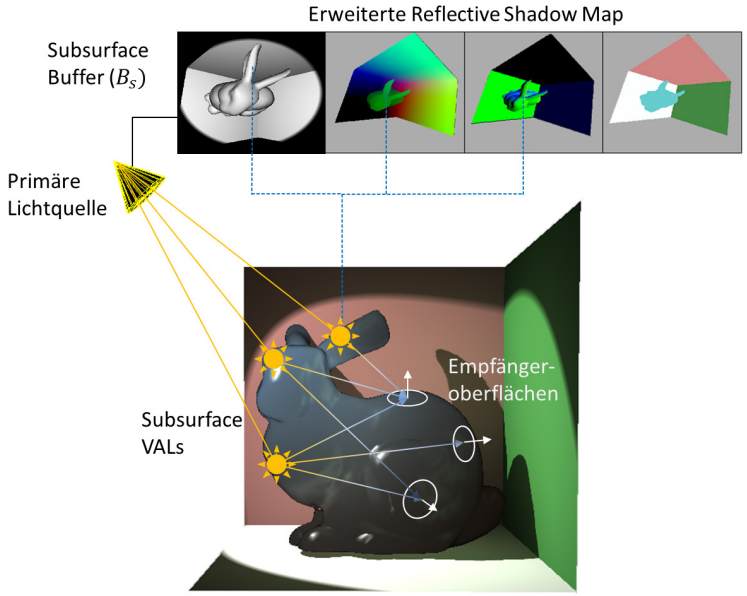

Abbildung 5.5: Erzeugung von Lichtern unter der Oberfläche. Für Subsurface-Scattering-Materialien wird die initiale Beleuchtungsstärke unterhalb der Oberfläche ebenfalls durch VALs repräsentiert (zusätzlicher RSM-Buffer). Diese VALs werden dann nur auf die Oberflächenpunkte des Subsurface-Scattering-Objekts angewendet. Die Streuung des Lichts innerhalb des Materials ist Thema von Kapitel 5.5.

5.2.4 Implementierung

Die erweiterte RSM (Kombination aus Translucent und Reflective Shadow-Map mit Flächenlichtquellen) für das LightSkin-Verfahren benötigt mehr Daten als die klassische RSM, der zusätzliche Berechnungsaufwand ist jedoch sehr gering. Die folgenden Daten müssen für jedes VAL gespeichert werden:

- Position des Oberflächenpunkts (XYZ-Koordinaten),
- Normale des Oberflächenpunkts (XYZ-Koordinaten),
- Reflektierter Lichtstrom (RGB-Komponenten),
- Hineingebrochene Beleuchtungsstärke (RGB-Komponenten),
- Fläche,
- Material-ID.

Bis auf die Material-ID sind alle Komponenten bereits erwähnt worden. Die Material-ID ist für das Subsurface-Scattering nötig, denn VALs, die unter der Oberfläche liegen, dürfen nur auf Empfängerpunkte angewendet werden, die auf dem gleichen Objekt wie die VALs liegen. Dieser Test ist durch einen einfachen Vergleich der Material-ID möglich. Da jedes VAL für die indirekte Beleuchtung auf jeden Cache (losen Oberflächenpunkt) angewendet werden muss, ist es wichtig, dass die Anzahl der Texturlesezugriffe nicht zu hoch ist. Aus diesem Grund werden die Daten so komprimiert, dass sie in drei Fließkomma-Buffer passen, wovon jeder vier Komponenten pro Pixel enthält. So ist der Leseaufwand für ein VAL nicht höher als für ein VPL bei den klassischen RSMs. Die einfache Kompression des reflektierten Lichtstroms in einem 32-Bit-Fließkommawert, wie in [DS06] vorgeschlagen, reicht bereits aus, um alle Daten in den insgesamt 12 Komponenten der Fließkomma-Buffer zu speichern.

Trotz der Komprimierung des Lichtstroms ist eine indirekte HDR-Beleuchtung möglich, denn die Akkumulation der VALs bei Empfängeroberflächen geschieht mit 32-Bit pro Farbkomponente.

Die Erzeugung der erweiterten RSM geschieht durch einen Fragment-Shader, der alle drei Render-Targets in einem Durchgang beschreibt (ab Shader-Model 3.0 möglich). Die Speicherung der Positionen und Normalen geschieht auf direktem Wege und bedarf keiner weiteren Erläuterung. Die Berechnung des reflektierten Lichtstroms und der hineingebrochenen Beleuchtungsstärke wurden bereits in Kapitel 5.2.1 und 5.2.3 erläutert. Um die Fläche eines VALs zu bestimmen, müssen die Auflösung der RSM und die verwendete Projektionsmatrix bekannt sein. Beim LightSkin-Verfahren wird ein einfacher Ansatz zur Bestimmung der Fläche angewendet. Wenn man eine perspektivische Projektion voraussetzt, kann man die Fläche mit der folgenden Formel bestimmen:

$$A_{disk} = z_{xl}^2 \frac{w_{near} h_{near}}{z_{near}^2 w_{rsm} h_{rsm}}. \tag{5.10}$$

Die Fläche des VALs entspricht der projektiven Fläche des Pixels. w_{near}, h_{near} und z_{near} bezeichnen die Breite, Höhe und Tiefe der Near Clipping-Plane, während w_{rsm} und h_{rsm} die Breite und Höhe der RSM in Pixeln sind. Die Entfernung des Oberflächenpunkts zur Kamera ist in z_{xl} gespeichert. Die Berechnung setzt voraus, dass jede Fläche senkrecht zur Projektionsrichtung der Kamera steht. Diese approximative Annahme ist erforderlich, da Flächenelemente, die steil zur Projektionsrichtung liegen, eine nahezu unendliche Ausdehnung annehmen könnten. Dies führt zu numerischen Problemen und ist wegen der punktuellen Abtastung der Fläche in einem Pixel auch nicht korrekt.

5.3 Indirekte Reflexionen

In diesem Kapitel wird beschrieben, wie die indirekten Reflexionen beim LightSkin-Verfahren berechnet werden. Die Berechnung erfolgt in zwei Schritten: Zuerst wird das indirekte Licht für die Caches berechnet, dann wird, ausgehend von den Ergebnissen in den Caches, eine durchgehende indirekte Reflexion mit einem neuen Interpolationsverfahren gebildet. Um das neue Interpolationsverfahren verstehen zu können, ist es erforderlich, das genutzte Reflexionsmodell für die Caches zu kennen. Das Reflexionsmodell soll anhand Kajiya's Rendergleichung [Kaj86] hergeleitet werden. Diese wurde bereits in Kapitel 2.1.2 vorgestellt und kann wie folgt wiedergegeben werden:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_S f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) L_o(\boldsymbol{s}, -\boldsymbol{\omega_i}) \frac{(N(\boldsymbol{s}) \cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s} - \boldsymbol{x}\|^2} V(\boldsymbol{x}, \boldsymbol{s}) (N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.11}$$

$\boldsymbol{x}$ und $N(\boldsymbol{x})$ bezeichnen hier die Position und die Normale eines Caches auf der Oberfläche eines Modells. $\boldsymbol{\omega}$ ist die Reflexionsrichtung zum Betrachter. Es wurde bereits in Kapitel 5.1 erwähnt, dass für die Ausbreitung des indirekten Lichts vorerst die Sichtbarkeit außer Acht gelassen wird. Es entfällt also der binäre Sichtbarkeitstest $V(\boldsymbol{x}, \boldsymbol{s})$ aus der Rendergleichung. Der Selbstleuchtanteil $L_e(\boldsymbol{x}, \boldsymbol{\omega})$ entfällt ebenfalls, da gegenwärtig nur Reflexionen betrachtet werden. Daraus ergibt sich die vereinfachte Form der Gleichung:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = \int_S f_r(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\omega_i}) L_o\,(\boldsymbol{s},-\boldsymbol{\omega_i}) \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} (N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.12}$$

Die Menge der reflektierenden Oberflächen S entspricht bei einem Instant-Radiosity-Ansatz mit RSMs der diskreten Menge m der virtuellen Lichter. Beim LightSkin-Verfahren hat jedes virtuelle Licht eine Ausbreitung (VAL) und somit ergibt sich:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) \approx \sum_{j=1}^{m} \int_{\boldsymbol{s}\in A_{disk,j}} f_r(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\omega_i}) L_o\,(\boldsymbol{s},-\boldsymbol{\omega_i}) \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} (N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.13}$$

Ferner handelt es sich bei den VALs um diffuse Flächenlichtquellen. Hierdurch sind zwar Kaustiken vom Lichtpfad ausgeschlossen, aber dafür lassen sich weitere Vereinfachungen durchführen. Wegen der konstanten Lichtausbreitung, unabhängig vom Betrachtungswinkel bei diffusen Flächenlichtquellen, gilt:

$$L_o(\boldsymbol{s}) = \frac{B_o(\boldsymbol{s})}{\pi}. \tag{5.14}$$

$B_o(\boldsymbol{x})$ (spezifische Ausstrahlung, Radiosity) ist anhand des Lichtstroms ϕ_l und der Fläche des VAL A_l (A_{disk}) zu bestimmen. Setzt man diese in die Gleichung (5.13) ein, so erhält man:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}}{\pi A_{l,j}} \int_{\boldsymbol{s}\in A_{l,j}} f_r(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\omega_i}) \frac{\left(\boldsymbol{n_{lj}}\cdot -\boldsymbol{\omega_i}\right)^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} (N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.15}$$

Da die empfangene Leuchtdichte vom VAL konstant ist, kann sie vor das Integral gezogen werden. Die Normale des VAL ist ebenfalls konstant, deswegen wurde $N(\boldsymbol{s})$ gegen $\boldsymbol{n_l}$ ausgetauscht.

In Gleichung (5.15) lassen sich nun beliebige BRDFs einsetzen. Normalerweise fasst eine BRDF die Reflexionseigenschaften eines Materials zusammen. Dies kann teilweise zu sehr komplexen Funktionen führen. Da das LightSkin-Verfahren jedoch nur diffuse und glänzende indirekte Reflexionen unterstützt, kann man diese BRDFs voneinander unabhängig betrachten. So ergeben sich weitere Vereinfachungen für die reflektierte Leuchtdichte. Aus diesem Grund wird das reflektierte indirekte Licht in zwei Komponenten zerlegt:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = L_{o,d}(\boldsymbol{x},\boldsymbol{\omega}) + L_{o,g}(\boldsymbol{x},\boldsymbol{\omega}). \tag{5.16}$$

Diese Zerlegung ist wegen der Superpositionseigenschaft des Lichts physikalisch korrekt, solange für $\int_{\Omega^+} L_{o,d}(\boldsymbol{x},\boldsymbol{\omega}) cos\vartheta_x d\boldsymbol{\omega} + \int_{\Omega^+} L_{o,g}(\boldsymbol{x},\boldsymbol{\omega}) cos\vartheta_x d\boldsymbol{\omega} \leq E_i(\boldsymbol{x})$ gilt. In den zwei folgenden Unterkapiteln werden nun vereinfachte Terme für die diffus ($L_{o,d}$) und glänzend ($L_{o,g}$) reflektierte Leuchtdichte gebildet.

5.3.1 Diffuse Reflexionen

Die BRDF für diffuse Reflexionen ist im eigentlichen Sinn keine gültige BRDF, denn diffuse Reflexionen kommen durch Streuungsprozesse unterhalb der Oberfläche eines Materials zustande. Diese Streuungsprozesse wurden bereits mehrmals in dieser Arbeit angesprochen und werden als Subsurface-Scattering bezeichnet. Demnach muss man eigentlich die BSSRDF für die Abbildung dieser Reflexionen verwenden. Wenn man jedoch davon ausgeht, dass das eintreffende Licht nicht beson-

ders tief in das Material eindringt, lässt sich die Reflexion durch eine einfache BRDF abbilden, denn die Streuung ist lokal so begrenzt, dass sie die Ausmaße eines Pixels im Framebuffer nicht überschreitet. Also ist eine Abbildung der Reflexion im Punkt durch eine BRDF legitim. Die Absorptionsprozesse innerhalb des Materials lassen sich in diesem Fall durch einen einfachen komponentenbasierten Reflexionskoeffizenten ρ_d abbilden. Dieser Koeffizient gibt an, welche Lichtfrequenzen vom Material absorbiert und welche reflektiert werden. Er bildet somit die „Materialfarbe“ ab. Ferner gilt für die diffuse Reflexion Proportionalität zwischen der Beleuchtungsstärke und der ausgehenden Leuchtdichte, wie sie in Gleichung (5.14) beschrieben wird.

Aus diesen Überlegungen ergibt sich für diffuse Materialien, die eine geringe geometrische Streuung unterhalb der Objektoberfläche aufweisen, die folgende BRDF:

$$f_r(\boldsymbol{x}) = \frac{\rho_d(\boldsymbol{x})}{\pi}. \tag{5.17}$$

Da diese BRDF weder vom Betrachtungswinkel, noch vom Eingangswinkel des Lichts abhängig ist, kann sie in Gleichung (5.15) vor das Integral gezogen werden. Für die reflektierte Leuchtdichte ergibt sich so:

$$L_{o,d}(\boldsymbol{x}, \boldsymbol{\omega}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_d(\boldsymbol{x})}{\pi A_{l,j}} \int_{\boldsymbol{s}\epsilon A_{l,j}} \frac{\left(\boldsymbol{n_{l,j}} \cdot -\boldsymbol{\omega_i}\right)^+}{\pi\|\boldsymbol{s}-\boldsymbol{x}\|^2} (N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.18}$$

Nun bildet das Integral einen Formfaktor von einem infinitesimal kleinen zu einem endlichen Flächenelement (vgl. Gleichung (5.6)). Da das Integral über das VAL aus der RSM gebildet wird und weil dieses VAL die Form einer Kreisfläche hat, lässt sich das Integral durch den approximativen, analytischen Formfaktor zwischen einem Punkt und einer Kreisfläche ersetzen (siehe Gleichung (5.7)). Damit ergibt sich die diffus reflektierte Leuchtdichte:

$$L_{o,d}(\boldsymbol{x}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_d(\boldsymbol{x})}{\pi A_{l,j}} \frac{A_{l,j}\left(\boldsymbol{n_{l,j}} \cdot -\boldsymbol{\omega_i}\right)^+ (N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+}{A_{l,j} + \pi\left\|\boldsymbol{x_{l,j}} - \boldsymbol{x}\right\|^2}. \tag{5.19}$$

Nun kann man noch weitere Kürzungen vornehmen und $\boldsymbol{\omega_i}$ durch Werte aus der RSM ersetzen:

$$L_{o,d}(\boldsymbol{x}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_d(\boldsymbol{x})}{\pi} \frac{\left(\boldsymbol{n_{l,j}} \cdot \frac{\boldsymbol{x}-\boldsymbol{x_{l,j}}}{\|\boldsymbol{x}-\boldsymbol{x_{l,j}}\|}\right)^+ \left(N(\boldsymbol{x}) \cdot \frac{\boldsymbol{x_{l,j}}-\boldsymbol{x}}{\|\boldsymbol{x_{l,j}}-\boldsymbol{x}\|}\right)^+}{A_{l,j} + \pi\left\|\boldsymbol{x_{l,j}} - \boldsymbol{x}\right\|^2}. \tag{5.20}$$

Die Gleichung (5.20) bildet die finale Form zur Berechnung der indirekten diffusen Reflexion beim LightSkin-Verfahren. Wenn man die Skalarprodukte gegen ihre korrespondierenden Kosinusterme ersetzt, ergibt sich die etwas kompaktere Form:

$$L_{o,d}(\boldsymbol{x}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_d(\boldsymbol{x})}{\pi} \frac{\cos^+ \vartheta_l \cos^+ \vartheta_x}{A_{l,j} + \pi\left\|\boldsymbol{x_{l,j}} - \boldsymbol{x}\right\|^2}. \tag{5.21}$$

Wobei ϑ_l den Winkel zwischen der Normalen der Flächenlichtquelle und der Strecke $\overline{\boldsymbol{x_{l,j}x}}$ und ϑ_x den Winkel zwischen der Normalen des Empfängerpunkts und der Strecke $\overline{\boldsymbol{xx_{l,j}}}$ abbildet.

5.3.2 Glänzende Reflexionen

Es existieren mehrere, empirisch entwickelte Modelle zur Simulation von glänzenden Materialien. Da es sich häufig um komplette Reflexionsmodelle handelt, eignen sich manche mehr und manche weniger, um in Form einer BRDF formuliert zu werden. Ein Modell, das sich direkt für die Verwendung als BRDF für isotrope sowie anisotrope Materialen nutzen lässt, wurde von Ward [War92] vorgestellt. Jedoch sind die Terme dieses Modells recht komplex und erscheinen im Hinblick auf die häufige Anwendung im Kontext des indirekten Lichts in einer Echtzeitanwendung als zu teuer. Schlick [Sch94] hat ein effizientes Modell entwickelt, das als Alternative zum hier gewählten Modell angesehen werden kann. Cook und Torrance [CT81] haben ein weiteres Modell vorgestellt, welches sich besonders gut im Kontext von metallischen Reflexionen eignet. Dieses Modell ist jedoch ebenfalls relativ rechenintensiv.

Die wohl bekanntesten Beleuchtungsmodelle für glänzende Reflexionen in der Echtzeit-Computergrafik sind das Phong- und das Blinn-Phong-Modell. Beide sind in ihrer ursprünglichen Form physikalisch inkorrekt. Im Folgenden soll deshalb eine abgewandelte Form des Phong-Modells [LW94, Lew94] betrachtet werden, die als physikalisch plausibel angesehen werden kann. Die hier aufgeführten Überlegungen gelten ebenfalls für das Blinn-Phong-Modell, so dass beide Modelle im LightSkin-Verfahren ohne große Veränderungen genutzt werden können.

Typischerweise wird das Phong-Reflexionsmodell wie folgt in Echtzeitanwendungen implementiert:

$$L_{phong}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = \rho_g(\boldsymbol{x}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i})(\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k}}. \tag{5.22}$$

Der Vektor $\boldsymbol{r}$ ist die Spiegelung des einfallenden Lichts $\boldsymbol{\omega_i}$ an der Oberfläche mit der Normalen $N(\boldsymbol{x})$. Das Modell ist geometrisch einfach zu interpretieren: Wandert die Spiegelungsrichtung $\boldsymbol{r}$ des Lichts näher zur Betrachtungsrichtung $\boldsymbol{\omega}$, so wird der Wert für die Reflexion durch das Skalarprodukt größer. Es bildet sich also ein Glanzpunkt. Über den Exponenten k lässt sich die Größe des Glanzpunktes steuern. Man bezeichnet dieses Modell inzwischen als *ursprüngliches* Phong-Modell. Setzt man das Modell als BRDF in die Gleichung (5.15) ein, ergibt sich:

$$f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = \frac{\rho_g(\boldsymbol{x})(\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k}}}{(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+}. \tag{5.23}$$

Der Nenner muss eingeführt werden, damit der Term $(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+$ in Gleichung (5.15) rausgekürzt wird. Durch diese Kürzung ergibt sich das ursprüngliche Phong-Modell aus Gleichung (5.22). Allerdings lässt sich argumentieren, dass die Entfernung des Terms $(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+$ aus der Rendergleichung physikalisch nicht plausibel ist [Lew94] (der Term $(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+$ resultiert aus geometrischen Zusammenhängen, die durch die BRDF nicht aufgehoben werden sollten). Deshalb kann auf die Kürzung des Terms verzichtet werden, was zu einer vereinfachten Form der BRDF führt (die Plausibilität dieser

BRDF ist in der Literatur umstritten, da sie dazu führt, dass bei flacher Lichteinstrahlung ein Spiegel schwarz wird):

$$f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = \rho_g(\boldsymbol{x})(\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k}}. \tag{5.24}$$

Diese BRDF ist ebenfalls physikalisch nicht plausibel, denn die energetische Bündelung der Reflexion durch den Exponenten k wird nicht abgebildet. Dies lässt sich anschaulich begründen, indem man annimmt, dass der Reflexionskoeffizient des Materials $\rho_g(\boldsymbol{x}) = 1$ ist. Es wird also kein Licht von dem Material absorbiert. In diesem Fall muss für die BRDF aus Energieerhaltungsgründen folgendes gelten:

$$\int_{\Omega^+} f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i})(N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^+ d\boldsymbol{\omega} = 1. \tag{5.25}$$

Setzt man die BRDF aus (5.24) in Gleichung (5.25) ein, gilt aber (siehe auch Abbildung 5.6 (links)):

$$\int_{\Omega^+} (\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k}}(N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^+ d\boldsymbol{\omega} > \int_{\Omega^+} (\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k+1}}(N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^+ d\boldsymbol{\omega}. \tag{5.26}$$

Mit zunehmendem Exponenten k nimmt die reflektierte Energie demnach unnatürlich ab. Um dem entgegenzuwirken, muss die BRDF normalisiert werden, so dass die Bedingung aus Gleichung (5.25) gilt. Dies kann durch einen Normalisierungskoeffizienten c geschehen, für den gilt:

$$c \int_{\Omega^+} (\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k}}(N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^+ d\boldsymbol{\omega} = 1. \tag{5.27}$$

Um den Koeffizienten zu bestimmen, reicht die Betrachtung einer Lösung des Integrals. Die einfachste Lösung ergibt sich für den Fall, in dem das Licht senkrecht zur Materialoberfläche einfällt, denn für diesen Fall gilt $\boldsymbol{r} = N(\boldsymbol{x})$. Der Spiegelvektor entspricht also der Normalen und damit gilt wiederum $(\boldsymbol{r} \cdot \boldsymbol{\omega}) = (N(\boldsymbol{x}) \cdot \boldsymbol{\omega})$. Nun kann man eine Lösung für das Integral berechnen:

$$
\begin{aligned}
&\int_{\Omega^+} (N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^k (N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^+ d\boldsymbol{\omega} \\
&= \int_{\Omega^+} cos\vartheta^k cos\vartheta d\boldsymbol{\omega} = \int_{\Omega^+} cos\vartheta^{k+1} d\boldsymbol{\omega} \\
&= \int_0^{2\pi} \int_0^{\frac{\pi}{2}} cos\vartheta^{k+1} sin\vartheta d\vartheta d\varphi \\
&= 2\pi \int_0^{\frac{\pi}{2}} cos\vartheta^{k+1} sin\vartheta d\vartheta.
\end{aligned}
\tag{5.28}
$$

Dieses Integral lässt sich durch die partielle Integration lösen. Es gilt $\int_a^b f'(x)g(x)dx = [f(x)g(x)]_a^b - \int_a^b f(x)g'(x)dx$, $f'(x) = sin\vartheta$ und $g(x) = cos\vartheta^{k+1}$. Wendet man diese an, ergibt das Integral:

$$
\begin{aligned}
&2\pi \int_0^{\frac{\pi}{2}} cos\vartheta^{k+1} sin\vartheta d\vartheta \\
&= 2\pi \left(\left[(-cos\vartheta) \left(\cos^{k+1} \vartheta \right) \right]_0^{\frac{\pi}{2}} - \int_0^{\frac{\pi}{2}} (-cos\vartheta)(k+1) \cos^k \vartheta \, (-sin\vartheta) d\vartheta \right) \\
&= 2\pi \left(1 - (k+1) \int_0^{\frac{\pi}{2}} \cos^{k+1} \vartheta \, sin\vartheta d\vartheta \right) \\
&= 2\pi - (k+1) 2\pi \int_0^{\frac{\pi}{2}} \cos^{k+1} \vartheta \, sin\vartheta d\vartheta \\
&= \frac{2\pi}{k+2}.
\end{aligned}
\tag{5.29}
$$

Damit steht fest, dass $\int_{\Omega^+} (\boldsymbol{r} \cdot \boldsymbol{\omega})^{\boldsymbol{k}} (N(\boldsymbol{x}) \cdot \boldsymbol{\omega})^+ d\boldsymbol{\omega}$ für $\boldsymbol{\omega_i} = -N(\boldsymbol{x})$ zu $\frac{2\pi}{k+2}$ evaluiert. Setzt man dieses Ergebnis in die Gleichung (5.27) ein, ergibt sich ein Normalisierungskoeffizient von $c = \frac{k+2}{2\pi}$. Somit ist die physikalisch plausible BRDF für glänzende Reflexionen wie folgt definiert (siehe auch Abbildung 5.6 (rechts)):

$$f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) = \rho_g(\boldsymbol{x}) \frac{k+2}{2\pi} (\boldsymbol{r} \cdot \boldsymbol{\omega})^k. \tag{5.30}$$

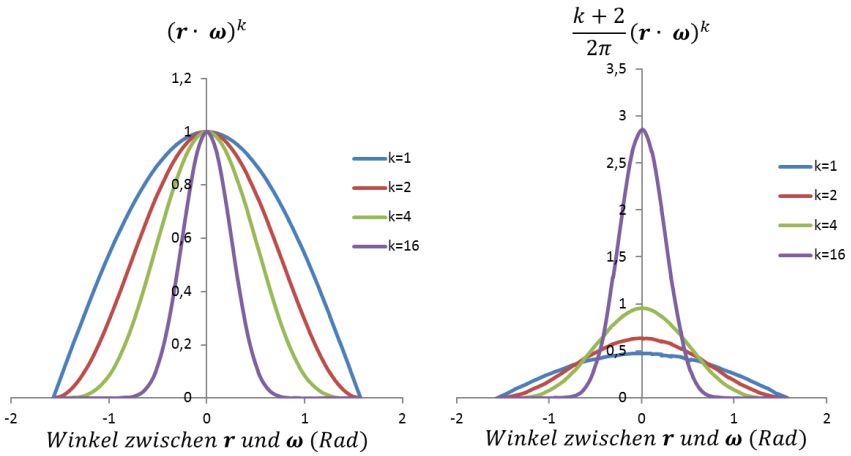

Abbildung 5.6: Vergleich der ursprünglichen und der normalisierten Phong-Reflexion (Querschnitt). Ohne Normalisierung (links) nimmt die Energie der Reflexion mit steigendem k unnatürlich ab. Durch die Normalisierung (rechts) bleibt die Energie konstant.

Setzt man diese BRDF ein, so ergibt sich für die indirekt reflektierte Leuchtdichte bei glänzenden Oberflächen:

$$L_{o,g}(\boldsymbol{x}, \boldsymbol{\omega}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}}{\pi A_{l,j}} \int_{\boldsymbol{s} \epsilon A_{l,j}} \rho_g(\boldsymbol{x}) \frac{k+2}{2\pi} (\boldsymbol{r} \cdot \boldsymbol{\omega})^k \frac{(\boldsymbol{n_{l,j}} \cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s} - \boldsymbol{x}\|^2} (N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.31}$$

Die glänzende BRDF ist im Gegensatz zur diffusen BRDF jedoch abhängig von $\boldsymbol{\omega_i}$ und kann deshalb nicht vor das Integral gezogen werden. Wenn man jedoch voraussetzt, dass die Fläche des VAL klein ist oder das VAL einen ausreichend hohen Abstand hat, kann die BRDF über das Integral der Fläche als nahezu konstant angesehen werden (siehe Abbildung 5.7). Für diesen Fall lässt sich die BRDF vor das Integral stellen, dadurch kann das Integral wieder, wie bereits bei der diffusen indirekten Reflexion, als Formfaktor zwischen einem Punkt und einer Kreisscheibe formuliert werden (siehe auch Gleichung (5.20)):

$$L_{o,g}(\boldsymbol{x}, \boldsymbol{\omega}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j} \rho_g(\boldsymbol{x})(k+2)(\boldsymbol{r} \cdot \boldsymbol{\omega})^k}{2\pi} \frac{\left(\boldsymbol{n_{l,j}} \cdot \frac{\boldsymbol{x} - \boldsymbol{x_{l,j}}}{\|\boldsymbol{x} - \boldsymbol{x_{l,j}}\|}\right)^+ \left(N(\boldsymbol{x}) \cdot \frac{\boldsymbol{x_{l,j}} - \boldsymbol{x}}{\|\boldsymbol{x_{l,j}} - \boldsymbol{x}\|}\right)^+}{A_{l,j} + \pi \|\boldsymbol{x_{l,j}} - \boldsymbol{x}\|^2}. \tag{5.32}$$

Diese Approximation ist nur dann annährend korrekt, wenn k nicht zu groß gewählt wird und der projizierte Raumwinkel der Flächenlichtquelle ebenfalls keine zu große Abbildung besitzt (siehe Abbildung 5.7). Es ist zu beachten, dass $\boldsymbol{r}$ jetzt die Spiegelung des Vektors $\frac{\boldsymbol{x_{l,j}} - \boldsymbol{x}}{\|\boldsymbol{x_{l,j}} - \boldsymbol{x}\|}$ darstellt. In der etwas kompakteren Kosinusschreibweise ergibt sich dann (der Winkel zwischen $\boldsymbol{r}$ und $\boldsymbol{\omega}$ ist mit ϑ_e notiert):

$$L_{o,g}(\boldsymbol{x},\boldsymbol{\omega}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_g(\boldsymbol{x})(k+2)\cos^k\vartheta_e}{2\pi} \frac{\cos^+\vartheta_l \cos^+\vartheta_x}{A_{l,j}+\pi\left\|\boldsymbol{x_{l,j}}-\boldsymbol{x}\right\|^2}. \tag{5.33}$$

Damit ist die Leuchtdichte für glänzend reflektierende Caches bestimmt. Nachfolgend wird nun das Interpolationsschema des LightSkin-Verfahrens erläutert.

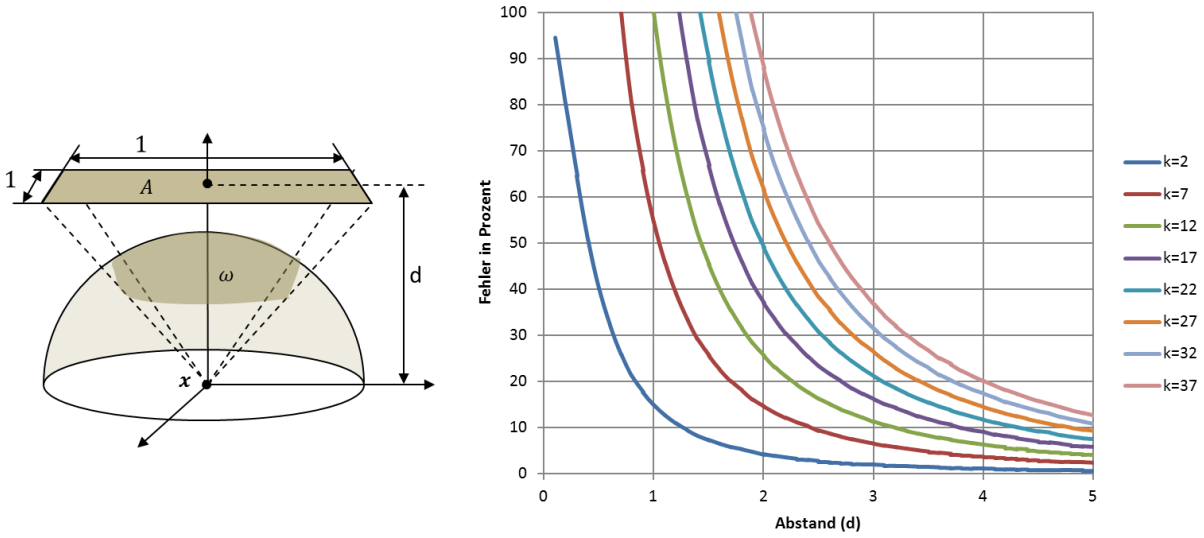

Abbildung 5.7: Fehlerbetrachtung für glänzende indirekte Reflexionen. Durch die Annahme, dass die BRDF über das Integral des VALs konstant ist, ergibt sich ein Fehler, wenn die Projektion des VALs einen weiten Raumwinkel einnimmt bzw. wenn der Reflexionsexponent groß gewählt wird (linke Abbildung). Das Diagramm auf der rechten Seite zeigt, wie sich der Fehler für unterschiedliche Reflexionsexponenten entwickelt, wenn sich die Fläche A des VALs vom Oberflächenpunkt $\boldsymbol{x}$ entfernt und sich somit der projizierte Raumwinkel verringert (der Raumwinkel verringert sich proportional zu $1/d^2$). Für die Berechnung des Fehlers wird die korrekte Lösung $L = \int_{s\epsilon A_{l,j}} \frac{k+2}{2\pi}(\boldsymbol{r}\cdot\boldsymbol{\omega})^k G(\boldsymbol{x},\boldsymbol{s})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}$ mit der approximativen Lösung $L' = \frac{k+2}{2\pi}(\boldsymbol{r}\cdot\boldsymbol{\omega})^k \int_{s\epsilon A_{l,j}} G(\boldsymbol{x},\boldsymbol{s})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}$ verglichen. Der prozentuale Fehler ergibt sich aus der folgenden Gleichung: $Fehler = 100\cdot\frac{|L-L'|}{L}$. Die Kurven zeigen, dass der Fehler unter 10% fällt, wenn $k < 30$ ist und die Fläche A um ca. das fünffache ihrer Seitenlänge vom Oberflächenpunkt $\boldsymbol{x}$ entfernt liegt. Für sehr nahe Flächen bzw. VALs ist der Fehler dagegen sehr groß.

5.3.3 Interpolation: Grundsätzliche Anforderungen

In den zwei vorhergehenden Unterkapiteln wurde beschrieben, wie die indirekt reflektierte Leuchtdichte für lose verteilte Caches berechnet werden kann. Für die Berechnung der Reflexion wurde jedes VAL auf jeden Cache angewendet. Die Anwendung jedes VALs auf jeden sichtbaren Oberflächenpunkt ist dagegen, selbst nach den vorgestellten Vereinfachungen zur Berechnung der indirekten Reflexionen, nicht in Echtzeit möglich. Demnach muss ein effizienteres Verfahren entwickelt werden, um die fehlenden Reflexionen approximativ zu ergänzen. Hierfür eignet sich ein Interpolationsverfahren. Idealerweise erfüllt dieses Interpolationsverfahren eine Reihe von Anforderungen, die sich aus Qualitäts- und Geschwindigkeitsüberlegungen ableiten.

Die Anforderungen, die sich aus dem Echtzeitverhalten der Anwendung ergeben, sind:

1. geringe Anzahl von Stützpunkten (Caches),
2. schnelle Erzeugung von Stützwerten,
3. geringer Speicheraufwand für Stützwerte,
4. schnelle Interpolation auf Basis der Stützpunkte,

5. Implementierbarkeit auf GPU.

Zusätzlich sollte die Interpolationsmethode auch gewisse Qualitätsmerkmale berücksichtigen:

1. plausible Reflexionsgradienten,
2. Erhaltung von Oberflächendetails,
3. Unterstützung von betrachtungsabhängigen Reflexionen (glänzende Reflexionen),
4. hochfrequente und niederfrequente Reflexionen,
5. temporäre Kohärenz bei Animationen.

Als weitere „weiche“ Anforderung kann eine geringe Komplexität bei der Implementierung des Verfahrens aufgeführt werden. Diese sollte zwar nicht als allgemeiner Anspruch verstanden werden, es sollte aber deutlich sein, dass sich eine hohe Komplexität bei der Verbreitung des Verfahrens in praktischen Anwendungsfeldern eher hinderlich auswirkt.

Die Interpolation von Reflexionen ist kein neues Thema in der Computergrafik. In dieser Arbeit wurde bereits erwähnt, dass Irradiance-Caching (IC) und Radiance-Caching (RC) gängige Methoden zur Berechnung globaler Beleuchtung sind. Jedoch existiert nach aktuellem Kenntnisstand kein Verfahren, welches diese Methoden so einsetzt, dass die oben beschriebenen Anforderungen für die Interpolation erfüllt werden. Dies wird nachfolgend kurz diskutiert.

Bei dem klassischen IC [WRC88] wird für jeden Cache $\boldsymbol{c}$ die eingehende Beleuchtungsstärke $E(\boldsymbol{c})$ als spektraler Wert gespeichert, der keine Richtungsinformationen aufweist. Diese Beleuchtungsstärke wird über die sichtbaren Oberflächenelemente nach dem folgenden Schema interpoliert:

$$E(\boldsymbol{x}) \approx \frac{\sum_{\boldsymbol{c_j} \in N} w_j(\boldsymbol{x}, \boldsymbol{c_j}) E(\boldsymbol{c_j})}{\sum_{c_j \in N} w_j(\boldsymbol{x}, \boldsymbol{c_j})}. \tag{5.34}$$

Die Funktion $w_j(\boldsymbol{x}, \boldsymbol{c_j})$ gibt die Gewichtung eines Caches $\boldsymbol{c_j}$ zum betrachteten Oberflächenpunkt $\boldsymbol{x}$ an**.** Die Menge N enthält nur Caches, für die $w_j(\boldsymbol{x}, \boldsymbol{c_j}) > \varepsilon$ gilt, also werden nur Caches betrachtet, die ein gewisses Minimalgewicht zum Oberflächenpunkt haben. Für die momentane Diskussion ist die genaue Funktionsweise der Gewichtungsfunktion nicht weiter wichtig, es sei jedoch erwähnt, dass sie im umgekehrt proportionalen Verhältnis zum Abstand und zur Normalendifferenz zwischen dem Oberflächenpunkt $\boldsymbol{x}$ und dem Cache $\boldsymbol{c_j}$ steht. Hat man die Beleuchtungsstärke für den Punkt $\boldsymbol{x}$ berechnet, kann man die reflektierte Leuchtdichte durch die Multiplikation mit der ideal diffusen BRDF, wie sie in Kapitel 5.3.1 vorgestellt wurde, berechnen. Dadurch ist für jeden interpolierten Wert ein individueller diffuser Reflexionskoeffizient ρ_d anwendbar. Dieser kann die gegebene Beleuchtungsstärke jedoch nur skalieren. Variationen der Normalen, sowie des Betrachtungswinkels werden bei der Interpolation nicht berücksichtigt (siehe Abbildung 5.8). Damit verletzt das IC die oben formulierten Anforderungen zur Erhaltung der Oberflächendetails, sowie die Unterstützung von betrachtungsabhängigen nieder- und hochfrequenten Reflexionen. Die Berücksichtigung der Oberflächendetails lässt sich erreichen, indem mehr Caches verwendet werden, jedoch führt dies wiederum zur Verletzung von Laufzeitanforderungen (geringe Anzahl von Stützpunkten).

Zur Verbesserung der Interpolationsergebnisse haben Ward und Heckbert [WH92] vorgeschlagen, die Gradienten für den Beleuchtungsstärkeverlauf im Cache für die Interpolation zu berücksichtigen:

$$E(\boldsymbol{x}) \approx \frac{\sum_{\boldsymbol{c_j} \in N} w_j(\boldsymbol{x}, \boldsymbol{c_j}) \left(E(\boldsymbol{c_j}) + \left(N(\boldsymbol{x}) \times N(\boldsymbol{c_j}) \right) \cdot \nabla_{r,j} E(\boldsymbol{c_j}) + (\boldsymbol{x} - \boldsymbol{c_j}) \cdot \nabla_{t,j} E(\boldsymbol{c_j}) \right)}{\sum_{\boldsymbol{c_j} \in N} w_j(\boldsymbol{x}, \boldsymbol{c_j})}. \quad (5.35)$$

Dabei wird der Gradient in einen Rotationsanteil $\nabla_{r,j}$ und einen Translationsanteil $\nabla_{t,j}$ zerlegt. Zwar erscheint die Interpolation durch die Verwendung dieser Gradienten glatter, jedoch bleiben die Probleme der Nicht-Berücksichtigung geometrischer Variationen in $\boldsymbol{x}$ bestehen. Prinzipiell erscheint das IC also, wegen der oben formulierten Qualitätsanforderungen, eher ungeeignet.

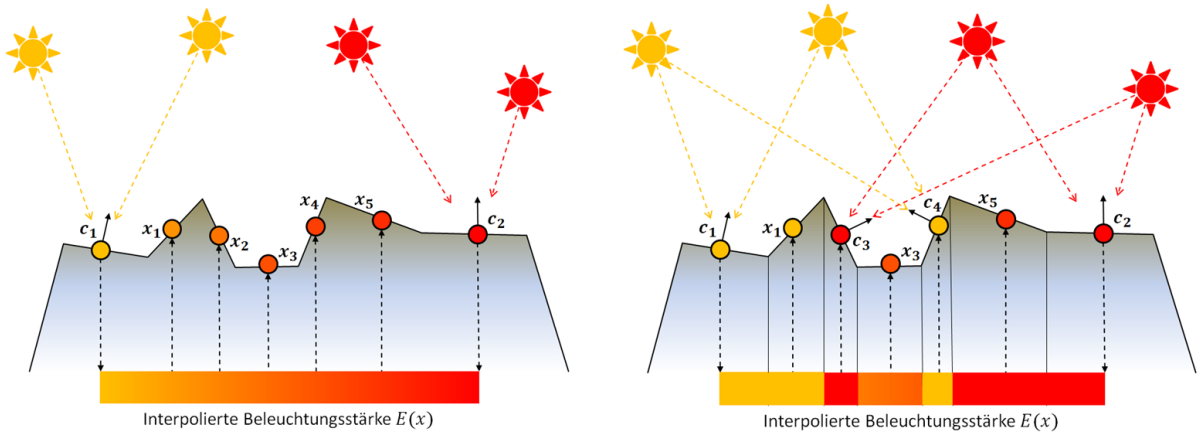

Abbildung 5.8: Interpolationsprinzip beim Irradiance-Caching. Es wird lediglich das Farbspektrum der Beleuchtungsstärke interpoliert. Werden zu wenig Caches verwendet, so werden geometrische Details (Position & Normale) für die Interpolation nicht ausreichend berücksichtigt (links). Die Berücksichtigung dieser Details kann nur durch Hinzunahme weiterer Caches geschehen (rechts).

Eine Alternative zum IC bietet das Radiance-Caching [KGP*05]. Beim Radiance-Caching wird pro Cache $\boldsymbol{c}$ anstatt der eingehenden Beleuchtungsstärke $E(\boldsymbol{c})$ die Leuchtdichte $L_i(\boldsymbol{c}, \boldsymbol{\omega_i})$ gespeichert. Da für jeden differentiellen Raumwinkel $\boldsymbol{\omega_i}$ die eintreffende Leuchtdichte bekannt sein sollte, sind die Speicheranforderungen pro Cache beim RC höher als beim IC. Um die Leuchtdichte über die Hemisphäre eines Caches kompakt und effizient zu repräsentieren, können Hemispherical-Harmonics-Koeffizienten (HSH) [GKP*04] genutzt werden, die von Spherical Harmonics (siehe Kapitel 3.1.3) abgeleitet sind. Hierdurch lässt sich die Verteilung der eintreffenden Leuchtdichte über eine moderate Anzahl von Koeffizienten im Cache abbilden: $L_i(\boldsymbol{c}, \boldsymbol{\omega_i}) \approx \sum_{j=0}^{n^2} c_{\boldsymbol{c},j} y_j(\boldsymbol{\omega_i})$. Wie diese Koeffizienten prinzipiell berechnet werden können, wurde bereits in Kapitel 3.1.3 beschrieben. Hier ist vorerst nur wichtig, dass jeder Cache das eingehende Licht durch n^2 HSH-Koeffizienten abbildet. Diese Koeffizienten müssen dann für alle sichtbaren Oberflächenpunkte $\boldsymbol{x}$ interpoliert werden. Dies kann durch die folgende Formel ausgedrückt werden [Jar08]:

$$C_{\boldsymbol{x}} \approx \frac{\sum_{\boldsymbol{c_j} \in N} w_j(\boldsymbol{x}, \boldsymbol{c_j}) R_j \left(C_j + \frac{\partial C_j}{\partial x} dx + \frac{\partial C_j}{\partial y} dy \right)}{\sum_{\boldsymbol{c_j} \in N} w_j(\boldsymbol{x}, \boldsymbol{c_j})}. \quad (5.36)$$

C_j beinhaltet alle HSH-Koeffizienten zum Cache $\boldsymbol{c_j}$, es gilt $C_j = \{c_{\boldsymbol{c},0}, c_{\boldsymbol{c},1}, \dots, c_{\boldsymbol{c},n^2-1}\}$. Die HSH-Koeffizienten C_j für den Oberflächenpunkt $\boldsymbol{x}$ müssen nicht nur interpoliert, sondern auch transformiert werden. Hierfür müssen zum einen die Translation der Positionen von $\boldsymbol{x} - \boldsymbol{c_j}$ und zum anderen die Rotation der Hemisphäre berücksichtigt werden. Um die Translation abzubilden, werden für die HSH-

Koeffizienten, wie bereits beim Irradiance-Caching, zwei Gradienten berechnet ($\frac{\partial c_j}{\partial x}$ und $\frac{\partial c_j}{\partial y}$). Diese Gradienten erlauben die Berücksichtigung einer moderaten Verschiebung des Empfängerpunkts (dx und dy liegen in der Ebene orthogonal zur Normalen des Caches und ergeben sich aus der projizierten Differenz zwischen $\boldsymbol{x}$ und $\boldsymbol{c_j}$). Die Rotation der Koeffizienten wird durch die Funktion R_j bewerkstelligt, die die Koeffizienten in ein gemeinsames Koordinatensystem transformiert.

Hat man die Koeffizienten für einen sichtbaren Oberflächenpunkt berechnet, müssen diese mit den Koeffizienten der Übertragungsfunktion $k_{\boldsymbol{x},j}(\boldsymbol{\omega}) = \int_\Omega f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i})(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ y_j(\boldsymbol{\omega_i}) d\boldsymbol{\omega_i}$ des Punkts multipliziert werden, um die resultierende Leuchtdichte zum Betrachter zu erhalten (siehe Kapitel 3.1.3):

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) \approx \sum_{j=0}^{n^2} k_{\boldsymbol{x},j}(\boldsymbol{\omega}) c_{\boldsymbol{x},j}. \tag{5.37}$$

Das grundlegende Prinzip der Interpolation auf Basis des RC ist in Abbildung 5.9 dargestellt. Im Gegensatz zum IC lassen sich mit dem RC betrachtungsabhängige Reflexionen für glänzende Materialien abbilden. Jedoch benötigen glänzende Reflexionen sehr viele HSH-Koeffizienten, typischerweise um die 200 [NRH03,GKB*05]. Jeder Koeffizient ist eine spektrale Größe, die sich über RGB-Komponenten beschreiben lässt. Man muss also für jeden sichtbaren Oberflächenpunkt 200 Lesezugriffe in RGB-Texturen vollziehen, um die Leuchtdichte $L_o(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i})$ zu bestimmen. Hier wurden die Koeffizienten der Übertragungsfunktion $k_{\boldsymbol{x},j}(\boldsymbol{\omega})$ noch nicht berücksichtigt, welche zu weiteren Lesezugriffen führen. Darüber hinaus müssen die Koeffizienten für die Übertragungsfunktion bei Animationen neu berechnet werden, wodurch sich dynamische Szenen überproportional auf die Laufzeit der Simulation auswirken. Schlussendlich lässt sich festhalten, dass das RC zwar qualitativ bessere Ergebnisse für die Interpolation liefert, dies jedoch auf Kosten der Berechnungszeit geschieht. Scherzer et al. [SNR*12] haben kürzlich einen interaktiven RC-Ansatz vorgestellt, der ohne SH-Koeffizienten auskommt, jedoch beschränkt sich dieses Verfahren ebenfalls auf statische Szenen.

Zusammenfassend kommt man zu dem Schluss, dass weder das IC noch das RC ein geeignetes Interpolationsverfahren für Echtzeitanwendungen bereithält. Bei dem IC sind die qualitativen Einschränkungen zu hoch, während bei dem RC die Berechnungszeit einen einschränkenden Faktor bildet. Bei Animationen erscheinen beide Ansätze ungeeignet, denn diese führen durch die adaptive Punktverteilung im Bildraum zu Artefakten oder zu einer höheren Berechnungszeit. Zwar haben Gautron et al. [GBP08] mit ihrem Temporal-Radiance-Caching-Ansatz ein bildraumbasiertes Verfahren vorgestellt, welches sowohl die Berechnungszeit, als auch die Artefakte bei Animationen verringert, allerdings setzt dieser Ansatz voraus, dass die komplette Animationssequenz im Vorhinein bekannt ist. Für solche statischen Animationssequenzen lassen sich spezielle zeitbasierte Gradienten berechnen, um die Beleuchtung in konsekutiven Bildern kohärenter abzubilden. Dynamische bzw. interaktive Animationen, wie sie für eine Virtual-Reality-Anwendung typisch sind, können mit diesem Verfahren aber nicht abgebildet werden.

Das in dieser Arbeit entwickelte neue Interpolationsverfahren erfüllt die oben beschriebenen Anforderungen, indem es einen neuen Ansatz zur Repräsentation der eingehenden Leuchtdichte durch Proxy-Lichtquellen vorstellt. Der Daten- und Berechnungsaufwand von den Stützpunkten/-werten ist

sehr gering. Auch die Berechnung von Interpolationsergebnissen kann sehr effizient erfolgen. Konzeptionell lässt sich das Verfahren als RC-Ansatz einordnen, jedoch werden auch Ideen aus dem IC entliehen.

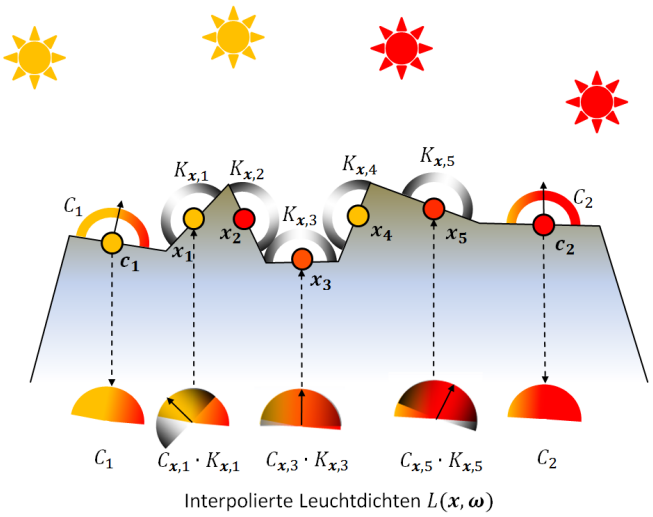

Abbildung 5.9: Interpolationsprinzip beim Radiance-Caching. Für jeden Cache $\boldsymbol{c}$ wird die einkehrende Leuchtdichte in Form von HSH-Koeffizienten (C) abgebildet. K_x bezeichnet die Koeffizienten der Übertragungsfunktion der sichtbaren Oberflächenpunkte $\boldsymbol{x}$. Das Skalarprodukt der Koeffizienten liefert die interpolierte Leuchtdichte zum Betrachter.

5.3.4 Interpolation: Erzeugung von Proxy-Lichtquellen

Das grundlegende Interpolationsprinzip des LightSkin-Verfahrens lässt sich einfach erklären: Jeder Cache erzeugt anhand des indirekten Lichts, das auf ihn einwirkt, genau eine Lichtquelle, die das komplette indirekte Licht im Cache repräsentiert. Diese Lichtquelle steht also stellvertretend für alle indirekten VALs, die auf den Cache wirken, weswegen sie nachfolgend als Proxy-Lichtquelle bezeichnet wird (siehe Abbildung 5.10). Die Proxy-Lichtquelle kann prinzipiell beliebig gestaltet werden (Punkt-, Flächen- oder Volumenlichtquelle), beim LightSkin-Verfahren wird jedoch von Punktlichtquellen ausgegangen, da diese besonders einfach zu interpretieren sind und nur sehr wenige Parameter benötigen (Position und Lichtstrom). Dies kommt der Forderung nach, dass die Stützwerte für die Interpolation geringe Speicheranforderungen aufweisen sollen. Für jeden Cache $\boldsymbol{c}$ gilt für die Proxy-Lichtquelle der folgende Zusammenhang:

$$\sum_{j=1}^{m} \frac{\phi_{l,j}}{\pi A_{l,j}} \int_{s \epsilon A_{l,j}} f_r(\boldsymbol{c}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) \frac{\left(\boldsymbol{n_{l,j}} \cdot -\boldsymbol{\omega_i}\right)^+}{\|\boldsymbol{s} - \boldsymbol{c}\|^2} (N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{s} = f_r(\boldsymbol{c}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) \frac{\phi_p}{4\pi} \frac{(N(\boldsymbol{c}) \cdot \boldsymbol{\omega_i})^+}{\|\boldsymbol{x_p} - \boldsymbol{c}\|^2}, \tag{5.38}$$

$$L_{o,VAL}(\boldsymbol{c}, \boldsymbol{\omega}) = L_{o,PROXY}(\boldsymbol{c}, \boldsymbol{\omega}).$$

Der Term links vom Gleichheitszeichen repräsentiert die reflektierte Leuchtdichte basierend auf der Beleuchtung der VALs, welcher zu Beginn des Kapitels 5.3 hergeleitet wurde, während der rechte Term die reflektierte Leuchtdichte auf Basis der Proxy-Punktlichtquelle darstellt. Die Parameter der Punktlichtquelle sind der Lichtstrom ϕ_p und die Position $\boldsymbol{x_p}$. Die Gleichsetzung in (5.38) zeigt, dass die Position und der Lichtstrom der Proxy-Lichtquelle so gewählt werden müssen, dass sie die gleiche indirekte Reflexion erzeugen wie die Anwendung sämtlicher VALs. Demnach gilt es, ϕ_p und $\boldsymbol{x_p}$ zu bestimmen.

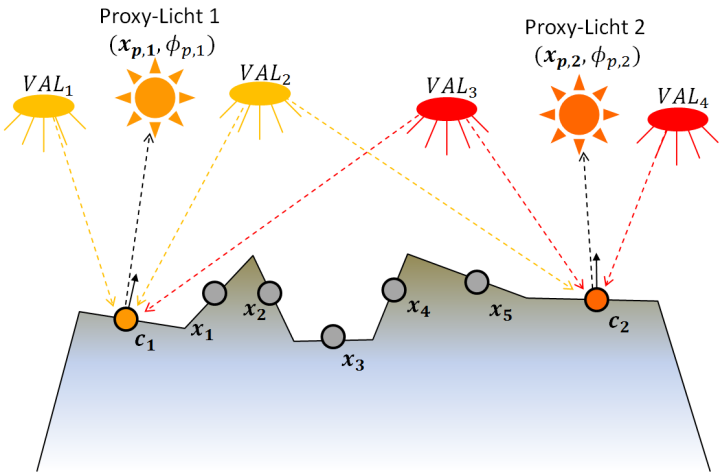

Abbildung 5.10: Prinzip der Proxy-Lichter. Die Proxy-Lichter werden auf Basis der einwirkenden VALs erzeugt. Jeder Cache erzeugt ein individuelles Proxy-Punktlicht.

Der Lichtstrom der Proxy-Lichtquelle ϕ_p lässt sich einfach durch einen Normalisierungsschritt bestimmen, wenn die Position $\boldsymbol{x_p}$ der Lichtquelle bekannt ist:

$$\phi_p = \frac{4\pi L_{o,VAL}(\boldsymbol{c}, \boldsymbol{\omega}) \left\| \boldsymbol{x_p} - \boldsymbol{c} \right\|^2}{f_r(\boldsymbol{c}, \boldsymbol{\omega}, \boldsymbol{\omega_i})(N(\boldsymbol{c}) \cdot \boldsymbol{\omega_i})^+}. \tag{5.39}$$

Die Berechnung der Position $\boldsymbol{x_p}$ kann anhand der Positionen $\boldsymbol{x_l}$ der VALs geschehen, die auf den Cache scheinen. Eine einfache Mittelwertbildung der Positionen ist jedoch nicht zielführend, denn diese hätte zur Folge, dass jeder Cache die gleiche Position für das Proxy-Licht berechnet, weil jedes VAL auf jeden Cache angewendet wird. Es muss also eine Gewichtung der VALs gefunden werden, die individuell für jeden Cache ist:

$$\boldsymbol{x_p} = \frac{\sum_{j=0}^{m} w_j(\boldsymbol{c}, \boldsymbol{x_l}) \boldsymbol{x_{l,j}}}{\sum_{j=0}^{m} w_j(\boldsymbol{c}, \boldsymbol{x_{l,j}})}. \tag{5.40}$$

Ein sinnvolles Maß für die Gewichtung ist die individuelle Beleuchtungsstärke eines VALs auf den Cache. So werden VAL-Positionen höher gewichtet, wenn sie einen großen Anteil zur Beleuchtung des Caches beitragen und geringer, wenn sie nur einen kleinen Teil beitragen. Hierdurch wird die Position der Proxy-Lichtquelle abhängig von den geometrischen Eigenschaften des Caches, denn die Position und Normale des Caches, sowie die Position, Normale und Fläche des VALs wirken sich auf die Beleuchtungsstärke aus. Somit ist die Proxy-Lichtquelle im strengen Sinne nur für den individuellen Cache gültig. Später wird sich aber noch zeigen, dass diese Proxy-Lichtquelle ebenfalls sinnvoll auf Oberflächenpunkte übernommen werden kann, die „ähnliche“ geometrische Eigenschaften wie der Cache aufweisen.

Am Anfang des Kapitel 5.3 wurde bereits beschrieben, dass beim LightSkin-Verfahren die reflektierte Leuchtdichte in einen diffusen und einen glänzenden Anteil zerlegt wird (vgl. Gleichung (5.16)). Dies erlaubt eine separate Berechnung der Proxy-Lichtquellen für diffuse und glänzende Oberflächen. Dadurch erhält man eine betrachtungsunabhängige Proxy-Lichtquelle für diffuse Reflexionen und eine betrachtungsabhängige für glänzende Reflexionen. Nachfolgend soll zuerst die Gewichtungs-

funktion für die diffusen Reflexionen bestimmt werden. Hierfür muss die Berechnung der diffusen Leuchtdichte erneut betrachtet werden (siehe auch Kapitel 5.3.1):

$$L_{o,d}(\boldsymbol{c}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_d(\boldsymbol{c})}{\pi} \frac{\cos^+\vartheta_l \cos^+\vartheta_x}{A_{l,j} + \pi\left\|\boldsymbol{x_{l,j}} - \boldsymbol{c}\right\|^2} = \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_d(\boldsymbol{c})}{\pi A_{l,j}} F_{c\leftarrow l,disk}. \tag{5.41}$$

Die Gleichung wurde so konstruiert, dass der Formfaktor von einem Oberflächenpunkt zu einer Kreisscheibe ($F_{c\leftarrow l,disk}$) zur Berechnung der Leuchtdichte genutzt werden kann. Dieser Formfaktor soll als Gewichtungsfunktion für die Proxy-Lichtquelle verwendet werden, denn er ist ein direktes Maß dafür, wie viel Licht von dem VAL beim Cache ankommt. Die Gewichtungsfunktion für die diffuse Proxy-Lichtquellenposition ist nun die folgende:

$$w_{j,d}(\boldsymbol{c}) = \varepsilon_p + \frac{A_{l,j}\cos^+\vartheta_l \cos^+\vartheta_x}{A_{l,j} + \pi\left\|\boldsymbol{x_{l,j}} - \boldsymbol{c}\right\|^2} = \varepsilon_p + F_{c\leftarrow l,disk}. \tag{5.42}$$

ε_p ist ein sehr kleiner Wert ($\varepsilon_p \ll \overline{F_{c\leftarrow l,disk}}$), der sicherstellt, dass in Gleichung (5.40) keine Division durch Null auftritt, wenn kein VAL auf den Cache einwirkt (zum Beispiel weil dessen Normale zur anderen Seite zeigt).

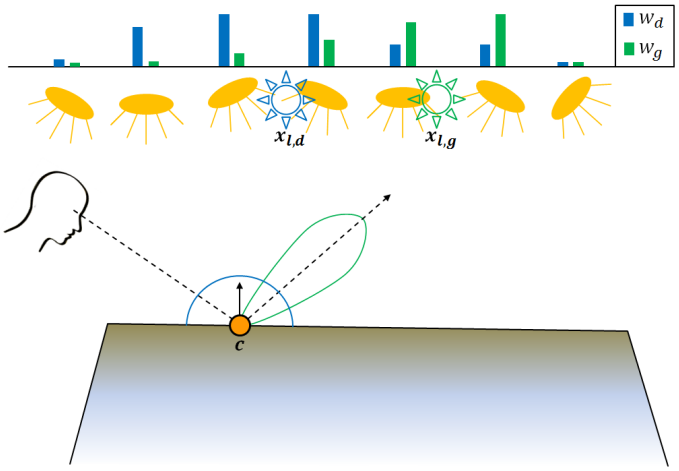

Abbildung 5.11: Darstellung der Gewichtungsfunktionen für die Proxy-Lichtquellen. Durch die Darstellung wird ersichtlich, dass für glänzende und diffuse Reflexionen nicht die gleiche Gewichtungsfunktion genutzt werden kann. Die dargestellte Gewichtung entspricht qualitativ der Gewichtungsfunktion zum jeweiligen VAL.

Abbildung 5.11 zeigt, wie die VALs für einen Cache gewichtet werden. Die Abbildung zeigt auch, dass die Gewichtungsfunktion aus Gleichung (5.42) für glänzende Reflexionen keine sinnvolle Proxy-Position liefert. Für glänzende Reflexionen ist der Betrachtungswinkel von Bedeutung, es ist also nicht möglich, die BRDF bei der Gewichtung unbeachtet zu lassen, wie dies bei der diffusen Reflexion gemacht wurde. Bei der Herleitung der Formel zur Berechnung der Leuchtdichte für glänzende Reflexionen in Kapitel 5.3.2 wurde jedoch darauf geachtet, dass eine gewisse Ähnlichkeit zur Formel für die Berechnung der diffusen Reflexionen besteht. Dies verursacht zwar Restriktionen, es erleichtert aber auch die Bildung einer Gewichtungsfunktion. Die Leuchtdichte für glänzende Reflexionen ergibt sich wie folgt (vgl. Gleichung (5.33)):

$$L_{o,g}(\boldsymbol{x},\boldsymbol{\omega}) \approx \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_g(\boldsymbol{x})(k+2)\cos^k\vartheta_e}{2\pi} \frac{\cos^+\vartheta_l \cos^+\vartheta_x}{A_{l,j}+\pi\left\|\boldsymbol{x_{l,j}}-\boldsymbol{x}\right\|^2} = \sum_{j=1}^{m} \frac{\phi_{l,j}\rho_g(\boldsymbol{x})(k+2)\cos^k\vartheta_e}{2\pi A_{l,j}} F_{c\leftarrow l,disk}. \tag{5.43}$$

Wenn man davon ausgeht, dass der Exponent k für die Oberflächenpunkte in der Nähe des Caches gleich ist, kann man die betrachtungsabhängige Gewichtung wie folgt bilden (unterschiedliche Exponenten würden dazu führen, dass die Funktion eine zu starke oder zu schwache Gewichtung auf den Winkel ϑ_e abbildet):

$$w_{j,g}(\boldsymbol{c}) = \varepsilon_p + \cos^k\vartheta_e\, F_{c\leftarrow l,disk}. \tag{5.44}$$

Damit sind alle nötigen Gleichungen zur Berechnung der Proxy-Punktlichtquelle bekannt. Die hier beschriebene Berechnung der Proxy-Lichtquellen hat den Vorteil, dass sie sehr einfach ist und entsprechend schnell berechnet werden kann. Die Erzeugung der Proxy-Lichtquellen stellt algorithmisch einen sehr geringen Mehraufwand zur eigentlichen Berechnung der Reflexion im Cache dar. Der folgende Pseudocode zeigt die Berechnung:

Listing 5.1: Algorithmus zur Berechnung der Proxy-Lichtquellen für indirekte Reflexionen.

```
for each Cache (c)
        L_o,d = 0, x_p,d = 0, w_ges,d = 0
        L_o,g = 0, x_p,g = 0, w_ges,g = 0
        f_r,d = ρ_d(c)/π
        f_r,g = ρ_g(c)(k + 2)/(2π)
        for each VAL (x_l, ϕ_l, A_l) in RSM
                F = (cos⁺ϑ_l cos⁺ϑ_x)/(A_l + π‖x_l − c‖²)
                w_d = ε_p + F
                w_g = ε_p + cos^k ϑ_e F
                x_p,d += w_d x_l
                x_p,g += w_g x_l
                L_o,d += ϕ_l f_r,d F
                L_o,g += ϕ_l f_r,g cos^k ϑ_e F
                w_ges,d += w_d
                w_ges,g += w_g
        end
        x_p,d /= w_ges,d
        x_p,g /= w_ges,g
        ϕ_p,d = 4πL_o,d ‖x_p,d − c‖² / (f_r,d (N(c) · ω_i)⁺)
        ϕ_p,g = 4πL_o,g ‖x_p,g − c‖² / (f_r,g cos^k ϑ_e (N(c) · ω_i)⁺)
end
```

Dieser Algorithmus lässt sich sehr gut parallelisieren und kann somit effizient in einem Shader-Programm implementiert werden. Dies ist unter anderem Thema in Kapitel 6.1.3.

Jeder Radiance-Cache beinhaltet jetzt zwei Proxy-Punktlichtquellen (eine für diffuse und eine für glänzende Reflexionen), die jeweils über zwei Parameter verfügen. Wie diese Lichtquellen zur Interpolation genutzt werden, wird im nächsten Unterkapitel beschrieben.

5.3.5 Interpolation: Beleuchtung mit Proxy-Lichtquellen

Um eine durchgehende Beleuchtung auf Basis der Cache-Proxy-Punktlichtquellen zu erzeugen, muss ein geeignetes Interpolationsschema gefunden werden. In Kapitel 5.2.2 wurde bereits darauf hingewiesen, dass es nicht ideal ist, Punktlichtquellen für die Approximation des indirekten Lichts zu verwenden, da diese unerwünschte Singularitäten erzeugen, wenn sich der Empfängerpunkt der Punktlichtquelle nähert. Diese Singularitäten führen zu punktförmigen Artefakten, die die falsche Form der Lichtquelle entlarven können. Trotzdem wird für das LightSkin-Verfahren die Verwendung von punktförmigen Proxy-Lichtquellen vorgeschlagen, was in gewisser Weise als Widerspruch zu vorherigen Erkenntnissen gesehen werden kann. Dieser Widerspruch soll im Folgenden aufgehoben werden.

Ein naiver Ansatz besteht darin, die neu erzeugten Proxy-Lichtquellen unmittelbar für die indirekte Beleuchtung der sichtbaren Oberflächenpunkte zu nutzen. Hierfür müssen für jeden Oberflächenpunkt die Proxy-Lichtquellen bestimmt werden, die den größten Einfluss auf diesen haben. Diese erhält man zum Beispiel, indem man die nächsten Caches zu einem Oberflächenpunkt sucht und deren Proxy-Lichtquellen nutzt. Dieser Ansatz funktioniert relativ gut, wenn die Oberflächen, die das indirekte Licht erzeugen und somit entsprechend viele VALs beherbergen, weit entfernt von dem Oberflächenpunkt liegen. In diesem Fall liegen die Proxy-Lichtquellen ebenfalls weit vom Oberflächenpunkt entfernt, so dass deren punktförmige Strahlungseigenschaft verborgen wird. Liegen die VALs jedoch nahe zum Oberflächenpunkt, liegen auch die Proxy-Lichtquellen nahe und man kann wieder deutlich punktförmige Artefakte erkennen (siehe Abbildung 5.12). Die direkte Verwendung der Punktlichtquellen auf die sichtbaren Oberflächen ist also keine sinnvolle Methode, um eine durchgehende indirekte Reflexion zu erzeugen.

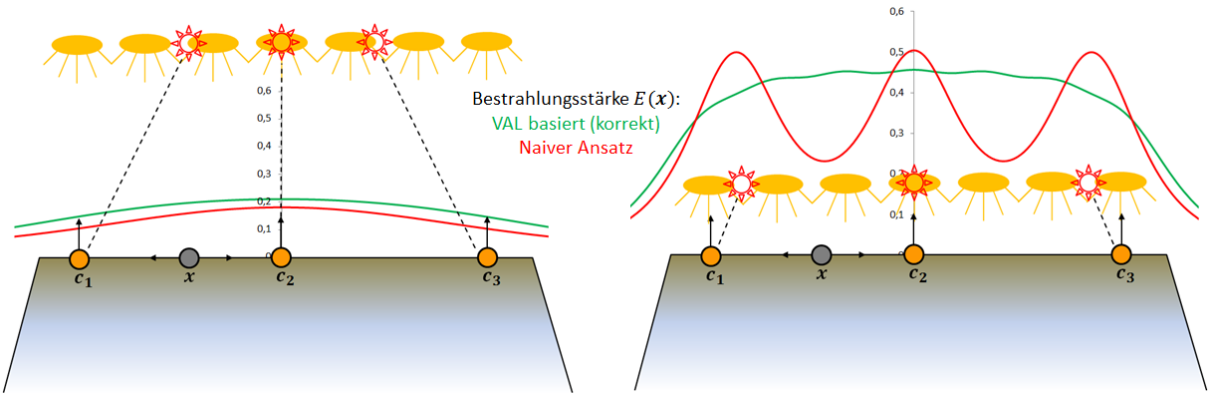

Abbildung 5.12: Indirekte Beleuchtung, basierend auf der direkten Nutzung der Proxy-Lichtquellen. Während dieser Ansatz für entfernte indirekte Lichtquellen akzeptable Verläufe für die Bestrahlungsstärke $E(x)$ berechnet (links), kommt es bei nahen indirekten Lichtquellen wieder zu charakteristischen punktförmigen Artefakten (rechts).

Um das Problem der punktuellen Artefakte für nahe Reflexionen zu lösen, wird in dieser Arbeit vorgeschlagen, die Proxy-Punktlichtquellen nicht als stationäre Lichtquellen zu verstehen, sondern sie stattdessen mit den Empfängerpunkten zu bewegen. Dies kann man erreichen, indem man die Position und den Lichtstrom der Lichtquelle interpoliert, anstatt diese direkt für die Beleuchtung zu verwenden. So erzeugt jeder sichtbare Oberflächenpunkt selbst eine Proxy-Lichtquelle aus den Proxy-Lichtquellen der Caches. Hierfür müssen die Caches betrachtet werden, die dem Oberflächenpunkt nahe bzw. diesem ähnlich sind. Wie viel eine Cache-Lichtquelle zur neuen Proxy-Lichtquelle beiträgt, muss wieder durch eine geeignete Gewichtungsfunktion beschrieben werden. Prinzipiell lässt sich sagen, dass eine Cache-Lichtquelle dann besonders wertvoll für einen Oberflächenpunkt ist,

wenn der Cache eine ähnliche Oberflächennormale besitzt und nahe zum Punkt liegt. Aus diesem Zusammenhang haben Ward et al. [WRC88] eine Metrik entwickelt, die als Gewichtungsfunktion beim Irradiance-Caching für diffuse Reflexionen Verwendung findet:

$$w_d'(\boldsymbol{x},\boldsymbol{c}) = \frac{1}{\varepsilon(\boldsymbol{x},\boldsymbol{c})},\quad \varepsilon(\boldsymbol{x},\boldsymbol{c}) = \frac{\|\boldsymbol{x}-\boldsymbol{c}\|}{R} + \sqrt{1-\big(N(\boldsymbol{x})\cdot N(\boldsymbol{c})\big)^{+}}. \tag{5.45}$$

ε bezeichnet die Fehlerheuristik. Der in der Heuristik vorkommende Term R repräsentiert das harmonische Mittel der Abstände zwischen dem Cache und den anderen Oberflächenpunkten. Dieses Mittel ist zur korrekten Interpolation, unter Einbeziehung von Verdeckungen, nötig. Da in diesem Unterkapitel die Verdeckung (noch) nicht betrachtet wird, kann der Term vorerst ignoriert bzw. gleich eins gesetzt werden. Der verbleibende Term zeigt, dass der Fehler in der Beleuchtung des Punktes $\boldsymbol{x}$ in linearem Zusammenhang zur Distanz zum Cache $\boldsymbol{c}$ und in nahezu linearem Zusammenhang zum Winkel zwischen den Normalen steht. Die Herleitung dieser Heuristik basiert auf dem Split-Sphere-Modell von Ward. Dieses Modell betrachtet die maximale Veränderung der Beleuchtungsstärke (Irradiance) in einem Punkt $\boldsymbol{x}$, wenn dieser verschoben bzw. rotiert wird (die Normale sich verändert). Es werden demnach die folgenden Gradienten abgeschätzt: $\frac{\partial E}{\partial \boldsymbol{x}}$ und $\frac{\partial E}{\partial \boldsymbol{n}}$. Für die Abschätzung dieser Gradienten müssen einige Überlegungen zur resultierenden Beleuchtungsstärke vorweggehen: Die Beleuchtungsstärke E ergibt sich aus dem Integral der Leuchtdichte über die Hemisphäre eines Punktes, welches durch den Kosinusterm $(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^{+}$ gewichtet wird:

$$E(\boldsymbol{x}) = \int_{\Omega^{+}} L_i\,(\boldsymbol{x},\boldsymbol{\omega_i})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^{+}\,d\boldsymbol{\omega_i}. \tag{5.46}$$

Nusselts Analogon [Nus28] besagt, dass der Term $(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^{+}d\boldsymbol{\omega_i}$ eine Projektion des differentiellen Raumwinkels $d\boldsymbol{\omega_i}$ auf eine Einheitskreisscheibe beschreibt (siehe Abbildung 5.13). Ergo kann die empfangene Beleuchtungsstärke als Integral der projizierten Leuchtdichte über diese Kreisscheibe interpretiert werden. Um die maximal mögliche Veränderung der Beleuchtungsstärke abzuschätzen, wird die Hemisphäre und damit die Kreisscheibe in zwei Hälften zerlegt (Split-Sphere). Eine Hälfte soll komplett dunkel (also kein eingehendes Licht enthalten) und eine Hälfte soll hell sein. Wenn man nun Abbildung 5.13 betrachtet, wird ersichtlich, dass sich das Integral über die Kreisscheibe und somit die Beleuchtungsstärke am stärksten verändert, wenn die Veränderung entlang der Halbierenden des Kreises vollzogen wird. In Abbildung 5.13 ist ebenfalls dargestellt, wie sich die Verteilung der Beleuchtungsstärke ändert, wenn der Oberflächenpunkt verschoben bzw. gedreht wird.

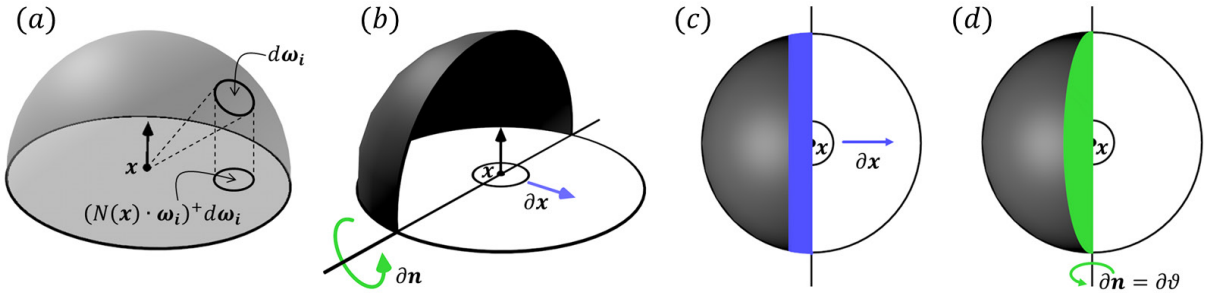

Abbildung 5.13: Nusselts Analogon (a) und Split-Sphere-Modell (b,c,d). Beim Split-Sphere-Modell (b) wird die maximal mögliche Veränderung der Beleuchtungsstärke für die Verschiebung (c) bzw. die Rotation (d) eines Oberflächenpunktes $\boldsymbol{x}$ abgeschätzt. Wird der Punkt $\boldsymbol{x}$ bewegt (rotiert oder verschoben), tritt die größte Veränderung in der Beleuchtungsstärke dann auf, wenn eine Hälfte der Hemisphäre als komplett dunkel und eine Hälfte als komplett hell angenommen wird (von einer Hälfte strahlt Licht auf $\boldsymbol{x}$ ein und von der anderen nicht). In (c) ist dargestellt, wie sich die Beleuchtungsstärke verändert, wenn $\boldsymbol{x}$ verschoben wird (blau), während (d) die Veränderung zeigt, wenn $\boldsymbol{x}$ rotiert wird (grün).

Zur Herleitung der Heuristik wird der Darstellung in [Jar08] gefolgt. Die Veränderung durch die Translation des Punktes $\frac{\partial E}{\partial \boldsymbol{x}}$ kann abgeschätzt werden, indem man die blau markierte Fläche aus Abbildung 5.13 als schmales Rechteck mit den Seitenlängen $2r$ und $\Delta\boldsymbol{x}$ interpretiert. Die initiale Beleuchtungsstärke ist proportional zur Hälfte der Kreisfläche ($\frac{1}{2}\pi r^2$), somit ergibt sich für die relative Veränderung:

$$\frac{\partial E}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} \approx \frac{2r\Delta\boldsymbol{x}}{\frac{1}{2}\pi r^2} = \frac{4}{\pi}\Delta\boldsymbol{x}. \tag{5.47}$$

Der Radius der Kreisscheibe entspricht eins. Man erkennt, dass die approximative Veränderung der Beleuchtungsstärke proportional zur Verschiebung des Punktes ist ($\Delta\boldsymbol{x} = \|\boldsymbol{x} - \boldsymbol{c}\|$).

Die relative Veränderung der Beleuchtungsstärke durch die Rotation des Punktes $\frac{\partial E}{\partial \boldsymbol{n}} = \frac{\partial E}{\partial \vartheta}$ kann ebenfalls leicht ermittelt werden: Die in Abbildung 5.13 grün gekennzeichnete Fläche ist die Hälfte einer Ellipse. Die Fläche einer Ellipse ist gegeben durch $A = \pi r_1 r_2$, wobei $r_1 = r$ ist und $r_2 = r\sin(\Delta\vartheta)$ bezeichnet. Daraus folgt:

$$\frac{\partial E}{\partial \vartheta}\Delta\vartheta \approx \frac{\frac{1}{2}\pi r^2 \sin(\Delta\vartheta)}{\frac{1}{2}\pi r^2} = \sin(\Delta\vartheta). \tag{5.48}$$

Der Sinusterm wird nun zwischen -90 und +90 Grad durch den Term $\sqrt{1 - \left(N(\boldsymbol{x})\cdot N(\boldsymbol{c})\right)^+}$ approximiert (durch die ersten beiden Terme der Taylor-Reihe).

Die Interpolation der Position und des Lichtstroms der diffusen Proxy-Lichtquelle, auf Basis der Gewichtung von Ward aus Gleichung (5.45), kann nun wie folgt formuliert werden:

$$\boldsymbol{x}_{v,d} = \frac{\sum_{j=0}^{n} w'_{d,j}(\boldsymbol{x},\boldsymbol{c})\boldsymbol{x}_{p,d,j}}{\sum_{j=0}^{n} w'_{d,j}(\boldsymbol{x},\boldsymbol{c})}, \qquad \phi_{v,d} = \frac{\sum_{j=0}^{n} w'_{d,j}(\boldsymbol{x},\boldsymbol{c})\phi_{p,d,j}}{\sum_{j=0}^{n} w'_{d,j}(\boldsymbol{x},\boldsymbol{c})}. \tag{5.49}$$

Siehe hierzu auch Abbildung 5.14. Wendet man die sich neu ergebenen Lichtquellen ($\boldsymbol{x_v}$, ϕ_v) auf die Oberflächenpunkte an, so ergeben sich immer noch Artefakte, wenn die Proxy-Lichtquellen der Caches sehr nahe zum Oberflächenpunkt $\boldsymbol{x}$ liegen. Dies rührt daher, weil nach der Gewichtung von Ward der Einfluss des Caches umgekehrt proportional zum Abstand zum Oberflächenpunkt steht. Entfernt sich der Oberflächenpunkt nur etwas vom Cache, wird der Einfluss des Caches stark abfallen. Die Singularitäten werden also durch die Metrik nicht behoben (siehe Abbildung 5.14). Wenn man jedoch das Split-Sphere-Modell betrachtet, so ist die folgende Gewichtung ebenfalls eine gültige Approximation (denn es besteht weiterhin ein linearer Zusammenhang):

$$w'_d(\boldsymbol{x},\boldsymbol{c}) = \left(1 - \frac{\|\boldsymbol{x}-\boldsymbol{c}\|}{d_{max}}\right)\sqrt{\left(N(\boldsymbol{x})\cdot N(\boldsymbol{c})\right)^+}. \tag{5.50}$$

d_{max} kennzeichnet hier den Abstand zum entferntesten Cache, der noch für die Interpolation in Betracht kommt. Diese Metrik hat den Vorteil, dass sie für jede Cache-Dichte saubere Verläufe bildet, wenn man von einer festen Anzahl von betrachteten Caches ausgeht (zum Beispiel 8 oder 16 Caches). Ferner liefert sie zwar nicht ganz exakte Energieverläufe für nahe Proxy-Lichtquellen, aber sie erzeugt keine Artefakte für diese Fälle. Dies ist in Abbildung 5.14 für translatorische Bewegungen und in Abbildung 5.15 für Rotationen des Oberflächenpunkts $\boldsymbol{x}$ dokumentiert.

Für die glänzende Reflexion wird eine leicht abgewandelte Form dieser Gewichtung genutzt. Da glänzende Reflexionen vom Betrachtungswinkel abhängen, wird anstatt der Normalen der Spiegelvektor (R) vom Betrachter zum Oberflächenpunkt bzw. zum Cache genutzt:

$$w'_d(\boldsymbol{x},\boldsymbol{c}) = \left(1 - \frac{\|\boldsymbol{x}-\boldsymbol{c}\|}{d_{max}}\right)\sqrt{\left(R(\boldsymbol{x})\cdot R(\boldsymbol{c})\right)^+}. \tag{5.51}$$

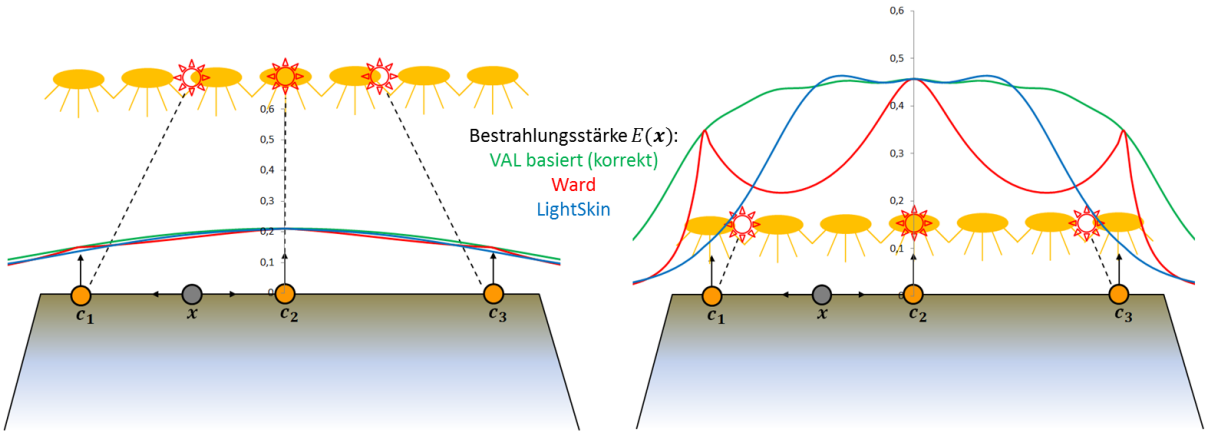

Abbildung 5.14: Vergleich der Bestrahlungsstärke für die verschiedenen Gewichtungsfunktionen aus (5.45) und (5.50) für die Translation des Oberflächenpunkts $\boldsymbol{x}$. Für ferne indirekte Punktlichtquellen funktionieren beide Gewichtungen sehr gut. Für nahe indirekte Lichtquellen versagt jedoch die Gewichtungsfunktion von Ward (5.45) und erzeugt Artefakte auf Höhe der Caches. Die Gewichtung des LightSkin-Verfahrens aus Gleichung (5.50) erzeugt einen glatten Bestrahlungsstärkeverlauf, selbst wenn die Lichtquellen sehr nahe zum Empfängerpunkt liegen. Jedoch kann man einen zu frühen energetischen Abfall erkennen, der dem Betrachter in der Praxis aber kaum auffällt.

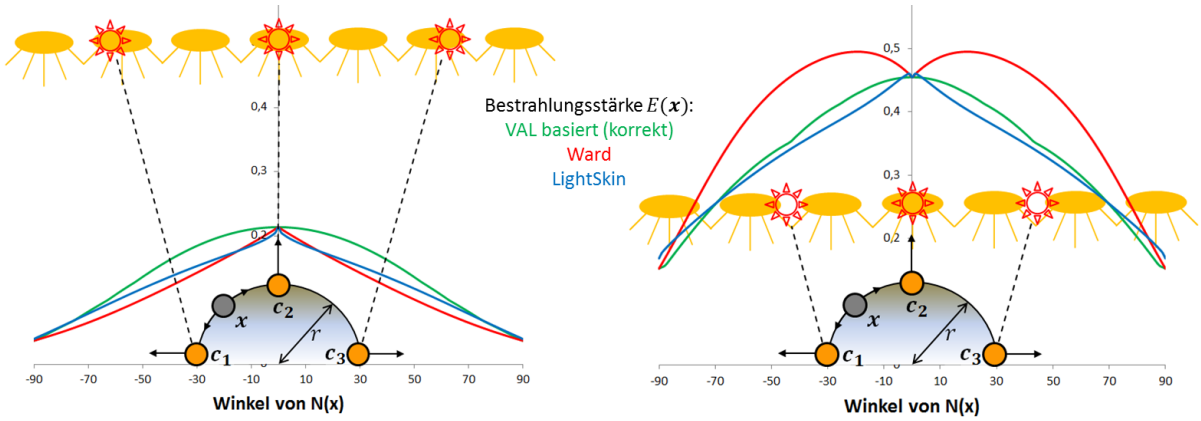

Abbildung 5.15: Vergleich der Bestrahlungsstärke für die verschiedenen Gewichtungsfunktionen (5.45) und (5.50) für die Rotation des Oberflächenpunkts $\boldsymbol{x}$. r wurde so klein gewählt, dass die Verschiebung des Punktes nicht ins Gewicht fällt. Für ferne Lichtquellen sind beide Funktionen gleichwertig, für nahe bildet wiederum die Gewichtungsfunktion (5.50) des Light-Skin-Verfahrens einen exakteren Bestrahlungsstärkeverlauf.

Algorithmisch lässt sich die Interpolation auf Basis der Proxy-Lichtquellen wie folgt zusammenfassen (die Menge Q enthält die Caches, deren Gewichtung $w' > \varepsilon$ ist. Wie diese Menge gebildet wird, wird in Kapitel 6.1.4 zur Implementierung beschrieben.):

Listing 5.2: Interpolationsalgorithmus zur Berechnung von indirekten Reflexionen auf Basis von Proxy-Lichtquellen.

```
for each Surfacepoint (x) in Framebuffer
        x_{p,x,d} = 0, φ_{p,x,d} = 0, w'_{ges,d} = 0
        x_{p,x,g} = 0, φ_{p,x,g} = 0, w'_{ges,g} = 0
        d_max = 0
        for each Cache (c) in Q
                d_max = max(d_max, ||x − c||)
        end
        for each Cache (c, x_{p,c,d}, φ_{p,c,d}, x_{p,c,g}, φ_{p,c,g}) in Q
                d_{x→c} = 1 − ||x − c||/d_max
                w'_d = d_{x→c} sqrt((N(c) · N(x))^+)
                w'_g = d_{x→c} sqrt((R(c) · R(x))^+)
                x_{p,x,d} += w'_d x_{p,c,d}
                x_{p,x,g} += w'_g x_{p,c,g}
                φ_{p,x,d} += w'_g φ_{p,c,d}
                φ_{p,x,g} += w'_g φ_{p,c,g}
                w'_{ges,d} += w'_d
                w'_{ges,g} += w'_g
        end
        x_{p,x,d} /= w'_{ges,d}
        x_{p,x,g} /= w'_{ges,g}
        φ_{p,x,d} /= w'_{ges,d}
        φ_{p,x,d} /= w'_{ges,g}
        L_o(x, ω) = (f_{r,d}(x)φ_{p,x,d})/(4π||x − x_{p,x,d}||) + (f_{r,g}(x, ω, ω_i)φ_{p,x,g})/(4π||x − x_{p,x,g}||)
end
```

Damit ist das Prinzip der Interpolation von indirekten Reflexionen für das LightSkin-Verfahren vollständig beschrieben. Im nächsten Unterkapitel werden einige Ergebnisse präsentiert und ein Überblick über die Einschränkungen des Verfahrens gegeben.

5.3.6 Diskussion

Das in diesem Kapitel (5.3) vorgestellte Verfahren zur Berechnung von indirekten Reflexionen auf Basis der LightSkin-Interpolationsmethode soll in diesem Abschnitt kritisch diskutiert werden. Eine Diskussion an dieser Stelle der Arbeit ist sinnvoll, denn die bis jetzt vorgestellten Konzepte lassen sich losgelöst vom noch folgenden Teil der Arbeit implementieren und wurden demzufolge bereits separat publiziert [LB13a]. Da bestimmte optische Phänomene, wie z. B. weiche Schatten und Subsurface-Scattering, erst später in dieser Arbeit vorgestellt werden, sollen sie für diese Diskussion vorerst unberücksichtigt bleiben.

In Kapitel 5.3.3 wurden Gütekriterien für Interpolationsmethoden beschrieben. Diese Kriterien werden in diesem Abschnitt aufgegriffen, um das vorgestellte Interpolationsverfahren zu bewerten. Dabei lassen sich bestimmte Kriterien zusammenhängend betrachten, da sie in Korrelation zueinander stehen. Zuerst sollen die Forderungen nach plausiblen und betrachtungsabhängigen Reflexionsgradienten, einer geringen Anzahl von Stützpunkten und die Erhaltung von Oberflächendetails betrachtet werden. Abbildung 5.16 zeigt einen exemplarischen Vergleich zwischen einer Szene, bei der jeder Oberflächenpunkt separat durch alle VALs beleuchtet wurde (rechts) und der Beleuchtung basierend auf dem LightSkin-Interpolationsverfahren (links). Jedes der drei gezeigten Modelle wird lediglich durch 200 Caches repräsentiert. Man kann deutlich erkennen, dass bereits diese geringe Anzahl an Caches ausreicht, um eine überzeugende Reflexion zu gewährleisten. Diffuse und leicht glänzende Reflexionen lassen sich mit weniger Caches approximieren als stark glänzende Reflexionen. Da für jeden sichtbaren Oberflächenpunkt eine eigene Proxy-Punktlichtquelle berechnet wird, werden Oberflächendetails wie variierende BRDFs, Oberflächennormalen und der Betrachtungswinkel vom Verfahren berücksichtigt. Selbst wenn für einen Oberflächenpunkt keine Caches gefunden werden, die eine passende Normale besitzen, werden trotzdem die Unebenheiten auf der Modelloberfläche bei der Interpolation abgebildet. In diesem Fall wird die Reflexion durch den Term $(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+$ bzw. die BRDF skaliert, so dass keine Details der Oberfläche entfernt werden. Will man auf der Meso-Ebene des Modells ebenfalls korrekte Reflexionen erzeugen, muss sichergestellt sein, dass die Caches die Oberflächennormalen des Modells ausreichend repräsentieren (siehe Abbildung 5.17).

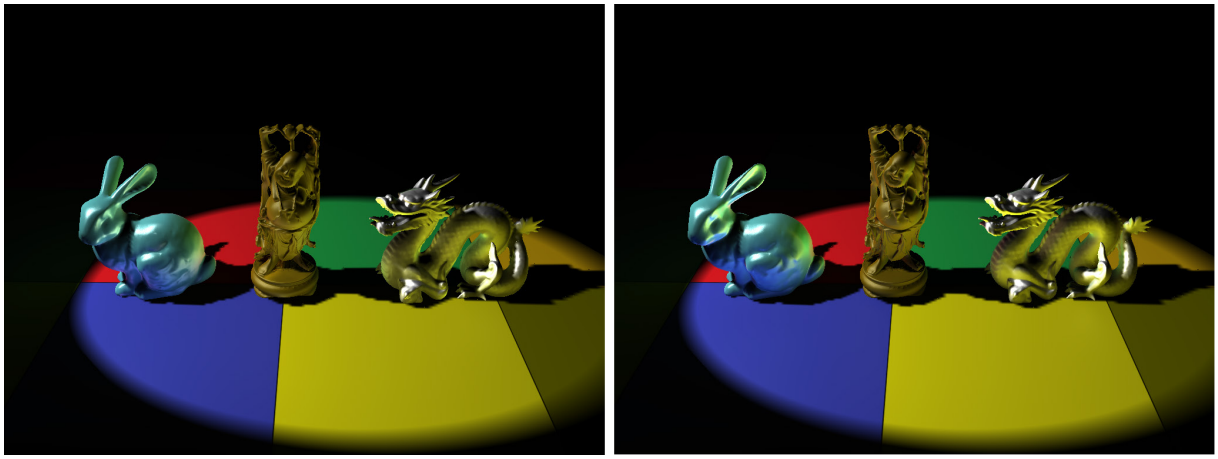

Abbildung 5.16: Vergleich der Ergebnisse des Interpolationsverfahrens (links) und einer Brute-Force-Lösung, bei der jedes VAL auf jeden Oberflächenpunkt angewendet wurde (rechts). Für die Interpolation verwendet jedes Modell lediglich 200 Caches und es werden 16.000 VALs erzeugt. Während die Berechnung des rechten Bilds ca. vier Sekunden benötigt, benötigt das linke Bild dank der Interpolation ca. 5ms.

Bezüglich der physikalischen Korrektheit der Reflexion muss vorweggenommen werden, dass keine Verdeckung des indirekten Lichts berücksichtigt wird und somit ein systematischer Fehler existiert. Dennoch wurde darauf geachtet, dass die Ausbreitung des indirekten Lichts und die energetischen Zusammenhänge korrekt abgebildet werden. Ein kleiner energetischer Fehler tritt bei spiegelnden Reflexionen auf. Dieser Fehler resultiert aus der vorher getroffenen approximativen Annahme, dass die spiegelnde BRDF für ein komplettes VAL konstant ist (siehe Kapitel 5.3.2). Dies führt zu Energieverlusten, wenn das VAL sehr nahe zum Oberflächenpunkt liegt. Der Energieverlust äußert sich optisch in einem unerwünschten Ambient-Occlusion-Effekt in den Modellecken. Sieht man von diesen prinzipiellen Fehlern durch die VAL-Beleuchtung ab, kann man sagen, dass für jeden Cache die korrekte Beleuchtungsstärke gespeichert wird. Demnach konvergiert das Verfahren mit zunehmender Anzahl der Caches zur korrekten Lösung. Allerdings wurde die Interpolation nicht entwickelt, um eine physikalisch korrekte Lösung zu erhalten, sondern um möglichst effizient eine plausible Lösung zu erzeugen, die nur wenige Caches benötigt.

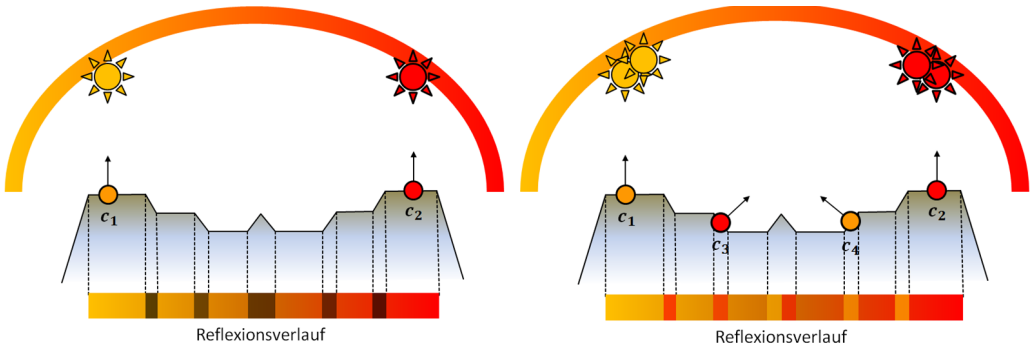

Abbildung 5.17: Selbst wenn das Umgebungslicht nicht ausreichend durch Proxy-Lichtquellen repräsentiert wird, werden Veränderungen in den Normalen der Oberfläche abgebildet (durch Skalierung der Reflexion, links). Es werden also keine Oberflächendetails durch die Interpolation „verschluckt“, wenn keine Caches mit passender Normale vorhanden sind. Durch das Hinzufügen von weiteren Caches erhält man auch für feine Strukturen korrekte Reflexionen (rechts).

Als nächstes sollen der Speicheraufwand der Stützpunkte (Caches), die schnelle Berechnung der Interpolation auf Basis der Stützpunkte und die Implementierung auf der GPU diskutiert werden. Jeder Cache speichert initial seine Normale, Position und den Phong-Reflexionsexponenten für glänzende Materialien. Darüber hinaus werden für jeden Cache zwei Proxy-Punktlichter berechnet. Diese lassen sich durch die Position und den Lichtstrom repräsentieren. Ein Cache besteht damit aus 18 Skalaren. Da prinzipiell nur wenige Caches pro Modell (50-1000) nötig sind, ist der zusätzliche Speicheraufwand für diese vernachlässigbar gering. Für die Interpolation gilt, dass die n nächsten Caches für jeden Oberflächenpunkt bestimmt werden müssen. Diese Operation ist teuer, denn sie benötigt eine Beschleunigungsstruktur für räumliche Anfragen, die viele inkohärente Speicherzugriffe erzeugt. Dies spricht gegen eine GPU-Implementierung. Jedoch lässt sich eine fixe Anzahl von Caches für jeden Modelloberflächenpunkt im Voraus bestimmen, so dass eine Suchanfrage zur Laufzeit der Simulation unnötig wird. Allerdings muss die Gewichtung der Caches, wie sie in Kapitel 5.3.5 beschrieben wurde, für jeden Oberflächenpunkt zur Laufzeit erfolgen, da sich die Caches durch Modellanimationen geometrisch verändern können. Es hat sich gezeigt, dass die Betrachtung von 8 bis 16 Caches pro Oberflächenpunkt ausreicht, um gute Interpolationsergebnisse zu erzeugen. Der Algorithmus aus Kapitel 5.3.5 hat also eine konstante Laufzeitkomplexität, bei dem die konstanten Kosten darüber hinaus sehr gering sind. Eine Implementierung als Shader ist somit möglich. Allerdings hat das einfache Interpolationsmodell auf Basis von Proxy-Lichtquellen auch einen Nachteil, wenn in-

direktes Licht aus zwei entgegengesetzten Richtungen auf ein Modell scheint und nicht ausreichend Caches zur Repräsentation der Oberfläche vorhanden sind. In diesem Fall kommt es zu falschen Interpolationsverläufen der Proxy-Lichtquellen. Diese sind in Abbildung 5.18 dargestellt. Das Phänomen lässt sich verringern, indem man pro Cache den Abstand zum nächstgelegenen VAL speichert. Für die Interpolation kann dann die Proxy-Lichtquelle so bewegt werden, dass der minimale Abstand nicht unterschritten wird und somit keine zu hellen Reflexionen resultieren. Für praktische Anwendungen kann man dieses Phänomen jedoch unbeachtet lassen, da die meisten Beleuchtungsszenarien diesen Fehler nicht hervorheben.

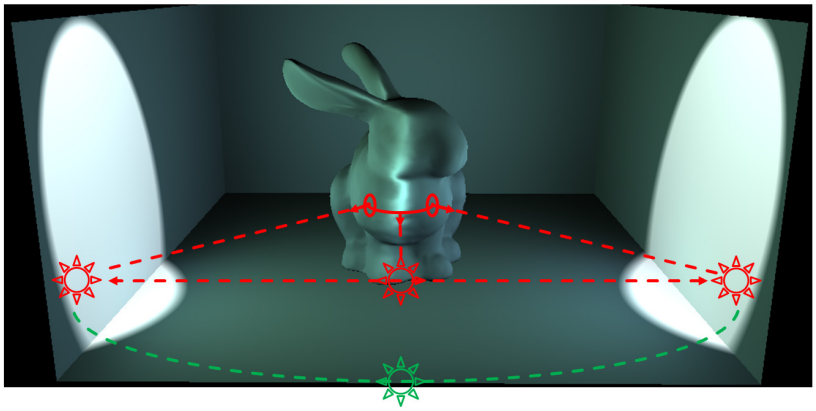

Abbildung 5.18: Die lineare Interpolation der Proxy-Lichtquellen kann durch eine ungünstige Konstellation des indirekten Lichts zu falschen Reflexionen führen (siehe helles Band in der Mitte des Bunny-Modells). Diese falschen Reflexionen entstehen, weil durch die lineare Interpolation der Lichtquellen (rot dargestellt) die resultierende neue Lichtquelle dem Oberflächenpunkt sehr nahe kommt und diesen somit stärker beleuchtet. Läge ein Cache auf Höhe der zu hellen Reflexionen, so würde dieser die Intensität der Lichtquelle passend regulieren. Um das Problem zu lösen, kann man bei der Interpolation der Lichtquelle einen Mindestabstand definieren, der sich aus dem geringsten Abstand aller VALs zu den verwendeten Caches ergibt (grün dargestellt).

Schlussendlich soll noch die temporäre Kohärenz bei Animationen diskutiert werden. Wird ein Modell animiert, so werden dessen Caches ebenfalls animiert (die Position und Normale wird angepasst). Hierdurch ergeben sich keine Inkonsistenzen bei der Bewegung und die Interpolationsergebnisse verändern sich im gleichen Verhältnis, wie dies der Betrachter durch die Animation erwartet. Auch die Animation der primären Lichtquelle führt zu adäquaten Veränderungen bei den Proxy-Lichtquellen. Allerdings kann eine zu geringe Auflösung der RSM dazu führen, dass flackernde Artefakte aus der mangelnden Abtastung resultieren. Die Verwendung der Caches erlaubt aber die Nutzung von RSMs mit höheren Auflösungen, so wird der Effekt deutlich reduziert. Die Bewegung der Kamera erzeugt ebenfalls keine Artefakte, da keine Neuverteilung der Caches erfolgt.

Zusammenfassend lässt sich sagen, dass das hier entwickelte Verfahren zur Erzeugung von indirekten Reflexionen bereits geeignet ist, um Szenen visuell deutlich aufzuwerten (siehe Abbildung 5.19). Jedoch erzeugt es keine indirekten weichen Schatten, wie dies andere aktuelle Verfahren machen. Die Erweiterung des Verfahrens um weiche Schatten ist Gegenstand des nächsten Unterkapitels.

Abbildung 5.19: Crytek-Sponza-Szene beleuchtet durch zwei Spotlichtquellen. (Links) nur direktes Licht, (Mitte) direktes Licht mit indirekten diffusen Reflexionen und (rechts) direktes Licht mit indirekten diffusen und glänzenden Reflexionen (die Helligkeit der indirekten Reflexionen wurde für den Druck erhöht).

5.4 Weiche Schatten

In diesem Unterkapitel wird ein neuer approximativer Algorithmus vorgestellt, der den in dieser Arbeit entwickelten Proxy-Lichtquellen-Ansatz aufgreift, um weiche Schatten effizient zu berechnen.

Die Berechnung von Schatten auf Basis von direkten analytischen Punkt- und Richtungslichtquellen ist verhältnismäßig einfach, da für jeden Oberflächenpunkt nur geprüft werden muss, ob dieser von der Lichtquelle aus sichtbar ist oder nicht. Aus diesem Zusammenhang haben sich das Shadow-Mapping [Wil78] und das Shadow-Volumes-Verfahren [Cro77] entwickelt. Beide Verfahren erzeugen in ihrer ursprünglichen Form hartkantige Schatten. Die Anwendung dieser Verfahren für indirektes Licht, welches beim Instant Radiosity durch virtuelle Lichtquellen repräsentiert wird, ist nicht sinnvoll, denn jedes indirekte Licht muss dann eine eigene Shadow-Map erzeugen. Die Kosten für eine virtuelle Lichtquelle liegen so bei $O(mn)$, wobei n die Anzahl der Lichtquellen und m die mittlere Anzahl der von den Lichtquellen sichtbaren Oberflächenpunkte ist. Da jede Lichtquelle noch auf die vom Betrachter aus k sichtbaren Oberflächenpunkte angewendet werden muss, ergibt sich eine Gesamtlaufzeitkomplexität von $O(mn + kn)$. Zwar existieren interaktive Verfahren, wie zum Beispiel das Imperfect Shadow-Mapping [RGK*08], die die konstanten Kosten für die Erzeugung einer Shadow-Map deutlich reduzieren, aber trotzdem kann nur eine geringe Menge von solchen Shadow-Maps gleichzeitig in Echtzeit erzeugt werden (um die 1000). Dieses Problem besteht ganz analog, wenn man direkte inhomogene Flächenlichtquellen mit Hilfe von endlich vielen virtuellen Lichtern simulieren will.

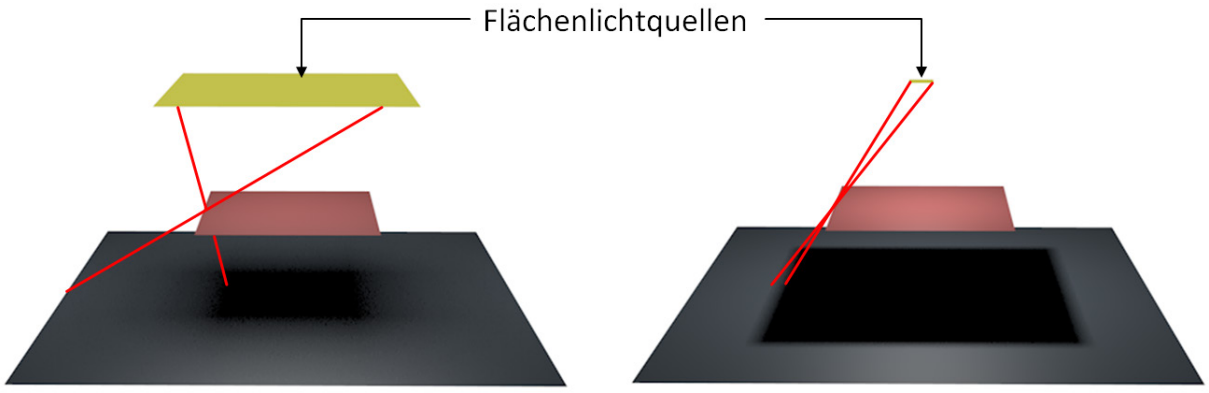

Abbildung 5.20: Durch die geometrische Ausbreitung von Lichtquellen resultieren weiche Schattenverläufe, weil Bereiche existieren, die nicht die komplette Lichtquelle wahrnehmen. Je kleiner die Lichtquelle wird, desto schärfer werden die Schattenkanten.

Weiche Schatten können nur in Szenen entstehen, wenn diese Lichtquellen mit endlichen Flächen enthalten. Durch die teilweise Verdeckung dieser Flächenlichtquellen kommt es zu weichen Schattierungsverläufen (siehe Abbildung 5.20). In der Echtzeit-Computergrafik wird die Berechnung von weichen Schatten, auf Basis von direkten Flächenlichtquellen, häufig als orthogonales Problem zur Berechnung von Reflexionen betrachtet. Es hat sich eine eigene Disziplin etabliert, die sich ausschließlich auf die Berechnung von weichen Schatten konzentriert. In dieser Disziplin wird von einer vereinfachten Form der Rendergleichung (vgl. Gleichung (2.7)) ausgegangen, die nur das Integral über die Flächen der direkten Lichtquellen (A_l) betrachtet [EAS*12]:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = \int_{A_l} f_r(\boldsymbol{x},\boldsymbol{\omega},\boldsymbol{\omega_i}) L_e\,(\boldsymbol{s},-\boldsymbol{\omega_i}) \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} V(\boldsymbol{x},\boldsymbol{s})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.52}$$

In dieser Gleichung wird nur das direkt emittierte Licht L_e betrachtet. Für alle weiteren Oberflächen der Szene wird davon ausgegangen, dass sie kein Licht zum Oberflächenpunkt $\boldsymbol{x}$ reflektieren. Dieses Integral wird weiter vereinfacht, indem davon ausgegangen wird, dass die lichtempfangenden Oberflächen eine konstante BRDF besitzen, sie also ideal diffuse Reflektoren sind:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = \frac{\rho(\boldsymbol{x})}{\pi} \int_{A_l} L_e\,(\boldsymbol{s},-\boldsymbol{\omega_i}) \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} V(\boldsymbol{x},\boldsymbol{s})(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+ d\boldsymbol{s}. \tag{5.53}$$

Die elementarste Vereinfachung besteht nun in der Annahme, dass die Reflexion und der Schatten in dieser Gleichung separiert werden können. Durch die Voraussetzung, dass die Lichtquelle relativ weit vom Oberflächenpunkt entfernt ist und sich somit der Term $\frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2}(N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+$ nur geringfügig über das Integral verändert, können die Integrale getrennt werden:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) \approx \frac{\rho(\boldsymbol{x})}{\pi} \frac{1}{A_l} \int_{A_l} \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+ (N(\boldsymbol{x})\cdot\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} d\boldsymbol{s} \int_{A_l} L_e\,(\boldsymbol{s},-\boldsymbol{\omega_i})\, V(\boldsymbol{x},\boldsymbol{s}) d\boldsymbol{s}. \tag{5.54}$$

Diese Separierung von Reflexion (linkes Integral) und der Schattenberechnung (rechtes Integral) erlaubt die dedizierte Betrachtung des Schattens. Die durch diese Vereinfachung entstehenden Fehler werden von Agrawala et al. in [ARH*00] diskutiert. Darüber hinaus geben Eisemann et al. in [EAS*09] und [EAS*12] einen umfangreichen Überblick über Echtzeitschattierungsverfahren, die auf dieser vereinfachten Form der Rendergleichung basieren. Für das LightSkin-Verfahren wird ebenfalls dieser Betrachtungsweise gefolgt, um weiche Schatten für indirekte Reflexionen zu erzeugen. Wie dies im Detail bewerkstelligt wird, ist Thema der folgenden Unterkapitel.

5.4.1 Schatten für indirektes Licht

Die Berechnung von weichen Schatten auf Basis von direkten Flächenlichtquellen ist für das LightSkin-Verfahren nur bedingt von Bedeutung, da nicht die Schattierung des direkten, sondern des indirekten Lichts berechnet werden soll. Grundsätzlich erlaubt die Separierung aus Gleichung (5.54) aber die folgende Schreibweise zur Berechnung von indirekten Reflexionen in Caches, unter Berücksichtigung der Verdeckung (vgl. Formel (5.41) und (5.43)):

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) \approx \sum_{j=1}^{n} \left(\frac{\rho_d(\boldsymbol{x})}{\pi} + \frac{\rho_g(\boldsymbol{x})(k+2)\cos^k \vartheta_e}{2\pi} \right) \frac{1}{A_l} F_{\boldsymbol{x} \leftarrow l,j,disk} \phi_{l,j} V(\boldsymbol{x}, \boldsymbol{x_{l,j}}). \tag{5.55}$$

Wobei n die Anzahl der VALs kennzeichnet. Es ist also ausreichend, nur einen Sichtbarkeitstest pro VAL durchzuführen. Jedoch bedeutet dieser Sichtbarkeitstest pro VAL auch, dass eine Beschleunigungsstruktur zur effizienten Strahlenverfolgung erzeugt werden muss. Die Aktualisierung dieser Beschleunigungsstruktur bei Animationen führt zu Laufzeitproblemen, die bereits in dieser Arbeit diskutiert wurden. Auch die Anwendung der VALs auf die Caches wird durch den Sichtbarkeitstest deutlich teurer, weswegen von dieser Implementierung abgesehen wurde.

Anstatt für jedes VAL separat die Sichtbarkeit zu prüfen, wird die Sichtbarkeit nur für die resultierende Proxy-Lichtquelle im Cache geprüft. Es ist allerdings nicht sinnvoll, die Sichtbarkeit für eine punktförmige Proxy-Lichtquelle zu prüfen, da diese nur vollständig verdeckt bzw. unverdeckt sein kann. Diese binäre Sichtbarkeitsprüfung würde flackernde Schatten verursachen. Um dies zu vermeiden, muss für jede Proxy-Lichtquelle eine geometrische Ausbreitung (also eine Fläche) für die Sichtbarkeitsprüfung gefunden werden. Die Ausbreitung der Proxy-Lichtquelle ist, wie deren Position, von den VALs abhängig, die auf den Cache wirken. Berechnet man außer der gewichteten Position auch die Standardabweichung aller VALs im Cache, so erhält man für die Proxy-Lichtquelle eine geometrische Größe (siehe Abbildung 5.21), die im Groben die Ausbreitung der VALs repräsentiert. Das Prinzip ist einfach: Sind die VALs, die auf einen Cache wirken, weit in der Szene verstreut, wird die Standardabweichung für die Position der Proxy-Lichtquelle entsprechend groß ausfallen. Sind die VALs dagegen sehr kompakt bzw. lokal verteilt, führt dies zu einer kleinen Standardabweichung. Für die Berechnung der Standardabweichung sollte man aus Laufzeitgründen berücksichtigen, dass die separate Berechnung der Varianz für jede Koordinatenachse schneller erfolgen kann, als die Berechnung der Varianz im Dreidimensionalen. Für die Standardabweichung der separierten Dimensionen gilt $\boldsymbol{\sigma} = (\sqrt{E(x^2) - E^2(x)}, \sqrt{E(y^2) - E^2(y)}, \sqrt{E(z^2) - E^2(z)})$. Hier bezeichnet E den Erwartungswert. Dieser entspricht der gewichteten Position der Proxy-Lichtquelle $\boldsymbol{x_p}$ im Cache, es gilt $\boldsymbol{x_p} = \big(E(x), E(y), E(z)\big)$. Demnach reicht es für die Berechnung der Standardabweichung, für jede VAL-Position $\boldsymbol{x_l}$ nur das gewichtete Quadrat $w\boldsymbol{x_l^2}$ zusätzlich zu speichern. Hierdurch werden nur zwei zusätzliche Vektormultiplikationen pro VAL für die Berechnung der Proxy-Lichtquellen benötigt. In einem zweiten Durchgang wird dann für jeden Cache die eigentliche Standardabweichung berechnet. So erhält man für jede Proxy-Lichtquelle eine Axis-Aligned-Bounding-Box (AABB), deren Seitenlängen $(2\sigma_x, 2\sigma_y, 2\sigma_z)$ entsprechen. Es ist wichtig zu erkennen, dass für die Erzeugung dieser AABB davon ausgegangen wird, dass jedes VAL vom Cache aus sichtbar ist. Für die Schattenberechnung wird nun geprüft, wie viel Prozent dieser Box aus Sicht des Caches durch andere Oberflächen verdeckt wird. Dabei wird davon ausgegangen, dass die Proxy-Lichtquelle über ihre komplette Fläche gleichmäßig strahlt. Diese Annahme erlaubt, den Lichtstrom der Lichtquelle ins Verhältnis zur Verdeckung im Cache zu setzen. Oder anders ausgedrückt: Der Lichtstrom der unverdeckten Proxy-Lichtquellen ϕ_p wird durch Verdeckungen (V) gedämpft:

$$\phi'_{p,x} = \phi_{p,x} \frac{1}{A_p} \int_{A_p} V(\boldsymbol{x}, \boldsymbol{s}) d\boldsymbol{s}. \tag{5.56}$$

Wegen des konstanten Lichtstroms über die Fläche der Lichtquelle kann dieser vor das Integral gezogen werden und es muss nur noch der Sichtbarkeitsterm über die Oberfläche der Proxy-Lichtquelle integriert werden. Es ist zu bedenken, dass dieser Schritt sowohl für die glänzende als auch für die diffuse Proxy-Lichtquelle im Cache durchgeführt werden muss. Im folgenden Abschnitt wird eine approximative analytische Lösung für das Integral über den Sichtbarkeitsterm vorgestellt.

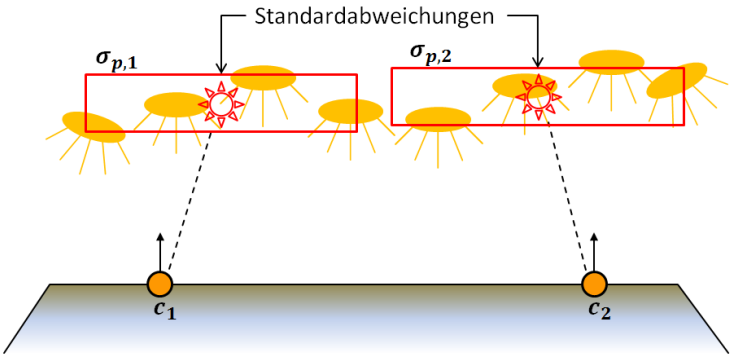

Abbildung 5.21: Für jede Proxy-Lichtquelle werden nicht mehr nur die gewichtete Position $\boldsymbol{x_p}$ und der Lichtstrom ϕ_p berechnet, sondern auch die Standardabweichung $\boldsymbol{\sigma_p}$ aller VAL-Positionen, um die ungefähre geometrische Ausbreitung der indirekten Lichtquelle als AABB abzubilden.

5.4.2 Vedeckungsapproximation

Die hier vorgestellte neue Lösung zur Berechnung des Verdeckungsgrades der Proxy-Lichtquelle wurde von Bunnell‘s Ansatz [Bun05] zur schnellen Berechnung von Ambient-Occlusion-Effekten inspiriert. Beim Ambient Occlusion wird davon ausgegangen, dass die komplette unverdeckte Hemisphäre über einen Punkt $\boldsymbol{x}$ gleichmäßig auf den Punkt einstrahlt. Das eingehende Licht wird nur durch andere Modelloberflächen innerhalb der Hemisphäre blockiert. Bunnell hat vorgeschlagen, die Oberflächen der Szene in Form von Surfels als Kreisflächen zu approximieren. Für eine triangulierte Szene wird für jedes Dreieck ein entsprechendes Surfel erzeugt, welches dem Inkreis des Dreiecks entspricht. Die Verwendung dieser Kreisflächen hat den Vorteil, dass der Formfaktor für diese Elemente sehr einfach analytisch berechnet werden kann (dies wurde bereits für die VALs ausgenutzt). Der Formfaktor ist nun ein direktes Maß für die Blockierung des indirekten Lichts und kann direkt die Reflexion dämpfen (Dämpfungsgrad a):

$$L'_o(\boldsymbol{x}, \boldsymbol{\omega}) = L_o(\boldsymbol{x}, \boldsymbol{\omega}) \left(1 - \max\left(1, \sum_{j=1}^{m} F_{x \leftarrow s,j}\right)\right) = L_o(\boldsymbol{x}, \boldsymbol{\omega}) a. \tag{5.57}$$

Das Prinzip, die blockierende Geometrie der Szene in Form von Kreisscheiben zu approximieren, wurde für diese Arbeit übernommen. Allerdings wird nicht für jedes Dreieck, sondern nur für jeden

Cache eine Kreisscheibe erzeugt. So werden Caches nicht nur verwendet, um das indirekte Licht einzusammeln und daraus Proxy-Lichtquellen zu erzeugen, sie wirken gleichzeitig als blockierende Elemente für andere Caches. Wie groß die Fläche eines Caches wird, ergibt sich aus der Szenengeometrie und der Dichte der Cache-Verteilung: Nachdem alle Caches in der Szene verteilt wurden, wird ein dreidimensionales Voronoi-Diagramm für diese berechnet [Ryc09]. Da jeder Cache auf einem Dreieck eines Modells liegt, besitzt er die Normale dieses Dreiecks. Nun wird die Schnittmenge der Ebene des Dreiecks $P_t(\boldsymbol{c})$, auf dem der Cache liegt, mit der Voronoi-Region $V_o(\boldsymbol{c})$ des Caches berechnet. Es ergibt sich ein konvexes Polygon. Auf die Ebene dieses konvexen Polygons werden nun alle Dreiecke projiziert, die in der Voronoi-Region des Caches liegen ($T_{proj} \in V_o(\boldsymbol{c})$). Die schlussendliche Fläche des Caches ergibt sich aus der Schnittmenge des konvexen Polygons mit den projizierten Dreiecken (siehe auch Abbildung 5.22):

$$A(\boldsymbol{c}) = Fläche(P_t(\boldsymbol{c}) \cap V_o(\boldsymbol{c}) \cap \left(\cup_{Tproj \in V_o(c)} T_{proj}\right)). \tag{5.58}$$

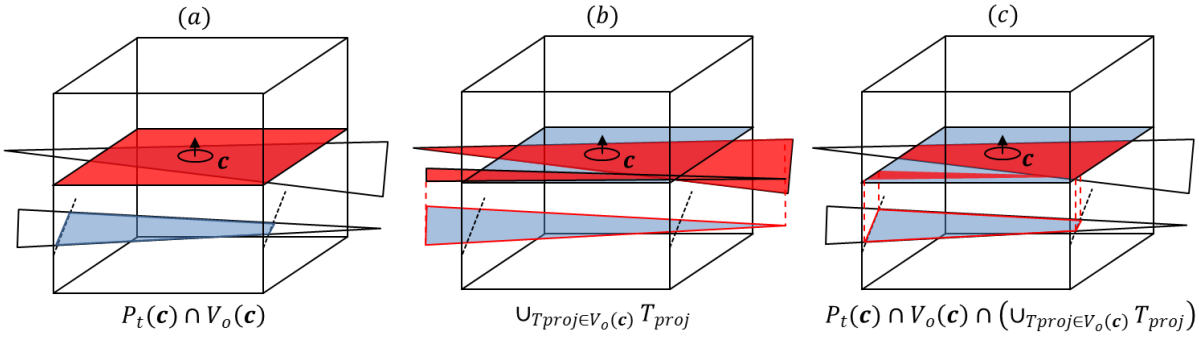

Abbildung 5.22: Berechnung der Cache-Fläche $A(\boldsymbol{c})$. (a) Es wird die Ebene $P_t(\boldsymbol{c})$ des Dreiecks bestimmt, in der der Cache $\boldsymbol{c}$ liegt und mit der Voronoi-Region $V_o(\boldsymbol{c})$ geschnitten (die Voronoi-Region ist in dieser Abbildung ein Quader). (b) Die Dreiecke, die die Voronoi-Regionen schneiden, werden auf die Ebene $P_t(\boldsymbol{c})$ projiziert und zusammengefasst. (c) Die Schnittmenge aus (a) und (b) wird berechnet, die Fläche dieser Menge wird dann als Fläche für die Kreisscheibe des Caches genutzt.

Nachdem die Flächen für die Caches bestimmt wurden, kann für jeden Cache anhand des Kreisscheibenformfaktors der Grad der Verdeckung durch andere Caches bestimmt werden. Damit lässt sich bereits der Ambient-Occlusion-Term aus Gleichung (5.57) berechnen. Allerdings wären diese Schatten vollkommen unabhängig von dem tatsächlich in der Szene existierenden indirekten Licht, das in Form von Proxy-Lichtquellen repräsentiert wird. Es gilt also nicht herauszufinden, wie viel von der Hemisphäre eines Oberflächenpunktes durch andere Caches verdeckt wird, sondern wie viel der indirekten Proxy-Flächenlichtquelle des Caches durch andere Caches verdeckt wird (siehe Abbildung 5.23).

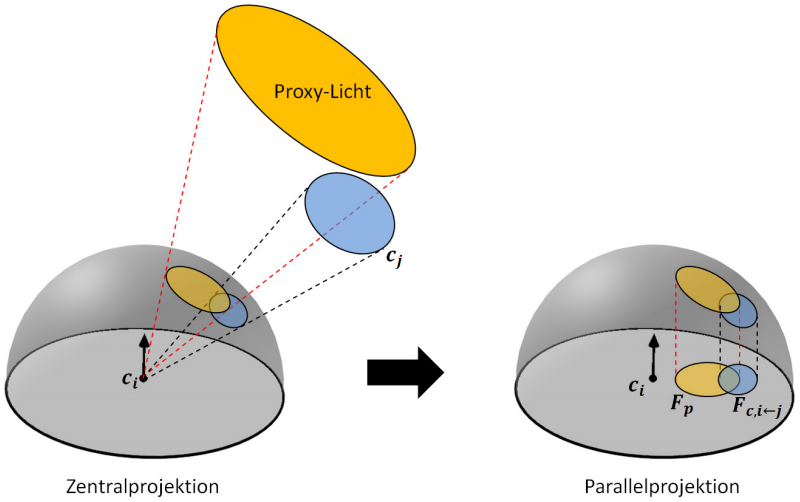

Abbildung 5.23: Im Gegensatz zum Ambient Occlusion ist beim LightSkin-Verfahren der Grad der Verdeckung der überlappenden Fläche der Projektionen der Proxy-Lichtquelle und des verdeckenden Caches von Bedeutung. Diese Überlappung lässt sich durch Nusselts Analogon [Nus28] auf eine Einheitskreisscheibe projizieren und bestimmen.

Die schnelle Berechnung der Verdeckung der indirekten Proxy-Lichtquelle kann nur approximativ erfolgen. Prinzipiell besteht das Problem, dass die relative Positionierung der Lichtquelle zu den Caches nun von Bedeutung ist, während dies für das Ambient Occlusion vernachlässigbar war. Da es zu prüfen gilt, wie viel der Projektion der Proxy-Lichtquelle des Caches durch andere Caches verdeckt wird, lässt sich das Problem in eine zweidimensionale Domäne übertragen, wenn eine passende Projektion gewählt wird. Im Zusammenhang mit den Formfaktoren erscheint hier Nusselts Analogon [Nus28] als hilfreich. Dieses besagt, dass der Formfaktor einer dreidimensionalen Fläche zu einem Oberflächenpunkt gleich der Projektion auf eine Kreisfläche ist. Diese Projektion wird in zwei Schritten durchgeführt: Zuerst wird die dreidimensionale Fläche auf die Fläche einer Einheitskugel zentralprojiziert und danach wird eine Parallelprojektion auf die Einheitskreisscheibe, um den Projektionspunkt, durchgeführt (siehe Abbildung 5.23). Damit lassen sich die Kreisscheiben der Caches auf die Oberfläche eines Einheitskreises projizieren. Allerdings erscheint die Bestimmung des Formfaktors für eine AABB, um die Lichtquelle in die zweidimensionale Domäne zu überführen, als ungeeignet bzw. rechenintensiv. Um diesem Umstand zu begegnen, wird die AABB in eine Bounding-Sphere gewandelt, da für diese die Bestimmung des Formfaktors einfacher möglich ist. Für die Umwandlung der AABB in eine Bounding-Sphere wird die längste Seite der Box durch zwei geteilt und als Radius für die Bounding-Sphere eingesetzt. Diese Approximation ist sehr grob, wenn die AABB sehr flach ausfällt, jedoch sollte man beachten, dass eine flächige Verteilung der VALs nicht zwangsläufig zu einer flachen AABB führt. Wenn die VALs zum Beispiel diagonal zu den Hauptachsen verteilt sind, wird die resultierende AABB ein hohes Volumen aufweisen, obwohl das indirekte Licht eine flache Ausbreitung besitzt. Die Flachheit der AABB lässt also nur bedingt Rückschlüsse auf die VAL-Verteilung zu, weswegen eine Approximation als Bounding-Sphere konsequent erscheint. Schlussendlich lassen sich der Formfaktor für die indirekte Proxy-Lichtquelle F_p und der Formfaktor für einen Cache F_c wie folgt berechnen (r_p bezeichnet den Radius der Bounding-Sphere der Proxy-Lichtquelle):

$$F_p = \frac{r_p^2 cos\vartheta_c}{r_p^2 + \|\boldsymbol{c} - \boldsymbol{x_p}\|^2}, \qquad F_{c,i\leftarrow j} = \frac{A_{disk,j} cos\vartheta_{c,j} cos\vartheta_{c,i}}{A_{disk,j} + \pi\|\boldsymbol{c_i} - \boldsymbol{c_j}\|^2}. \tag{5.59}$$

Damit stehen die Flächen der Projektionen nach Nusselts Analogon sowohl für die Proxy-Lichtquelle als auch für die Caches fest. Allerdings erlaubt diese Berechnung noch nicht die Bestimmung der Überlappung der beiden Projektionen. Hier ergibt sich das Problem, dass die Formen der Flächen nach der doppelten Projektion (Zentralprojektion auf Kugeloberfläche und Parallelprojektion auf Kreisscheibe) sehr komplex ausfallen können. Die Form der Projektion ergibt sich, wenn man jeden vom Cache aus sichtbaren Oberflächenpunkt $\boldsymbol{x}$ der anderen Caches bzw. der Proxy-Lichtquelle wie folgt projiziert:

$$Proj(\boldsymbol{x}, \boldsymbol{c}) = \left(\frac{\boldsymbol{x} - \boldsymbol{c}}{\|\boldsymbol{x} - \boldsymbol{c}\|} \cdot T(\boldsymbol{c}), \frac{\boldsymbol{x} - \boldsymbol{c}}{\|\boldsymbol{x} - \boldsymbol{c}\|} \cdot B(\boldsymbol{c})\right). \tag{5.60}$$

Wobei T die Tangente und B die Binormale im Cache berechnet. Die Berechnung der exakten Projektionsformen und die Bestimmung der Überlappung ist in Echtzeit nicht möglich, weswegen die tatsächliche Form der Projektion verworfen und stattdessen ein achsenausgerichtetes Quadrat als Form angenommen wird. Dieses Quadrat hat die gleiche Fläche wie die komplexe Form (siehe Abbildung 5.24). So vereinfacht sich die Bestimmung der Überlappungsfläche von den Projektionen der Proxy-Lichtquelle und des Caches deutlich. Um die Quadrate zu positionieren, reicht die Projektion der Zentren der Cache-Kreisscheiben und der Proxy-Kugellichtquelle: $Proj(\boldsymbol{c_j}, \boldsymbol{c_i})$ und $Proj(\boldsymbol{l_p}, \boldsymbol{c_i})$. Es hat sich gezeigt, dass die Approximation überzeugende Ergebnisse liefert. Wie sich die Approximation zur exakten Überlappungsberechnung verhält, ist für verschiedene Fälle in Abbildung 5.24 festgehalten. Der hier beschriebene Algorithmus lässt sich wie folgt in einer Implementierung zusammenfassen:

Listing 5.3: Algorithmus zur Verdeckungsrechnung (weiche Schatten).

```
for each Cache (c_i, x_p, σ_p, φ_p)
      a = 0
      (x_p, r_p) = ConvertProxyAABBtoSphere(x_p, σ_p)
      x_p,proj = Proj(x_p, c_i)          // Position für Quadrat-Proxy-Licht
      l_p,proj = sqrt(F_p(x_p, r_p))     // Seitenlänge für Quadrat-Proxy-Licht
      for each Cache (c_j ≠ c_i)
            c_j,proj = Proj(c_j, c_i)                  // Position für Quadrat-Cache
            l_c,j,proj = sqrt(F_c,i←j(c_j, A(c_j)))    // Seitenlänge für Quadrat-Cache
            a += OverlappingArea(x_p,proj, l_p,proj, c_j,proj, l_c,j,proj)
      end
      φ_p *= max(0, 1-a/F_p(x_p, r_p))
end
```

Dieser Algorithmus ist noch nicht vollständig, der noch fehlende Teil betrifft die Entfernungen der Caches zueinander und zur Lichtquelle. Im nächsten Unterkapitel wird auf dieses Thema detailliert eingegangen.

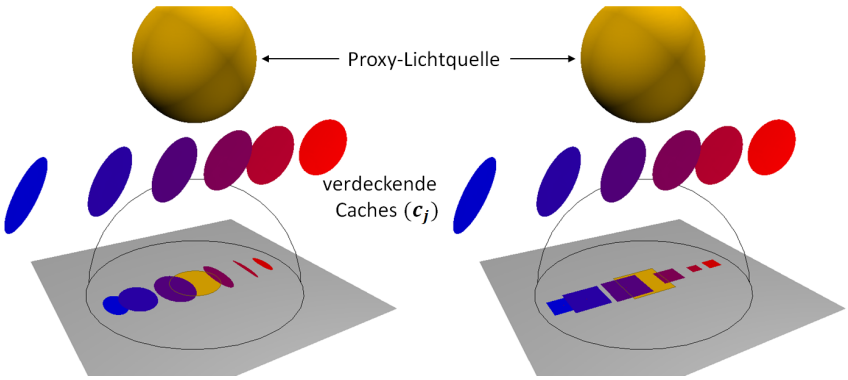

Abbildung 5.24: Approximation der komplexen Projektionsformen durch achsenausgerichtete Quadrate.

5.4.3 Vermeidung falscher Verdeckungsergebnisse

Gegenwärtig dämpfen Caches, die sich hinter der Proxy-Flächenlichtquelle befinden, fälschlicherweise den Lichtstrom der Lichtquelle. Um dieses Problem approximativ zu lösen, wird der Verdeckungsgrad a linear auf null skaliert, wenn der verdeckende Cache innerhalb der Bounding-Sphere der Proxy-Lichtquelle liegt (siehe auch Abbildung 5.25):

$$a' = \max(0, \frac{a}{r_p} \min\left(r_p, (c_j - x_p) \cdot \frac{c_i - x_p}{\|c_i - x_p\|}\right)). \tag{5.61}$$

Je näher ein Cache also zur Proxy-Lichtquelle steht, desto weniger stark dämpft dieser den Lichtstrom der Lichtquelle. Für den Fall, dass der Cache komplett hinter dem Zentrum der Lichtquelle liegt, dämpft er diese überhaupt nicht mehr.

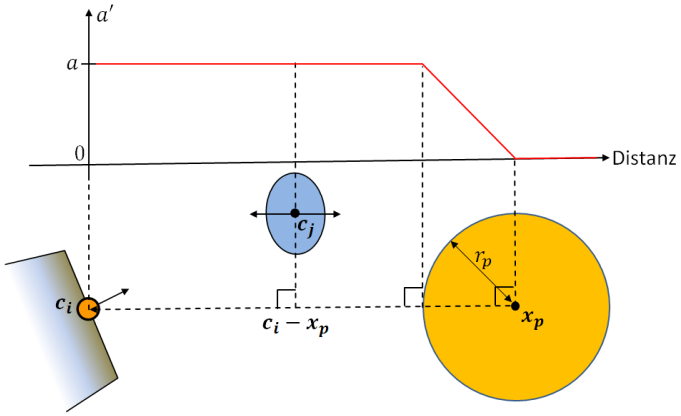

Abbildung 5.25: Abnahme des Verdeckungsgrades mit der Entfernung nach Gleichung (5.61).

Ein weiteres Problem existiert, weil verdeckende Objekte die Lichtquelle auch dann dämpfen, wenn sie nach der Projektion eigentlich hinter anderen Objekten liegen (fehlende Occluder-Fusion). So dämpfen Verdeckungsobjekte, die in einer Reihe zum schattierten Cache liegen, dessen Proxy-Lichtquelle überproportional. Eigentlich dürfen die Verdeckungsobjekte nur dann die Lichtquelle dämpfen, wenn sie die Nächsten zum Cache sind. Eine exakte Prüfung würde jedoch einen Z-Buffer aus Sicht des beleuchteten Caches erfordern. Dies kommt der Verwendung einer Shadow-Map für die Lichtquellen gleich und wurde wegen dem benötigten Rechenaufwand bereits ausgeschlossen. Dieses Problem besteht ganz analog für den Ambient-Occlusion-Ansatz von Bunnell [Bun05]. Deshalb schlägt dieser vor, die falsche doppelte Verdeckungsrechnung durch ein Multipass-Verfahren zu

korrigieren. Im ersten Pass wird die Verdeckung, wie bereits beschrieben, berechnet. Nach diesem Pass steht fest, welcher Cache wie stark verdeckt wird. Jetzt wird die Verdeckungsrechnung erneut ausgeführt, allerdings verdecken Caches, die selbst stark verdeckt werden, andere Caches nun weniger (umgekehrt proportional zum eigenen Verdeckungsgrad). Dies wird mehrmals wiederholt, bis das fertige Bild zur korrekten Lösung konvergiert ist. Dieser Ansatz ist rechenintensiv, da er mehrere Durchgänge erfordert, von denen jeder Durchgang verhältnismäßig teuer ist. In dieser Arbeit wird deshalb eine laufzeiteffizientere Methode vorgeschlagen, um die ausgeprägtesten Fälle dieser falschen Verdeckungsrechnung abzufangen:

Für jeden schattenempfangenden Cache $\boldsymbol{c_j}$ wird zusätzlich ein zweidimensionales Rechteck berechnet, welches alle projizierten Rechtecke der schattensendenden Caches kompakt zusammenfasst. Dabei wird jedoch nur der Anteil der Fläche der schattensendenden Caches berücksichtigt, der das Quadrat der Proxy-Lichtquelle überlappt (siehe Abbildung 5.26). Um den finalen Verdeckungsgrad zu bestimmen, wird die Summe der verdeckenden Quadrate mit der Fläche des zweidimensionalen Rechtecks verglichen. Ist die Summe größer als die Fläche, müssen viele verdeckte Caches vorhanden sein und die Fläche des Rechtecks wird als Verdeckungsgrad genutzt. Ist die Summe kleiner, wird diese für die Verdeckung verwendet. Dieses Verfahren ist nicht optimal, aber günstiger als das Multipass-Verfahren von Bunnell und verbessert ebenfalls das Problem der doppelten Verdeckung.

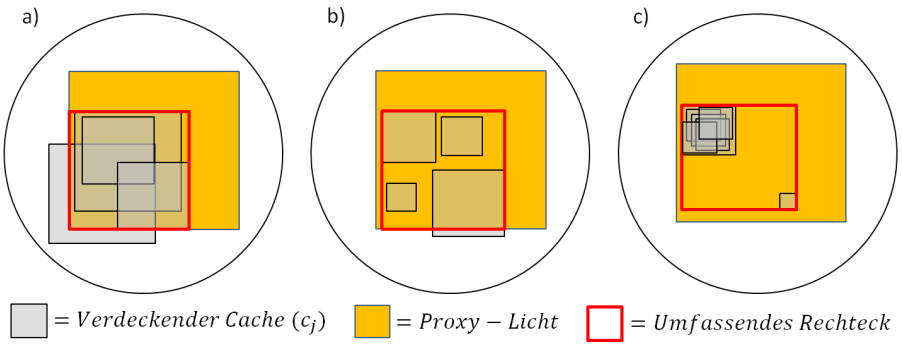

Abbildung 5.26: Prinzip zur Vermeidung doppelter Verdeckungen. In (a) wird ein Fall gezeigt, bei dem die Summe der Quadratflächen größer ist als die Rechteckfläche, also wird die Rechteckfläche als Verdeckungsgrad genutzt. (b) zeigt einen Fall, bei dem die Summe der Quadratflächen kleiner ist als die Rechteckfläche, also wird diese Summe für die Verdeckungsrechnung genutzt. (c) gibt ein Beispiel wieder, bei dem der Ansatz fehlschlägt: Der berechnete Verdeckungsgrad würde in diesem Fall zu groß ausfallen.

5.4.4 Diskussion

Der hier vorgestellte neue Schattierungsalgorithmus des LightSkin-Verfahrens ist ein optionaler Ansatz, um die Qualität der indirekten Beleuchtung zu verbessern. Die Effizienz des Verfahrens wird durch vier Ansätze sichergestellt:

1. Die Blockierung des indirekten Lichts wird nur für die Caches der Szene berechnet.
2. Es wird nicht jedes indirekte VAL für den Sichtbarkeitstest genutzt, sondern nur die zusammenfassende Proxy-Lichtquelle.
3. Die Szenengeometrie wird vereinfacht durch Kreisscheiben repräsentiert.
4. Es wird ein schneller approximativer Ansatz für den Verdeckungstest genutzt.

Betrachtet man die Laufzeitkomplexität des Verfahrens (zusammen mit der indirekten Beleuchtung aus Kapitel 5.3), so ergibt sich $O(mn + m^2)$, wobei n die Anzahl der VALs aus der RSM und m die

Anzahl der Caches bezeichnet. Auf den ersten Blick scheint kein großer Vorteil gegenüber Shadow-Mapping-Verfahren zu bestehen, für die in Kapitel 5.4 eine Laufzeitkomplexität von $O(pn + kn)$ ermittelt wurde (k ist hier die Menge der sichtbaren Oberflächenpunkte vom Betrachter gesehen und p von der Lichtquelle gesehen). Der Vorteil liegt zum einen in den geringeren konstanten Kosten und zum anderen in der Tatsache, dass $m \ll p$ und $m \ll k$. Die Interpolation auf Basis der Caches wurde extra so entworfen, dass bereits eine möglichst geringe Menge von diesen ausreicht, um überzeugende Ergebnisse zu liefern. Typischerweise reichen für Modelle wie den Stanford-Dragon, -Buddha oder -Bunny bereits weniger als 200 Caches aus. Szenen wie die Crytek-Sponza-Scene lassen sich bereits mit 3.000 Caches ausreichend approximieren. Geht man davon aus, dass jede Szene mit maximal 16.000 Caches ausreichend approximiert werden kann und dass der Framebuffer eine Full-HD-Auflösung von 1920x1080 Bildpunkten hat, so ergeben sich bei der Verwendung von einer 128x128 Texel großen RSM 512 Millionen Berechnungen beim LightSkin-Verfahren. Für ein Shadow-Mapping-Verfahren ergeben sich bereits aus dem Term kn mehr als 33 Milliarden Berechnungen, ohne dass eine einzige Shadow-Map berechnet wurde. Man erkennt, dass die Laufzeitkosten bei dem LightSkin-Verfahren deutlich geringer sind. In Kapitel 6.1.6 werden weitere Optimierungen vorgestellt, die die Laufzeitkomplexität weiter verringern können.

Mit dem Gewinn von Berechnungszeit geht ein Verlust von Qualität einher. Allerdings ist der qualitative Verlust in der Wahrnehmung des Betrachters relativ gering und deshalb durchaus vertretbar. Im Folgenden sollen einige Einschränkungen beschrieben werden, die die Qualität des indirekten Schattens betreffen.

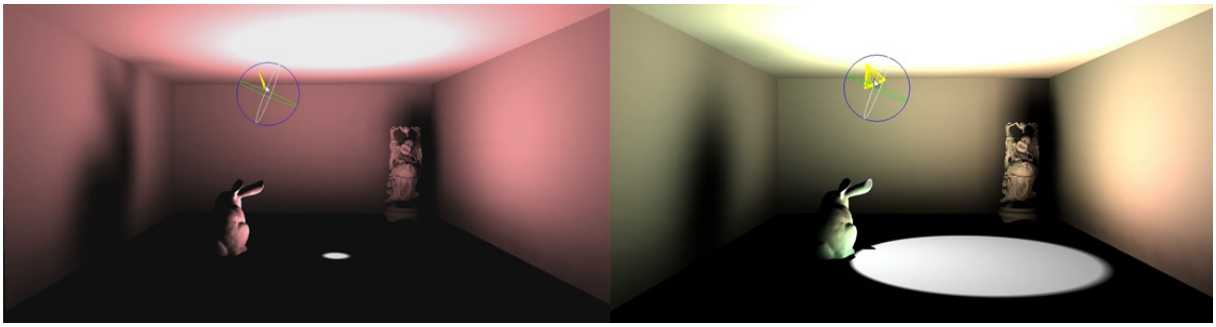

Abbildung 5.27: Adaptive Schatten in Abhängigkeit der Größe der indirekten Lichtquelle (Reflexion vom Boden). Scharfe Schatten können mit dem Verfahren nicht abgebildet werden, selbst wenn die indirekte Lichtquelle sehr kompakt ist (links). Die Leuchtdichte der indirekten Reflexionen wurde verstärkt, damit die Konturen besser erkennbar sind.

Durch die Tatsache, dass die Verdeckung der indirekten Proxy-Lichtquelle durch andere Elemente nur für die Caches bestimmt wird, ergibt sich die Einschränkung, dass die Schatten keine scharfen Konturen annehmen können. Eine scharfe Kontur ist nur dann möglich, wenn die Caches sehr dicht verteilt sind, was aus Laufzeitgründen nicht sinnvoll ist. Hierdurch wird selbst bei einer räumlich kompakten indirekten Lichtquelle ein nicht korrekter weicher Schatten erzeugt (siehe Abbildung 5.27). Dieser Nachteil kann jedoch auch als Vorteil ausgelegt werden, da für die Verdeckungsrechnung die Szene durch Kreisscheiben approximiert wird. Diese Approximation würde bei scharfen Schatten offensichtlich werden und wäre demnach kontraproduktiv. Allerdings hat die Approximation der Szene durch Kreisscheiben wiederum den Nachteil, dass lange schmale Objekte wie Masten oder Äste nur sehr schlecht approximiert werden können. Für diese Modelltypen bilden polygonale analytische Formfaktoren eine denkbare Alternative zu den Kreisformfaktoren.

Eine weitere Einschränkung resultiert aus der Zusammenfassung des indirekten Lichts zu einer Proxy-Kugellichtquelle mit konstantem Lichtstrom. Hierdurch wird für die Schattierung davon ausgegangen, dass das indirekte Licht homogen ist, wodurch verschiedenfarbige Schatten nicht abgebildet werden können. Darüber hinaus sind wegen der Repräsentation des indirekten Lichts durch nur *eine* Proxy-Lichtquelle keine Mehrfachschlagschatten möglich und die Richtung des indirekten Schattens ist nicht korrekt, wenn das direkte Licht auf mehrere Bereiche verteilt wird. Mit zunehmender Verbreitung des indirekten Lichts nähert sich die Schattierung einer Ambient-Occlusion-Lösung.

Trotz dieser Einschränkungen ist der optische Gewinn der Schattenberechnung deutlich erkennbar. Im Vergleich zu einer Ambient-Occlusion-Lösung bietet das hier vorgestellte Verfahren dynamische Schlagschatten und volle Interaktivität zur approximativen indirekten Beleuchtung. Abbildung 5.28 zeigt zwei Beispiele von Szenen mit und ohne Schatten.

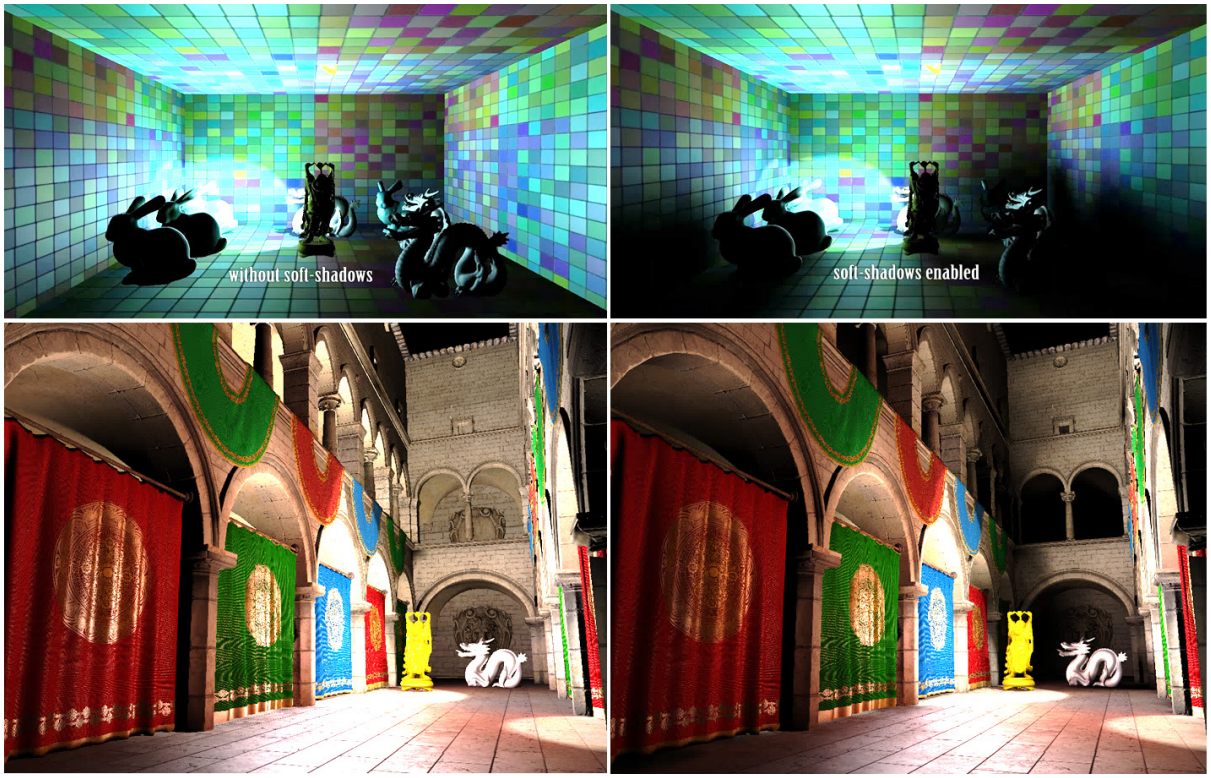

Abbildung 5.28: Vergleich indirekte Beleuchtung ohne Schatten (links) und mit Schatten (rechts) (die indirekten Reflexionen wurden für den Druck verstärkt).

5.5 Subsurface-Scattering

In diesem Kapitel wird eine Lösung zur Echtzeitdarstellung von transluzenten Objekten mit hoher Lichtstreuung unter der Oberfläche mit Hilfe eines neuen Interpolationsverfahrens vorgestellt. Objekte mit diesen Eigenschaften sind in Abbildung 5.29 exemplarisch dargestellt. Für die Materialien dieser Objekte gilt, dass sie nicht durch ein einfaches Reflexionsmodell auf Basis der BRDF simuliert werden können. Die BRDF bildet zwar diese Prozesse unter der Oberfläche ab, aber ein dediziertes Modell wird nötig, wenn die Streuungsprozesse unter der Oberfläche dazu führen, dass der Einfluss des Lichts über die geometrische Dimension eines sichtbaren Oberflächenpixels hinausgeht. Dies ist dann der Fall, wenn Licht in einem Punkt $\boldsymbol{x_i}$ eintrifft und (teilweise) in einem anderen Punkt $\boldsymbol{x_o}$ austritt. Der Weg des Lichts innerhalb des Mediums (unter der Oberfläche des Objekts)

wird durch die Gleichungen des Volume-Renderings beschrieben, die in Kapitel 2.1.5 vorgestellt wurden. Betrachtet man nur die Oberfläche eines transluzenten Objekts, beschreibt die erweiterte Rendergleichung für das Subsurface-Scattering (SSS) (2.28) aus Kapitel 2.1.6, wie die Leuchtdichte für einen Oberflächenpunkt $\boldsymbol{x}$ in Richtung $\boldsymbol{\omega}$ berechnet werden kann:

$$\mathrm{L_o}(\boldsymbol{x}, \boldsymbol{\omega}) = \int_A \int_{\Omega^+} f_{ss}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{x_i}, \boldsymbol{\omega_i}) L_i(\boldsymbol{x_i}, \boldsymbol{\omega_i}) (N(\boldsymbol{x_i}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i} dA. \tag{5.62}$$

Die BSSRDF f_{ss} beschreibt das Verhältnis zwischen dem im Punkt $\boldsymbol{x_i}$ eintreffenden differentiellen Lichtstrom aus Richtung $\boldsymbol{\omega_i}$ zur austretenden differentiellen Leuchtdichte im Punkt $\boldsymbol{x}$ in Richtung $\boldsymbol{\omega}$. Sie bildet also die Reflexion im Punkt, sowie die Ausbreitung des Lichts im Medium ab.

Je nach Material kommen verschiedene approximative Modelle für die BSSRDF zum Einsatz. Konzentriert man sich auf die Streuungsprozesse innerhalb des Materials, wird häufig zwischen Single-Scattering- und Multiple-Scattering-Phänomenen unterschieden (siehe Kapitel 2.1.5). Beim Single Scattering wird davon ausgegangen, dass das Licht innerhalb des Mediums nur einmalig gestreut wird. Demgegenüber wird beim Multiple Scattering von sehr vielen Streuungsereignissen ausgegangen (siehe Abbildung 2.6 in Kapitel 2.1.5). Die resultierende Leuchtdichte eines Oberflächenpunkts eines transluzenten Objekts lässt sich durch die separate Betrachtung von Single Scattering und Multiple Scattering berechnen. Die jeweiligen Anteile der Leuchtdichte werden einfach summiert. Für Materialien mit geringen Streuungseigenschaften, wie zum Beispiel leichter Nebel, überwiegt der Single-Scattering-Anteil, während bei stark streuenden Materialien mit einem hohen Albedo das Multiple Scattering überwiegt.

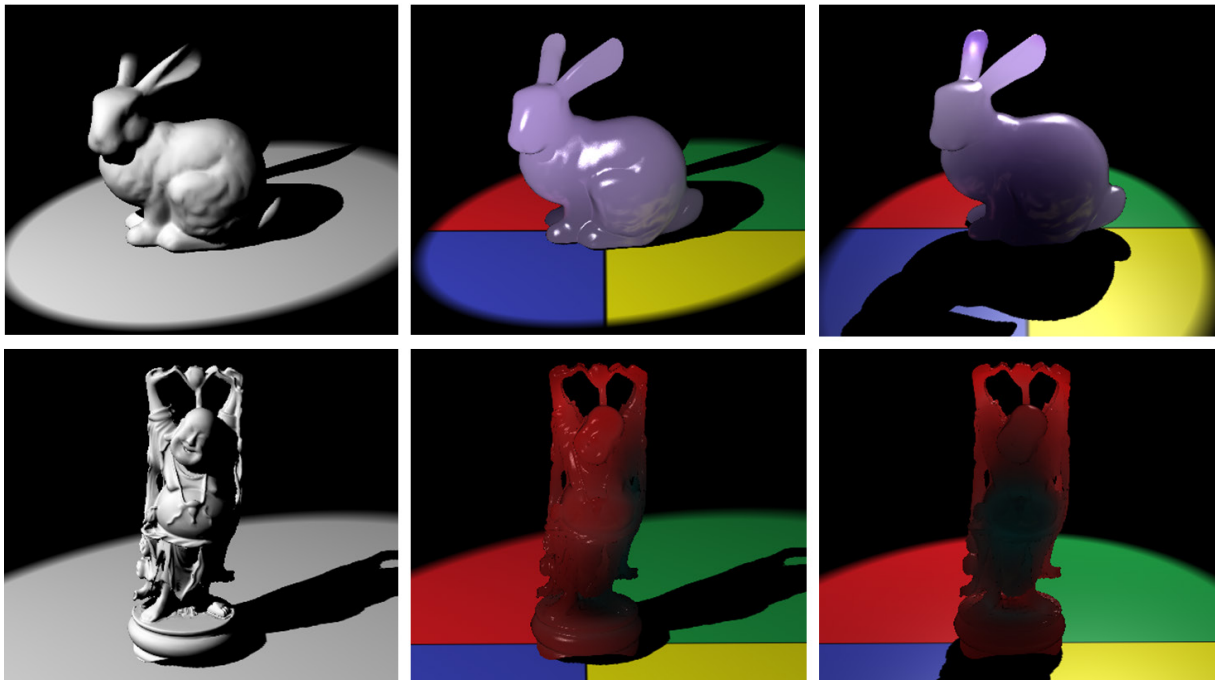

Abbildung 5.29: Zwei Subsurface-Scattering-Materialien mit verschiedener Dichte. Die Objekte wurden mit dem LightSkin-Verfahren gerendert (ca. 200 Caches pro Objekt). Man erkennt deutlich, wie das Licht innerhalb des Objekts gestreut wird.

Die im Folgenden vorgestellte neue Subsurface-Scattering-Interpolationsmethode kann nur Materialien approximieren, bei denen der Multiple-Scattering-Term überwiegt. Diese Materialien zeichnen

sich dadurch aus, dass die Richtungseigenschaften des Lichts mit jedem Streuungsereignis abnehmen. Dies erlaubt nach Stam [Sta95], den Transport des Lichts innerhalb des Modells als einen Diffusionsprozess zu formulieren. Typische Materialien dieser Art sind zum Beispiel Wachs, Marmor, Jade oder Milchglas.

Das in dieser Arbeit entwickelte neue Interpolationsverfahren basiert wieder auf der Verwendung von Proxy-Lichtquellen. Zum Verständnis dieses neuen Verfahrens ist es erforderlich, dass das Dipol-Diffusions-Modell von Jensen et al. [JML*01] zur Abbildung der BSSRDF bekannt ist. Deswegen wird dieses Modell in den nächsten beiden Unterkapiteln erklärt. Es wird gezeigt, wie sich die BSSRDF für stark streuende, homogene Materialien mit einem hohen Albedo approximieren lässt. Anschließend wird auf Basis dieser BSSRDF das neue Proxy-Lichtquellen-Interpolationsverfahren entwickelt, das mit der gleichen losen Cache-Verteilung funktioniert, wie das bereits vorgestellte Interpolationsverfahren für indirekte Reflexionen aus Kapitel 5.3.

5.5.1 Multiple Scattering als Diffusionsprozess

In Kapitel 2.1.5 wurde bereits darauf hingewiesen, dass für die Ausbreitung des Lichts innerhalb eines Mediums vier Prozesse von Bedeutung sind:

1. Absorption,
2. In-Scattering,
3. Out-Scattering und
4. Emission.

In der folgenden Betrachtung soll die Emission unbeachtet bleiben, da keine Materialien beschrieben werden, die selbstleuchtend sind. Stellt man die Streuung und die Absorption innerhalb eines Mediums in einen differentiellen Zusammenhang, so erhält man die Strahlungstransportgleichung [Cha50] (Equation of Radiative Transport, hier ohne Emissionsterm; die Integration dieser Gleichung führt zum Volume-Rendering-Integral (2.27) aus Kapitel 2.1.5):

$$(\boldsymbol{\omega} \cdot \boldsymbol{\nabla}) L(\boldsymbol{x}, \boldsymbol{\omega}) = -\sigma_t(x, \boldsymbol{\omega}) L(\boldsymbol{x}, \boldsymbol{\omega}) + \sigma_s(\boldsymbol{x}, \boldsymbol{\omega}) \int_{\boldsymbol{\Omega}} f_p(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) L(\boldsymbol{x}, \boldsymbol{\omega_i}) d\boldsymbol{\omega_i}. \tag{5.63}$$

Die Koeffizienten wurden bereits in Kapitel 2.1.5 beschrieben. Der Term $-\sigma_t L(\boldsymbol{x}, \boldsymbol{\omega})$ bildet die differentielle Dämpfung, die durch das Out-Scattering und die Absorption entsteht, ab (siehe Gleichung (2.25)). Das Integral beschreibt, wie viel Licht in den Pfad entlang $\boldsymbol{\omega}$ hineingestreut wird (In-Scattering). f_p ist die Phasenfunktion, die angibt, mit welcher Wahrscheinlichkeit das eintreffende Licht aus Richtung $\boldsymbol{\omega_i}$ in Richtung $\boldsymbol{\omega}$ gestreut wird. Da die Phasenfunktion dem Prinzip der Energieerhaltung gehorcht, gilt für sie $\int_{\boldsymbol{\Omega}} f_p(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) d\boldsymbol{\omega_i} = 1$. Die Funktion ist häufig nur abhängig von dem relativen Winkel zwischen $\boldsymbol{\omega}$ und $\boldsymbol{\omega_i}$ und wird deshalb nur durch zwei Parameter bestimmt: $f_p(\boldsymbol{x}, \boldsymbol{\omega} \cdot \boldsymbol{\omega_i})$. Wie die Phasenfunktion gerichtet ist, wird durch den Anisotropiefaktor g beschrieben:

$$g = \int_{\boldsymbol{\Omega}} (\boldsymbol{\omega} \cdot \boldsymbol{\omega_i}) f_p(\boldsymbol{x}, \boldsymbol{\omega} \cdot \boldsymbol{\omega_i}) d\boldsymbol{\omega_i}. \tag{5.64}$$

Abbildung 5.30 zeigt die analytische Phasenfunktion von Henyey und Greenstein [HG41], die häufig in der Computergrafik Anwendung findet. Ist $g \to -1$, wird das Licht hauptsächlich zurück in Richtung $\boldsymbol{\omega}$ gestreut, bei $g \to 1$ wird es nach vorne gestreut und bei $g = 0$ ist die Streuungswahrscheinlichkeit gleichmäßig über alle Richtungen verteilt (isotrope Phasenfunktion).

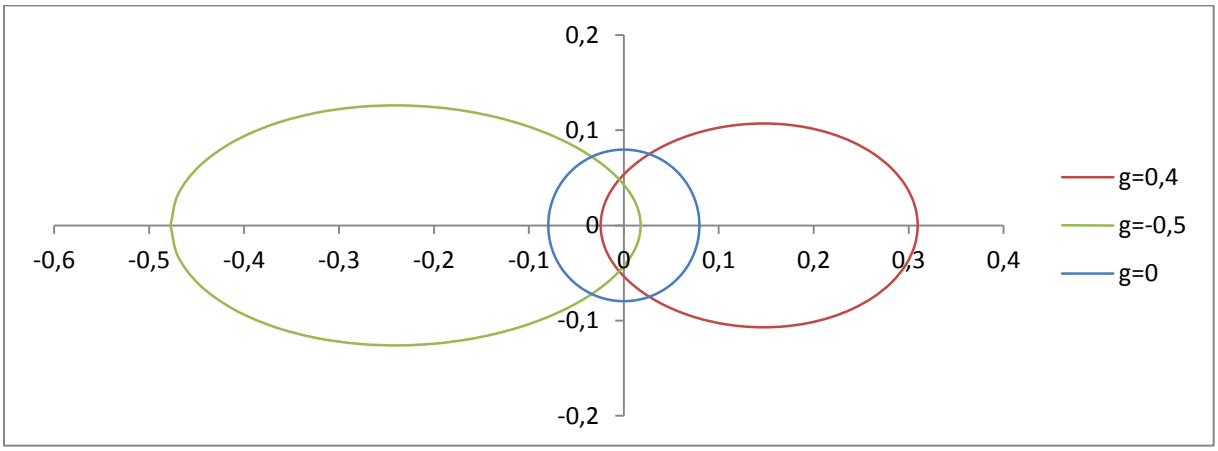

Abbildung 5.30: Henyey-Greenstein-Phasenfunktion für verschiedene Anisotropiefaktoren g. $\boldsymbol{\omega}$ zeigt in Richtung der positiven x-Achse.

Die Transportgleichung (5.63) ist sehr komplex und kann nur approximativ gelöst werden. Stam [Sta95] hat aus der Strahlungstransportgleichung die klassische Diffusionsgleichung $\frac{\partial c}{\partial t} = D\nabla^2 c$ nach Fick [Fic55] entwickelt, um damit das Ausbreitungsverhalten des Lichts in einem Medium zu beschreiben. Hierfür werden unter anderem die folgenden zwei Vereinfachungen vorgenommen:

1. Das Medium ist homogen und
2. die Phasenfunktion f_p ist isotrop ($f_p = \frac{1}{4\pi}$).

Durch die Homogenität des Mediums sind die Phasenfunktion und die Koeffizienten nicht mehr ortsabhängig. Die Verwendung einer isotropen Phasenfunktion ist für Materialien mit hohen Streuungseigenschaften keine direkte Einschränkung. Denn durch jedes Streuungsereignis erhält die Phasenfunktion eine zunehmend isotrope Ausprägung. Dieser Zusammenhang wurde von Yanovitskij [Yan11] beschrieben, der eine veränderte Form der Henyey-Greenstein-Phasenfunktion vorgeschlagen hat, die von der Anzahl der Streuungsereignisse abhängt:

$$f_p(\boldsymbol{\omega} \cdot \boldsymbol{\omega_i}, n) = \frac{1 - g^{2n}}{4\pi\left(1 + g^{2n} - 2g|g^{n-1}|(\boldsymbol{\omega} \cdot \boldsymbol{\omega_i})\right)^{\frac{3}{2}}}. \tag{5.65}$$

n entspricht hier der Anzahl der Streuungsereignisse. Abbildung 5.31 zeigt, wie sich die Phasenfunktion mit zunehmender Streuung verändert.

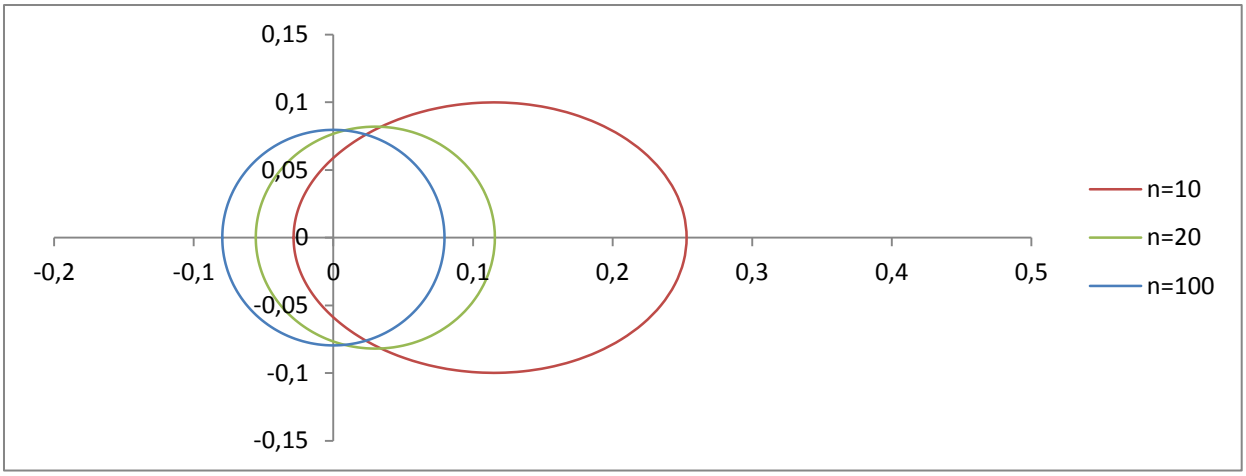

Abbildung 5.31: Anpassung der Phasenfunktion durch wiederholte Streuung nach Gleichung (5.65) (g=0,9). Je mehr Streuungsereignisse auftreten, desto isotroper wird die Phasenfunktion.

Mit dem Hintergrundwissen der zunehmenden Isotropie der Phasenfunktion bei häufiger Streuung kann man für stark streuende Materialien eine anisotrope Phasenfunktion durch die isotrope Phasenfunktion approximieren, wenn man einen angepassten Streuungskoeffizienten nutzt:

$$\sigma_s' = (1-g)\sigma_s, \qquad \sigma_t' = \sigma_s' + \sigma_a. \tag{5.66}$$

Der angepasste (reduzierte) Streuungskoeffizient σ_s' kann intuitiv interpretiert werden. Ist g groß, dann wird das meiste Licht nach vorne gestreut und σ_s' wird klein. Da σ_s' die Wahrscheinlichkeit eines Streuungsereignisses wiedergibt, kann das Licht tiefer in das Objekt wandern, bevor es zu einem Streuungsereignis kommt. Analog führt ein negativer Anisotropiefaktor dazu, dass σ_s' größer wird. Also wird das Licht sehr häufig gestreut und kann deshalb nicht tief in das Material eindringen.

Es ist nun ersichtlich, dass durch die häufige Streuung die ursprüngliche Ausrichtung des Lichts, beim Eintreffen ins Medium, keine große Relevanz für die Verbreitung innerhalb des Mediums hat. Dies erlaubt die Repräsentation der Leuchtdichte für einen Punkt $\boldsymbol{x}$ in Richtung $\boldsymbol{\omega}$ im Medium durch eine einfache Approximation:

$$L(\boldsymbol{x},\boldsymbol{\omega}) = \frac{1}{4\pi}\phi(\boldsymbol{x}) + \frac{3}{4\pi}\boldsymbol{\omega}\cdot\boldsymbol{E}(\boldsymbol{x}). \tag{5.67}$$

ϕ bezeichnet hier die skalare Beleuchtungsstärke und $\boldsymbol{E}$ ist die vektorielle Beleuchtungsstärke, die sich wie folgt zusammensetzen:

$$\phi(\boldsymbol{x}) = \int_{\boldsymbol{\Omega}} L(x,\boldsymbol{\omega})d\boldsymbol{\omega}\,, \qquad \boldsymbol{E}(\boldsymbol{x}) = \int_{\boldsymbol{\Omega}} L(x,\boldsymbol{\omega})\boldsymbol{\omega}d\boldsymbol{\omega}. \tag{5.68}$$

Die Leuchtdichte L kann durch die massive Streuung im Medium durch einen konstanten Term $\frac{1}{4\pi}\phi(\boldsymbol{x})$ und einen linearen Richtungsterm $\frac{3}{4\pi}\boldsymbol{\omega}\cdot\boldsymbol{E}(\boldsymbol{x})$ approximiert werden. Ersetzt man die Leuchtdichte in der Strahlungstransportgleichung (5.63) durch die approximativen Terme aus Gleichung (5.67) und integriert die Gleichung über $\boldsymbol{\omega}$, so ergibt sich der folgende Zusammenhang für die vektorielle Bestrahlungsstärke aus Gleichung (5.67) (siehe [Ish78] für die Herleitung):

$$\boldsymbol{E}(\boldsymbol{x}) = -\frac{\boldsymbol{\nabla}\phi(\boldsymbol{x})}{3\sigma_t'}. \tag{5.69}$$

Die richtungsabhängige Ausbreitung der vektoriellen Beleuchtungsstärke $\boldsymbol{E}$ ist also proportional zum negativen Gradienten der skalaren Beleuchtungsstärke. Dies bedeutet nichts anderes, als dass das Licht von Bereichen mit hoher energetischer Dichte, in Bereiche mit niedriger energetischer Dichte wandert. Es findet also ein Diffusionsprozess statt.

Die klassische Form der Diffusionsgleichung ergibt sich, indem man das Integral über alle Richtungen $\boldsymbol{\omega}$ von der Strahlungstransportgleichung bildet (f_p ist, wie oben bereits erwähnt, isotrop):

$$\int_{\boldsymbol{\Omega}} (\boldsymbol{\omega} \cdot \nabla) L(\boldsymbol{x}, \boldsymbol{\omega}) d\boldsymbol{\omega} = \int_{\boldsymbol{\Omega}} (-\sigma_t L(\boldsymbol{x}, \boldsymbol{\omega}) + \sigma_s \int_{\boldsymbol{\Omega}} f_p(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) L(\boldsymbol{x}, \boldsymbol{\omega_i}) d\boldsymbol{\omega_i}) d\boldsymbol{\omega}\,, \tag{5.70}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{E}(\boldsymbol{x}) = -\sigma_t \phi(\boldsymbol{x}) + \sigma_s \phi(\boldsymbol{x}) = -\sigma_a \phi(\boldsymbol{x}),$$

und für dieses die vektorielle Bestrahlungsstärke aus Gleichung (5.69) einsetzt:

$$\frac{1}{3\sigma_t'} \nabla^2 \phi(\boldsymbol{x}) = \sigma_a \phi(\boldsymbol{x}). \tag{5.71}$$

Für einfache Fälle kann man die Diffusionsgleichung analytisch lösen. Eine analytische Lösung, die für die Entwicklung der BSSRDF auf Basis der Dipol-Diffusion von Bedeutung ist, ist die Berechnung der Ausbreitung des Lichts, ausgehend von einer Punktlichtquelle $(\boldsymbol{x_d}, \phi_d)$ zu einem Punkt $\boldsymbol{x}$ innerhalb eines unendlichen, homogenen Mediums ($\sigma_{tr} = \sqrt{3\sigma_a \sigma_t'}$):

$$\phi(\boldsymbol{x}, \boldsymbol{x_d}) = \frac{3\sigma_t' \phi_d}{4\pi} \frac{e^{-\sigma_{tr}\|x - x_d\|}}{\|\boldsymbol{x} - \boldsymbol{x_d}\|}. \tag{5.72}$$

5.5.2 Subsurface-Scattering als Dipol-Diffusionsprozess

Mit Hilfe der Gleichung (5.72) kann das Dipol-Diffusionsmodell von Jensen et al. [JML*01] für das Subsurface-Scattering hergeleitet werden. Bei dem Subsurface-Scattering interessiert man sich nur für die Berechnung der Leuchtdichte auf der Oberfläche des Modells (siehe Gleichung (5.62)). Diese Oberfläche ist durch die analytische Lösung der Diffusionsgleichung noch nicht berücksichtigt, denn diese geht von einem unendlichen Medium aus. Es muss also eine Randbedingung formuliert werden. Idealisiert kann für einen Oberflächenpunkt $\boldsymbol{x_s}$ eines Modells angenommen werden, dass dieser auf einer unendlichen Fläche liegt, die zwei unendliche Halbräume voneinander trennt. Ein Halbraum liegt über und einer unterhalb der Modelloberfläche. Um die Leuchtdichte auf der Oberfläche (Trennfläche) zu berechnen, muss berücksichtigt werden, dass aus dem Halbraum außerhalb des Modells kein Licht in das Modell einstrahlt (dies ist in Abbildung 5.32 veranschaulicht).

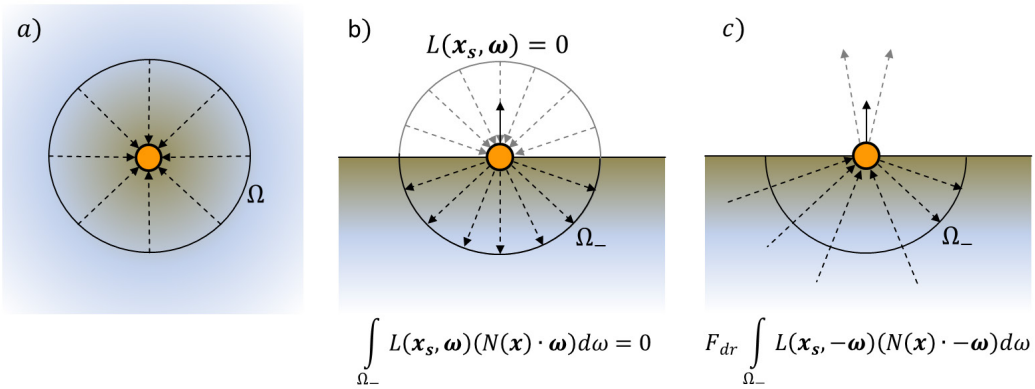

Abbildung 5.32: Grenzbedingung für Punkte auf der Trennoberfläche. (a) zeigt, dass Punkte in einem unendlichen Medium von allen Seiten Licht empfangen können. Liegt ein Punkt, wie in (b) gezeigt, auf der Oberfläche, so gelangt von der äußeren Hemisphäre kein Licht in das Material. In (c) haben beide Halbräume verschiedene Brechungsindizes, so kommt es zu Reflexionen unter der Materialoberfläche (innerhalb des Materials). Wie viel Licht reflektiert wird, wird durch den diffusen Fresnel-Reflexionsterm F_{dr} bestimmt.

Somit muss die folgende Bedingung gelten:

$$\int_{\Omega_-} L(\boldsymbol{x_s}, \boldsymbol{\omega})(N(\boldsymbol{x}) \cdot \boldsymbol{\omega}) d\boldsymbol{\omega} = 0. \tag{5.73}$$

Dieser Zusammenhang führt dazu, dass die Energie des Lichts in einem Punkt innerhalb des Modells, der zum Modellrand hinbewegt wird, schwächer wird, denn je näher der Punkt zum Rand kommt, desto weniger Energie erhält er aus der Richtung des Randes. Für die Lichtverteilung innerhalb des Modells wurde wegen der massiven Streuung von einer vereinfachten Abbildung der Leuchtdichte ausgegangen, die in Gleichung (5.67) beschrieben ist. Setzt man diese approximative Leuchtdichte in das Integral (5.73) ein und substituiert $\boldsymbol{E}(x)$ durch die Gleichung (5.69), erhält man die folgende vereinfachte Bedingung (die mathematische Herleitung ist in [XS06] zu finden):

$$\phi(\boldsymbol{x_s}) - \frac{2}{3\sigma_t'}(N(\boldsymbol{x_s}) \cdot \boldsymbol{\nabla})\phi(\boldsymbol{x_s}) = 0. \tag{5.74}$$

Diese Grenzbedingung berücksichtigt noch nicht verschiedene Brechungsindizes für die beiden Halbebenen. Für diesen Fall muss der Fresnel-Term einbezogen werden (siehe Kapitel 2.1.4), denn ein Teil des Lichts wird unter der Oberfläche wieder in das Material zurückreflektiert. Wie groß der Anteil des in das Material zurückreflektierten Lichts ist, wird durch den durchschnittlichen diffusen Fresnel-Term für Reflexionen F_{dr} gesteuert (siehe ebenfalls Abbildung 5.32c):

$$\int_{\Omega_-} L(\boldsymbol{x_s}, \boldsymbol{\omega})(N(\boldsymbol{x}) \cdot \boldsymbol{\omega}) d\boldsymbol{\omega} = F_{dr} \int_{\Omega_-} L(\boldsymbol{x_s}, -\boldsymbol{\omega})(N(\boldsymbol{x}) \cdot -\boldsymbol{\omega}) d\boldsymbol{\omega}. \tag{5.75}$$

Der durchschnittliche diffuse Reflexions-Fresnel-Term ergibt sich wie folgt (n ist der Brechungsindex des SSS-Materials):

$$F_{dr} = \int_{2\pi} F_r(n, N(\boldsymbol{x_s}) \cdot \boldsymbol{\omega})(N(\boldsymbol{x_s}) \cdot \boldsymbol{\omega}) d\boldsymbol{\omega}. \tag{5.76}$$

Eine Lösung für dieses Integral muss nicht berechnet werden, stattdessen kann die Approximation, die von Egan und Hilgeman [EH79] genutzt wird, verwendet werden:

$$F_{dr} = -\frac{1{,}44}{n^2} + \frac{0{,}71}{n} + 0{,}0636n + 0{,}668. \tag{5.77}$$

Unter Berücksichtigung der Reflexionen nach „innen“ durch die verschiedenen Brechungsindizes ergibt sich nun eine neue Grenzbedingung, die der Gleichung (5.74) sehr ähnlich ist, jedoch die Brechungsindizes miteinbezieht:

$$\phi(\boldsymbol{x_s}) - \frac{2A}{3\sigma_t'}(N(\boldsymbol{x_s}) \cdot \boldsymbol{\nabla})\phi(\boldsymbol{x_s}) = 0, \quad A = \frac{1 + F_{dr}}{1 - F_{dr}}. \tag{5.78}$$

Eine analytische Lösung für diese Grenzbedingung zu berechnen, ist relativ aufwändig [JML*01]. Man kann aber eine alternative approximative Bedingung formulieren, wenn man davon ausgeht, dass das Licht, welches das Material entlang der Normalen verlässt, linear abnimmt (siehe Abbildung 5.33).

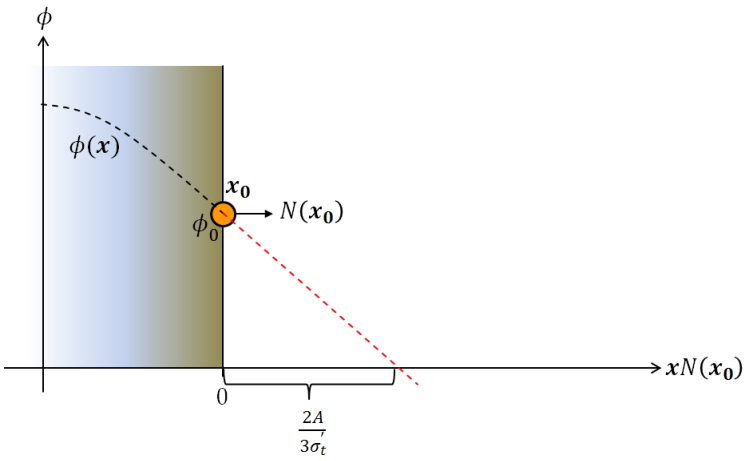

Abbildung 5.33: Approximative Grenzbedingung durch lineare Extrapolation.

In diesem Fall kann man von einer einfachen linearen Extrapolation ausgehen [Don06]:

$$\phi_o - \frac{2A}{3\sigma_t'}(N(\boldsymbol{x_0}) \cdot \boldsymbol{\nabla})\phi(\boldsymbol{x_0}) = 0 \;\rightarrow\; (N(\boldsymbol{x_0}) \cdot \boldsymbol{\nabla})\phi(\boldsymbol{x_0}) = \frac{3\sigma_t'}{2A}\phi_o. \tag{5.79}$$

ϕ_o entspricht der skalaren Beleuchtungsstärke in einem Oberflächenpunkt $\boldsymbol{x_0} \in \forall \boldsymbol{x_s}$. Für die Extrapolation des Lichts gilt nun, dass

$$\phi(\boldsymbol{x_0} + \Delta\boldsymbol{c}) = \phi_o - \frac{3\sigma_t'}{2A}\phi_o(\Delta\boldsymbol{c} \cdot N(\boldsymbol{x_0})) \quad \rightarrow \quad \phi\left(\boldsymbol{x_0} - \frac{2A}{3\sigma_t'}N(\boldsymbol{x_0})\right) = 0. \tag{5.80}$$

Anschaulich sagt diese Formel aus, dass die skalare Beleuchtungsstärke null sein muss für einen Punkt mit einer Entfernung von $\frac{2A}{3\sigma_t'}$ zur Grenzoberfläche. Diese vereinfachte Randbedingung kann durch die Verwendung eines Dipols eingehalten werden. Der Dipol besteht aus zwei analytischen

Punktlichtquellen (vgl. Gleichung (5.72)), wovon die eine eine negative Beleuchtungsstärke und die andere eine positive Beleuchtungsstärke besitzt. Die Lichtquellen werden so positioniert, dass sie sich im Punkt $\boldsymbol{x_0} + \frac{2A}{3\sigma_t'} N(\boldsymbol{x_0})$ gegenseitig aufheben. Dabei wird eine Lichtquelle innerhalb und eine oberhalb des Materials positioniert. Die Lichtquelle innerhalb des Materials wird als reale Lichtquelle bezeichnet und soll nach Farrell et al. [FPW92] $z_r = 1/\sigma'_t$ Einheiten unter der Oberfläche liegen. Die Lichtquelle oberhalb des Materials wird virtuelle Lichtquelle genannt und muss nach der Grenzbedingung aus Gleichung (5.80) auf der Höhe von $z_v = z_r + \frac{4A}{3\sigma_t'}$ Einheiten liegen. Damit kann die skalare Beleuchtungsstärke für einen Oberflächenpunkt $\boldsymbol{x}$ wie folgt berechnet werden (siehe Abbildung 5.34 und Gleichung (5.72)):

$$\phi(\boldsymbol{x}) = \frac{3\sigma_t' \phi_d}{4\pi} \left(\frac{e^{-\sigma_{tr} d_r}}{d_r} - \frac{e^{-\sigma_{tr} d_v}}{d_v} \right). \tag{5.81}$$

Hier gilt $d_r = \|\boldsymbol{x} - \boldsymbol{x_r}\|$ mit $\boldsymbol{x_r} = \boldsymbol{x} - z_r N(\boldsymbol{x})$ und $d_r = \|\boldsymbol{x} - \boldsymbol{x_v}\|$ mit $\boldsymbol{x_v} = \boldsymbol{x} + z_v N(\boldsymbol{x})$. Anhand dieser Dipol-Gleichung lässt sich die diffuse BSSRDF für das Multiple Scattering berechnen, die das Verhältnis der spezifischen Lichtausstrahlung M_o für einen Austrittspunkt $\boldsymbol{x_o}$ auf der Oberfläche zur skalaren Beleuchtungsstärke im Eintrittspunkt $\boldsymbol{x_i}$ setzt:

$$R_d(\|\boldsymbol{x_i} - \boldsymbol{x_o}\|) = \frac{dM_0(\boldsymbol{x_o})}{d\phi_i} = \frac{N(\boldsymbol{x_o}) \cdot \boldsymbol{E}(\boldsymbol{x_o})}{d\phi_i} = -\frac{1}{3\sigma_t'} \frac{N(\boldsymbol{x_o}) \cdot \boldsymbol{\nabla}\phi(\boldsymbol{x_o})}{d\phi_i}. \tag{5.82}$$

Die letzte Gleichsetzung resultiert aus Gleichung (5.69). Nachfolgend wird der Dipol aus Gleichung (5.81) für ϕ eingesetzt und die Ableitung gebildet:

$$R_d(\|\boldsymbol{x_i} - \boldsymbol{x_o}\|) = \frac{\sigma_s'}{4\pi\sigma_t'} \left(z_r(\sigma_{tr} d_r + 1) \frac{e^{-\sigma_{tr} d_r}}{\sigma_t' d_r^3} + z_v(\sigma_{tr} d_v + 1) \frac{e^{-\sigma_{tr} d_v}}{\sigma_t' d_v^3} \right). \tag{5.83}$$

Schlussendlich erhält man die finale BSSRDF für das Multiple Scattering, wenn man noch die Fresnel-Reflexion für das einfallende und das ausfallende Licht berücksichtigt:

$$f_{ss}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{x_i}, \boldsymbol{\omega_i}) = \frac{1}{\pi} F_t(\boldsymbol{x_i}, \boldsymbol{\omega_i}) R_d(\|\boldsymbol{x_i} - \boldsymbol{x}\|) F_t(\boldsymbol{x}, \boldsymbol{\omega}). \tag{5.84}$$

Es gilt $F_t = 1 - F_r$, wobei F_r der Fresnel-Reflexionsterm ist, der in Kapitel 2.1.4 beschrieben wurde. Diese BSSRDF repräsentiert nur Licht, das von stark streuenden Materialien unterhalb der Oberfläche transportiert wird. Licht, welches nur einmal im Objekt gestreut wird und dann direkt austritt (Single Scattering), wird mit dieser BSSRDF nicht abgebildet. Viele Materialien können aber bereits durch die Betrachtung der reinen Mehrfachstreuung ausreichend approximiert werden [JML*01]. Trägt der Single-Scattering-Term einen deutlich größeren Anteil am Ergebnis als der Multiple-Scattering-Term, wie dies zum Beispiel bei optisch durchlässigeren Medien der Fall ist (beispielsweise leichter Nebel), ist diese BSSRDF ungeeignet.

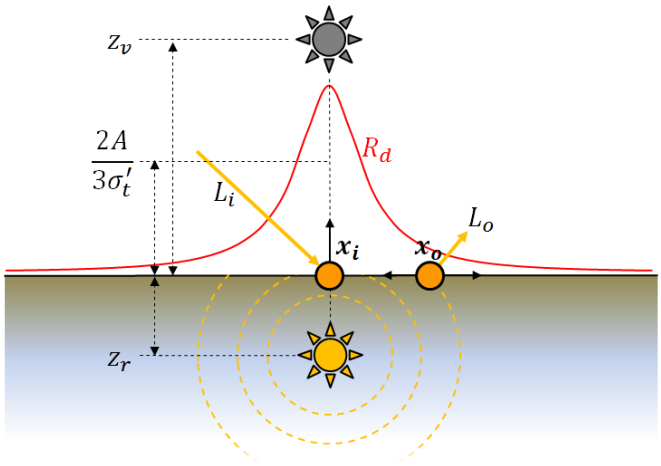

Abbildung 5.34: Subsurface-Scattering durch Dipol-Diffusionsapproximation. Für einen Lichtstrahl L_i werden im Auftreffpunkt $\boldsymbol{x_i}$ zwei Lichtquellen erzeugt. Der Anteil des Lichts, der im Punkt $\boldsymbol{x_o}$ austritt, ist abhängig von der Funktion R_d. Der qualitative Funktionsverlauf für Marmor für einen wandernden Punkt $\boldsymbol{x_o}$ ist in rot aufgetragen.

5.5.3 Integrationsmodell

Im vorhergehenden Kapitel wurde die BSSRDF für das Subsurface-Scattering vorgestellt. Diese wird nun für die Integration des Lichts in den Oberflächenpunkten eines Modells genutzt. Das Licht, das auf die Oberfläche eines SSS-Modells eintrifft, wird in dieser Arbeit in Form von virtuellen Lichtquellen aus der RSM abgebildet (siehe Kapitel 5.2.3). Demnach existiert eine endliche Menge von virtuellen Lichtern für ein Objekt, die durch ihre Position $\boldsymbol{x_l}$, ihre Oberflächennormale $\boldsymbol{n_l}$ und ihre spezifische Lichtausstrahlung B_l (siehe Gleichung (5.9) in Kapitel 5.2.3) beschrieben sind. Nachfolgend wird gezeigt, wie die Rendergleichung für das Subsurface-Scattering auf Basis dieser Lichtrepräsentation approximiert werden kann.

Die austretende Leuchtdichte für einen Punkt $\boldsymbol{x}$ auf der Oberfläche eines Modells ist durch die Rendergleichung wie folgt definiert:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) = \int_A \int_{\Omega^+} f_{ss}(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{x_i}, \boldsymbol{\omega_i}) L_i(\boldsymbol{x_i}, \boldsymbol{\omega_i}) (N(\boldsymbol{x_i}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i} dA. \tag{5.85}$$

Setzt man in diese Gleichung die BSSRDF aus dem vorherigen Kapitel ein, so erhält man:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}) = \int_A \int_{\Omega^+} \frac{1}{\pi} F_t(\boldsymbol{x}, \boldsymbol{\omega}) R_d(\|\boldsymbol{x_i} - \boldsymbol{x_o}\|) F_t(\boldsymbol{x_i}, \boldsymbol{\omega_i}) L_i(\boldsymbol{x_i}, \boldsymbol{\omega_i}) (N(\boldsymbol{x_i}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i} dA. \tag{5.86}$$

Das einfallende Licht im Punkt $\boldsymbol{x_i}$ wird durch direktes Licht aus der RSM ersetzt, weswegen die Integration über die Hemisphäre nicht mehr nötig ist, denn es wird davon ausgegangen, dass auf das SSS-Modell nur direktes Licht einfällt. In der RSM ist die spezifische Lichtausstrahlung unter der Oberfläche für jede virtuelle Lichtquelle gespeichert (siehe Kapitel 5.2.3). Der Fesnel-Term $F_t(\boldsymbol{x_i}, \boldsymbol{\omega_i})$ wurde bereits bei der Erzeugung der RSM mit dem eintreffenden Licht von der primären Lichtquelle multipliziert. Die Integration über die Oberfläche des SSS-Modells beschränkt sich auf die Oberflächenpunkte, die direktes Licht empfangen. Dies sind genau die Oberflächenpunkte, die als virtuelle Lichtquellen in der RSM gespeichert sind. Da nur endlich viele virtuelle Lichter betrachtet werden, wird das Integral zu einer Summe:

$$L_o(\boldsymbol{x},\boldsymbol{\omega}) = \frac{1}{\pi}F_t(\boldsymbol{x},\boldsymbol{\omega})\sum_{j=1}^{m} R_d\left(\left\|\boldsymbol{x_{l,j}} - \boldsymbol{x}\right\|\right)B_{l,j}\left(\boldsymbol{x_{l,j}}\right)A_{l,j} = \frac{1}{\pi}F_t(\boldsymbol{x},\boldsymbol{\omega})E(\boldsymbol{x}). \tag{5.87}$$

Der Summenterm aus dieser Gleichung entspricht der eintreffenden Beleuchtungsstärke E im Austrittspunkt $\boldsymbol{x}$ unter der Oberfläche des Modells (siehe Abbildung 5.35). Die Berechnung der Summe für jeden sichtbaren Oberflächenpunkt im Framebuffer ist in Echtzeit nicht möglich, denn jedes SSS-Modell kann mehrere tausend virtuelle Lichtquellen auf seiner Oberfläche besitzen (je nachdem, wie viele Pixel das Objekt in der RSM einnimmt). Es wird also ganz analog zu den indirekten Reflexionen ein Interpolationsverfahren benötigt. Dies erlaubt die Bestimmung der „inneren" Beleuchtungsstärke E für wenige Caches, deren Ergebnisse dann über das ganze Modell interpoliert werden. Es muss noch darauf hingewiesen werden, dass bei der Entwicklung der BSSRDF von einer unendlichen Ebene ausgegangen wurde. Praktische Modelle besitzen aber komplexe Oberflächen, weswegen die BSSRDF für diese im eigentlichen Sinne keine korrekten Ergebnisse liefern kann. Jensen et al. [JML*01] haben jedoch bemerkt, dass beliebig geformte Modelle mit der BSSRDF ebenfalls gut simuliert werden können, wenn sichergestellt ist, dass der Abstand vom Austrittspunkt $\boldsymbol{x}$ zur realen und virtuellen Dipol-Lichtquelle nicht kleiner als $z_r = 1/\sigma'_t$ bzw. $z_v = 4A/3\sigma'_t$ wird. Diese Bedingungen vermeiden unerwünschte Singularitäten, die sich in zu stark beleuchteten Oberflächenpunkten äußern würden.

Im nächsten Unterkapitel wird das neue Interpolationsmodell des LightSkin-Verfahrens vorgestellt.

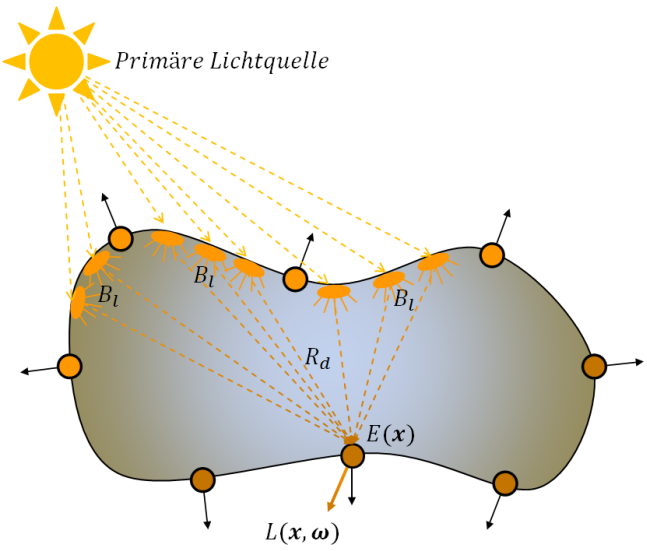

Abbildung 5.35: Das Integrationsmodell für Subsurface-Scattering-Objekte. Das Licht, das in ein Modell durch eine primäre Lichtquelle eindringt, wird über virtuelle Lichtquellen unter der Oberfläche approximiert. Die spezifische Lichtausstrahlung B_j der virtuellen Lichtquellen wird mit der BSSRDF gefaltet und ergibt die Bestrahlungsstärke E im Austrittspunkt.

5.5.4 Interpolation mit Proxy-Lichtquellen

Wie bereits für die indirekte Reflexion soll auch für das Subsurface-Scattering ein neues Interpolationsverfahren vorgestellt werden, das auf der Verwendung von Proxy-Lichtquellen basiert. Die Ausprägung der Proxy-Lichtquellen entspricht hier der der Dipol-Lichtquellen: Es wird für jeden Cache auf der Oberfläche ein Proxy-Dipol erzeugt, der alle auf einen Cache wirkenden Dipole zusammenfasst. Dies erlaubt wiederum eine losere Verteilung der Caches auf dem Modell als die direkte Interpolation der spektralen inneren Beleuchtungsstärke $E(\boldsymbol{x})$. Zwar ist die Beleuchtungs-

stärke diffus und richtungsunabhängig, dennoch sorgt das schnelle Abklingen der Energie des Lichts mit wachsender Entfernung zwischen dem Ein- und Austrittspunkt dafür, dass eine lineare Interpolation auf Basis der Hardware-Rastereinheit das Wesen der BSSRDF verfälscht (siehe Abbildung 5.36). Ganz analog kommt es bei Animationen zu Artefakten, wenn sich der Cache der virtuellen Oberflächenlichtquelle nähert: Der starke Anstieg in der Beleuchtung des Punktes breitet sich in diesem Fall wegen der losen Verteilung der Caches geometrisch zu weit aus. Wird der Abstand zur Lichtquelle wiederum größer, beobachtet man einen unnatürlich starken Abfall des Lichts. Nachfolgend wird deshalb erklärt, wie ein Proxy-Dipol für die Caches berechnet werden kann und wie dieser für die Interpolation genutzt wird. Hier findet im Wesentlichen das gleiche Prinzip wie bereits bei den indirekten Reflexionen Anwendung.

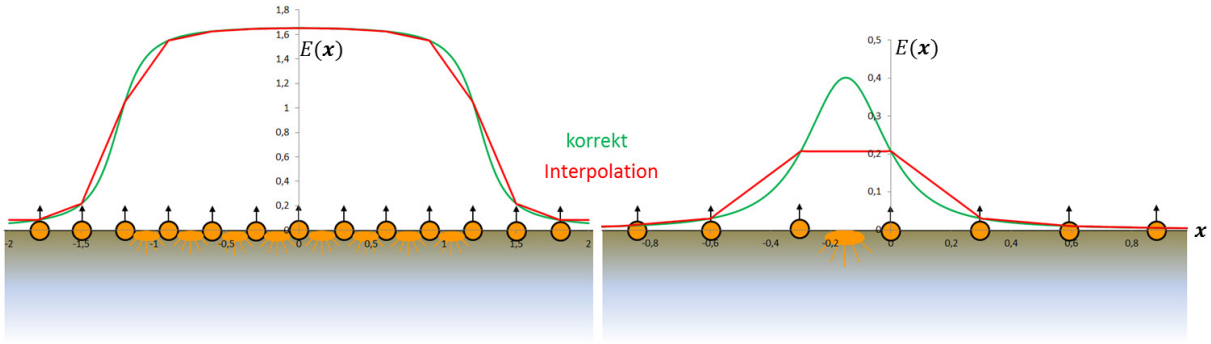

Abbildung 5.36: Die direkte Interpolation der spektralen Beleuchtungsstärke $E(\boldsymbol{x})$ liefert gute Ergebnisse, wenn die virtuellen Lichter breit unter der Oberfläche verteilt sind und die Abtastpunkte nahe beieinander liegen (links). Wenn jedoch das Licht zwischen den Abtastpunkten einfällt, bzw. wenn weniger Abtastpunkte vorhanden sind, wird das Signal nicht ausreichend approximiert (rechts).

Ein Proxy-Dipol für einen Cache beinhaltet die gemittelte Position für eine virtuelle und eine reale Dipol-Lichtquelle ($\boldsymbol{x_{sp,v}}$ und $\boldsymbol{x_{sp,r}}$). Die Gewichtung für die Mittelung basiert auf der diffusen BSSRDF (5.83) aus Kapitel 5.5.2:

$$\begin{aligned} \boldsymbol{x_{sp,v}} &= \frac{\sum_{j=1}^{m} w(\boldsymbol{x_{l,j}}, \boldsymbol{c})(\boldsymbol{x_{l,j}} + z_v \boldsymbol{n_{l,j}})}{\sum_{j=1}^{m} w(\boldsymbol{x_{l,j}}, \boldsymbol{c})}, \\ \boldsymbol{x_{sp,r}} &= \frac{\sum_{j=1}^{m} w(\boldsymbol{x_{l,j}}, \boldsymbol{c})(\boldsymbol{x_{l,j}} - z_r \boldsymbol{n_{l,j}})}{\sum_{j=1}^{m} w(\boldsymbol{x_{l,j}}, \boldsymbol{c})}, \\ w(\boldsymbol{x_{l,j}}, \boldsymbol{c}) &= R_d(\|\boldsymbol{x_{l,j}} - \boldsymbol{c}\|) + \varepsilon_p. \end{aligned} \tag{5.88}$$

Der Epsilonwert ε_p wird aus dem gleichen Grund wie bei den indirekten Reflexionen addiert: Er soll eine Division durch Null verhindern. Zusätzlich wird für jeden Cache die tatsächliche innere Beleuchtungsstärke berechnet:

$$E_{sp} = E(\boldsymbol{c}) = \sum_{j=1}^{m} B_{l,j} R_d(\|\boldsymbol{x_{l,j}} - \boldsymbol{c}\|) A_{l,j}. \tag{5.89}$$

Auf Basis der inneren Beleuchtungsstärke E_{sp} wird in einem zweiten Schritt eine Normalisierungskonstante für den Cache berechnet. Hierfür existiert eine zweite Version der Funktion R_d, die die

Dipol-Lichtquellen als Parameter erwartet: $R_d(\boldsymbol{x_v}, \boldsymbol{x_r}, \boldsymbol{c})$. Mit Hilfe dieser Funktion wird die Normalisierungskonstante für den Cache berechnet:

$$s_{sp} = \frac{E_{sp}}{R_d(\boldsymbol{x_v}, \boldsymbol{x_r}, \boldsymbol{c})}. \tag{5.90}$$

Für jeden Cache werden $\boldsymbol{x_{sp,v}}$, $\boldsymbol{x_{sp,r}}$ und s_{sp} gespeichert. Da sowohl die Normalisierungskonstante, als auch die Positionen der Dipol-Lichtquellen von der jeweiligen Farbkomponente R, G bzw. B abhängig sind, müssen sie für jede Komponente separat gespeichert werden. Es müssen folglich sechs dreidimensionale Vektoren und drei Skalare gespeichert werden. Man könnte aber auch eine kompaktere Repräsentation wählen, bei der alle Proxy-Lichtquellen in der Geraden $\boldsymbol{x_{sp}} + c\boldsymbol{n_{sp}}$ zu einer gemittelten Position $\boldsymbol{x_{sp}}$ und Normalen $\boldsymbol{n_{sp}}$ liegen. Für diesen Fall müssten für jeden Cache außer der gemittelten Position $\boldsymbol{x_{sp}}$ und der Normalen $\boldsymbol{n_{sp}}$ nur noch neun zusätzliche Skalare gespeichert werden. In dieser Arbeit wird jedoch ein Proxy-Dipol pro Farbkomponente berechnet. Zur besseren Lesbarkeit beschränken sich die weiteren Erläuterungen auf nur eine Komponente.

Um ein durchgehendes Ergebnis für jeden sichtbaren Oberflächenpunkt $\boldsymbol{x}$ zu interpolieren, wird eine ähnliche Gewichtung wie bei der indirekten Reflexion genutzt. Jedoch ist für die Interpolation die Richtung der Normalen im Austrittspunkt unerheblich, weswegen die Cache-Gewichtung nur vom Abstand zum Punkt $\boldsymbol{x}$ abhängig ist:

$$w'(\boldsymbol{x}, \boldsymbol{c}) = 1 - \frac{\|\boldsymbol{x} - \boldsymbol{c}\|}{d_{max}}. \tag{5.91}$$

d_{max} entspricht wieder dem Maximalabstand von allen betrachteten nächsten Caches. Basierend auf dieser Gewichtung wird für jeden Oberflächenpunkt ein neuer Proxy-Dipol berechnet, der sich wie folgt zusammensetzt (n entspricht der Anzahl der nächsten Caches):

$$\boldsymbol{x_{s,v}} = \frac{\sum_{j=1}^{n} w'(\boldsymbol{x}, \boldsymbol{c_j})\boldsymbol{x_{sp,v,j}}}{\sum_{j=1}^{n} w'(\boldsymbol{x}, \boldsymbol{c_j})}, \boldsymbol{x_{s,r}} = \frac{\sum_{j=1}^{n} w'(\boldsymbol{x}, \boldsymbol{c_j})\boldsymbol{x_{sp,r,j}}}{\sum_{j=1}^{n} w'(\boldsymbol{x}, \boldsymbol{c_j})}, s_{sp} = \frac{\sum_{j=1}^{n} w'(\boldsymbol{x}, \boldsymbol{c_j})s_{sp,j}}{\sum_{j=1}^{n} w'(\boldsymbol{x}, \boldsymbol{c_j})}. \tag{5.92}$$

Schlussendlich ergibt sich die finale innere Beleuchtungsstärke durch die Anwendung der diffusen BSSRDF multipliziert mit der Normalisierungskonstanten:

$$E(\boldsymbol{x}) = s_{sp} R_d(\boldsymbol{x_{s,v}}, \boldsymbol{x_{s,r}}). \tag{5.93}$$

Abbildung 5.37 zeigt die Ergebnisse der Interpolation im Vergleich zur Ideallösung ohne Interpolation. Aus der Abbildung wird ersichtlich, dass die Interpolation für eine lose Cache-Verteilung bessere Approximationen liefert als die direkte lineare Interpolation der inneren Beleuchtungsstärke E. Dies gilt insbesondere, wenn Licht zwischen den Caches einfällt, was aufgrund der weiten Verteilung der Caches häufig vorkommen kann. In der Abbildung 5.37 wird außerdem ein Vergleich zu einer alternativen Gewichtungsfunktion $w' = 1/(\|\boldsymbol{x} - \boldsymbol{c}\| + \varepsilon)$ gezeigt. Während die alternative Gewichtungsfunktion für punktuell einfallendes Licht geeignet erscheint, wird das Signal bei breit einfallendem Licht nicht zufriedenstellend approximiert.

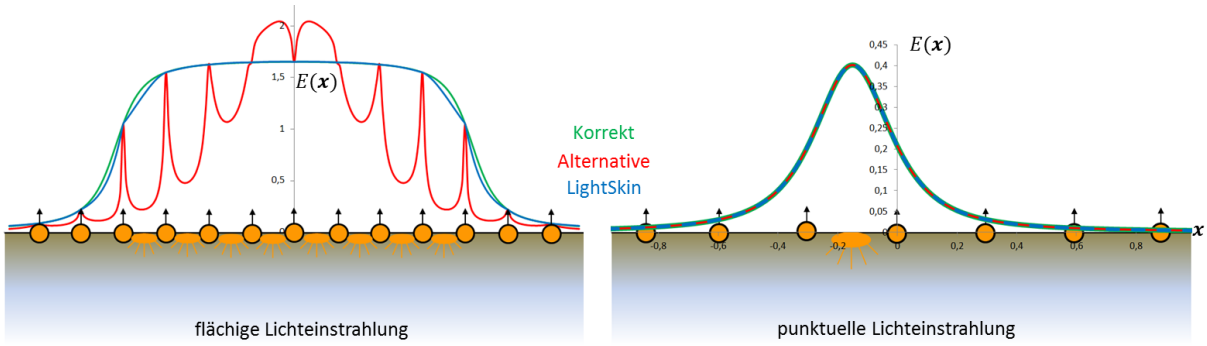

Abbildung 5.37: Im rechten Bild erkennt man, dass lokale Lichtquellen, die zwischen den Caches liegen, sehr gut approximiert werden (die Approximation ist deckungsgleich zur korrekten Lösung). Bei breit einfallendem Licht, wie auf der linken Seite dargestellt, überzeugt die alternative Gewichtungsfunktion $1/(\|\boldsymbol{x}-\boldsymbol{c}\|+\varepsilon)$ nicht und bildet deutliche Artefakte um die Caches herum, während das LightSkin-Verfahren gute Approximationen liefert. d_{max} wurde so gewählt, dass es dem Abstand zwischen zwei benachbarten Caches entspricht.

Die resultierende Leuchtdichte für einen Austrittspunkt kann ermittelt werden, indem, wie in Gleichung (5.87) beschrieben, die innere Beleuchtungsstärke mit dem Fresnel-Transmissionsterm multipliziert und durch π geteilt wird. Für diese Berechnung muss die Normale im Austrittspunkt dann wieder berücksichtigt werden.

5.5.5 Diskussion

Die hier vorgestellte Lösung zur Echtzeit-Darstellung von SSS-Materialien beinhaltet keine Überprüfung der Durchgängigkeit eines Objekts. Demnach wird davon ausgegangen, dass zwischen den Oberflächenpunkten immer ein durchgängiges Medium liegt, dies gilt allerdings nur für konvexe Objekte. Praktisch lässt sich das Verfahren aber auch für konkave Objekte einsetzen, denn die Fehler sind für den Betrachter nicht direkt offensichtlich [DS03]. Aus diesem Grund gehen viele interaktive Subsurface-Scattering-Verfahren von einem durchgängigen Medium zwischen den Oberflächenpunkten aus [DS03,BBN*07,KLO07]. Eine approximative Dämpfung für konkave Objekte kann durch den Algorithmus aus Kapitel 5.4 implementiert werden, jedoch bestehen diesbezüglich noch keine Erfahrungswerte. Offensichtliche Fehler entstehen, wenn Licht zum Beispiel auf die Innenfläche eines Hohlkörpers scheint. In diesem Fall wird die dem Lichteinfall gegenüberliegende Innenseite des Körpers ebenfalls stark beleuchtet, obwohl diese nicht direkt durch das Medium verbunden ist. Diese Fälle sind aber selten und lassen sich meistens durch eine geeignete Wahl der Streuungs- bzw. Dämpfungskoeffizienten optisch eingrenzen.

Beim Vergleich mit einer Brute-Force-Lösung, die für jeden sichtbaren Oberflächenpunkt die innere Beleuchtungsstärke direkt bestimmt, hat sich herausgestellt, dass das LightSkin-Verfahren andere Beleuchtungsverläufe zeichnet (die Brute-Force-Lösung geht ebenfalls von einem durchgehenden Medium aus). Ein Beispiel hierfür ist in Abbildung 5.38 zu finden. In dieser ist zu erkennen, dass der Brute-Force-Algorithmus dazu neigt, falsch-wirkende helle „Spots“ zu erzeugen (beispielsweise am Hals oder Sockel des Buddha-Modells). Es ist zu vermuten, dass diese Spots durch die Anwendung der Dipol-Diffusion für beliebig geformte Oberflächen entstehen. Allerdings ist nicht auszuschließen, dass die vorgenommene Implementierung des Brute-Force-Ansatzes fehlerhaft ist. Interessanterweise sind diese Fehler bei den meisten Modellen, die mit dem LightSkin-Verfahren gerendert wurden, nicht aufgetreten, obwohl das Verfahren ebenfalls auf der Dipol-Diffusion aufbaut.

Beim Vergleich des Proxy-Interpolationsverfahrens mit einem Verfahren, bei dem die Beleuchtungsstärke direkt interpoliert wird (Abbildung 5.38 (c) und (f)), werden die Vorteile der Proxy-Interpolation direkt ersichtlich. Während die direkte Interpolation der Beleuchtungsstärke zwar zu der Brute-Force-Lösung konvergiert, erzeugt sie bei einer losen Cache-Verteilung von ca. 150 Caches deutliche Artefakte, die die polygonale Struktur des Modells hervorheben. Dieser Effekt wird noch deutlicher, wenn eine kräftige Lichtquelle punktuell auf das Modell wirkt (f). In diesem Fall wird die punktförmige Reflexion bzw. Streuung deutlich ausgefranst und strukturelle Veränderungen im Modell werden nicht korrekt berücksichtigt. Dies erkennt man zum Beispiel beim Hinterlauf des Bunny-Modells in (f) und (g). Während das LightSkin-Verfahren den Hinterlauf abzeichnet, wird er beim simpleren Interpolationsverfahren nicht abgesetzt.

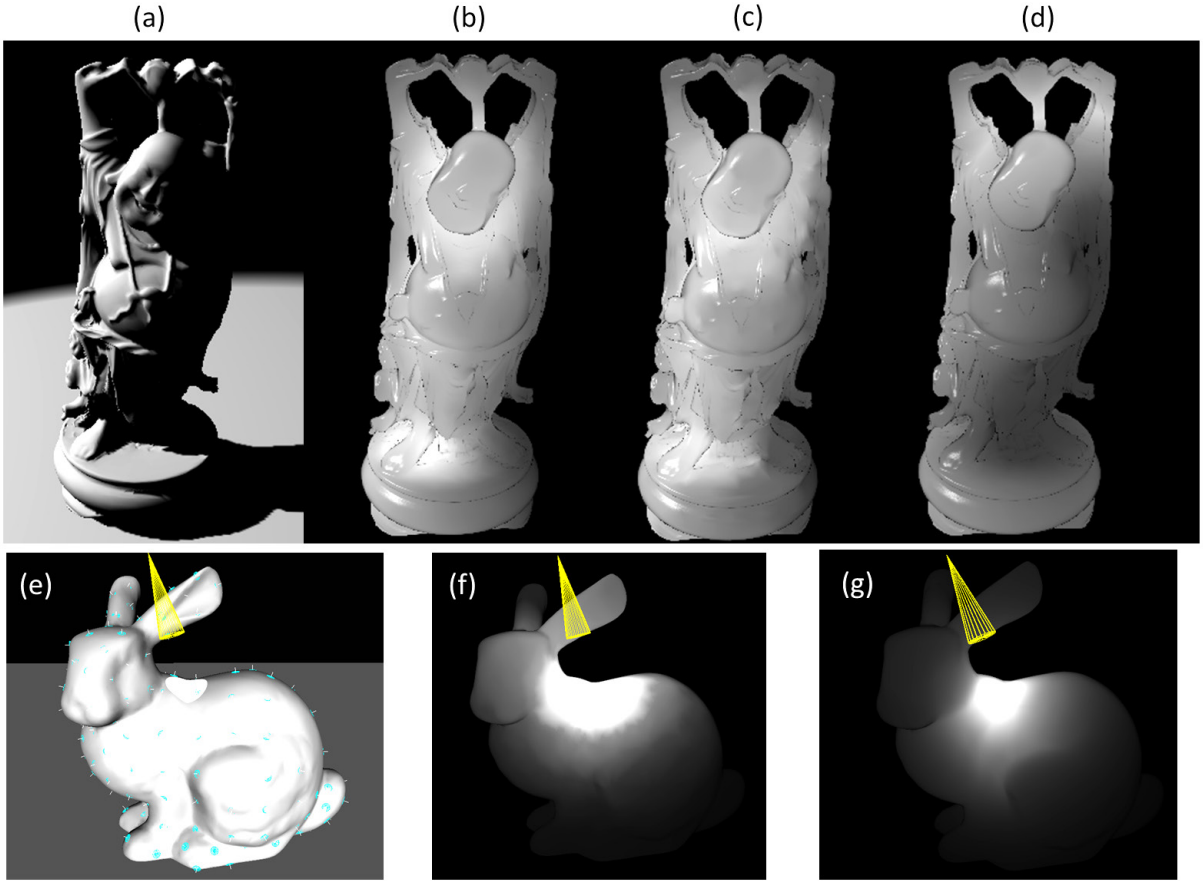

Abbildung 5.38: Vergleich des LightSkin-Verfahrens (d, g) zu einer Brute-Force-Lösung (b) und der direkten Interpolation der inneren Beleuchtungsstärke (c, f). Das LightSkin-Verfahren erzeugt andere Beleuchtungsverläufe als die Brute-Force-Lösung, die jedoch optisch plausibler wirken. Die direkte Interpolation der Beleuchtungsstärke führt zu deutlichen Artefakten, die die polygonale Struktur des Modells hervorheben. Darüber hinaus wird punktuell einfallendes Licht ausgefranst (f), wenn das Modell nur wenige Caches verwendet (e).

5.6 Caches

Die in dieser Arbeit vorgestellten Interpolationsverfahren setzen eine a-priori-Verteilung der Caches im Modellraum voraus. Wie diese Verteilung konkret aussieht, bzw. ob diese ohne viel Rechenaufwand automatisiert erfolgen kann, wurde bis jetzt noch nicht erläutert. In diesem Kapitel wird deshalb detaillierter auf die Verteilung der Caches eingegangen, indem eine analytische Betrachtung für gute und schlechte Verteilungen vorgenommen wird. Aus dieser Betrachtung lässt sich dann ein automatisiertes Verfahren ableiten, das eine möglichst gute Verteilung erzeugt. Abschließend wird noch auf die Transformation der Caches bei Animationen eingegangen und den Zusatzaufwand, der sich durch diese ergibt. Doch bevor diese Themen erläutert werden, sollen die Daten eines Caches zusammenfassend aufgeführt werden, um dessen Datenvolumen abschätzen zu können.

5.6.1 Daten

Durch die vorgestellten Interpolationsverfahren wird ersichtlich, dass ein Cache als Flächenelement verstanden werden kann, das die folgenden initialen Eigenschaften besitzt, die im Voraus erzeugt werden müssen:

1. Position,
2. Normale,
3. Fläche,
4. Reflexionsexponent.

Zusätzlich werden für die Interpolation der indirekten Reflexionen auf Basis der BRDF die folgenden Daten gespeichert:

5. Proxy-Lichtquelle für diffuse Reflexionen (Position & Lichtstrom),
6. Proxy-Lichtquelle für glänzende Reflexionen (Position & Lichtstrom).

Sollen die Reflexionen zusätzlich weiche Schatten erzeugen, muss die Standardabweichung für das indirekte Licht ebenfalls abgespeichert werden:

7. Standardabweichung für diffuse Proxy-Lichtquelle,
8. Standardabweichung für glänzende Proxy-Lichtquelle.

Schließlich kann ein Objekt ein Subsurface-Scattering-Material besitzen, das die Speicherung von Proxy-Dipolen, sowie einer Material-Identifikationsnummer pro Cache erfordert:

9. virtuelle Proxy-Dipol-Lichtquelle für die Rot-, Grün- und Blau-Komponente,
10. reale Proxy-Dipol-Lichtquelle für die Rot-, Grün- und Blau-Komponente,
11. spektraler Normalisierungskoeffizient,
12. Material-ID.

Alle Eigenschaften der Caches werden in der folgenden Tabelle zusammengefasst:

Tabelle 5.1: Eigenschaften der Caches und deren Formate.

Art	Verwendung	Datentyp	Speicher (Bytes)
A-priori-Cache-Daten	Position	3D-Vektor (x,y,z)	12
	Normale	3D-Vektor (x,y,z)	12
	Fläche	Fließkomma	4
	Reflexionsexp.	Integer	4
Proxy-Licht. diffuse Reflexionen	Position	3D-Vektor (x,y,z)	12
	Lichtstrom	3D-Vektor (r,g,b)	12
Proxy-Licht. glänzende Reflexionen	Position	3D-Vektor (x,y,z)	12
	Lichtstrom	3D-Vektor (r,g,b)	12
Weiche Schatten diffuse Reflexionen	Standardabweichung	3D-Vektor (x,y,z)	12
Weiche Schatten glänz. Reflexionen	Standardabweichung	3D-Vektor (x,y,z)	12
Subsurface-Scattering (Proxy-Dipol)	Virtuelle Position	3x 3D-Vektor (x,y,z)	36
	Reale Position	3x 3D-Vektor (x,y,z)	36
	Normalisierungskoeff.	3D-Vektor (r,g,b)	12
	Material-ID	Integer	4

Zusammengefasst benötigt ein Cache also maximal 192 Bytes. Modelle wie das Stanford-Bunny, der Buddha oder der Stanford-Dragon lassen sich typischerweise gut durch 100-300 Caches approximieren, während Szenen wie die Crytek-Sponza-Scene um die 3000 Caches benötigt. Hierbei ist zu beachten, dass die Qualität der Approximation von den Materialeigenschaften abhängig ist. Prinzipiell benötigen stark glänzende Materialien mehr Caches als moderat glänzende. Geht man von einer sehr komplexen Szene aus, können ca. 10.000-15.000 Caches nötig werden, die einen Speicheraufwand von ca. 3-4 MB darstellen. Der Speicheraufwand für die Caches ist demnach vernachlässigbar gering. Allerdings darf nicht vergessen werden, dass zusätzliche Daten zur Referenzierung der Caches pro Oberflächenpunkt benötigt werden. Wie viel Speicher diese Daten zusätzlich belegen, ist von der gewählten Implementierung abhängig und soll deshalb nicht Gegenstand dieses Kapitels sein.

Für die a-priori-Cache-Verteilung sind nur die Daten der Punkte 1-4 relevant, da die restlichen Daten bei der Echtzeitverarbeitung erzeugt werden. Der Reflexionsexponent lässt sich bestimmen, indem man den nächsten Oberflächenpunkt zum Cache betrachtet und dessen Reflexionskoeffizienten übernimmt. Die Bestimmung der Fläche war bereits Gegenstand von Kapitel 5.4, so dass nur noch geeignete Werte für die Position und Normale des Caches gefunden werden müssen.

5.6.2 Gute und schlechte Cache-Verteilungen

Die Verteilung der Caches hängt maßgeblich von dem gewählten Interpolationsverfahren ab. Das in dieser Arbeit vorgestellte Interpolationsverfahren erlaubt eine relativ lose bzw. weite Verteilung der Caches für die meisten Beleuchtungsszenarien. Nichtsdestotrotz entsteht ein Fehler bei der Interpolation, der streng in Korrelation zur „Ähnlichkeit“ der Oberflächenpunkte zu den Caches steht. Oder anders ausgedrückt: Werden für einen beliebigen Oberflächenpunkt keine passenden Caches in seiner näheren Umgebung gefunden, wird das Ergebnis der Interpolation für diesen Punkt entsprechend schlecht ausfallen. Es muss also sichergestellt werden, dass für jeden Oberflächenpunkt ausreichend ähnliche Caches als Stützpunkte für die Interpolation existieren. Die Ähnlichkeit eines Caches wird dabei durch sein Interpolationsgewicht beschrieben. Für das LightSkin-Verfahren wurden drei verschiedene Interpolationsgewichte vorgestellt (Kapitel 5.3.5 und 5.5.4), um drei verschiedene optische Phänomene zu behandeln: eine Funktion für diffuse und eine für glänzende

Reflexionen, sowie eine für die Simulation von Subsurface-Scattering-Materialien. Betrachtet man die Interpolationsfunktionen nebeneinander, fällt auf, dass sich diese nur marginal unterscheiden:

$$\begin{aligned} w_d(\boldsymbol{x},\boldsymbol{c}) &= \left(1-\frac{\|\boldsymbol{x}-\boldsymbol{c}\|}{d_{max}}\right)\sqrt{\left(N(\boldsymbol{x})\cdot N(\boldsymbol{c})\right)^+}, \\ w_g(\boldsymbol{x},\boldsymbol{c}) &= \left(1-\frac{\|\boldsymbol{x}-\boldsymbol{c}\|}{d_{max}}\right)\sqrt{\left(R(\boldsymbol{x})\cdot R(\boldsymbol{c})\right)^+}, \\ w_{ss}(\boldsymbol{x},\boldsymbol{c}) &= \left(1-\frac{\|\boldsymbol{x}-\boldsymbol{c}\|}{d_{max}}\right). \end{aligned} \tag{5.94}$$

Durch die Gleichungen wird ersichtlich, dass alle drei optischen Phänomene schlecht repräsentiert werden, wenn der Abstand zwischen dem Cache und dem Oberflächenpunkt sehr groß wird. d_{max} entspricht dem Abstand zum maximal entferntesten Cache, der noch im Oberflächenpunkt $\boldsymbol{x}$ betrachtet wird. Wird d_{max} für jeden Oberflächenpunkt separat berechnet, ist darauf zu achten, dass die Verteilung der Caches möglichst gleichmäßig ist. Ansonsten wird die Gewichtungsfunktion bei Veränderungen von $\boldsymbol{x}$ deutlich unstetig. Aber auch wenn d_{max} für das komplette Modell konstant ist, müssen die Caches möglichst gleichmäßig auf dem Modell verteilt liegen, damit d_{max} aussagekräftig ist.

Für die Interpolation der diffusen Reflexion ist ebenfalls die Abweichung der Normalen vom Cache zum Oberflächenpunkt von Bedeutung. Ist diese größer als neunzig Grad, trägt der Cache zur Interpolation nicht bei (Gewichtung von null). Glänzende Reflexionen sind dagegen vom Spiegelvektor R abhängig, der sich aus dem Betrachtungswinkel und der Oberflächennormalen ergibt (Eintrittswinkel ist gleich Austrittswinkel). Da der Betrachtungswinkel im Vorhinein nicht bekannt ist, kann dieser nicht für die a-priori-Verteilung der Caches berücksichtigt werden. Jedoch kann man eine praktische Annahme voranstellen: Ob die Spiegelvektoren von einem Oberflächenpunkt und einem Cache ähnlich sind, hängt vom Betrachtungsabstand und dem Abstand zwischen Oberflächenpunkt und dem Cache ab. Ist der Abstand zum Betrachter verhältnismäßig groß, im Vergleich zum Abstand der beiden Punkte zueinander, so können die Verbindungsstrahlen vom Betrachter zu den Punkten als annähernd parallel angenommen werden. In diesem Fall steht der Reflexionsvektor in proportionalem Verhältnis zur Normalen. Also ähneln sich die Reflexionsvektoren für Cache und Oberflächenpunkt dann, wenn deren Normalen sich gering unterscheiden (unter der Annahme, dass der Betrachter entsprechend weit entfernt ist).

Aus den vorherigen Schlussfolgerungen sollte ersichtlich geworden sein, dass die besten Ergebnisse dann erzielt werden, wenn die Abstände der Oberflächenpunkte zu den Caches minimal sind und sich deren Normalen möglichst wenig unterscheiden. Darüber hinaus sollte die Verteilung der Caches relativ gleichmäßig vorgenommen werden. Dies kann durch eine einfache Fehlermetrik abgebildet werden, die dann für das komplette Modell minimalisiert wird:

$$\varepsilon(\boldsymbol{x},\boldsymbol{c}) = \|\boldsymbol{x}-\boldsymbol{c}\|\sqrt{\left(N(\boldsymbol{x})\cdot N(\boldsymbol{c})\right)^{+}}, \qquad \varepsilon(\boldsymbol{x}) = \min\left(\varepsilon(\boldsymbol{x},\boldsymbol{c})\right) f\ddot{u}r\ \forall \boldsymbol{c}\epsilon C(\boldsymbol{x}),$$
$$E = arg\,min\left(\sum_{\boldsymbol{x}\in X}\varepsilon^2(\boldsymbol{x})\right). \tag{5.95}$$

In den Gleichungen ist X die Menge aller Oberflächenpunkte und $C(\boldsymbol{x})$ die Menge aller Caches, die nahe zu $\boldsymbol{x}$ liegen. Das Quadrieren des Summenterms soll dafür sorgen, dass der Fehler möglichst gleichmäßig für alle Punkte auf der Modelloberfläche ausfällt, denn dies spricht für eine gleichmäßige Verteilung der Caches (Ausreißer werden quadratisch „bestraft“). Es ist wichtig zu erwähnen, dass die hier beschriebene Fehlermetrik nicht den absoluten Fehler in der empfangenen Leuchtdichte begrenzt. Dieser ist abhängig vom Betrachter und von den gewählten Lichteinstellungen in der Szene.

5.6.3 Automatisierte Verteilung der Caches

Auf Basis der Überlegungen aus dem vorherigen Kapitel lässt sich ein automatisiertes Verfahren zur Erzeugung von Cache-Verteilungen ableiten. Grundsätzlich stellt die Findung einer idealen Lösung für die Gleichungen (5.95) für eine feste Anzahl von Caches ein rechenaufwändiges Optimierungsproblem dar. Für das LightSkin-Verfahren soll aber gelten, dass keine aufwändigen Vorberechnungen nötig sind. Aus diesem Grund wurde ein Ansatz gewählt, der durch zwei einfache und intuitive Parameter gesteuert werden kann. Der Fehler wird so eingegrenzt, dass $E \leq c$ gilt. Die Verteilung kann innerhalb weniger Sekunden (1-10 Sekunden) berechnet und direkt betrachtet und getestet werden (für Szenen mit 50.000 – 500.000 Dreiecken).

Die Steuerungsparameter zur Verteilung ergeben sich aus der Fehlermetrik und sind

1. der maximal erlaubte Abstand d_ε von einem Cache zu einem Oberflächenpunkt des Modells und
2. die maximale Winkelabweichung ϑ_ε in Grad zwischen der Cache- und der Oberflächenpunktnormalen.

Das Verfahren stellt sicher, dass für jeden Oberflächenpunkt mindestens ein Cache existiert, der den maximal erlaubten Abstand und Winkel nicht überschreitet. Zusätzlich wird dafür gesorgt, dass die Verteilung der Caches möglichst gleichmäßig vorgenommen wird. Damit der Verteilungsalgorithmus dies gewährleisten kann, erhält er als Eingabe die Vertizes und Dreiecke eines Modells und besteht aus fünf Verarbeitungsstufen:

1. **Voxelisierung**: Die Modelloberfläche wird in gleich große Würfel (Voxel) mit einer Seitenlänge von $d_\varepsilon/2$ unterteilt.
2. **Gruppierung der Normalen**: Die Normalen aller Oberflächen innerhalb eines Voxels werden nach ihrer Richtung in gleich große Behälter (Bins) sortiert. Jeder Behälter beinhaltet den Anteil der Flächen, der zu seiner Richtung passt.
3. **Bildung der Caches**: Für jeden Voxel wird der Normalenbehälter ausgesucht, der die größte Fläche im Voxel repräsentiert. Jede Fläche bzw. Flächennormale im Voxel, die eine geringere Winkelabweichung als ϑ_ε zur Normalen des Behälters aufweist, wird zur Bildung einer gemittelten Cache-Position und -Normalen herangezogen. Dabei werden die Flächen aus den Normalenbehältern entfernt, die zur Mittelung beigetragen haben. Dieser Vorgang wird so lange wiederholt, bis alle Normalenbehälter leer sind.

4. **Cache-Neupositionierung zu existierenden Dreiecken (optional)**: Die Caches werden innerhalb des Voxels auf existierende Flächen positioniert, die ihren Eigenschaften am ähnlichsten sind (Snapping).
5. **Entfernung ähnlicher Caches**: Es kann passieren, dass Caches zwischen zwei angrenzenden Voxeln sehr ähnliche Eigenschaften aufweisen, so dass Caches entfernt werden können.

Die einzelnen Schritte sind in Abbildung 5.39 ebenfalls grafisch dargestellt. Da es ausreicht, wenn die Ergebnisse des Algorithmus für die Cache-Verteilung erst nach ein paar Sekunden vorliegen, kann dieser komplett für die CPU implementiert werden. Dadurch reduziert sich die Komplexität der Implementierung deutlich. Es folgt eine detaillierte Beschreibung der einzelnen Verarbeitungsschritte:

Voxelisierung: Die Voxelisierung der Modelloberflächen basiert auf der Idee des Stratified Sampling [PH10]. Dieses wird unter anderem zur Erzeugung von losen Abtastpunkten in zweidimensionalen Bildern genutzt, um eine „Verklumpung“ von Abtastpunkten zu vermeiden. Die Idee ist folgende: Ein Bild wird in gleichmäßige Rechtecke zerlegt, in die jeweils ein Abtastpunkt gelegt wird. Hierdurch ist sichergestellt, dass die Abtastpunkte relativ gleichmäßig verteilt liegen (innerhalb eines Rechtecks werden die Positionen zufällig gewählt). Das Prinzip des Stratified Sampling lässt sich ohne große Veränderungen auf den dreidimensionalen Raum erweitern, indem man das Modell in gleich große Quader bzw. wie beim LightSkin-Verfahren in gleich große Würfel (Voxel) zerlegt. Zwar stellt die reine Zerlegung der Modelloberfläche in Würfel noch nicht sicher, dass nicht doch eine leichte Verklumpung der Caches stattfindet, aber diese Klumpen enthalten nur wenige Caches und können im Nachhinein einfach aufgelöst werden (es kann passieren, dass benachbarte Zellen die Caches so ungünstig zum Rand positionieren, dass sie sehr nahe zueinander liegen). In diesem Verarbeitungsschritt werden aber noch keine Caches verteilt, sondern nur die Dreiecke des Modells den entsprechenden Voxeln zugeordnet. Praktisch lässt sich dies effizient durch einen Octree implementieren. Die Größe der Voxel entspricht der Hälfte des maximal erlaubten Abstands d_ε. Da jeder Voxel mindestens einen Cache erhalten wird, ist sichergestellt, dass jeder Oberflächenpunkt $\boldsymbol{x}$ einen Cache $\boldsymbol{c}$ in seiner Umgebung besitzt, für den gilt: $\|\boldsymbol{x} - \boldsymbol{c}\| \leq d_\varepsilon$.

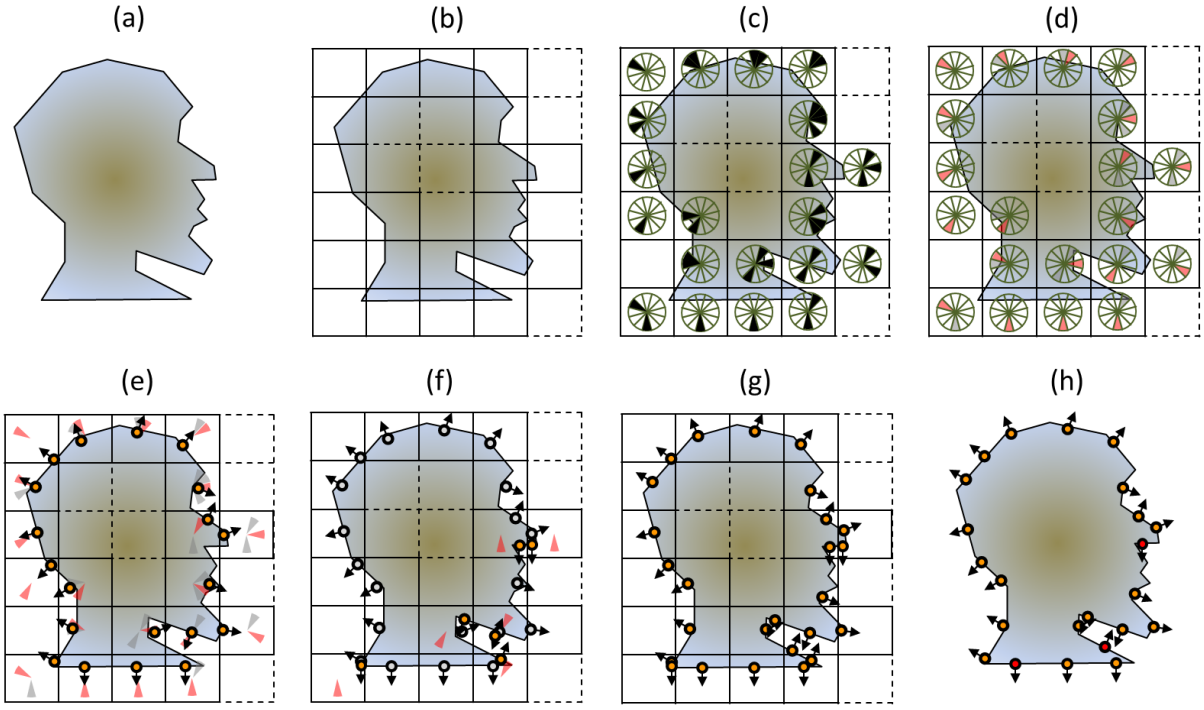

Abbildung 5.39: Automatisierte Verteilung von Caches. (a) Als Eingabe für den Algorithmus dient das triangulierte Modell. (b) Die Oberfläche des Modells wird in gleich große Voxel unterteilt. (c) Die Oberflächen werden nach ihrer Normalen in Behälter sortiert. (d) Für jeden Voxel wird der Normalenbehälter gewählt, der die größte Fläche repräsentiert. (e) Die Caches werden erzeugt. Die Position und Normale der Caches sind das gewichtete Mittel aus allen Oberflächen, die zur Normalen des gewählten Behälters passen. (f) Schritt (d) wird für die nicht-leeren Normalenbehälter wiederholt. (g) Die berechneten Caches werden auf die „ähnlichste" Oberfläche positioniert. (h) Sehr ähnliche Caches aus benachbarten Voxeln werden gelöscht. In diesem Beispiel ist $\vartheta_\varepsilon = 90°$.

Gruppierung der Normalen: Die Gruppierung der Normalen ist nötig, weil die Dreiecke innerhalb eines Voxels beliebig komplex orientiert sein können. Ein naiver Ansatz würde darin bestehen, eine gewichtete Mittelung über alle Oberflächennormalen im Voxel vorzunehmen. Da aber das Gewicht für einen Cache null wird, wenn dessen Normale mehr als neunzig Grad von der Normalen des Oberflächenpunkts abweicht, kann es passieren, dass Oberflächenpunkte im Voxel keinen passenden Cache erhalten.

Für die Gruppierung müssen gleichmäßig große Behälter geschaffen werden. Diese Behälter speichern den Anteil der Fläche, der durch die Normale des Behälters repräsentiert wird. Eine kompakte Möglichkeit, eine Normale zu beschreiben, besteht darin, sie in Polarkoordinaten darzustellen. In diesen wird die Normale durch zwei Winkel beschrieben. Die Behälter könnten dann gleich große Rechtecke im Polarkoordinatenraum sein, in die die Normalen sortiert werden. Die Verwendung von Polarkoordinaten hat jedoch den Nachteil, dass gleich große Flächen im Winkelraum nicht zu gleich großen Flächen auf der Oberfläche einer Kugel führen. Die Behälter würden also verschieden groß werden. Die Lösung für dieses Problem ist, eine Darstellungsform zu wählen, die eine gleichmäßige Verteilung von 2D-Punkten in eine gleichmäßige Verteilung von Punkten auf der Oberfläche einer Kugel abbildet. Hier findet man in der Abtasttheorie zur Erzeugung von gleichverteilten Zufallskoordinaten auf Kugeloberflächen verschiedene Lösungen [HW59,Mar72,SB96]. In dieser Arbeit wird der Darstellung aus [SB96] gefolgt, die eine Normale $\boldsymbol{n}$ durch zwei Koordinaten beschreibt ($u \in [0,1]$ und $\theta \in [0,2\pi)$):

$$n_x = \sqrt{1-u^2}cos\theta, \qquad n_y = \sqrt{1-u^2}sin\theta, \qquad n_z = u. \tag{5.96}$$

(Um von der Normalen zu den Koordinaten u und θ zu gelangen, muss bedacht werden, dass die Ergebnisse von n_x für θ und $2\pi - \theta$ identisch sind). Die Koordinatenumwandlung erlaubt jetzt die einfache Darstellung eines Behälters als zweidimensionale Region: $[u_0, u_1, \theta_0, \theta_1]$. Bei der Implementierung hat sich eine Unterteilung der Kugeloberfläche in 18x18 Behälter als sinnvoll erwiesen (ein Behälter deckt also einen Winkel von ca. 20° ab).

Die Akkumulation der Flächen in einem Normalenbehälter erfordert gegebenenfalls das Zuschneiden von Dreiecken, falls diese über den Rand eines Voxels ragen. Der Zuschnitt kann durch eine Implementierung des Sutherland-Hodgman-Algorithmus [SH74] vorgenommen werden. Es ist wichtig zu erkennen, dass ohne diesen Zuschnitt der Algorithmus, insbesondere für grob triangulierte Objekte, fehlerhafte Ergebnisse liefert.

Bildung der Caches: Zur Bildung der Caches wird zuerst der Normalenbehälter bestimmt, dessen Fläche den größten Teil innerhalb des Voxels ausmacht. Für diesen Behälter wird eine Normale gebildet ($u_b = (u_0 + u_1)/2, \theta_b = (\theta_1 + \theta_0)/2$). Als Nächstes wird für jede Fläche innerhalb des Voxels geprüft, ob dessen Normale stärker als ϑ_ε von der Normalen $\boldsymbol{n_b}$ des Behälters abweicht. Ist dies nicht der Fall, so wird die Fläche zur Bestimmung der Cache-Normalen $\boldsymbol{n_c}$ und der Cache-Position $\boldsymbol{c}$ hinzugenommen (gewichtete Mittelung):

$$\boldsymbol{n_c} = \frac{\sum_{i=0}^{m} A_i \boldsymbol{n_i}}{\sum_{i=0}^{m} A_i}, \qquad \boldsymbol{c} = \frac{\sum_{i=0}^{m} A_i \boldsymbol{p_i}}{\sum_{i=0}^{m} A_i}, \quad f\ddot{u}r\ \forall \boldsymbol{n_i}\colon (\boldsymbol{n_i} \cdot \boldsymbol{n_b}) > \cos(\vartheta_\varepsilon). \tag{5.97}$$

$\boldsymbol{p_i}$ entspricht dem Massenschwerpunkt (Centroid) des Polygons (durch den Zuschnitt der Dreiecke durch die Voxelgrenzen können mögliche Polygone entstehen). Abschließend werden die Flächen, deren Normalen zur Bildung des Caches beigetragen haben, aus ihren korrespondierenden Normalenbehältern entfernt. Existieren nach dem Vorgang noch gefüllte Normalenbehälter, wird wieder der Behälter gewählt, der am vollsten ist. Dann wird nach dem gleichen Prinzip ein weiterer Cache erzeugt. Maximal können pro Voxel $360°/\vartheta_\varepsilon$ Caches erzeugt werden.

Cache-Neupositionierung zu existierenden Dreiecken (optional): Durch die Mittelung der Positionen kann es passieren, dass Caches nicht auf der Oberfläche des Modells liegen, sondern darüber oder darunter. Dies ist zwar nicht zwingend ein Problem, kann aber zu einem werden, wenn indirekte Reflexionen weiche Schatten werfen sollen oder wenn auf dem Modell Animationen abgespielt werden sollen, die dieses verformen. Im ersten Fall kann ein in der Luft „schwebender" Cache zu einem fehlerhaften Verdeckungsverhalten führen, da davon ausgegangen wird, dass auf seiner Position eine Kreisscheibe liegt. Im zweiten Fall erleichtert die Neupositionierung des Caches auf ein bestehendes Dreieck die Transformation des Caches (Konvexkombination aus den Eckpunkten des Dreiecks). Die Neupositionierung des Caches geschieht wie folgt: Der Cache wird auf der Fläche positioniert, die seinen Eigenschaften am meisten entspricht, für die also gilt: $\min(\varepsilon(\boldsymbol{x}, \boldsymbol{c}))$.

Entfernung ähnlicher Caches: Der letzte Schritt des Algorithmus besteht darin, die Caches aus der Verteilung zu entfernen, die sich sehr ähnlich sind, also die, deren Positionen und Normalen annähernd gleich sind. Da es ausgeschlossen ist, dass die Caches innerhalb eines Voxels zu ähnlich sind, muss diese Prüfung für alle erzeugten Caches des Modells durchgeführt werden. Aus Laufzeitgründen sollten hierfür die Caches in eine räumliche Beschleunigungsstruktur gespeichert werden (für die

Referenzimplementierung wurde wieder ein Octree genutzt). Gute Ergebnisse wurden erzielt, wenn die Caches für die Entfernung markiert wurden, die weniger als 15% von ϑ_ε und d_ε voneinander abweichen. Von den markierten Caches wird immer der Cache entfernt, der eine kleinere Fläche in seinem Voxel repräsentiert.

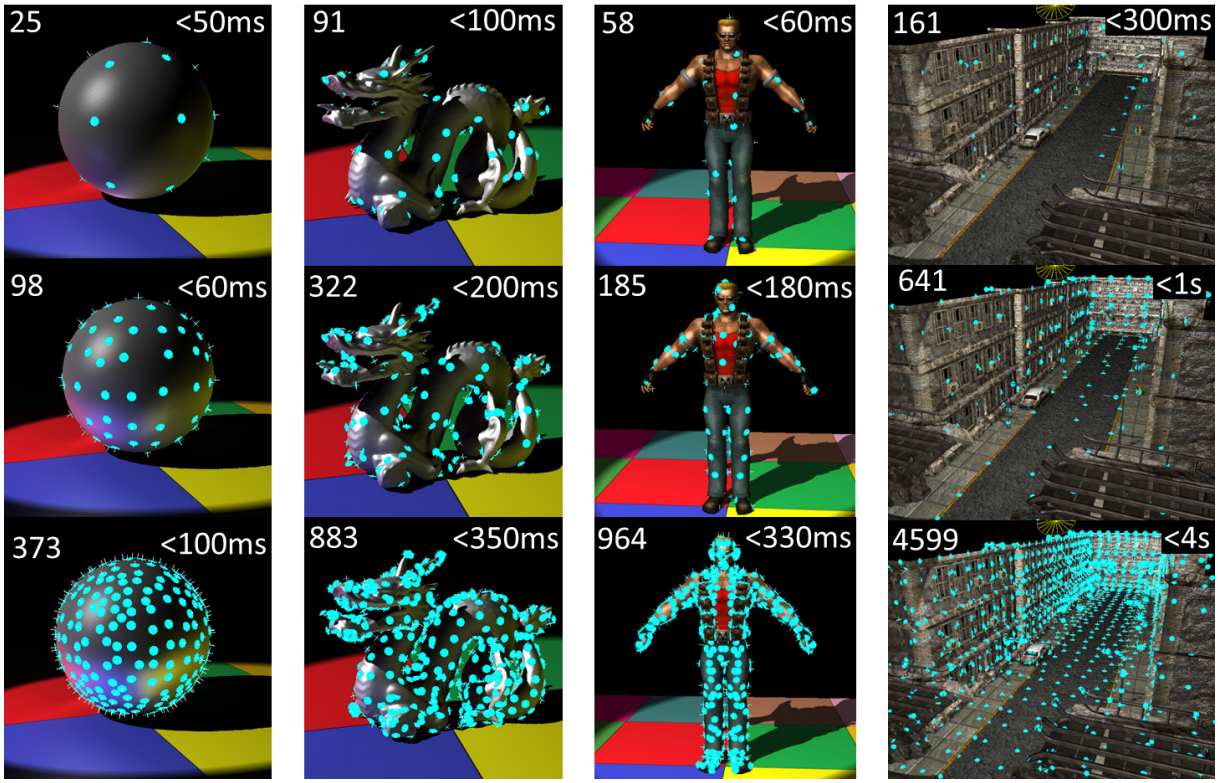

Abbildung 5.40: Beispiele für Verteilungen, die mit dem Algorithmus innerhalb von 1-4 Sekunden berechnet wurden. Die Anzahl der erzeugten Caches steht jeweils oben links, während die Erzeugungszeiten oben rechts stehen. Man erkennt deutlich, dass der Algorithmus für verschiedene Modelle und verschiedene Mengen von Caches gute Verteilungen erzeugt.

Abbildung 5.40 zeigt einige Ergebnisse von Cache-Verteilungen für verschiedene Modelle. Die Laufzeitkomplexität des Algorithmus kann wie folgt abgeschätzt werden:

$$O\big(t_m \log(t_m) + v(t_v \log(t_m) + c_v t_v + c_v \log(t_v)) + vc_v log(vc_v)\big). \tag{5.98}$$

t_m bzw. t_v kennzeichnet die Menge der Dreiecke für ein Modell bzw. einen Voxel. v entspricht der Menge der Voxel und c_v ist die Menge der Caches pro Voxel, die maximal $360°/\vartheta_\varepsilon$ sein kann. Der Term $t_m \log(t_m)$ resultiert aus der Konstruktion des Octrees, $t_v \log(t_m)$ entspricht der Abfrage der Dreiecke für einen Voxel, $c_v t_v$ steht für die Bildung der Caches, $c_v \log(t_v)$ für die Neupositionierung der Caches und schlussendlich $vc_v log(vc_v)$ für die Entfernung von ähnlichen Caches.

5.6.4 Animationen

Die Verteilung der Caches im Objektraum des Modells birgt einen essentiellen Vorteil bei Animationen. Wird das Objekt als Ganzes transformiert, können die Caches die gleiche Transformation nutzen. Geht man davon aus, dass das Modell nur verschoben, rotiert oder gleichmäßig skaliert wird, dann können die Gewichtungen für die Oberflächenpunkte zu den Caches im Voraus berechnet werden, denn diese werden durch die Transformation nicht verändert. Verändert sich jedoch die Form des Objekts, müssen die Gewichtungen zur Laufzeit angepasst werden. Dies ist aber dank der einfachen

Berechnungsgrundlage, ohne große Kosten, in einem Shader-Programm möglich. Durch die Zuordnung eines Caches zu einem Dreieck können auch Bone-Animations auf diesen übertragen werden. Hierfür müssen im Vorhinein nur die Gewichtungen und die Referenzen zu den Bones für jeden Eckpunkt des Dreiecks bekannt sein. Anhand der Lage des Caches im Dreieck können die Bone-Gewichtungen für den Cache durch die Eckpunkte des Dreiecks berechnet werden. Hierbei muss jedoch beachtet werden, dass der Cache unter Umständen mehr Bone-Referenzen halten muss als ein einzelner Vertex, denn es kann vorkommen, dass die drei Eckpunkte des Dreiecks verschiedene Bones referenzieren. In diesem Fall muss sichergestellt werden, dass der Cache alle korrespondierenden Bones berücksichtigt. Sofern die Modelltopologie es zulässt, können alternativ die Caches auf die Positionen vorhandener Vertizes des Modells verteilt werden. Hierdurch kann der Cache die Bone-Referenzen und -Gewichtungen des Vertex kopieren und muss keine eigenen Werte berechnen.

Grundsätzlich lässt sich festhalten, dass das LightSkin-Verfahren jede Art von Modellanimationen unterstützt, die die Nachbarschaftsverhältnisse der Caches nicht verändern. Animationen, die die Topologie der Dreiecke verändern, in denen die Caches liegen, können nicht ohne weiteres abgebildet werden. Dies hängt damit zusammen, dass die Referenzen zu den Caches für die Oberflächenpunkte im Voraus festgelegt werden müssen. Daher können die Caches während der Laufzeit der Simulation nicht umsortiert, sondern nur neu gewichtet werden. Allerdings verändern die meisten Animationen die Topologie des Dreiecksnetzes nicht, selbst komplexe Morphing-Animationen belassen die Topologie in der Regel unverändert und verschieben stattdessen nur die Vertizes eines Modells.

6 Implementierung des LightSkin-Verfahrens

6.1 Implementierung für die Grafikkarte

In diesem Kapitel wird der Fokus auf die effiziente Implementierung des LightSkin-Verfahrens für die GPU gelegt. Die Implementierung wurde so entworfen, dass bereits das Shader-Model 3.0 zur Simulation der globalen Beleuchtungseffekte ausreicht. Die benötigten Features beschränken sich im Wesentlichen auf

- das gleichzeitige Schreiben von mehreren Buffern durch einen Fragment-Shader,
- die Unterstützung von Fließkomma-Texturen,
- das additive Mischen (Blenden) von Fließkomma-Texturen und
- das Lesen von Texturen innerhalb des Vertex-Shaders.

Die Implementierung profitiert ebenfalls von weiteren Features, die mit dem Shader-Model 4 bzw. 5 eingeführt wurden, diese sind jedoch nicht zwingend erforderlich.

Eine wichtige Laufzeitoptimierung für die LightSkin-Pipeline bestand darin, den Speicherzugriff innerhalb der Shader-Programme möglichst effizient zu organisieren. Im Allgemeinen ist der Zugriff auf den Speicher, sowohl bei der CPU- als auch bei der GPU-Programmierung, bedeutend langsamer als die Ausführung einer arithmetischen Operation auf Registern [WPS*10]. Typischerweise ist der sequentielle Zugriff auf den Grafikkartenspeicher bedeutend schneller als der zufällige Zugriff [Buc05], zusätzlich besitzen moderne Grafikkarten L1/L2 Cache-Kaskaden, die den Zugriff enorm beschleunigen können, wenn eine hohe Wiederverwendung der Inhalte sichergestellt wird [Dog12]. Crassin formuliert in seiner Dissertation [Cra11] die Vermutung, dass für zukünftige Generationen von Grafikkarten die Optimierung des Speicherzugriffs eines der wichtigsten Ziele zur effizienten Parallelisierung darstellen wird.

In den nachfolgenden Unterkapiteln wird zuerst erklärt, wie die Caches im Gesamtsystem repräsentiert werden, anschließend wird ein Überblick über die komplette LightSkin-Render-Pipeline gegeben. Schlussendlich werden noch optionale Optimierungsverfahren vorgestellt, die die Beleuchtungsberechnung weiter beschleunigen können.

6.1.1 Cache-Texturen

Im vorherigen Unterkapitel wurde bereits beschrieben, dass der effiziente Speicherzugriff eine der elementarsten Herausforderungen für die Parallelisierung von Algorithmen auf der GPU ist. Aus diesem Grund werden die LightSkin-Caches in zweidimensionalen Texturen gespeichert, bei denen jeder Pixel einen Cache repräsentiert. Da moderne Grafikkarten häufig sowohl einen dedizierten L1- als auch einen L2-Textur-Cache besitzen, ist durch die Speicherung der Interpolations-Caches in Texturen genau dann ein schneller Zugriff sichergestellt, wenn eine hohe Wiederverwendung der Daten möglich ist. Darüber hinaus ist es bei dieser Form der Datenrepräsentation leicht möglich, die Interpolations-Caches durch die Grafikkarte zu aktualisieren bzw. zu erweitern. Hierfür müssen lediglich entsprechende Render-Targets erzeugt werden.

Nachdem die Caches erzeugt wurden (wie in Kapitel 5.6 beschrieben), werden deren initiale Daten in zwei Fließkomma-Texturen quadratischer Größe gespeichert. Eine Textur nimmt die Position und den

Reflexionsexponenten und die andere Textur die Normale und die Fläche des Caches auf. Wird ein Modell gezeichnet, werden dessen Cache-Texturen an die Render-Routine übergeben, die diese wiederum in einem Texturatlas speichert. Nachdem alle Modelle gezeichnet wurden, enthält dieser Texturatlas dann alle Cache-Texturen der Modelle, die im Blickfeld der Kamera liegen (siehe Abbildung 6.1). Die Erzeugung des Texturatlas geschieht auf der Grafikkarte. Jedes Modell erzeugt einen Zeichenaufruf mit seinen Cache-Texturen und den Modelltransformationen als Parameter, hierdurch lassen sich die Cache-Koordinaten bereits bei der Einordnung in den Texturatlas in Weltkoordinaten transformieren. Um später wieder einen Cache einem Modelloberflächenpunkt zuzuordnen, müssen nur sein lokaler Index innerhalb der Modell-Cache-Textur und sein globaler Index (Offset) für den Texturatlas bekannt sein.

Die Erweiterung der Cache-Daten zur Laufzeit der Simulation wird bewerkstelligt, indem weitere Texturen erzeugt werden: Möchte man zum Beispiel für alle Caches die Position und den Lichtstrom der Proxy-Lichtquellen bestimmen, müssen nur zwei zusätzliche Render-Targets erzeugt werden (eines für die Position und eines für den Lichtstrom der Proxy-Lichtquelle), die die gleichen Abmessungen wie der Texturatlas besitzen. Die Beleuchtungsberechnung findet dann durch einen Fragment-Shader statt, der als Eingangstextur den Texturatlas erhält. Nach der Beleuchtungsberechnung korrespondiert ein Ergebnispixel im Render-Target zu einem Eingangspixel im Texturatlas. Also erhalten die Caches zusätzliche Laufzeitdaten durch das Erzeugen weiterer Texturen. Ein Cache entspricht dann einer eindeutigen Texturkoordinate in diesen Texturen.

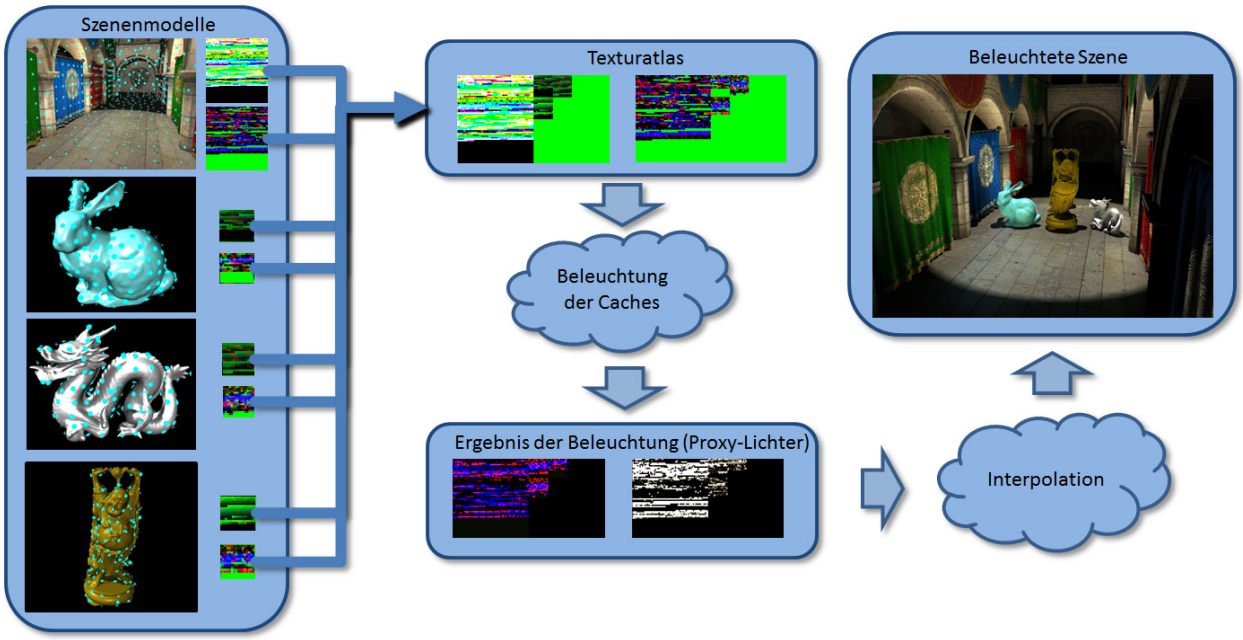

Abbildung 6.1: Prinzip der Cache-Texturen. Jedes Modell speichert seine Caches in zwei Texturen. Diese werden vor der Beleuchtung in einem Texturatlas gesammelt. Die Beleuchtung wird auf dem kompletten Texturatlas ausgeführt. Das Ergebnis der Beleuchtung sind weitere Texturen, die die Eigenschaften der Caches erweitern. Ein Cache entspricht einer Texturkoordinate in allen erzeugten Texturen.

6.1.2 Pipeline

Das Prinzip der Cache-Texturen ist ein essentieller Baustein zum Verständnis der LightSkin-Pipeline. Diese besteht aus 11 Basisverarbeitungsschritten, die in Abbildung 6.2 aufgeführt sind. Es folgt eine kurze Zusammenfassung der einzelnen Verarbeitungsstufen:

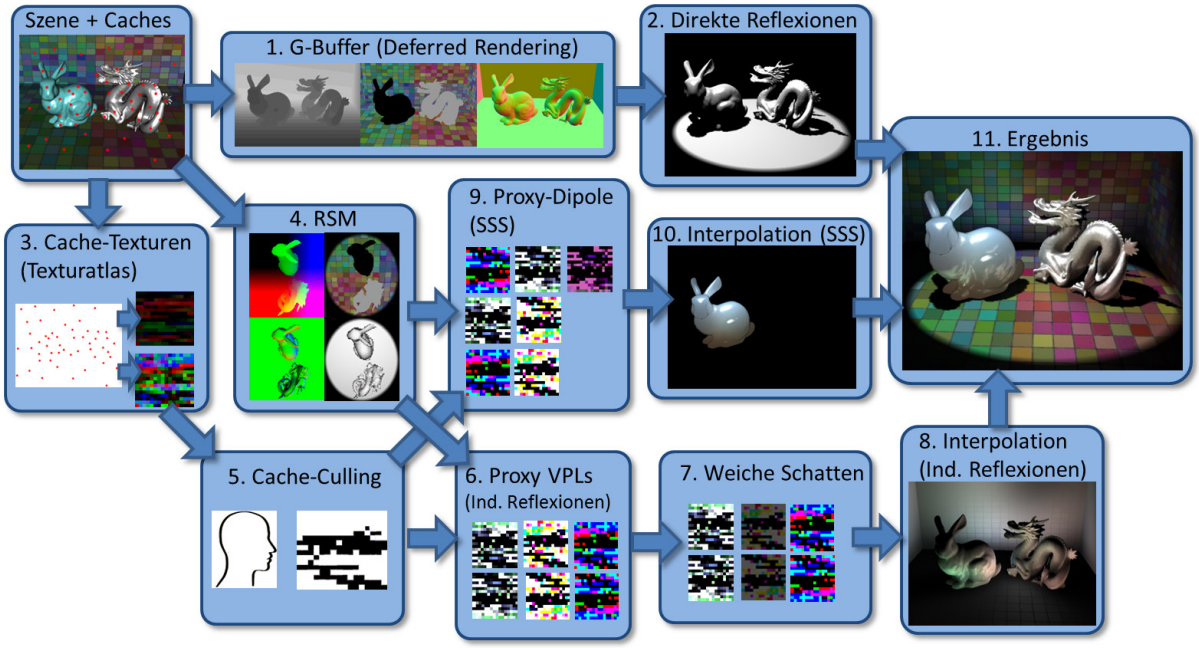

Abbildung 6.2: Darstellungs-Pipeline des LightSkin-Verfahrens. (1) Erzeugung der G-Buffer für das Deferred Rendering, (2) Berechnung der direkten Beleuchtung mit Schlagschatten, (3) Transformation der Caches und Speicherung im Texturatlas, (4) Erzeugung der erweiterten RSM, (5) Ausmaskierung der Caches, die nicht zum Kamerabild beitragen (Culling), (6) Erzeugung der Proxy-Lichtquellen für die indirekten Reflexionen, (7) Dämpfung des Lichtstroms der Proxy-Lichtquellen, um weiche Schatten zu erzeugen, (8) Interpolation des indirekten Lichts über die Modelloberflächen mit Hilfe der Proxy-Lichtquellen, (9) Berechnung der Proxy-Dipole für das Subsurface-Scattering, (10) Interpolation des gestreuten Lichts über die Modelloberfläche mit Hilfe der Proxy-Dipole, (11) Zusammenfassung der Teilergebnisse zum fertigen Bild.

Stufe 1 & 2: In diesen Verarbeitungsstufen werden auf Basis der übergebenen Modelldaten die G-Buffer für das Deferred Shading [ST90] gefüllt, die wiederum für die Berechnung von lokalen direkten Reflexionen genutzt werden. Da es sich bei diesen Arbeitsschritten um konventionelle Verfahren handelt, bedürfen sie keiner weiteren Erläuterung.

Stufe 3: Hier werden die Cache-Texturen in den Texturatlas übertragen und entsprechend ihrer Modelltransformation in den Weltkoordinatenraum transformiert. Dieser Verarbeitungsschritt wurde bereits in Kapitel 6.1.1 beschrieben.

Stufe 4: Die Erzeugung der erweiterten Reflective Shadow-Map wird in dieser Stufe vorgenommen. Die RSM enthält vier Fließkomma-Texturen, in denen die folgenden Daten gespeichert sind: Position, Normale, Fläche, diffus reflektierter Lichtstrom, die „innere“ Beleuchtungsstärke für SSS-Modelle und die Material-IDs der Objektoberflächen. Für Subsurface-Scattering-Materialien wird der diffus reflektierte Lichtstrom auf null gesetzt. Wie die Daten im Einzelnen erzeugt werden, war Thema von Kapitel 5.2. Weitere Implementierungsdetails sind in [DS06] und [DS03] zu finden.

Stufe 5: In dieser Stufe werden die Caches aus dem Texturatlas ausmaskiert, die zum betrachteten Kameraausschnitt keinen Betrag liefern. Eine einfache Sichtbarkeitsprüfung für einen Cache im Kamerabild, zum Beispiel durch Z-Buffering, reicht nicht aus, um sagen zu können, ob dieser zum

Kamerabild beiträgt. Deshalb musste ein neues Verfahren entwickelt werden. Wie dieses Verfahren funktioniert, ist Thema von Kapitel 6.1.5.

Stufe 6: Für die indirekte Beleuchtung werden für jeden Cache zwei Proxy-Lichtquellen erzeugt: eine für diffuse und eine für glänzende Reflexionen (siehe Kapitel 5.3). Eine Proxy-Lichtquelle wird durch ihre Position und ihren Lichtstrom repräsentiert. Demnach werden vier zusätzliche Cache-Texturen benötigt. Für die Erzeugung weicher Schatten wird zusätzlich die Standardabweichung der VAL-Positionen berechnet. Da diese für diffuse und glänzende Reflexionen verschieden sind, werden zwei weitere Cache-Texturen benötigt. In dieser Verarbeitungsstufe muss jedes VAL aus der RSM auf jeden Cache angewendet werden. Dies ist ein rechenintensiver Prozess, weswegen in Kapitel 6.1.3 eine effiziente Implementierung vorgestellt wird.

Stufe 7: Zur Erzeugung von weichen Schatten wird der Lichtstrom der virtuellen Proxy-Lichtquellen aus dem vorherigen Verarbeitungsschritt individuell gedämpft. Hierfür muss, wie in Kapitel 5.4 beschrieben, geprüft werden, ob andere Caches die Proxy-Lichtquellen verdecken.

Stufe 8: Die Ergebnisse für die Proxy-Lichtquellen aus Schritt 6 respektive Schritt 7 werden in dieser Verarbeitungsstufe zur durchgehenden Interpolation über die sichtbaren Oberflächen genutzt. Hier finden die vorher erläuterten Interpolationsverfahren Anwendung. In Kapitel 6.1.4 werden zwei praktische Implementierungen vorgeschlagen, wie diese Interpolation mit Hilfe eines Vertex- bzw. Fragment-Shaders durchgeführt werden kann. Die Ergebnisse der Interpolation werden in einem separaten Buffer gespeichert.

Stufe 9: Für das Subsurface-Scattering werden für die drei Farbkomponenten drei Proxy-Dipole erzeugt, sowie ein Normalisierungsfaktor gespeichert. Diese Proxy-Daten werden wieder in Cache-Texturen gespeichert. Die Implementierung dieser Stufe ähnelt stark der von Stufe 6, weswegen auf eine dedizierte Betrachtung verzichtet wird. Im Wesentlichen kommt es auch in dieser Verarbeitungsstufe darauf an, dass die VALs möglichst effizient auf die Caches angewendet werden.

Stufe 10: In dieser Stufe werden die erzeugten Proxy-Dipole zur Interpolation angewendet, wie dies in Kapitel 5.5.4 beschrieben wurde. Die Implementierung unterscheidet sich nur in den Interpolationsformeln von der aus Stufe 8, weswegen hier ebenfalls von einer dedizierten Beschreibung Abstand genommen wurde. Das Ergebnis der Interpolation wird in einem separaten Buffer gespeichert.

Stufe 11: Im letzten Verarbeitungsschritt werden die Beleuchtungsergebnisse für die direkten und indirekten Reflexionen, sowie für das Subsurface-Scattering, mit dem G-Buffer für diffuse Materialen verrechnet und ergeben das fertige Bild.

Für die Verarbeitungsstufen eins bis vier ist eine separate Betrachtung in den folgenden Kapiteln nicht vorgesehen, da diese in dieser Arbeit bereits erfolgte. Des Weiteren wurde von separaten Beschreibungen der einzelnen Verarbeitungsstufen Abstand genommen, da diese zu großen Redundanzen in dieser Dissertation führen würden, die nicht zum Verständnis beitragen würden. Konzeptionell lassen sich die Stufen 6, 7 und 9 gut zusammenfassen, da es in diesen im Wesentlichen darum geht, eine große Menge von VALs oder Caches (Verdeckung) auf die sichtbaren Caches der Szene anzuwenden. Die Implementierung dieser Verarbeitungsschritte variiert nur durch das Ein-

setzen der passenden Gleichungen zur Bildung von Proxy-Lichtquellen bzw. zur Verdeckungsrechnung. Die gleiche Argumentation lässt sich auch für die Stufen 8 und 10 führen. Für diese Stufen werden zwei alternative Verfahren zur Implementierung vorgeschlagen, für die die entsprechenden Interpolationsgleichungen aus den vorherigen Kapiteln eingesetzt werden können.

6.1.3 Erzeugung der Proxy-Lichtquellen, -dipole & weicher Schatten (Stufe 6,7 & 9)

Die Erzeugung der Proxy-Lichtquellen für indirekte Reflexionen, sowie die Erzeugung der Proxy-Dipole für das Subsurface-Scattering teilen sich die Eigenschaft, dass eine große Menge von VALs aus der RSM auf die sichtbaren Caches angewendet werden muss, um die Proxy-Daten zu erzeugen. Analog muss für die Bildung weicher Schatten eine große Menge an Caches auf die sichtbaren Caches angewendet werden, um den Lichtstrom der Proxy-Lichtquellen zu dämpfen. Es wird also nach einer effizienten Implementierung gesucht, die es erlaubt, mehrere tausend VALs auf mehrere tausend Caches anzuwenden. Vereinfacht lassen sich die Stufen wie folgt darstellen:

Listing 6.1: Prinzip der Berechnung von indirekten Reflexionen, Subsurface-Scattering und weichen Schatten.

```
// Indirekte Reflexionen
for each visible Cache
  1. Lese Cache-Daten: Position & Normale.
  for each VAL in RSM
    2. Lese VAL-Daten: Position, Normale, Fläche & reflektierter Lichtstrom.
    3. Akkumuliere Proxy-Lichtquellen: Position, Lichtstrom & Standardabweichung.
    4. Akkumuliere Mittelungsgewicht.
  end
  5. Teile akkumulierte Proxy-Lichtquellen durch akkumuliertes Mittelungsgewicht.
end

// Subsurface-Scattering
for each visible Cache
  1. Lese Cache-Daten: Position & Normale.
  for each VAL in RSM
    2. Lese VAL-Daten: Position, Normale, Fläche & innere Beleuchtungsstärke.
    3. Akkumuliere Proxy-Dipole: Positionen & Normalisierungsfaktor.
    4. Akkumuliere Mittelungsgewicht.
  end
  5. Teile akkumulierte Proxy-Dipole durch akkumuliertes Mittelungsgewicht.
end

// Weiche Schatten
for each visible Cache
  1. Lese Cache-Daten: Pos., Normale, Proxy-Pos., Proxy-Lichtstrom, Proxy-Standardabweichung.
  for each Cache
    2. Lese fremde Cache-Daten: Position, Fläche, Normale.
    3. Akkumuliere Verdeckungsgrad der Proxy-Lichtquelle durch fremden Cache.
  end
  4. Dämpfe Proxy-Lichtstrom durch Verdeckungsgrad.
end
```

Durch den Pseudocode wird die hohe Kohärenz zwischen den Verfahren sofort ersichtlich. Zur besseren Lesbarkeit wird in den folgenden Erläuterungen davon ausgegangen, dass VALs auf die Caches angewendet werden, alle folgenden Überlegungen gelten aber ebenfalls für die Berechnung der weichen Schatten.

Die äußere Schleife im obigen Listing wird pro sichtbarem Cache ausgeführt. Da die Ergebnisse für die Caches in Texturen bzw. Render-Targets gespeichert werden, lässt sich der äußere Schleifenkörper als Fragment-Shader implementieren. Eine naive GPU-Implementierung würde darin bestehen, pro

VAL ein Rechteck im Vertex-Shader zu erzeugen, das sich über den kompletten Bildschirm erstreckt. Dies entspricht dem Prinzip eines Splatting-Ansatzes, bei dem jeder Pixel durch einen Fragment-Shader bearbeitet wird, der von dem Splatting-Geometrieobjekt (hier Rechteck) überlappt wird. So lässt sich die innere Schleife im obigen Listing simulieren. Es werden so viele Rechtecke erzeugt, wie VALs in der RSM stehen. Die Akkumulation der Proxy-Daten kann erreicht werden, indem die Ergebnisse des Fragment-Shaders auf die vorhandenen Werte des Framebuffers addiert werden (additives Blending). Allerdings muss dann ein zusätzliches Render-Target erzeugt werden, das die Mittelungsgewichte speichert (diese werden ebenfalls akkumuliert). In einem zweiten Verarbeitungsschritt werden die akkumulierten Proxy-Daten durch die akkumulierten Mittelungsgewichte geteilt.

Listing 6.2: Shader zur Erzeugung von Proxy-Daten auf Basis eines Splatting-Ansatzes.

```
// Durchgang 1
SetBlendState(ADDITIVE);
VertexShader1()
{
        Übergebe Texturkoordinaten für RSM.
}
FragmentShader1()
{
        Lese VAL aus RSM.
        Lese Cache-Daten aus Cache-Texturatlas.
        Berechne & speichere Proxy-Daten von VAL.
        Berechne & speichere Gewicht von VAL.
}

// Durchgang 2
SetBlendState(OPAQUE);
VertexShader2()
{
        Übergebe Bildschirmposition als Texturkoordinaten.
}

FragmentShader2()
{
        Lese Proxy-Daten und Gewicht.
        Speichere Proxy-Daten geteilt durch das Gewicht.
}
```

Der erste Durchgang erhält beim Zeichenaufruf ein Modell, das aus so vielen Rechtecken besteht, wie es VALs in der RSM gibt, während der zweite Durchgang nur ein Modell, bestehend aus einem Rechteck, erhält.

Diese Implementierung hat den elementaren Nachteil, dass der Fragment-Shader jetzt mn mal ausgeführt werden muss, wobei m der Anzahl der Caches und n der der VALs entspricht. Dies hat zur Folge, dass mn mal mehrere Render-Targets beschrieben werden, dass mn mal die Daten aus der RSM gelesen werden müssen und dass mn mal die Daten des Caches aus dem Cache-Texturatlas gelesen werden müssen. Der Textur-Cache wird zwischen den Fragment-Shadern geteilt, weswegen die Daten aus der RSM direkt vom Cache bedient werden können, alle anderen Lese- und Schreibzugriffe sind jedoch entsprechend ineffizient. Die Implementierung kann also als ineffizient angesehen werden, was den Speicherzugriff betrifft. Dies ist nicht weiter verwunderlich, denn Splatting-Ansätze sind nur dann effizient, wenn die Splatting-Geometrie nur einen Teil des Framebuffers überlappt. In diesem Fall entsprechen die Bereiche, die durch die Geometrie aktualisiert werden sollen, nur einem Bruchteil des kompletten Framebuffers und können demnach effizient verarbeitet werden.

Für das LightSkin-Verfahren wird daher ein Gathering-Ansatz anstelle eines Splatting-Ansatzes genutzt. Anstatt das Licht von den Lichtquellen zu den Oberflächenpunkten zu verteilen, wird das Licht aus der RSM durch die Oberflächenpunkte „eingesammelt“. Dieses Einsammeln lässt sich realisieren, indem man im Fragment-Shader über die Lichtquellen der RSM iteriert. Um ein gutes Textur-Cache-Verhalten für das Lesen der VALs zu gewährleisten, wird pro Fragment-Programm nur über eine beschränkte Menge von VALs iteriert. Wie viele VALs in einem Durchgang verarbeitet werden, hängt von der Größe des Textur-Caches der Grafikkarte ab. Es werden genau so viele VALs verarbeitet, wie im Textur-Cache gespeichert werden können. Da der Textur-Cache von allen Fragment-Shadern gemeinsam genutzt wird, erhalten alle Fragmente nach dem ersten Lesezyklus ihre Daten aus dem Cache. Im obigen Pseudocode muss der *Fragmentshader1* ersetzt werden, um das Gathering zu realisieren:

Listing 6.3: Angepasster Fragment-Shader zur Erzeugung von Proxy-Daten auf Basis eines Gathering-Ansatzes.

```
FragmentShader1()
{
  Lese Cache-Daten: Position & Normale.
  for k VALs in RSM
    Lese VAL aus RSM
    Akkumuliere Proxy-Daten von VAL.
    Akkumuliere Gewicht von VAL.
  end
  Schreibe Gewicht und Proxy-Daten.
}
```

Dieses Fragment-Programm muss jetzt nur noch n/k mal aufgerufen werden und liest bzw. schreibt deshalb nur n/k mal die Cache-Daten. Die k Leseaufrufe auf die RSM pro Fragment werden alle aus dem Textur-Cache bedient. Moderne Grafikkarten besitzen nach [Dog12] einen 4-16 KByte großen L1-Textur-Cache, so dass k zwischen 128 und 512 gewählt werden kann (für indirekte Reflexionen müssen 32 Bytes aus der RSM für ein VAL gelesen werden). Implementierungstests haben ergeben, dass die Gathering-Variante um das zehnfache schneller ist als ein Splatting-Ansatz [LB13a,LB13b].

Die Algorithmen aus Kapitel 5.3.4 und 5.4.2 können mit der hier vorgestellten Shader-Implementierung nun effizient berechnet werden.

6.1.4 Interpolation: Anwendung von Caches auf Modelloberflächen (Stufe 8 & 10)

Die Übertragung bzw. Anwendung der Cache-Ergebnisse auf alle sichtbaren Oberflächen der Modelle kann auf zwei Arten geschehen: auf Vertex- oder Texturebene. Beide Arten unterscheiden sich hauptsächlich darin, wie die Referenzen zu den Caches gespeichert werden (siehe Abbildung 6.3). Dabei ist eine Cache-Referenz nichts anderes als ein 16-Bit-Index, der mit Hilfe einer Offset-Adresse zu einer Texturkoordinate für den Cache-Texturatlas umgerechnet werden kann. Bei der Interpolation auf Vertexebene werden acht bis sechzehn dieser Indizes im Vertex gespeichert, während bei der Interpolation auf Texturebene diese Indizes in separaten Texturen gespeichert werden. Wie viele Indizes letztendlich in einem Vertex bzw. in einem Texel gespeichert werden, hängt von Qualitäts- und Laufzeitüberlegungen ab. Viele Referenzen führen zu einer besseren Qualität, benötigen jedoch mehr Speicher und mehr Rechenzeit bei der Interpolation. Eine geringe Anzahl von Referenzen führt wiederum zu einer schlechteren Qualität, jedoch zu einem besseren Laufzeitverhalten. In den Testszenarien dieser Arbeit haben sich acht bis sechzehn Referenzen als guter Kompromiss zwischen Qualität und Laufzeit bewährt. Die Zuordnung der Referenzen zu den Vertizes bzw. Texeln muss im Voraus geschehen und kann während der Laufzeit nicht mehr verändert werden.

Welcher Cache welchem Vertex bzw. Texel zugeordnet wird, kann anhand der Fehlermetrik in Gleichung (5.95) aus Kapitel 5.6.2 entschieden werden. Die Caches, die den kleinsten Fehler zum Oberflächenpunkt aufweisen, werden in dessen Indexliste aufgenommen. Wenn das Modell während der Simulation durch Animationen verformt werden kann, sollte man bei der a-priori-Zuordnung die nächstliegenden Caches für einen Vertex bzw. Texel wählen. Die Normalendifferenz kann für diesen Fall ignoriert werden, da zu erwarten ist, dass sich diese während der Animation deutlich verändert.

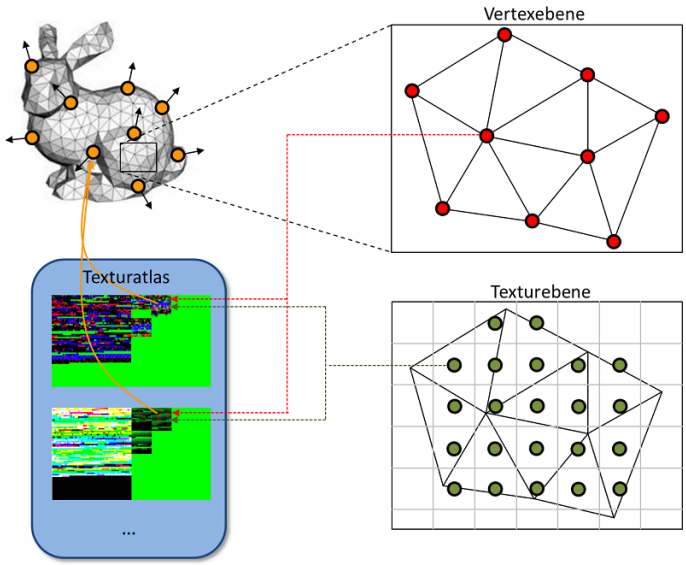

Abbildung 6.3: Zwei verschiedene Arten, die Interpolation zu implementieren. Die Indizes zu den Caches können entweder pro Vertex (oben) oder pro Texel (unten) gespeichert werden. Der Zugriff zu den Caches erfolgt über den Cache-Texturatlas.

Sowohl die Interpolation auf Vertexebene, als auch die Interpolation auf Texturebene sind relativ einfach zu implementieren. Die Vertex- und Fragment-Shader für die Interpolation auf Vertexebene lassen sich wie folgt zusammenfassen:

Listing 6.4: Interpolations-Shader auf Vertexebene.

```
VertexShader()
{
        1. Lese Cache-Indizes aus Vertexattributen.
        2. Bestimme Texturkoordinaten für Caches.
        3. Lese Daten von Caches aus Cache-Texturatlas.
        4. Berechne Gewichte für die Caches.
        5. Berechne interpolierte Proxy-Lichtquelle für Vertex anhand der Gewichte.
        6. Übergebe interpolierte Proxy-Lichtquelle.
}
FragmentShader()
{
        1. Lese Fragment-Normale von G-Buffer.
        2. Verwende interpolierte Proxy-Lichtquelle mit lokalem Beleuchtungsmodell.
        3. Übergebe Farbe für Pixel.
}
```

Der Vertex-Shader erhält als Eingang die Geometriedaten der Szene (mit den Cache-Indizes in den Vertexattributen). Man erkennt anhand dieses Listings, dass für jedes Fragment eine eigene Proxy-Lichtquelle berechnet wird. Dies erlaubt die Berücksichtigung individueller Normalen aus dem G-Buffer pro Fragment. So können filigrane Strukturen auf den Objektoberflächen bei der Beleuchtung berücksichtigt werden. Da aber nur ein Proxy-Licht pro Fragment erzeugt wird, können Variationen der Fragment-Normalen dieses nur skalieren, jedoch nicht das Spektrum der Reflexion verändern.

Darüber hinaus ist bei diesem Ansatz die Reflexion von der Triangulation des Modells abhängig. Die individuellen Caches werden nur für jeden Vertex berücksichtigt. Deshalb sollte die Vertexdichte höher als die Cache-Dichte sein. Ansonsten werden Details bei der Reflexion reduziert, wie dies in Abbildung 6.4 dargestellt ist.

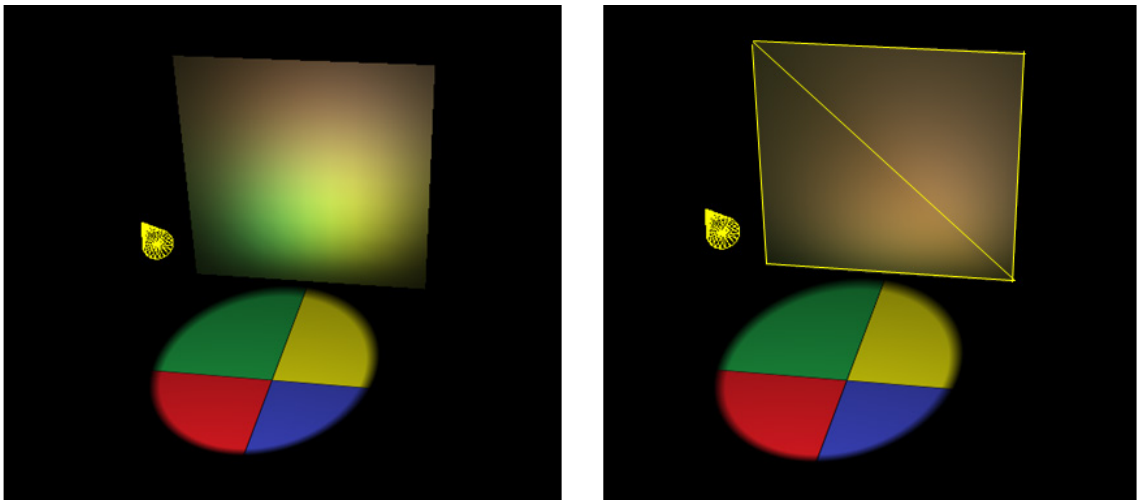

Abbildung 6.4: Abhängigkeit der Reflexion von der Tesselierung des Objekts. In beiden Szenen wurden 64 Caches auf der Fläche verteilt. In der linken Szene besteht die Fläche aus 128 Dreiecken und in der rechten nur aus zwei Dreiecken. Man kann in der rechten Szene deutlich erkennen, dass Details in der Reflexion fehlen.

Die Abhängigkeit der Reflexion von der Tesselierung des Objekts kann durch die Interpolation auf Texturebene aufgehoben werden. Allerdings müssen dann pro Objekt zusätzliche Texturen gespeichert werden, die die Cache-Indizes enthalten. Diese Texturen müssen so auf das Modell „gezogen“ werden, dass diese ein möglichst verzerrungsfreies Texture-Mapping besitzen. Da statische Light-Maps ebenfalls ein solches Mapping benötigen, existieren in den meisten 3D-Modellierungsprogrammen spezielle Tools, die dieses Mapping automatisiert erzeugen können. Die Implementierung der Interpolation auf Texturebene ist intuitiv und wird hier nur aus Gründen der Vollständigkeit aufgeführt:

Listing 6.5: Interpolations-Shader auf Texturebene.

```
FragmentShader()
{
	1. Lese Texturkoordinaten von Vertexattributen (verzerrungsfreies Mapping).
	2. Lese Cache-Indizes aus Zusatztexturen.
	3. Bestimme Texturkoordinaten für Caches.
	4. Lese Daten von Caches aus Cache-Texturatlas.
	5. Lese Fragment-Normale aus G-Buffer.
	6. Berechne Gewichte für die Caches.
	7. Berechne interpolierte Proxy-Lichtquelle für Fragment anhand der Gewichte.
	8. Verwende interpolierte Proxy-Lichtquelle mit lokalem Beleuchtungsmodell.
	9. Übergebe Farbe für Pixel.
}
```

Bei der Interpolation auf Texturebene ist zu bedenken, dass der maximale Abstand d_{max} für das komplette Modell konstant und insbesondere sinnvoll gewählt werden muss. Ansonsten kann man bei hochauflösenden Texturen deutliche Kanten in den Reflexionen beobachten. Diese Kanten resultieren aus der sprunghaften Veränderung von d_{max}, wenn benachbarte Texel verschiedene Caches referenzieren (d_{max} ist der Abstand zum entferntesten Cache). Wegen dieser Kanten ist es auch nicht sinnvoll, die Proxy-Lichtquellen der Caches direkt auf das Fragment anzuwenden, da in diesem

Fall noch größere Sprünge entstehen könnten. Die Implementierung auf Texturebene hat außer der Unabhängigkeit von der Tesselierung noch den weiteren Vorteil, dass die Variation der Fragment-Normalen dazu führen kann, dass sich das Spektrum der Reflexion ändert und dieses nicht nur skaliert wird.

Ob das Verfahren auf Vertex- oder das auf Texturebene für die Interpolation verwendet wird, hängt von dem Anwendungsszenario der Simulation ab. Beim vertexbasierten Verfahren steigen die Kosten mit der Komplexität der Modellgeometrie, denn der Vertex-Shader übernimmt einen Großteil der Interpolationsberechnungen. Beim texturbasierten Verfahren steigen die Kosten mit der Auflösung des Framebuffers, da der Hauptteil der Interpolation im Fragment-Shader durchgeführt wird. Laufzeitmessungen zu den Interpolationsverfahren befinden sich in Kapitel 6.2.2.

6.1.5 Culling von Caches

Die bis hierhin vorgestellten Konzepte zum LightSkin-Verfahren erlauben noch keine Darstellung von wirklich großen und komplexen Szenen, denn gegenwärtig wird noch jeder Cache in der Szene durch eine Vielzahl von VALs beleuchtet. Sehr große Szenen benötigen entsprechend viele solcher Caches und sind demzufolge nicht effizient zu simulieren. Für die Darstellung ist es jedoch ausreichend, nur die Caches für die indirekte Beleuchtung zu nutzen, die tatsächlich einen Beitrag zum Kameraausschnitt liefern. Es wird also ein Culling-Algorithmus für Caches gesucht, der Caches von der teuren indirekten Beleuchtung ausschließt. Eine einfache Prüfung, ob ein Cache auf einer sichtbaren Oberfläche liegt, ist nicht ausreichend, um festzustellen, ob dieser zum Kamerabild beiträgt. Auch Caches auf nicht sichtbaren Oberflächen können zu sichtbaren Oberflächenpunkten beitragen (siehe Abbildung 6.5).

Die Lösung für dieses Problem besteht darin, nur die Caches zu beleuchten, die durch *sichtbare* Oberflächenpunkte referenziert werden. Je nach Art der Interpolation sind die Cache-Indizes entweder in einer Textur oder in den Vertexattributen gespeichert. So lässt sich für jeden gezeichneten Oberflächenpixel bestimmen, welche Caches dieser benötigt. Wenn die Caches in Texturen gespeichert werden, reicht das Auslesen dieser Texturen im Fragment-Shader. Wenn die Caches jedoch im Vertex gespeichert werden, muss beachtet werden, dass die Oberfläche eines Dreiecks mehr Cache-Indizes enthalten kann als seine einzelnen Vertizes (im schlimmsten Fall haben alle drei Vertizes eines Dreiecks verschiedene Cache-Indizes. Wenn man beispielsweise von acht Cache-Indizes pro Vertex ausgeht, benötigt das Dreieck 24 Cache-Indizes). In diesem Fall muss für die Interpolation auf Vertexebene für jeden Vertex ein separates Dreieck gezeichnet werden, das alle Cache-Indizes dieses Vertex repräsentiert. Daraus folgt, dass die Szene in dreifacher Form gezeichnet werden muss, damit für jeden Oberflächenpunkt feststeht, welche Caches dieser benötigt.

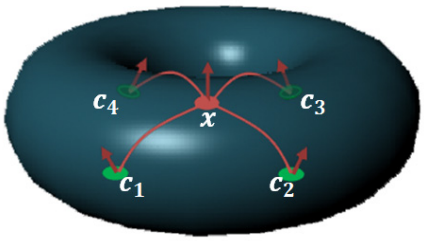

Abbildung 6.5: Culling von Caches. Für den Oberflächenpunkt $\boldsymbol{x}$ sind nicht nur die sichtbaren Caches auf der Vorderseite des Modells von Bedeutung ($\boldsymbol{c_1}, \boldsymbol{c_2}$), sondern auch die hinteren, nicht sichtbaren Caches ($\boldsymbol{c_3}, \boldsymbol{c_4}$).

Die vorherigen Ausführungen haben gezeigt, dass die Möglichkeit besteht, für jeden Oberflächenpixel des Framebuffers zu bestimmen, welche Cache-Indizes dieser enthält bzw. nutzt. Diese Cache-Indizes können dann in einem oder mehreren Render-Targets gespeichert werden (acht Cache-Indizes benötigen ein RGBA-Render-Target mit 32 Bit pro Kanal). Die Caches sind in diesen Render-Targets jedoch unorganisiert abgespeichert, so dass für einzelne Caches keine einfache Abfrage durchgeführt werden kann, ob diese im Render-Target enthalten sind oder nicht. Die Caches innerhalb des Render-Targets müssen also effizient gefiltert werden. Hierfür ist es hilfreich, die Caches nicht als Indizes, sondern als Graustufen zu verstehen. Man möchte dann wissen, welche Graustufen in dem Render-Target enthalten sind. Dies führt zu dem analogen Ansatz der Histogrammerzeugung von Bildern. Ein Histogramm gibt an, wie oft ein Grau- bzw. Farbwert in einem Bild vorhanden ist. Für den hier vorgestellten Culling-Ansatz ist weniger interessant, wie oft ein Cache-Index im Bild vorkommt, sondern *ob* er im Bild vorkommt. Diese Information steckt implizit im Histogramm. Der Culling-Algorithmus lässt sich also auf die Erzeugung eines Histogramms reduzieren. Die Erzeugung eines Histogramms mit der GPU ist jedoch problematisch, insbesondere, wenn nur das Shader-Model 3.0 Verwendung finden soll. Es existieren aber Ansätze, die ein Histogramm per GPU erzeugen können, wenn im Voraus bekannt ist, welche Graustufen bzw. Farben im Bild vorkommen können. Scheuermann und Hensley [SH07] haben einen solchen Ansatz vorgestellt. Das Prinzip dieses Ansatzes lässt sich wie folgt erklären:

Es wird ein Vertexgitter erzeugt, das genauso viele Vertizes enthält, wie es Pixel in dem zu untersuchenden Bild gibt (siehe Abbildung 6.6). Im Vertex-Shader wird nun für jeden Vertex ein anderer Pixel des Bildes gelesen. Die Farbe des gelesenen Pixels bestimmt, an welche Stelle der Vertex transformiert wird. Geht man beispielsweise von einem Eingangsbild mit 256 Graustufen aus, wird ein eindimensionales Render-Target, bestehend aus 256 Pixeln, erzeugt. Je nach gelesenem Wert im Vertex-Shader wird der Vertex dann auf die passende Stelle im Render-Target transformiert, der zu dieser Pixelfarbe passt. Wurde jeder Pixel im Eingangsbild abgefragt, steht in den Pixeln des Render-Targets die Verteilung der 256 Graustufen für das Bild.

Der Ansatz von Scheuermann und Hensley lässt sich nun für das LightSkin-Verfahren dahingehend modifizieren, dass der gelesene Cache-Index im Bild darüber entscheidet, an welche Stelle der Vertex transformiert wird. Als Ausgabe-Render-Target dient die Cache-Atlastextur. Der Vertex wird an die Stelle transformiert, an der sich der Cache im Texturatlas befindet, der zum Index passt. Anstatt die eigentliche Cache-Atlastextur zu aktualisieren, wird nur der Tiefenwert des Z-Buffers für den sich ergebenen Pixel verändert. Initial ist der Z-Buffer mit null gefüllt und jeder Cache, der sich im Eingangsbild befindet, setzt den Wert des Z-Buffers für den Cache-Atlas auf eins.

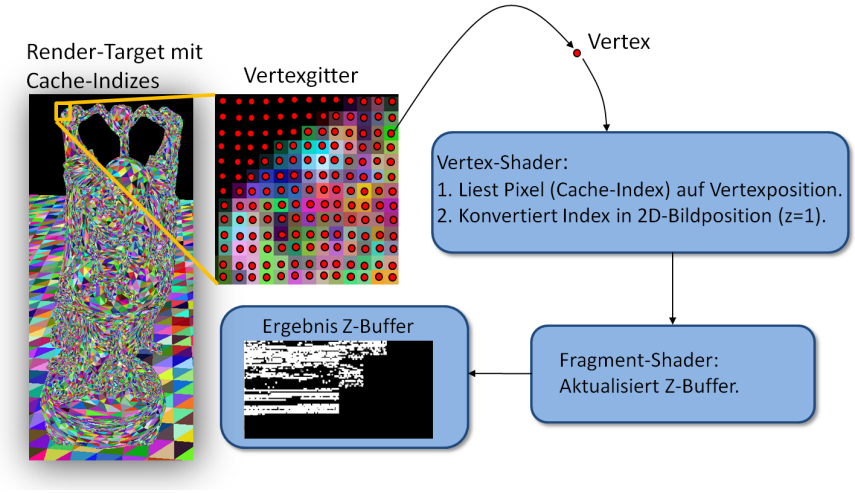

Abbildung 6.6: Prinzip des GPU-Cullings.

Das Resultat dieses Ansatzes ist eine binäre Maske im Z-Buffer, die diejenigen Caches ausmaskiert, die nicht zum finalen Bild beitragen. Für die anschließende Berechnung der Proxy-Lichtquellen für die Caches muss dieser Z-Buffer nur angewendet werden. Das Standardverhalten des Z-Buffers sorgt dann dafür, dass genau jene Fragmente nicht aktualisiert werden, die einen geringen Tiefenwert besitzen. Es werden also nur für die Caches Proxy-Lichtquellen bestimmt, die einen Tiefenwert von eins besitzen.

Schlussendlich können die einzelnen Schritte des Verfahrens wie folgt zusammengefasst werden (siehe auch Abbildung 6.6):

1. Erzeugung eines Render-Targets, das die Cache-Indizes in jedem Pixel speichert (hierfür muss jedes Modell der Szene gezeichnet werden).
2. Verwendung des Render-Targets als Eingangstextur für die GPU-Histogramm-Routine und Initialisierung des Ausgangs-Z-Buffers mit null.
3. Transfer des Gittermodells zur Histogramm-Routine. Dieses Modell besitzt genauso viele Vertizes, wie Pixel in der Eingangstextur vorhanden sind (mal der Anzahl der Caches pro Pixel).
4. Im Vertex-Shader: Nutzung der Position des Vertizes, um den passenden Pixel aus der Eingangstextur zu lesen.
5. Im Vertex-Shader: Der Wert des Pixels entspricht dem Cache-Index. Anhand des Cache-Index wird die Koordinate für die Position im Texturatlas bestimmt.
6. Im Fragment-Shader: Aktualisierung des Z-Buffers.
7. Anwendung des Z-Buffers für die Erzeugung der Proxy-Lichtquellen. Es werden nur für die Caches Proxy-Lichtquellen berechnet, deren Tiefenwerte größer null sind.

6.1.6 Weitere Optimierungen

Außer dem Cache-Culling-Verfahren existieren zwei weitere Möglichkeiten, die Laufzeit des LightSkin-Verfahrens zu optimieren. Eine Möglichkeit besteht darin, die Anzahl der VALs zu reduzieren, die auf einen Cache angewendet werden. Hierfür eignen sich Many-Light-Ansätze, die in Kapitel 3.1.7 vorgestellt wurden. Die andere Möglichkeit ist, die Anzahl der Caches adaptiv zu verringern bzw. zu erhöhen. Beide Ansätze sollen hier kurz erläutert werden:

Eine RSM kann sehr viele VALs enthalten, die alle auf die sichtbaren Caches der Szene angewendet werden müssen. Eine simple Reduktion der Auflösung der RSM ist nicht sinnvoll, da dies zu flackernden Artefakten bei Animationen führt, wie dies bereits mehrmals in dieser Dissertation erläutert wurde. Stattdessen kann ein adaptives Light-Clustering-Verfahren genutzt werden, welches homogene Lichtquellen zusammenfasst und so die Gesamtanzahl der VALs reduziert. Eine solche Zusammenfassung kann beispielsweise durch die Erzeugung von Bildpyramiden, nach den Prinzipien von Ki und Oh [KO08], vorgenommen werden. Bei diesem Ansatz resultiert eine statische Menge von verschieden großen VALs, die auf alle Caches angewendet werden können. Da die VALs allerdings lose in einer Bildpyramide verteilt liegen, kann es bei der Anwendung zu einer schlechten Textur-Cache-Kohärenz kommen. Dieses Problem lässt sich lösen, indem man eine lineare Zeigerliste für aktive VALs erzeugt. Hierfür liefern Ziegler et al. [ZTT*06] ein GPU-basiertes Verfahren, das auf Histogramm-Pyramiden basiert.

Nutzt man das Light-Clustering-Verfahren aus [KO08], so werden VALs nur auf Basis der Daten der RSM zusammengefasst. Caches werden bei der Zusammenfassung nicht berücksichtigt. Es ist aber durchaus sinnvoll, ein adaptives Clustering für jeden einzelnen Cache durchzuführen. Sind Caches weit von indirekten Lichtquellen entfernt, können die individuellen Lichtquellen zu einer Lichtquelle zusammengefasst und einmalig auf den Cache angewendet werden. Liegt der Cache dagegen sehr nahe, sollten stattdessen die individuellen Lichtquellen genutzt werden. In dieser Arbeit wurde ein experimenteller Ansatz implementiert, der auf den Ideen von Ki und Oh [KO08] und Walter et al. [WFA*05] basiert. Der Ansatz besteht aus zwei Verarbeitungsschritten:

Im ersten Schritt wird für jede RSM eine Mip-Map-Kette (Bildpyramide) erzeugt, bei der jeweils vier VALs einer Mip-Map-Ebene zu einem VAL in der nächsthöheren Ebene zusammengefasst werden. Für diese Zusammenfassung wird zusätzlich der maximale Abstand d_{val} zwischen dem zusammengefassten VAL und den vier individuellen VALs gespeichert. Werden VALs zusammengefasst, die sehr weit auseinander liegen, wird der maximale Abstand d_{val} besonders groß. Lassen sich die VALs dagegen gut zusammenfassen, weil sie nahe beieinander liegen, wird d_{val} klein.

Im zweiten Schritt werden die Proxy-Lichtquellen für die Caches berechnet. Dies geschieht auf Basis der VALs in der Mip-Map-Kette. Der folgende Fragment-Shader übernimmt die Erzeugung der Proxy-Lichtquellen (dieser wird für jeden Cache in den Cache-Texturen ausgeführt):

Listing 6.6: Fragment-Shader zur Erzeugung von Proxy-Lichtquellen auf Basis eines Light-Clustering-Ansatzes.

```
Fragmentshader()
{
  Cache = Lese Cache-Daten aus Cache-Textur.
  VALListe = Füge VAL aus gröbster Mip-Map-Ebene N-1 in Liste ein.

  while( !VALListe.isEmpty() )
    VAL = VALListe.pop_front();
    if( Distance( VAL.Position, Cache.Position) > (c_VAL * VAL.d_VAL) || VAL.Ebene == 0)
      Wende VAL auf Cache an.
    else
      Füge die vier nächstfeineren VALs in VAL-Liste ein.
    end
  end
}
```

Dem Algorithmus ist zu entnehmen, dass anhand der Entfernung des VALs zum Cache entschieden wird, ob dieses zur Anwendung kommt oder nicht. Dabei werden nur VALs angewendet, deren Ab-

stand zum Cache größer als $c_{val}d_{val}$ ist oder die aus der feinsten Ebene der Mip-Map-Kette stammen. Zusammengefasste VALs mit einem großen Wert für d_{val} werden erst genutzt, wenn der Abstand zum Cache entsprechend groß ist. Dies ist sinnvoll, denn diese VALs bilden ihre korrespondierenden „Kinder-VALs“ schlecht ab (da die Kinder-VALs deutliche Diskontinuitäten aufweisen). Der Faktor c_{val} wird benutzerdefiniert gesteuert. Tests haben ergeben, dass ein Faktor $c_{val} < 4$ zu sichtbarem Flackern führt. Ein größerer Faktor hilft, flackernde Artefakte zu vermeiden, jedoch werden hierdurch mehr VALs auf einen Cache angewendet. Das Flackern resultiert aus der Tatsache heraus, dass eine binäre Entscheidung getroffen wird, ob ein VAL angewendet wird oder nicht. Durch eine lineare Interpolation zwischen verschiedenen Mip-Map-Ebenen könnte das Flackern reduziert werden, ohne den Faktor zu vergrößern. Dies wurde jedoch in dieser Arbeit nicht getestet und verbleibt für zukünftige Forschungsansätze. Abbildung 6.7 zeigt, wie sich der Faktor c_{val} auf die Qualität der indirekten Beleuchtung auswirkt.

Da für jeden Cache eine individuelle Menge von VALs zur Anwendung kommen kann, sinkt der Anteil der Daten, die direkt aus dem Textur-Cache der Grafikkarte geladen werden können. Tests für diese Arbeit haben gezeigt, dass die obige Implementierung auf aktuellen Grafikkarten erst dann eine spürbar bessere Laufzeit erzielt, wenn eine RSM eine Auflösung von 128x128 Pixeln überschreitet oder wenn die RSM nur einen sehr kleinen Bereich der sichtbaren Caches abdeckt.

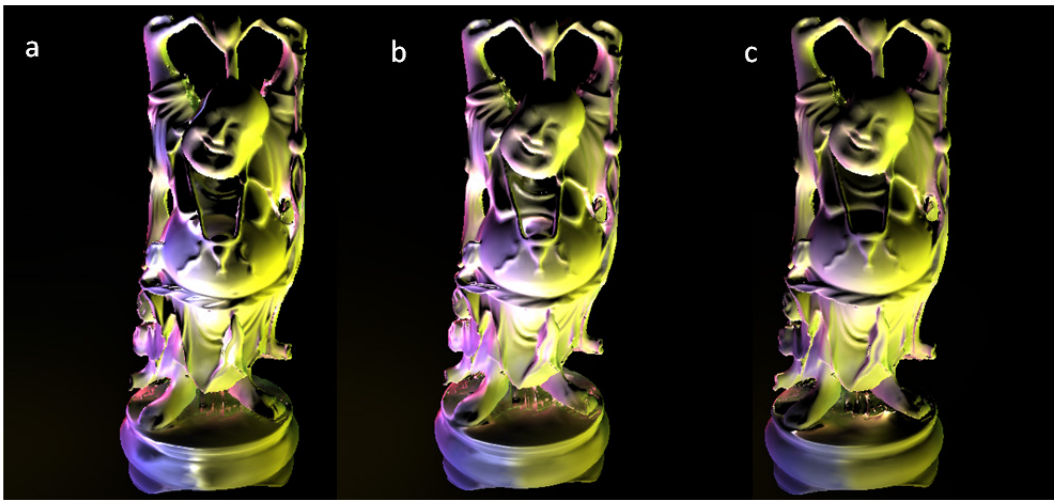

Abbildung 6.7: Indirekte Reflexionen in Verbindung mit Light-Clustering für verschiedene Faktoren c_{val}. (a) zeigt das Modell mit $c_{val} = 1$, (b) mit $c_{val} = 4$ und (c) zeigt die indirekte Reflexion ohne Light-Clustering. Betrachtet man den Sockel des Buddha-Modells, fällt auf, dass dieser für den geringen Faktor (a) deutliche Fehler aufweist, die durch einen höheren Faktor (b) behoben werden. Man kann in der Abbildung außerdem erkennen, dass das Clustering der VALs zu leicht veränderten Reflexionsverläufen führt.

Analog zur Reduzierung der VALs, die auf einen Cache angewendet werden, kann die Anzahl der Caches, die ein Modell zur indirekten Beleuchtung nutzt, adaptiv angepasst werden. Die Anzahl der Caches für ein Modell lässt sich ebenfalls von der Entfernung zu den indirekten Lichtquellen abhängig machen. Allerdings wird hier nicht die Entfernung zu den VALs gemessen, sondern die Entfernung zu den Proxy-Lichtquellen. Das Verfahren funktioniert nach dem folgenden Prinzip:

1. Für ein Modell werden mehrere Cache-Texturen erzeugt, die in einer Mip-Map-Kette gespeichert werden. In der obersten Ebene der Mip-Map-Kette stehen nur 16 Caches, in der

nächstfeineren 64 und so weiter. Jedem Cache einer Ebene können vier Caches der nächstfeineren Ebene zugeordnet werden.

2. Die Erzeugung der Proxy-Lichtquellen geschieht in mehreren Stufen. Eine Stufe entspricht einer Ebene der Mip-Map-Kette. Für jede Ebene existiert ein Z-Buffer, der für alle Caches einen Tiefenwert von null enthält (außer für die gröbste Ebene). Das Verfahren berechnet zuerst die Proxy-Lichtquellen für die 16 Caches aus der gröbsten Mip-Map-Ebene. Für die sich ergebenen 16 Proxy-Lichtquellen wird dann geprüft, wie groß deren Abstand zum jeweiligen Cache ist. Unterschreitet der Abstand einen Schwellenwert, werden die Z-Werte für die korrespondierenden feineren Caches auf eins gesetzt, alle anderen verbleiben auf null. Dann werden für die nächstfeinere Ebene die Proxy-Lichtquellen berechnet. Durch die Z-Maske ist sichergestellt, dass nur für die Caches Proxy-Lichtquellen berechnet werden, deren Tiefenwert größer null ist. Für die neuen Proxy-Lichtquellen wird dann, basierend auf der Entfernung zum Cache, wieder eine Z-Maske erzeugt, die für die nächstfeineren Caches genutzt werden kann und so weiter.
3. Jeder Oberflächenpunkt (Vertex oder Textur) enthält nur die Indizes zu den gröbsten 16 Caches. Für diese wird der Abstand zur Proxy-Lichtquelle ausgelesen. Überschreitet dieser einen Schwellenwert, werden aus einer weiteren Textur die nächsten vier Cache-Indizes ausgelesen. Für diese Caches wird ebenfalls der Abstand zur Proxy-Lichtquelle geprüft und so fort. Schlussendlich erhält man pro Oberflächenpunkt eine variable Menge an Caches, die in das Interpolationsverfahren übernommen werden können.

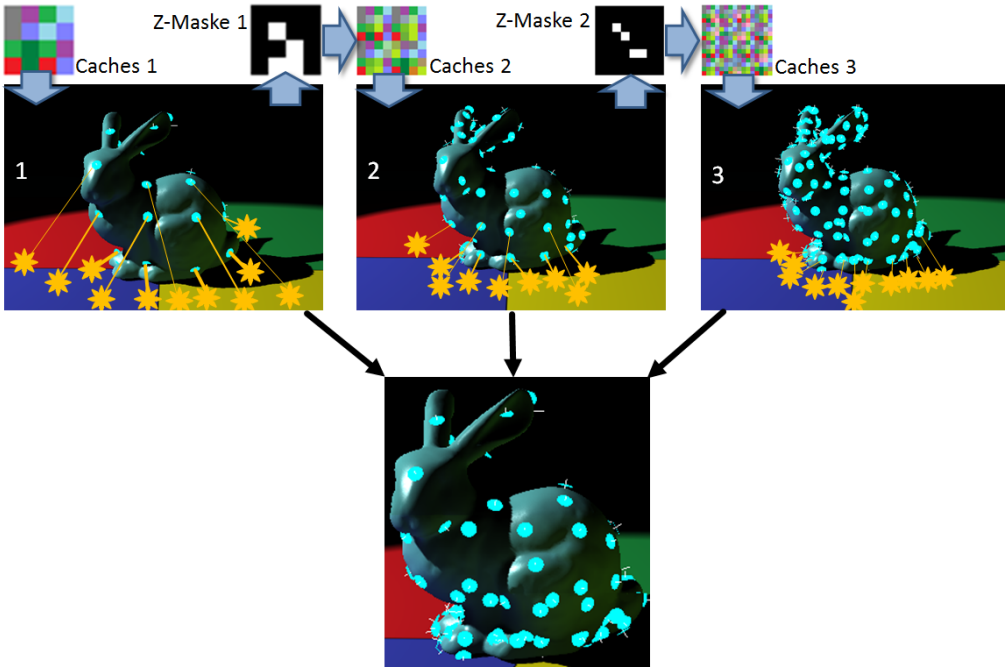

Abbildung 6.8: Beleuchtung für eine adaptive Anzahl von Caches. Im ersten Schritt werden für eine geringe Anzahl an Caches die Proxy-Lichtquellen bestimmt (1). Die Caches, deren Proxy-Lichtquellen sehr nahe zum Cache liegen, werden maskiert (siehe Z-Maske 1). Im zweiten Schritt wird die Maske genutzt, um die nächstfeineren Caches auszuwählen, für die die Proxy-Lichtquellen bestimmt werden sollen (2). Hieraus resultiert wieder eine Z-Maske, die für die feinste Stufe (3) genutzt wird. Schlussendlich wurden nur für die Caches Proxy-Lichtquellen berechnet, die im unteren Bild zu erkennen sind.

Das Prinzip des Ansatzes ist in Abbildung 6.8 dargestellt. In dieser Arbeit wird keine konkrete Implementierung für diesen Ansatz geliefert, es soll lediglich verdeutlicht werden, dass die Prinzipien des

Light-Clusterings auf einfache Art und Weise auf die Caches übertragen werden können, wenn man den Z-Buffer einsetzt (wie dies bereits beim Cache-Culling im vorherigen Kapitel getan wurde). Darüber hinaus sollte gezeigt werden, dass es über indirekte Texturzugriffe möglich ist, einem Oberflächenpunkt eine dynamische Anzahl von Caches zuzuordnen.

6.1.7 Mehrfachreflexionen

Bis jetzt wurde in dieser Arbeit nur erläutert, wie jeweils die erste indirekte Reflexion mit Hilfe des LightSkin-Verfahrens berechnet werden kann. Dies geschah aus Laufzeitüberlegungen heraus, da das LightSkin-Verfahren, sowie die meisten anderen globalen Echtzeit-Beleuchtungsverfahren, nur die erste indirekte Reflexion mit einer ansprechenden Bildwiederholrate (>30fps) darstellen kann. Dennoch soll nicht verschwiegen werden, dass weitere indirekte Reflexionen einfach abgebildet werden können, indem Caches zu VALs werden.

Für die Berechnung der ersten indirekten Reflexion werden Caches durch VALs aus der RSM beleuchtet. Für die Berechnung der zweiten indirekten Reflexion werden dann die Caches durch die Caches beleuchtet, die indirektes Licht im ersten Durchgang empfangen haben. Praktisch lässt sich dies einfach realisieren, da die Positionen, Normalen und Flächen der Caches bekannt sind und nur noch der ausgesendete Lichtstrom für diese berechnet werden muss. Durch die Anwendung der Proxy-Lichtquelle auf den Cache kann der Lichtstrom berechnet werden. Da die Caches selbst in Texturen organisiert sind, ist es möglich, diese direkt als RSM zu interpretieren und für das indirekte Beleuchtungsverfahren ohne große Veränderungen wieder zu verwenden (als indirekte Lichtquellen). Für Mehrfachreflexionen wird also zuerst die erste indirekte Reflexion berechnet, dann werden die beleuchteten Caches in eine RSM kopiert, die dann wiederum für die zweite indirekte Reflexion genutzt wird und so weiter. Allerdings muss hierbei beachtet werden, dass die Caches nicht dicht, sondern lose in der Szene verteilt liegen, so dass hochfrequente Reflexionen nicht abgebildet werden können. Es können also nur diffuse Mehrfachreflexionen simuliert werden. Darüber hinaus existiert eine Einschränkung, die das Subsurface-Scattering betrifft: Es ist zwar möglich, dass die Caches eines SSS-Materials indirektes Licht zu anderen Objekten senden, aber es ist nicht direkt möglich, dass indirektes Licht von anderen Objekten zu den Caches des SSS-Objekts gelangt. Dies ergibt sich aus dem Integrationsmodell, welches für das SSS-Verfahren gewählt wurde. Bei diesem werden nur VALs berücksichtigt, die auf dem SSS-Objekt liegen. Da jedoch bei der zweiten indirekten Reflexion Caches anderer Objekte das Licht aussenden, werden diese nicht berücksichtigt. Das Problem lässt sich nur lösen, wenn eine zusätzliche Reflexionsberechnung für SSS-Materialien durchgeführt wird. Bei dieser wird das ausgesendete Licht anderer Caches auf die SSS-Caches übernommen und anschließend in das Material hineingestreut (zusätzliche Streuung). SSS-Materialien verursachen also für Mehrfachreflexionen entweder einen größeren Aufwand oder sie werden nicht berücksichtigt.

6.2 Evaluation

In diesem Kapitel werden die Ergebnisse des LightSkin-Verfahrens vorgestellt und diskutiert. Die Diskussion bezieht sich sowohl auf die Darstellungsqualität als auch auf das Laufzeitverhalten des vorgestellten Ansatzes. In Kapitel 4.2 zur Konzeption des Verfahrens wurde bereits darauf hingewiesen, dass die Light-Propagation-Volumes von Kaplanyan und Dachsbacher [KD10] und das Voxel-Cone-Tracing von Crassin et al. [CNS*11] als vergleichbare Ansätze zum LightSkin-Verfahren verstanden werden können. Deshalb ist in diesem Unterkapitel auch ein Vergleich des LightSkin-Verfahrens zu

diesen Ansätzen zu finden. Schlussendlich erfolgt eine Kategorisierung des Verfahrens nach dem Kategoriensystem von Ritschel et al. [RDG*12].

6.2.1 Qualität

Die Qualität der globalen Beleuchtungseffekte, die durch das LightSkin-Verfahren erreicht werden kann, hängt von der Anzahl der Caches ab, die für die Simulation genutzt werden. In dieser Arbeit wurde bereits mehrmals erwähnt, dass wenige Caches ausreichen, um plausible Ergebnisse zu erzeugen. In Abbildung 6.9 wird deshalb gezeigt, wie sich die indirekten Reflexionen für verschiedene Modelle mit steigender Anzahl der Caches verändern (ohne weiche Schatten zu erzeugen). Man kann deutlich erkennen, dass die indirekten Reflexionen bereits mit sehr wenigen Caches überzeugend interpoliert werden können. Dies gilt insbesondere auch für komplexe, filigrane Objekte, wie das Dinosaurierskelett, das in Abbildung 6.9 exemplarisch gezeigt wird. In Abbildung 6.10 und Abbildung 6.11 wird ferner gezeigt, wie sich die indirekten Reflexionen verändern, wenn der Algorithmus zur Erzeugung der weichen Schattierungen eingeschaltet wird. Für die meisten Objekte erscheinen so die Reflexionen realistischer, allerdings erzeugen sehr filigrane Objekte einige falsche Schattierungen (zu erkennen in Abbildung 6.11). Diese falschen Schattierungen resultieren aus der approximativen Darstellung des Modells durch Kreisflächen. Filigrane, längliche Objekte werden durch diese Kreisflächen unzureichend approximiert. Die meisten Modelle profitieren jedoch von der Erzeugung der weichen Schatten. Die Darstellung des Multiple Subsurface-Scatterings ist ebenfalls abhängig von der Anzahl der Caches, allerdings zeigen Abbildung 6.12 und Abbildung 6.13, dass für die meisten Modelle bereits fünfzig Caches ausreichen, um optisch plausible SSS-Phänomene darzustellen. Dies hängt mit den stark streuenden Eigenschaften der Materialien, sowie mit der hier vorgestellten Interpolationsmethode zusammen.

Komplette Szenen können ebenfalls mit sehr wenigen Caches überzeugend dargestellt werden. In Abbildung 6.14 und Abbildung 6.15 wird gezeigt, welcher optische Gewinn durch den Einsatz des LightSkin-Verfahrens erzielt werden kann. Dabei enthält keine Szene mehr als 3000 Caches. Um die physikalische Korrektheit des Verfahrens beurteilen zu können, werden in Abbildung 6.16 einige Ergebnisse des Verfahrens den Ergebnissen eines Offline-Renderings gegenübergestellt. Die Ähnlichkeit der Bilder lässt den Schluss zu, dass die physikalischen Eigenschaften der Lichtausbreitung durch das Verfahren gut approximiert werden. Allerdings muss man diese Aussage für glänzende Reflexionen einschränken, da diese einen systematischen Fehler enthalten, der in Abbildung 6.17 dargestellt ist. Typischerweise kann man an den Rändern eines glänzenden Objekts hochfrequente spiegelnde Abbildungen erkennen. Diese Spiegelungen zum Rand können mit dem LightSkin-Verfahren nicht hinreichend dargestellt werden, stattdessen ist ein leichter Energieabfall zu beobachten. Dieser resultiert aus der gewählten Approximation der glänzenden Reflexionen durch die Formfaktoren. Die Abbildung zeigt auch den maximalen Grad der Spiegelung an, der noch mit dem LightSkin-Verfahren effizient berechnet werden kann. Stärker spiegelnde Objekte würden zu viele Caches voraussetzen.

Weitere vergleichende Abbildungen befinden sich in einer höheren Auflösung in Anhang B.

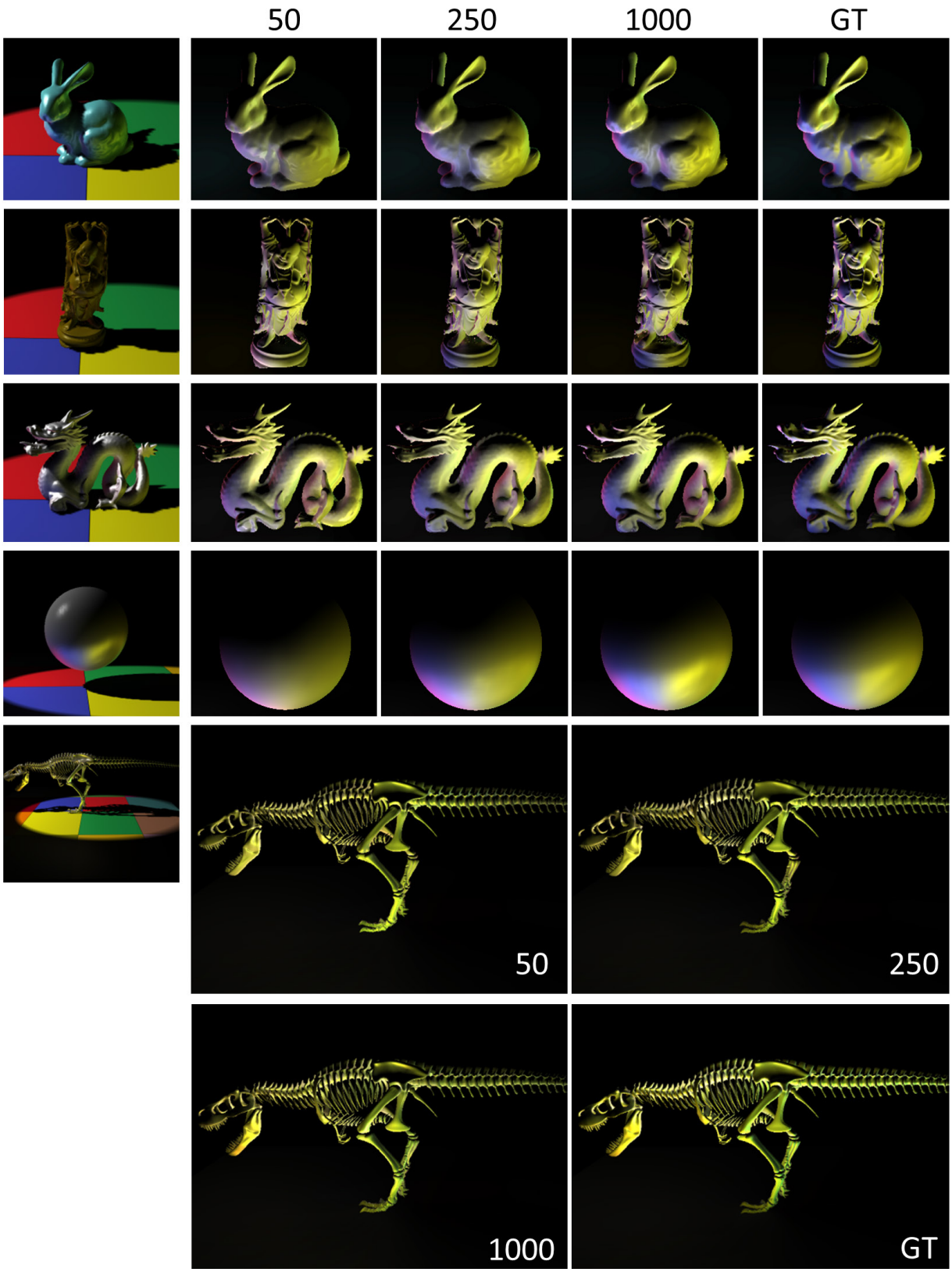

Abbildung 6.9: Indirekte Reflexionen für unterschiedliche Mengen von Caches. Jede Spalte zeigt jeweils die indirekten Reflexionen basierend auf 50, 250 und 1000 Caches. Die letzte Spalte zeigt die Ground-Truth-Reflexionen (GT), die ohne Interpolation erzeugt wurden.

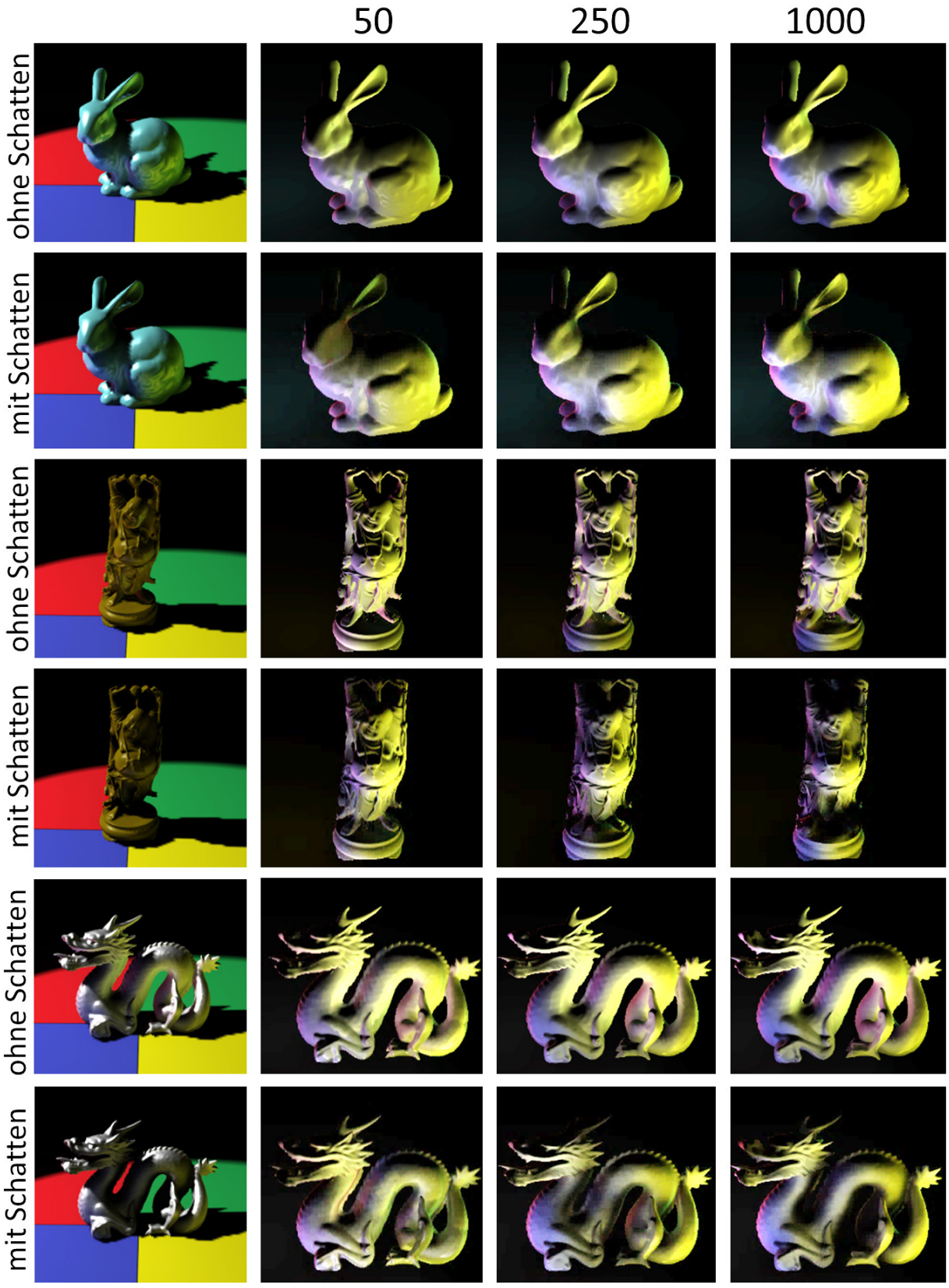

Abbildung 6.10: Indirekte Beleuchtung für verschiedene Mengen von Caches mit weichen Schatten. Die erste, dritte und fünfte Zeile zeigen die indirekten Reflexionen ohne und die zweite, vierte und sechste mit weichen Schatten.

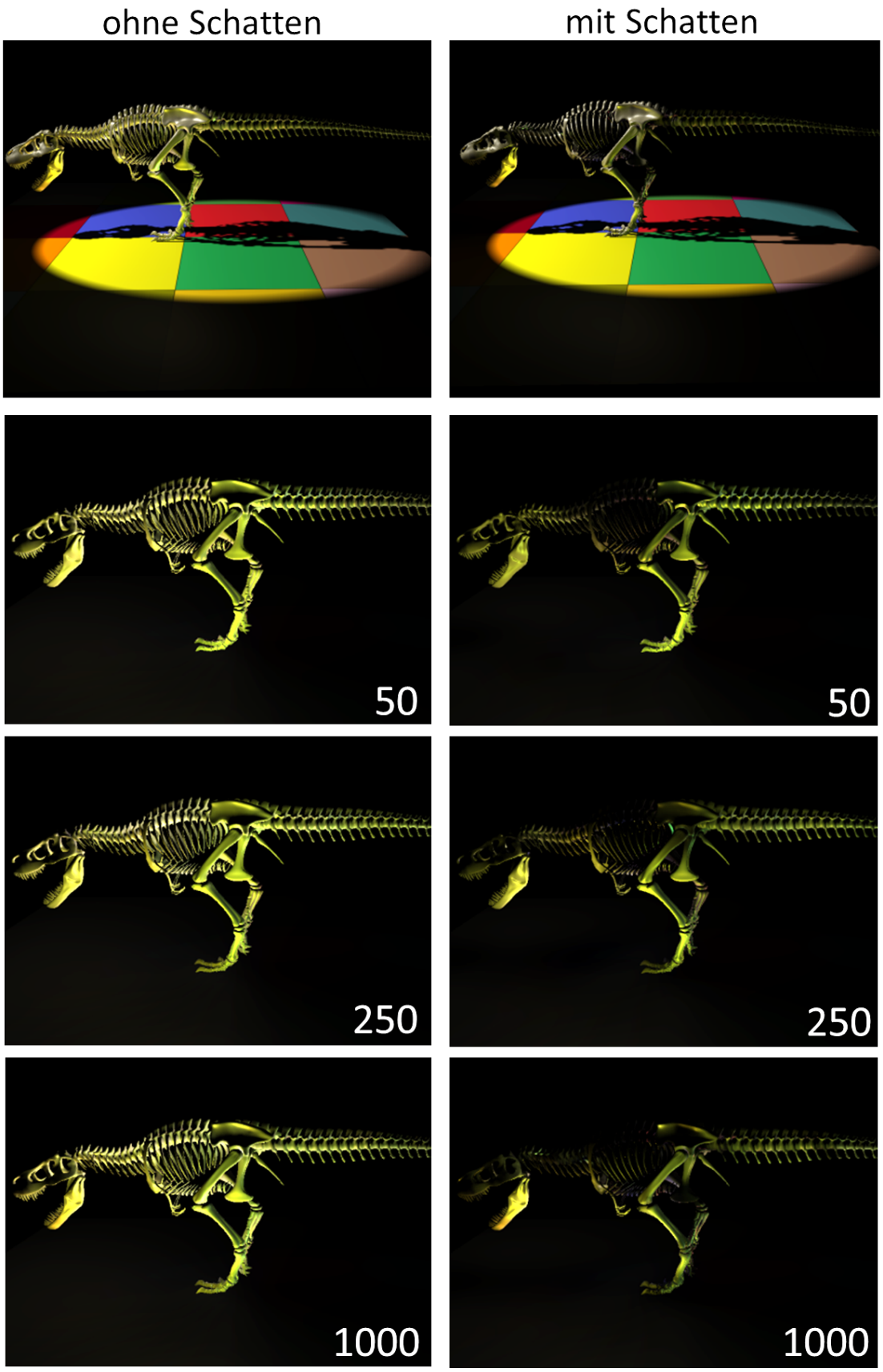

Abbildung 6.11: Indirekte Reflexionen für verschiedene Cache-Mengen ohne und mit weichen Schatten für ein filigranes Objekt. Rechts wird das Modell jeweils ohne weiche Schatten, links dagegen mit weichen Schatten gezeigt. Man erkennt, dass das Modell nicht gut durch Kreisflächen approximiert werden kann.

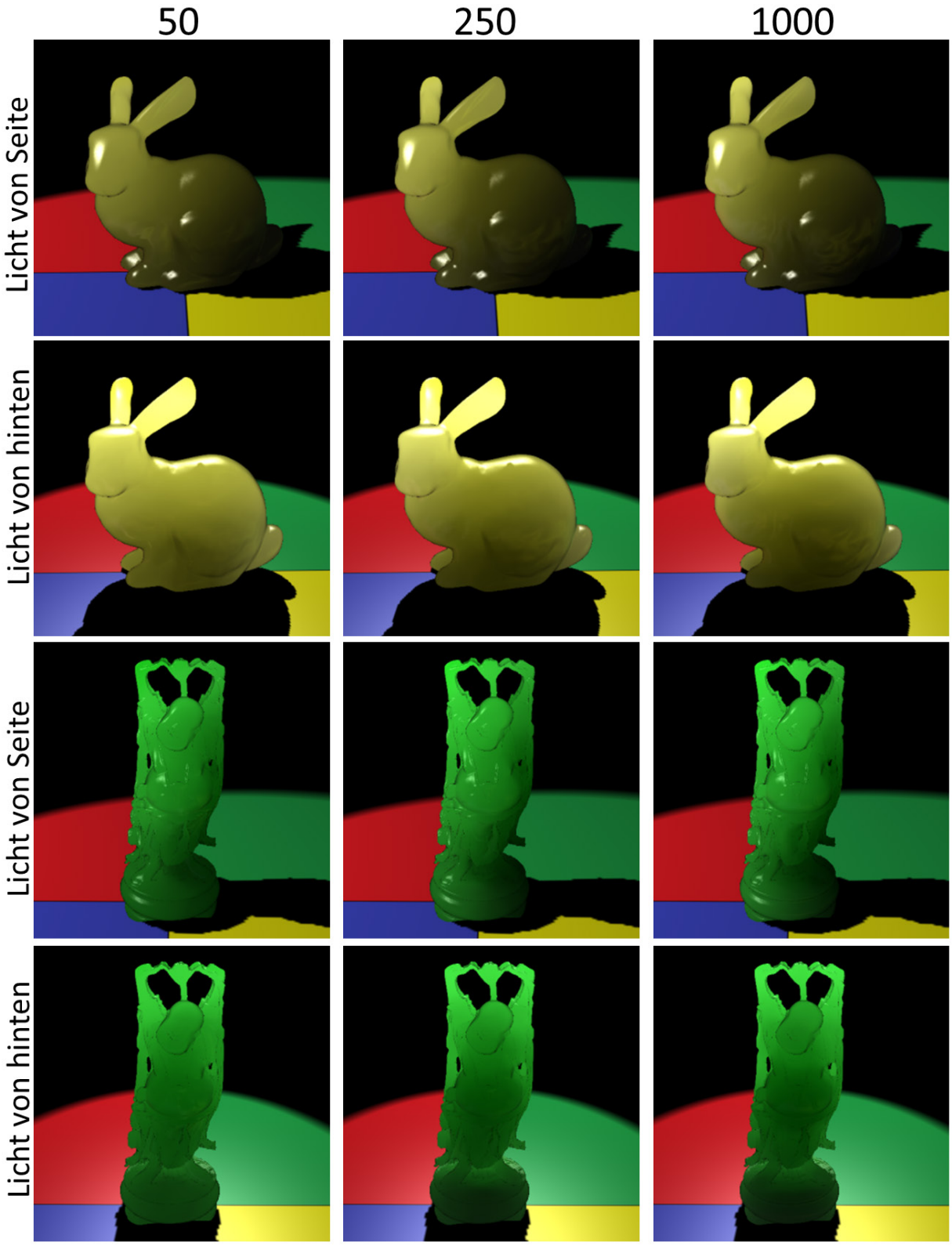

Abbildung 6.12: Multiple Subsurface-Scattering mit verschiedenen Cache-Mengen (Spalten). Die Modelle wurden jeweils von der Seite und von hinten beleuchtet.

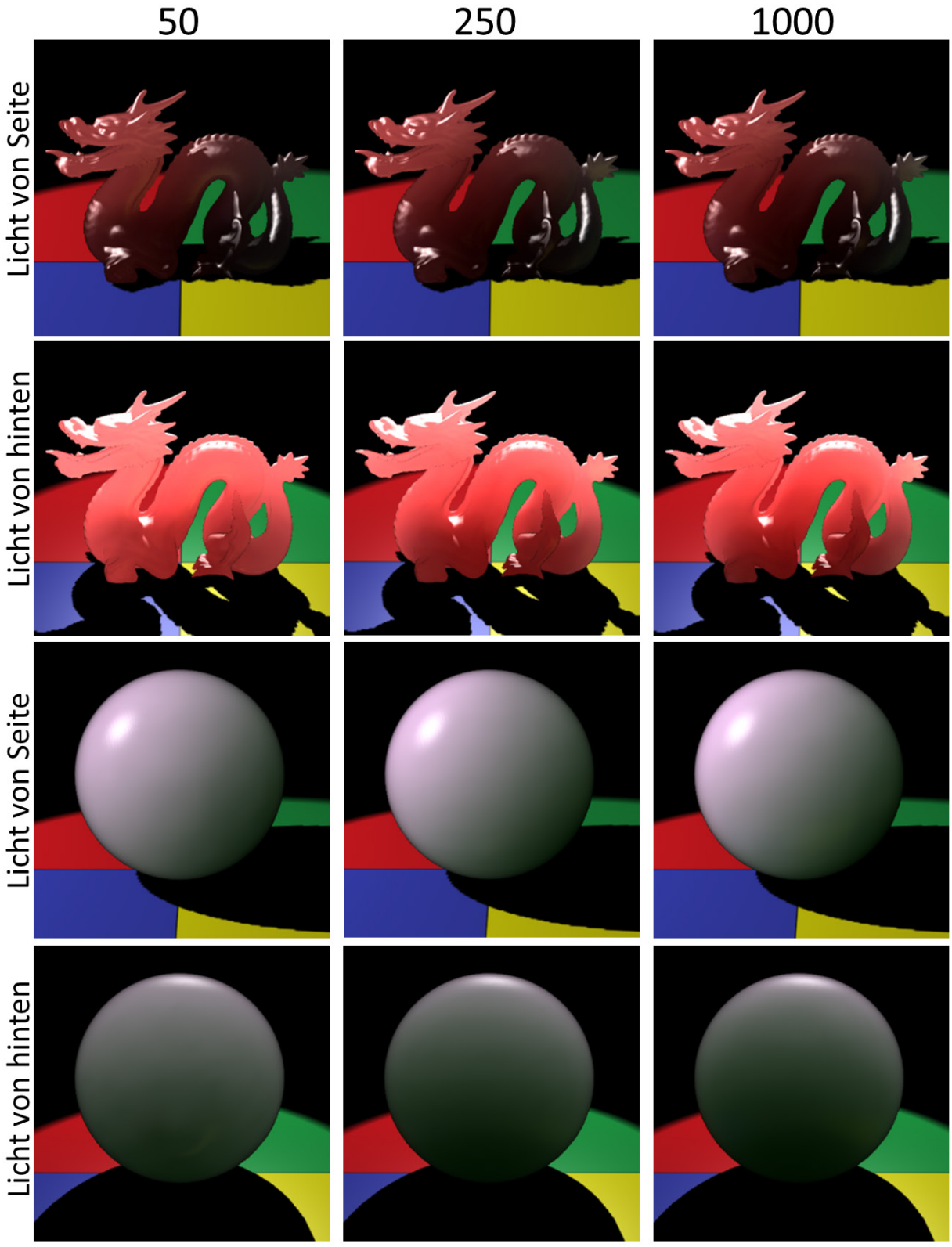

Abbildung 6.13: Multiple Subsurface-Scattering mit verschiedenen Cache-Mengen (Spalten). Die Modelle wurden jeweils von der Seite und von hinten beleuchtet.

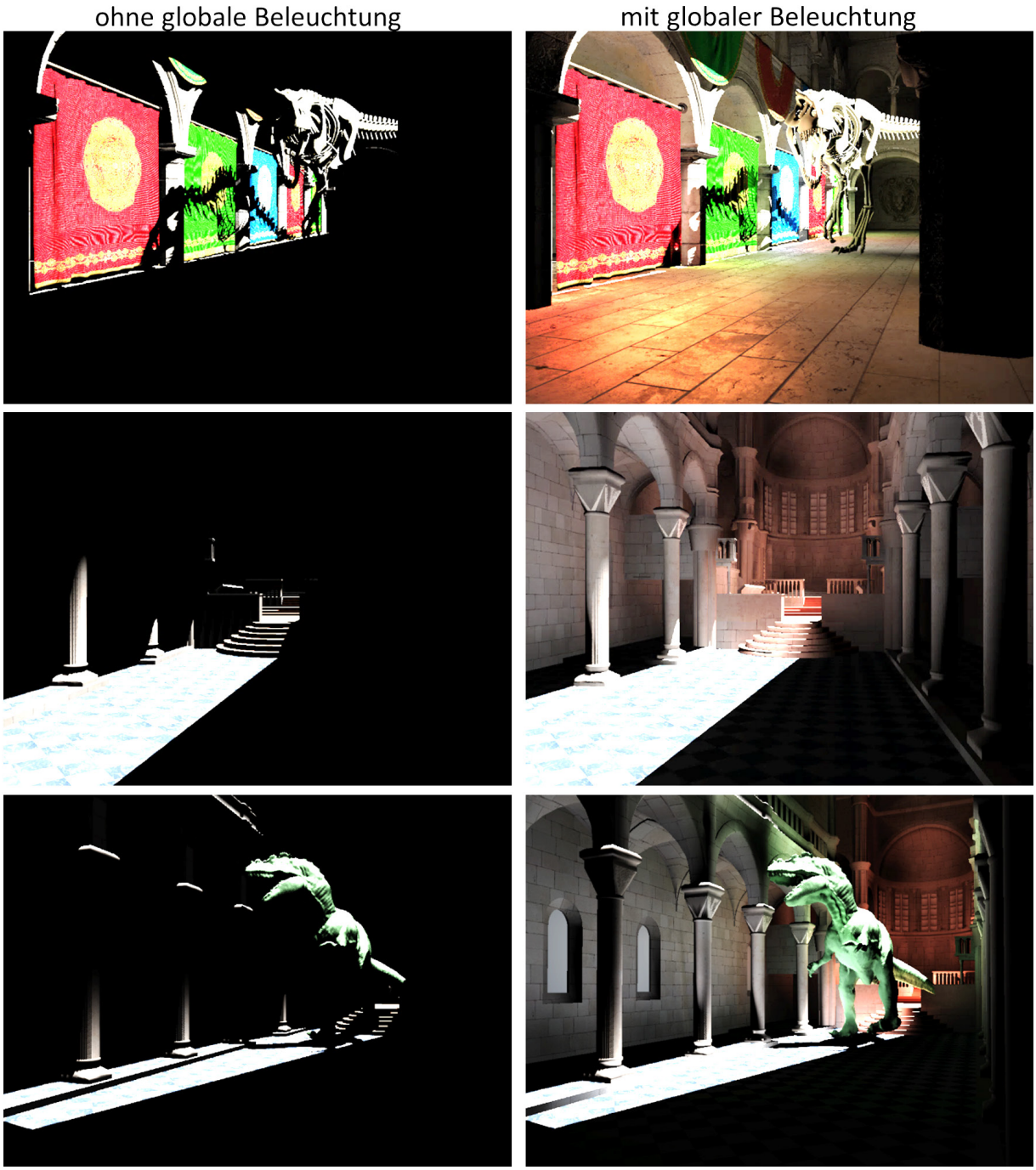

Abbildung 6.14: Vergleich zwischen Szenen ohne globale Beleuchtung (links) und mit globaler Beleuchtung, berechnet durch das LightSkin-Verfahren (rechts). Die Szenen nutzen nicht mehr als 3000 Caches und können mit ca. 100 FPS dargestellt werden (mit einer NVidia GTX 780 Grafikkarte).

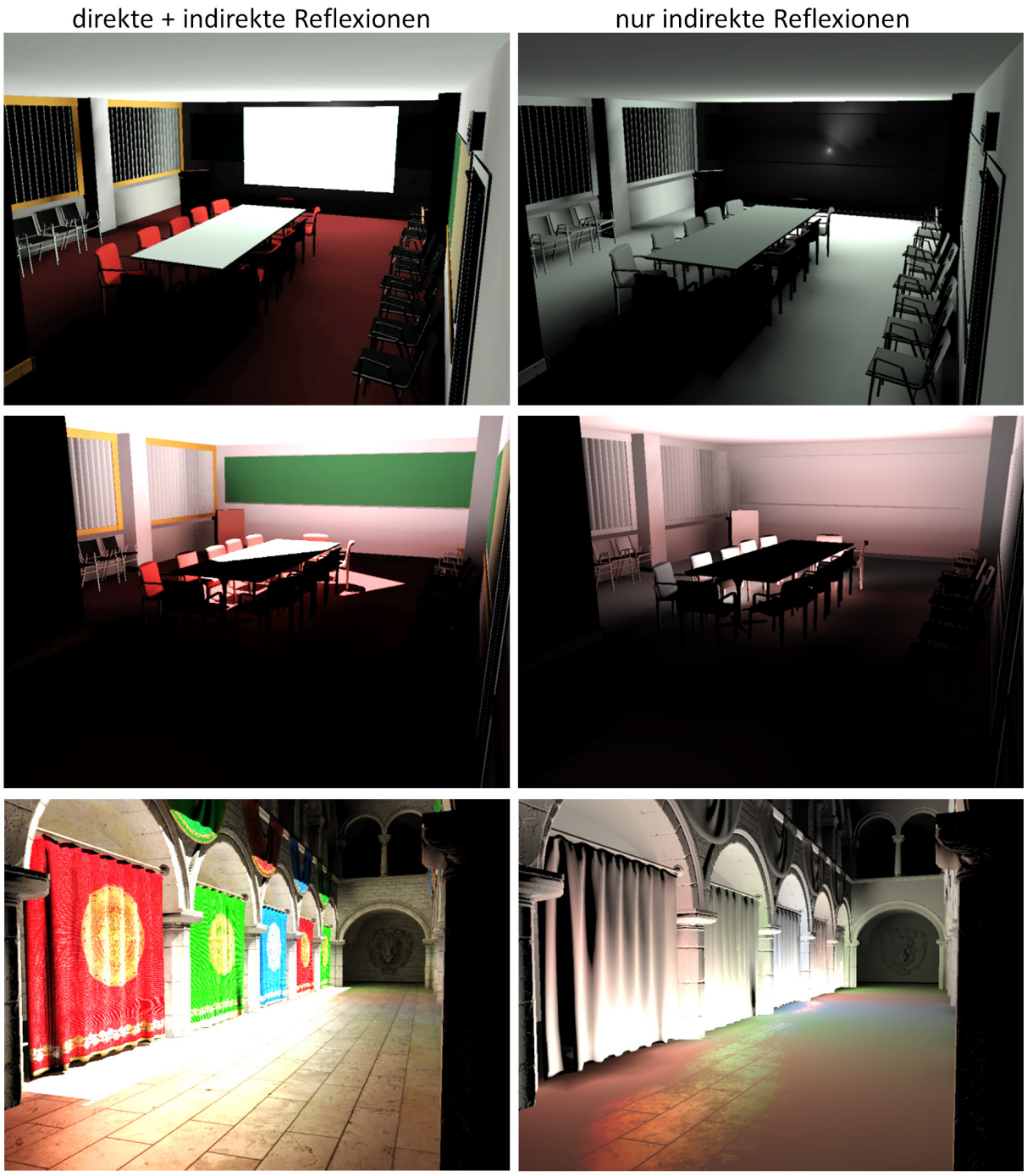

Abbildung 6.15: Weitere Szenen mit globaler Beleuchtung, berechnet durch das LightSkin-Verfahren (links). Rechts wird jeweils nur die indirekte Beleuchtung dargestellt.

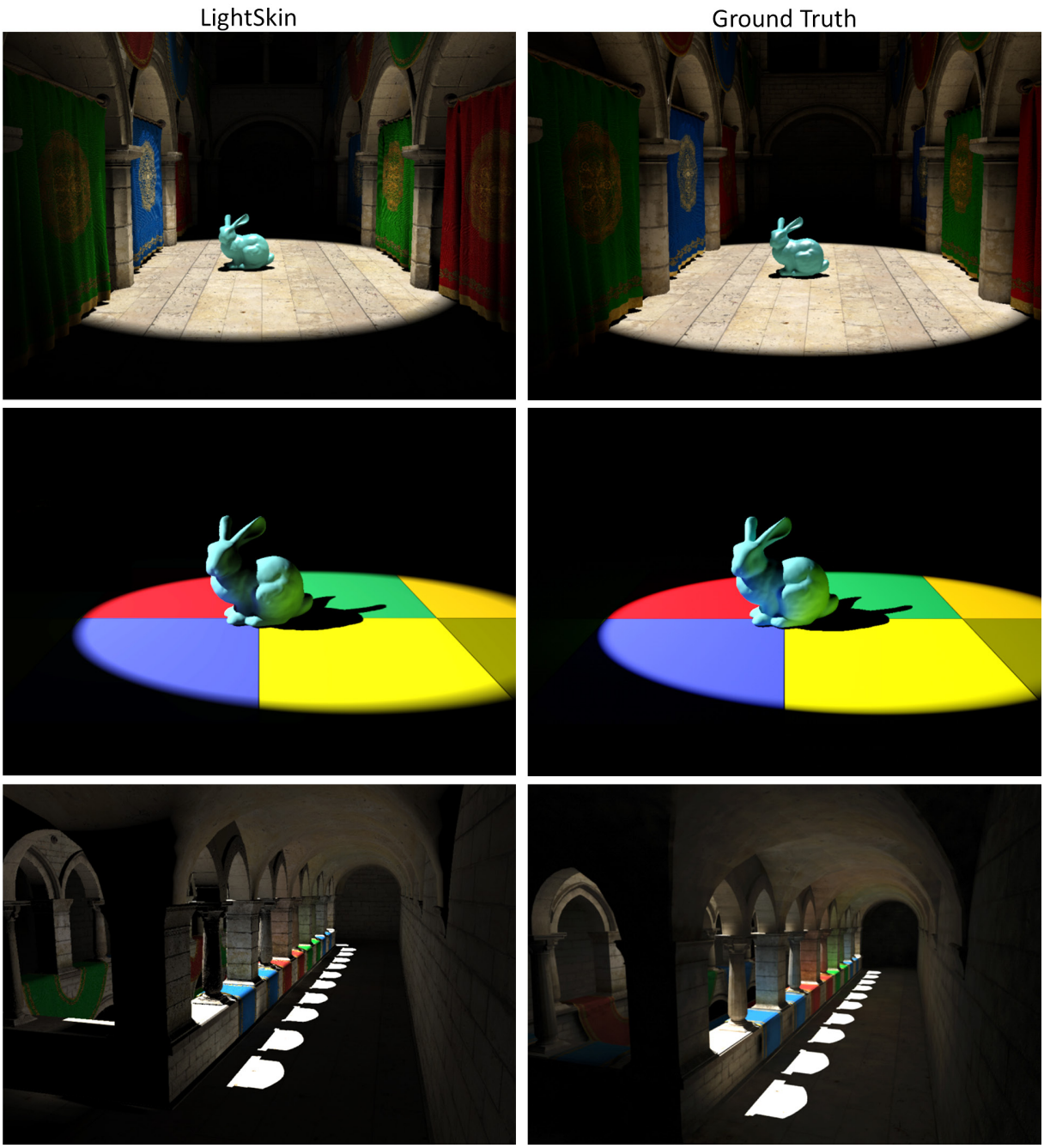

Abbildung 6.16: Vergleich zwischen der globalen Beleuchtung des LightSkin-Verfahrens (links) und Ground-Truth-Lösungen eines Offline-Renderers (rechts). Als Offline-Renderer wurde Cinema 4D genutzt. Die Bilder links benötigten ca. neun ms zur Berechnung, während die Bilder rechts ca. vier Minuten benötigten. Das Bild unten rechts wurde aus [CNS*11] entnommen.

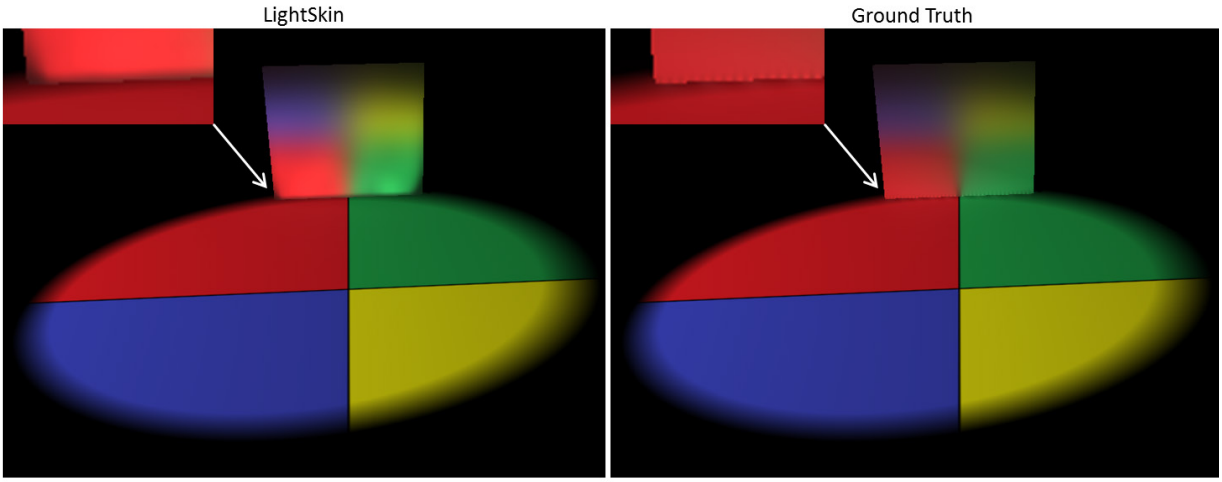

Abbildung 6.17: Vergleich von glänzenden Reflexionen zwischen dem LightSkin-Verfahren (links) und einer Ground-Truth-Berechnung ohne Interpolation (rechts). An der unteren Kante des Objekts kann man beim LightSkin-Verfahren einen falschen Energieabfall erkennen.

6.2.2 Performance

Um das Laufzeitverhalten des LightSkin-Verfahrens beurteilen zu können, soll dieses in verschiedenen Kontexten gemessen werden:

1. Kontext: Abhängigkeit der Laufzeit von der Anzahl der Dreiecke einer Szene.
2. Kontext: Abhängigkeit der Laufzeit von der Auflösung des Framebuffers.
3. Kontext: Abhängigkeit der Laufzeit von der Anzahl der Caches.
4. Kontext: Abhängigkeit der Laufzeit von der Anzahl der VALs.

Alle Messungen in diesem Kapitel wurden auf einem PC-System mit einer Intel Core 2 Quad CPU Q9650 (3 GHz), 8 GB RAM und einer NVidia Geforce GTX 780 Grafikkarte mit Kepler GK 110 Architektur vorgenommen. Im Anhang A befinden sich die gleichen Messungen mit einem alternativen System, das eine AMD HD 7970 Grafikkarte mit Tahiti-Chipsatz nutzt.

Im ersten Kontext wird betrachtet, wie sich die Laufzeit verändert, wenn die Anzahl der Dreiecke einer Szene erhöht wird. Dieses Szenario gibt Aufschluss darüber, wie schnell ein Ergebnis für jeden Oberflächenpunkt anhand der Caches interpoliert werden kann. Komplexe Szenen besitzen in der Regel sehr viele Dreiecke, so dass die Interpolation nicht zu langsam sein darf, damit diese Szenen noch in Echtzeit dargestellt werden können. Die Erzeugung der Proxy-Lichtquellen spielt in diesem Kontext eine untergeordnete Rolle, weswegen nur 500 Caches für diese Messung genutzt wurden. Das Interpolationsverfahren wurde in vier verschiedenen Versionen gemessen:

- Interpolation von nur diffusen Reflexionen mit acht (D8) und sechzehn (D16) Caches pro Oberflächenpunkt.
- Glänzende und diffuse Reflexionen mit acht (DG8) und sechzehn (DG16) Caches pro Oberflächenpunkt.

Die Messergebnisse sind in Tabelle 6.1 und in Abbildung 6.18 aufgeführt. Zum Vergleich wurde ebenfalls gemessen, wie schnell die Dreiecke ohne globale Beleuchtungseffekte gezeichnet werden können (LI, nur direktes Licht mit einem Deferred-Shading-Ansatz). Die Ergebnisse zeigen, dass die Darstellungsrate pro Dreieck um einen konstanten Faktor von zwei bzw. maximal vier verringert wird (im Vergleich zur direkten Beleuchtung).

Im zweiten Kontext wurde gemessen, inwieweit sich die Darstellungsrate durch die Erhöhung der Auflösung verändert. Die Tabelle 6.2 zeigt, dass die Erhöhung der Auflösung des Framebuffers für das Verfahren relativ unkritisch ist. Selbst für Auflösungen, die jenseits der Full-HD-Auflösung liegen, können ausreichend schnell Bilder generiert werden. Somit sind hochauflösende Renderings in CAVE-Umgebungen oder die Darstellung von Side-by-Side-Stereobildern nicht ausgeschlossen. Für die Messung wurden 1.000 Caches genutzt.

Die letzten beiden Kontexte wurden in Tabelle 6.3 und in Abbildung 6.19 bis Abbildung 6.22 zusammengefasst. Die Messungen zeigen, wie sich die Bildwiederholrate zur Anzahl der Caches und der VALs verhält. Diese wurde für die verschiedenen Phänomene gemessen:

- nur diffuse Reflexionen (D),
- diffuse und glänzende Reflexionen (DG),
- diffuse und glänzende Reflexionen mit weichen Schatten (DGS),

- nur Subsurface-Scattering (SSS).

Die Darstellungsraten wurden in einer Szene mit 150.000 Dreiecken gemessen. Der in Abbildung 6.19 und Abbildung 6.20 deutlich zu erkennende Einbruch der Bildwiederholungsraten bei 5.000 Caches für diffuse und glänzende Reflexionen mit weichen Schatten (DGS) resultiert aus einer nicht optimalen Implementierung der weichen Schatten: Die Caches werden in quadratischen Cache-Texturen gespeichert. Wenn mehr als 4.096 Caches genutzt werden, erhöht sich die Auflösung der Cache-Textur von 64x64 auf 128x128 Pixel. Bei der Berechnung der weichen Schatten wird nun jeder Pixel der Cache-Textur als potentiell verdeckender Cache angenommen und für die Prüfung genutzt (siehe Kapitel 5.4.2). Demnach wird für 5.000 Caches geprüft, ob diese durch ca. 16.000 Caches verdeckt werden. Eine Optimierung der Implementierung ist einfach möglich, indem Caches auch in nicht-quadratischen Texturen gespeichert werden können. Dies führt zu einer geringfügig komplexeren Erzeugung des Texturatlas (für quadratische Texturen kann für die Erzeugung eine abgewandelte Form des Buddy-Systems für Speicherallokationen genutzt werden, das von Knowlton [Kno65] vorgestellt wurde).

Setzt man voraus, dass 30 FPS eine akzeptable Bildwiederholungsrate für eine Echtzeit-Simulation ist, zeigen die Messungen, dass ca. 16.000 Caches durch ca. 16.000 VALs in Echtzeit beleuchtet werden können. Nutzt man 64.000 VALs, können noch 7.000 Caches und bei 256.000 VALs noch ca. 2.000 Caches in Echtzeit beleuchtet werden. Um diese Zahlen besser einschätzen zu können, sei erwähnt, dass die Szenen in Abbildung 6.14 und Abbildung 6.15 mit 2.000-3.000 Caches approximiert wurden und dass nur für sichtbare Caches die Proxy-Lichtquellen berechnet werden müssen. In den Testszenarien dieser Arbeit variierte die Anzahl der Caches nach der Durchführung des Cache-Cullings (siehe Kapitel 6.1.5) zwischen 400 und 1.800 Caches.

Tabelle 6.1: Abhängigkeit der Darstellungsrate in FPS von der Anzahl der Dreiecke. D8: nur diffuse Interpolation mit 8 Caches pro Oberflächenpunkt; D16: nur diffuse Interpolation mit 16 Caches; DG8: diffuse und glänzende Interpolation mit 8 Caches; DG16: diffuse und glänzende Interpolation mit 16 Caches; LI: ohne globale Beleuchtung. Es wurden jeweils 500 Caches genutzt.

Dreiecke	D8	D16	DG8	DG16	LI	LI/2	LI/4
800	772,6	772,6	776,7	774,5	1554,4	777,2	388,6
3.200	771,3	771,1	776,7	774,3	1536,7	768,4	384,2
12.800	771,6	771	777	772,9	1486,6	743,3	371,7
51.200	771,1	771,1	776,5	775,6	1311,9	656,0	328,0
102.400	760	724,3	702,6	582,3	1146	573,0	286,5
204.800	529,4	491,5	474,7	372	928,8	464,4	232,2
409.600	377,41	312,7	322,8	221,9	679,2	339,6	169,8
819.200	231,7	180,7	199,1	123,7	444,3	222,2	111,1
1.638.400	126,3	96,6	107,2	64,5	263,6	131,8	65,9
2.457.600	86,3	65,5	74,37	42,9	174,5	87,3	43,6
3.276.800	65,2	49,64	54,5	32,3	134,2	67,1	33,6
4.096.000	52,9	39,6	43,8	25,8	105,5	52,8	26,4
4.915.200	43,6	33,5	36,4	21,7	91,9	46,0	23,0
5.734.400	36,7	28,5	33,6	18,5	76,1	38,1	19,0
6.553.600	32,3	25	29,1	16,2	68,2	34,1	17,1

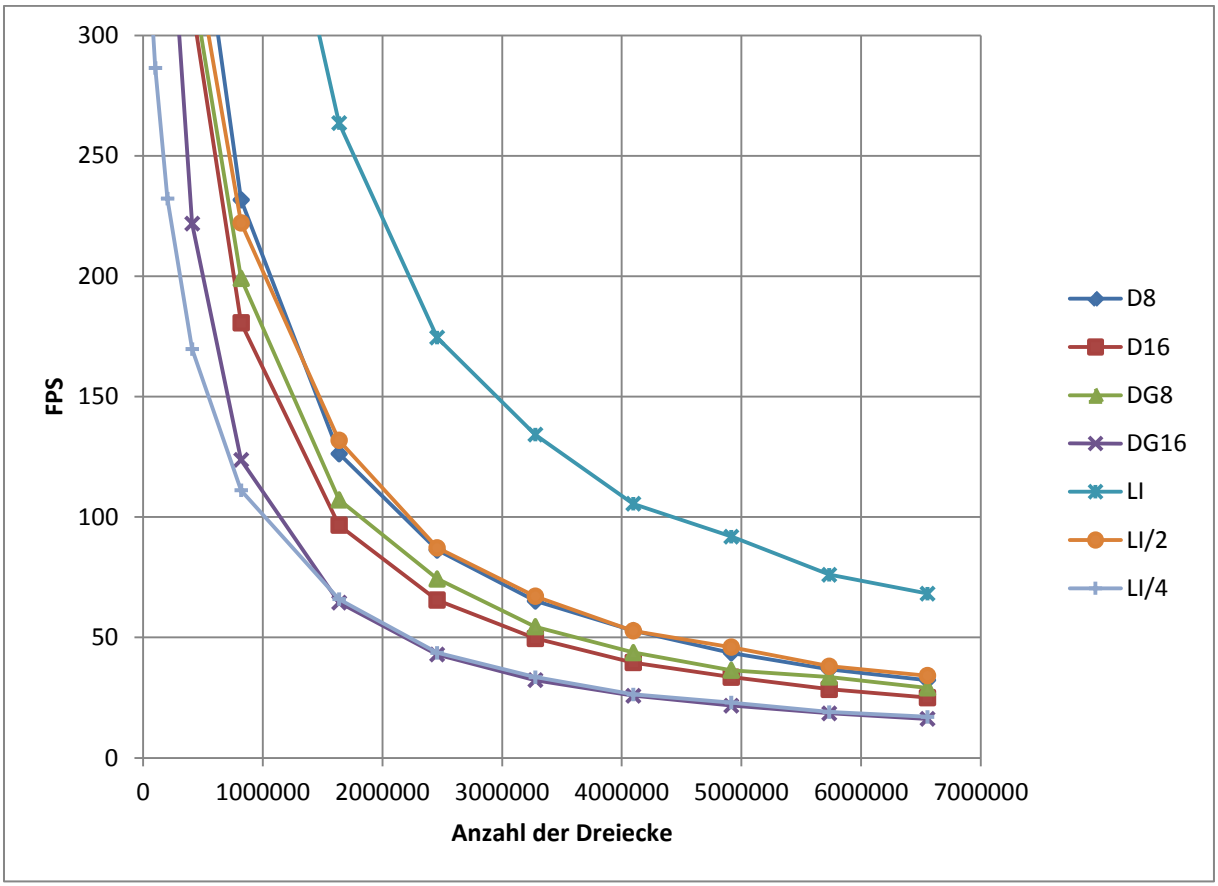

Abbildung 6.18: Darstellung der Abhängigkeit der Bildwiederholungsrate in FPS von der Anzahl der Dreiecke (siehe auch Tabelle 6.1).

Tabelle 6.2: Abhängigkeit der Bildwiederholungsrate von der Auflösung des Framebuffers. Zum Vergleich wurden ebenfalls die Raten ohne globale Beleuchtung in der Spalte LI festgehalten. Man erkennt, dass selbst 4K-Auflösungen noch mit ca. 60 FPS dargestellt werden können. Für die Berechnungen wurden 1.000 Caches genutzt.

Auflösung	LI	FPS
256*256	2549	616,5
320*240	2547	612,1
512*512	2523	548
640*480	2600	537,8
800*600	2137	492,4
1024*768	1355,6	424,5
1280*720	1200	414,5
1024*1024	1022,4	369,3
1920*1080	588,7	291,2
2048*2048	273,8	169
4096*4096	78,3	57,3

Tabelle 6.3: Bildwiederholungsraten in Abhängigkeit von der Anzahl der Caches und von der Anzahl der VALs. D: nur diffuse Reflexionen, DG: diffuse und glänzende Reflexionen, DGS: diffuse und glänzende Reflexionen mit weichen Schatten, SSS: Subsurface-Scattering. Gemessen in einer Szene mit 150.000 Dreiecken.

	64x64 VALs (ca. 4.000)				128x128 VALs (ca. 16.000)				256x256 VALs (ca. 64.000)				512x512 VALs (ca. 256.000)			
Caches	**D**	**DG**	**DGS**	**SSS**	**D**	**DG**	**DGS**	**SSS**	**D**	**DG**	**DGS**	**SSS**	**D**	**DG**	**DGS**	**SSS**
250	1014	937,6	709,8	711,3	761	563,7	445,5	336,4	276	265,6	242,3	113,7	102,6	80,4	77,7	31,3
500	1013	936	712	713,6	760	403,9	326,8	340,5	276,3	266,8	242,8	115,5	102,8	80	76,4	31,6
1.000	920	810,1	550,7	532,4	580,4	405	327,5	208,8	234	157,3	143,4	62,5	69,2	43,7	42,6	16,3
2.000	839,7	711,3	440,1	433,5	471,1	334,63	259,46	154,3	169,5	105,4	96,2	43	47,5	29,9	29,2	11
3.000	770,4	636,5	370	360,8	392,7	274,4	209,8	115,5	132,4	80,3	72,4	31,2	36,1	20,8	20,4	8,2
4.000	719,1	559,9	315,1	317,6	327,6	225,8	179,4	96	109,1	64,4	61,7	25,7	29,4	16,2	15,8	6,4
5.000	662,9	508,1	112,2	269,1	298,7	199,4	83,5	78,3	92,3	58,5	41,6	20,5	24,3	15,2	13,7	5,2
6.000	623,2	433,8	103,7	232,6	268,6	177,8	77,2	67,4	80	46,8	34,8	17,5	21	11,9	11,1	4,5
7.000	581,7	400,6	85,6	205,1	240,1	155,7	63,8	58,7	69,4	41,3	29,4	15,5	17	10,5	9,5	3,8
8.000	540	365,9	78,4	183,7	201	136,9	66,9	53,3	60,5	35,4	25,5	13,2	15,7	9	8,1	3,3
9.000	520,8	374,7	71,2	172,8	198,9	122,1	50,7	49	56,1	33,4	23,4	12,4	13,5	8,4	7,6	3,1
10.000	494,3	328,7	68	157,5	168,7	110,1	47,7	44,1	48,3	30,1	21,8	11,1	13	7,5	7	2,8
11.000	471,1	337,5	60,5	146,8	171,1	103,2	43,8	40,6	44	27,3	19,9	10,3	11,2	7	6,3	2,6
12.000	450,3	319,4	58,7	136,2	147,2	97,9	42	37,7	40,9	25,8	18,5	9,5	11,1	6,3	5,9	2,4
13.000	429,6	302,7	52,8	127,5	148,3	95,9	38,2	35	38,6	23,3	17,2	8,8	9,5	5,9	5,4	2,2
14.000	411,3	287,2	52,7	121,5	139,6	91,1	37,8	33,2	36	24,1	17,6	8,1	9	6	5,6	2
15.000	394,2	274,3	45,4	115	131,7	85	33,2	31,2	33,4	20,7	15,2	7,7	8,2	5,3	4,8	1,9
16.000	369,9	257,68	44,1	105,9	122,6	79,3	31,8	28,8	30,9	20,7	15,1	7,2	7,8	5,2	4,7	1,8

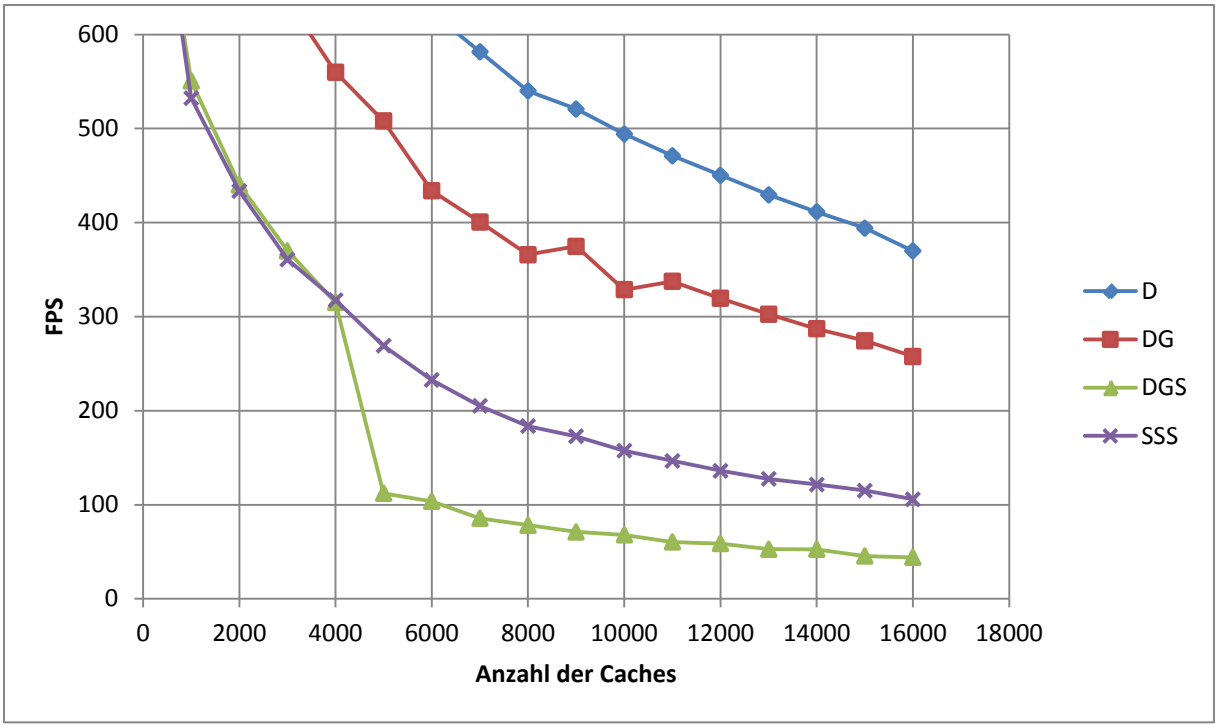

Abbildung 6.19: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 4.096 VALs genutzt werden (64x64).

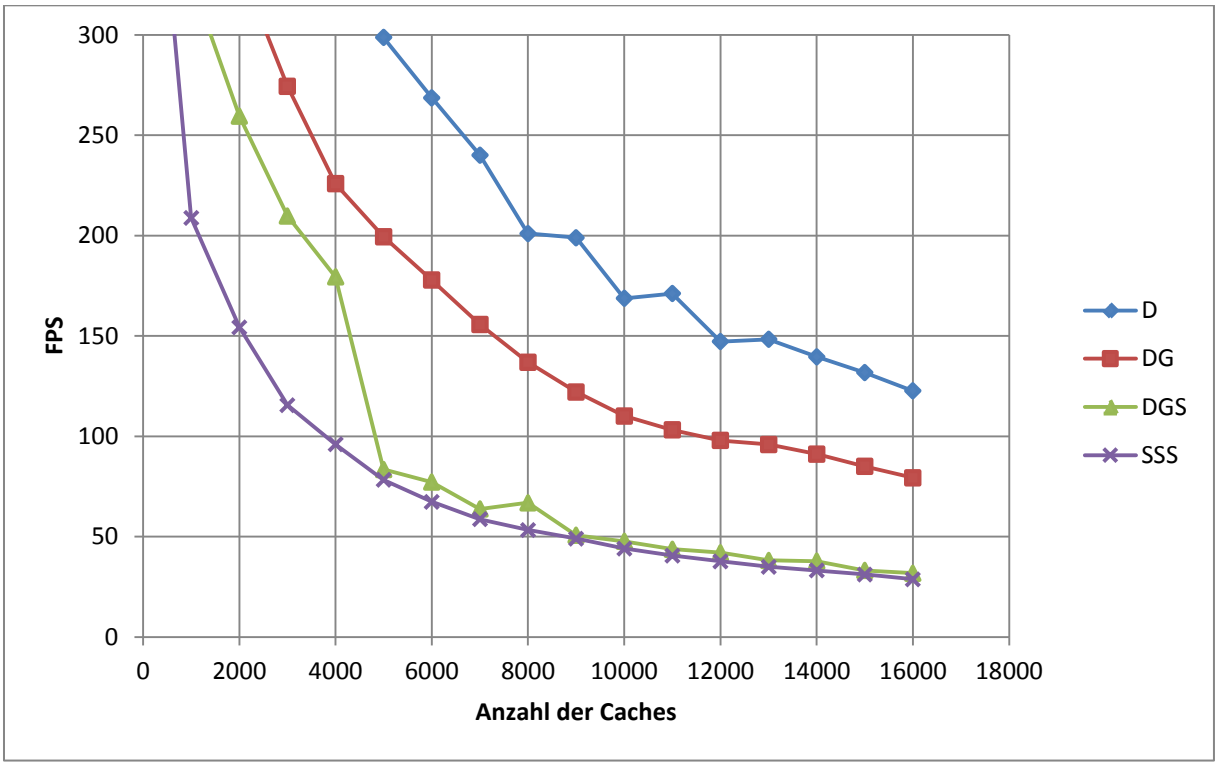

Abbildung 6.20: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 16.384 VALs genutzt werden (128x128).

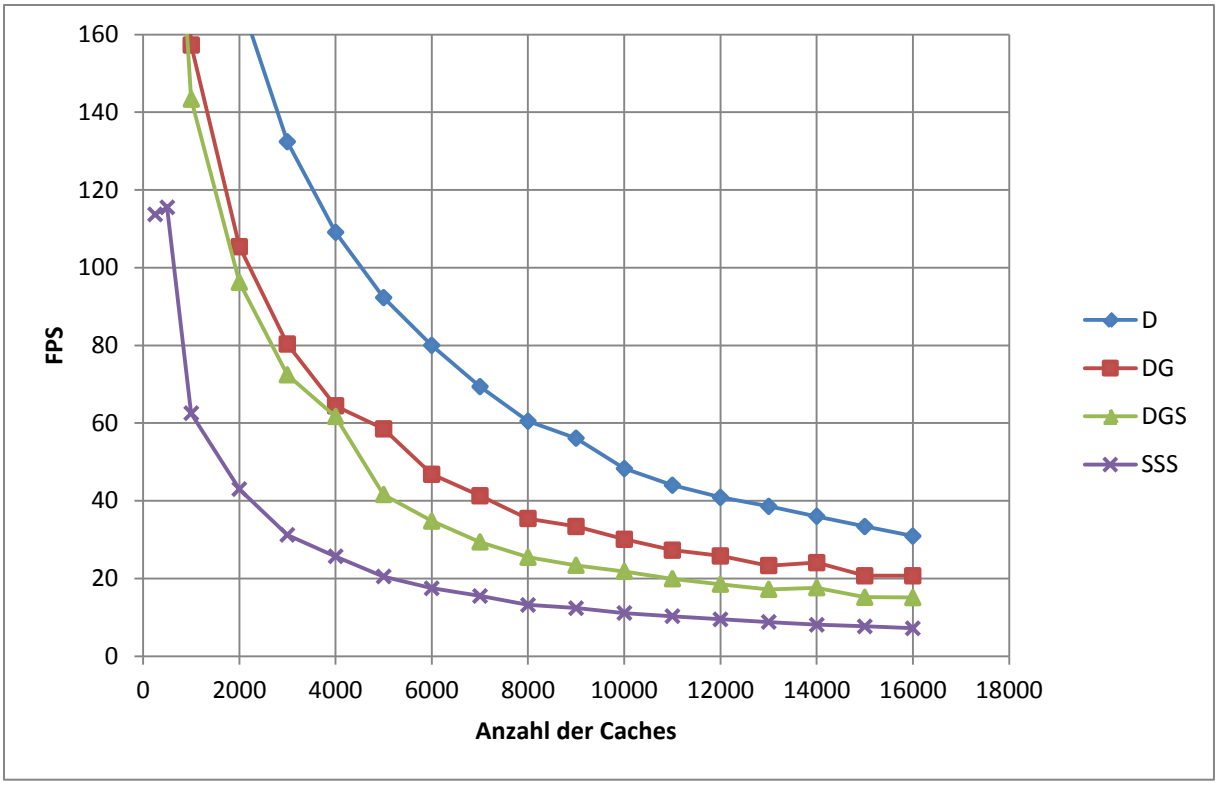

Abbildung 6.21: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 65.536 VALs genutzt werden (256x256).

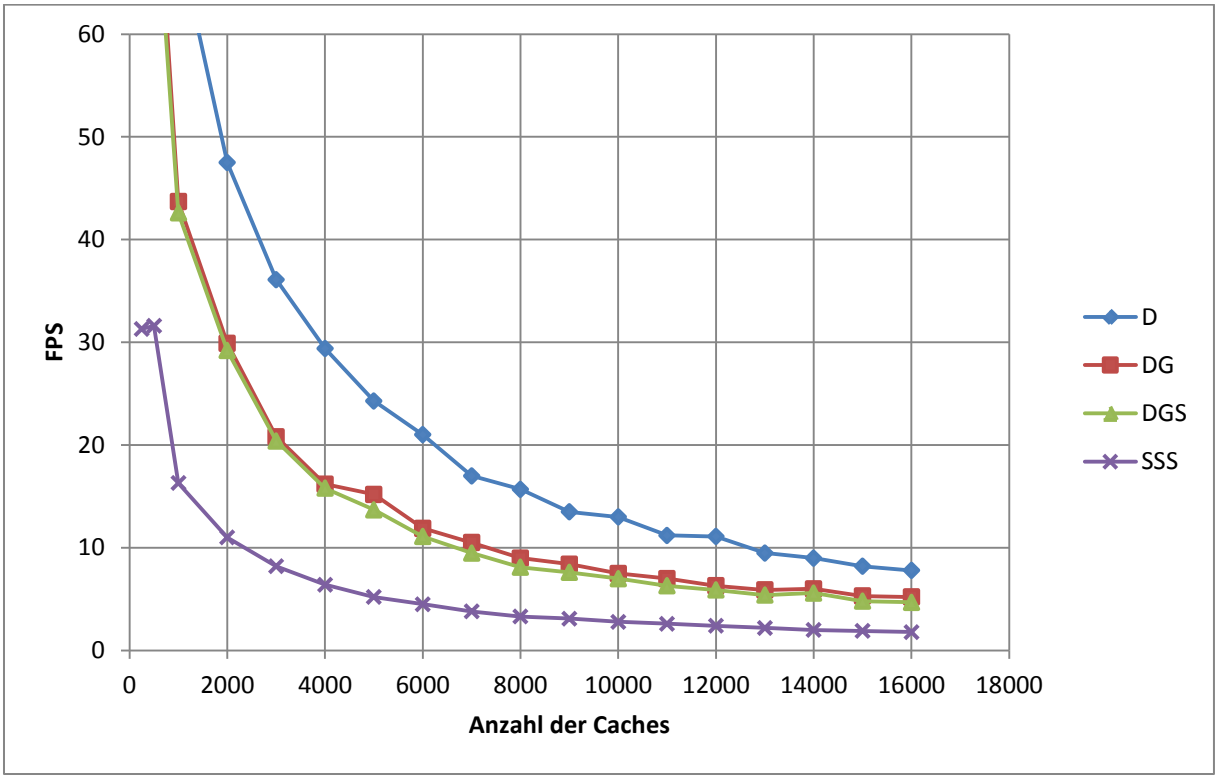

Abbildung 6.22: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 262.144 VALs genutzt werden (512x512).

6.2.3 Vergleich zu Light-Propagation-Volumes

In Kapitel 4.2 wurde darauf hingewiesen, dass die Light-Propagation-Volumes (LPV) von Kaplanyan und Dachsbacher [KD10] eine Alternative zum hier vorgestellten Ansatz zur Berechnung von globalen Beleuchtungseffekten darstellen. Bei den LPV werden globale Beleuchtungsphänomene durch einen Discrete-Ordinate-Ansatz berechnet, bei dem das indirekte Licht innerhalb einer kubischen Gitterstruktur von Nachbarknoten zu Nachbarknoten fortberechnet wird.

Für einen Vergleich zum LightSkin-Verfahren sollen die folgenden Kriterien betrachtet werden, die für das Anwendungsfeld VR und AR von besonderer Bedeutung sind (siehe auch Kapitel 4.1):

1. Dynamik und Interaktivität,
2. Darstellungsrate und Auflösung,
3. Qualität/Plausibilität und
4. Komplexität der abbildbaren Szenen.

Weitere Kriterien, wie beispielsweise die Details in der Implementierung, können an dieser Stelle nicht verglichen werden, da dem Autor dieser Arbeit keine Implementierungsdetails zu den LPV vorliegen. Im Folgenden werden die Kriterien im Einzelnen diskutiert.

Dynamik und Interaktivität: Der LPV-Ansatz unterstützt dynamische Szenen ohne Beeinträchtigung der Laufzeit des Verfahrens, wie dies bei dem LightSkin-Verfahren ebenfalls der Fall ist. Allerdings können bei Animationen Artefakte entstehen, wenn die Gitterstruktur im Vergleich zum animierten Objekt verhältnismäßig grob gewählt wird. Durch diese grobe Gitterstruktur werden feine Animationen nur unzureichend in der indirekten Beleuchtung widergespiegelt, so dass diese die Veränderungen nicht adäquat repräsentiert. Ferner kann man bei der Erzeugung von weichen Schatten die diskrete Natur des Gitters erkennen, insbesondere wenn Objekte innerhalb des Gitters kontinuierlich verschoben werden.

Darstellungsrate und Auflösung: Dem Autor dieser Arbeit lag nur eine vorkompilierte Version der LPV vor, die freundlicherweise von Tobias Franke [Fra13a] zur Verfügung gestellt wurde. Diese Implementierung konnte nur im geringen Maße angepasst werden. Es hat sich aber in Tests gezeigt, dass die LPV die Crytek-Sponza-Szene unter Verwendung einer NVidia Geforce GTX 780 Grafikkarte mit ca. 160 FPS darstellen können (64 Iterationen innerhalb des Gitters), während die gleiche Szene mit dem LightSkin-Verfahren mit ca. 103 FPS dargestellt werden kann. Dabei wurden beim LightSkin-Verfahren 3.000 Caches in der Szene verteilt und eine RSM-Auflösung von 128x128 Pixeln genutzt (damit keine temporären Artefakte bei Animationen entstehen). Diese Einstellungen lieferten bereits deutlich bessere Ergebnisse als der LPV-Ansatz (alle Abbildungen der Sponza-Szene in dieser Dissertation wurden mit der gleichen Cache-Verteilung vorgenommen, die lediglich 3.000 Caches für die komplette Szene enthält). Hieraus lässt sich ableiten, dass der LPV-Ansatz bei vergleichbaren Szenen ca. um das 1,5fache schneller als der LightSkin-Ansatz ist. Für diese Angabe übernimmt der Autor aber keine Gewähr, da die Details der Implementierung nicht einsehbar waren. Prinzipiell kann man davon ausgehen, dass die LPV schneller Ergebnisse erzeugen, allerdings sind diese qualitativ deutlich schlechter als die des LightSkin-Verfahrens. Die Laufzeit beider Verfahren ist relativ unabhängig von der Auflösung des Framebuffers, so dass hohe Auflösungen dargestellt werden können. Da bei den LPV die indirekte Beleuchtung der Szene durch ein volumetrisches Gitter abgebildet wird, ist es wie

das LightSkin-Verfahren weitestgehend unabhängig von der Anzahl der Dreiecke (siehe vorheriges Unterkapitel), weshalb Messungen für unterschiedliche Szenen nicht sinnvoll erscheinen.

Qualität/Plausibilität: In Abbildung 6.23 wird ein Vergleich der diffusen Lichtausbreitung basierend auf den LPV und dem LightSkin-Verfahren zu einer Ground-Truth-Lösung gezeigt. Man erkennt in der Abbildung deutlich, dass die Ausbreitung des indirekten Lichts durch die LPV unzureichend approximiert wird. Das Licht nimmt zu schnell mit der Entfernung ab und die direktionalen Eigenschaften gehen verloren. Dies hängt mit der ungenauen Fortberechnung des indirekten Lichts innerhalb des diskreten Gitters zusammen und dessen Abbildung in einer geringen Anzahl von Spherical-Harmonics-Koeffizienten. Vergleicht man dagegen die indirekten Reflexionen des LightSkin-Verfahrens mit der Ground-Truth-Lösung, erkennt man, dass die Lösungen sehr ähnlich sind. Prinzipiell können die LPV auch glänzende Reflexionen abbilden, allerdings hat der Autor dieser Arbeit keine Implementierung finden können, die diese unterstützt. Es ist jedoch zu erwarten, dass die glänzenden Reflexionen auch nur sehr approximativ abgebildet werden können.

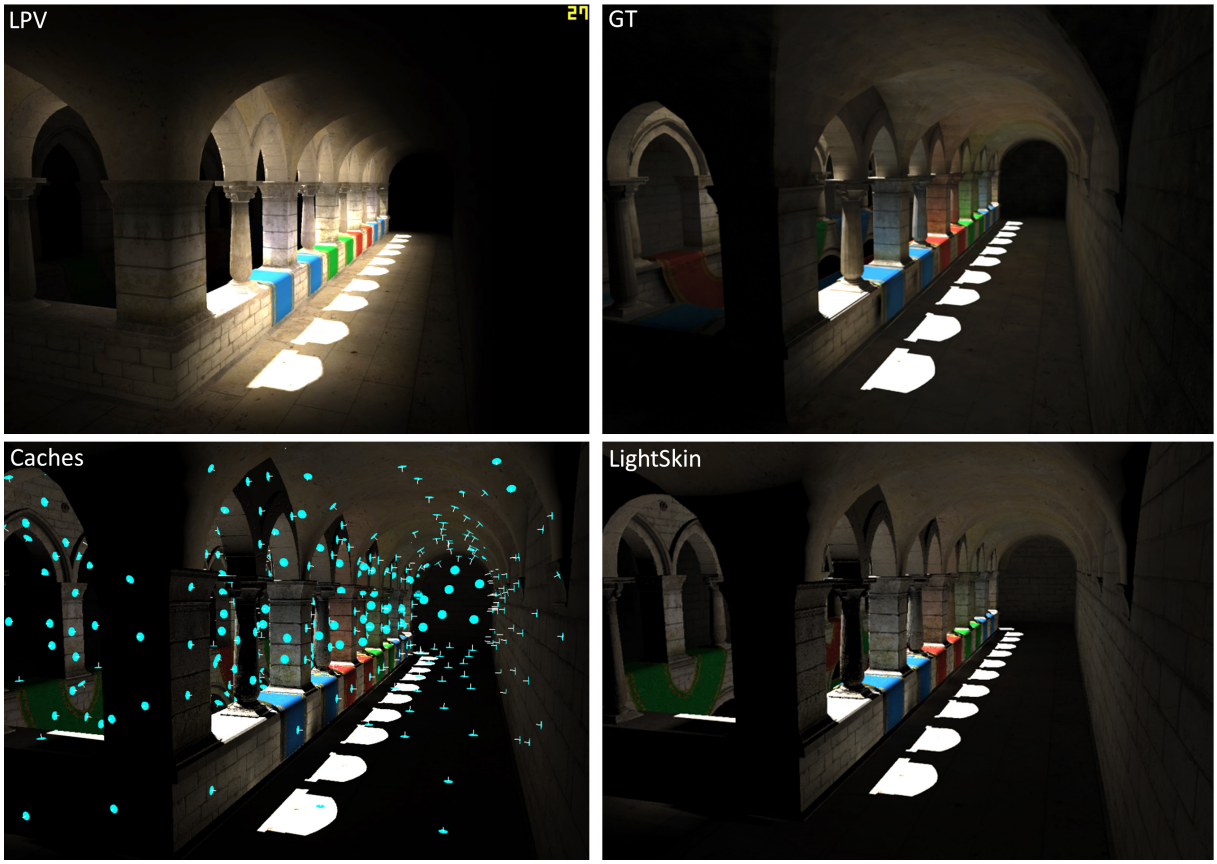

Abbildung 6.23: Vergleich zwischen LPV (oben, links), Ground-Truth (oben, rechts) und LightSkin (unten, rechts). Man erkennt deutlich, dass das LightSkin-Verfahren näher an der Ground-Truth-Lösung ist. Unten links wird die genutzte Cache-Verteilung gezeigt. Zum Bild tragen nur wenige hundert Caches bei. Die oberen Bilder wurden aus [CNS*11] entnommen.

Komplexität der abbildbaren Szenen: Die LPV unterstützen komplexe Szenen durch die Nutzung von verschiedenen Kaskaden des volumetrischen Gitters. Für nahe Objekte wird ein sehr feines Gitter genutzt, während ferne Objekte durch ein grobes Gitter abgebildet werden. Demnach skaliert das Verfahren sehr gut mit komplexen Szenen. Dies gilt ebenfalls für das LightSkin-Verfahren, da nur Caches im Sichtbereich genutzt werden und darüber hinaus nur sehr wenig Caches vorhanden sein müssen,

um gute Beleuchtungsergebnisse zu erzeugen. Ferner können weitere Skalierungsmethoden implementiert werden, die in Kapitel 6.1.6 vorgestellt wurden.

Es ist erkennbar, dass das LightSkin-Verfahren bessere Beleuchtungsergebnisse erzeugt und ungefähr die gleichen Laufzeitanforderungen erfüllt wie das LPV-Verfahren. Dies gilt insbesondere, wenn die Szenen Animationen enthalten.

6.2.4 Vergleich zum Voxel-Cone-Tracing

Der Voxel-Cone-Tracing-Ansatz (VCT), welcher von Crassin et al. in [CNS*11] vorgestellt wurde, soll ebenfalls mit dem LightSkin-Verfahren verglichen werden. Beim VCT handelt es sich um einen Finite-Elements-Ansatz, bei dem die Struktur der Oberflächen durch einen Voxel-Octree approximiert wird. Dieser Octree wird nur für die Berechnung des indirekten Lichts genutzt (siehe Kapitel 3.1.5). Das VCT soll in den gleichen Kategorien mit dem LightSkin-Verfahren verglichen werden, wie bereits die LPV:

Dynamik und Interaktivität: Animationen können beim VCT berücksichtigt werden, jedoch verursachen diese eine teilweise Rekonstruktion des Voxel-Octrees. Diese Rekonstruktion führt dazu, dass die Laufzeit des Verfahrens deutlich beeinflusst wird. Allerdings verursachen Animationen keine Artefakte, da die im Octree gespeicherten Ergebnisse für jeden Raumpunkt quadrilinear interpoliert werden, so dass Übergänge sauber ineinander gemischt werden.

Darstellungsrate und Auflösung: Wie bei den LPV lag dem Autor dieser Arbeit nur eine vorkompilierte Version einer VCT-Implementierung vor, die ebenfalls freundlicherweise von Tobias Franke [Fra13a] zur Verfügung gestellt wurde. Mit dieser Implementierung wurden die Bildwiederholungsraten für die Crytek-Sponza-Szene gemessen, mit dem Ergebnis, dass diese mit dem VCT mit ca. 38 FPS dargestellt werden kann (gleiche Systemvoraussetzungen wie im vorherigen Kapitel, es wurde eine Auflösung von 1280x720 Pixeln gewählt). Die gleiche Szene kann mit dem LightSkin-Verfahren mit 103 FPS dargestellt werden. Demnach ist das LightSkin-Verfahren bei einer vergleichbaren Szene schneller als das VCT (wie bereits bei den LPV ist ein Vergleich für mehrere Szenen wenig sinnvoll, da die Geschwindigkeit des VCTs mit der Auflösung des Voxel-Octrees skaliert und nur bedingt mit der Szenenkomplexität zusammenhängt). Diese Ergebnisse sind jedoch ohne Gewähr, da dem Autor keine Implementierungsdetails bekannt waren. Die Laufzeitkosten für einen einzelnen Pixel im Framebuffer sind sowohl beim LightSkin-Verfahren als auch beim VCT konstant, allerdings ist der konstante Faktor beim VCT wesentlich höher. Da für jeden FrameBuffer-Pixel sehr viele Lesezugriffe in den Octree durchgeführt werden müssen, sind hohe Auflösungen in Echtzeit auf aktuellen PC-Systemen noch nicht mit dem VCT möglich. Dies gilt insbesondere, wenn das Bild viele glänzende Oberflächen enthält, die weitere Lesezugriffe in den Octree erfordern (Ray-Marching).

Qualität/Plausibilität: In Abbildung 6.24 befindet sich ein Vergleich des VCT-Verfahrens mit dem LightSkin-Verfahren zu einer Ground-Truth-Lösung. Man erkennt in der Abbildung, dass die diffusen Reflexionen durch das LightSkin-Verfahren exakter abgebildet werden. Beim VCT kann man in der Abbildung erkennen, dass das Licht fälschlicherweise um die direkten Reflexionen herum ausblutet. Dies ist kein gewünschter Effekt eines Blooming-Shaders, sondern das Ergebnis einer unzureichenden Abbildung der Szene durch die Voxel-Struktur. Ferner erkennt man, dass das diffuse Licht mit der Entfernung zu stark abnimmt, wie dies bereits bei den LPV zu beobachten war. Die diffuse Lichtausbreitung wird demnach beim LightSkin-Verfahren exakter abgebildet als beim VCT. Allerdings können dagegen glänzende Reflexionen subjektiv besser mit dem VCT-Verfahren abgebildet werden. Insbe-

sondere im Kontaktbereich zwischen den Objekten sind diese deutlicher ausgeprägt. In diesen räumlichen Bereichen erzeugt das LightSkin-Verfahren einen Energieabfall, der in Abbildung 6.17 dargestellt ist. Glänzende Reflexionen erfordern allerdings beim VCT-Verfahren einen erhöhten Speicher- und Laufzeitaufwand. Spiegelnde Reflexionen lassen sich mit beiden Verfahren nicht erzeugen, da diese eine zu exakte Repräsentation der Szene erfordern würden. Es lässt sich festhalten, dass das VCT glänzende Reflexionen besser abbildet, während das LightSkin-Verfahren diffuse Reflexionen korrekter darstellt.

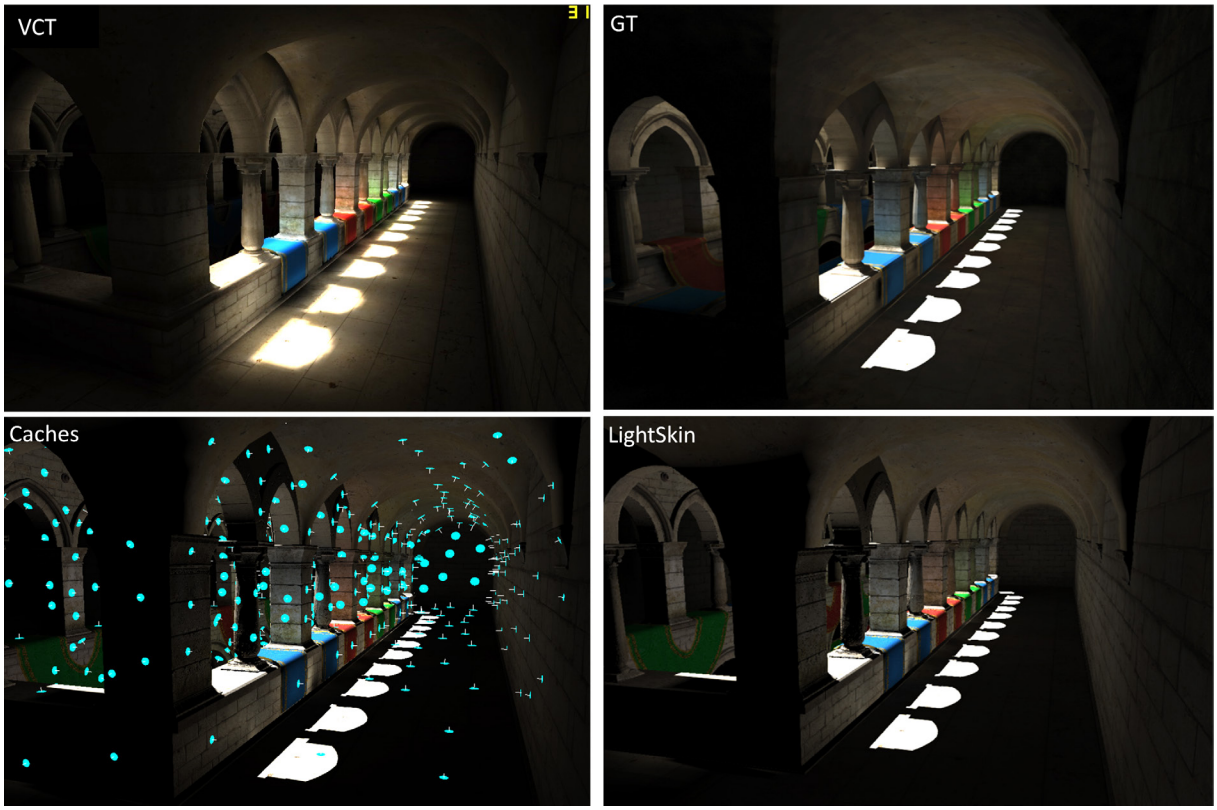

Abbildung 6.24: Vergleich zwischen VCT (oben, links), Ground-Truth (oben, rechts) und LightSkin (unten, rechts). Man erkennt deutlich, dass das LightSkin-Verfahren am nächsten zu der Ground-Truth-Lösung ist. Unten links wird die genutzte Cache-Verteilung gezeigt. Zum Bild tragen nur wenige hundert Caches bei. Die oberen beiden Bilder wurden aus [CNS*11] entnommen.

Komplexität der abbildbaren Szenen: Die Abbildung durch den Voxel-Octree erfordert viel Speicher, weshalb dieser für eine komplexe Szene nicht vollständig im Grafikkartenspeicher vorliegen kann. Crassin et al. nutzen eine komplexe Out-of-Core-Technik, um trotzdem große Szenen unterstützen zu können. Müssen die Dreiecksdaten der Szene selbst durch einen Out-of-Core-Ansatz geladen werden, muss die zur Verfügung stehende Datenrate mit dem VCT geteilt werden. Dies ist ein deutlicher Nachteil des VCTs bei der Verarbeitung von komplexen Szenen. Unterstützen die Szenen zusätzlich Animationen, müssen die komplexen Modelle permanent „voxelisiert“ werden. Bei dieser Voxelisierung wird jedes Dreieck des Modells gerastert, ohne die Sichtbarkeit des Dreiecks beachten zu können. Besteht das Modell aus sehr vielen Dreiecken, ist dieser Vorgang entsprechend aufwändig. Durch die geringen Speicheranforderungen und die einfache Transformation der Caches mit dem Modell liefert das LightSkin-Verfahren hier bessere Ergebnisse.

Die Diskussion hat gezeigt, dass das VCT für glänzende Oberflächen bessere Ergebnisse liefert, während diffus reflektierende Oberflächen durch das LightSkin-Verfahren exakter approximiert werden können. Das LightSkin-Verfahren ist im Vergleich zur vorliegenden Implementierung des VCT ca. doppelt so schnell bei vergleichbaren Szenen, allerdings lässt sich behaupten, dass beide Verfahren in der gleichen Leistungsklasse liegen. Im Gegensatz zum VCT können Animationen und komplexe Szenen beim LightSkin-Verfahren deutlich effizienter dargestellt werden. Sehr hohe Bildauflösungen sind mit dem VCT derweil noch nicht in Echtzeit möglich. Es ist jedoch davon auszugehen, dass dies mit zukünftigen Generationen von Grafikkarten möglich sein wird, da die Kosten pro Pixel konstant sind.

6.2.5 Klassifikation

Alle in dieser Arbeit vorgestellten interaktiven Verfahren zur Erzeugung von globalen Beleuchtungseffekten wurden nach dem Kategoriensystem von Ritschel et al. [RDG*12] klassifiziert (siehe Kapitel 3.1). Für das LightSkin-Verfahren soll in dieser Arbeit ein Vorschlag für eine Klassifikation gemacht werden, die im Vergleich zu den anderen Verfahren vorgenommen wurde:

Methode		Geschw.	Qualität	Dynam.	Skalierb.	Implem.	GPU	Transport
Lensing & Broll	LightSkin	5	3	5	4	5	5	$LD^{+}(S\|D\|VD)E$

Die Klassifikation der Geschwindigkeit resultiert aus der Tatsache, dass das LightSkin-Verfahren mit seinen Bildwiederholungsraten zwischen dem LPV-Ansatz und dem VCT-Ansatz liegt, die beide mit fünf bewertet wurden. Die Darstellungsqualität von drei ergibt sich ebenfalls durch den Vergleich zum LPV- und zum VCT-Ansatz. Diese liegt beim LightSkin-Verfahren deutlich über der der LPV und kann als gleichwertig zum VCT angesehen werden. Sowohl das VCT- als auch das LightSkin-Verfahren unterstützen den gleichen Lichtpfad für globale Beleuchtungseffekte, der diffuse und glänzende indirekte Reflexionen mit weichen Schatten beinhaltet. Die Abbildung von Kaustiken ist mit beiden Verfahren nicht ohne weiteres möglich. Allerdings unterstützt das LightSkin-Verfahren zusätzlich noch die Abbildung von Subsurface-Scattering-Materialien.

Die Höchstwertung für die Dynamik ergibt sich dadurch, dass Animationen keine Artefakte und keine zusätzlichen Kosten verursachen. Darüber hinaus werden sämtliche Animationstypen unterstützt. Die Skalierbarkeit des Verfahrens mit komplexen Szenen und hohen Auflösungen ist vergleichbar zu den LPV, weswegen deren Wert übernommen wurde. Die Implementierung und die Unterstützung der GPU wurden jeweils mit der höchsten Note versehen, da die Implementierung komplett für die GPU vorgenommen werden kann, kein komplexes Shader-Modell genutzt werden muss und der Speicherzugriff so optimiert wurde, dass die meisten Daten aus dem Textur-Cache geladen werden können. Ferner basiert die Implementierung auf intuitiven Ansätzen.

7 Adaption des LightSkin-Verfahrens für Augmented Reality

7.1 Überblick

In Kapitel 5 wurden die grundlegenden Prinzipien des LightSkin-Verfahrens erläutert. Dabei wurde der Anwendungsfall Augmented Reality vorerst nicht berücksichtigt (obwohl dieser ebenfalls von den Optimierungen für Virtual-Reality-Anwendungen profitiert). Das Anwendungsgebiet AR beinhaltet jedoch einige Herausforderungen, die in dieser Arbeit nicht unbeachtet bleiben sollen.

Durch das Differential Rendering [FGR92,Deb08b], das in Kapitel 2.2.2 vorgestellt wurde, lässt sich prinzipiell jeder Ansatz zur Simulation von globalen Echtzeit-Beleuchtungseffekten in ein AR-Szenario übertragen. Dabei muss man allerdings den Begriff des AR-Szenarios relativieren, denn die Berechnung der Wechselwirkung des Lichts zwischen einer realen und einer virtuellen Szene setzt voraus, dass die reale Szene dem System als Modell bekannt sein muss. Dies bedeutet nichts anderes, als dass die Geometrie der Oberflächen, also die Positionen und die Normalen, sowie deren Materialeigenschaften, bekannt sein müssen. Die meisten AR-Beleuchtungsansätze kommen dieser Forderung nur bedingt nach, indem sie voraussetzen, dass die reale Szene als Modell im Voraus erstellt wurde und von dem Verfahren direkt genutzt werden kann. Folgt man diesem Prinzip, so lässt sich das LightSkin-Verfahren direkt für eine AR-Anwendung verwenden und bedarf bis auf die Erweiterung durch das Differential Rendering keiner weiteren Modifikation. Der Nachteil ist, dass eine a-priori-Rekonstruktion der realen Szene dazu führt, dass diese während der Simulation statisch bleiben muss. Dies ist in gewisser Weise unbefriedigend, denn das LightSkin-Verfahren soll eine höchstmögliche Dynamik erlauben. Deshalb soll in dieser Arbeit nicht davon ausgegangen werden, dass die Szene im Vorhinein erstellt wurde. Stattdessen wird ein Verfahren vorgeschlagen, das mit Hilfe eines aktiven RGB-D-Sensors die Rekonstruktion der Szene in Echtzeit vornehmen kann. Damit diese Rekonstruktion in Echtzeit erfolgen kann und Veränderungen der Szene direkt berücksichtigt werden können, müssen jedoch einige Einschränkungen hingenommen werden, die in Kapitel 7.3 noch erläutert werden.

Ein Problem, das sich für das LightSkin-Verfahren durch die Echtzeit-Szenenkonstruktion ergibt, besteht in der Verteilung der Caches. In einer VR-Anwendung können die Caches im Voraus auf den Modelloberflächen verteilt werden. In einer AR-Anwendung dagegen müssen diese, wegen der Echtzeitrekonstruktion, zur Laufzeit der Simulation verteilt werden. Darüber hinaus wurde bei der Verteilung der Caches von einem triangulierten Modell ausgegangen. Die Konstruktion eines expliziten Modells aus Dreiecken auf Basis der Daten eines RGB-D-Sensors ist aber sehr fehleranfällig, insbesondere im Kontext der Echtzeitausführung. Verfahren, wie sie zum Beispiel in [MRB09] vorgestellt werden, erzeugen in diesem Anwendungsfeld keine zufriedenstellenden Ergebnisse. Für die Verteilung der Caches muss ein neuer Ansatz gefunden werden. Dieser neue Ansatz soll sich nahtlos in das vorhandene LightSkin-Verfahren eingliedern lassen.

Aus den vorangegangenen Überlegungen lassen sich die folgenden Arbeitsschritte ableiten:

1. Rekonstruktion der Szene anhand eines aktiven RGB-D-Sensors. Dies ist Thema des Kapitels 7.2.

2. Abbildung der direkten und indirekten realen Beleuchtung, sowie die Verteilung der Caches in Echtzeit. Dies wird in Kapitel 7.3 beschrieben.

7.2 Abbildung der realen Geometrie

Zur Rekonstruktion der realen Szene wird in dieser Arbeit ein RGB-D-Sensor eingesetzt. Ein solcher Sensor liefert zum herkömmlichen RGB-Farbbild für jeden Pixel zusätzlich einen Tiefenwert (siehe Abbildung 7.1). In dieser Arbeit wurde der Microsoft Kinect-Sensor eingesetzt. Im Wesentlichen gelten die hier aufgeführten Prinzipien aber für jeden beliebigen Tiefensensor, insbesondere auch für den Kinect2-Sensor.

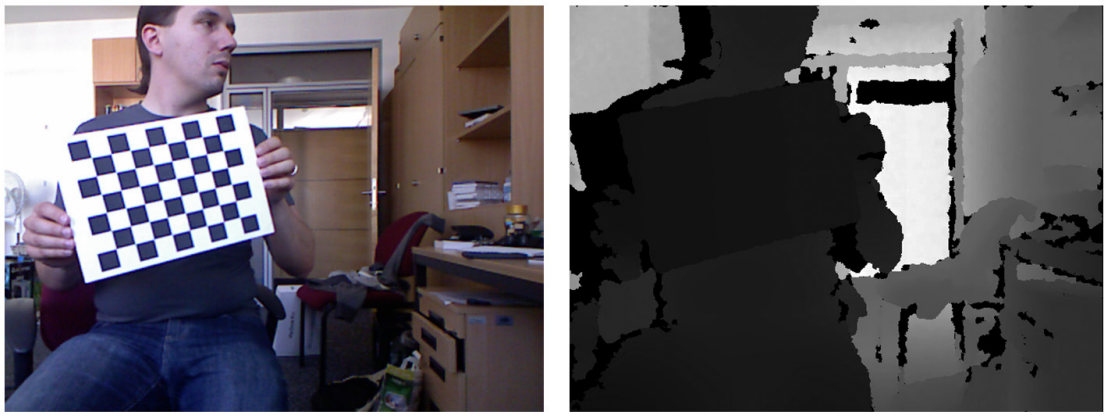

Abbildung 7.1: Daten eines RGB-D-Sensors (hier der Microsoft Kinect-Sensor). Zusätzlich zum Farbbild (links) wird eine Tiefenabbildung erzeugt (rechts).

Die Forderung, dass dynamische Szenen im globalen Beleuchtungsverfahren berücksichtigt werden sollen, beinhaltet einige Implikationen, die die Qualität der Daten der realen Szene betreffen. Für statische Szenen existieren Verfahren [NIH*11,ND10], die eine relativ hochwertige Abbildung der Szene im Voraus erlauben. Diese Verfahren basieren auf iterativen Ansätzen, die sukzessiv die Qualität der Abbildung verbessern, je länger die reale Szene durch den Sensor aufgenommen wird. Dabei nimmt der Sensor die Szene aus verschiedenen Blickwinkeln auf, um so die Abbildung zu vervollständigen. Der Nachteil dieser iterativen Ansätze besteht darin, dass Veränderungen in der Szene (zum Beispiel verschobene Objekte) nicht direkt als solche erkannt werden. Stattdessen werden die Veränderungen erst nach einigen Sekunden (3-10 Sekunden) in die Abbildung übernommen. Ein weiteres Problem stellt das verwendete Datenformat der Abbildung dar. Hier werden häufig implizite Volumenmodelle [NIH*11,KGX*11] genutzt. Hieraus resultieren hohe Speicheranforderungen für die Abbildung. Außerdem ist im Vergleich zu triangulierten Modellen das Zeichnen dieser Volumenmodelle verhältnismäßig teuer. Für rechenaufwändige Beleuchtungsverfahren, die die Szene aus mehreren Perspektiven abtasten müssen, ist diese Datenrepräsentation ungeeignet. Zwar bestehen etablierte Verfahren, um aus Volumenmodellen triangulierte Modelle zu erzeugen [LC87], diese sind aber für komplexe Modelle nicht echtzeitfähig.

Eine Alternative zu iterativen Verfahren, sind Verfahren, die nur die aktuellsten Daten des Sensors für die Abbildung berücksichtigen. Diese Verfahren haben den Vorteil, dass sie Veränderungen in der Szene unmittelbar abbilden können, da sie kein iteratives Modell aktualisieren müssen. Allerdings impliziert die reine Verwendung der aktuellsten Daten ebenfalls, dass von der Szene nur eine Perspektive in Form eines Farb- und Tiefenbilds besteht. Die reale Szene wird demnach nicht vollständig

abgebildet. Diese Unvollständigkeit ist gerade für globale Beleuchtungsverfahren ein nicht zu unterschätzendes Problem, das in Kapitel 7.3 noch genauer behandelt wird. Darüber hinaus ist die Qualität des Tiefenbilds unzureichend, um in einem Beleuchtungsverfahren direkt verwendet zu werden. Aus diesem Grund wird in Kapitel 7.2.2 ein Verfahren zur Filterung des Tiefenbilds vorgestellt, dass die Qualität der Tiefenabbildung verbessert.

Beim LightSkin-Verfahren wird die reale Szene durch eine Perspektive des RGB-D-Sensors repräsentiert, die aus dem Farb- und Tiefenbild besteht. In den noch folgenden Unterkapiteln wird erläutert, wie diese Repräsentation qualitativ so stark verbessert werden kann, dass sie für die Nutzung in einem Beleuchtungsverfahren geeignet ist.

7.2.1 Die Tiefenabbildung

Sowohl das Tiefenbild, als auch das Farbbild eines RGB-D-Sensors lassen sich prinzipiell direkt für die Initialisierung der G-Buffer bei einem Deferred-Shading-Ansatz [ST90] verwenden. Das Farbbild kann direkt in den G-Buffer für diffuse Materialien und das Tiefenbild in den Tiefen-G-Buffer kopiert werden. Der G-Buffer, der die Normalen enthält, kann befüllt werden, indem die Pixelnachbarschaften im Tiefenbild genutzt werden, um den Gradienten und somit die Normale eines Pixels zu bestimmen. Dabei werden nur die Nachbarpixel für die Bildung des Gradienten berücksichtigt, die sich vom untersuchten Pixel nicht zu stark in der Tiefe unterscheiden. Die Qualität des Tiefenbildes erlaubt jedoch keine direkte Nutzung in einem Deferred-Shading-System, wie Abbildung 7.2 zeigt. In dieser Darstellung wird nur direktes Licht berücksichtigt, man erkennt aber trotzdem deutlich, dass die Oberflächen der Szene stark verrauscht sind.

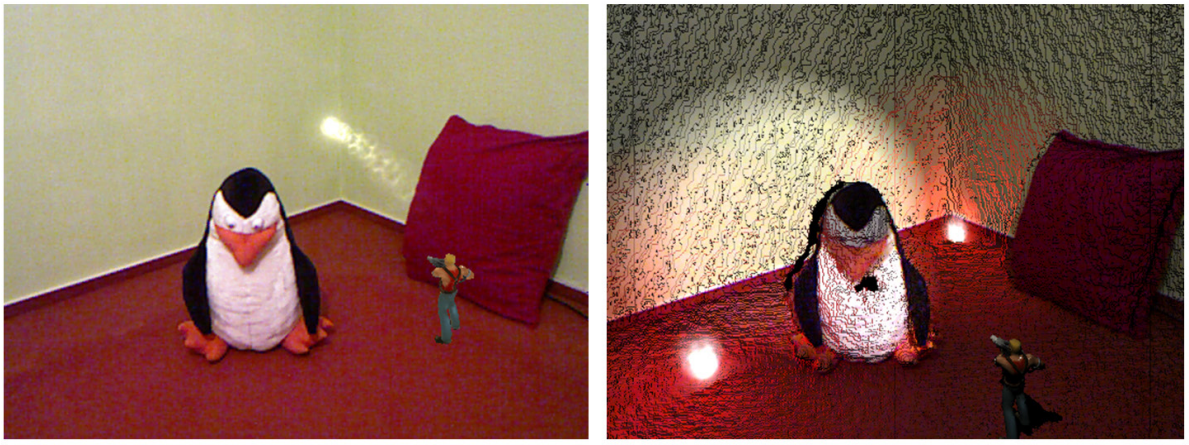

Abbildung 7.2: Die reale Szene (links) wird durch einen RGB-D-Sensor aufgenommen. Verwendet man die Daten des Sensors direkt in einem Deferred-Shading-System (rechts), so erkennt man, dass die Tiefenrepräsentation stark verrauscht ist. In der rechten Szene wurden virtuelle Lichter hinzugefügt, damit man die Unebenheiten in der Tiefe erkennen kann.

Neben diesem Rauschen existieren weitere Probleme bei der Verwendung des Tiefenbildes für die Beleuchtungsberechnung (siehe auch [LB11]):

1. Die laterale Auflösung des Tiefenbilds ist deutlich geringer als die des Farbbilds.
2. Die Abbildung der Tiefe wird mit zunehmender Entfernung ungenauer.
3. Es werden lokal falsche Tiefenwerte berechnet.
4. Die Tiefenwerte sind temporär instabil und flackern.

5. Es existieren Regionen, in denen keine Tiefeninformationen vorhanden sind (durch Abschattung, steile Oberflächen, spiegelnde oder glänzende Materialien, etc.).

In dieser Dissertation wird der fünfte Punkt nicht behandelt, da dieses bereits in früheren Veröffentlichungen und Qualifikationsarbeiten des Autors [LB11,Len12] geschehen ist. Liu et al. [LGL12] schlagen ebenfalls Verfahren zur kontextbezogenen Füllung (Inpainting) von undefinierten Regionen im Tiefenbild vor. Die Probleme, die in den Punkten zwei bis vier beschrieben werden, können durch eine kantenerhaltende Filterung des Tiefenbildes gelöst werden. Diese Art der Filterung vermindert das Rauschen in ebenen Flächen und erhält feine Strukturen in Bereichen, in denen hochfrequente Veränderungen in der Tiefenabbildung vorliegen. Das Tiefenbild des Kinect-Sensors beinhaltet jedoch nicht nur hoch-, sondern auch niederfrequentes Rauschen. Dies führt dazu, dass große Kernel zur Filterung eingesetzt werden müssen (Filtermasken von 16x16 bzw. 32x32 Pixel sind nicht unüblich, siehe Abbildung 7.3). Die meisten kantenerhaltenden Filtermethoden, wie zum Beispiel die Median-Filterung, werden durch große Kernel deutlich langsamer und können deshalb nicht in einer Echtzeitanwendung eingesetzt werden. Zwar existieren Implementierungen von bilateralen und Median Filtern [YTA09,PH07], deren Laufzeitkomplexität nicht von der Kernelgröße abhängig ist, aber diese Filter setzen einen relativ begrenzten Wertebereich für jeden Pixel voraus. Wertebereiche über 256 Stufen können damit nicht effizient abgebildet werden. Das Tiefenbild hat jedoch einen Wertebereich von ca. 16.000 Abstufungen. Eine Lösung für dieses Problem bietet das Guided Image-Filtering (GIF), das 2010 von He et al. [HST10] vorgestellt wurde. Hierbei handelt es sich um einen kantenerhaltenden Filter, dessen Laufzeitkomplexität unabhängig von der Kernelgröße ist. Die Filterung wird bei diesem Ansatz durch ein separates Bild, dem sogenannten Guidance-Image, gesteuert. Das Frequenzverhalten des Guidance-Image regelt, wie stark ein Tiefenpixel durch seine Nachbarpixel gemittelt wird. Wählt man als Guidance-Image das zu filternde Tiefenbild, so erhält man einen klassischen bilateralen Filter. In dieser Arbeit wird jedoch vorgeschlagen, anstatt des Tiefenbilds das Farbbild des RGB-D-Sensors als Guidance-Image zu verwenden. Das Farbbild bietet den Vorteil, dass es deutlich stabilere Werte als das Tiefenbild aufweist, insbesondere in Bereichen, in denen starke Kontraste auftreten. Hierdurch wird das Flackern der Tiefenwerte deutlich reduziert. Darüber hinaus besitzt das Farbbild eine höhere Auflösung, die es erlaubt, das Tiefenbild kontrolliert hoch zu skalieren. Somit wird auch der erste Punkt aus der obigen Aufzählung, der die geringe laterale Auflösung des Tiefenbilds benennt, durch das GIF aufgehoben.

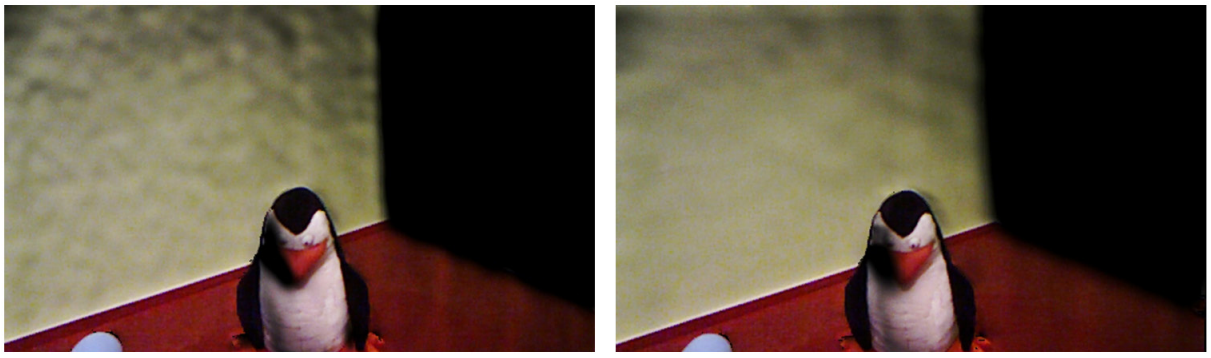

Abbildung 7.3: Kernelgrößen für Tiefenbildfilterung. Auf beiden Bildern wurde eine Gaussian-Blur-Filterung durchgeführt. Bei dem linken Bild wurde eine Standardabweichung von 6 gewählt, die durch einen relativ kleinen diskreten Kernel abgebildet werden kann. Das rechte Bild verwendet eine Standardabweichung von 12, die einen großen diskreten Filterkernel voraussetzt. Man kann deutlich erkennen, dass die flache Wand beim kleinen Filterkernel deutliche „Beulen“ aufweist, die durch den größeren Kernel weitestgehend entfernt werden.

Nachfolgend soll erklärt werden, wie das Guided Image-Filtering nach He et al. [HST10] funktioniert und wie durch eine einfache Erweiterung kritische kontrastarme Fälle im Guidance-Image behandelt werden können.

7.2.2 Guided Image-Filtering

Beim Guided Image-Filtering von He et al. [HST10] ergibt sich das fertig gefilterte Bild q aus einem linearen Zusammenhang zum Guidance-Image I. Dieser Zusammenhang kann in einem Kernel-Fenster ω_k um den Pixel k wie folgt formuliert werden:

$$q_i = a_k I_i + b_k, \quad \forall i \in \omega_k. \tag{7.1}$$

a_k und b_k sind lineare Koeffizienten, die innerhalb des Fensters ω_k konstant sind. Um diese Koeffizienten zu berechnen, wird eine Kostenfunktion E gebildet, die die Differenzen zwischen dem Eingangsbild p und dem Ausgangsbild q abschätzt. Diese Kostenfunktion kann durch die Bildung der Fehlerquadrate wie folgt formuliert werden:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \varepsilon a_k^2). \tag{7.2}$$

Die Gleichung entspricht im Wesentlichen der Kostenfunktion zur Berechnung einer Regressionsgeraden, bei der p_i die Stützwerte zur Findung dieser Geraden bilden. Der Term εa_k^2 sorgt durch den Regulierungsparameter ε dafür, dass a_k nicht zu groß wird. Wie dieser Parameter zu deuten ist, wird später noch erläutert. Die Minimierung der Kostenfunktion E liefert die Gleichungen für die Koeffizienten (Prinzip der linearen Regression):

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \varepsilon}, \qquad b_k = \bar{p}_k - a_k \mu_k. \tag{7.3}$$

In diesen Gleichungen bezeichnet $|\omega|$ die Anzahl der Pixel im Kernel-Fenster und σ_k^2 bzw. μ_k entsprechen der Varianz bzw. dem Mittelwert im Fenster ω_k des Guidance-Images I. $\bar{p}_k$ bezeichnet dagegen den Mittelwert des Eingangsbildes in ω_k.

Nach der Bestimmung der Koeffizienten können diese in die lineare Gleichung (7.1) eingesetzt werden. Jedoch muss bedacht werden, dass der resultierende Pixel q_i im Ausgangsbild nicht für jedes Fenster konstant ist, denn für verschiedene Fenster ω_k resultieren verschiedene Werte für den Ausgangspixel q_i. Deshalb wird eine abschließende Mittelung über alle Werte von q_i vorgenommen, die zu der finalen Form für die Berechnung eines Ausgangspixels führt:

$$q_i = \frac{1}{|\omega|} \sum_{k: i \in \omega_k} (a_k I_i + b_k) = \bar{a}_i I_i + \bar{b}_i. \tag{7.4}$$

Um zu verstehen, wie bei dieser Art der Filterung die Kantenerhaltung funktioniert, wird der Fall betrachtet, bei dem das Guidance-Image I gleich dem zu filternden Eingangsbild p ist. Setzt man für diesen Fall den Regulierungsparameter ε gleich null, so folgt $a_k = 1$ und $b_k = 0$ (denn es gilt $\frac{1}{|\omega|}\sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k = \frac{1}{|\omega|}\sum_{i \in \omega_k} I_i^2 + \mu_k^2 = \sigma_k^2$). Daraus folgt wiederum, dass das Ausgangsbild

gleich dem Guidance-Image ist. Wenn man nun $\varepsilon > 0$ wählt, können zwei Extremfälle betrachtet werden:

1. Das Guidance-Image I ist konstant im Fenster ω_k. Hieraus folgt, dass a_k zu null und b_k zu $\bar{p}_k$ tendiert. Also wird das Ausgangsbild stark geglättet.
2. Das Guidance-Image I besitzt eine hohe Varianz im Fenster ω_k. In diesem Fall tendiert a_k zu eins und b_k zu null. Also wird das Ausgangsbild nicht geglättet und es gilt $q \approx I$.

Durch diese beiden Fälle und die Gleichung (7.3) kann man erkennen, dass der Regulierungsparameter ε wie eine Art Grenzwert funktioniert, der steuert, was als hohe und was als niedrige Varianz angesehen werden kann. Die Erhöhung von ε führt dazu, dass mehr Bereiche des Eingangsbildes geglättet werden, während die Verringerung dafür sorgt, dass weniger Bereiche geglättet werden.

Dass die Laufzeitkomplexität des Guided Image-Filterings unabhängig von der Kernelgröße ist, hängt damit zusammen, dass eine einfache Implementierung für die Berechnung des gleitenden Mittelwerts und somit auch für die Varianz existiert, die eine Laufzeitkomplexität von $O(N)$ besitzt, wobei N die Anzahl der Pixel bezeichnet. Der komplette GIF-Algorithmus lässt sich wie folgt formulieren:

Listing 7.1: Guided-Image-Filtering-Algorithmus.

```
D      = Tiefenbild auf Größe des RGB-Bilds skaliert (Nearest Neighbor).
I      = RGB-Bild in Graustufen.
D̄      = gleitender Mittelwert von D.
Ī      = gleitender Mittelwert von I.
I_v    = Varianz von I.
for each pixel: ID = I*D.
ID̄     = gleitender Mittelwert von ID.
for each pixel: A = (ID̄ - D* Ī) / ( I_v+ε).
for each pixel: B = D̄ - A*Ī.
Ā      = gleitender Mittelwert von A.
B̄      = gleitender Mittelwert von B.
for each pixel: Q = Ā*I+B̄.
```

Das GIF lässt sich relativ einfach für die Grafikkarte implementieren, für Bildauflösungen bis 640x480 ist aber eine Implementierung für Multicore-CPUs ausreichend, um ca. 30 Bilder pro Sekunde filtern zu können.

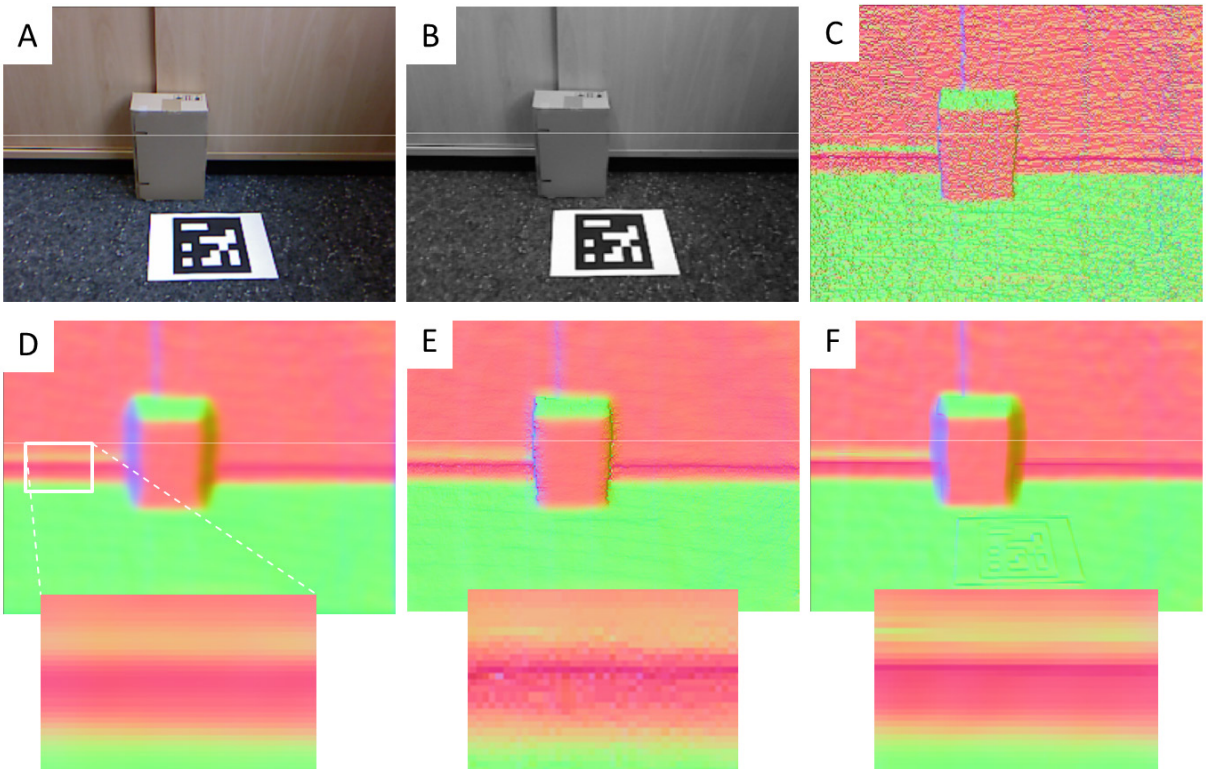

Abbildung 7.4: Verschiedene Filterungsmethoden für das Tiefenbild. (A) Eingangsfarbbild, (B) Eingangsfarbbild in Graustufen konvertiert, (C) ungefiltertes Tiefenbild, (D) Tiefenbild mit Gaussian-Blur-Approximation [Der93] gefiltert, (E) Tiefenbild bilateral gefiltert, (F) Tiefenbildfilterung durch GIF. Die Kolorierung der Tiefenbilder erfolgte nach den Normalen der Oberflächen. Man kann in den Nahaufnahmen erkennen, dass der Gaussian-Blur-Ansatz (D) filigrane Details aus der Tiefenabbildung entfernt, während diese bei der bilateralen Filterung (E) erhalten bleiben. Nach der bilateralen Filterung verbleiben im Bild jedoch deutliche Rauschanteile, die beim GIF (F) entfernt werden. Allerdings ist für das GIF (F) ebenfalls zu erkennen, dass die Silhouette des Kartons zu stark geglättet wird und dass der schwarz-weiße Marker auf dem Boden ein Relief im Tiefenbild erzeugt.

Abbildung 7.4 zeigt eine Szene, die mit dem GIF-Ansatz gefiltert wurde, bei der als Guidance-Image das RGB-Bild des Sensors genutzt wurde. In der Abbildung sind ebenfalls Beispiele für eine bilaterale und eine Gaussian-Blur-Filterung nach Deriche [Der93] zu finden. Die Abbildungen lassen erkennen, dass der GIF-Ansatz Kanten besser abbildet als der Gaussian-Blur-Ansatz. Man kann aber ebenfalls feststellen, dass kontrastarme Bereiche im Guidance-Image zu einer unerwünscht starken Glättung im Tiefenbild führen (siehe die Kontur des Kartons). Diese fehlerhafte Glättung wird dann besonders offensichtlich, wenn das Tiefenbild in diesen Bereichen besonders kontrastreich ist. Im nächsten Kapitel wird gezeigt, wie dieser Fehler durch eine Anpassung des Guidance-Images reduziert werden kann. Man erkennt in Abbildung 7.4 auch, dass kontrastreiche Kanten im Guidance-Image dazu führen, dass sich auf ebenen Flächen im Tiefenbild Reliefs abzeichnen. Dies kann man deutlich am schwarz-weißen AR-Marker im Bild erkennen. Diese fehlerhaften Reliefs sind jedoch in den meisten Fällen für den Betrachter nicht offensichtlich, da sie auf kontrastreichen Kanten im finalen Bild liegen.

7.2.3 Modifikation des Guidance-Image

Im vorherigen Abschnitt wurde bereits darauf hingewiesen, dass eine kontrastreiche Kante im Tiefenbild durch das GIF fälschlicherweise geglättet werden kann, wenn die korrespondierende Kante im Farbbild kontrastarm ist. Dieser Fall ist in Abbildung 7.4 dargestellt: Nach der Wandlung des Farbbilds in Graustufen ist der Übergang zum Karton relativ kontrastarm, was zu der starken Glättung der Kartonsilhouette führt. Diese falsche Glättung wird erst dann zu einem offensichtlichen Problem,

wenn der Kontrast im Tiefenbild entsprechend hoch ist. Um diese (zu) starke Glättung zu vermeiden, kann man das Guidance-Image anpassen, indem sehr kontrastreiche Kanten aus dem Tiefenbild auf das Guidance-Image übertragen werden. Diese Modifikation wird durch Addition eines Modifikationsbilds M ausgedrückt:

$$I_{new} = I + M. \tag{7.5}$$

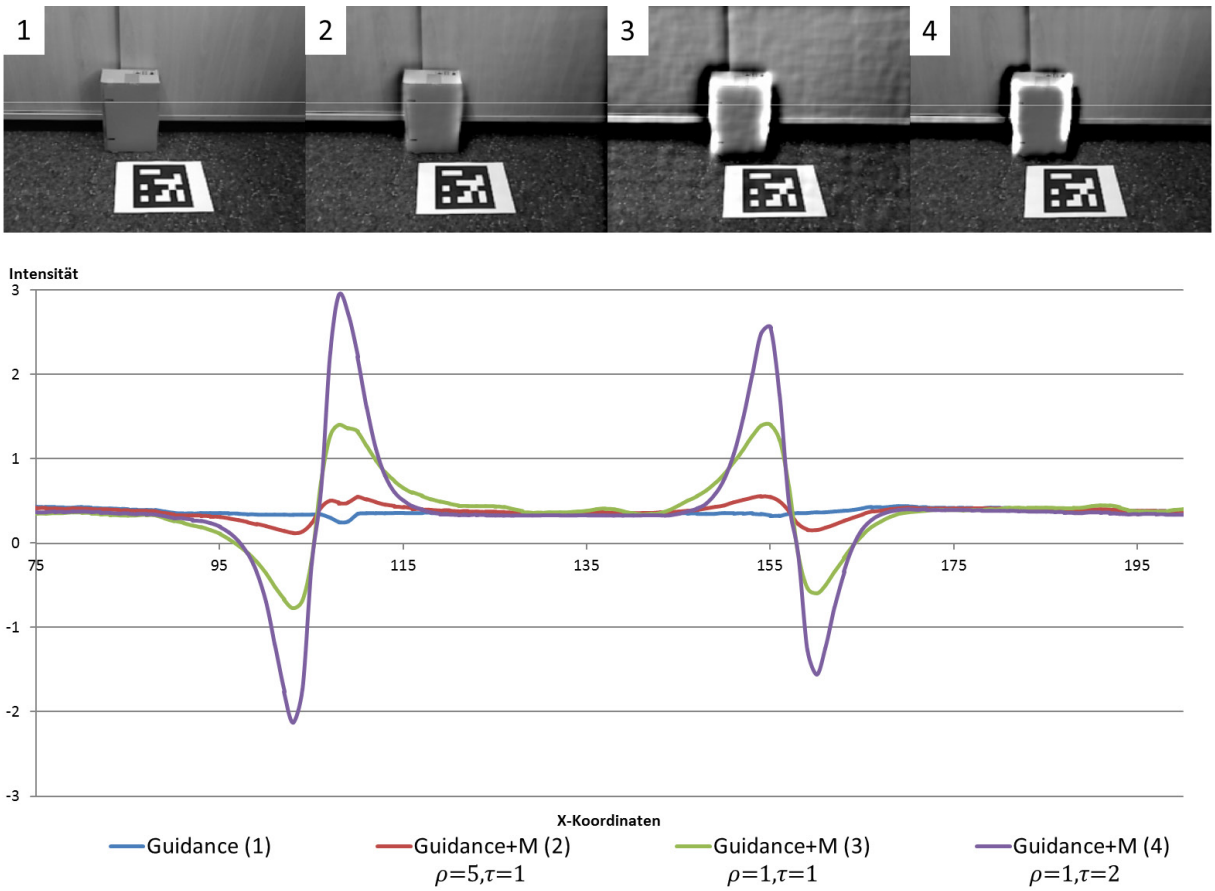

Abbildung 7.5: Modifiziertes Guidance-Image. Die Kurve im unteren Teil des Bildes zeigt die Intensitäten jeweils für eine Zeile mittig im Bild an (weiße Linie). (1) zeigt das unveränderte Guidance-Image, (2) zeigt das modifizierte Guidance-Image mit $\rho = 5$ und $\tau = 1$, (3) mit $\rho = 1$ und $\tau = 1$, (4) mit $\rho = 1$ und $\tau = 2$. In den oberen Bildern kann man deutlich erkennen, wie kontrastreiche Kanten im Tiefenbild zu kontrastreichen Kanten im modifizierten Guidance-Image werden.

Das Modifikationsbild soll starke Kontraste im Tiefenbild betonen. Diese lassen sich durch die Bildung der zweiten Ableitung hervorheben, bei der diese Kontrastkanten zu Nulldurchgängen führen. Um diese Nulldurchgänge kommt es zu starken Ausschlägen, die wiederum den Übergang der Kante kontrastreich hervorheben (siehe Abbildung 7.5). Demnach lässt sich das Modifikationsbild M aus der zweiten Ableitung des Tiefenbilds gewinnen:

$$M(x,y) = \frac{1}{\rho}\frac{|\nabla^2 D(x,y)|^\tau}{\nabla^2 D(x,y)}\nabla^2 D(x,y). \tag{7.6}$$

ρ und τ sind Parameter, die zur Kontraststeuerung genutzt werden. In Abbildung 7.5 wird gezeigt, wie diese sich auf das Bild auswirken. $D(x,y)$ liefert einen Tiefenwert für den Pixel auf Position (x,y). Die Bildung der zweiten Ableitung des Tiefenbildes lässt sich einfach approximieren, indem man das Tiefenbild mit dem approximativen Gaussian-Blur-Ansatz von Deriche [Der93] glättet und anschlie-

ßend zweimal nacheinander das Differenzbild bestimmt. Durch den Ansatz von Deriche kann die Glättung ebenfalls in $O(n)$ durchgeführt werden.

Die Anpassung der Parameter ρ und τ muss für jede Szene von Hand vorgenommen werden, um gute Ergebnisse zu erhalten. Da nur deutliche Kanten im Tiefenbild das Guidance-Image modifizieren, wird kaum Rauschen aus dem Tiefenbild in das Guidance-Image übernommen (siehe Abbildung 7.6). Allerdings kann es passieren, dass die Kanten im Tiefenbild die Objektsilhouette nicht korrekt abbilden. In solchen Fällen wird eine leicht verschobene Kante hervorgehoben.

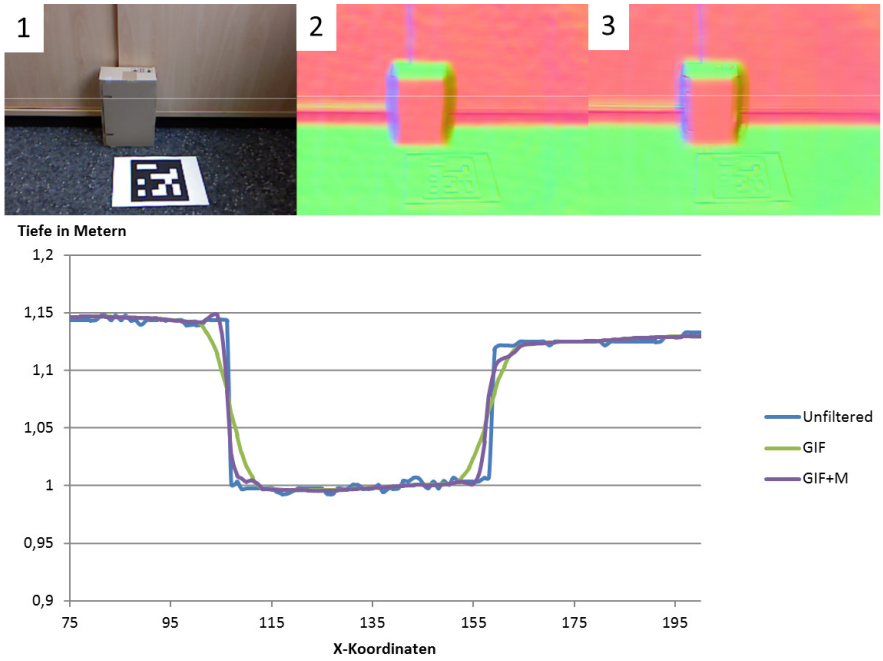

Abbildung 7.6: Vergleich GIF ohne Modifizierung (2) und mit Modifizierung (3). Das Diagramm zeigt den Querschnitt der Intensitäten für eine Zeile. Man kann erkennen, dass durch das Modifikationsbild der Übergang an der Silhouette des Kartons steiler wird. Allerdings entstehen auch leichte Artefakte durch Ungenauigkeiten im Tiefenbild.

7.2.4 Diskussion

Das hier vorgestellte Verfahren zur Verbesserung der Qualität des Tiefenbilds löst einige Probleme, die durch die Verwendung eines handelsüblichen RGB-D-Sensors für die Szenenrekonstruktion entstehen können. Dies erlaubt die Nutzung des Tiefenbilds für globale Beleuchtungsansätze. Die Qualität der Filterung hängt mit der Qualität des Sensors zusammen, doch es ist zu erwarten, dass die Tiefenbilder von zukünftigen Sensoren ebenfalls von der hier vorgestellten Filterung profitieren. Dies gilt insbesondere, wenn die Auflösung des RGB-Bildes höher ist als die des Tiefenbilds und wenn darüber hinaus das RGB-Bild stabilere Werte liefert. Liu et al. [LGL12] haben zeitgleich zur Veröffentlichung des Autors dieser Arbeit [LB12] das GIF zur Tiefenbildfilterung vorgeschlagen, was als Indiz dafür gesehen werden kann, dass dieser Ansatz für dieses Aufgabenfeld besonders geeignet ist.

Ein großer Nachteil des vorgestellten Ansatzes besteht darin, dass nur ein Tiefen- und Farbbild zur Repräsentation der kompletten realen Szene genutzt wird. Dies führt zu Einschränkungen bei der Simulation der globalen Beleuchtung, die im nächsten Kapitel erläutert werden.

7.3 Virtuelle Beleuchtung

In diesem Kapitel wird beschrieben, wie die globale Beleuchtung für eine AR-Anwendung mit Hilfe des LightSkin-Verfahrens simuliert werden kann. Dabei unterstützt das Verfahren den Lichttransport von der realen Szene in die virtuelle und umgekehrt. Wegen der unvollständigen Repräsentation der realen Szene müssen jedoch einige Einschränkungen formuliert werden:

1. **Nur sichtbare reale Oberflächen reflektieren indirektes Licht in die virtuelle Szene**. Da dem System nur sichtbare reale Oberflächen bekannt sind, können auch nur diese für die Reflexionen berücksichtigt werden. Virtuelle Objekte können dagegen Licht von allen Oberflächenpunkten in die virtuelle und reale Szene reflektieren.
2. **Sämtliche Oberflächen der realen Szene werden als ideal diffuse Reflektoren angenommen**. Für jeden sichtbaren Oberflächenpunkt ist nur dessen Pixelfarbe, Tiefe und Normale bekannt. Geht man davon aus, dass ein Oberflächenpunkt ideal diffus reflektiert, kann man den reflektierten Lichtstrom des Punkts anhand der Pixelfarbe approximieren. In dieser Arbeit bleiben die Reflexionseigenschaften der realen Oberflächen also weitestgehend unberücksichtigt.
3. **Reale Oberflächen können keinen Schatten auf virtuelle Objekte werfen**. Die Tiefe und Breite eines realen Objekts kann aus einer einzigen Perspektive heraus nicht ausreichend approximiert werden. Eine reine Verwendung der sichtbaren Oberflächenpunkte zur Schattierung führt jedoch zu deutlich falschen Schatten, die dem Betrachter ins Auge fallen.

Darüber hinaus setzt die Berechnung der globalen Beleuchtung mit Hilfe des LightSkin-Verfahrens voraus, dass Caches im Modellraum verteilt wurden, für die dann Proxy-Lichtquellen berechnet werden. Während diese Caches für die virtuellen Objekte vorhanden sind, fehlen sie für den realen Teil der Szene, der durch das Farb- und das Tiefenbild des RGB-D-Sensors repräsentiert wird. Da sich das Tiefenbild von Frame zu Frame verändern kann, können keine Caches im Voraus verteilt werden. Deshalb wird in Kapitel 7.3.1 ein Algorithmus zur Verteilung der Caches im Bildraum des Sensors (zur Laufzeit der Simulation) vorgestellt. Die adaptive Verteilung führt dazu, dass reale Oberflächen nur diffuses Licht reflektieren können, da hochfrequente Reflexionen durch die ständig variierende Verteilung der Caches bei Kamerabewegungen zu flackernden Artefakten führen würden (dieses Phänomen wurde bereits in Kapitel 4.4.2 diskutiert).

Abbildung 7.7 zeigt in einer Übersicht, welche Komponenten des LightSkin-Verfahrens prinzipiell erweitert bzw. angepasst werden müssen, damit dieses in einer dynamischen Augmented-Reality-Anwendung eingesetzt werden kann.

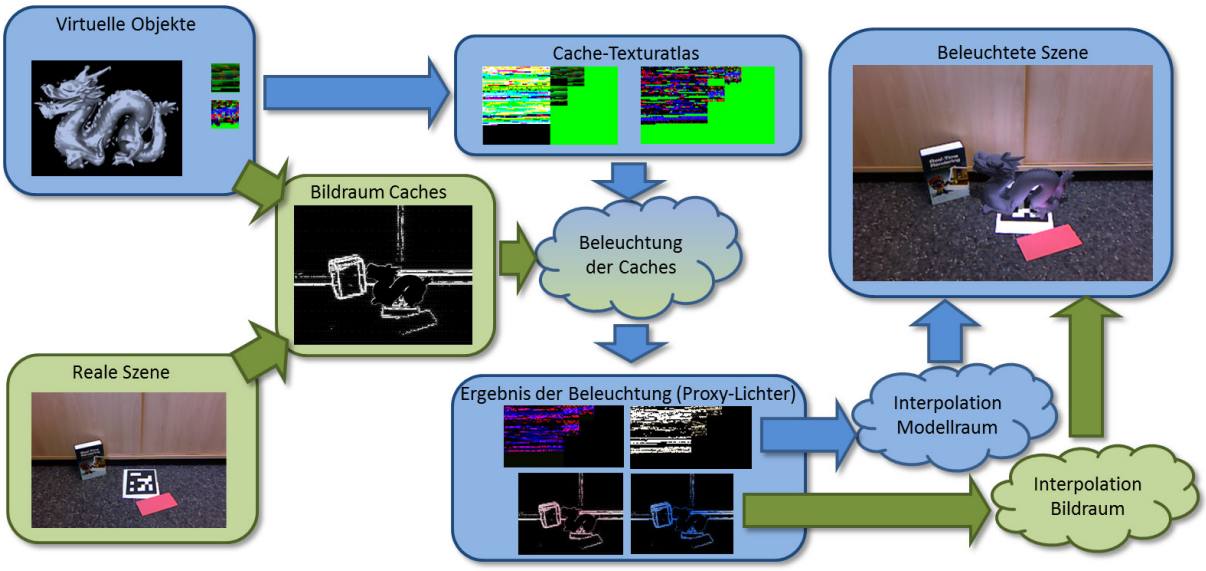

Abbildung 7.7: Prinzipielle Erweiterung des LightSkin-Verfahrens für AR-Anwendungen. Die grünen Komponenten stellen Erweiterungen dar. Zusätzlich zum Cache-Texturatlas, der alle Caches der virtuellen Objekte im Modellraum enthält, müssen Cache-Texturen im Bildraum für die reale Szene berechnet werden. Das Prinzip für die Beleuchtung der Caches bleibt erhalten, allerdings muss die Erzeugung der VALs und deren Anwendung auf das Differential Rendering angepasst werden. Da die Caches im Bildraum verteilt werden, kann die Interpolation der Cache-Ergebnisse ebenfalls nur im Bildraum erfolgen.

Im Wesentlichen müssen drei Erweiterungen bzw. Veränderungen vorgenommen werden:

1. **Erzeugung von Bildraum-Caches für die reale Szene**: Die Verteilung der Caches kann nur im Bildraum erfolgen, da nur Oberflächendaten im Bildraum vorhanden sind. Für die Verteilung der Caches wird der Multiresolution-Splatting-Ansatz von Nichols et al. [NW09] genutzt bzw. angepasst.
2. **Anpassung der Beleuchtung der Caches**: Bei der Beleuchtung der Caches müssen die Prinzipien des Differential Renderings berücksichtigt werden. Darüber hinaus muss erläutert werden, wie die primären Lichtquellen und VALs in einer AR-Anwendung erzeugt werden können.
3. **Interpolation der Bildraum-Caches**: Da die Caches im Bildraum verteilt werden, müssen diese auch im Bildraum interpoliert werden. Hierfür wird eine geeignete Implementierung vorgeschlagen.

In den folgenden drei Unterkapiteln werden die einzelnen Erweiterungen im Detail erläutert.

7.3.1 Erzeugung der Caches im Bildraum

Das LightSkin-Verfahren ist nur deshalb echtzeitfähig, weil es nicht jeden sichtbaren Oberflächenpunkt für die indirekte Beleuchtung durch die VALs berücksichtigt. Stattdessen werden die VALs nur auf eine geringe Menge von Caches angewendet. Mit den Proxy-Lichtquellen dieser Caches können dann, unter Anwendung eines Interpolationsverfahrens, die restlichen sichtbaren Oberflächenpunkte beleuchtet werden. Die Caches werden hierfür im Modellraum verteilt. Für das Interpolationsverfahren ist also für jeden Oberflächenpunkt die Lage der Caches auf dem Modell bekannt. Dieses Prinzip funktioniert für die gewählte Modellrepräsentation der realen Szene nicht. Die reale Szene wird in dieser Arbeit durch drei G-Buffer repräsentiert. Ein G-Buffer enthält die diffusen Reflexionen (RGB-Bild des Sensors), einer das Tiefenbild (gefiltert) und einer die Oberflächennormalen (die aus dem

Tiefenbild berechnet wurden). Eine Unterscheidung zwischen einzelnen Modellen, die sich in diesen G-Buffern befinden, ist nicht mehr möglich, weswegen eine Verankerung der Caches auf Modelloberflächen nicht vorgenommen werden kann.

Grundsätzlich ist es aber möglich, vereinzelte Pixel aus den G-Buffern als Caches zu markieren, indem man das gleiche Prinzip wie bereits beim Culling der Modellraum-Caches anwendet (siehe Kapitel 6.1.5). Die Oberflächenpunkte in den G-Buffern, die zu Caches werden sollen, können durch eine Z-Buffer-Maske ausmaskiert werden. Dabei erhalten die Cache-Oberflächenpunkte einen hohen Z-Wert, während die restlichen Oberflächenpunkte einen Z-Wert von null erhalten. Da die Anwendung der VALs in einem Fragment-Shader durchgeführt wird, kann so sichergestellt werden, dass der Fragment-Shader nur für Pixel ausgeführt wird, die einen hohen Z-Wert besitzen. Es wird also wieder das Prinzip des Z-Buffers ausgenutzt. Der Vorteil dieses Ansatzes ist, dass zwischen der Beleuchtung des Cache-Texturatlas (für Modellraum-Caches) und der Beleuchtung der Bildraum-Caches der realen Szene kein Unterschied gemacht werden muss. Bei beiden Ansätzen werden die Caches in Cache-Texturen gespeichert, die vom Beleuchtungsverfahren gleich behandelt werden können.

Ein naiver Ansatz zur Bildraum-Cache-Verteilung würde darin bestehen, eine fixe, zweidimensionale Verteilung (Z-Maske) für die G-Buffer zu wählen und diese für die komplette Simulation unverändert zu lassen. Wenn die Caches dann durch die VALs beleuchtet wurden, können deren Ergebnisse durch einen einfachen GPU-Pull-Push-Algorithmus [SKE06] auf die restlichen Pixel der G-Buffer interpoliert werden. Dieser Ansatz würde keine permanente Neuverteilung der Caches erfordern und wäre demnach effizient. Allerdings führt dieser Ansatz zu einem Problem, wenn die Kamera oder die reale Szene verändert werden. Vollzieht beispielsweise die Kamera eine Seitwärtsbewegung, so sorgt die statische Verteilung der Caches im Bildraum dafür, dass die Caches über die Oberflächen der realen Szene wandern. Wenn diese Oberflächen Diskontinuitäten in der Tiefe oder in den Normalen aufweisen, verändern sich die Eigenschaften der Caches sprunghaft (Position und Normale). Diese sprunghafte Änderung in den Cache-Eigenschaften führt zu deutlich sichtbarem Flackern bei der Interpolation, da sich die resultierenden Proxy-Lichtquellen für die Caches ebenfalls sprunghaft verändern. Das Problem wird in Abbildung 7.8 veranschaulicht. Eine feste Verteilung der Caches im Bildraum, die a priori entschieden wird, wird also zu deutlichen temporären Inkohärenzen bei Animationen führen und kann deshalb nicht sinnvoll eingesetzt werden.

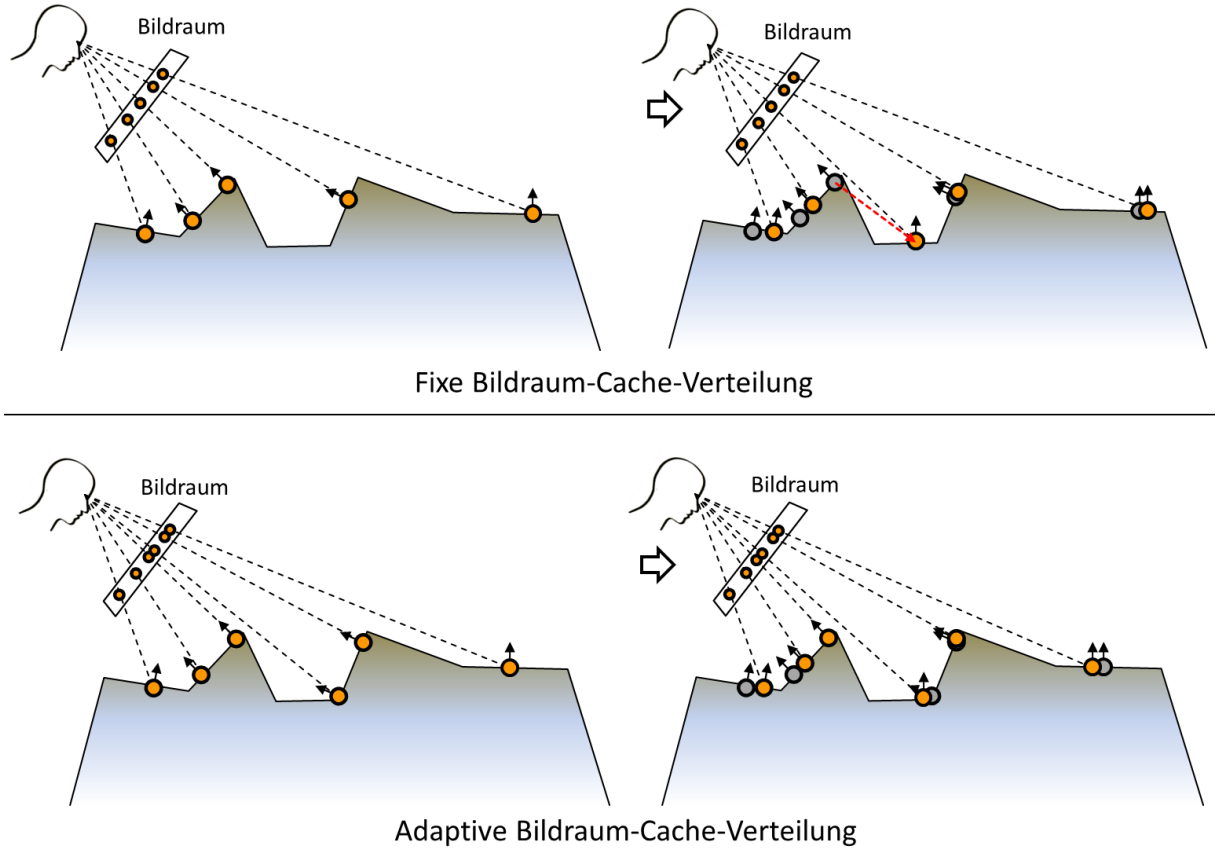

Abbildung 7.8: Fixe vs. adaptive Bildraum-Cache-Verteilung. In der oberen Hälfte des Bildes wird von einer fixen Verteilung der Caches im Bildraum ausgegangen. Bewegt sich der Betrachter bzw. der Sensor ein wenig nach rechts (rechtes Bild), können Diskontinuitäten in der Szene dazu führen, dass Caches große Veränderungen erfahren (rote Linie). Diese großen Veränderungen führen zu deutlichen temporären Inkohärenzen bei der indirekten Beleuchtung. In der unteren Bildhälfte wird eine adaptive Bildraum-Cache-Verteilung gezeigt, die mehr Caches in Bereichen des Bildes verteilt, in denen hohe Diskontinuitäten auftreten. Man erkennt, dass durch diesen Ansatz die Caches keine großen Sprünge vollziehen.

Das Problem bei der fixen Verteilung der Caches im Bildraum liegt in der sprunghaften Änderung der Attribute der Caches bei Animationen. Diese Veränderungen werden durch Diskontinuitäten in den G-Buffern hervorgerufen. Diese Diskontinuitäten können aber nicht einfach entfernt werden, da beispielsweise das Weichzeichnen der G-Buffer zu einer Fehlinterpretation der realen Szene führt. Gerade im Kontext der in Kapitel 7.2.2 vorgestellten kantenerhaltenden Filterung des Tiefenbilds würde die Weichzeichnung sämtliche Verbesserungen revidieren. Die Lösung für dieses Problem besteht in einer adaptiven Verteilung der Caches, die dafür sorgt, dass in Bereichen, in denen hohe Diskontinuitäten auftreten, mehr Caches verteilt werden als in homogenen Bereichen. Ob ein Bereich des G-Buffers hohe Diskontinuitäten aufweist, kann anhand der Variation der Normalen und der Tiefe bestimmt werden. Kommt es beispielsweise zu einem Sprung in der Tiefe, so wird ein Cache vor dem Sprung und einer hinter dem Sprung platziert (siehe Abbildung 7.8). Um eine solche adaptive Verteilung der Caches anhand der Diskontinuitäten effizient durch eine GPU-Implementierung vorzunehmen, kann man den Multiresolution-Splatting-Algorithmus von Nichols et al. [NW09, NSW09] modifizieren (siehe auch Kapitel 4.4.2). Der Ansatz zur Verteilung lässt sich in drei konsekutive Teilschritte zerlegen (siehe Abbildung 7.9 (Schritte 1, 2, 3)):

1. Schritt: Erzeugung einer Diskontinuitätsbildpyramide (Mip-Maps).
2. Schritt: Generierung eines Ebenenatlas auf Basis der Diskontinuitätsbildpyramide.

3. Schritt: Erzeugung der Z-Buffer-Maske auf Basis des Ebenenatlas.

Für die Erzeugung der Z-Buffer-Maske ist es wichtig, dass die G-Buffer ebenfalls die augmentierten Objekte beinhalten, da auf deren Silhouette Bildraum-Caches verteilt werden müssen. Deshalb werden die virtuellen Objekte für die Bildraum-Cache-Verteilung berücksichtigt, anschließend werden jedoch die Caches, deren Pixel auf einem virtuellen Objekt liegen, von der indirekten Beleuchtung wieder ausgeschlossen (siehe Abbildung 7.9 (4)). Dieser Ausschluss ist nötig, weil sonst virtuelle Objekte sowohl durch Bildraum-Caches als auch durch Modellraum-Caches indirekt beleuchtet werden würden.

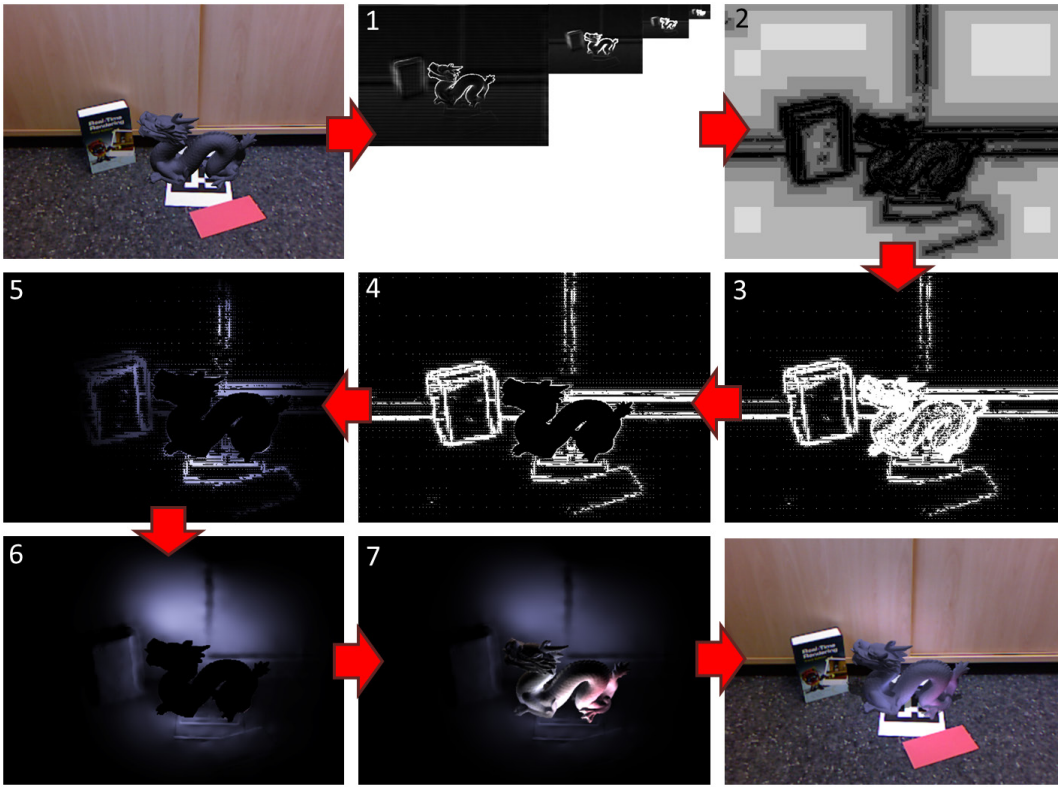

Abbildung 7.9: Die einzelnen Verarbeitungsschritte für die reale Szene. (1) Erzeugung der Diskontinuitätsbildpyramide, (2) Generierung des Ebenenatlas, (3) Erzeugung der Z-Buffer-Maske, (4) Entfernung der Bildraum-Caches, die auf virtuellen Oberflächen liegen, (5) Erzeugung der Proxy-Lichtquellen für Bildraum-Caches, (6) Bildraum-Interpolation der indirekten Beleuchtung, (7) indirekte Beleuchtung für virtuelle Objekte wird hinzugenommen.

Schritt 1: Erzeugung einer Diskontinuitätsbildpyramide (Mip-Maps):
Die Bildpyramide besteht aus mehreren Ebenen. Die niedrigste Ebene ist die größte Ebene und hat eine Auflösung, die der Hälfte der G-Buffer-Auflösung entspricht. Mit jeder nächsthöheren Ebene wird die Auflösung halbiert. Ein Pixel in einer Ebene speichert zwei Fließkommawerte. Der erste Fließkommawert speichert die maximale Differenz in der Tiefe und der zweite Wert speichert die maximale Abweichung der Normalen. Die Berechnung dieser Werte erfolgt durch den folgenden Algorithmus:

Listing 7.2: Algorithmus zur Erzeugung der Diskontinuitätsbildpyramide.

```
for each Fragment in Ebene i
  // Maximale Tiefendifferenz bestimmen.
  1. Lese die Tiefenwerte d_j für vier Texel aus Ebene (i − 1), die von diesem Fragment abgedeckt
     werden.
  2. Bestimme für jeden Texel j die maximale Tiefendifferenz zu seinen acht Nachbartexeln.
  3. Speicher die maximale Tiefendifferenz.

  // Maximale Normalendifferenz bestimmen.
  4. Lese die Normalen n_j für vier Texel aus Ebene (i − 1), die von diesem Fragment abgedeckt
     werden.
  5. Berechne für jeden Texel j das Skalarprodukt mit der gemittelten Normalen von den acht
     Nachbartexeln.
  6. Speicher das minimale Skalarprodukt.
end
```

Dieser Algorithmus wird für jede Ebene der Bildpyramide ausgeführt und sorgt dafür, dass jeder Texel jeder Ebene die maximale Tiefen- und Normalendifferenz enthält, die durch seine „Texelfläche“ abgedeckt wird. Dabei wird davon ausgegangen, dass jeder Texel der Ebene i vier Texel der Ebene $(i-1)$ abdeckt. Diese Diskontinuitätsbildpyramide wird im nächsten Schritt genutzt, um daraus einen Ebenenatlas zu erzeugen.

Schritt 2: Generierung des Ebenenatlas

Der Ebenenatlas hat die gleiche Auflösung wie die G-Buffer und speichert für jeden Pixel die korrespondierende Ebene aus der Diskontinuitätsbildpyramide. In welche Ebene ein Pixel sortiert wird, wird anhand von zwei nutzerdefinierten Grenzwerten bestimmt: der maximal erlaubten Tiefe T_d und der maximal erlaubten Normalenabweichung T_n. Für die Erzeugung des Atlas wird wieder ein Fragment-Shader genutzt, der folgende Operationen ausführt:

Listing 7.3: Algorithmus zur Erzeugung des Ebenenatlas.

```
for each Fragment in Ebenenatlas
  for each Ebene i = {1,..,N − 1} in Diskontinuitätspyramide
    Lese maximale Tiefendifferenz d_i und maximale Normalenabweichung n_i aus Ebene.
    if( d_i > T_d or n_i < cos(T_n))
      return i-1;
  end
end
```

Der sich ergebene Ebenenatlas ist in Abbildung 7.9 (2) dargestellt. Anhand dieses Atlas kann schlussendlich die Z-Buffer-Maske für die Caches erzeugt werden.

Schritt 3: Erzeugung der Z-Buffer-Maske

Durch die Z-Buffer-Maske wird bestimmt, welche Pixel der G-Buffer zu Caches werden. Grundsätzlich gilt, dass jeder Pixel, der einen Z-Wert größer null besitzt, einen Cache markiert. Die Z-Buffer-Maske wird auf Basis des Ebenenatlas erzeugt. Dabei kommt das Prinzip zur Anwendung, dass Bereiche des Bildes, die hohe Ebenen im Ebenenatlas enthalten, wenige Caches benötigen, während tiefe Ebenen hohe Diskontinuitäten kennzeichnen und deshalb mehrere Caches benötigen. Während für die Ebene Null für jeden Pixel ein Cache erzeugt wird, wird für die Ebene Eins nur jeder zweite Pixel als Cache markiert, bei der Ebene Zwei wird dann nur noch jeder vierte markiert und so weiter. Die Erzeugung der Z-Buffer-Maske kann mit dem folgenden Prinzip vorgenommen werden (siehe Abbildung 7.10):

Listing 7.4: Algorithmus zur Erzeugung der Z-Buffer-Maske.

```
for each Fragment in G-Buffer
  Lese Ebene von Ebenenatlas.
  Bestimme Cache-Abstand c_d für Ebene.
  if XY-Koordinate vom Fragment Vielfaches von c_d
    return 1 (Z-Wert)
  else
    return 0 (Z-Wert)
  end
end
```

Nach der Durchführung der drei vorgestellten Verarbeitungsschritte ist die Maskierung der Bildraum-Caches abgeschlossen und die G-Buffer können zusammen mit der Maske für die indirekte Beleuchtung durch die VALs genutzt werden. Für die indirekte Beleuchtung müssen jedoch die Prinzipien des Differential Renderings [FGR92] berücksichtigt werden. Dies ist Gegenstand des nächsten Kapitels.

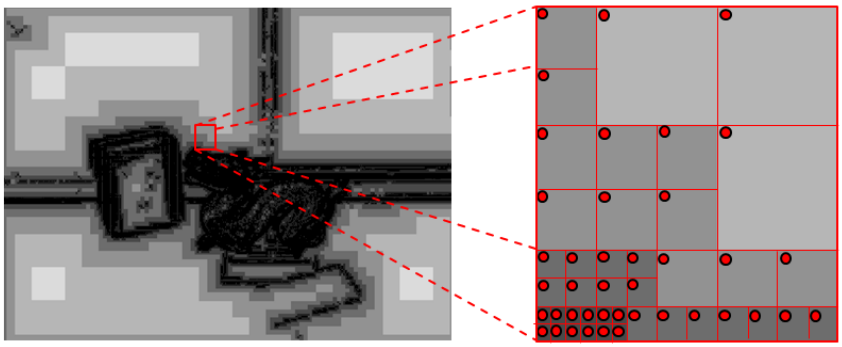

Abbildung 7.10: Erzeugung der Z-Buffer-Maske anhand des Ebenenatlas. Es werden die Pixel auf z=1 gesetzt (rote Punkte), die ein Vielfaches der Quadratseiten sind (in Pixelkoordinaten). Nur für diese maskierten Punkte werden die Proxy-Lichtquellen berechnet.

7.3.2 Beleuchtung der Caches

Die Beleuchtung der Caches geschieht im Wesentlichen durch die gleichen Algorithmen, die in Kapitel 6.1.3 vorgestellt wurden. Jedoch muss für die Simulation berücksichtigt werden, dass VALs aus der realen Szene nur auf Caches der virtuellen Szene angewendet werden, ansonsten würden reale Oberflächen doppelt indirekt beleuchtet (natürlich und synthetisch durch die VALs der realen Oberflächen). VALs, die dagegen auf Oberflächen virtueller Objekte liegen, können sowohl auf virtuelle als auch auf reale Objekte angewendet werden. Es muss also zwischen realen und virtuellen VALs unterschieden werden. Diese Unterscheidung ist bereits für die Erzeugung der VALs von Bedeutung. In den vorherigen Ausführungen wurden VALs durch den Einsatz von Reflective Shadow-Maps erzeugt. In diesen repräsentiert ein Pixel ein VAL. Für die realen Oberflächenpunkte bei einer AR-Anwendung wird nun davon ausgegangen, dass die G-Buffer aus Sicht des Sensors eine RSM bilden. Jeder Pixel des G-Buffers erzeugt also ein VAL. Aus Laufzeitgründen kann es aber erforderlich sein, dass nicht jeder Pixel ein VAL abbildet, sondern nur jeder zweite oder vierte. Bei einer Sensorauflösung von 640x480 Pixeln wurden mit 160x120 VALs gute Ergebnisse erzielt. Die Normalen und Positionen der VALs können direkt aus dem G-Buffer ausgelesen werden. Der Lichtstrom des VALs kann berechnet werden, wenn man die Farbe des Pixels aus Sicht des Sensors kennt und davon ausgeht, dass der Oberflächenpunkt zu dem Pixel ideal diffus reflektiert. Die Fläche des VALs ergibt sich aus den Projektionseigenschaften des Sensors und kann nach dem gleichen Prinzip berechnet werden, wie es in Kapitel 5.2.4 beschrieben wurde.

Die Erzeugung von VALs auf virtuellen Oberflächen setzt voraus, dass die direkten Lichtquellen in der realen Szene bekannt sind. Um diese Lichtquellen zu bestimmen, können verschiedene Ansätze gewählt werden, die exemplarisch in Kapitel 2.2.2 vorgestellt wurden. Es kann beispielsweise eine Weitwinkelkamera genutzt werden, die die obere Hemisphäre einer Szene aufzeichnet. Innerhalb des Bildes dieser Kamera werden dann die Bereiche ausgewählt, die eine besonders hohe Intensität aufweisen, um dort direkte Lichtquellen zu platzieren. Die Verteilung der primären Lichtquellen wird in dieser Dissertation nicht im Detail beschrieben, sie kann aber mit den gleichen Importance-Sampling Ansätzen durchgeführt werden, wie sie von Knecht et al. [KTM*10] oder Franke [Fra13a] vorgeschlagen wurden. Je nach Ausprägung der realen direkten Beleuchtung können 16 bis 64 direktionale primäre Lichtquellen verteilt werden (direktional, weil die Entfernung der Lichtquellen nicht bekannt ist). Jede Lichtquelle erzeugt dann eine RSM mit 1000 bis 4000 VALs. Für die Erzeugung der RSMs werden nur virtuelle Objekte berücksichtigt, so dass nur diese in die RSMs gezeichnet werden müssen.

7.3.3 Interpolation der Bildraum-Caches

Wenn alle VALs nach den Regeln des vorherigen Kapitels auf die Caches angewendet wurden, existieren für jeden Cache Proxy-Lichtquellen, die dessen indirekte Beleuchtung repräsentieren. Für die virtuellen Objekte kann durch die Interpolation der Proxy-Lichtquellen im Modellraum die durchgehende Reflexion für die Modelloberfläche berechnet werden (wie dies in Kapitel 6.1.4 beschrieben wurde). Für die Caches, die auf realen Oberflächen liegen (Bildraum-Caches), kann eine Interpolation im Modellraum nicht durchgeführt werden, stattdessen muss ein Interpolationsverfahren entworfen werden, das im Bildraum funktioniert. Hierfür lässt sich wieder der Multiresolution-Splatting-Ansatz von Nichols und Wyman [NW09] modifizieren. Das Verfahren von Nichols und Wyman interpoliert direkt die Farbkomponenten für die indirekten Reflexionen. Bei diesem Ansatz wird das finale Bild aus verschiedenen Mip-Map-Ebenen zusammengesetzt, bei denen nicht jeder Texel aus jeder Ebene einen Reflexionswert speichert. Ein Texel enthält nur dann einen Reflexionswert, wenn der Bildbereich, der durch diesen Texel abgedeckt wird, keine zu hohen Diskontinuitäten aufweist. Da die Mip-Map-Ebenen verschiedene Auflösungen besitzen, deckt ein Texel, der in einer höheren Mip-Map-Ebene liegt einen größeren Teil des Bildes ab, als ein Texel in einer niedrigeren Ebene. Für die Interpolation werden für jede Mip-Map-Ebene zwei Operationen nacheinander ausgeführt (die Ebenen werden hintereinander verarbeitet. Zurerst wird die gröbste Ebene, mit der geringsten Auflösung und danach die nächstfeinere Ebene verarbeitet, und so weiter):

1. Die Reflexionswerte in der Ebene werden geglättet. Diese Glättung wird durch einen gleitenden Mittelwert, mit einer Maske von 3x3 Texeln, realisiert. Bei der Mittelung werden nur die Texel berücksichtigt, die einen gültigen Reflexionswert aufweisen.
2. Nachdem die Glättung durchgeführt wurde, wird die Ebene auf die nächstfeinere Ebene hochskaliert. Bei der Hochskalierung wird eine lineare Interpolation durchgeführt, die durch einen Texturfilter realisiert werden kann. Anschließend wird das hochskalierte Bild zu der feineren Ebene addiert. Es werden aber nur Texel übernommen, die einen gültigen Reflexionswert enthalten.

Das Prinzip der Interpolation von Nichols und Wyman ist in Abbildung 7.11 dargestellt.

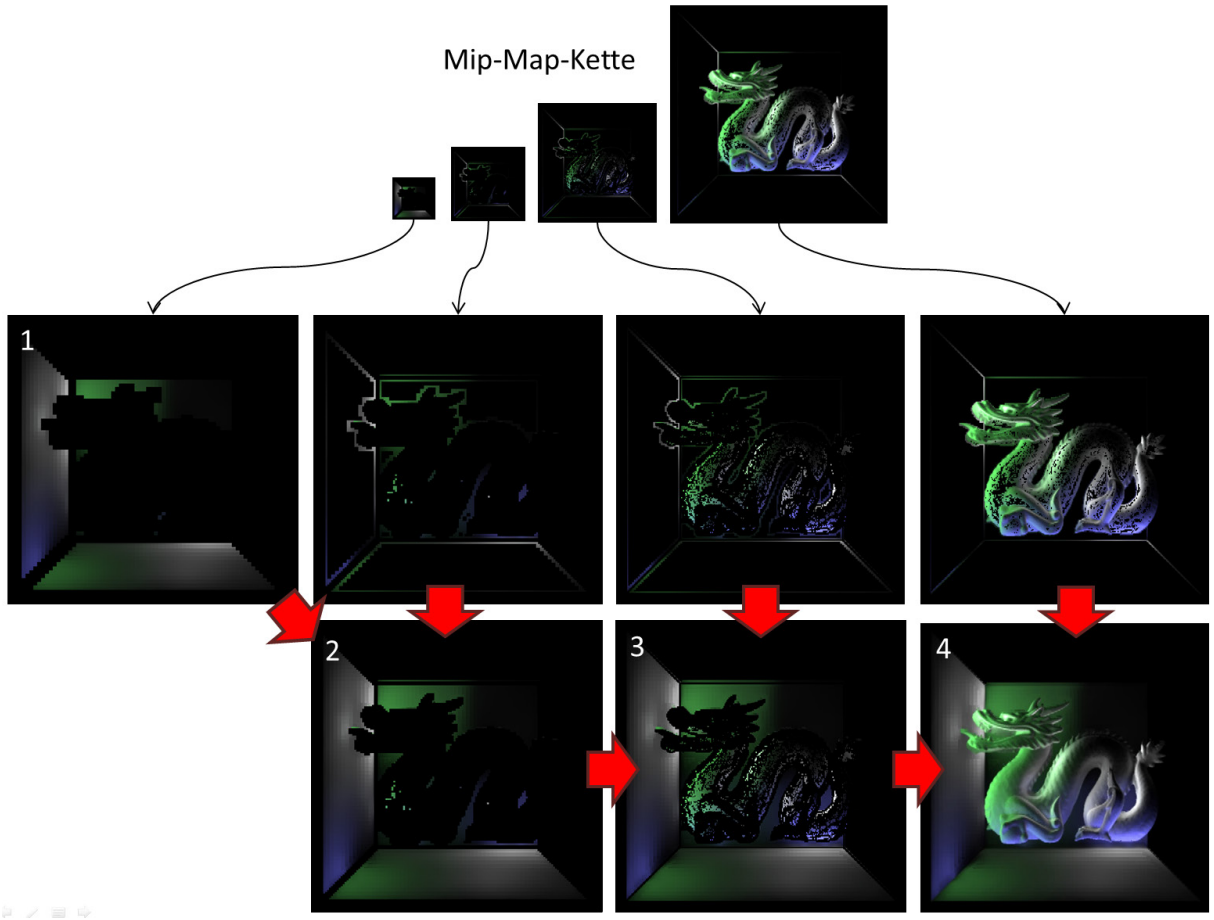

Abbildung 7.11: Interpolation der indirekten Beleuchtung nach Nichols und Wyman [NW09]. Als Eingang dient eine Mip-Map-Kette, die nach Diskontinuitäten verteilte Reflexionswerte enthält. Für die Interpolation wird von der gröbsten Mip-Map-Ebene zur feinsten das Bild geglättet, hochskaliert und zusammengefasst, bis ein durchgängiges Ergebnis in voller Auflösung erreicht wird.

Das Interpolationsverfahren von Nichols und Wyman kann nicht unmittelbar für das LightSkin-Verfahren genutzt werden. Anstatt der Farbkomponenten der indirekten Reflexionen werden beim LightSkin-Verfahren die Attribute der Proxy-Lichtquellen interpoliert. Da die Attribute der Proxy-Lichtquellen lose in den Pixeln des G-Buffers verteilt liegen und für diese keine Mip-Maps existieren (siehe Abbildung 7.9 (5)), muss das Verfahren von Nichols et al. so modifiziert werden, dass es ohne verschiedene Mip-Map-Ebenen funktioniert. Die lose Verteilung der Caches im Bildraum musste vorgenommen werden, damit die Erzeugung der Proxy-Lichtquellen effizient durchgeführt werden kann (Kapitel 6.1.3). Da die Verteilung der Caches im Bildraum auf Basis der Diskontinuitätsbildpyramide (Mip-Maps, siehe Kapitel 7.3.1) erfolgte, weist diese Verteilung jedoch strukturelle Eigenschaften auf, die ausgenutzt werden können, um ein Interpolationsverfahren zu entwerfen, das äquivalent zu dem von Nichols und Wyman ist. Hierfür wird zu Beginn der Simulation eine statische Z-Buffer-Maske erstellt, deren Struktur in Abbildung 7.12 gezeigt wird. Die Mip-Map-Struktur, die beim Ansatz von Nichols und Wyman vorausgesetzt wird, wird emuliert, indem ein bildraumweites Quad gezeichnet wird, das von vorne nach hinten durch die statische Z-Buffer-Maske bewegt wird. Das Glätten und Skalieren übernimmt hierbei ein Fragment-Shader, für den das Verhalten des Z-Buffers so eingestellt wird, dass nur die Fragmente aktualisiert werden, die einen niedrigeren Z-Wert besitzen, als der Z-

Wert des bildraumweiten Quads. Abbildung 7.12 zeigt das Prinzip. Der Interpolationsalgorithmus kann wie folgt zusammengefasst werden:

Listing 7.5: Algorithmus zur Interpolation der Bildraum-Caches.

```
for each Ebene i = {N − 1,..,0} in Z-Buffer-Maske
1. Setze bildraumweites Quad zwischen Ebene i und (i − 1).
2. Verwende Fragment-Shader, der den gleitenden Mittelwert berechnet (wie Nichols et al.).
3. Setze bildraumweites Quad zwischen Ebene (i − 1) und (i − 2).
4. Verwende Fragment-Shader, der die Ebene linear hochskaliert (wie Nichols et al.).
5. Verwende Fragment-Shader, der Ebene i mit (i − 1) kombiniert.
end
```

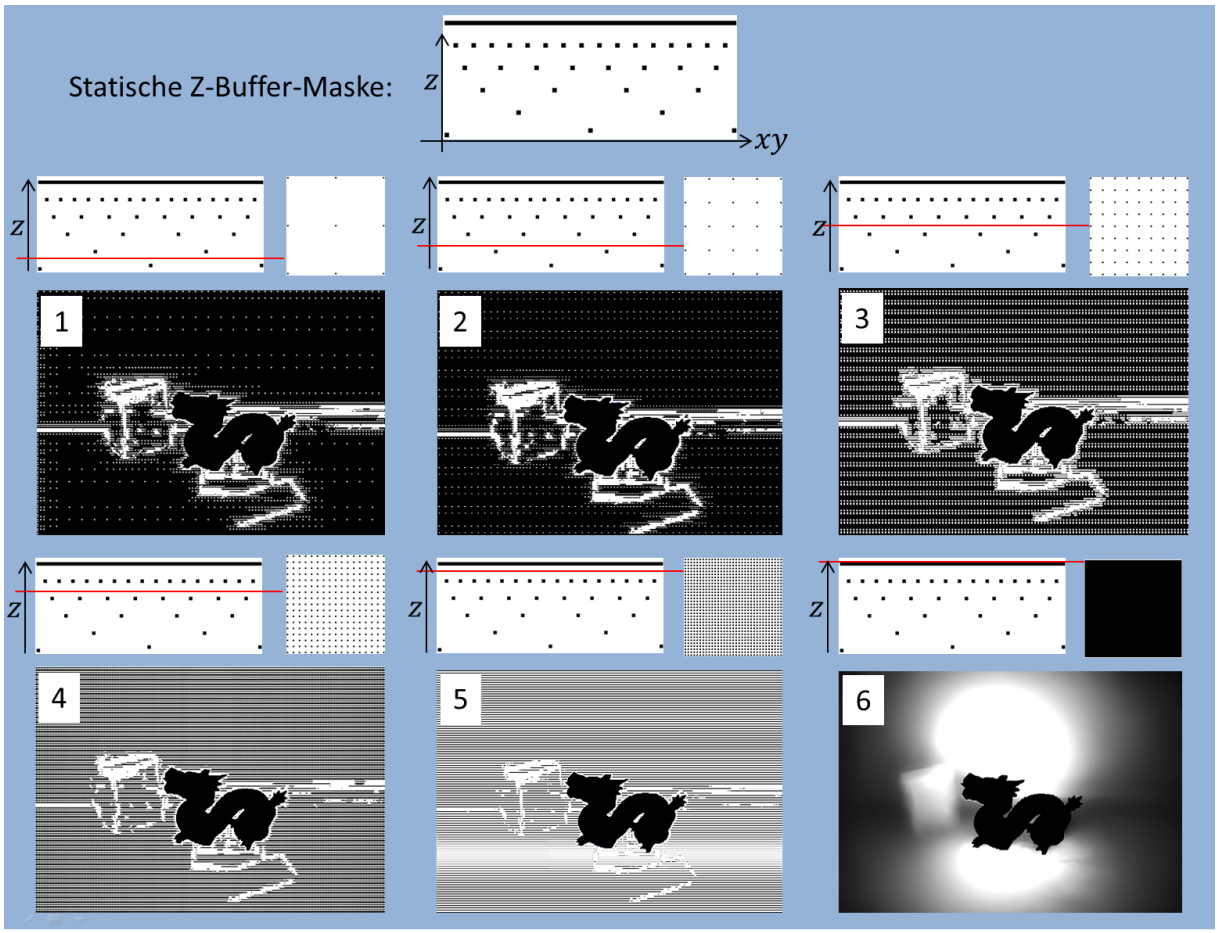

Abbildung 7.12: Prinzip der Bildraum-Interpolation beim LightSkin-Verfahren. Es wird eine statische Z-Buffer-Maske erstellt, die für die sukzessive Interpolation genutzt wird. Die rote Linie in der Z-Maske kennzeichnet das bildraumweite Quad, das für jede Verfeinerungsstufe weiter nach hinten bewegt wird. In jeder Verarbeitungsstufe werden die Pixel bearbeitet, die vor dem Quad liegen. Mit jedem Verarbeitungsschritt werden mehr Pixel verarbeitet, bis für jeden Pixel interpolierte Proxy-Lichtquellen zur Verfügung stehen.

Nach der Anwendung des Algorithmus ist für jeden sichtbaren Oberflächenpixel von der realen Szene die diffuse Proxy-Lichtquelle bekannt. Diese Proxy-Lichtquelle wird in einem letzten Verarbeitungsschritt genutzt, um für jeden Pixel die indirekte Reflexion zu berechnen. Da hier die Proxy-Lichtquelle nur in einem lokalen Beleuchtungsmodell verwendet werden muss, bedarf dieser Schritt keiner weiteren Erläuterungen.

7.3.4 Diskussion

Abbildung 7.13 zeigt eine Szene, die mit dem hier vorgestellten Verfahren beleuchtet wurde. Die Vergleichbarkeit zu existierenden globalen Beleuchtungsverfahren für AR-Anwendungen, wie sie in Kapitel 3.2 vorgestellt wurden, ist nicht ohne weiteres möglich (Franke hat einen qualitativen Vergleich in [Fra13a] vorgenommen, der in Kapitel 3.2 diskutiert wurde). Viele Einschränkungen, die die virtuelle Beleuchtung der realen Szene betreffen, wurden wegen der Verwendung des RGB-D-Sensors zur echtzeitfähigen Rekonstruktion der realen Szene formuliert. Würde man das LightSkin-Verfahren mit dem Differential-Rendering-Ansatz kombinieren und davon ausgehen, dass die reale Szene a priori bekannt ist, könnten alle Features des LightSkin-Verfahrens für die reale Szene simuliert werden. In diesem Fall könnte die reale Szene nicht nur diffuse Reflexionen abbilden, sondern ebenfalls glänzende Reflexionen und weiche Schatten. Diese Features werden von anderen aktuellen Verfahren unterstützt, jedoch vernachlässigen diese die Dynamik der realen Szene. Für diese Dissertation ist die Dynamik eine essentielle Forderung, die auch für die Abbildung der realen Szene gelten muss. Aus diesem Grund wurden Features vom LightSkin-Verfahren nicht angewendet, die durch die Echtzeitrekonstruktion nur unzureichend abgebildet werden können. Diese unzureichende Abbildung resultiert aus der adaptiven Bildraum-Verteilung der Caches. Die Verteilung basiert auf geometrischen Diskontinuitäten in den G-Buffern und berücksichtigt nicht die Reflexionseigenschaften der realen Objekte (da diese nicht bekannt sind). Eine Abbildung von glänzenden Oberflächen und weichen Schatten würde wegen der ständig wechselnden Cache-Verteilung deutliches Flackern erzeugen, das für den Betrachter nicht akzeptabel wäre.

Abbildung 7.13: Vergleich AR-Anwendung ohne globale Beleuchtungseffekte (links) und mit globalen Beleuchtungseffekten durch das LightSkin-Verfahren (rechts).

Es soll an dieser Stelle nicht verschwiegen werden, dass nicht jede beliebig komplexe reale Szene mit dem vorgestellten Filterungsverfahren abgebildet werden kann. Welche Szenen als zu komplex angesehen werden können, hängt von der Qualität des Sensors ab. Der Kinect-Sensor neigt bei komplexen Szenen zu großen undefinierten Tiefenbereichen, die durch ein Inpainting-Verfahren gefüllt werden müssen. Komplexe Strukturen können durch das Inpainting-Verfahren nur bedingt abgebildet werden, weswegen diese Bereiche unzureichend im Tiefenbild repräsentiert werden. Durch die Weiterentwicklung von Tiefensensoren wird dieses Problem aber zunehmend unbedeutender. Der Kinect2-Sensor von Microsoft erzeugt bereits deutlich weniger undefinierte Regionen im Tiefenbild.

8 Zusammenfassung

8.1 Übersicht und Schlussfolgerungen

In dieser Dissertation wurde ein neues globales Echtzeit-Beleuchtungsverfahren vorgestellt, das durch seinen neuartigen Interpolationsansatz qualitativ hochwertige Ergebnisse erzeugt, ohne die Anforderungen einer Echtzeitanwendung einzuschränken. Dabei wurde das Verfahren unter Berücksichtigung existierender wissenschaftlicher Ansätze entworfen und kombiniert die Prinzipien von Radiance-Caching-, Instant-Radiosity-, Finite-Elements- und Many-Light-Ansätzen.

Existierende globale Beleuchtungsverfahren für interaktive Anwendungen haben unterschiedliche Nachteile, wie zum Beispiel ein unzureichendes Laufzeitverhalten, fehlende Unterstützung von Animationen, temporäre Artefakte, langfristige und fehlerträchtige Vorberechnungen, unzureichende Qualität oder hohe Speicheranforderungen. All diese Problemfälle wurden bei der Entwicklung des LightSkin-Verfahrens berücksichtigt, indem vier Gütekriterien formuliert wurden, die insbesondere für das Anwendungsfeld Virtual Reality und Augmented Reality bedeutend sind. Diese Gütekriterien sind hohe Dynamik bzw. Interaktivität, Unterstützung von komplexen Szenen, plausible Beleuchtungsergebnisse und hohe Bildwiederholraten für große Bildschirmauflösungen. Um die Gütekriterien einzuhalten, wurde ein neues Interpolationsverfahren vorgestellt, mit dem es möglich ist, diffuse und glänzende indirekte Reflexionen abzubilden, die ebenfalls weiche Schatten erzeugen. Darüber hinaus lässt sich das Verfahren für die Synthese von Subsurface-Scattering-Materialien einsetzen, wenn diese ein hohes Streuungsverhalten und eine geringe Dämpfung innerhalb des Materials aufweisen.

Die Grundprinzipien des LightSkin-Verfahrens sind intuitiv und sollen in diesem Unterkapitel nochmals zusammengefasst werden.

Die Simulation der indirekten Beleuchtung basiert auf den Ideen des Instant Radiosity, bei dem indirektes Licht durch eine endliche Menge von virtuellen Lichtquellen abgebildet wird, die in den Bereichen der Szene verteilt werden, in denen direktes Licht einfällt. Ein effizientes Verfahren zur Erzeugung dieser indirekten Lichtquellen ist das Reflective Shadow-Mapping [DS05]. Dieses stellt eine Erweiterung des klassischen Shadow-Mappings dar, bei dem ein einzelner Pixel eine indirekte Punktlichtquelle repräsentiert. In dieser Arbeit wurde gezeigt, dass eine punktuelle Repräsentation der indirekten Lichtquellen zu einer artefaktbehafteten Darstellung führt. Deshalb werden beim LightSkin-Verfahren stattdessen homogene Kreisflächenlichtquellen genutzt, um das indirekte Licht zu repräsentieren. Hierdurch konnten bereits erste Artefakte reduziert werden.

Die Berechnung der indirekten Reflexionen für alle sichtbaren Oberflächenpunkte einer Szene, auf Basis der virtuellen Flächenlichtquellen, ist aus Laufzeitgründen nicht möglich, insbesondere weil Animationen in der Szene hohe Auflösungen für die RSMs voraussetzen, damit keine flackernden Artefakte auftreten. Aus diesem Grund wurde in dieser Arbeit nur eine lose, nicht dichte Menge von Oberflächenpunkten (Caches) genutzt, um das indirekte Licht von den Flächenlichtquellen zu evaluieren. Die Verteilung dieser Caches wurde im Modellraum vorgenommen und nicht, wie in vielen anderen Ansätzen, im Bildraum. Dies hat den Vorteil, dass die Caches nicht permanent neu verteilt werden müssen, wenn sich die Szene verändert. So ist es möglich, die Verteilung der Caches vor der

Simulation festzulegen. Diesbezüglich wurde ein Verfahren vorgestellt, das die Verteilung anhand zweier intuitiver qualitätsbestimmender Parameter automatisiert vornimmt. Die Parameter wurden mittels eines Fehlermodells hergeleitet, das ebenfalls Bestandteil dieser Dissertation ist. Die Berechnung der Verteilung dauert nur wenige Sekunden und kann interaktiv angepasst werden, so dass keine langen Wartezeiten entstehen. Das Ergebnis kann direkt getestet werden. Während der Echtzeitsimulation können die Caches auf einfache Art und Weise mit dem Modell transformiert werden. Durch die gleichförmige Bewegung mit dem Modell kommt es zu keinen störenden, flackernden Artefakten, stattdessen sind sanfte Übergänge zu beobachten, selbst wenn nur eine geringe Anzahl von Caches genutzt wird.

Die lose Verteilung der Caches im Modellraum verlangt nach einer neuen Art der Interpolation, die es erlaubt, durchgehende indirekte Reflexionen zu berechnen, ohne filigrane Oberflächendetails zu vernachlässigen. Hierfür wurde ein Verfahren entwickelt, bei dem für jeden Cache individuelle Punktlichtquellen erzeugt werden. Diese Punktlichtquellen approximieren das komplette indirekte Licht, das auf einen Cache einstrahlt. Die Punktlichtquellen stehen also repräsentativ für alle virtuellen Flächenlichtquellen und werden deshalb in der Arbeit als Proxy-Lichtquellen bezeichnet. Um physikalisch plausible Proxy-Lichtquellen zu erzeugen, wurden approximative Reflexionsmodelle hergeleitet, die effizient berechnet werden können und keine zu großen Fehler in der Reflexion produzieren.

Mit Hilfe der Proxy-Lichtquellen kann eine durchgehend indirekte Reflexion für jeden sichtbaren Oberflächenpunkt berechnet werden. Jedem Oberflächenpunkt wird eine feste Menge naher Caches zugewiesen. Basierend auf den geometrischen Ähnlichkeiten der jeweiligen Caches zum Oberflächenpunkt wird erneut eine gewichtete Proxy-Lichtquelle erzeugt, die für die Beleuchtung im Punkt genutzt wird. Zur Implementierung des Interpolationsverfahrens auf der GPU wurden zwei Ansätze vorgestellt, ein vertex- und ein fragmentbasierter Ansatz. Beide Ansätze erlauben die Berücksichtigung von feinen Oberflächendetails, die beispielsweise durch Normal-Maps erzeugt werden.

Das in dieser Arbeit vorgeschlagene Modell zur Interpolation von indirekten Reflexionen beinhaltet keine Prüfung, ob virtuelle Lichtquellen von einem Cache aus sichtbar sind oder nicht. Dies führt dazu, dass durch indirekte Reflexionen keine weichen Schatten entstehen können. Um dennoch weiche Schatten zu unterstützen, wurde ein approximatives Verfahren vorgeschlagen. Bei diesem wird die Sichtbarkeit nicht individuell für jeden Oberflächenpunkt und jede virtuelle Lichtquelle geprüft, sondern nur für jeden Cache und dessen Proxy-Lichtquellen. Hier wird für jede Proxy-Lichtquelle eine räumliche Ausdehnung gespeichert, die die Verteilung der virtuellen Lichtquellen repräsentiert. Um die Sichtbarkeitsprüfung pro Cache möglichst effizient zu halten, wird die Geometrie der Szene in Form von Kreisscheiben approximiert, wobei pro Cache eine Kreisscheibe erzeugt wird. Basierend auf diesen Kreisscheiben und der räumlichen Ausbreitung der Proxy-Lichtquellen wurde ein einfaches projektives Verfahren vorgeschlagen, mit dem effizient geprüft werden kann, ob eine Kreisscheibe einen Teil der Proxy-Lichtquelle verdeckt. Diese Verdeckungsprüfung geschieht ohne explizite Verwendung eines Z-Buffers und führt dazu, dass die Intensitäten der Proxy-Lichtquellen gegebenenfalls gedämpft werden. Allerdings kann es wegen des fehlenden Z-Bufferings vorkommen, dass Kreisscheiben eine Proxy-Lichtquelle blockieren, obwohl sie hinter einer anderen Kreisscheibe liegen (doppelte Verdeckung). In dieser Arbeit befindet sich aber auch ein Vorschlag, wie dieses Phänomen reduziert werden kann, indem die projizierte geometrische Ausbreitung aller verdeckenden Scheiben durch ein

umspannendes Rechteck zusammengefasst wird, dessen Fläche alternativ zur Dämpfung der Proxy-Lichtquelle genutzt werden kann (Kapitel 5.4.3).

Das Prinzip der Interpolation mit Hilfe von gewichteten Proxy-Lichtquellen kann auf beliebige Reflexionsmodelle übertragen werden, wenn diese analytische Formen von Lichtquellen unterstützen. Dies wurde im Rahmen dieser Arbeit evaluiert, indem das Interpolationsverfahren für das Dipol-Diffusions-Modell von Jensen et al. [JML*01] zur Simulation von Multiple-Subsurface-Scattering-Effekten modifiziert wurde. Bei diesem Modell wird die BSSRDF durch eine Dipol-Lichtquelle simuliert, die aus einem positiven und einem negativen Punktlicht besteht. Es wurde gezeigt, dass diese Dipole durch Proxy-Dipole abgebildet werden können, die grundsätzlich nach den gleichen Prinzipien erzeugt werden wie die indirekten Proxy-Lichtquellen bei indirekten Reflexionen, die auf einem BRDF-Modell basieren. Somit können Subsurface-Scattering-Materialien ebenfalls mit einer geringen Menge von Caches ansprechend in Echtzeit dargestellt werden, ohne Details der Objektoberfläche auszublenden. Allerdings wird bei diesem Ansatz davon ausgegangen, dass das Objekt konvex ist.

Echtzeitanwendungen erfordern hohe Bildwiederholfrequenzen. Zwar erlauben die vorgestellten Konzepte zur Interpolation eine deutlich schnellere Berechnung der Reflexionen, aber erst durch eine effiziente Implementierung auf einer modernen Grafikkarte werden Bildwiederholraten von über 30 Bildern pro Sekunde erreicht. In dieser Dissertation wurde deshalb eine komplette Pipeline beschrieben, die eine effiziente Implementierung darstellt. Ein Augenmerk wurde dabei speziell auf die effiziente Nutzung des GPU-Textur-Caches gelegt. Da das Verfahren für die Erzeugung der Proxy-Lichtquellen sehr viele Daten aus der RSM lesen muss, ist es besonders wichtig, dass die Lesezugriffe möglichst aus dem Textur-Cache bedient werden können. Dies gilt insbesondere für heutzutage übliche Grafikkarten, bei denen der Speicherzugriff bedeutend langsamer als die Durchführung einer arithmetischen Operation auf Registern ist. Darüber hinaus wurde ein GPU-Culling-Verfahren präsentiert, das die Zurückweisung von individuellen Caches erlaubt, die nicht zum finalen Bild beitragen. Hierfür wurde ein Histogramm-Algorithmus [SH07] modifiziert, der einzelne Caches durch eine Z-Buffer-Maske ausmaskiert. Weitere Optimierungsmethoden, wie Light-Clustering oder die adaptive Verteilung der Caches zur Laufzeit, wurden ebenfalls aufgezeigt.

Schlussendlich wurde erläutert, wie das LightSkin-Verfahren für eine Augmented-Reality-Anwendung adaptiert werden kann. Für diese Adaption wurde ein essentieller Nachteil bestehender globaler Beleuchtungsansätze für AR-Anwendungen aufgehoben, indem dynamische reale Szenen unterstützt werden. Bisherige Ansätze haben diese Szenen als statisch vorausgesetzt, so dass sie nur durch ein a priori (manuell) erzeugtes virtuelles Abbild repräsentiert werden können. Um dynamische Szenen zu unterstützen, wurde in dieser Dissertation die Verwendung eines RGB-D-Sensors vorgeschlagen. Mit diesem lässt sich in Echtzeit eine unvollständige perspektivische Abbildung erzeugen. Die direkte Nutzung dieses Modells für globale Beleuchtungseffekte ist mit den derzeit erhältlichen RGB-D-Sensoren nicht möglich, da deren Datenqualität nicht ausreichend ist. Deshalb wurde ein kantenerhaltendes Filterverfahren genutzt, um die subjektive Qualität der Abbildung so weit zu verbessern, dass diese für das Anwendungsgebiet der globalen Beleuchtung nutzbar ist. Da das LightSkin-Verfahren Caches im Modellraum voraussetzt, wurde ein alternatives Bildraum-Verfahren zur Verteilung und zur Interpolation der Caches entwickelt, das auf den Ideen von Nichols und Wyman [NW09] basiert. Bei der

Entwicklung des Verfahrens wurde eine nahtlose Integration in das LightSkin-Verfahren sichergestellt, indem der Z-Buffer zur Ausmaskierung der Caches genutzt wird.

Die unvollständige Darstellung der realen Szene durch einen RGB-D-Sensor führt zu der eingeschränkten Nutzbarkeit von bestimmten Features des LightSkin-Verfahrens für AR-Anwendungen. So wird für alle Oberflächen der realen Szenen vorausgesetzt, dass diese diffuse Reflektoren sind. Darüber hinaus wird auf die Abbildung von weichen Schatten durch indirekte Reflexionen wegen der permanenten Neuverteilung der Caches im Bildraum verzichtet. Diese Einschränkungen wurden bewusst gewählt, um eine größtmögliche Dynamik zu erlauben, die, wie bereits erwähnt, in anderen Ansätzen vernachlässigt wurde.

Nachfolgend soll noch aufgezeigt werden, wie die in dieser Arbeit formulierten Gütekriterien durch das LightSkin-Verfahren berücksichtigt werden.

1. Dynamik und Interaktivität: Hierbei handelt es sich um das wichtigste Kriterium, das bei der Entwicklung des LightSkin-Verfahrens berücksichtigt wurde. Die Dynamik wird auf zwei Arten unterstützt: Zum einen wurde sichergestellt, dass Animationen kaum Extrakosten verursachen und zum anderen wurde darauf geachtet, dass Animationen keine temporären Artefakte bei der Beleuchtung hervorrufen. Ersteres wurde erreicht, indem keine statischen Beleuchtungsergebnisse a priori durch ein Offline-Verfahren berechnet wurden, die durch Animationen ungültig werden würden. Außerdem setzt das Verfahren keine komplexen Beschleunigungsstrukturen voraus, die zu aufwändigen Aktualisierungen bei Veränderungen der Szene führen würden. Um Artefakte bei Animationen zu vermeiden, wurden die Caches im Modellraum verteilt. Dadurch sind temporäre Artefakte bei Kamerabewegungen per se ausgeschlossen, während Modell- und Lichtanimationen von der gleichförmigen Bewegung der Caches mit der Animation profitieren. Artefakte können allerdings trotzdem auftreten, wenn die Auflösung der RSMs unzureichend ist. Diesbezüglich wurde jedoch ein Interpolationsverfahren vorgestellt, das mit sehr wenigen Caches funktioniert, so dass sehr viele virtuelle Lichtquellen aus der RSM angewendet werden können, ohne die Laufzeit des Programms stark zu beeinflussen. Alternativ können zusätzlich Light-Clustering- und Importance-Sampling-Ansätze genutzt werden.

Bei der Adaption des LightSkin-Verfahrens für Augmented-Reality-Anwendungen wurde darauf geachtet, dass die Dynamik nicht eingeschränkt wird. So unterstützt das vorgestellte Verfahren sowohl dynamische virtuelle als auch reale Szenen. Um Artefakte bei Animationen der realen Szene zu vermeiden, wurden Einschränkungen bei den Ausbreitungseigenschaften des indirekten Lichts hingenommen.

2. Unterstützung von komplexen Szenen: Durch unterschiedliche Ansätze wurde sichergestellt, dass komplexe Szenen mit dem vorgestellten Verfahren simuliert werden können. Eine wichtige Eigenschaft ist der geringe Speicherbedarf des Verfahrens. Im Gegensatz zu anderen Echtzeitbeleuchtungsverfahren werden keine speicherineffizienten kubischen Strukturen genutzt. Stattdessen werden die Daten direkt auf den Oberflächen der Modelle gespeichert. Durch die geringe Menge der Caches und deren vergleichsweise geringen Speicherbedarf wurde gewährleistet, dass diese nur einen geringen Anteil des GPU-Speichers belegen. Lediglich die Zuordnung der Caches zu den Oberflächenpunkten erfordert, dass entweder pro Vertex oder pro Texel zusätzliche Indizes gespeichert werden müssen. Hierfür reichen in der Regel pro Element acht 16-Bit-Indizes aus. Zusätzlich wurden

zwei unterschiedliche Implementierungen für die Interpolation vorgeschlagen, die es erlauben, den Großteil der Berechnungen entweder im Modellraum pro Vertex oder im Bildraum pro Pixel durchzuführen. Dabei verursachen beide Implementierungen pro Oberflächenelement konstante Laufzeitkosten, weswegen beide Ansätze als effizient angesehen werden können. Sehr große und weitläufige Szenen können dargestellt werden, indem das vorgestellte Culling-Verfahren genutzt wird. Da große Szenen entsprechend viele Caches benötigen, wird durch das Culling-Verfahren sichergestellt, dass nur die Caches beleuchtet werden, die tatsächlich zur Beleuchtung des Framebuffers beitragen. Ferner wurden zwei Skalierungsmöglichkeiten aufgezeigt, die die geometrischen Zusammenhänge zwischen virtuellen Lichtquellen und Caches ausnutzen, um die Kosten für die Erzeugung der Proxy-Lichtquellen zu verringern. Zwar wurde kein Ansatz vorgestellt, der die Kosten in Abhängigkeit zur Entfernung zwischen Modell und Betrachter verringert, aber prinzipiell können hierfür die gleichen Ansätze genutzt werden.

3. Plausible Beleuchtungsergebnisse: In dieser Arbeit wurde ein Verfahren vorgestellt, das wichtige optische Phänomene der globalen Beleuchtung abbildet, die den subjektiven Eindruck der Szene deutlich verbessern. Es wurde gezeigt, dass die Beleuchtungsergebnisse des LightSkin-Verfahrens bei diffusen Reflexionen existierende Echtzeitverfahren, wie die Cascaded Light-Propagation-Volumes (CLPV) [KD10] oder das Voxel-Cone-Tracing (VCT) [CNS*11], in ihrer Plausibilität übertreffen. Im Gegensatz zu diesen Verfahren werden beim LightSkin-Ansatz die globalen Beleuchtungseffekte in ihrer räumlichen Ausbreitung nicht eingegrenzt. Durch den gewählten Ansatz konnten für die Berechnungen der indirekten Reflexionen physikalisch plausible Reflexionsmodelle eingesetzt werden. Ein „Auswaschen“ des indirekten Lichts mit wachsender Entfernung von der indirekten Lichtquelle, wie dies bei den CLPV und dem VCT zu beobachten ist, findet nicht statt. Jedoch soll nicht verschwiegen werden, dass das VCT bei glänzenden Reflexionen naher Objekte subjektiv bessere Ergebnisse liefert als das LightSkin-Verfahren. Dies hängt mit den vorgenommenen Vereinfachungen des Reflexionsmodells zusammen, die zu einem Energieabfall bei Reflexionen von sehr nahen Objekten führen. Nichtsdestotrotz ist das hier vorgestellte Verfahren einer der wenigen Instant-Radiosity-Ansätze, bei dem es möglich ist, glänzende Reflexionen in Echtzeit abzubilden. Dies wurde im Wesentlichen durch die Verwendung und geeignete Gewichtung der Proxy-Lichtquellen erreicht. Wie exakt diese Reflexionen sind, hängt mit der Anzahl der Caches zusammen. Es hat sich aber gezeigt, dass bereits wenige Caches ausreichen, um einen subjektiv guten Eindruck zu erhalten, selbst wenn die Reflexionen nicht exakt sind. Dies gilt insbesondere, weil wenig Caches nicht dazu führen, dass Oberflächendetails bei der Schattierung unberücksichtigt bleiben.

4. Hohe Bildwiederholraten bei hohen Auflösungen: Es wurde gezeigt, dass sich alle Algorithmen des LightSkin-Verfahrens effizient auf der Grafikkarte implementieren lassen, ohne ein komplexes Shader-Modell vorauszusetzen. Die hohen Bildwiederholraten kommen hauptsächlich durch die geringe Menge der Caches zustande, für die Proxy-Lichtquellen berechnet werden. Die Berechnungskosten für einen einzelnen Pixel des Framebuffers sind konstant und gering, so dass hohe Auflösungen kein Problem darstellen. Ferner sind die optischen Phänomene, die durch das Verfahren simuliert werden, separat ein- und ausschaltbar, so dass auch leistungsschwache Systeme genutzt werden können.

Zusammenfassend kann festgehalten werden, dass in dieser Arbeit ein neues globales Beleuchtungsverfahren vorgestellt wurde, welches mit wenig Speicher und ohne komplexe Vorberechnungen voll dynamische Szenen in Echtzeit plausibel darstellt.

8.2 Ausblick

Es entspricht der Auffassung des Autors, dass das Potential des in dieser Arbeit vorgestellten Beleuchtungsansatzes noch nicht ausgeschöpft ist. Dies betrifft sowohl wissenschaftliche, als auch anwendungsseitige Entwicklungen. Deshalb sollen in den nachfolgenden zwei Kapiteln die wissenschaftliche Anschlussfähigkeit und der Anwendungsnutzen diskutiert werden.

8.2.1 Wissenschaftliche Anschlussfähigkeit

Die Ergebnisse dieser Arbeit erlauben eine Vielzahl von weiteren Forschungsarbeiten, die sich im Bereich der Laufzeit- oder Qualitätsoptimierung eingruppieren lassen. Zuerst sollen mögliche Optimierungen zur Laufzeit diskutiert werden.

Bei der Entwicklung des LightSkin-Verfahrens wurde großer Wert auf die effiziente Berücksichtigung von Animationen gelegt. Deshalb wird für alle Modelle und Lichtquellen angenommen, dass diese durchgehend animiert sind. Es werden also pro Frame alle sichtbaren Proxy-Lichtquellen neu berechnet. Bei den meisten praxisbezogenen Szenarien kann man jedoch davon ausgehen, dass nicht alle Objekte durch eine permanente Animation verändert werden. Für diese Fälle können die temporären Kohärenzen in der Szene ausgenutzt werden, um nur für die Bereiche neue Proxy-Lichtquellen zu bestimmen, die sich tatsächlich verändert haben. Zusätzlich kann man das Prinzip des Incremental Instant Radiositys von Laine et al. [LSK*07] für das LightSkin-Verfahren adaptieren, indem nur eine inkrementelle Aktualisierung der Proxy-Lichtquellen durchgeführt wird. Pro Frame würden also nur Veränderungen im indirekten Licht auf die Proxy-Lichtquellen übertragen. Sofern sich die Szene durch kontinuierliche Animationen verändert, kann man von einem signifikanten Anstieg der Ausführungsgeschwindigkeit ausgehen.

Eine andere Optimierung bietet sich durch die Entwicklung effizienterer Level-of-Detail-Ansätze für die Caches der Szene an. Erweiterungen können darin bestehen, dass die Menge der Caches für weit entfernte Objekte reduziert wird (vom Betrachter aus gesehen). Sofern für die Speicherung der Cache-Indizes Texturen eingesetzt werden, könnten hierfür angepasste Mip-Map-Ketten genutzt werden. Bei diesen Ansätzen ist es besonders wichtig, dass die verschiedenen Detailstufen ineinander gemischt werden, so dass keine sprunghaften Veränderungen in der Beleuchtung zu erkennen sind. Außer der Entfernung zum Betrachter könnten auch andere Metriken zur adaptiven Verteilung von Caches genutzt werden, wie beispielsweise die Reflexionseigenschaften der Caches. In diesem Fall erhalten Objekte mit glänzenden Oberflächen mehr Caches als Objekte mit diffusen Oberflächen. Möglicherweise lassen sich unter Berücksichtigung der Reflexionseigenschaften auch optische Phänomene wie Kaustiken effizient abbilden, wenn man in den Bereichen besonders viele Caches verteilt, in denen eine Kaustik erwartet wird.

Es lässt sich ebenfalls ein verbessertes Light-Clustering-Verfahren für das LightSkin-Verfahren entwickeln. Das in dieser Arbeit vorgestellte Verfahren nutzt nur die Entfernung zwischen Lichtquelle und Cache, um zu entscheiden, ob eine zusammengefasste Lichtquelle angewendet werden kann oder nicht. Dies hat zur Folge, dass flackernde Artefakte auftreten können, wenn der Schwellenwert für die Entfernung schlecht gewählt wurde. Ein alternatives Verfahren könnte auf einen solchen

Schwellenwert verzichten und stattdessen prüfen, ob die nächstfeinere Detailstufe der virtuellen Lichtquellen eine nennenswerte Veränderung für die indirekte Reflexion hervorruft. Ist dies nicht der Fall, müssen keine weiteren Lichtquellen angewendet werden. Bei diesem Verfahren könnte das Light-Clustering durch einen prozentualen Wert gesteuert werden, der in direkter Korrelation zur Qualität der indirekten Beleuchtung steht. Alternativ zum Light-Clustering können aber auch schnelle Importance-Sampling-Ansätze getestet werden, die den Betrachtungsausschnitt einbeziehen, wie dies beispielsweise in [REH*11] gemacht wurde.

Es können weitere Forschungen vorgenommen werden, bei denen die Verbesserung der Bildqualität im Vordergrund steht. Beispielsweise wäre interessant, ob bessere Ergebnisse für die indirekte Beleuchtung erzielt werden können, wenn anstatt Proxy-Punktlichtquellen homogene Proxy-Flächenlichtquellen verwendet werden. Grundsätzlich ist davon auszugehen, dass eine Flächenlichtquelle die komplette indirekte Beleuchtung besser abbilden kann als eine Punktlichtquelle. Es stellt sich aber die Frage, ob Proxy-Flächenlichtquellen effizient und artefaktfrei interpoliert werden können. Jedoch bestehen auch wenn das Prinzip der Punktlichtquellen beibehalten wird Möglichkeiten zur Optimierung. Gegenwärtig wird für jeden Cache nur eine Proxy-Lichtquelle erzeugt, unabhängig davon, ob das indirekt einfallende Licht durch diese hinreichend abgebildet werden kann. Hierdurch können Artefakte entstehen, die in Kapitel 5.3.6 erläutert wurden. In dieser Arbeit wurde eine ad-hoc-Lösung zur Behebung dieser Artefakte vorgestellt, die Spielraum für Optimierungen lässt, wenn man erlaubt, dass ein Cache mehrere Proxy-Lichtquellen generieren kann. Findet man eine effiziente Möglichkeit, zu evaluieren, ob eine Proxy-Lichtquelle das indirekte Licht unzureichend approximiert, könnte man in diesen Fällen zusätzliche Proxy-Lichtquellen erzeugen. Eine mögliche Lösung für dieses Problem wäre, dass man für mehrere Abschnitte der Hemisphäre separate Proxy-Lichtquellen bildet. In einem zweiten Durchgang werden dann die Lichtquellen untereinander verglichen und geprüft, ob diese zusammengefasst werden können oder nicht.

Auch für die Berechnung der weichen Schatten sind Optimierungsmöglichkeiten denkbar. Gegenwärtig wird die komplette Szene in Form von Kreisscheiben für die Verdeckungsrechnung approximiert. Dies geschieht aus Laufzeitgründen, da für Kreisscheiben einfache analytische Formfaktoren existieren. Kreisscheiben bilden jedoch längliche Objekte, wie zum Beispiel Stangen, Masten oder Säulen, schlecht ab. Hier wären beispielsweise Ellipsenscheiben besser geeignet. Allerdings stellt sich die Frage, ob für diese ein effizienter Überlappungstest mit der indirekten Proxy-Lichtquelle entwickelt werden kann. Bei der Berechnung der Schatten besteht ebenfalls die Möglichkeit, einen binären Microbuffer im Sinne des Ansatzes von Ritschel et al. [REG*09] einzusetzen, um das Problem der doppelten Verdeckungsrechnung zu lösen.

Die Adaption des Verfahrens für AR-Anwendungen kann möglicherweise ebenfalls verbessert werden, um weitere globale Beleuchtungsphänomene auf realen Oberflächen abzubilden (glänzende Reflexionen oder weiche Schatten). Durch die Verwendung eines robusten räumlichen Registrierungsverfahrens wäre es beispielsweise möglich, Caches stationär in der realen Szene zu verteilen. Damit wäre deren Verteilung nicht mehr vom Bildraum abhängig und es würden weniger temporäre Artefakte entstehen. Allerdings müsste die Registrierung bewegte Objekte erkennen und die Caches korrekt mittransformieren, was gegenwärtig noch nicht möglich ist.

Schlussendlich wäre für zukünftige Forschungen interessant, ob sich mit dem Prinzip der Proxy-Lichtquellen auch Single-Scattering-Terme für Subsurface-Scattering-Materialien approximieren lassen. Diesbezüglich wurden keine geeigneten Ansätze gefunden.

8.2.2 Anwendungen

Das LightSkin-Verfahren wurde für Virtual- und Augmented-Reality-Anwendungen entwickelt, für die spezifische Anforderungen formuliert wurden. So erlaubt das Verfahren beispielsweise eine realistischere Darstellung der Beleuchtungssituation in stereoskopischen CAVE-Systemen, bei denen über mehrere Leinwände hohe Auflösungen eingesetzt werden. Auch im Bereich der digitalen Spieleentwicklung ist das Verfahren gut einsetzbar, da es eine realistischere Darstellung von Spielwelten erlaubt, die die Immersion des Spiels verbessern kann. Dabei muss für die jeweiligen Anwendungsfälle berücksichtigt werden, dass die Rechenleistung der genutzten Hardware sehr verschieden sein kann. Das LightSkin-Verfahren erlaubt eine Skalierung der Rechenleistung, indem die Menge der genutzten Caches reduziert wird. Dies hat zwar zur Folge, dass hochfrequentes indirektes Licht schlechter abgebildet wird, dies macht sich aber nur auf glänzenden, homogenen Oberflächen bemerkbar. Die indirekten Reflexionen bleiben dagegen hochfrequent und wirken in den meisten Fällen plausibel für den Betrachter. Zusätzlich ist es möglich, Features für die indirekte Beleuchtung zu reduzieren. In der simpelsten Form würde man nur die indirekten diffusen Reflexionen berechnen, ohne Berücksichtigung von blockierenden Elementen. Da das Verfahren mit dem Shader-Model 3.0 bereits implementierbar ist, kann eine Implementierung für Smartphones vorgenommen werden, wenn diese die Grafikschnittstelle OpenGL ES 3.0 unterstützen. Jedoch ist die Grafikleistung von Smartphones deutlich geringer als von Desktop-Systemen, so dass man davon ausgehen kann, dass sich auf den mobilen Geräten derzeit nur diffuse indirekte Reflexionen abbilden lassen.

Der Nutzen des Verfahrens beschränkt sich nicht nur auf dessen Anwendung in Echtzeitsimulationen. Bereits beim Authoring von dreidimensionalen Szenen mit entsprechenden 3D-Modellierungsprogrammen kann das Verfahren eingesetzt werden, um einen direkten Eindruck für ein Licht-Setup zu erhalten. Der Grafiker kann sich indessen sicher sein, dass das Licht-Setup identisch in der Echtzeitsimulation abgebildet wird. Er könnte das Ergebnis der indirekten Beleuchtung sogar aktiv beeinflussen, indem er die Cache-Verteilung von Hand optimiert, wenn er dies für erforderlich hält. Der Prozess der aufwändigen, fehlerträchtigen und unflexiblen Erzeugung von vorberechneten Beleuchtungsergebnissen in Form von Light-Maps oder des Precomputed Radiance-Transfers wäre dann nicht mehr erforderlich. Demnach lässt sich sagen, dass der Prozess der Erzeugung von virtuellen Szenen durch das Verfahren deutlich beschleunigt werden kann.

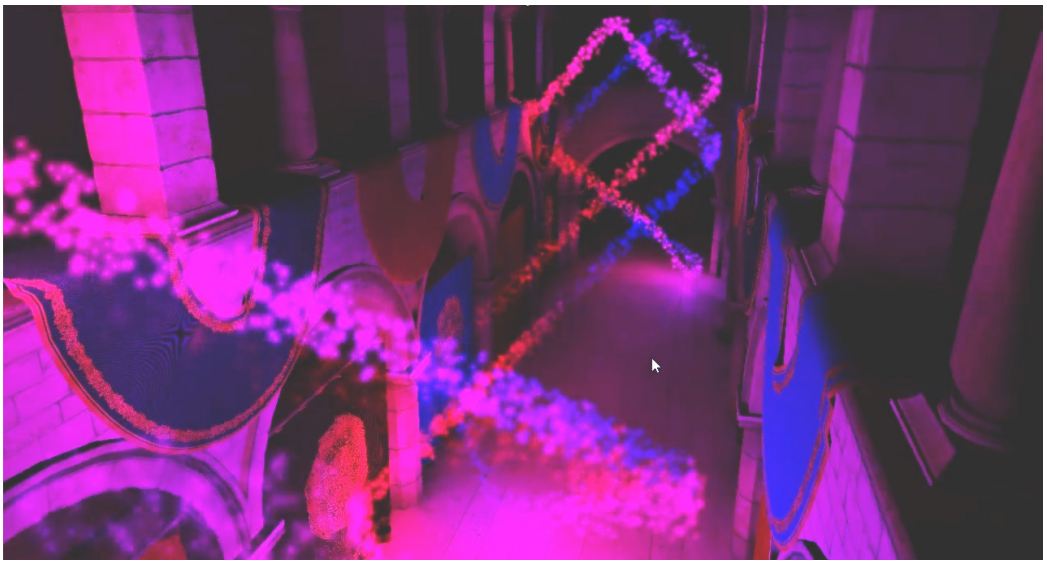

Abbildung 8.1: Nutzung des LightSkin-Verfahrens für die direkte Beleuchtung. Jedes Partikel entspricht einer direkten Lichtquelle. Mit dem LightSkin-Verfahren können mehrere zehntausend direkte Lichtquellen simuliert werden.

Bisher wurde in dieser Arbeit immer davon ausgegangen, dass das LightSkin-Verfahren zur Berechnung von indirekten Beleuchtungseffekten genutzt wird. Es lässt jedoch auch für die direkte Beleuchtung einer Szene nutzen. Wendet man das Prinzip der Proxy-Lichtquellen für die direkte Beleuchtung an, ist es möglich, Reflexionen für mehrere zehntausend primäre Lichtquellen in Echtzeit zu berechnen. Dies ist realisierbar, weil über die Verteilung der virtuellen Lichter keine Annahmen getroffen werden. Demnach können sie beliebig in der Szene verteilt liegen. Dies ist besonders für Partikelsimulationen interessant, bei denen jeder Partikel eine individuelle Quelle für Licht sein kann (siehe Abbildung 8.1). Dabei lässt sich das Verfahren sehr gut mit einem Light-Splatting- oder Tiled-Based-Deferred-Shading-Ansatz [Eng09] kombinieren. Diese können genutzt werden, um die direkte Beleuchtung für sehr nahe liegende Oberflächen darzustellen, während mit dem LightSkin-Verfahren entfernte Reflexionen abgebildet werden können.

Alternativ kann man das Verfahren auch einsetzen, um Partikel in einer Deferred Shading-Pipeline zu beleuchten (anstatt diese als Lichtquellen zu interpretieren). Das Deferred Shading erlaubt keine effiziente Beleuchtung von transluzenten Objekten. Partikel sind aber in der Regel transluzent und lassen sich demnach nicht effizient beleuchten (transluzente Objekte erfordern hohe Füllraten, so dass die Kosten pro Pixel gering sein müssen). Durch das LightSkin-Verfahren ist es möglich, für jeden Partikel nur wenige Caches zu nutzen. Dies erlaubt die Anwendung von sehr vielen Lichtquellen auf einen Partikel. Die Laufzeitkosten für einen einzelnen Pixel des Partikels sind dann konstant und gering, so dass eine hohe Füllrate erreicht werden kann.

Im Bereich der Augmented Reality kann das vorgestellte Verfahren genutzt werden, um die Integration von virtuellen Objekten in reale Szenen realistischer abzubilden. Ein möglicher Anwendungsfall wäre zum Beispiel die Raumgestaltung. Bei dieser spielen die Wechselwirkungen des Lichts eine große Rolle. Beispielsweise können Oberflächen bestehend aus Rottönen in einem Raum ein warmes Stimmungsbild hervorrufen. Die Stimmungsbilder lassen sich mit dem Verfahren abbilden, indem die indirekten Reflexionen in dem Raum direkt betrachtet werden können. Dies ist dann für verschiede-

ne Räume ohne aufwändige Vorberechnungen möglich, so dass ein mobiles System eingesetzt werden kann, das dem Kunden eine günstige Vorschau erlaubt.

Es existiert wahrscheinlich noch eine Vielzahl von weiteren Anwendungsgebieten, die in dieser Arbeit aber nicht erschöpfend aufgeführt werden können. Es sollte jedoch hinreichend herausgestellt worden sein, dass das Verfahren großes Potential birgt und einen hohen Nutzen für viele interessante Anwendungsgebiete in der Echtzeitbildsynthese hat.

Anhang A Laufzeitmessungen mit AMD Grafikkarte

In Kapitel 6.2.2 wurden Ergebnisse für Laufzeitmessungen präsentiert, die auf einem aktuellen Rechnersystem mit einer NVidia GTX 780 Grafikkarte gemessen wurden. In diesem Abschnitt werden Laufzeitergebnisse für die gleichen Testszenarien für ein alternatives PC-System präsentiert. Es wurde ein Desktop-System mit AMD Phenom II X4 945 3 GHz CPU, 4GB RAM und einer AMD Radeon HD 7970 Grafikkarte genutzt. Die Beschreibungen der jeweiligen Messungen können aus Kapitel 6.2.2 entnommen werden.

Tabelle 8.1: Abhängigkeit der Darstellungsrate in FPS von der Anzahl der Dreiecke. D8: nur diffuse Interpolation mit 8 Caches pro Oberflächenpunkt; D16: nur diffuse Interpolation mit 16 Caches; DG8: diffuse und glänzende Interpolation mit 8 Caches; DG16: diffuse und glänzende Interpolation mit 16 Caches; LI: ohne globale Beleuchtung.

Dreiecke	D8	D16	DG8	DG16	LI	LI/4	LI/8
800	771,9	763,4	755,9	686,7	1423,9	356,0	178,0
3.200	763,3	748,2	739,8	608,5	1412,9	353,2	176,6
12.800	687,4	647,4	638,8	410,2	1360,4	340,1	170,1
51.200	507,1	432,1	419,2	170,9	1114,3	278,6	139,3
102.400	369,8	297,5	285,8	96,5	886,4	221,6	110,8
204.800	245,9	185,7	178,5	51,1	661,4	165,4	82,7
409.600	146,8	105,6	101,1	26,5	443,1	110,8	55,4
819.200	82,5	57,6	55	13,4	270,1	67,5	33,8
1.638.400	44	30,1	28,7	6,7	155,7	38,9	19,5
2.457.600	30	20,4	19,4	4,5	109,1	27,3	13,6
3.276.800	22,8	15,4	14,7	3,4	84	21,0	10,5
4.096.000	18,3	12,4	11,8	2,7	68,3	17,1	8,5
4.915.200	15,2	10,4	9,8	2,3	57,5	14,4	7,2
5.734.400	13,1	8,9	8,4	1,9	49,7	12,4	6,2
6.553.600	11,5	7,8	7,4	1,7	43,7	10,9	5,5

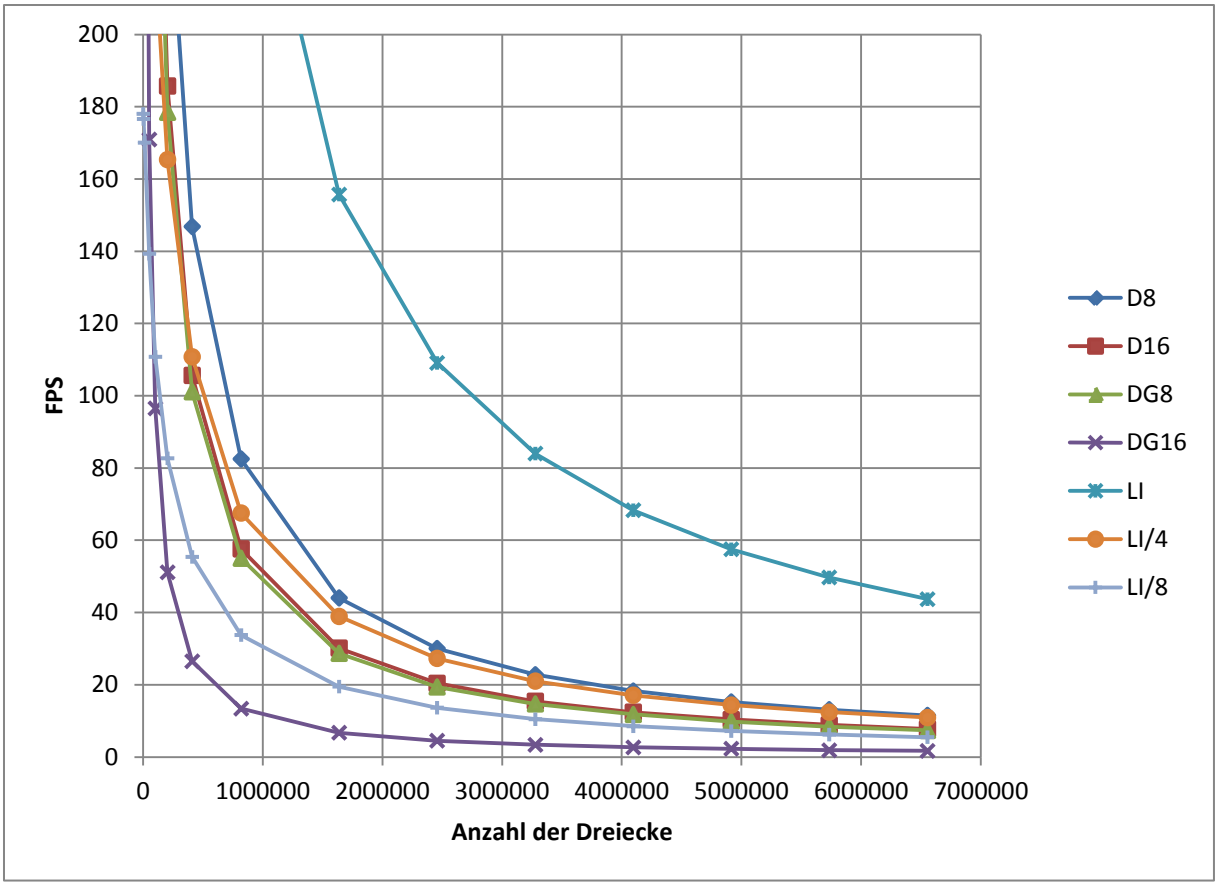

Abbildung 8.2: Darstellung der Abhängigkeit der Bildwiederholungsrate in FPS von der Anzahl der Dreiecke (siehe auch Tabelle 8.1).

Tabelle 8.2: Abhängigkeit der Bildwiederholungsrate von der Auflösung des Framebuffers. Zum Vergleich wurden ebenfalls die Raten ohne globale Beleuchtung in der Spalte LI festgehalten. Man erkennt, dass selbst 4K-Auflösungen noch mit ca. 60 FPS dargestellt werden können.

Auflösung	LI	FPS
256*256	2656	322
320*240	2725	322,1
512*512	1990	290,7
640*480	2591	306
800*600	1922,5	293
1024*768	1432,4	273,8
1280*720	1266,5	264
1024*1024	1038,7	251,2
1920*1080	585	211,5
2048*2048	300,8	145,2
4096*4096	76,7	64,1

Tabelle 8.3: Bildwiederholungsraten in Abhängigkeit von der Anzahl der Caches und von der Anzahl der VALs. D: nur diffuse Reflexionen, DG: diffuse und glänzende Reflexionen, DGS: diffuse und glänzende Reflexionen mit weichen Schatten, SSS: Subsurface-Scattering. Gemessen in einer Szene mit 150.000 Dreiecken.

	64x64 VALs (ca. 4.000)				128x128 VALs (ca. 16.000)				256x256 VALs (ca. 64.000)				512x512 VALs (ca. 256.000)			
Caches	**D**	**DG**	**DGS**	**SSS**	**D**	**DG**	**DGS**	**SSS**	**D**	**DG**	**DGS**	**SSS**	**D**	**DG**	**DGS**	**SSS**
250	483,3	466,8	413,8	464,2	425	381,6	345,4	346,8	314	251,3	235,1	158,9	147,1	99,9	97,2	49,6
500	473,9	434,3	368,5	386,8	382,3	326,1	286,9	248,8	234,3	174,7	163	91,9	90,3	59	57,6	26,4
1.000	438,4	395	314,8	282,6	315,3	251,5	215,2	158,8	156	108,4	101,4	51,1	51	32,4	31,7	14
2.000	426,1	386,9	306,3	280,7	301,2	238,9	204,9	159	143,9	99,1	92,8	51,4	46,1	29,2	28,6	14
3.000	402,7	357	269,3	231,9	259,6	197,3	167,1	119,8	110,2	78,5	69	36,2	33,1	22,3	20,3	9,5
4.000	381,8	332,2	240,5	192,8	228,3	172,4	141	93,3	89,3	61,7	54,8	27,4	25,8	17	15,8	7,1
5.000	355,3	306	117,9	169,4	201,2	145,5	82,7	76,1	74,7	48,5	38,7	21,5	21,2	13,2	12,3	5,5
6.000	339,5	287,4	103,8	152,7	182,3	129,5	78,2	65,1	64,5	41,5	33	18,2	17,9	11,1	10,4	4,7
7.000	323,4	270,2	92,7	138,6	165,9	116,2	63,8	57,4	56,7	36,2	28,7	15,7	15,5	9,6	9	4
8.000	309,7	254,8	83,4	119,9	153	105,3	57,1	51	50,6	32,2	25,5	13,8	13,7	8,5	7,9	3,5
9.000	297,6	241,5	76	113,8	141,6	97	51,9	46,6	45,7	29	22,9	12,5	12,3	7,6	7,1	3,2
10.000	287	230,8	69,2	104,6	131,6	89,4	47,1	42,4	41,7	26,3	20,8	11,3	11,1	6,9	6,4	2,9
11.000	275,3	218,6	57	93,7	123	82,6	39,6	38,8	38,2	24,1	18,3	10,3	10,2	6,3	5,8	2,6
12.000	267,3	209,3	50,4	81,4	115,5	77,4	35,8	35,9	35,4	22,2	16,6	9,5	9,4	5,7	5,3	2,4
13.000	257,5	200,2	45,3	72,3	108,5	72	32,4	33,3	32,9	20,6	15,3	8,7	8,7	5,3	4,9	2,2
14.000	248,1	192	41,4	62,8	102,3	67,9	29,7	31,1	30,7	19,2	14,1	8,2	8	4,9	4,5	2,1
15.000	240,2	184,3	38,4	56,3	97,3	64,2	27,7	29,2	28,8	18	13,1	7,6	7,6	4,6	4,2	1,9
16.000	233,3	177,2	36	51,5	92,5	60,6	25,8	27,5	27,2	16,9	12,3	7,1	7,1	4,4	4	1,8

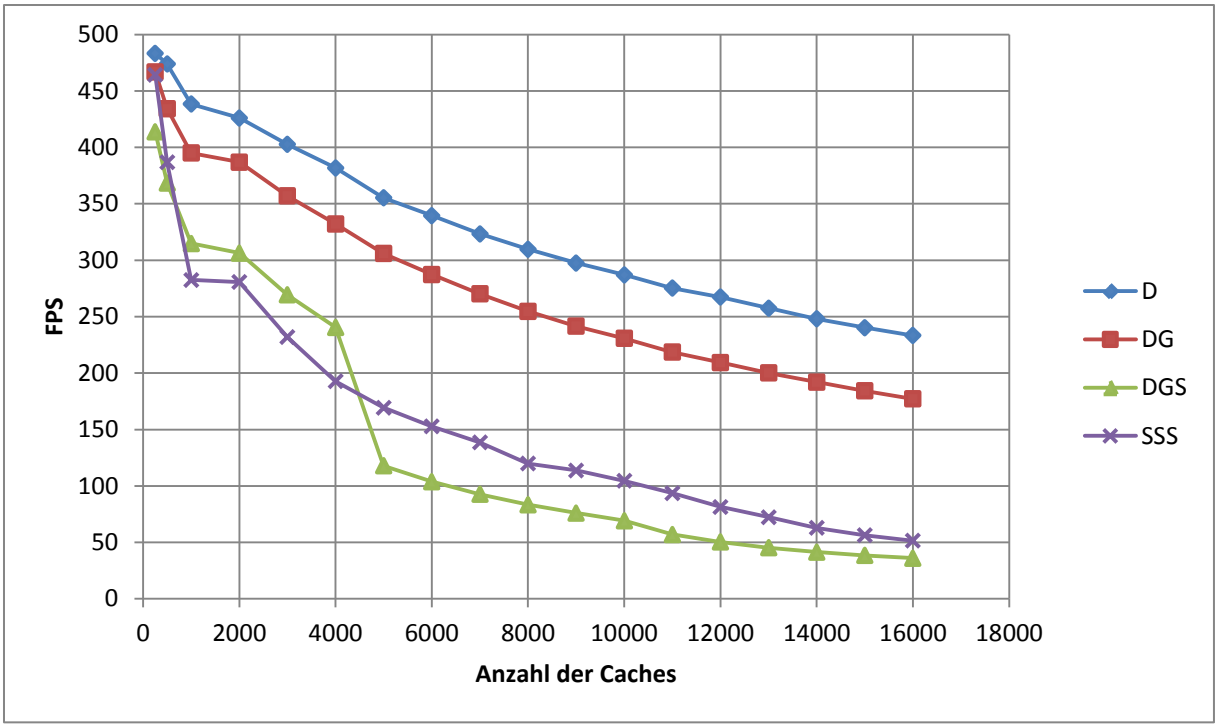

Abbildung 8.3: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 4.096 VALs genutzt werden (64x64).

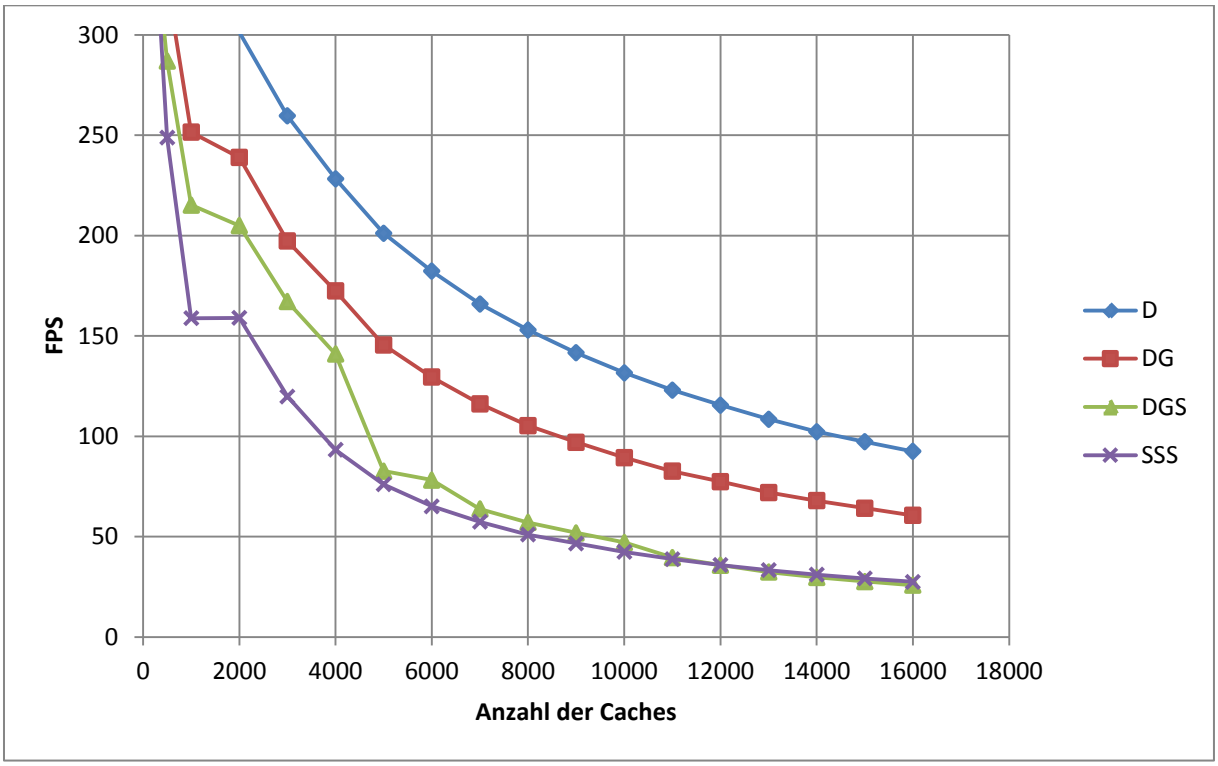

Abbildung 8.4: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 16.384 VALs genutzt werden (128x128).

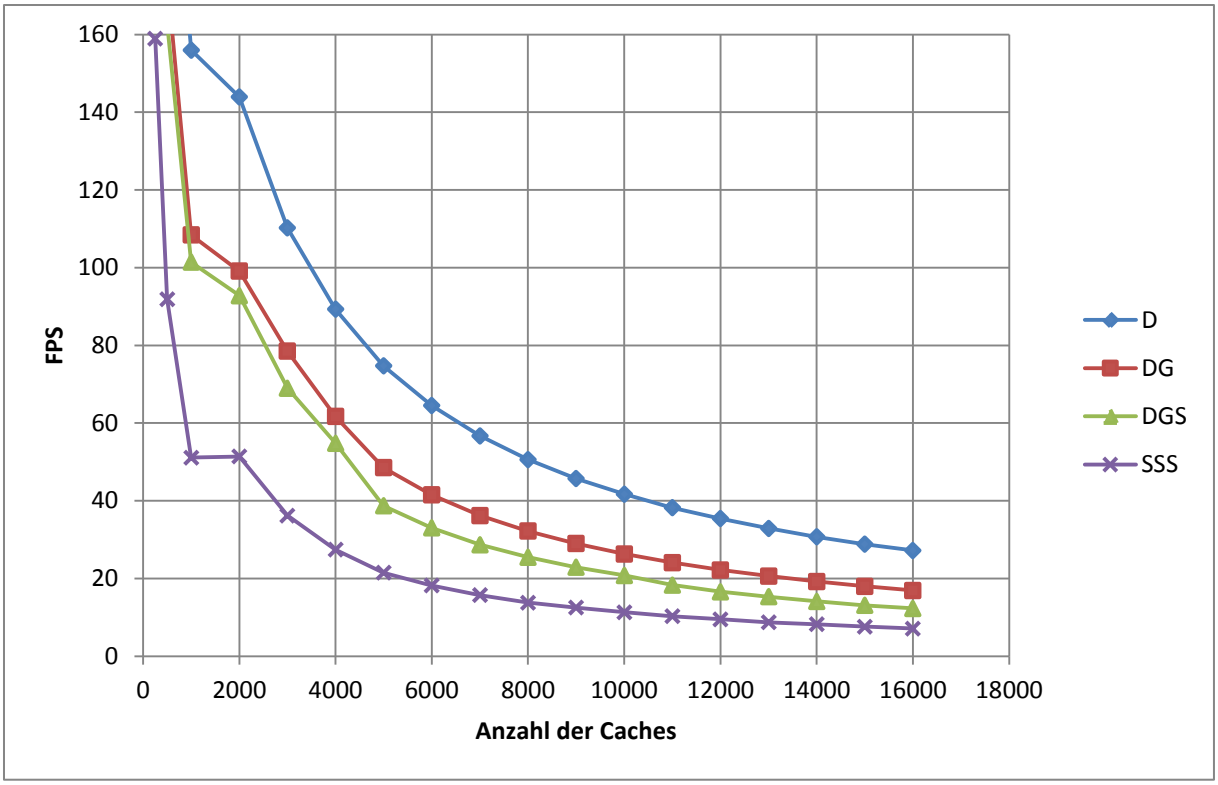

Abbildung 8.5: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 65.536 VALs genutzt werden (256x256).

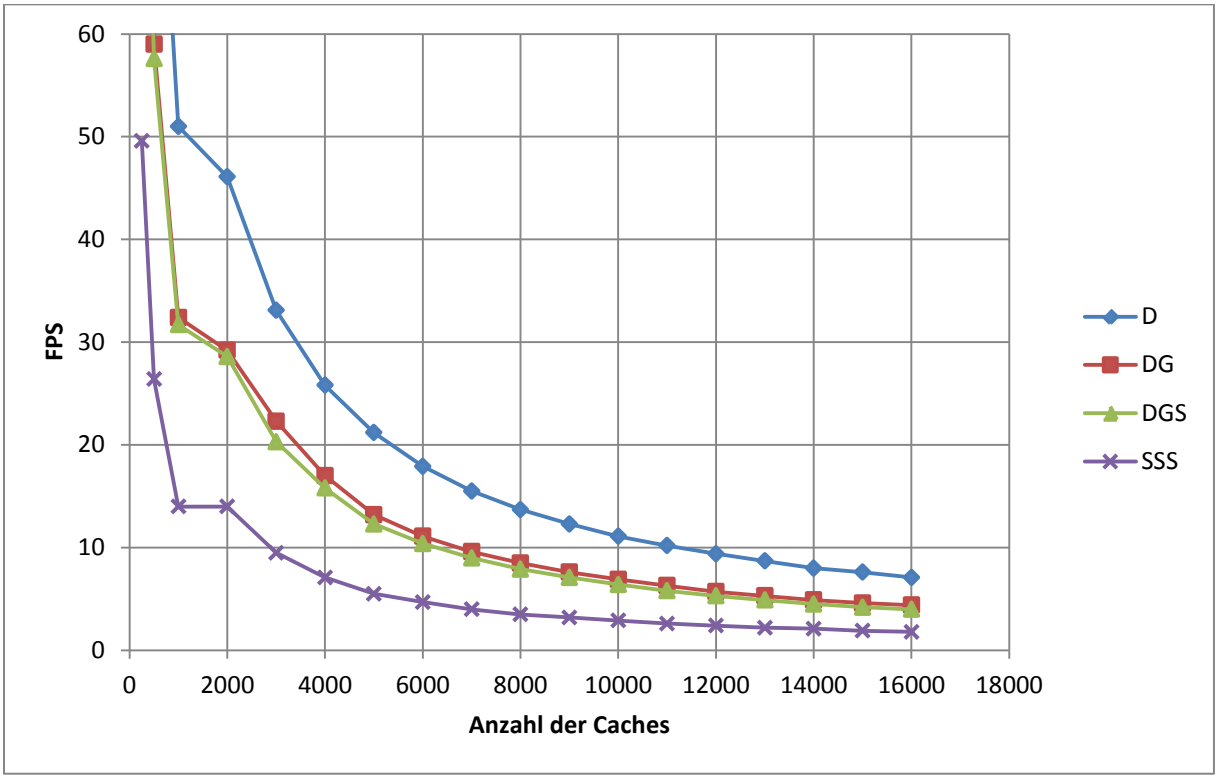

Abbildung 8.6: Abhängigkeit der Darstellungsrate von der Anzahl der Caches, wenn 262.144 VALs genutzt werden (512x512).

Anhang B Weitere Beleuchtungsergebnisse

In diesem Abschnitt befinden sich weitere Beleuchtungsbeispiele, Vergleichsbilder und Abbildungen aus vorherigen Kapiteln in höheren Auflösungen.

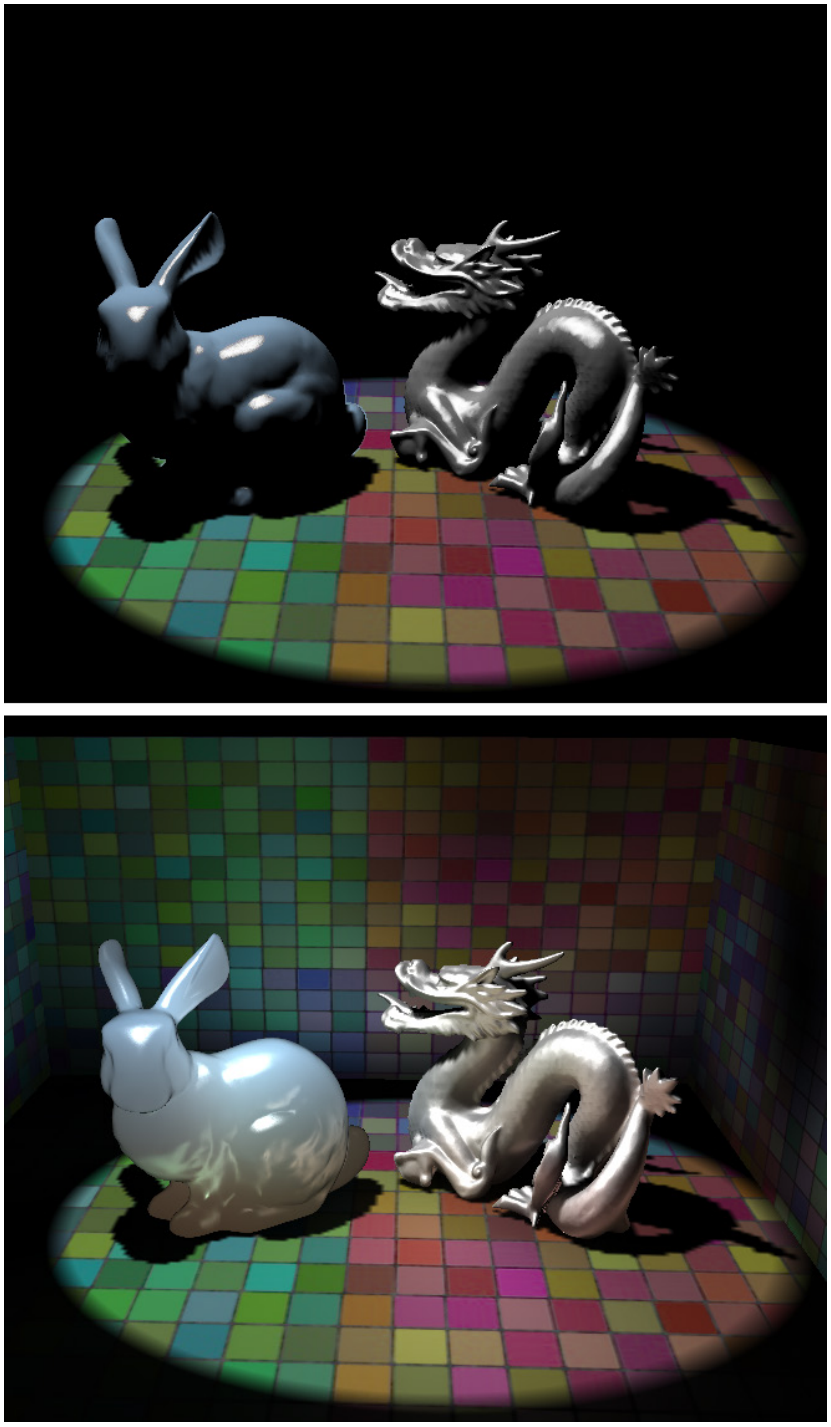

Abbildung 8.7: Vergleich zwischen lokaler Beleuchtung (oberes Bild) und globaler Beleuchtung mit dem LightSkin-Verfahren (unteres Bild). Die Szene wird durch insgesamt 400 Caches repräsentiert.

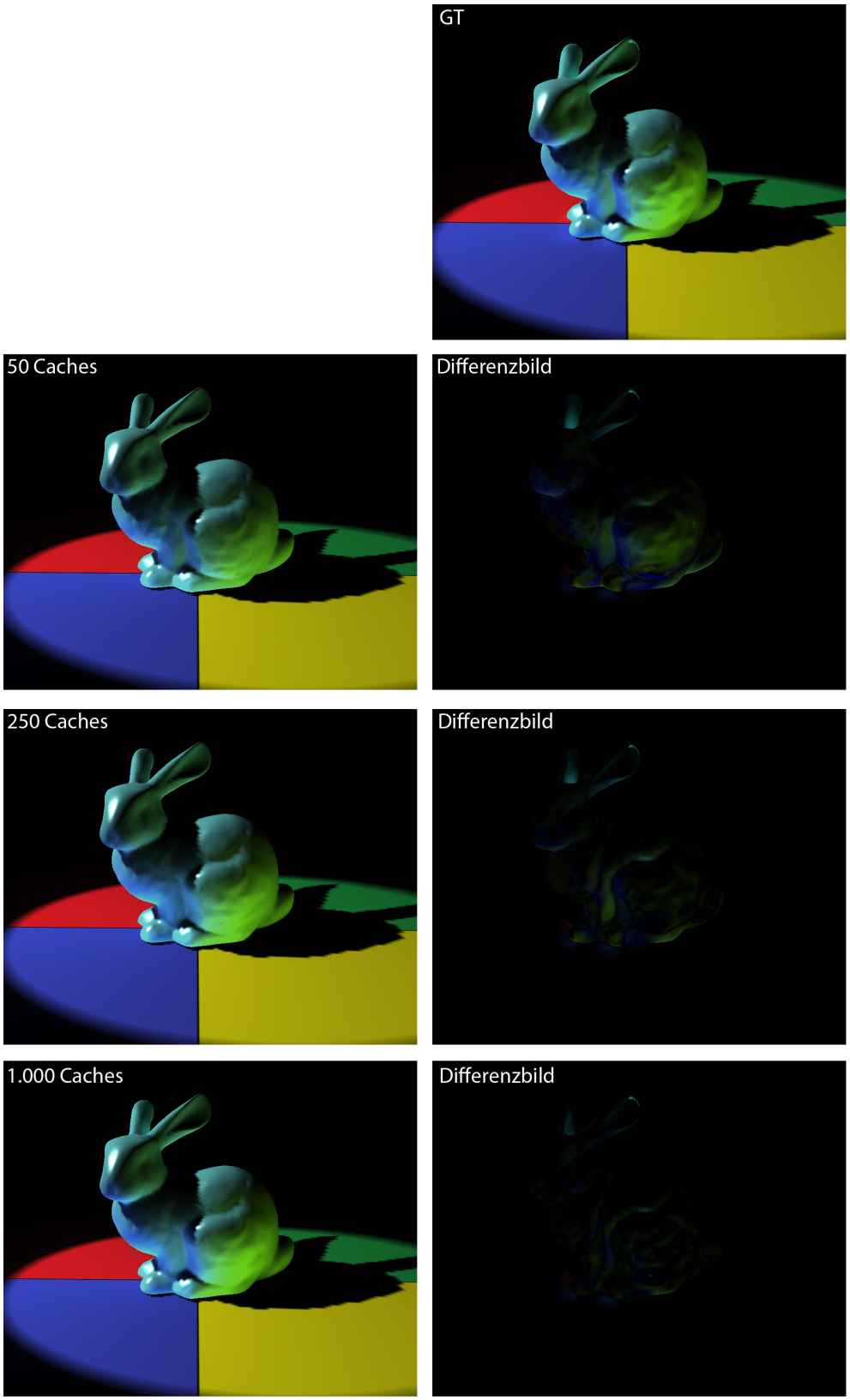

Abbildung 8.8: Vergleich zwischen diffusen indirekten Reflexionen für unterschiedliche Cache-Mengen, die mit dem Light-Skin-Verfahren berechnet wurden (links) zu einer Ground-Truth-Lösung ohne Interpolation (rechts oben). Anhand der Differenzbilder erkennt man, dass mit zunehmender Anzahl der Caches die diffuse Reflexion zur Ground-Truth-Lösung konvergiert. Man erkennt auch, dass wenige Caches ausreichen, um optisch plausible Reflexionen zu erzeugen.

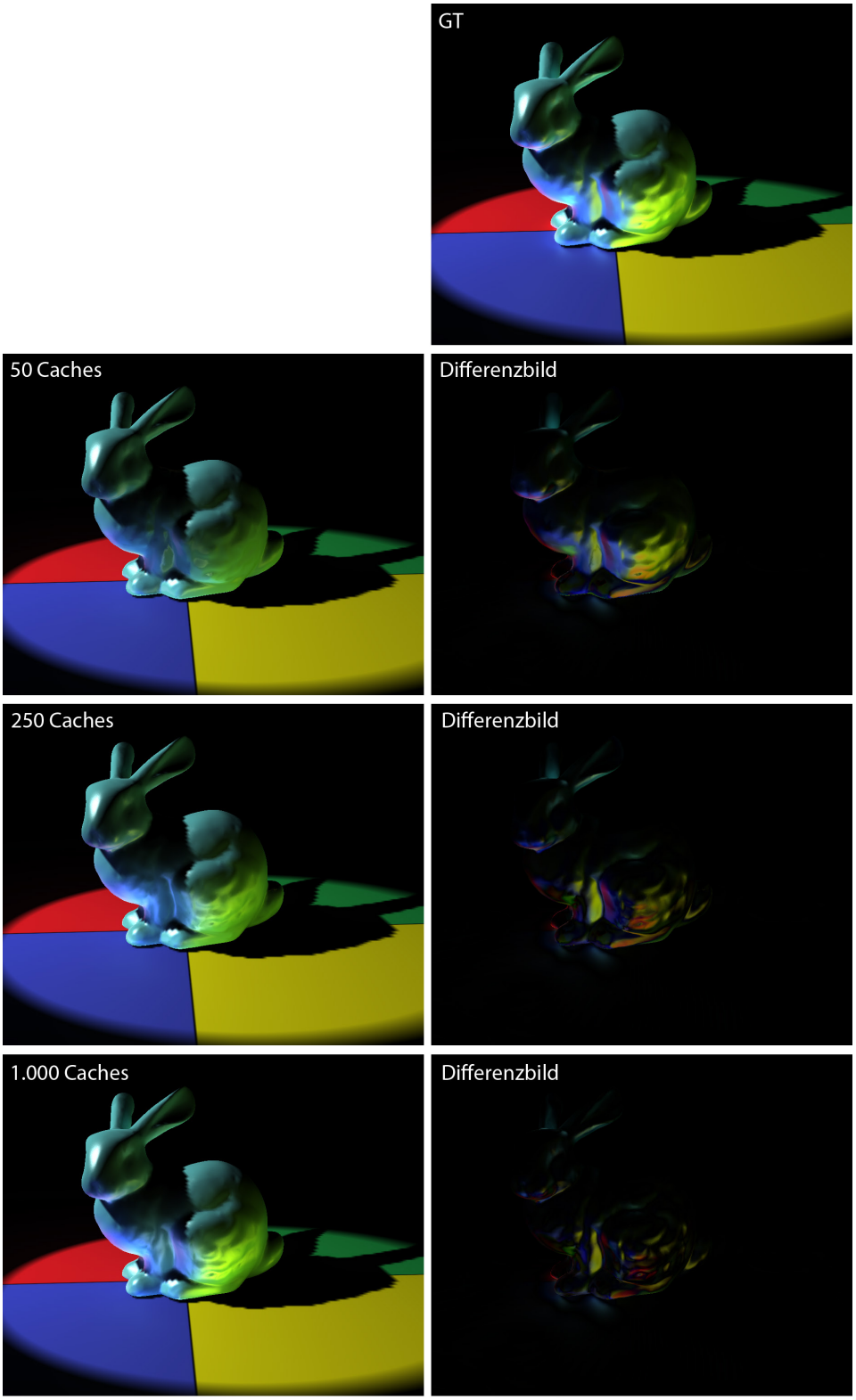

Abbildung 8.9: Vergleich zwischen glänzenden indirekten Reflexionen des LightSkin-Verfahrens (links) zu einer Ground-Truth-Lösung (rechts oben). Für das Modell wurde ein hoher Phong-Reflexionskoeffizient von $k = 40$ gewählt. Man erkennt, dass die interpolierte Lösung mit zunehmender Anzahl der Caches korrekter wird, allerdings reichen selbst 1.000 Caches nicht aus, um die Ground-Truth-Lösung fehlerfrei abzubilden. Dies hängt mit dem hohen Reflexionsexponenten zusammen, der beim LightSkin-Verfahren zu weniger exakten Abbildungen führt.

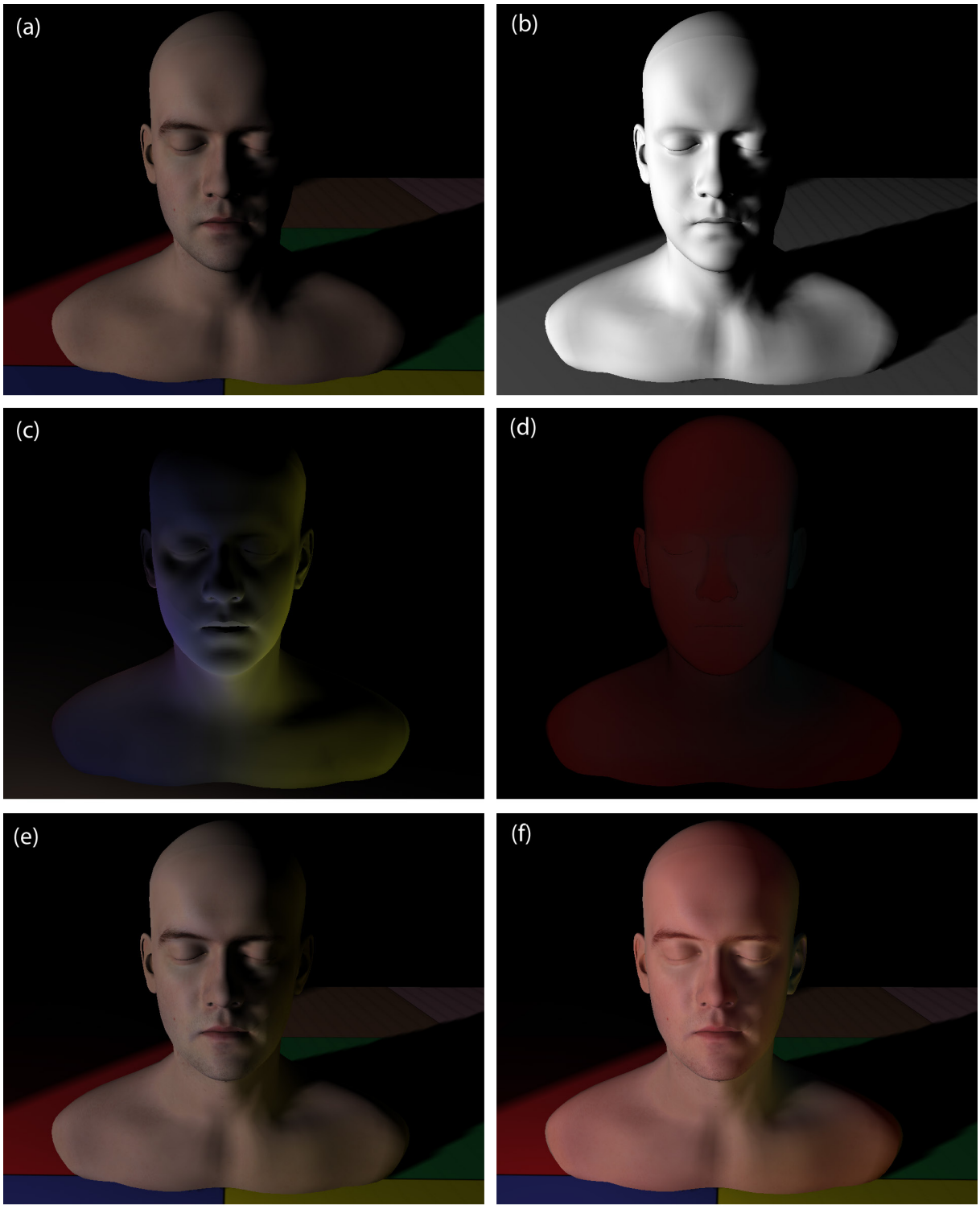

Abbildung 8.10: Nutzung des LightSkin-Verfahrens zur Abbildung menschlicher Haut. Die Kombination von indirekten Reflexionen und Multiple-Subsurface-Scattering-Effekten kann genutzt werden, um menschliche Haut realistisch abzubilden. (a) Kopf mit lokaler direkter Beleuchtung, (b) direkte Reflexionen, (c) indirekte Reflexionen, (d) Subsurface-Scattering, (e) direkte und indirekte Reflexionen, (f) Kombination von direkten Reflexionen, indirekten Reflexionen und Subsurface-Scattering. Für das Modell wurden 158 Caches genutzt.

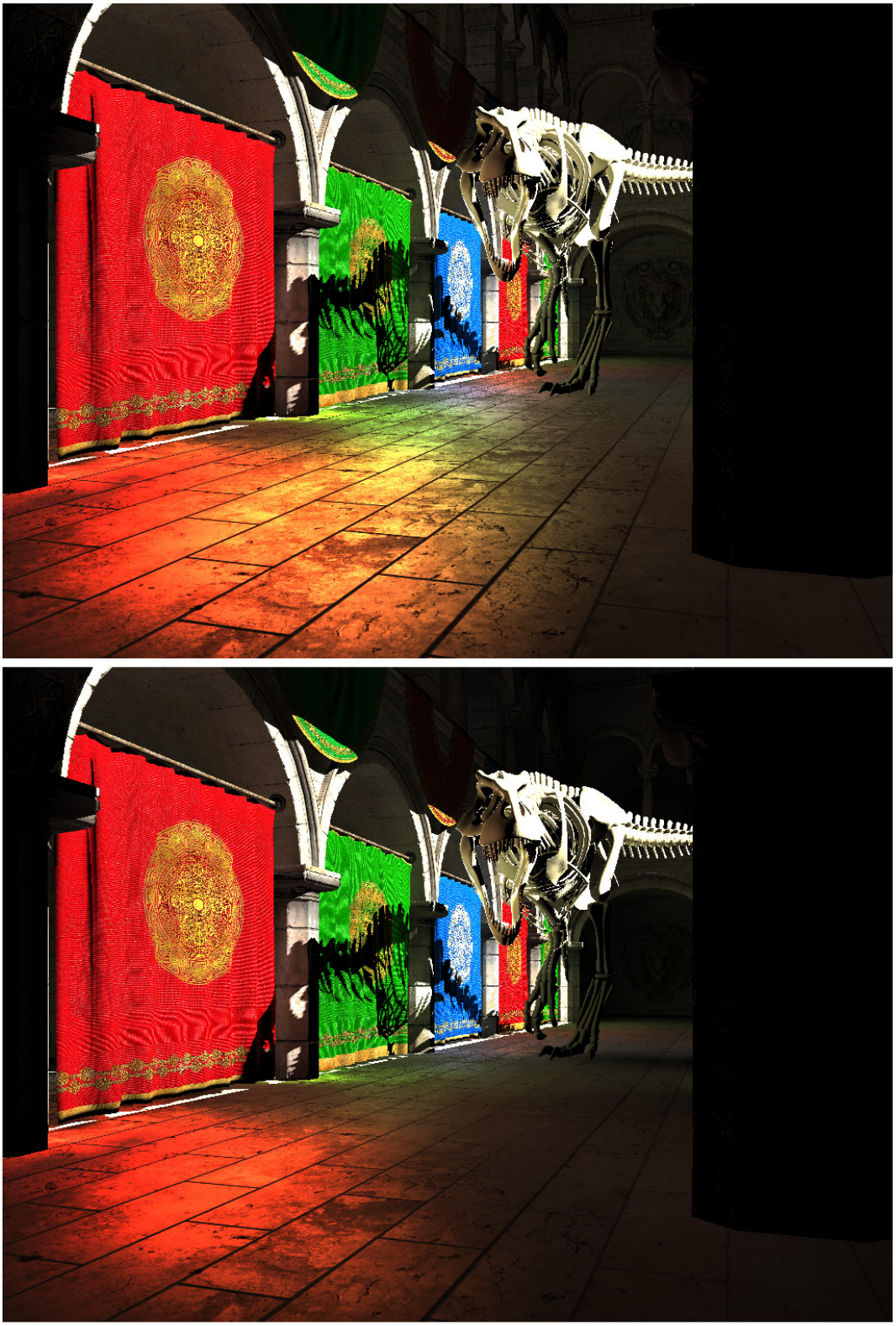

Abbildung 8.11: Vergleich indirekte Reflexionen ohne (oben) und mit weichen Schatten (unten). Die Szene beinhaltet 3118 Caches, wovon 826 zum Kamerabild beitragen. Man erkennt, dass das Dinosaurier-Skelett zu viel indirektes Licht blockiert. Dies hängt mit der Approximation des Skeletts durch Kreisscheiben zusammen.

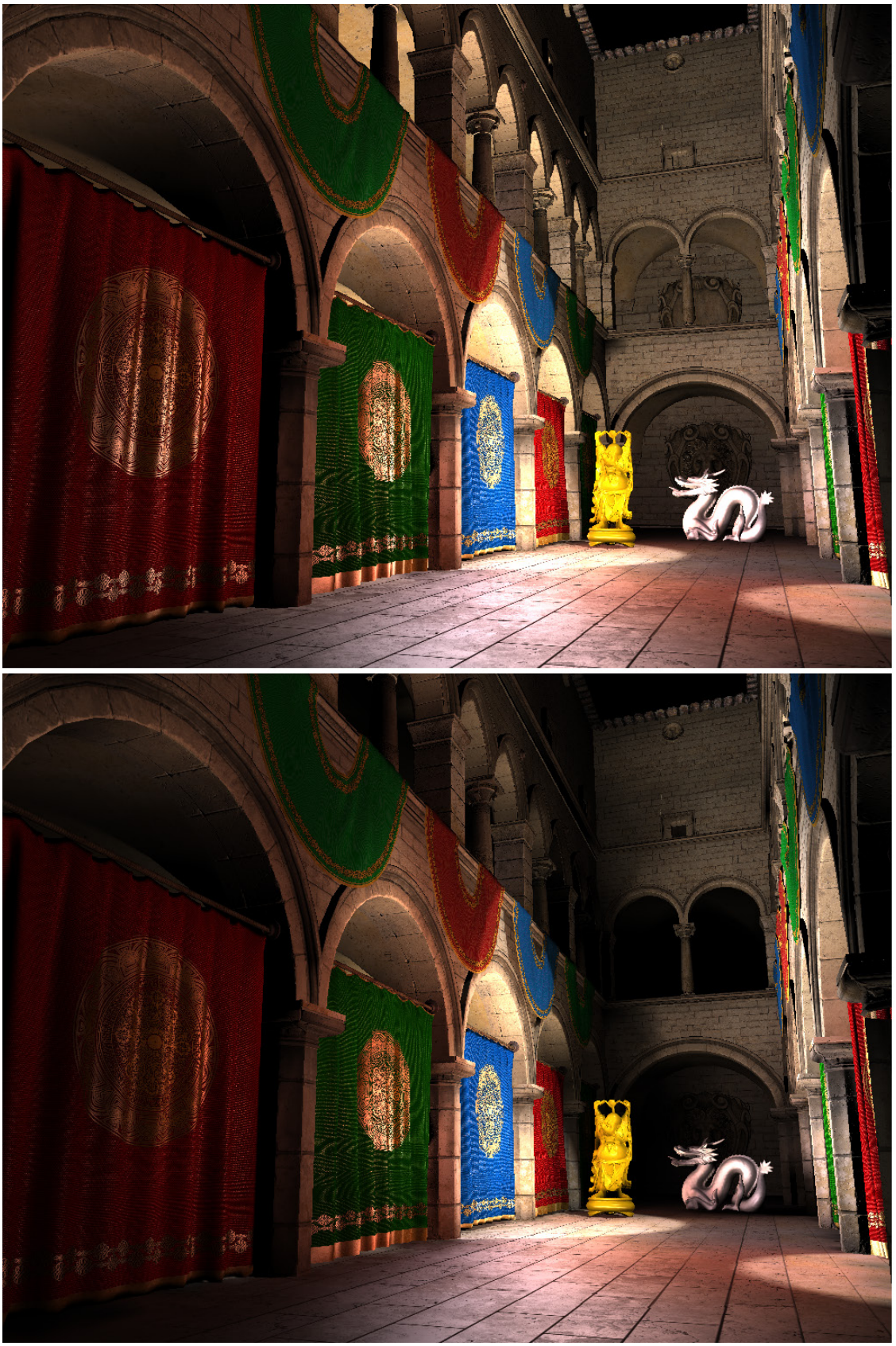

Abbildung 8.12: Vergleich indirekte Reflexionen ohne (oben) und mit weichen Schatten (unten). Die Szene beinhaltet 3.246 Caches, wovon 1.081 zum Kamerabild beitragen.

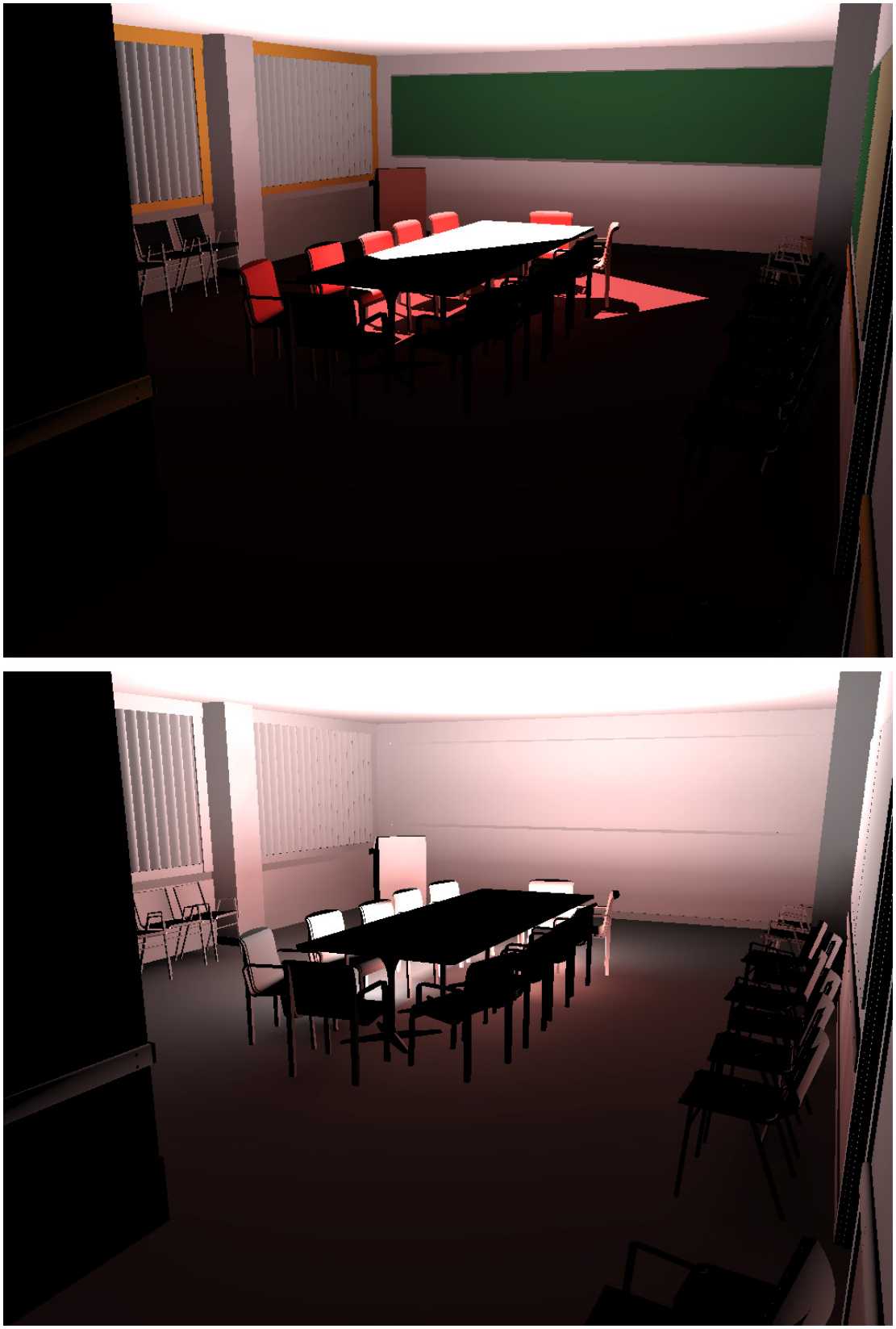

Abbildung 8.13: Indirekte Reflexionen ohne weiche Schatten. Das obere Bild zeigt die Kombination von direkter und indirekter Beleuchtung und das untere nur die indirekte Beleuchtung. Die Szene besteht aus 2.014 Caches.

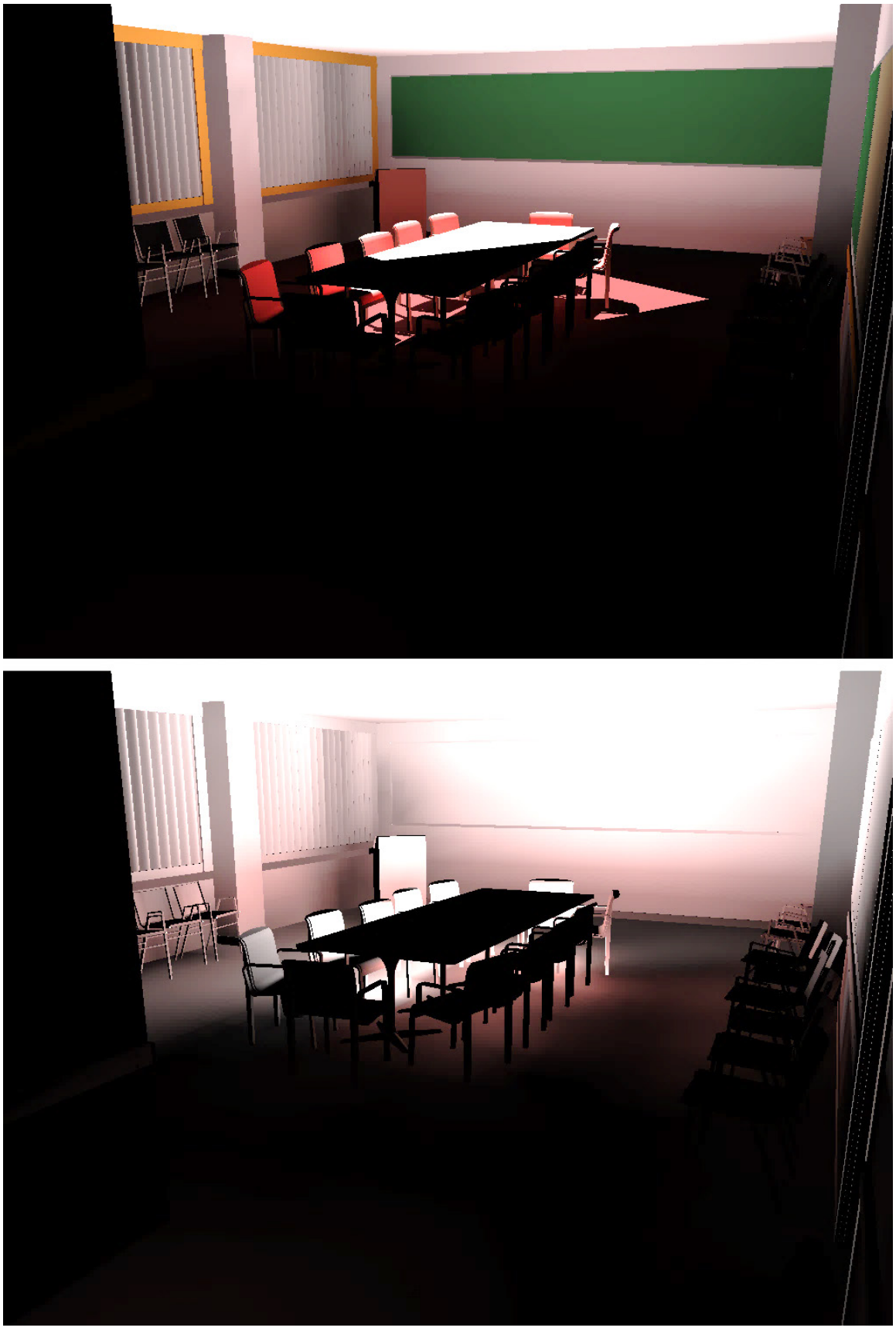

Abbildung 8.14: Indirekte Reflexionen mit weichen Schatten. Das obere Bild zeigt die Kombination von direkter und indirekter Beleuchtung und das untere nur die indirekte Beleuchtung. Die Szene besteht aus 2.014 Caches.

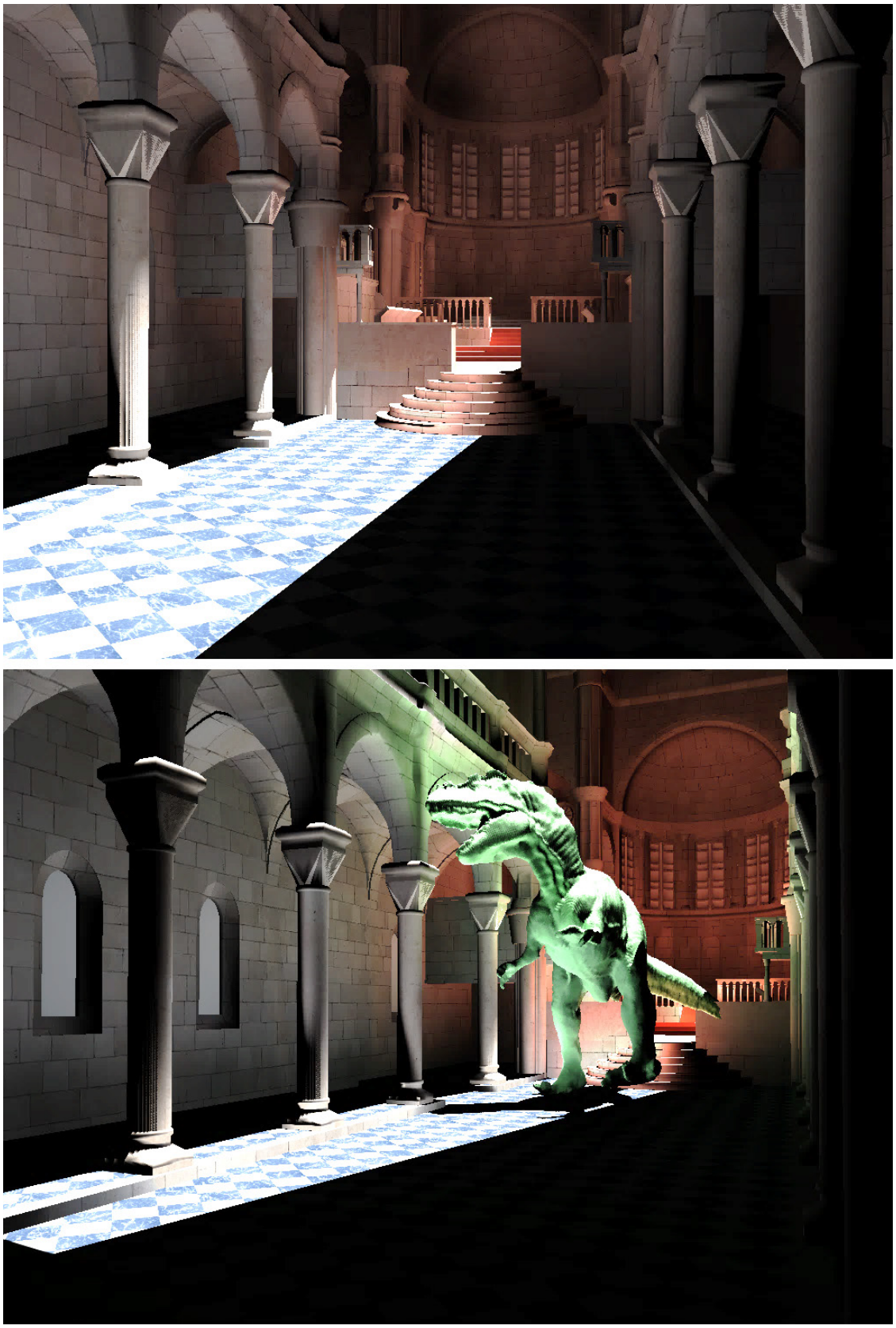

Abbildung 8.15: Sibenik-Cathedral-Szene mit direkten und indirekten Reflexionen. Die obere Szene nutzt 2.810 und die untere 2.950 Caches.

Anhang C Glossar

Absorption	-	Wandlung des Lichts in nicht sichtbare energetische Formen (z. B. Wärme).
Adaptive Cache-Verteilung	-	Permanente Neuverteilung von Caches auf sichtbare Oberflächen im Bildraum der Kamera.
Aktiver Sensor	-	Sensor, der aktiv Veränderungen an der zu messenden Szene vornimmt, um zusätzliche Informationen bereitzustellen.
Albedo	-	„Weißheitsgrad“, diffuser Streuungsgrad. Beim Scattering: Verhältnis des Streuungskoeffizienten zum Absorptionskoeffizienten (σ_s/σ_a). Bei diffusen Reflexionen: Verhältnis des eintreffenden Lichtstroms zum reflektierten Lichtstrom (ϕ_o/ϕ_i).
Ambient Occlusion	-	Erzeugung von weichen Schatten unter der Annahme, dass aus allen Richtungen der gleiche konstante Lichtstrom empfangen wird (ambientes Licht).
Artefakt (optisches)	-	Bildfehler in der Beleuchtung, der vom Betrachter als störend bzw. inkonsistent wahrgenommen wird.
Beleuchtungsstärke	-	Differentieller Lichtstrom pro differentieller Fläche: $E = d\phi/dA$.
Beschleunigungsstruktur	-	Struktur zur effizienten Verarbeitung räumlicher Anfragen, z. B. Octree oder KD-Tree.
Betrachtungsabhängige Reflexionen	-	Glänzende und spiegelnde Reflexionen (die sich mit dem Betrachtungswinkel verändern).
Bidirektionale Reflektanzverteilungsfunktion (BRDF)	-	Beschreibt die Reflexionseigenschaften eines Oberflächenpunktes. Sie gibt das Verhältnis der reflektierten differentiellen Leuchtdichte (dL_o) in Richtung des Betrachters ($\boldsymbol{\omega}$) im Punkt $\boldsymbol{x}$ zur empfangenen differentiellen Beleuchtungsstärke dE aus Richtung ($\boldsymbol{\omega_i}$) an.
Bidirectional-Scattering-Surface-Reflectance-Distribution-Function (BSSRDF)	-	Ist eine Verallgemeinerung der BRDF und beschreibt die Reflexions- und Transmissionseigenschaften für Materialien, die Licht ebenfalls unter die Oberfläche brechen (transluzente Objekte). Sie gibt das Verhältnis der reflektierten differentiellen Leuchtdichte (dL_o) für einen Austrittspunkt ($\boldsymbol{x}$) in Richtung ($\boldsymbol{\omega}$) zum von einem Eintrittspunkt ($\boldsymbol{x_i}$) empfangenen differentiellen Lichtstrom ($d\phi_i$) aus Richtung ($\boldsymbol{\omega_i}$) an.
Bildpyramide	-	Bilderkette eines Ursprungsbildes. Das Ursprungsbild wird in geringer werdenden Auflösungen (Stufen) hierarchisch abgespeichert. In jeder Stufe wird die Breite und Höhe der vorherigen Stufe halbiert. Die Art der Bildfilterung ist abhängig vom gewählten Verfahren.

Bildraum-Cache	-	Caches, die im Bildraum der Kamera verteilt wurden und somit von diesem abhängig sind.
Cache (Light)	-	Caches sind lose verteilte Oberflächenpunkte, die verschiedene Attribute besitzen können. Sie erlauben, aufwändige Berechnungen nur für eine geringe Anzahl von Oberflächenpunkten durchzuführen und für diese kompakte Zwischenergebnisse zu speichern. Anschließend werden die Zwischenergebnisse genutzt, um mit Hilfe eines weniger aufwändigen Interpolationsverfahrens ein durchgehendes Ergebnis für alle sichtbaren Oberflächenpunkte zu erzeugen.
Cache-Culling	-	Verfahren zur Zurückweisung von Caches, die nicht zum fertigen Bild beitragen. Für zurückgewiesene Caches werden keine aufwändigen Berechnungen durchgeführt.
Cache-Textur	-	Ist eine Fließkomma-Textur, die ein spezielles Cache-Attribut speichert (z. B. die Position oder die Normale des Caches oder die Position seiner Proxy-Lichtquelle).
Cache-Texturatlas	-	Speichert Cache-Attribute für alle sichtbaren Modelle einer Szene (die im Sichtfeld liegen).
Culling	-	Zurückweisung von Primitiven, die nicht zum fertigen Bild beitragen.
Deferred Rendering	-	Siehe Deferred Shading.
Deferred Shading	-	Bildraum-Shading-Verfahren [ST90], bei dem geometrische und photometrische Größen einer Szene in mehreren 2D-Buffern gespeichert werden, die die Szene aus der Sicht der Kamera repräsentieren. Die Buffer werden als G-Buffer bezeichnet und beinhalten in der Regel die Position, Normale und Reflexionseigenschaften der sichtbaren Oberflächen in diskreter Pixelform.
Differential Rendering	-	Photometrisches Verfahren zur optisch nahtlosen Integration von synthetischen globalen Beleuchtungsergebnissen in die Aufnahme einer Realumgebung, unter der Voraussetzung, dass die geometrischen und photometrischen Eigenschaften der realen Szene bekannt sind [FGR92].
Diffuse indirekte Reflexion	-	Wiederholte diffuse Reflexion (Lichtpfad: LDD^+E).
Diffuse Reflexion	-	Entsteht durch LD^+E Lichtpfade; bei der ideal diffusen Reflexion wird eintreffendes Licht gleichmäßig in alle Richtungen verteilt (Lambertscher Strahler).

Dipol-Diffusion - Ein Verfahren zur Approximation einer BSSRDF für transluzente Materialien, die hohe Streuungseigenschaften unterhalb der Oberfläche aufweisen (hohes Albedo). Die Grenzbedingung auf der Modelloberfläche wird durch einen analytischen Dipol simuliert, der aus einer negativen (virtuellen) und einer positiven (realen) Punktlichtquelle besteht.

Direkte Beleuchtung - Beleuchtungsergebnisse, die nur das direkte Licht von primären Lichtquellen berücksichtigen, die also den Lichtpfad L(D|S|V)E abbilden.

Direktes Licht - Licht, das von einer primären Lichtquelle entsendet und nicht reflektiert wurde.

Ebenenatlas - Eine Textur beim LightSkin-Verfahren, in der Oberflächenpunkte (Texel) einen Ebenenindex erhalten. Welchen Index ein Texel bekommt, hängt davon ab, ob er in einer geometrisch homogenen oder in einer hochfrequenten Region liegt.

Echtzeitverfahren - Simulationen, die mindestens 30 Bilder pro Sekunde erzeugen.

Final Gathering - Kann im letzten Schritt eines globalen Beleuchtungsverfahrens eingesetzt werden, um die eintreffende Leuchtdichte für jeden sichtbaren Oberflächenpunkt zu bestimmen. Hierfür wird die eintreffende Leuchtdichte in jedem Oberflächenpunkt von den anderen Oberflächen „eingesammelt“ (zum Beispiel durch Strahlenverfolgung).

Formfaktor - Beschreibt den geometrischen Zusammenhang zwischen zwei Oberflächenelementen. Der Formfaktor gibt das Verhältnis des Lichtstroms, der von einem Element i auf ein Element j entsendet wird, zum gesamten Lichtstrom, der auf das Element j eintrifft, an. Hierbei wird vorausgesetzt, dass alle Elemente ideal diffuse Strahler sind. Der Formfaktor liegt immer zwischen null und eins.

Fresnel-Reflexionsterm - Der Fresnel-Reflexionsterm ist das Verhältnis des differentiellen empfangenen Lichtstroms $d\phi_i$ in einem Oberflächenpunkt zum direkt reflektierten Lichtstrom $d\phi_r$ bei transluzenten Objekten: $F_r(\vartheta_i) = d\phi_r / d\phi_i$.

Fresnel-Transmissionsterm - Der Fresnel-Transmissionsterm ist das Verhältnis des differentiellen empfangenen Lichtstroms $d\phi_i$ in einem Oberflächenpunkt zum direkt transmittierten Lichtstrom $d\phi_r$ bei transluzenten Objekten: $F_t(\vartheta_i) = (1 - F_r(\vartheta_i))$.

G-Buffer - Ein 2D-Buffer eines Deferred-Shading-Ansatzes, der Geometrie- oder Materialeigenschaften der sichtbaren Oberflächenelemente enthält (Geometry-Buffer).

Gathering	-	Ausgehend von einem Oberflächenpunkt wird die eintreffende Leuchtdichte von anderen Oberflächen „eingesammelt“, die von diesem Punkt aus sichtbar sind. Das „Einsammeln der Leuchtdichte“ kann auf unterschiedliche Arten erfolgen (z. B. Strahlenverfolgung vom Oberflächenpunkt aus).
Geometrieterm	-	Ein Term aus der Rendergleichung [Kaj86], der die geometrischen Zusammenhänge zwischen zwei differentiellen Oberflächenelementen unter Berücksichtigung der Sichtbarkeit beschreibt: $$G(\boldsymbol{x},\boldsymbol{s}) = \frac{(N(\boldsymbol{s})\cdot -\boldsymbol{\omega_i})^+}{\|\boldsymbol{s}-\boldsymbol{x}\|^2} V(\boldsymbol{x},\boldsymbol{s}).$$
Glänzende indirekte Reflexion	-	Glänzende Reflexion von Licht, welches bereits diffus in der Szene reflektiert wurde (Lichtpfad: LD^+SE).
Glänzende Reflexion	-	Diese Art der Reflexion entsteht bei LD^+S^+E Lichtpfaden. Im Gegensatz zur diffusen Reflexion produziert die glänzende Reflexion häufig hochfrequente Signale, die vom Betrachtungswinkel abhängig sind.
Globale Beleuchtung	-	Beleuchtungssimulation, die die Wechselwirkungen des Lichts zwischen allen Elementen einer Szene berücksichtigt. In dieser Arbeit ist ein globales Beleuchtungsverfahren dadurch definiert, dass es einen Lichtpfad von $L(D\|S\|V)^{++}E$ unterstützt.
High-Dynamic-Range-Darstellung (HDR-Darstellung)	-	Eine hoch kontrastreiche Darstellung, bei der die Lichtberechnung einen hohen dynamischen Kontrast abbilden kann (durch Verwendung von Fließkomma-Texturen und –Rendertargets). Um die kontrastreichen Beleuchtungsergebnisse auf dem Bildschirm darzustellen, werden in der Regel Tone-Mapping-Verfahren genutzt.
Importance-Sampling	-	Verteilung einer festen Anzahl von Stichproben, bei der die Stichproben-Dichte in Bereichen der zu approximierenden Funktion erhöht wird, die voraussichtlich einen besonders hohen Anteil am Integralergebnis beitragen. Hierdurch kann das Ergebnis der Monte-Carlo-Integration verbessert werden.
Indirekte Beleuchtung	-	Beleuchtungsergebnisse, die durch mehrfache Reflexion des Lichts entstehen.
Indirekte Flächenlichtquelle	-	Eine diffus-hemisphärisch abstrahlende (homogene) Flächenlichtquelle, die eine diffuse Reflexion von einer Modelloberfläche approximiert.
Indirektes Licht	-	Licht, das von Oberflächen reflektiert oder transmittiert wurde.
Indirekte Lichtquelle	-	Eine diffuse Lichtquelle, die so parametrisiert wird, dass sie die Lichtausbreitung einer diffusen Reflexion approximiert.
Interaktives Verfahren	-	Beleuchtungsverfahren, das ein Bild innerhalb weniger Sekunden oder schneller berechnet.

In-Scattering - Volumetrische Streuungsprozesse, bei denen Licht in den betrachteten Sichtstrahl hineingestreut wird. Dies führt dazu, dass die Leuchtdichte im betrachteten Sichtstrahl zunimmt.

Irradiance-Caching - Verfahren zur Beschleunigung von globalen Beleuchtungsverfahren [WRC88], bei dem Caches die Beleuchtungsstärke speichern. Mit dem Verfahren können nur diffuse Reflexionen abgebildet werden.

Kaustiken - Kaustiken entstehen, wenn Licht durch eine glänzende oder spiegelnde Oberfläche gebündelt reflektiert oder durch transluzente Objekte gebündelt transmittiert wird und dann auf eine diffuse Oberfläche trifft (Lichtpfad: LD^*S^+DE). Obwohl der Lichtpfad glänzende bzw. spiegelnde Reflexionen enthält, sind Kaustiken betrachtungsunabhängig.

Leuchtdichte - Differentieller Lichtstrom, der von einer differentiellen Fläche in einen differentiellen Raumwinkel entsendet bzw. aus einem differentiellen Raumwinkel empfangen wird: $L = \frac{d^2\phi}{dAd\boldsymbol{\omega}}$.

Lichtpfad - Weg des Lichts von einer primären Lichtquelle zum Auge des Betrachters. Beim Lichtpfad werden die folgenden Wegknoten unterschieden:
L: Beginn des Lichtpfads in der Lichtquelle,
D: diffuse Reflexion,
S: glänzende Reflexion (Specular),
V: Interaktion mit einem Medium und
E: Ankunft des Lichtpfads im Auge des Betrachters.

Lichtstärke - Differentieller Lichtstrom $d\phi$ pro differentiellem Raumwinkel $d\boldsymbol{\omega}$: $I = d\phi/d\boldsymbol{\omega}$.

Light-Clustering - Verfahren, um die Laufzeitkosten für Beleuchtungsverfahren zu reduzieren, die sehr viele individuelle Lichtquellen verwenden. Beim Light-Clustering werden sukzessiv ähnliche Lichtquellen zu einer neuen Lichtquelle zusammengefasst, die das Verhalten der individuellen Lichtquellen approximiert. Hierdurch reduziert sich die Anzahl der zu nutzenden Lichtquellen und somit die Laufzeit des Verfahrens.

Light-Propagation-Volumes - Echtzeit-Discrete-Ordinate-Verfahren von Kaplanyan und Dachsbacher [KD10], bei dem die indirekte Beleuchtung mit Hilfe eines 3D-Gitters berechnet wird. Die Ausbreitung des indirekten Lichts wird iterativ von Nachbarknoten zu Nachbarknoten des Gitters vorgenommen. Das Strahlungsverhalten innerhalb eines Knotens wird durch SH-Koeffizienten abgebildet.

Lokale Beleuchtung - Beleuchtungsverfahren, bei denen nur die Lichtquelle und ein isolierter Oberflächenpunkt berücksichtigt werden.

Mip-Maps - Siehe Bildpyramide.

Modellraum-Cache	-	Bezeichnet einen Cache, der einem Oberflächenpunkt eines Modells fix zugeordnet ist und der mit der Modelloberfläche transformiert wird.
Multiple Subsurface-Scattering	-	Bildet Beleuchtungsergebnisse nach mehrfacher Streuung unter der Oberfläche eines Objekts ab.
Multiresolution Splatting	-	Ein bildraumbasiertes Verfahren zur Erzeugung von indirekten diffusen Reflexionen von Nichols und Wyman [NW09], bei dem effizient sehr viele indirekte Lichtquellen auf die G-Buffer angewendet werden. Anstatt jede indirekte Lichtquelle auf jeden Pixel der G-Buffer anzuwenden, wird eine Bildpyramide genutzt, um eine Untermenge von lose verteilten Pixeln zu erhalten. Die indirekten Lichtquellen werden dann nur auf die lose verteilen Pixel angewendet und die restlichen Pixel erhalten ihre diffusen Reflexionswerte durch ein Interpolationsverfahren.
Nusselts Analogon	-	Das Analogon besagt, dass der Term $(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i}$ aus der Rendergleichung eine Parallelprojektion des differentiellen Raumwinkels $d\boldsymbol{\omega_i}$ auf eine Einheitskreisscheibe beschreibt.
Optisch plausibel	-	Als optisch plausibel sollen die Beleuchtungsergebnisse gelten, die bei dem Betrachter den Eindruck hinterlassen, dass sie konsistent und somit erwartungskonform sind. Verfahren sind nicht optisch plausibel, wenn die Beleuchtungsergebnisse temporär inkohärent sind oder starke kontrastreiche Fehler enthalten.
Out-of-Core-Technik	-	Technik, um effizient große Datenmengen zu verarbeiten, die nicht vollständig im primären (schnellen) Speicher gehalten werden können und deshalb über mehrere Speicher-Kaskaden verteilt liegen.
Out-Scattering	-	Volumetrische Streuungsprozesse, bei denen Licht aus dem betrachteten Sichtstrahl hinaus gestreut wird. Dies führt zu einer Abnahme der Leuchtdichte.
Phasenfunktion	-	Liefert für einen Punkt innerhalb eines Mediums die Verteilung der Wahrscheinlichkeit, ob Licht aus einem differentiellen Raumwinkel um Richtung $\boldsymbol{\omega_i}$ in einen differentiellen Raumwinkel um Richtung $\boldsymbol{\omega}$ gestreut wird.
Photometrische Registrierung	-	Bezeichnet in dieser Arbeit einen Prozess, der die photometrische Interaktion zwischen virtuellen und realen Objekten bei einer AR-Anwendung optisch konsistenter machen soll.

Physikalisch korrekt - In dieser Arbeit gilt eine Simulation als physikalisch korrekt, wenn deren Ergebnisse im betrachteten Kontext auf physikalisch korrekten Grundlagen der Energieerhaltung und der Strahlenoptik (geometrische Optik) basieren und sie absolut zu den Eingangsgrößen sind. Der Kontext wird hier durch das Phänomen bestimmt. Wellenoptische und quantenoptische physikalische Gesetze, sowie farbmetrische Zusammenhänge werden bei dieser Betrachtung außen vor gelassen.

Physikalisch plausibel - Ein Beleuchtungsverfahren ist in dieser Arbeit physikalisch plausibel, wenn die verwendeten BRDF- und BSSRDF-Modelle die Gesetze der Energieerhaltung korrekt abbilden, die erzeugten Beleuchtungsgradienten optisch plausibel sind und der Geometrieterm aus der Rendergleichung korrekt abgebildet wird (wobei die Sichtbarkeitsfunktion V vernachlässigt werden kann).

Primäre Lichtquelle - Eine Lichtquelle, die in der Szene existiert und nicht aufgrund von Reflexionen oder Transmissionen algorithmisch erzeugt wurde. Die primäre Lichtquelle nimmt die initiale Verteilung des Lichts in der Szene vor (direktes Licht).

Proxy-Dipol - Ein Proxy-Dipol wird beim LightSkin-Verfahren für jeden Cache erstellt. Er fasst die einzelnen Diffusions-Dipole, die nach dem BSSRDF-Modell von Jensen et al. [JML*01] für jeden Eintrittspunkt des Lichts unter der Modelloberfläche eines SSS-Modells erzeugt werden, zusammen. Der Proxy-Dipol besteht aus einer realen und einer virtuellen Dipol-Lichtquelle, die so parametrisiert werden, dass die Anwendung des Proxy-Dipols auf den Cache zu der gleichen spezifischen Lichtausstrahlung führt, wie die Anwendung der einzelnen Diffusions-Dipole.

Proxy-Dipol-Lichtquelle - Der Proxy-Dipol besteht aus zwei Proxy-Dipol-Lichtquellen: einer positiven (realen) unter der Oberfläche und einer negativen (virtuellen) über der Oberfläche.

Proxy-Lichtquelle - Ein Cache kann beim LightSkin-Verfahren eine diffuse und eine glänzende punktförmige Proxy-Lichtquelle besitzen. Eine Proxy-Lichtquelle fasst mehrere auf den Cache einwirkende VALs zusammen und wird so parametrisiert, dass ihre Anwendung auf den Cache zum gleichen Beleuchtungsergebnis führt wie die Anwendung der individuellen VALs. Für die Parametrisierung der Proxy-Lichtquellen werden die Reflexionseigenschaften im Cache genutzt. Die Proxy-Lichtquelle steht also stellvertretend (Proxy) für mehrere VALs.

Proxy-Objekt - Ein Objekt, welches stellvertretend für ein anderes Objekt für komplexe Berechnungen genutzt werden kann. Hierbei ist das Proxy-Objekt häufig wesentlich weniger komplex als das eigentliche Objekt, was zu einem besseren Laufzeitverhalten bei umfangreichen Berechnungen führt.

Räumliche Registrierung - Bezeichnet hier die Zuordnung von virtuellen Objekten zu einer dreidimensionalen Position und Orientierung in einer realen Szene bei einer AR-Anwendung.

Radiance-Caching - Verfahren zur Beschleunigung von globalen Beleuchtungsverfahren [KGP*05], bei dem die Caches die eintreffende Leuchtdichte richtungsabhängig durch Spherical-Harmonics-Koeffizienten repräsentieren. Hierdurch können bei der Interpolation betrachtungsabhängige Reflexionen abgebildet werden.

Reale Dipol-Lichtquelle - Eine der beiden Dipol-Diffusions-Lichtquellen. Die reale Lichtquelle ist die positive Lichtquelle, die unter der SSS-Materialoberfläche liegt.

Reflective Shadow-Map (RSM) - Erweiterung einer klassischen Shadow-Map, bei der anstatt eines einfachen Tiefenbuffers aus Sicht der Lichtquelle drei Buffer erzeugt werden. Ein Buffer enthält die Positionen, einer die Normalen und einer die diffusen Reflexionseigenschaften der gesehenen Oberflächenpunkte. Ein Pixel, gelesen aus den drei Buffern, bildet ein VPL ab.

Reflexionsexponent - Bezeichnet in dieser Arbeit den Phong-Reflexionsexponenten k, der die Bündelung der Reflexion entlang des Spiegelvektors steuert.

Rendergleichung - Die Rendergleichung [Kaj86] beschreibt rekursiv, wie die Leuchtdichte für einen Oberflächenpunkt $\boldsymbol{x}$ in Richtung $\boldsymbol{\omega}$ unter Berücksichtigung globaler Wechselwirkungen berechnet werden kann:

$$L_r(\boldsymbol{x}, \boldsymbol{\omega}) = \int_{\Omega^+} f_r(\boldsymbol{x}, \boldsymbol{\omega}, \boldsymbol{\omega_i}) L_i\,(\boldsymbol{x}, \boldsymbol{\omega_i})\,(N(\boldsymbol{x}) \cdot \boldsymbol{\omega_i})^+ d\boldsymbol{\omega_i}.$$

Sichtbarkeitsfunktion - Eine binäre Funktion, die zu eins evaluiert, wenn Sichtbarkeit zwischen zwei Punkten besteht und zu null, wenn nicht. Die Sichtbarkeitsfunktion ist Teil des Geometrieterms der modifizierten Rendergleichung.

Single Subsurface-Scattering - Bildet Beleuchtungsergebnisse nach einmaligem Streuungsereignis unter der Oberfläche eines transluzenten Objekts ab.

Spherical Harmonics (SH) - Orthonormale Basisfunktionen basierend auf Legendre-Polynomen, die über eine Einheitskugel definiert sind. Mit diesen Funktionen lässt sich zum Beispiel die eintreffende Leuchtdichte über eine Sphäre vom Orts- in den Frequenzraum transformieren und approximativ durch Spherical-Harmonics-Koeffizienten abbilden. Durch die Transformation in den Frequenzraum können Faltungen effizient durch Skalarprodukte der Koeffizienten abgebildet werden.

Spiegelnde Reflexion	-	Betrachtungsabhängige Reflexion, die im Gegensatz zur glänzenden Reflexion wie ein Spiegel wirkt und deutlich hochfrequente Details der gespiegelten Szene erkennen lässt.
Splatting	-	Eine Technik, die in Deferred-Shading-Systemen genutzt wird, um eine direkte oder indirekte Lichtquelle auf eine Menge von Pixeln in den G-Buffern anzuwenden. Hierfür erhält jede Lichtquelle ein geometrisches Modell, welches den räumlichen Einfluss der Lichtquelle begrenzt. Das Modell wird in die G-Buffer gerastert. Jeder Pixel, der von dem Modell verdeckt wird, wird dann im Fragment-Shader durch die Lichtquelle beleuchtet.
Subsurface-Scattering (SSS)	-	Berechnung von Streuungseffekten unter der Oberfläche eines transluzenten Modells. Modelle, die SSS unterstützen, verwenden anstatt einer BRDF eine BSSRDF.
Surfel	-	Ein endliches (approximatives) Flächen-Element.
Temporäre Inkohärenz	-	Flackernde Artefakte, die durch Animationen entstehen können und die Beleuchtung temporär inkonsistent erscheinen lassen.
Translucent Shadow-Map	-	Erweiterung einer klassischen Shadow-Map, bei der anstatt eines einfachen Tiefenbuffers aus Sicht der Lichtquelle drei Buffer erzeugt werden. Ein Buffer enthält die Positionen, einer die Normalen und einer die diffusen Transmissionsseigenschaften der gesehenen Oberflächenpunkte. Ein Pixel, aus den drei Buffern gelesen, bildet ein VPL unter der Modelloberfläche ab.
Verdeckungsgrad	-	Der Verdeckungsgrad gibt approximativ für einen Oberflächenpunkt an, wie viel Prozent des einfallenden indirekten Lichts durch andere Objekte blockiert wird. Ist er null, wird das komplette indirekte Licht blockiert und bei eins wird kein Licht blockiert.
Verdeckungsrechnung	-	Betrachtung, ob Licht zwischen zwei Oberflächenpunkten durch andere Objekte blockiert wird, um weiche Schatten zu erzeugen.
Virtual Area-Light (VAL)	-	Siehe indirekte Flächenlichtquelle. Beim LightSkin-Verfahren ist ein VAL eine homogene hemisphärisch-diffus abstrahlende Kreisflächenlichtquelle, die aus einem Pixel der erweiterten Reflective Shadow-Map resultiert.
Virtual Point-Light (VPL)	-	Eine punktförmige hemisphärisch strahlende virtuelle Lichtquelle. Der Begriff des VPL wird bei den RSMs und bei Instant-Radiosity-Ansätzen verwendet.
Virtuelle Dipol-Lichtquelle	-	Eine der beiden Dipol-Diffusions-Lichtquellen. Die virtuelle Lichtquelle ist die negative Lichtquelle, die über der SSS-Materialoberfläche liegt.
Virtuelle Flächenlichtquelle	-	Siehe indirekte Flächenlichtquelle.

Virtuelle Lichtquelle	-	Siehe indirekte Lichtquelle. Der Begriff der virtuellen Lichtquelle hat sich bei Instant-Radiosity-Verfahren etabliert.
Voxel-Cone-Tracing (VCT)	-	Eine echtzeitfähige Finite-Elements-Methode von Crassin et al. [CNS*11], um optisch plausible globale Beleuchtungseffekte zu erzeugen. Beim VCT werden die Oberflächen einer Szene durch einen Voxel-Octree approximiert. In diesen Voxel-Octree wird indirektes Licht mit Hilfe von RSMs injiziert, das später für die indirekte Beleuchtung eines sichtbaren Oberflächenpunkts genutzt werden kann, indem Stichproben aus den verschiedenen Ebenen des Octrees entnommen werden.

Anhang D Abbildungsverzeichnis

Anhang E Literaturverzeichnis

[AM04] Ababsa, Fakhr-eddine; Mallem, Malik. Robust Camera Pose Estimation Using 2D Fiducials Tracking for Real-time Augmented Reality Systems. Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry. ACM. New York, NY, USA. Seiten 431–435. 2004.

[ARH*00] Agrawala, Maneesh; Ramamoorthi, Ravi; Heirich, Alan; Moll, Laurent. Efficient Image-based Methods for Rendering Soft Shadows. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA. Seiten 375–384. 2000.

[AUW07] Akerlund, Oskar; Unger, Mattias; Wang, Rui. Precomputed Visibility Cuts for Interactive Relighting with Dynamic BRDFs. Proceedings of the 15th Pacific Conference on Computer Graphics and Applications. IEEE Computer Society. Washington, DC, USA. Seiten 161–170. 2007.

[AKD*04] Annen, Thomas; Kautz, Jan; Durand, Frédo; Seidel, Hans-Peter. Spherical harmonic Gradients for Mid-Range Illumination. Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques (EGSR'04). Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 331–336. 2004.

[Azu97] Azuma, R. T. 'A Survey of Augmented Reality', (6, 4). Presence: Teleoperators and Virtual Environments. Jg. 6. H. 4. Seiten 355–385. 1997.

[BSD08] Bavoil, Louis; Sainz, Miguel; Dimitrov, Rouslan. Image-space Horizon-based Ambient Occlusion. ACM SIGGRAPH 2008 Talks. ACM. New York, NY, USA. Seiten 22:1-22:1. 2008.

[Ben75] Bentley, Jon Louis. Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM. Jg. 18. H. 9. Seiten 509–517. 1975.

[BLS93] Blasi, Philippe; Le Saec, Bertrand; Schlick, Christophe. A Rendering Algorithm for Discrete Volume Density Objects. Computer Graphics Forum. Jg. 12. H. 3. Seiten 201–210. 1993.

[Bli77] Blinn, James F. Models of Light Reflection for Computer Synthesized Pictures. SIGGRAPH Comput. Graph. Jg. 11. H. 2. Seiten 192–198. 1977.

[BN76] Blinn, James F.; Newell, Martin E. Texture and Reflection in Computer Generated Images. Commun. ACM. Jg. 19. H. 10. Seiten 542–547. 1976.

[BL03] Borshukov, George; Lewis, J. P. Realistic Human Face Rendering for The Matrix Reloaded. ACM SIGGRAPH 2003 Sketches and Applications. ACM. New York, NY, USA. Seiten 1. 2003.

[BL05] Borshukov, George; Lewis, J. P. Fast Subsurface Scattering. ACM SIGGRAPH 2005 Courses. ACM. New York, NY, USA. 2005.

[BBN*07] Bouthors, Antoine; Bruneton, Eric; Neyret, Fabrice; Max, Nelson. Real-time subsurface scattering on the GPU. Technical Report. INRIA, Grenoble, France. 2007.

[Buc05] Buck, Ian. Taking the Plunge into GPU Computing. Fernando, Randima; Pharr, Matt. Programming techniques for high-performance graphics and general-purpose computation. Addison-Wesley. Upper Saddle River, NJ [u.a.]. Seiten 509–519. 2005.

[Bun05] Bunnell, M. Dynamic Ambient Occlusion and Indirect Lighting. Fernando, Randima; Pharr, Matt. Programming techniques for high-performance graphics and general-purpose computation. Addison-Wesley. Upper Saddle River, NJ [u.a.]. Seiten 223–233. 2005.

[Cha50] Chandrasekar, S. Radiative Transfer. Oxford Univ. Press. New York. ISBN: 0486605906. 1950.

[Chr08] Christensen, Per. Point-Based Approximate Color Bleeding. PIXAR Technical Memo #08-01. 2008.

[CJA*05] Clarberg, Petrik; Jarosz, Wojciech; Akenine-Möller, Tomas; Jensen, Henrik Wann. Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. ACM Trans. Graph. Jg. 24. H. 3. Seiten 1166–1175. 2005.

[CCW*88] Cohen, Michael F.; Chen, Shenchang Eric; Wallace, John R.; Greenberg, Donald P. A Progressive Refinement Approach to Fast Radiosity Image Generation. SIGGRAPH Comput. Graph. Jg. 22. H. 4. Seiten 75–84. 1988.

[CG85] Cohen, Michael F.; Greenberg, Donald P. The Hemi-cube: A Radiosity Solution for Complex Environments. Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. Seiten 31–40. 1985.

[CT81] Cook, Robert L.; Torrance, Kenneth E. A Reflectance Model for Computer Graphics. SIGGRAPH Comput. Graph. Jg. 15. H. 3. Seiten 307–316. 1981.

[CHL04] Coombe, Greg; Harris, Mark J.; Lastra, Anselmo. Radiosity on Graphics Hardware. Proceedings of the 2004 Graphics Interface Conference. Canadian Human-Computer Communications Society. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Seiten 161–168. 2004.

[Cra11] Crassin, Cyril. GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes. PHD Thesis. UNIVERSITE DE GRENOBLE. Grenoble, France. 2011.

[CNL*09] Crassin, Cyril; Neyret, Fabrice; Lefebvre, Sylvain; Eisemann, Elmar. GigaVoxels: Ray-guided Streaming for Efficient and Detailed Voxel Rendering. Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 15–22. 2009.

[CNS*11] Crassin, Cyril; Neyret, Fabrice; Sainz, Miguel; Green, Simon; Eisemann, Elmar. Interactive Indirect Illumination Using Voxel Cone Tracing. Computer Graphics Forum. Jg. 30. H. 7. Seiten 1921–1930. 2011.

[Cro77] Crow, Franklin C. Shadow Algorithms for Computer Graphics. SIGGRAPH Comput. Graph. Jg. 11. H. 2. Seiten 242–248. 1977.

[CSD*92] Cruz-Neira, Carolina; Sandin, Daniel J.; DeFanti, Thomas A.; Kenyon, Robert V.; Hart, John C. The CAVE: Audio Visual Experience Automatic Virtual Environment. Commun. ACM. Jg. 35. H. 6. Seiten 64–72. 1992.

[DS03] Dachsbacher, Carsten; Stamminger, Marc. Translucent Shadow Maps. Proceedings of the 14th Eurographics Workshop on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 197–201. 2003.

[DS05] Dachsbacher, Carsten; Stamminger, Marc. Reflective Shadow Maps. Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 203–231. 2005.

[DS06] Dachsbacher, Carsten; Stamminger, Marc. Splatting Indirect Illumination. Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 93–100. 2006.

[DSD*07] Dachsbacher, Carsten; Stamminger, Marc; Drettakis, George; Durand, Frédo. Implicit Visibility and Antiradiance for Interactive Global Illumination. ACM Trans. Graph. Jg. 26. H. 3. 2007.

[Deb08a] Debevec, Paul. A Median Cut Algorithm for Light Probe Sampling. ACM SIGGRAPH 2008 Classes. ACM. New York, NY, USA. Seiten 33:1-33:3. 2008.

[Deb08b] Debevec, Paul. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. ACM SIGGRAPH 2008 Classes. ACM. New York, NY, USA. Seiten 32:1-32:10. 2008.

[DM97] Debevec, Paul E.; Malik, Jitendra. Recovering High Dynamic Range Radiance Maps from Photographs. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA. Seiten 369–378. 1997.

[Der93] Deriche, Rachid. Recursively implementating the Gaussian and its derivatives. Research Report 1893. INRIA. 1993.

[DBM*02] Dmitriev, Kirill; Brabec, Stefan; Myszkowski, Karol; Seidel, Hans-Peter. Interactive Global Illumination Using Selective Photon Tracing. Proceedings of the 13th Eurographics Workshop on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 25–36. 2002.

[Dog12] Doggett, Michael. Texture Caches. IEEE Micro. Jg. 32. H. 3. Seiten 136–141. 2012.

[DGR*09] Dong, Zhao; Grosch, Thorsten; Ritschel, Tobias; Kautz, Jan; Seidel, Hans-Peter. Real-time Indirect Illumination with Clustered Visibility. Vision, Modeling, and Visualization Workshop. Seiten 187–196. 2009.

[DKT*07] Dong, Zhao; Kautz, Jan; Theobalt, Christian; Seidel, Hans-Peter. Interactive Global Illumination Using Implicit Visibility. Proceedings of the 15th Pacific Conference on Computer Graphics and Applications. IEEE Computer Society. Washington, DC, USA. Seiten 77–86. 2007.

[Don06] Donner, Craig Steven. Ph.D Thesis: Towards Realistic Image Synthesis of Scattering Materials. University of California, San Diego. ISBN: 978-0-542-77805-6. 2006.

[EH79] Egan, W. G.; Hilgeman, T. W. Optical properties of inhomogeneous materials: applications to geology, astronomy, chemistry, and engineering. Elsevier Science & Technology. Oxford. ISBN: 9780122326509. 1979.

[EAS*12] Eisemann, Elmar; Assarsson, Ulf; Schwarz, Michael; Valient, Michal; Wimmer, Michael. Efficient Real-time Shadows. ACM SIGGRAPH 2012 Courses. ACM. New York, NY, USA. Seiten 18:1-18:53. 2012.

[EAS*09] Eisemann, Elmar; Assarsson, Ulf; Schwarz, Michael; Wimmer, Michael. Casting Shadows in Real Time. ACM SIGGRAPH ASIA 2009 Courses. ACM. New York, NY, USA. 2009.

[Eng09] Engel, Wolfgang. The light pre-pass renderer: Renderer design for efficient support of multiple lights. Engel, Wolfgang. ShaderX 7. Course Technology. Boston, Mass. Seiten 655–666. 2009.

[FD09] Fabianowski, B.; Dingliana, J. Interactive Global Photon Mapping. Computer Graphics Forum. Jg. 28. H. 4. Seiten 1151–1159. 2009.

[FPW92] Farrell, T. J.; Patterson, M. S.; Wilson, B. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo. Medical Physics. Jg. 19. H. 4. Seiten 879–888. 1992.

[Fat09] Fattal, Raanan. Participating Media Illumination Using Light Propagation Maps. ACM Trans. Graph. Jg. 28. H. 1. Seiten 7:1-7:11. 2009.

[Fic55] Fick, A. Ueber diffusion. Annalen der Physik. Jg. 170. H. 1. Seiten 59–86. 1855.

[FGR92] Fournier, Alain; Gunawan, Atjeng S.; Romanzin, Chris. Common Illumination Between Real and Computer Generated Scenes. Technical Report. University of British Columbia. Vancouver, BC, Canada. 1992.

[Fra13a] Franke, Tobias Alexander. Delta Light Propagation Volumes for Mixed Reality. IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2013. IEEE Computer Society. Washington, DC, USA. Seiten 125–132. 2013.

[Fra13b] Franke, Tobias Alexander. Near-field Illumination for Mixed Reality with Delta Radiance Fields. ACM SIGGRAPH 2013 Posters. ACM. New York, NY, USA. Seiten 76:1-76:1. 2013.

[GHT*03] Gao, Xiao-Shan; Hou, Xiao-Rong; Tang, Jianliang; Cheng, Hang-Fei. Complete Solution Classification for the Perspective-Three-Point Problem. IEEE Trans. Pattern Anal. Mach. Intell. Jg. 25. H. 8. Seiten 930–943. 2003.

[GBP08] Gautron, Pascal; Bouatouch, Kadi; Pattanaik, Sumanta. Temporal Radiance Caching. ACM SIGGRAPH 2008 Classes. ACM. New York, NY, USA. Seiten 79:1-79:29. 2008.

[GKP*04] Gautron, Pascal; Krivanek, Jaroslav; Pattanaik, Sumanta; Bouatouch, Kadi. A Novel Hemispherical Basis for Accurate and Efficient Rendering. Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 321–330. 2004.

[GKB*05] Gautron, Pascal; Křivánek, Jaroslav; Bouatouch, Kadi; Pattanaik, Sumanta. Radiance Cache Splatting: A GPU-friendly Global Illumination Algorithm. ACM SIGGRAPH 2005 Sketches. ACM. New York, NY, USA. 2005.

[GRW*04] Geist, Robert; Rasche, Karl; Westall, James; Schalkoff, Robert J. Lattice-Boltzmann Lighting. Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 355–362. 2004.

[Gil79] Gilchrist, A. The perception of surface blacks and whites. Scientific American. Jg. 24. H. 3. Seiten 88–97. 1979.

[GTG*84] Goral, Cindy M.; Torrance, Kenneth E.; Greenberg, Donald P.; Battaile, Bennett. Modeling the Interaction of Light Between Diffuse Surfaces. SIGGRAPH Comput. Graph. Jg. 18. H. 3. Seiten 213–222. 1984.

[GKD07] Green, Paul; Kautz, Jan; Durand, Frédo. Efficient Reflectance and Visibility Approximations for Environment Map Rendering. Computer Graphics Forum. Jg. 26. H. 3. Seiten 495–502. 2007.

[GKM*06] Green, Paul; Kautz, Jan; Matusik, Wojciech; Durand, Frédo. View-dependent Precomputed Light Transport Using Nonlinear Gaussian Function Approximations. Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 7–14. 2006.

[Gre86] Greene, Ned. Environment Mapping and Other Applications of World Projections. IEEE Comput. Graph. Appl. Jg. 6. H. 11. Seiten 21–29. 1986.

[GSH*98] Greger, Gene; Shirley, Peter; Hubbard, Philip M.; Greenberg, Donald P. The Irradiance Volume. IEEE Comput. Graph. Appl. Jg. 18. H. 2. Seiten 32–43. 1998.

[GEM07] Grosch, Thorsten; Eble, Tobias; Mueller, Stefan. Consistent Interactive Augmentation of Live Camera Images with Correct Near-field Illumination. Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology. ACM. New York, NY, USA. Seiten 125–132. 2007.

[HJB*12] Hachisuka, Toshiya; Jarosz, Wojciech; Bouchard, Guillaume; Christensen, Per; Frisvad, Jeppe Revall; Jakob, Wenzel; Jensen, Henrik Wann; Kaschalk, Michael; Knaus, Claude; Selle, Andrew; Spencer, Ben. State of the Art in Photon Density Estimation. ACM SIGGRAPH 2012 Courses. ACM. New York, NY, USA. Seiten 6:1-6:469. 2012.

[HPT*06] Häkkinen, Jukka; Pölönen, Monika; Takatalo, Jari; Nyman, Göte. Simulator Sickness in Virtual Display Gaming: A Comparison of Stereoscopic and Non-stereoscopic Situations. Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services. ACM. New York, NY, USA. Seiten 227–230. 2006.

[HPB07] Hašan, Miloš; Pellacini, Fabio; Bala, Kavita. Matrix Row-column Sampling for the Many-light Problem. ACM SIGGRAPH 2007 Papers. ACM. New York, NY, USA. 2007.

[HST10] He, Kaiming; Sun, Jian; Tang, Xiaoou. Guided Image Filtering. Proceedings of the 11th European Conference on Computer Vision: Part I. Springer-Verlag. Berlin, Heidelberg. Seiten 1–14. 2010.

[Hec90] Heckbert, Paul S. Adaptive Radiosity Textures for Bidirectional Ray Tracing. Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. Seiten 145–154. 1990.

[HG41] Henyey, L. G.; Greenstein, J. L. Diffuse radiation in the Galaxy. Astrophysical Journal, vol. 93. Jg. 93. Seiten 70–83. 1941.

[HW59] Hicks, J. S.; Wheeling, R. F. An Efficient Method for Generating Uniformly Distributed Points on the Surface of an N-dimensional Sphere. Commun. ACM. Jg. 2. H. 4. Seiten 17–19. 1959.

[HRE*11] Holländer, Matthias; Ritschel, Tobias; Eisemann, Elmar; Boubekeur, Tamy. ManyLoDs: Parallel Many-view Level-of-detail Selection for Real-time Global Illumination. Proceedings of the Twenty-second Eurographics Conference on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 1233–1240. 2011.

[Ish78] Ishimaru, A. Wave Propagation and Scattering in Random Media. Academic Press. ISBN: 9780123747020. 1978.

[IDY*07] Iwasaki, Kei; Dobashi, Yoshinori; Yoshimoto, Fujiichi; Nishita, Tomoyuki. Precomputed Radiance Transfer for Dynamic Scenes Taking into Account Light Interreflection.

Proceedings of the 18th Eurographics Conference on Rendering Techniques. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 35–44. 2007.

[IKH*11] Izadi, Shahram; Kim, David; Hilliges, Otmar; Molyneaux, David; Newcombe, Richard; Kohli, Pushmeet; Shotton, Jamie; Hodges, Steve; Freeman, Dustin; Davison, Andrew; Fitzgibbon, Andrew. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. ACM. New York, NY, USA. Seiten 559–568. 2011.

[Jar08] Jarosz, Wojciech. Efficient Monte Carlo Methods for Light Transport in Scattering Media. PHD Thesis. UC San Diego. San Diego, California, USA. 2008.

[Jen96] Jensen, Henrik Wann. Global Illumination Using Photon Maps. Proceedings of the Eurographics Workshop on Rendering Techniques ’96. Springer-Verlag. London, UK, UK. Seiten 21–30. 1996.

[JML*01] Jensen, Henrik Wann; Marschner, Stephen R.; Levoy, Marc; Hanrahan, Pat. A Practical Model for Subsurface Light Transport. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. Seiten 511–518. 2001.

[Kaj86] Kajiya, James T. The Rendering Equation. Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. Seiten 143–150. 1986.

[KK13] Kán, Peter; Kaufmann, Hannes. Differential Irradiance Caching for Fast High-Quality Light Transport Between Virtual and Real Worlds. Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR). IEEE Computer Society. Seiten 133–141. 2013.

[KD10] Kaplanyan, Anton; Dachsbacher, Carsten. Cascaded Light Propagation Volumes for Real-time Indirect Illumination. Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 99–107. 2010.

[KSS02] Kautz, Jan; Sloan, Peter-Pike; Snyder, John. Fast, Arbitrary BRDF Shading for Low-frequency Lighting Using Spherical Harmonics. Proceedings of the 13th Eurographics Workshop on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 291–296. 2002.

[Kel97] Keller, Alexander. Instant Radiosity. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA. Seiten 49–56. 1997.

[KLO07] Ki, Hyunwoo; Lyu, Jihye; Oh, Kyoungsu. Real-Time Approximate Subsurface Scattering on Graphics Hardware. Proceedings of the 15th Pacific Conference on Computer Graphics and Applications. IEEE Computer Society. Washington, DC, USA. Seiten 403–406. 2007.

[KO08] Ki, Hyunwoo; Oh, Kyoungsu. A GPU-based Light Hierarchy for Real-time Approximate Illumination. Vis. Comput. Jg. 24. H. 7. Seiten 649–658. 2008.

[KM07] Klein, Georg; Murray, David. Parallel Tracking and Mapping for Small AR Workspaces. Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. IEEE Computer Society. Washington, DC, USA. Seiten 1–10. 2007.

[KTM*10] Knecht, Martin; Traxler, Christoph; Mattausch, Oliver; Purgathofer, Werner; Wimmer, Michael. Differential Instant Radiosity for Mixed Reality. Proceedings of the 2010 IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2010). IEEE Computer Society. Washington, DC, USA. Seiten 99–107. 2010.

[KTM*12] Knecht, Martin; Traxler, Christoph; Mattausch, Oliver; Wimmer, Michael. Augmented Reality: Reciprocal Shading for Mixed Reality. Comput. Graph. Jg. 36. H. 7. Seiten 846–856. 2012.

[KTW*13] Knecht, Martin; Traxler, Christoph; Winklhofer, Christoph; Wimmer, Michael. Reflective and Refractive Objects for Mixed Reality. IEEE Transactions Visualization and Computer Graphics. Jg. 19. H. 4. Seiten 576–582. 2013.

[Kno65] Knowlton, Kenneth C. A Fast Storage Allocator. Commun. ACM. Jg. 8. H. 10. Seiten 623–624. 1965.

[KGP*05] Krivanek, J.; Gautron, P.; Pattanaik, S.; Bouatouch, K. Radiance caching for efficient global illumination computation. IEEE Transactions on Visualization and Computer Graphics. Jg. 11. H. 5. Seiten 550–561. 2005.

[KFB10] Křivánek, Jaroslav; Ferwerda, James A.; Bala, Kavita. Effects of Global Illumination Approximations on Material Appearance. ACM SIGGRAPH 2010 Papers. ACM. New York, NY, USA. Seiten 112:1-112:10. 2010.

[KBW06] Krüger, Jens; Bürger, Kai; Westermann, Rüdiger. Interactive Screen-Space Accurate Photon Tracing on GPUs. Rendering Techniques (Eurographics Symposium on Rendering - EGSR). Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 319–329. 2006.

[KGX*11] Kun Zhou; Gong, Minmin; Xin Huang; Baining Guo. Data-Parallel Octrees for Surface Reconstruction. IEEE Transactions on Visualization and Computer Graphics. Jg. 17. H. 5. Seiten 669–681. 2011.

[LW93] Lafortune, Eric P.; Willems, Yves D. Bi-Directional Path Tracing. Proceedings of Third International Conference on Computational Graphics And Visualization Techniques (Compugraphics ´93). ACM. New York, NY, USA. Seiten 145–153. 1993.

[LW94] Lafortune, Eric P.; Willems, Yves D. Using the Modified Phong Reflectance Model for Physically Based Rendering. Report CW 197. KU Leuven. Department of Computer Science. Leuven, Belgium. 1994.

[LSK*07] Laine, Samuli; Saransaari, Hannu; Kontkanen, Janne; Lehtinen, Jaakko; Aila, Timo. Incremental Instant Radiosity for Real-time Indirect Illumination. Proceedings of the 18th Eurographics Conference on Rendering Techniques. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 277–286. 2007.

[Len12] Lensing, Philipp. Echtzeit Beleuchtung für Mixed Reality Anwendungen. Master Thesis. FernUniversität Hagen. Fakultät für Mathematik und Informatik. Hagen, Germany. 2012.

[LB11] Lensing, Philipp; Broll, Wolfgang. Fusing the real and the virtual: A depth-camera based approach to Mixed Reality. 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2011. IEEE Computer Society. Washington, DC, USA. Seiten 261–262. 2011.

[LB12] Lensing, Philipp; Broll, Wolfgang. Instant Indirect Illumination for Dynamic Mixed Reality Scenes. Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE Computer Society. Washington, DC, USA. Seiten 109–118. 2012.

[LB13a] Lensing, Philipp; Broll, Wolfgang. Efficient Shading of Indirect Illumination Applying Reflective Shadow Maps. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 95–102. 2013.

[LB13b] Lensing, Philipp; Broll, Wolfgang. LightSkin: Real-Time Global Illumination for Virtual and Mixed Reality. Proceedings of Joint Virtual Reality Conference of EGVE - EuroVR 2013. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 17–24. 2013.

[Lew94] Lewis, Robert R. Making Shaders More Physically Plausible. In Fourth Eurographics Workshop on Rendering. Seiten 47–62. 1994.

[LGL12] Liu, Junyi; Gong, Xiaojin; Liu, Jilin. Guided inpainting and filtering for Kinect depth maps. 21st International Conference on Pattern Recognition (ICPR) 2012. IEEE Computer Society. Washington, DC, USA. Seiten 2055–2058. 2012.

[LSS*04] Liu, Xinguo; Sloan, Peter-Pike; Shum, Heung-Yeung; Snyder, John. All-frequency Precomputed Radiance Transfer for Glossy Objects. Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 337–344. 2004.

[LAM*11] Loos, Bradford J.; Antani, Lakulish; Mitchell, Kenny; Nowrouzezahrai, Derek; Jarosz, Wojciech; Sloan, Peter-Pike. Modular Radiance Transfer. Proceedings of the 2011 SIGGRAPH Asia Conference. ACM. New York, NY, USA. Seiten 178:1-178:10. 2011.

[LNJ*12] Loos, Bradford J.; Nowrouzezahrai, Derek; Jarosz, Wojciech; Sloan, Peter-Pike. Delta Radiance Transfer. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 191–196. 2012.

[LC87] Lorensen, William E.; Cline, Harvey E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. Seiten 163–169. 1987.

[Mac67] MacQueen, J. B. Some Methods for Classification and Analysis of MultiVariate Observations. Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. Seiten 281–297. 1967.

[MW11] Maletz, David; Wang, Rui. Importance Point Projection for GPU-based Final Gathering. Proceedings of the Twenty-second Eurographics Conference on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 1327–1336. 2011.

[MLM13] Mara, Michael; Luebke, David; McGuire, Morgan. Toward Practical Real-time Photon Mapping: Efficient GPU Density Estimation. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 71–78. 2013.

[Mar72] Marsaglia, George. Choosing a Point from the Surface of a Sphere. The Annals of Mathematical Statistics. Jg. 43. H. 2. Seiten 645–646. 1972.

[MRB09] Marton, Z. C.; Rusu, R. B.; Beetz, M. On fast surface reconstruction methods for large and noisy point clouds. IEEE International Conference on Robotics and Automation, 2009 (ICRA ’09). IEEE Computer Society. Washington, DC, USA. Seiten 3218–3223. 2009.

[ML09] McGuire, Morgan; Luebke, David. Hardware-accelerated Global Illumination by Image Space Photon Mapping. Proceedings of the Conference on High Performance Graphics 2009. ACM. New York, NY, USA. Seiten 77–89. 2009.

[MS09] Méndez-Feliu, Àlex; Sbert, Mateu. From Obscurances to Ambient Occlusion: A Survey. Vis. Comput. Jg. 25. H. 2. Seiten 181–196. 2009.

[MES*09] Meyer, Quirin; Eisenacher, Christian; Stamminger, Marc; Dachsbacher, Carsten. Data-parallel Hierarchical Link Creation for Radiosity. Proceedings of the 9th Eurographics Conference on Parallel Graphics and Visualization. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 65–70. 2009.

[MTU*94] Milgram, Paul; Takemura, Haruo; Utsumi, Akira; Kishino, Fumio. Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum. SPIE Vol. 2351, Telemanipulator and Telepresence Technologies. Society of Photo-Optical Instrumentation Engineers (SPIE). Bellingham, WA, USA. Seiten 282–292. 1994.

[Mit07] Mittring, Martin. Finding Next Gen: CryEngine 2. ACM SIGGRAPH 2007 Courses. ACM. New York, NY, USA. Seiten 97–121. 2007.

[ND10] Newcombe, Richard A.; Davison, Andrew J. Live dense reconstruction with a single moving camera. The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010). IEEE Computer Society. Washington, DC, USA. Seiten 1498–1505. 2010.

[NIH*11] Newcombe, Richard A.; Izadi, Shahram; Hilliges, Otmar; Molyneaux, David; Kim, David; Davison, Andrew J.; Kohi, Pushmeet; Shotton, Jamie; Hodges, Steve; Fitzgibbon, Andrew. KinectFusion: Real-time dense surface mapping and tracking. 10th IEEE International Symposium on Mixed and Augmented Reality 2011 (ISMAR). IEEE Computer Society. Washington, DC, USA. Seiten 127–136. 2011.

[NRH03] Ng, Ren; Ramamoorthi, Ravi; Hanrahan, Pat. All-frequency Shadows Using Non-linear Wavelet Lighting Approximation. ACM SIGGRAPH 2003 Papers. ACM. New York, NY, USA. Seiten 376–381. 2003.

[NRH04] Ng, Ren; Ramamoorthi, Ravi; Hanrahan, Pat. Triple Product Wavelet Integrals for All-frequency Relighting. ACM SIGGRAPH 2004 Papers. ACM. New York, NY, USA. Seiten 477–487. 2004.

[NSW09] Nichols, Greg; Shopf, Jeremy; Wyman, Chris. Hierarchical Image-space Radiosity for Interactive Global Illumination. Proceedings of the Twentieth Eurographics Conference on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 1141–1149. 2009.

[NW09] Nichols, Greg; Wyman, Chris. Multiresolution Splatting for Indirect Illumination. Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 83–90. 2009.

[NW10] Nichols, Greg; Wyman, Chris. Interactive Indirect Illumination Using Adaptive Multiresolution Splatting. IEEE Transactions on Visualization and Computer Graphics. Jg. 16. H. 5. Seiten 729–741. 2010.

[NSS10] Nießner, Matthias; Schäfer, Henry; Stamminger, Marc. Fast indirect illumination using layered depth images. The Visual Computer. Jg. 26. H. 6-8. Seiten 679–686. 2010.

[NED11] Novák, Jan; Engelhardt, Thomas; Dachsbacher, Carsten. Screen-space Bias Compensation for Interactive High-quality Global Illumination with Virtual Point Lights. Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 119–124. 2011.

[NHD10] Novák, Jan; Havran, Vlastimil; Dachbacher, Carsten. Path Regeneration for Interactive Path Tracing. The European Association for Computer Graphics 28th Annual Conference: EUROGRAPHICS 2007, short papers. Seiten 61–64. 2010.

[Nus28] Nusselt, Wilhelm. Graphische Bestimmung des Winkelverhältnisses bei der Wärmestrahlung. VDI Z. Jg. 1928. H. 72. Seiten 673 ff. 1928.

[Owe98] Owen, A. B. Monte Carlo extension of quasi-Monte Carlo. Proceedings of the 30th Conference on Simulation (Winter). IEEE Computer Society. Washington, DC, USA. Seiten 571–577. 1998.

[PBD*10] Parker, Steven G.; Bigler, James; Dietrich, Andreas; Friedrich, Heiko; Hoberock, Jared; Luebke, David; McAllister, David; McGuire, Morgan; Morley, Keith; Robison, Austin; Stich, Martin. OptiX: A General Purpose Ray Tracing Engine. ACM SIGGRAPH 2010 Papers. ACM. New York, NY, USA. Seiten 66:1-66:13. 2010.

[PH07] Perreault, S.; Hebert, P. Median Filtering in Constant Time. IEEE Transactions on Image Processing. Jg. 16. H. 9. Seiten 2389–2394. 2007.

[PH10] Pharr, Matt; Humphreys, Greg. Physically Based Rendering. 2. Auflage. Morgan Kaufmann Publishers. Burlington, MA, USA. ISBN: 9780123750792. 2010.

[Pho75] Phong, Bui Tuong. Illumination for Computer Generated Pictures. Commun. ACM. Jg. 18. H. 6. Seiten 311–317. 1975.

[PGS*07] Popov, Stefan; Günther, Johannes; Seidel, Hans-Peter; Slusallek, Philipp; Cohen-Or, Daniel; Slavik, Pavel. Stackless KD-Tree Traversal for High Performance GPU Ray Tracing. Computer Graphics Forum. Jg. 26. H. 3. Seiten 415–424. 2007.

[PKD12] Prutkin, Roman; Kaplanyan, Anton; Dachsbacher, Carsten. Reflective Shadow Map Clustering for Real-Time Global Illumination. Eurographics 2012 - Short Papers Proceedings. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 9–12. 2012.

[PBM*02] Purcell, Timothy J.; Buck, Ian; Mark, William R.; Hanrahan, Pat. Ray Tracing on Programmable Graphics Hardware. ACM Trans. Graph. Jg. 21. H. 3. Seiten 703–712. 2002.

[PDC*03] Purcell, Timothy J.; Donner, Craig; Cammarano, Mike; Jensen, Henrik Wann; Hanrahan, Pat. Photon Mapping on Programmable Graphics Hardware. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 41–50. 2003.

[RSC87] Reeves, William T.; Salesin, David H.; Cook, Robert L. Rendering Antialiased Shadows with Depth Maps. SIGGRAPH Comput. Graph. Jg. 21. H. 4. Seiten 283–291. 1987.

[RWG*13] Ren, Peiran; Wang, Jiaping; Gong, Minmin; Lin, Stephen; Tong, Xin; Guo, Baining. Global Illumination with Radiance Regression Functions. ACM Trans. Graph. Jg. 32. H. 4. Seiten 130:1-130:12. 2013.

[REG*09] Ritschel, T.; Engelhardt, T.; Grosch, T.; Seidel, H.-P.; Kautz, J.; Dachsbacher, C. Micro-rendering for Scalable, Parallel Final Gathering. ACM Trans. Graph. Jg. 28. H. 5. Seiten 132:1-132:8. 2009.

[RGK*08] Ritschel, T.; Grosch, T.; Kim, M. H.; Seidel, H.-P.; Dachsbacher, C.; Kautz, J. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. ACM Trans. Graph. Jg. 27. H. 5. Seiten 129:1-129:8. 2008.

[RDG*12] Ritschel, Tobias; Dachsbacher, Carsten; Grosch, Thorsten; Kautz, Jan. The State of the Art in Interactive Global Illumination. Comput. Graph. Forum. Jg. 31. H. 1. Seiten 160–188. 2012.

[REH*11] Ritschel, Tobias; Eisemann, Elmar; Ha, Inwoo; Kim, James D.K.; Seidel, Hans-Peter. Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes. Computer Graphics Forum (presented at EGSR 2011). Jg. 30. H. 8. Seiten 2258–2269. 2011.

[RGS09] Ritschel, Tobias; Grosch, Thorsten; Seidel, Hans-Peter. Approximating Dynamic Global Illumination in Image Space. Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 75–82. 2009.

[Ryc09] Rycroft, Chris H. VORO++: A three-dimensional Voronoi cell library in C++. Chaos: An Interdisciplinary Journal of Nonlinear Science. Jg. 19. H. 4. 2009.

[ST90] Saito, Takafumi; Takahashi, Tokiichiro. Comprehensible Rendering of 3-D Shapes. Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques. ACM. New York, NY, USA. Seiten 197–206. 1990.

[SNR*12] Scherzer, Daniel; Nguyen, Chuong H.; Ritschel, Tobias; Seidel, Hans-Peter. Pre-convolved Radiance Caching. Comp. Graph. Forum. Jg. 31. H. 4. Seiten 1391–1397. 2012.

[SH07] Scheuermann, Thorsten; Hensley, Justin. Efficient Histogram Generation Using Scattering on GPUs. Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 33–37. 2007.

[SFE*07] Schjøth, Lars; Frisvad, Jeppe Revall; Erleben, Kenny; Sporring, Jon. Photon Differentials. Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia. ACM. New York, NY, USA. Seiten 179–186. 2007.

[Sch94] Schlick, Christophe. An Inexpensive BRDF Model for Physically-based Rendering. Computer Graphics Forum. Jg. 13. H. 2. Seiten 121–131. 1994.

[SIM*06] Segovia, B.; Iehl, J. C.; Mitanchey, R.; Péroche, B. Bidirectional Instant Radiosity. Proceedings of the 17th Eurographics Conference on Rendering Techniques. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 389–397. 2006.

[SB96] Shao, Min-Zhi; Badler, Norman. Spherical Sampling by Archimedes Theorem. Technical Report. University of Pennsylvania. Computer & Information Science. Philadelphia, Pennsylvania, USA. 1996.

[Slo06] Sloan, Peter-Pike. Normal Mapping for Precomputed Radiance Transfer. Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 23–26. 2006.

[SKS02] Sloan, Peter-Pike; Kautz, Jan; Snyder, John. Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. ACM Trans. Graph. Jg. 21. H. 3. Seiten 527–536. 2002.

[SHR10] Soler, Cyril; Hoel, Olivier; Rochet, Frank. A Deferred Shading Pipeline for Real-time Indirect Illumination. ACM SIGGRAPH 2010 Talks. ACM. New York, NY, USA. Seiten 18:1-18:1. 2010.

[Sta95] Stam, Jos. Multiple Scattering as a Diffusion Process. Eurographics Workshop on Rendering Techniques 95. Springer. London, UK, UK. Seiten 41–50. 1995.

[SKE06] Strengert, M.; Kraus, M.; Ertl, T. Pyramid Methods in GPU-Based Image Processing. Workshop on Vision, Modelling, and Visualization VMV ’06. Seiten 169–176. 2006.

[Suf07] Suffern, Kevin. Ray Tracing from the Ground Up. A. K. Peters, Ltd. Natick, MA, USA. ISBN: 1568812728. 2007.

[SR09] Sun, Bo; Ramamoorthi, Ravi. Affine Double- and Triple-product Wavelet Integrals for Rendering. ACM Trans. Graph. Jg. 28. H. 2. Seiten 14:1-14:17. 2009.

[SZC*07] Sun, Xin; Zhou, Kun; Chen, Yanyun; Lin, Stephen; Shi, Jiaoying; Guo, Baining. Interactive Relighting with Dynamic BRDFs. ACM SIGGRAPH 2007 Papers. ACM. New York, NY, USA. 2007.

[SH74] Sutherland, Ivan E.; Hodgman, Gary W. Reentrant Polygon Clipping. Commun. ACM. Jg. 17. H. 1. Seiten 32–42. 1974.

[TL04] Tabellion, Eric; Lamorlette, Arnauld. An Approximate Global Illumination System for Computer Generated Films. ACM Trans. Graph. Jg. 23. H. 3. Seiten 469–476. 2004.

[THG*11] Thiedemann, Sinje; Henrich, Niklas; Grosch, Thorsten; Müller, Stefan. Voxel-based Global Illumination. Symposium on Interactive 3D Graphics and Games. ACM. New York, NY, USA. Seiten 103–110. 2011.

[TSS*98] Thompson, William B.; Shirley, Peter; Smits, Brian; Kersten, Daniel J.; Madison, Cindee. Visual Glue. Technical Report. University of Utah. Salt Lake City, Utah, USA. 1998.

[TS06] Tsai, Yu-Ting; Shih, Zen-Chung. All-frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation. ACM Trans. Graph. Jg. 25. H. 3. Seiten 967–976. 2006.

[van11] van Antwerpen, Dietger. Improving SIMD Efficiency for Parallel Monte Carlo Light Transport on the GPU. Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics. ACM. New York, NY, USA. Seiten 41–50. 2011.

[WDS05] Wald, Ingo; Dietrich, Andreas; Slusallek, Philipp. An Interactive Out-of-core Rendering Framework for Visualizing Massively Complex Models. ACM SIGGRAPH 2005 Courses. ACM. New York, NY, USA. 2005.

[WKB*02] Wald, Ingo; Kollig, Thomas; Benthin, Carsten; Keller, Alexander; Slusallek, Philipp. Interactive Global Illumination Using Fast Ray Tracing. Proceedings of the 13th Eurographics Workshop on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 15–24. 2002.

[WFA*05] Walter, Bruce; Fernandez, Sebastian; Arbree, Adam; Bala, Kavita; Donikian, Michael; Greenberg, Donald P. Lightcuts: A Scalable Approach to Illumination. ACM Trans. Graph. Jg. 24. H. 3. Seiten 1098–1107. 2005.

[WTL06] Wang, Rui; Tran, John; Luebke, David. All-frequency Relighting of Glossy Objects. ACM Trans. Graph. Jg. 25. H. 2. Seiten 293–318. 2006.

[WZP*09] Wang, Rui; Zhou, Kun; Pan, Minghao; Bao, Hujun. An Efficient GPU-based Approach for Interactive Global Illumination. ACM Trans. Graph. Jg. 28. H. 3. Seiten 91:1-91:8. 2009.

[WH92] Ward, Greg; Heckbert, Paul. Irradiance Gradients. Proceedings of Eurographics Workshop on Rendering. Seiten 85–98. 1992.

[War92] Ward, Gregory J. Measuring and Modeling Anisotropic Reflection. SIGGRAPH Comput. Graph. Jg. 26. H. 2. Seiten 265–272. 1992.

[WRC88] Ward, Gregory J.; Rubinstein, Francis M.; Clear, Robert D. A Ray Tracing Solution for Diffuse Interreflection. SIGGRAPH Comput. Graph. Jg. 22. H. 4. Seiten 85–92. 1988.

[Whi80] Whitted, Turner. An Improved Illumination Model for Shaded Display. Commun. ACM. Jg. 23. H. 6. Seiten 343–349. 1980.

[Wil78] Williams, Lance. Casting Curved Shadows on Curved Surfaces. SIGGRAPH Comput. Graph. Jg. 12. H. 3. Seiten 270–274. 1978.

[WPS*10] Wong, H.; Papadopoulou, M.-M.; Sadooghi-Alvandi, M.; Moshovos, A. Demystifying GPU microarchitecture through microbenchmarking. IEEE International Symposium on Performance Analysis of Systems Software 2010 (ISPASS). IEEE Computer Society. Washington, DC, USA. Seiten 235–246. 2010.

[XJF*08] Xu, Kun; Jia, Yun-Tao; Fu, Hongbo; Hu, Shimin; Tai, Chiew-Lan. Spherical Piecewise Constant Basis Functions for All-Frequency Precomputed Radiance Transfer. IEEE Transactions on Visualization and Computer Graphics. Jg. 14. H. 2. Seiten 454–467. 2008.

[XS06] Xu, Yi Ru Huiying; Sun, Yinlong. From physics to illumination models of subsurface scattering. Technical Report. Purdue University. Dapartment of Computer Science. West Lafayette, Indiana, USA. 2006.

[YTA09] Yang, Qingxiong; Tan, Kar-Han; Ahuja, N. Real-time O(1) bilateral filtering. IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Computer Society. Washington, DC, USA. Seiten 557–564. 2009.

[Yan11] Yanovitskij, E. G. Light Scattering in Inhomogeneous Atmospheres. Springer Berlin Heidelberg. ISBN: 9783642644177. 2011.

[YWC*10] Yao, Chunhui; Wang, Bin; Chan, Bin; Yong, Junhai; Paul, Jean-Claude. Multi-image Based Photon Tracing for Interactive Global Illumination of Dynamic Scenes. Proceedings of the 21st Eurographics Conference on Rendering. Eurographics Association. Aire-la-Ville, Switzerland, Switzerland. Seiten 1315–1324. 2010.

[YCK*09] Yu, Insu; Cox, Andrew; Kim, Min H.; Ritschel, Tobias; Grosch, Thorsten; Dachsbacher, Carsten; Kautz, Jan. Perceptual Influence of Approximate Visibility in Indirect Illumination. ACM Trans. Appl. Percept. Jg. 6. H. 4. Seiten 24:1-24:14. 2009.

[ZSM*07] Zalevsky, Z.; Shpunt, A.; Maizels, A.; Garcia, J. Method and system for object reconstruction. Prime Sense Ltd. H. WO2007043036A1. 2007.

[ZHW*08] Zhou, Kun; Hou, Qiming; Wang, Rui; Guo, Baining. Real-time KD-tree Construction on Graphics Hardware. ACM SIGGRAPH Asia 2008 Papers. ACM. New York, NY, USA. Seiten 126:1-126:11. 2008.

[ZTT*06] Ziegler, Gernot; Tevs, Art; Theobalt, Christian; Seidel, Hans-Peter. GPU point list generation through histogram pyramids. Research Report MPI-I-2006-4-002. Max-Planck-Institut für Informatik. 66123 Saarbrücken, Germany. 2006.