

Friedrich-Schiller-Universität Jena
Physikalisch-Astronomische Fakultät
Theoretisch-Physikalisches Institut

Generierung von Irradiance Maps

Bachelorarbeit zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.)
betreut durch Dr. Carsten Lojewski und Prof. Reinhard Meinel

vorgelegt von Markus Pawellek
Matrikelnummer: 144645
geboren am 7.Mai 1995 in Meiningen

Erstgutachter/-in:
Zweitgutachter/-in:

Jena, 16.Juni 2017

Zusammenfassung

Der Gegenstand dieser Arbeit ist es, eine Datenstruktur zu entwickeln, die die Lichtverteilung eines Modells speichert, um unnötige Berechnungen, die bei der Simulation globaler Beleuchtungseffekte im Regelfall ausgeführt werden müssen, zu eliminieren. Dies erlaubt, das Modell von beliebigen Punkten im Raum aus zu beobachten, ohne dessen gesamte Beleuchtung ständig neu berechnen zu müssen. Dafür nehme ich eine genauere Betrachtung der Irradianzbestimmung vor und konstruiere einen Algorithmus, der die erhaltenen Werte auf der Oberfläche des Modells speichert. Das Verfahren ermöglicht damit unter Verwendung zusätzlichen Speichers und einer gewissen Vorberechnungszeit die simultane Darstellung der Beleuchtungsverteilung ohne störende Rauscheffekte, wie sie für andere Verfahren typisch sind.

Inhaltsverzeichnis

1 Einleitung	1
2 Grundlagen und Methoden	3
2.1 Szenengeometrie	3
2.2 Streuung von Licht an Oberflächen	5
2.3 Beleuchtung und Szene	6
2.4 Raytracing	8
2.5 Radiometrie	10
2.6 Rendergleichung	11
3 Bestimmung der Irradianz	13
3.1 Konstruktion komplexer Materialien	13
3.2 Äquivalenz Lambertscher Strahler und der Irradianz	14
3.3 Schätzung der Irradianz	16
4 Aufbau und Generierung der Irradiance Map	29
5 Fazit und Aussicht	35
Literatur	37
A Verwendete Szenen	i
B Verwendete HDR Maps	iii

Abbildungsverzeichnis

1	»Shaderball«-Szene mit einfachen Irradiance Maps	1
2	Facetten-Muster der »Dragon«-Szene ohne Shading-Normale	4
3	Beispiel des Verlaufs einer Vertex-Shading-Normalen	4
4	Struktur des Beispiels einer BSDF	6
5	Wirkungsweise einer Umgebungsbeleuchtung	7
6	Skizze des Sichtbarkeitsproblems und des Raytracing-Verfahrens	8
7	Raytracing als Render-Verfahren	9
8	Skizzenhaftes Beispiel der Strahldichte	10
9	Relation zwischen der ein- und ausfallenden Strahldichte	12
10	»Audi R8«-Szene als Beispiel für die Lösung der Rendergleichung durch Path Tracing	12
11	Unterschied direkter und indirekter Beleuchtung anhand der »Shaderball«-Szene	15
12	Wirkung zusammengesetzter Materialien anhand der »Shaderball«-Szene	16
13	Rauschen durch Path Tracing anhand der »Fairy«-Szene	17
14	Vertex-Irradiance-Map anhand der »Dragon«-Szene	18
15	Vertex-Irradiance-Map anhand der »Shaderball«-Szene	19
16	Fehler der Vertex-Irradiance-Map anhand der »Dragon«-Szene	20
17	Fehler der Vertex-Irradiance-Map anhand der »Shaderball«-Szene	20
18	Erste adaptive Vertex-Irradiance-Map anhand der »Dragon«-Szene	20
19	Zweite adaptive Vertex-Irradiance-Map anhand der »Dragon«-Szene	21
20	Erste adaptive Vertex-Irradiance-Map anhand der »Shaderball«-Szene	21
21	Zweite adaptive Vertex-Irradiance-Map anhand der »Shaderball«-Szene	21
22	Dritte adaptive Vertex-Irradiance-Map anhand der »Shaderball«-Szene	22
23	Vierte adaptive Vertex-Irradiance-Map anhand der »Shaderball«-Szene	23
24	Adaptive Vertex-Irradiance-Map anhand der »Audi R8«-Szene	24
25	Adaptive Vertex-Irradiance-Map anhand der »Audi R8«-Szene	25
26	Adaptive Vertex-Irradiance-Map anhand der »Audi R8«-Szene	26
27	Die Abbildung zeigt das Schema der Dreieck Irradianc Map Datenstruktur auf einem Dreieck (A, B, C) für verschiedene Ordnungen.	29
28	Die Abbildung zeigt das Schema der Dreieck Irradianc Map Datenstruktur für verschiedene Ordnungen.	29
29	Irradiance-Map anhand der »Cornell Box«-Szene	30
30	Irradiance-Map anhand der »Cornell Box«-Szene	31
31	Irradiance-Map anhand der »Cornell Box«-Szene	32

Symboltabelle

Symbol	Definition
\mathbb{N}	Menge der natürlichen Zahlen $\{1, 2, 3, \dots\}$
\mathbb{R}	Menge der reellen Zahlen
$\overline{\mathbb{R}}$	$\mathbb{R} \cup \{-\infty, \infty\}$
(a, b)	$\{x \in \mathbb{R} \mid a < x < b\}$ mit $a, b \in \overline{\mathbb{R}}$
$[a, b]$	$\{x \in \mathbb{R} \mid a \leq x \leq b\}$ mit $a, b \in \overline{\mathbb{R}}$
$[a, b)$	$\{x \in \mathbb{R} \mid a \leq x < b\}$ mit $a, b \in \overline{\mathbb{R}}$
$(a, b]$	$\{x \in \mathbb{R} \mid a < x \leq b\}$ mit $a, b \in \overline{\mathbb{R}}$
$\langle x, y \rangle$	$\sum_{i=1}^n x_i y_i$ mit $x, y \in \mathbb{R}^n$ für $n \in \mathbb{N}$
$\ x\ $	$\sqrt{\langle x, x \rangle}$ mit $x \in \mathbb{R}^n$ für $n \in \mathbb{N}$
\mathcal{S}^2	$\{x \in \mathbb{R}^3 \mid \ x\ = 1\}$
\mathcal{H}_μ^2	$\{x \in \mathcal{S}^2 \mid \langle \mu, x \rangle \geq 0\}$ mit $\mu \in \mathcal{S}^2$
$\text{im } f$	$\{f(x) \mid x \in X\} \subset Y$ mit der Abbildung $f: X \rightarrow Y$ und Mengen X, Y

1 Einleitung

In der 3D-Computergrafik ist für die Erzeugung von realistischen Bildern die Simulation globaler Beleuchtungseffekte notwendig [32]. Diese Lichteffekte ergeben sich formal als Lösung der »Rendergleichung« [13]. Seit der Einführung dieser Gleichung im Jahre 1986 wurden verschiedene Algorithmen entwickelt, welche diese für beliebige Szenen numerisch lösen. Herauskristallisiert hat sich vor allem das »Path Tracing« [13, 32]. Dieses Verfahren kann die reale Beleuchtung beliebig genau und erwartungstreu schätzen. Aus diesem Grund werden die durch Path Tracing generierten Bilder meist als Referenzbild für die Bilder anderer Algorithmen verwendet, um deren Qualität zu untersuchen.

Um nun verschiedene Materialien von Objekten simulieren zu können, werden häufig verschiedene Arten von Lichtstreuung an Oberflächen betrachtet. Eine der wichtigsten Arten ist gerade die ideale diffuse Streuung, welche der Szene ein grundlegendes plastisches Aussehen gibt. Sie ist unabhängig von der Richtung des einfallenden Lichtstrahls und damit auch invariant unter Änderung des Beobachtungspunktes [42]. Für Path Tracing bedeutet dies, dass für jede Änderung der Kamera die eigentlich konstante Lichtverteilung neu berechnet werden muss. Diese Evaluierung nimmt aber auch den größten Rechenaufwand in Anspruch, da für jeden dieser Punkte Licht aus dessen gesamten Hemisphere eingesammelt werden muss. Um also das Verfahren des Path Tracings zu optimieren, müssten die diffusen Lichtverhältnisse vorberechnet und auf der Oberfläche der Szene gespeichert werden [41].

Besonders in der Computerspieleindustrie wird dieses Problem mithilfe von sogenannten »Lightmaps« gelöst, welche für viele Punkte der Szene deren »Irradianz« in einer Textur speichern [18]. Während des Renderings werden diese Irradianzen dann zusammen mit der Farbe der Materialtextur ausgelesen, miteinander multipliziert und dargestellt. Die Generierung einer solchen Lightmap ist jedoch mit diversen Tücken verbunden, welche in vielen Fällen nur durch manuelle Optimierung beseitigt werden können.

Aus diesem Grund führe ich im Laufe dieser Arbeit die sogenannten »Irradiance Maps« ein, die Irradianzen eines Punktes auf der Oberfläche der zugehörigen Szene speichern. Die benötigte Datenstruktur soll dabei speziell für die Verwendung von »Raytracing« und Path Tracing ausgelegt sein. Ich werde eine genauere Betrachtung der Irradianzbestimmung vornehmen, um so deren Prozess zu vereinfachen. Des Weiteren präsentiere ich einen Algorithmus zur adaptiven und automatischen Generierung einer solchen Irradiance Map.

Abbildung 1 (Quelle aller Abbildungen: Markus Pawellek 2017) zeigt ein Beispiel, in welchem deutlich wird, dass Irradiance Maps in der Lage sind die diffusen Lichtverhältnisse sehr gut zu approximieren, jedoch bei schlechter Generierung viel Speicher benötigen (siehe 1b). Außerdem ist in 1c erkennbar, dass für einen Großteil der Szene eine vergleichsweise kleine Auflösung der Irradiance Map ausreicht. Es wird also auch darum gehen, diesen Sachverhalt bei der Konstruktion auszunutzen.



Abbildung 1: Die Bilder zeigen alle denselben Ausschnitt der »Shaderball«-Szene mit unterschiedlichen Shading-Verfahren. Die Lichtquellen bestehen aus einer Sonne und einer Umgebungsbeleuchtung.

2 Grundlagen und Methoden

In den folgenden Kapiteln wird eine grundlegende Einführung und Definition der Verfahren gegeben, die für den weiteren Verlauf dieser Arbeit von Belang sind. Viele dieser Themen können hier nur angezissen werden, da ihre komplette Behandlung den Rahmen und das Ziel des Themas sprengen würden. Für eine genauere Einführung in die einzelnen Themengebiete, wird dem Leser geraten, sich mit den genannten Quellen auseinander zu setzen.

2.1 Szenengeometrie

Um in einem Computer ein zweidimensionales Bild einer dreidimensionalen Umwelt oder auch »Szenen« (engl.: *scene*) zu generieren, benötigen wir ein mathematisches Modell, welches diese hinreichend gut beschreibt. Wir wollen uns einen Beobachter vorstellen, der die Umwelt und die Objekte in ihr von einem bestimmten Punkt im Raum aus betrachtet. Für viele Algorithmen, die diese Aufgabe lösen, ist dabei vor allem das Verhalten von Licht auf den Oberflächen dieser Objekte wichtig [13, 32, 36]. Der Einfachheit halber wollen wir davon ausgehen, dass Licht nur in den oberen Schichten eines Körpers mit dessen Material wechselwirkt. Diese Annahme reduziert die Komplexität des mathematischen Modells, die dann noch aus der Charakterisierung der Oberflächen besteht.

Die Oberflächen realer physikalischer Körper sind im Allgemeinen beliebig geformt und können nicht in geschlossener Form durch eine Gleichung beschrieben werden. Dennoch lassen sie sich im analytischen Sinne durch einfache Hyperflächen (engl.: *shape* [32, S. 123 ff]) im Raum approximieren [15]. Für die Bildgenerierung wählt man für solch eine Fläche meist ein Dreieck. Es ist einfach zu beschreiben und flexibel genug um die meisten Objekte beliebig genau zu approximieren [3, 23]. Dabei wollen wir entartete Dreiecke, die nur aus einem Punkt oder einer Strecke bestehen, ausschließen, da sie für die betrachteten Render-Verfahren nicht darstellbar sind.

DEFINITION 2.1: (Dreieck)

Ein Dreieck Δ wird durch eine Sequenz (A, B, C) von Eckpunkten in \mathbb{R}^3 , für die die Menge $\{B - A, C - A\}$ linear unabhängig ist, charakterisiert. Seien weiterhin

$$M := \{(u, v) \in [0, 1]^2 \mid u + v \leq 1\}$$

$$\varphi: M \rightarrow \mathbb{R}^3, \quad \varphi(u, v) := (1 - u - v)A + uB + vC$$

Dann ist die Menge der Punkte S des Dreiecks gegeben durch im φ und (M, φ) stellt deren Standardparametrisierung dar. Der Notation wegen, identifizieren wir Δ mit S und definieren für alle $(u, v) \in M$

$$\Delta(u, v) := \varphi(u, v)$$

Die baryzentrischen Koordinaten (u, v, w) eines Punktes $x \in \Delta$ sind weiterhin durch die folgenden Eigenschaften gegeben.

$$(u, v) \in M, \quad w = 1 - u - v, \quad \Delta(u, v) = x$$

Die analytische äußere Normale oder auch Normale ist definiert durch

$$\mu := \frac{(B - A) \times (C - A)}{\|(B - A) \times (C - A)\|}$$

Jedes Dreieck besitzt auf seiner gesamten Fläche eine eindeutige konstante äußere analytische Normale. Für die Simulation von globalen Beleuchtungseffekten ist diese Eigenschaft jedoch ein Nachteil,

weil die Beleuchtung eines Objektes stark vom Verlauf seiner Normalen abhängt. Näheren wir ein Objekt durch $n \in \mathbb{N}$ Dreiecke an, so nähern wir auch den Normalenverlauf des Objektes durch die stückweise konstanten Normalen der Dreiecke an. Der Fehler dieser Approximation tritt in Form eines facettenhaften Musters auf, welcher für das menschliche Auge gut erkennbar ist [32, S. 166]. In Abbildung 2 wird dieser Effekt genauer an einem Beispiel demonstriert. Folglich muss darauf geachtet werden, dass bei stetig differenzierbaren Oberflächen, die Stetigkeit der Normalen erhalten bleibt [15, S. 39 ff].

DEFINITION 2.2: (Normalen-Funktion / Shading-Normale)

Sei Δ ein Dreieck mit der Normalen μ . Dann wird $\nu: \Delta \rightarrow \mathcal{H}_\mu^2$ als Normalen-Funktion oder auch Shading-Normale von Δ bezeichnet.

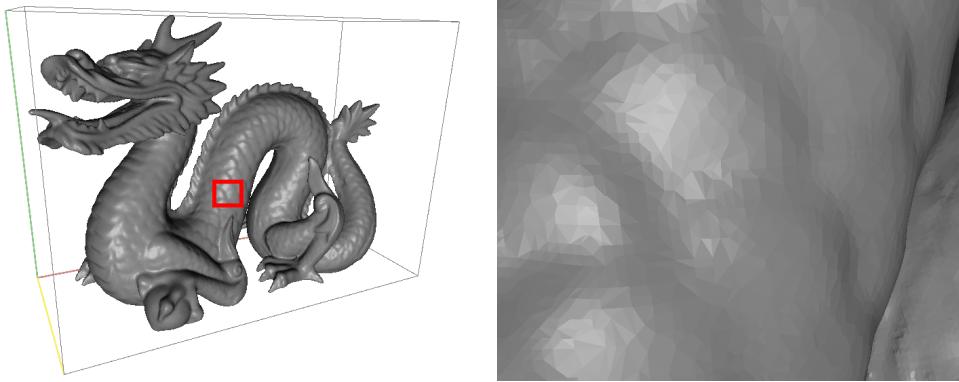


Abbildung 2: Die Bilder zeigen die gerenderte »Dragon«-Szene. Das rechte Bild entspricht dem roten Bereich des Linken. In dieser Szene werden die Normalen des Objektmodells durch die analytischen Normalen der Dreiecke angenähert. Da das Modell sehr fein trianguliert ist, fällt dies im ersten Bild nicht auf. Zoomt man jedoch mit der Kamera heran werden die Fehler durch die Approximation deutlich und die einzelnen Dreiecke sind mit dem menschlichen Auge auszumachen.

Nach [32, S. 166, 584 ff] und [1, S. 38 ff, 183 ff] sind die typischen Verfahren für eine genauere Interpolation die »Vertex-Shading-Normalen« und das »Bump Mapping« (engl.: *bump mapping*). Formal gesehen werden bei beiden Verfahren die Normalen der vorhandenen Geometrie durch eine Normalen-Funktion ersetzt.

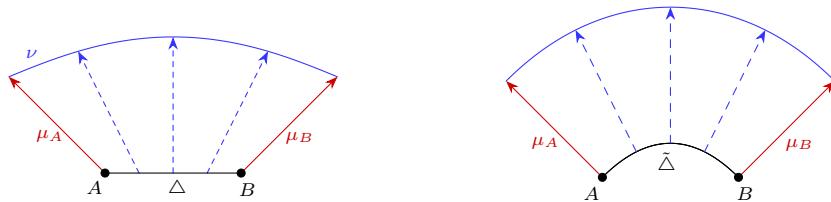


Abbildung 3: Die erste Skizze auf der linken Seite zeigt den Verlauf einer Vertex-Shading-Normalen ν anhand eines Beispiels. A und B sind dabei zwei Eckpunkte eines Dreiecks Δ . μ_A und μ_B sind die jeweilig gegebenen Vertex-Normalen an den Eckpunkten. Im rechten Bereich der Abbildung ist die durch ν approximierte gekrümmte Fläche $\tilde{\Delta}$, auf der die Shading-Normale ν senkrecht steht, eingezeichnet.

DEFINITION 2.3: (Vertex-Shading-Normale)

Seien Δ ein Dreieck mit der Normalen μ und $\mu_A, \mu_B, \mu_C \in \mathcal{H}_\mu^2$ Normalen an den Eckpunkten des Dreiecks. Sei weiterhin ν eine Shading-Normale auf Δ , sodass für alle

$x \in \Delta$ mit den baryzentrischen Koordinaten (u, v, w) gilt

$$\nu(x) := \frac{w\mu_A + u\mu_B + v\mu_C}{\|w\mu_A + u\mu_B + v\mu_C\|}$$

Dann ist ν stetig und man nennt es eine Vertex-Shading-Normale von Δ .

Durch das Setzen der Normalen an den Eckpunkten (engl.: *vertex*) eines Dreiecks können wir sicher gehen, dass der Verlauf der Normalen stetig von einem Dreieck zu einem anderen übergeht. Ein beißiger Verlauf einer Vertex-Shading-Normalen wird in Abbildung 3 gezeigt. Zu beachten ist, dass die Vektoren der Shading-Normale im Allgemeinen nicht mehr senkrecht auf der Geometrie stehen. Dadurch kann es, wie in [32, S. 574 ff] und [36, S. 150 ff] gezeigt, zu Artefakten bei der Generierung des Bildes kommen.

Der letzte Schritt zur vollständigen Beschreibung der Szenengeometrie, besteht darin, eine Menge von Dreiecken zu bilden, sodass diese ein physikalisches Objekt gut genug approximieren. In [32] und [3] werden solche Mengen auch »Meshs« (engl.: *triangle mesh*) genannt.

DEFINITION 2.4: (Mesh)

Seien $n \in \mathbb{N}$ und Dreiecke Δ_i mit Shading-Normalen ν_i für alle $i \in \mathbb{N}, i \leq n$. Weiterhin definieren wir

$$\mathcal{T} := \bigcup_{i=1}^n \Delta_i \quad \mathcal{A} := \{x \in \mathcal{T} \mid \exists! i \in \mathbb{N}, i \leq n: x \in \Delta_i\}$$

$$\nu: \mathcal{T} \rightarrow \mathbb{S}^2 \cup \{0\}, \quad \nu(x) := \sum_{i=1}^n \nu_i(x) \mathbb{I}_{\mathcal{A} \cap \Delta_i}(x)$$

Dann nennen wir \mathcal{T} eine Mesh mit Shading-Normaler ν , wenn für σ -fast-alle $x \in \mathcal{T}$ auch $x \in \mathcal{A}$ gilt.

Die Definition der Mesh verbietet, dass die Schnittpunkte der Dreiecke eine eigene Fläche im Raum bilden. So können wir sicher gehen, dass die Integration über die Punkte der Mesh eine eindeutige Lösung ergibt. Es ist jedoch erlaubt, dass sich Dreiecke in einem Punkt oder einer Strecke schneiden dürfen. Dies ist auch nötig, um komplexere Objekte formen zu können. In der Praxis ist die genannte Bedingung im Allgemeinen erfüllt und folglich auch keine fundamentale Einschränkung [3, 23, 32, 36].

2.2 Streuung von Licht an Oberflächen

Im letzten Abschnitt wurden die Grundbausteine einer Szene eingeführt, die deren Geometrie beschreiben. Wie bereits erwähnt, benötigen wir jedoch zusätzlich die Informationen, auf welche Art und Weise Licht mit den Oberflächen der Objekte interagiert. Dieses Verhalten wird durch die sogenannte »Bidirektionale Streuungsverteilungsfunktion« (engl.: *bidirectional scattering distribution function*, BSDF), die sich aus einer »Bidirektionalen Reflektanzverteilungsfunktion« (engl.: *bidirectional reflectance distribution function*, BRDF) und einer »Bidirektionalen Transmissionsverteilungsfunktion« (engl.: *bidirectional transmittance distribution function*, BTDF) zusammensetzt, geklärt. Strukturierte Einführungen zu diesem Thema erhält man in [1, 5, 26, 32, 36].

DEFINITION 2.5:

Sei $\mu \in \mathbb{S}^2$ die Normale am Punkt einer Oberfläche. Dann ist eine BSDF bezüglich μ gegeben durch eine integrierbare Abbildung $f: \mathbb{S}^2 \times \mathbb{S}^2 \rightarrow [0, \infty)$ mit den folgenden

Eigenschaften.

(i) Für σ^2 -fast-alle (ω_i, ω_o) mit $\omega_i \in S^2$, $\omega_o \in \mathcal{H}_\nu^2$ und $\nu := \text{sgn}(\langle \mu, \omega_i \rangle) \cdot \mu$ gilt

$$f(\omega_i, \omega_o) = f(\omega_o, \omega_i) \quad (\text{Helmholtz-Reziprozität})$$

(ii) Für σ -fast-alle $\omega_i \in S^2$ gilt

$$\int_{S^2} f(\omega_i, \omega) |\langle \mu, \omega \rangle| d\sigma(\omega) \leq 1 \quad (\text{Energieerhaltung})$$

Eine BSDF f gibt an, welcher Anteil des einfallenden Lichtes aus der Richtung $\omega_i \in S^2$ in die Richtung $\omega_o \in S^2$ gestreut wird. Gilt dabei $\omega_o \in \mathcal{H}_\nu^2$, so befindet sich ω_o in der gleichen Hemisphere bezüglich μ wie ω_i und es handelt sich um eine Reflexion des Lichtes. Der verbleibende Fall beschreibt die Transmission. Die BSDF stellt damit eine Verallgemeinerung der idealen Reflexion und Brechung an Oberflächen dar. Die Quellen [32, S. 571 ff] und [36, S. 135 ff] weisen zudem nach, dass die Helmholtz-Reziprozität nur für den reflektierten Anteil des Lichtes gilt. In Abbildung 4 ist das Beispiel einer typischen BSDF gezeigt, die keine Transmission des Lichtes zulässt [32, S. 509 ff].

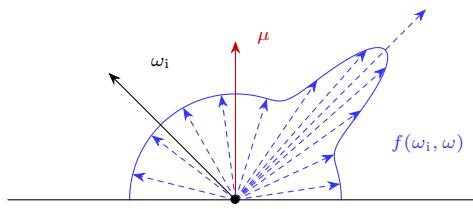


Abbildung 4: Die Abbildung zeigt die Verteilung des reflektierten Lichtes für alle $\omega \in \mathcal{H}_\mu^2$ bezüglich einer konstanten Einfallsrichtung $\omega_i \in \mathcal{H}_\mu^2$ einer BSDF f bezüglich der Oberflächennormalen $\mu \in S^2$.

Die hier eingeführten Funktionen für die Charakterisierung von Materialien können auf verschiedene Weisen verallgemeinert werden. Zu beachten ist vor allem die fehlende Abhängigkeit von der Wellenlänge des einfallenden und des gestreuten Lichtes, welche Effekte wie Dispersion, Irisieren und Lumineszenz verhindert. Ein weiteres physikalisches Phänomen stellt die Volumenstreuung dar. Bei der Betrachtung dieser wird ein Material durch die sogenannte »BSSRDF« beschrieben [32, S. 671 ff].

BSDFs sind im Allgemeinen nicht in geschlossener Form notierbar [32, S. 507 ff]. Aus diesem Grund wollen wir für die Konstruktion realistischer Materialien wie bei der Approximation von Oberflächen verschiedene einfache BSDFs zu Grunde legen. In den beiden folgenden Beispielen handelt es sich um die BSDFs f bezüglich der Normalen $\mu \in S^2$ für die lambertsche diffuse Reflexion (engl.: *lambertian reflection*, [32, S. 531 ff]) und für die ideale Reflexion (engl.: *specular reflection*, [36, S. 144 ff]). Weitere BSDF-Modelle sind in [5, 32, 36] zu finden. Es seien $\omega_i, \omega_o \in S^2$ mit $\nu := \text{sgn}(\langle \mu, \omega_i \rangle) \cdot \mu$ gegeben.

$$f(\omega_i, \omega_o) = \frac{1}{\pi} \mathbb{1}_{\mathcal{H}_\nu^2}(\omega_o) \quad (\text{lambertsche diffuse Reflexion})$$

$$f(\omega_i, \omega_o) = \frac{\delta_{\omega_o}(2 \langle \mu, \omega_i \rangle \mu - \omega_i)}{|\langle \mu, \omega_i \rangle|} \quad (\text{ideale Reflexion})$$

2.3 Beleuchtung und Szene

Die BSDF an einem Punkt beschreibt lediglich die Streuung des einfallenden Lichtes. Um den Render-Verfahren einen Sinn zu geben, benötigen wir demnach Lichtquellen. In [32, S. 707 ff] und [12] werden verschiedene Arten von Lichtquellen definiert, implementiert und gesampled. Der einfachste Weg

Lichtquellen einzuführen, besteht in einer abstrakten Formulierung, die unabhängig von der Szenengeometrie agiert. Wir wollen eine sogenannte »Umgebungsbeleuchtung« (engl.: *hdr environment map* oder *infinite area light*, [32, S. 737 ff]) einführen.

DEFINITION 2.6: (Umgebungsbeleuchtung)

Eine Umgebungsbeleuchtung ist definiert als eine integrierbare Abbildung $f: (0, \infty) \times \mathbb{S}^2 \rightarrow [0, \infty)$.

Diese Funktion beschreibt das Licht verschiedener Wellenlängen, welches von einer Kugeloberfläche mit quasi unendlich großem Radius in die Szene ausgesandt wird. Für einen Punkt der Szene, ist dessen Beleuchtung durch diese Funktion also unabhängig von dessen Position. Diese Tatsache wird in Abbildung 5 wieder an einem Beispiel gezeigt.

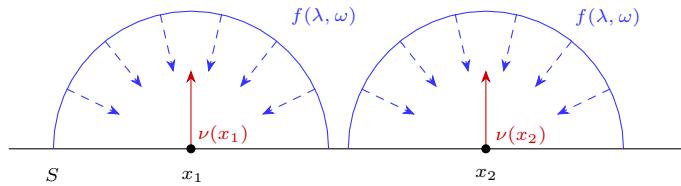


Abbildung 5: Die Darstellung zeigt, dass die Beleuchtung zweier Punkte x_1 und x_2 einer Oberfläche S mit den Normalen $\nu(x_1)$ und $\nu(x_2)$ durch eine Umgebungsbeleuchtung f nur abhängig von der Verdeckung der Punkte und nicht von deren Position ist. Bei x_1 und x_2 sind in diesem Falle die gleichen Lichtintensitäten zu messen. Es ist $\lambda \in (0, \infty)$ die Wellenlänge des Lichtes und $\omega \in \mathbb{S}^2$ der Raumwinkel.

Aber auch die Materialien der Szene sollten in der Lage sein Licht auszusenden. Wir führen hierfür eine Abbildung ein, die die dafür nötigen Eigenschaften erfüllt. Häufig werden diese Lichtquellen auch »Feldlichtquellen« (engl.: *area light*) genannt [32, S. 733 ff].

DEFINITION 2.7: (Emission)

Sei \mathcal{T} eine Mesh mit einer Shading-Normalen ν . Dann ist eine Emission von \mathcal{T} durch eine integrierbare Abbildung $E: \mathcal{T} \times (0, \infty) \times \mathbb{S}^2 \rightarrow [0, \infty)$ gegeben.

Wie bei der vorherigen Definition beschreibt diese Funktion das ausgesandte Licht in Abhängigkeit der Wellenlänge und des Raumwinkels. Der Unterschied besteht darin, dass sie auf der Oberfläche einer Mesh variieren kann und sich damit auch die Beleuchtung eines Punktes je nach Position verändert.

Durch die Einführung der Lichtquellen erhalten wir nun die vollständige Beschreibung einer Szene durch Zusammenführung der bereits definierten Strukturen.

DEFINITION 2.8: (Szene)

Eine Szene ist ein Tupel $(\mathcal{T}, \nu, f, E, U)$ bestehend aus einer Mesh \mathcal{T} mit einer Shading-Normalen ν , einer integrierbaren Abbildung $f: \mathcal{T} \times \mathbb{S}^2 \times \mathbb{S}^2 \rightarrow [0, \infty)$, wobei $f(x, \cdot, \cdot)$ für σ -fast-alle $x \in \mathcal{T}$ ein BSDF bezüglich $\nu(x)$ darstellt, einer Emission E von \mathcal{T} und einer Umgebungsbeleuchtung U .

2.4 Raytracing

Im einfachsten Falle bezeichnet das Wort »Raytracing« (engl.: *ray tracing*) einen Algorithmus zur Ermittlung der Sichtbarkeit von dreidimensionalen Objekten bezüglich eines Ursprungspunktes (engl.: *origin*) im Raum [30, 32]. Häufig versteht man darunter jedoch auch eine Render-Technik für die Generierung eines gesamten Bildes aus einer gegebenen Szene, die auf dem eben genannten Raytracing-Algorithmus basiert [27, 30, 32].

Grundsätzlich gibt es viele Verfahren, um ein Szene auf ein Bild zu rendern [1, 6]. Die Erfahrung zeigt aber, dass vor allem in Bereichen, in denen die globalen Beleuchtungseffekte realistisch simuliert werden sollen, Raytracing eine wichtige Grundlage darstellt. Der Grund dafür besteht in der Tatsache, dass Raytracing das »Sichtbarkeitsproblem« (engl.: *visibility problem*, [6, 8]) löst und durch die Verwendung von Strahlen eine Basis für die Lichtberechnung im Sinne der geometrischen Optik bereitstellt [30, 32, 36].

DEFINITION 2.9: (Sichtbarkeitsproblem)

Sei \mathcal{T} eine Mesh. Dann ist die Sichtbarkeitsfunktion von \mathcal{T} die folgende Abbildung.

$$V_{\mathcal{T}}: \mathbb{R}^3 \times \mathcal{T} \rightarrow \{0, 1\}$$

$$V_{\mathcal{T}}(o, x) = \begin{cases} 1 & : \mathcal{T} \cap \{(1 - \gamma)o + \gamma x \mid \gamma \in (0, 1)\} = \emptyset \\ 0 & : \text{sonst} \end{cases}$$

Das Sichtbarkeitsproblem beschreibt die Aufgabe diese Funktion für gegebene Parameter zu evaluieren.

Die Sichtbarkeitsfunktion gibt an, ob der Oberflächenpunkt x vom Beobachtungspunkt o aus in gerader Linie gesehen werden kann oder ob zwischen diesen Punkten ein weiterer Punkt der Mesh \mathcal{T} den Punkt x verdeckt [8, S. 30]. Daran anknüpfend besteht die Basis des Raytracing-Verfahrens auf dem Aussenden von »Strahlen« (engl.: *ray*) bezüglich eines Ursprungspunktes [27, 30, 32]. Für diese Strahlen kann der Schnittpunkt mit \mathcal{T} ermittelt werden. Liegt der Schnittpunkt zwischen o und x , so beträgt der Wert der Sichtbarkeitsfunktion 0. Im noch bleibenden Fall ergibt sich der Wert zu 1. Die Sichtbarkeitsfunktion kann somit für alle gegebenen Parameter durch Raytracing berechnet werden. Abbildung 6 zeigt diese Methode anhand einer Skizze.

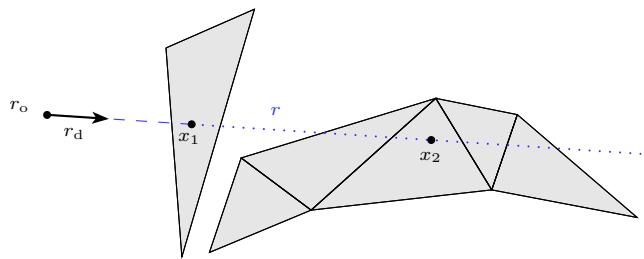


Abbildung 6: Die Abbildung zeigt eine Skizze, welche das Sichtbarkeitsproblem und den Raytracing-Algorithmus verdeutlicht. Die grau melierten Dreiecke sollen die gegebene Mesh \mathcal{T} darstellen. Der ausgesendete Strahl r , gegeben durch (r_o, r_d) , trifft in der Mesh genau zwei Punkte x_1 und x_2 . Dabei wird x_2 durch x_1 verdeckt. Es gilt also $V_{\mathcal{T}}(r_o, x_1) = 1$ und $V_{\mathcal{T}}(r_o, x_2) = 0$.

DEFINITION 2.10: (Strahl)

Seien ein Ursprung $o \in \mathbb{R}^3$, eine Richtung $d \in \mathbb{S}^2$ und die folgende Abbildung gegeben.

$$\varphi: [0, \infty) \rightarrow \mathbb{R}^3, \quad \varphi(\gamma) := o + \gamma d$$

Dann wird das Bild im φ ein Strahl r genannt. Dabei ist $([0, \infty), \varphi)$ die Standardparametrisierung von r . Der Notation wegen, definieren wir für alle $\gamma \in [0, \infty)$

$$r(\gamma) := \varphi(\gamma)$$

Durch das Tupel (o, d) charakterisieren und identifizieren wir den Strahl r . Die Menge aller Strahlen definieren wir als \mathcal{R} .

Die vollständige Evaluierung von V_T ist jedoch häufig nicht notwendig. In Abbildung 6 ist klar, dass alle Punkte von r , die hinter x_1 liegen von dem Beobachtungspunkt r_o aus nicht sichtbar sind. Beim eigentlichen Raytracing-Algorithmus wirkt sich diese Eigenschaft positiv aus. Es reicht für eine gegebene Richtung und einen gegebenen Beobachtungspunkt den nächsten Schnittpunkt des resultierenden Strahles zu ermitteln. Für alle weiteren Schnittpunkte, sofern diese existieren, ergibt sich die Sichtbarkeitsfunktion zu Null, wodurch sie für das weitere Vorgehen ignoriert werden können [32, S. 4 ff, 866].

DEFINITION 2.11: (Raytracing-Funktion)

Sei T eine Mesh. Dann ist die Raytracing-Funktion von T durch die folgende Abbildung gegeben.

$$rt_T: \mathcal{R} \rightarrow (0, \infty]$$

$$rt_T(r) := \begin{cases} \min \{\gamma \in (0, \infty) \mid r(\gamma) \in T\} & : T \cap (r \setminus \{r(0)\}) \neq \emptyset \\ \infty & : \text{sonst} \end{cases}$$

Wie bereits erwähnt, erweitert man diesen Algorithmus häufig mit der Berechnung eines gesamten Bildes, indem man für jeden Pixel des Bildes einen oder mehrere Strahlen durch einen analogen virtuellen Pixel im Szenenraum schießt und die Raytracing-Funktion für diese evaluiert [27, 30, 32]. Abbildung 7 zeigt dieses Verfahren anhand einer Skizze. Um gleichzeitig das sogenannte »Sha-

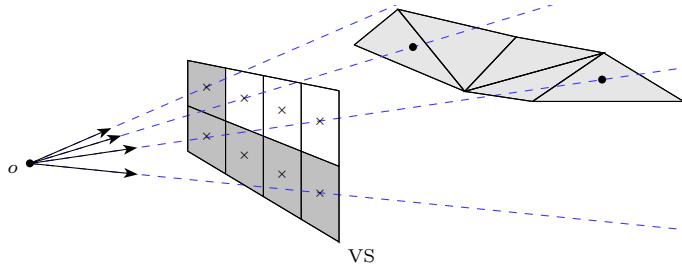


Abbildung 7: In der Skizze ist ein typisches Render-Verfahren auf der Basis des Raytracing-Algorithmus dargestellt. Bezuglich eines Beobachtungspunktes $o \in \mathbb{R}^3$ wird durch jeden Pixel eines virtuellen Bildschirms VS ein Strahl geschossen und der Wert der Raytracing-Funktion evaluiert. Die grau schattierten Dreiecke bilden die Mesh T . Je nachdem, ob ein Strahl einen Schnittpunkt mit T aufweist, wird ein entsprechendes Shading-Verfahren ausgeführt. Der Übersicht wegen sind im Bild nur vier der eigentlich acht Strahlen sichtbar.

ding« zu ermöglichen, ermittelt man nicht nur den Schnittpunkt, sondern auch in welchem Dreieck sich dieser befindet und welche baryzentrischen Koordinaten er besitzt [22, 32]. Die Implementierung des Raytracing-Verfahrens soll hier nicht gezeigt werden, da eine einfache Implementierung einen zu

großen Rechenaufwand darstellt und die hier verwendete optimierte Variante weit über das Thema dieser Arbeit hinausgeht. Für eine strukturierte Einführung von Beschleunigungsstrukturen sei auf [32, S. 247 ff] und [27, 30] verwiesen.

2.5 Radiometrie

Ein physikalischer Körper K mit der Oberfläche ∂K und äußerer Normalen-Funktion μ sendet aufgrund verschiedener physikalischer Effekte, wie zum Beispiel Reflexion oder Emission, elektromagnetische Wellen aus [28]. Für die Simulation von Beleuchtungseffekten ist es nötig die Abhängigkeit dieser Abstrahlung für verschiedene Wellenlängen $\lambda \in (0, \infty)$, verschiedene Oberflächenpunkte $x \in \partial K$ und verschiedene Richtungen $\omega \in \mathbb{S}^2$ zu betrachten. Dafür führen wir die sogenannte »spektrale Strahldichte« (engl.: *spectral radiance*) ein. Sie gibt an, wie viel Energie pro Zeit, pro Wellenlängenänderung, pro Flächenelement und pro Raumwinkel abgestrahlt beziehungsweise empfangen wird [20, 29, 42]. Sind also messbare Teilmengen $\Lambda \subset (0, \infty)$, $U \subset \partial K$ und $S \subset \mathbb{S}^2$ gegeben, so kann man die spektrale Strahldichte über die abgegebene Strahlungsleistung $\Phi(U, \Lambda, S)$ der Oberfläche U in den Raumwinkelbereich S definieren [42]. Die zugehörige Abbildung ist dann wie folgt gegeben.

$$\mathcal{L}: \partial K \times (0, \infty) \times \mathbb{S}^2 \rightarrow [0, \infty)$$

$$\Phi(U, \Lambda, S) =: \int_U \int_{\Lambda} \int_S \mathcal{L}(x, \lambda, \omega) |\langle \mu(x), \omega \rangle| d\sigma(\omega) d\lambda(\lambda) d\sigma(x)$$

Abbildung 8 zeigt eine Skizze, welche die Struktur dieser Abbildung verdeutlicht. Der Erfahrung nach stellt die spektrale Strahldichte für das menschliche Auge die sinnvollste messbare Größe für die empfundene Helligkeit einer Oberfläche bezüglich eines Beobachtungspunktes dar [20, 29].

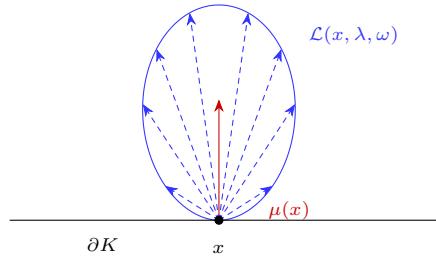


Abbildung 8: Die Skizze stellt das qualitative Beispiel einer spektralen Strahldichte \mathcal{L} dar, die an einem Punkt x auf der Oberfläche ∂K mit der äußeren Normalen μ und für eine Wellenlänge λ eine Verteilung in Abhängigkeit von $\omega \in \mathbb{S}^2$ besitzt.

Eine weitere Größe, die wir hier einführen möchten, ist die sogenannte »Bestrahlungsstärke« oder auch spektrale Irradianz (engl.: *spectral irradiance*). Sie wird vor allem bei der noch folgenden Konstruktion der Irradianz Maps benötigt werden. Wir definieren sie für eine messbare Teilmenge $S \subset \mathbb{S}^2$ [11, S. 9 f].

$$\mathcal{E}: \partial K \times (0, \infty) \rightarrow [0, \infty), \quad \mathcal{E}(x, \lambda) := \int_S \mathcal{L}(x, \lambda, \omega) |\langle \mu(x), \omega \rangle| d\sigma(\omega)$$

Die spektrale Bestrahlungsstärke quantifiziert die Energie pro Zeit, pro Wellenlängenänderung und pro Flächenelement, die an einem Punkt auf der Oberfläche des Objektes aus dem Raumwinkelbereich S empfangen wird [11, 42].

Für die weiteren Algorithmen und Implementierungen reicht es entkoppelte Abbildungen zu betrachten, welche für jeden Punkt einer Szene und jede gegebene Richtung der Strahldichte beziehungsweise der Irradianz entsprechen. Später werden dies die Funktionen sein, die wir durch »Path Tracing« simulieren und durch geeignetes »Surface Caching« zwischenspeichern wollen.

DEFINITION 2.12: (Strahldichte und Irradianz)

Sei $\Sigma := (\mathcal{T}, \nu, f, E, U)$ eine Szene. Dann ist eine Strahldichte von Σ gegeben durch eine integrierbare Abbildung

$$L: \mathcal{T} \times (0, \infty) \times \mathcal{S}^2 \rightarrow [0, \infty)$$

Die zu L gehörige Irradianz definieren wir durch die folgende Funktion.

$$R: \mathcal{T} \times (0, \infty) \rightarrow [0, \infty), \quad R(x, \lambda) := \int_{\mathcal{H}_{\nu(x)}^2} L(x, \lambda, \omega) \langle \mu(x), \omega \rangle \, d\sigma(\omega)$$

2.6 Rendergleichung

Die »Rendergleichung« (engl.: *rendering equation* oder *light transport equation*, [32, S. 861]) ist eine Integralgleichung, die 1986 von James T. Kajiya entwickelt wurde, um die bis zu diesem Zeitpunkt verbreiteten Rendertechniken für die Simulation globaler Beleuchtungseffekte auf eine gemeinsame Basis zu stellen [13]. In den Quellen [13], [32, S. 349 ff, 862 ff] und [36] wird sie mithilfe der Prinzipien der geometrischen Optik und dem Energieerhaltungssatz hergeleitet. Quelle [36] stellt dabei die resultierende Gleichung zusätzlich in einer Operator-Formulierung dar. Insbesondere handelt es sich bei der Rendergleichung um eine Fredholm-Integralgleichung 2. Art, die damit unter gewissen Voraussetzungen eine eindeutige Lösung besitzt (siehe hierzu [36, S. 103 ff] und [33]).

DEFINITION 2.13: (Rendergleichung)

Seien $\Sigma := (\mathcal{T}, \nu, f, E, U)$ eine Szene und L eine Strahldichte von Σ . Dann gehorcht L der Rendergleichung, wenn für alle $x \in \mathcal{T}$, $\lambda \in (0, \infty)$ und $\omega_0 \in \mathcal{S}^2$ das Folgende gilt.

$$L(x, \lambda, \omega_0) = E(x, \lambda, \omega_0) + \int_{\mathcal{S}^2} f(x, \omega, \omega_0) \tilde{L}(x, \lambda, \omega) |\langle \nu(x), \omega \rangle| \, d\sigma(\omega)$$

$$\tilde{L}(x, \lambda, \omega) := \begin{cases} L(r(\text{rt}_{\mathcal{T}}(r)), \lambda, -\omega) & : r \in \mathcal{R}, r \equiv (x, \omega), \text{rt}_{\mathcal{T}}(r) < \infty \\ U(\lambda, \omega) & : \text{sonst} \end{cases}$$

In der Definition entspricht $\tilde{L}(x, \lambda, \omega)$ der aus ω einfallenden Strahldichte am Punkt x der Wellenlänge λ . Die Funktionen \tilde{L} und L sind über die Raytracing-Funktion miteinander verbunden. Existiert jedoch für den gegebenen Strahl r kein Schnittpunkt mit der Szene, so entspricht die einfallende Strahldichte gerade der Umgebungsbeleuchtung. In Abbildung 9 wird dieser Zusammenhang an einem Beispiel demonstriert. Die Strahldichte in Richtung ω entspricht damit der Summe des emittierten und des gestreuten Lichtes am Punkt x .

Gehorcht die Strahldichte einer Szene der Rendergleichung, so simuliert sie die globalen Beleuchtungseffekte dieser Szene auf einer physikalischen Basis, wodurch die entstehenden Lichtverhältnisse für das menschliche Auge real wirken [13, 32, 36]. Die Rendergleichung lässt sich aber im Allgemeinen nicht analytisch lösen [13, 32, 36]. Aus diesem Grund sind verschiedene numerische Lösungsmethoden für die Berechnung einer Strahldichte, die der Rendergleichung gehorcht, entwickelt worden. Die berühmtesten Beispiele stellen das »Path Tracing« [13], das »Bidirectional Path Tracing« [17], der »Metropolis Light Transport« [37] und das »Photon Mapping« [12] dar. Eine strukturierte Einführung zu diesen Verfahren gibt es auch hier in [32]. Das Beispiel für eine durch Path Tracing erzeugte Lösung ist in Abbildung 10 zu sehen.

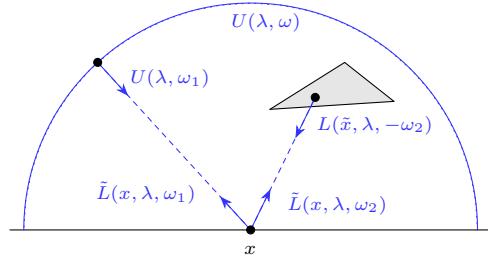


Abbildung 9: Die Abbildung skizziert an einem Beispiel die Relation der Funktionen L und \tilde{L} aus der Definition der Rendergleichung. Dabei stellen das Dreieck einen Teil der Mesh, U die Umgebungsbeleuchtung für $\omega \in S^2$, ω_1, ω_2 Raumrichtungen, λ die betrachtete Wellenlänge und x einen Punkt auf der Mesh dar. Für \tilde{x} gilt nach Definition $\tilde{x} = r(rt_T(r))$, wobei der Strahl r durch (x, ω_2) gegeben ist. Der Strahl in Richtung ω_1 besitzt keinen Schnittpunkt mit der Szene, wodurch sich die einfallende Strahldichte aus der Umgebungsbeleuchtung ergibt.



Abbildung 10: Die Abbildung stellt eine durch Path Tracing simulierte Lösung der Rendergleichung für die »Audi R8«-Szene dar. Die Lichtquelle besteht hier alleine aus der Umgebungsbeleuchtung. Für die BSDFs ist immer eine Mischung aus idealer Reflexion, lambertsch diffuser Reflexion und idealer Brechung gewählt worden.

Alle genannten Verfahren arbeiten auf der Basis von Zufallszahlen. Da die Rendergleichung eine mehrdimensionale Integralgleichung ist, bietet sich die Verwendung verschiedener »Monte-Carlo-Methoden« an, weil diese eine effiziente Schätzung mehrdimensionaler Integrale erlauben [24]. Für uns bedeutet diese Tatsache, dass es sich bei der Abfrage der Strahldichte für diese numerischen Verfahren um die Realisierung einer Zufallsvariable handelt, deren Erwartungswert im besten Falle der der wahren Strahldichte entspricht. Eine detailliertere Betrachtung dieser Eigenschaft wird in den weiteren Kapiteln folgen. Für die Konstruktion der Irradiance Maps ist es im Grunde genommen unwichtig, welcher Algorithmus verwendet wird, sofern er in der Lage ist, die Strahldichte für gegebene Parameter zu evaluieren. Wir verwenden hier einen einfachen Path Tracing Algorithmus.

3 Bestimmung der Irradianz

Nach der Einführung der Rendergleichung und dem Nennen verschiedener Methoden zur Simulation dieser, wollen wir nun die bereits in Kapitel 1 angesprochenen Probleme aufgreifen und Lösungsmethoden entwerfen. In diesem Kapitel wird es vor allem um die Eigenschaften spezieller Materialien, die auch Lambertsche Strahler genannt werden, und die Beschreibung einer Messmethode der Irradianz gehen. Im noch folgenden Kapitel werden die gewonnenen Informationen implizit bei der Konstruktion der Irradiance Map angewendet.

3.1 Konstruktion komplexer Materialien

Wie bereits in Abschnitt 2.2 erwähnt, ist es im Allgemeinen nicht möglich, die BSDF eines Materials in geschlossener Form anzugeben. Es gibt jedoch verschiedene Verfahren um einfache BSDF-Modelle zu konstruieren, die in der Simulation der Strahldichte Anwendung finden. In Quelle [32, S. 507 f] wird hierfür die Verwendung von Messdaten, phenomenologischen Beobachtungen, Simulationsergebnissen und physikalischen Gesetzen als Beispiel genannt. Durch die Kombination solcher Modelle ist man in der Lage auch diverse komplexe Materialien zu simulieren. Das folgende Theorem formuliert dieses Vorgehen genauer.

THEOREM: (BSDF-Reihe)

Seien $\mu \in \mathcal{S}^2$, $(f_n)_{n \in \mathbb{N}}$ eine Folge von BSDFs bezüglich μ und $(\alpha_n)_{n \in \mathbb{N}}$ eine Folge von Werten in $[0, 1]$, sodass die beiden folgenden Eigenschaften für σ^2 -fast-alle $(\omega_i, \omega_o) \in \mathcal{S}^2 \times \mathcal{S}^2$ gelten.

$$\sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) < \infty, \quad \sum_{n \in \mathbb{N}} \alpha_n \leq 1$$

Dann ist auch die Abbildung f mit der folgenden Definition eine BSDF bezüglich μ .

$$f: \mathcal{S}^2 \times \mathcal{S}^2 \rightarrow [0, \infty), \quad f(\omega_i, \omega_o) := \begin{cases} \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) & : \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) < \infty \\ 0 & : \text{sonst} \end{cases}$$

Beweis:

Für die Wohldefiniertheit von f betrachten wir dessen Definitions- und Wertebereich. Der Definitionsbereich von f entspricht dem der f_n für alle $n \in \mathbb{N}$. Weiterhin folgt aus der Definition von f , dass $f < \infty$ gilt. Für $n \in \mathbb{N}$ betrachtet man dann

$$f_n \geq 0 \implies \alpha_n f_n \geq 0 \implies f = \sum_{n \in \mathbb{N}} \alpha_n f_n \geq 0$$

Insbesondere ist also $f(\omega_i, \omega_o) \in [0, \infty)$ für alle $\omega_i, \omega_o \in \mathcal{S}^2$. Damit ist f eine Abbildung der Form $f: \mathcal{S}^2 \times \mathcal{S}^2 \rightarrow [0, \infty)$ und wohldefiniert.

Kommen wir nun zur Integrierbarkeit. Für alle $n \in \mathbb{N}$ sind die Abbildungen f_n und infolgedessen auch $\alpha_n f_n$ integrierbar. Wir definieren die Folge $(g_n)_{n \in \mathbb{N}}$ von Funktionen durch

$$g_n := \sum_{i=1}^n \alpha_i f_i$$

Dann ist g_n aufgrund der Linearität des Integrals integrierbar und es gilt $g_n \leq g_{n+1}$ für alle $n \in \mathbb{N}$. Weiterhin erhält man durch die Anwendung der Definition die folgende Aussage für σ^2 -fast-alle $(\omega_i, \omega_o) \in \mathcal{S}^2 \times \mathcal{S}^2$.

$$g_n(\omega_i, \omega_o) \xrightarrow{n \rightarrow \infty} f(\omega_i, \omega_o)$$

Nach dem Satz über die Monotone Konvergenz [9, S. 125] ist damit f eine integrierbare Funktion, für die das Folgende aufgrund der Linearität des Integrals gilt.

$$\begin{aligned}\int_{\mathbb{S}^2 \times \mathbb{S}^2} f \, d\sigma^2 &= \lim_{n \rightarrow \infty} \int_{\mathbb{S}^2 \times \mathbb{S}^2} g_n \, d\sigma^2 = \lim_{n \rightarrow \infty} \int_{\mathbb{S}^2 \times \mathbb{S}^2} \sum_{i=1}^n \alpha_i f_i \, d\sigma^2 \\ &= \lim_{n \rightarrow \infty} \sum_{i=1}^n \alpha_i \int_{\mathbb{S}^2 \times \mathbb{S}^2} f_i \, d\sigma^2 = \sum_{n \in \mathbb{N}} \alpha_n \int_{\mathbb{S}^2 \times \mathbb{S}^2} f_n \, d\sigma^2\end{aligned}$$

Die Helmholtz-Reziprozität von f wird direkt durch die Verwendung der Helmholtz-Reziprozität der $f_{n,n} \in \mathbb{N}$ klar. Für σ^2 -fast-alle (ω_i, ω_o) mit $\omega_i \in \mathbb{S}^2, \omega_o \in \mathcal{H}_\nu^2$ und $\nu := \text{sgn}(\langle \mu, \omega_i \rangle) \cdot \mu$ gilt demnach

$$f(\omega_i, \omega_o) = \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_i, \omega_o) = \sum_{n \in \mathbb{N}} \alpha_n f_n(\omega_o, \omega_i) = f(\omega_o, \omega_i)$$

Für die Energieerhaltung erhalten wir eine entsprechende Aussage durch die Anwendung des Satzes von Fubini [9, S. 175 f]. Es ist damit $f(\omega_i, \cdot)$ σ -integrierbar für σ -fast-alle $\omega_i \in \mathbb{S}^2$. Die Funktion $|\langle \mu, \cdot \rangle|$ ist stetig und durch 1 beschränkt. Mithin ist auch $f(\omega_i, \cdot) |\langle \mu, \cdot \rangle|$ messbar und es gilt für σ -fast-alle $\omega_i \in \mathbb{S}^2$

$$f(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \leq f(\omega_i, \cdot) \implies \int_{\mathbb{S}^2} f(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \, d\sigma \leq \int_{\mathbb{S}^2} f(\omega_i, \cdot) \, d\sigma < \infty$$

Analog zum Beweis der Integrierbarkeit von f lässt sich dann mithilfe der Energieerhaltung der $f_{n,n} \in \mathbb{N}$ die Energieerhaltung von f formulieren.

$$\int_{\mathbb{S}^2} f(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \, d\sigma = \sum_{n \in \mathbb{N}} \alpha_n \int_{\mathbb{S}^2} f_n(\omega_i, \cdot) |\langle \mu, \cdot \rangle| \, d\sigma \leq \sum_{n \in \mathbb{N}} \alpha_n \leq 1$$

□

Nach dieser Aussage ist man in der Lage, abzählbar viele BSDFs mithilfe entsprechender Koeffizienten miteinander zu kombinieren. Eine direkte Folgerung bildet die Anwendung des Satzes auf nur endlich viele BSDFs.

KOROLLAR: (BSDF-Summe)

Seien $\mu \in \mathbb{S}^2$ und $n \in \mathbb{N}$. Weiterhin seien f_k eine BSDF bezüglich μ und $\alpha_k \in [0, 1]$ für alle $k \in \mathbb{N}$ mit $k \leq n$, sodass $\sum_{k=1}^n \alpha_k \leq 1$ gilt. Dann ist die folgende Abbildung eine BSDF bezüglich μ .

$$f := \sum_{k=1}^n \alpha_k f_k$$

3.2 Äquivalenz Lambertscher Strahler und der Irradianz

Eines der einfachsten BSDF-Modelle ist das der Lambertschen diffusen Reflexion, welches in Abschnitt 2.2 als Beispiel eingeführt wurde. Die einfallende Strahldichte wird bei diesem Modell gleichmäßig in alle Richtungen der Halbkugel gestreut, was auch der idealen diffusen Reflexion entspricht. Eine Oberfläche mit dieser Eigenschaft wird auch Lambertscher Strahler (engl.: *Lambertian radiator*, [42, S. 17]) genannt. Zu beachten ist, dass diese BSDF keinerlei Transmission von Licht beschreibt und auch im Allgemeinen keiner physikalischen Basis entspricht. Dennoch stellt sie eine gute Approximation vieler realer Oberflächen, die ein mattes Aussehen besitzen, dar [32, S. 532]. Das Modell eignet sich aufgrund seiner Simplizität für die numerische Simulation der globalen Beleuchtung. In Abbildung 11 ist erkennbar, dass der Szene durch die Verwendung des Modells ein plastisches und damit auch realistischeres Aussehen zu Teil wird.

Wir wollen nun eine Szene $\Sigma := (\mathcal{T}, \nu, f, E, U)$ und einen Punkt $x \in \mathcal{T}$ betrachten, sodass $f(x, \cdot, \cdot)$ einen Lambertschen Strahler bezüglich $\nu(x)$ darstellt. Die Rendergleichung für die Strahldichte L am

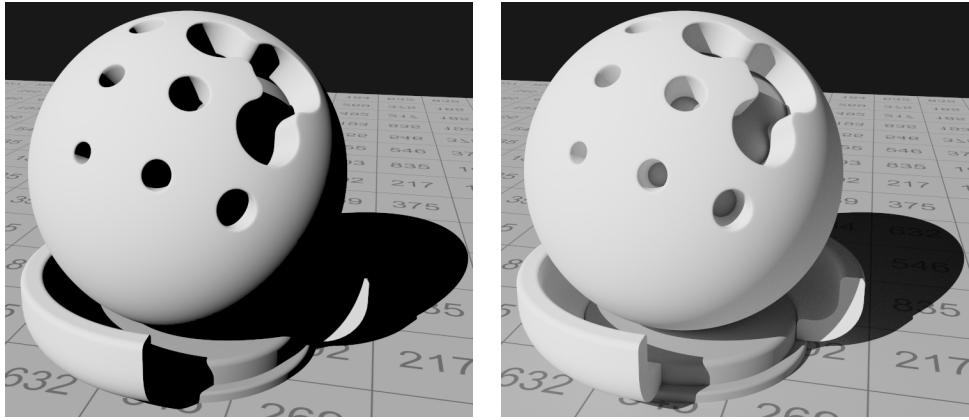


Abbildung 11: Die Bilder zeigen die gerenderte »Shaderball«-Szene, beleuchtet durch eine direktionale Lichtquelle. Im linken Bild wurde die Strahldichte der Szene nur durch direkte Beleuchtung ohne die Verwendung einer entsprechenden BSDF berechnet. Im rechten Bild wurde für jedes Material eine lambertsch diffuse Reflexion angenommen. Die Szene wirkt hier wesentlich plastischer und realistischer als im linken Bild.

Punkt x in Richtung $\omega_o \in \mathcal{H}_{\nu(x)}^2$ für eine Wellenlänge $\lambda \in (0, \infty)$ lautet dann wie folgt.

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \underbrace{\frac{1}{\pi} \int_{S^2} \mathbb{1}_{\mathcal{H}_{\nu(x)}^2}(\omega) \tilde{L}(x, \lambda, \omega) |\langle \nu(x), \omega \rangle| d\sigma(\omega)}_{=: L_r(x, \lambda, \omega_o)}$$

$L_r(x, \lambda, \omega_o)$ beschreibt den reflektierten Anteil der Strahldichte. Die charakteristische Funktion im Integranden ändert die Integrationsmenge. Bezeichnen wir die Irradianz von L durch die Variable R , so folgt

$$L_r(x, \lambda, \omega_o) = \frac{1}{\pi} \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \nu(x), \omega \rangle d\sigma(\omega) = \frac{R(x, \lambda)}{\pi}$$

Der reflektierte Anteil $L_r(x, \lambda, \omega_o)$ ergibt sich damit aus der mit $\frac{1}{\pi}$ skalierten Irradianz $R(x, \lambda)$ [31, S. 785 f]. Insbesondere ist er damit unabhängig von der Richtung ω_o . Das Einsetzen in den übrigen Teil der Gleichung ergibt dann direkt

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \frac{1}{\pi} R(x, \lambda)$$

In den meisten Fällen ist E selbst unabhängig von ω_o und an vielen Punkten im Raum sogar identisch mit Null. Für matte Oberflächen, deren Reflexionsverhalten durch einen Lambertschen Strahler charakterisiert werden kann, reicht es dementsprechend die Irradianz zu betrachten [40, 41].

Verwendet man jetzt das Korollar des vorigen Abschnittes, so lässt sich das einfache Modell der Lambertschen diffusen Reflexion zum Beispiel mit einer idealen Reflexion verbinden, wie es in Abbildung 12 gezeigt ist. Das resultierende Material ist komplexer aufgebaut und beschreibt eine andere Klasse von BSDFs. Die Verwendung eines Lambertschen Strahlers kann somit durch Kombination mit anderen BSDF-Modellen auf einen größeren Bereich von Materialarten ausgeweitet werden. Mithin ist es in der Praxis in fast allen Szenen nötig, diffuse Reflexionen an den Oberflächen der Objekte auszuwerten.

Wir nehmen jetzt an, dass sich die BSDF am Punkt x aus der Kombination eines Lambertschen Strahlers mit Koeffizient α und einer beliebigen BSDF $\tilde{f}(x, \cdot, \cdot)$ mit Koeffizient β ergibt. Dabei fordern wir, dass $\alpha, \beta \in [0, 1]$ und $\alpha + \beta \leq 1$ gilt. Das Korollar über die Summe von BSDFs tätigt dann eine klare Aussage über die Form des reflektierten Anteils.

$$L(x, \lambda, \omega_o) = E(x, \lambda, \omega_o) + \frac{\alpha}{\pi} R(x, \lambda) + \beta \int_{S^2} \tilde{f}(x, \omega, \omega_o) \tilde{L}(x, \lambda, \omega) |\langle \nu(x), \omega \rangle| d\sigma(\omega)$$

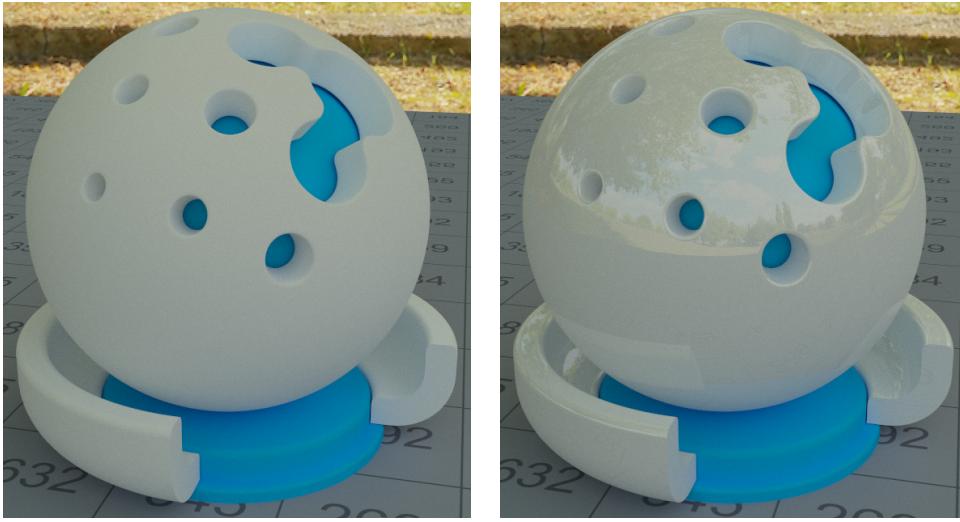


Abbildung 12: Die Bilder zeigen die gerenderte »Shaderball«-Szene, beleuchtet durch eine Umgebungsbeleuchtung. Im linken Bild bestehen die verwendeten BSDFs nur aus der lambertsch diffusen Reflexion. Im rechten Bild wurde das Material der äußeren Körpers zusätzlich mit der BSDF einer idealen Reflexion kombiniert. Das entstehende Material ist kein idealer Spiegel, aber auch kein idealer diffuser Strahler.

Diese Form der Rendergleichung für Lambertsche Strahler besitzt für die Verwendung in der Praxis einen Vorteil. Die BSDF $\tilde{f}(x, \cdot, \cdot)$ ergibt sich meistens aus stark richtungsabhängigen Anteilen, wie zum Beispiel der idealen Reflexion, der idealen Brechung oder ähnlich glänzenden Materialien. Diese Anteile können vergleichsweise gut durch Raytracing beziehungsweise auch Path Tracing gerendert werden. Die Berechnung von $R(x, \lambda)$ erfordert jedoch immer eine Integration über die gesamte Hemisphäre, bei der jeder erhaltene Wert gleichermaßen gewichtet werden muss. Diese Tatsache erschwert die Evaluierung und führt zu entsprechend langen Zeiträumen des Renderings.

Ein Vorteil der Irradianz besteht jedoch darin, dass sie unabhängig von der betrachteten Richtung ist. Die empfangene Strahldichte von x ist konstant, egal von welchem Punkt im Raum aus man x betrachtet. Diese Eigenschaft ermöglicht es, $R(x, \lambda)$ vorzuberechnen und am Punkt x zu speichern. Jedes Mal, wenn man nun den Punkt x aus einer Richtung $\tilde{\omega} \in \mathcal{H}_{\nu(x)}^2$ betrachtet, wird das Integral des rechten Teils der Gleichung effizient durch Path Tracing geschätzt. Der Wert $R(x, \lambda)$ kann mit $E(x, \lambda, \tilde{\omega})$ ausgelesen und zum resultierenden Wert hinzugefügt werden. Infolgedessen verkürzen sich die benötigten Renderzeiten. Da es sich bei R um die Irradianz handelt, wollen wir die zugehörige Datenstruktur, die ihre Werte speichert, »Irradiance Map« nennen.

Wir möchten hier bemerken, dass die genannten Eigenschaften nur für die Translation des Beobachtungspunktes gelten. Die Änderung der Beleuchtung oder Geometrie führt zu einer neuen Szene $\tilde{\Sigma}$, deren Irradianz \tilde{R} im Allgemeinen nicht R entspricht. Die Irradiance Map muss demnach für jede Modifizierung der Szene neu berechnet werden, um die korrekten Werte der Irradianz zu beinhalten.

3.3 Schätzung der Irradianz

In Abschnitt 2.6 wurde erwähnt, dass die meisten Renderverfahren, wie zum Beispiel das hier verwendete Path Tracing, auf der Basis von Zufallsvariablen arbeiten. Das Integral in der Rendergleichung wird dabei mithilfe geeigneter Monte-Carlo-Methoden berechnet. Aus diesem Grund bildet sich bei den Lösungen der Rendergleichung immer ein gewisses »Rauschen« (engl.: *noise*) aus. In Abbildung 13 wird dies an einem Beispiel demonstriert.

Möchte man die Irradianz für die Irradiance Map ermitteln, so muss darauf geachtet werden, dass das Rauschen des Endresultates und damit dessen Fehler klein bleibt. Erst durch die Beschränkung des Fehlers können gemessene Irradianzen an verschiedenen Punkten interpoliert werden, ohne das



Abbildung 13: Die Bilder zeigen die gerenderte »Fairy«-Szene. Für das linke Bild wurde an jedem sichtbaren Punkt der Szene die Irradianz durch 16 Messungen ermittelt. Das Resultat beinhaltet noch einen großen Anteil des Rauschens. Im rechten Bild wurden jeweils 1024 Messungen durchgeführt, wodurch es wesentlich rauschärmer als das Linke ist.

Artefakte wie in der späteren Abbildung 15b entstehen.

Für die folgende Betrachtung nehmen wir wieder an, dass eine Szene $\Sigma := (\mathcal{T}, \nu, f, E, U)$ und eine Strahldichte L von Σ , die der Rendergleichung genügt, gegeben seien. Wir bezeichnen R als die Irradianz von L und wählen eine festen Punkt $x \in \mathcal{T}$ und eine feste Wellenlänge $\lambda \in (0, \infty)$. Um die Zufälligkeit der Simulation mathematisch zu fassen, verwenden wir einen Wahrscheinlichkeitsraum (X, \mathcal{X}, p) und für jedes $\omega \in \mathcal{S}^2$ die folgende Zufallsvariable.

$$\mathcal{L}_\omega: X \rightarrow [0, \infty), \quad \mathbb{E} \mathcal{L}_\omega = \tilde{L}(x, \lambda, \omega)$$

Die Simulation von $\tilde{L}(x, \lambda, \omega)$ durch Path Tracing oder einem ähnlichen Algorithmus entspricht dann der Realisierung $\mathcal{L}_\omega(\xi)$ mit $\xi \in X$. Für die Irradianz $R(x, \lambda)$ führen wir einen weiteren Wahrscheinlichkeitsraum (Y, \mathcal{Y}, q) ein. Seien $n \in \mathbb{N}$ und unabhängige, identische Zufallsvariablen $w_i: Y \rightarrow \mathcal{H}_{\nu(x)}^2$ für alle $i \in \mathbb{N}, i \leq n$ mit der folgenden zugehörigen Wahrscheinlichkeitsdichte bezüglich σ gegeben.

$$\varrho: \mathcal{H}_{\nu(x)}^2 \rightarrow [0, \infty), \quad \varrho(\omega) := \frac{1}{2\pi}$$

Die w_i sind damit für alle $i \in \mathbb{N}, i \leq n$ auf der Hemisphere $\mathcal{H}_{\nu(x)}^2$ gleichverteilt. Wir sind nun in der Lage weitere Zufallsvariablen \mathcal{R}_i für alle $i \in \mathbb{N}, i \leq n$ auf dem Produkt-Wahrscheinlichkeitsraum $(X \times Y, \mathcal{X} \otimes \mathcal{Y}, p \otimes q)$ zu definieren, deren Erwartungswert eine Aussage über die Irradianz trifft.

$$\mathcal{R}_i: X \times Y \rightarrow [0, \infty), \quad \mathcal{R}_i(\xi, \zeta) := \mathcal{L}_{w_i(\zeta)}(\xi) \langle \nu(x), w_i(\zeta) \rangle$$

Die folgende Rechnung zeigt den Zusammenhang des Erwartungswertes von \mathcal{R}_i mit $R(x, \lambda)$ für alle $i \in \mathbb{N}, i \leq n$.

$$\begin{aligned} \mathbb{E} \mathcal{R}_i &= \int_{X \times Y} \mathcal{R}_i \, d(p \otimes q) \\ (\text{Satz von Fubini [9, S. 175 f]}) \quad &= \int_Y \int_X \mathcal{R}_i(\xi, \zeta) \, dp(\xi) \, dq(\zeta) \\ (\text{Definition } \mathcal{R}_i) \quad &= \int_Y \int_X \mathcal{L}_{w_i(\zeta)}(\xi) \langle \nu(x), w_i(\zeta) \rangle \, dp(\xi) \, dq(\zeta) \\ (\text{Linearität Integral}) \quad &= \int_Y \langle \nu(x), w_i(\zeta) \rangle \int_X \mathcal{L}_{w_i(\zeta)}(\xi) \, dp(\xi) \, dq(\zeta) \end{aligned}$$

$$\begin{aligned}
(\text{Definition Erwartungswert}) &= \int_Y \langle \nu(x), w_i(\zeta) \rangle \mathbb{E} \mathcal{L}_{w_i(\zeta)} dq(\zeta) \\
(\text{Definition } \mathcal{L}_{w_i(\zeta)}) &= \int_Y \tilde{L}(x, \lambda, w_i(\zeta)) \langle \nu(x), w_i(\zeta) \rangle dq(\zeta) \\
(\text{Transformation [9, S. 191 f]}) &= \int_{w_i(Y)} \tilde{L}(x, \lambda, \zeta) \langle \nu(x), \zeta \rangle dq_{w_i}(\zeta) \\
(\text{Maße mit Dichten [9, S. 127 f]}) &= \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \nu(x), \omega \rangle \varrho(\omega) d\sigma(\omega) \\
(\text{Definition } \varrho) &= \frac{1}{2\pi} \int_{\mathcal{H}_{\nu(x)}^2} \tilde{L}(x, \lambda, \omega) \langle \nu(x), \omega \rangle d\sigma(\omega) \\
(\text{Definition Irradianz}) &= \frac{R(x, \lambda)}{2\pi}
\end{aligned}$$

Durch diese Äquivalenz erlangen wir die Möglichkeit einen erwartungstreuen Schätzer $\bar{\mathcal{R}}$ der Irradianz $R(x, \lambda)$ zu definieren.

$$\bar{\mathcal{R}} := \frac{2\pi}{n} \sum_{i=1}^n \mathcal{R}_i$$

Nach Quelle [14, S. 249 ff] ergibt sich dann mit einem $i \in \mathbb{N}, i \leq n$ für den Erwartungswert und die Varianz

$$\mathbb{E} \bar{\mathcal{R}} = R(x, \lambda), \quad \text{var } \bar{\mathcal{R}} = \frac{4\pi^2}{n} \text{ var } \mathcal{R}_i$$

Wir können also zufällige Richtungen der Hemisphere auswählen und für diese die Strahldichte ermitteln. Das arithmetische Mittel sollte dann nach dem »schwachen Gesetz der großen Zahlen« [14, S. 254] für eine steigende Anzahl von Messwerten gegen den Wert der Irradianz konvergieren. Es sei hier auch auf die Quellen [13, 32, 36] verwiesen, die diese Herleitung mithilfe der »Pfadintegral-Formulierung« der Rendergleichung vollführen.

Das eben beschriebene Verfahren ist ein typisches Beispiel einer Monte-Carlo-Methode [24]. Die Abbildungen 14 und 15 zeigen es anhand zweier Beispielezenen für verschiedene $n \in \mathbb{N}$.

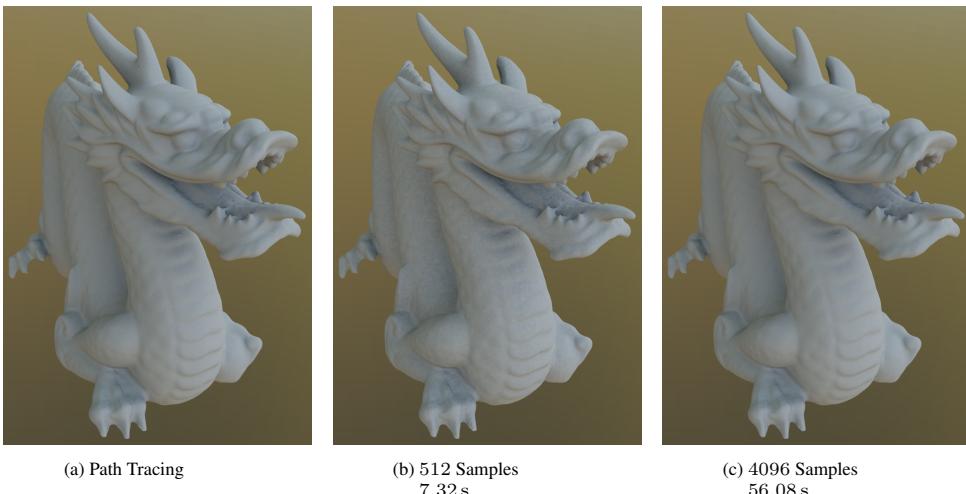


Abbildung 14: Die Bilder zeigen die gerenderte »Dragon«-Szene mit einer schwach variierenden Umgebungsbeleuchtung. 14a ist ein durch Path Tracing aufgenommenes Referenzbild. Für 14b und 14c wurden die Irradianzen an den Eckpunkten der Mesh geschätzt und gespeichert. Zwischen den Eckpunkten fand eine lineare Interpolation statt. Die Anzahl der Samples gibt an, wie viele Stichprobenwerte verwendet wurden, um die Irradianz an jedem Punkt zu schätzen. Die Zeit entspricht dabei der Berechnungsdauer. 14c weist zum Referenzbild keine Unterschiede auf. In 14b sind jedoch einige schwache Artefakte in Form von Lichtflecken zu beobachten.

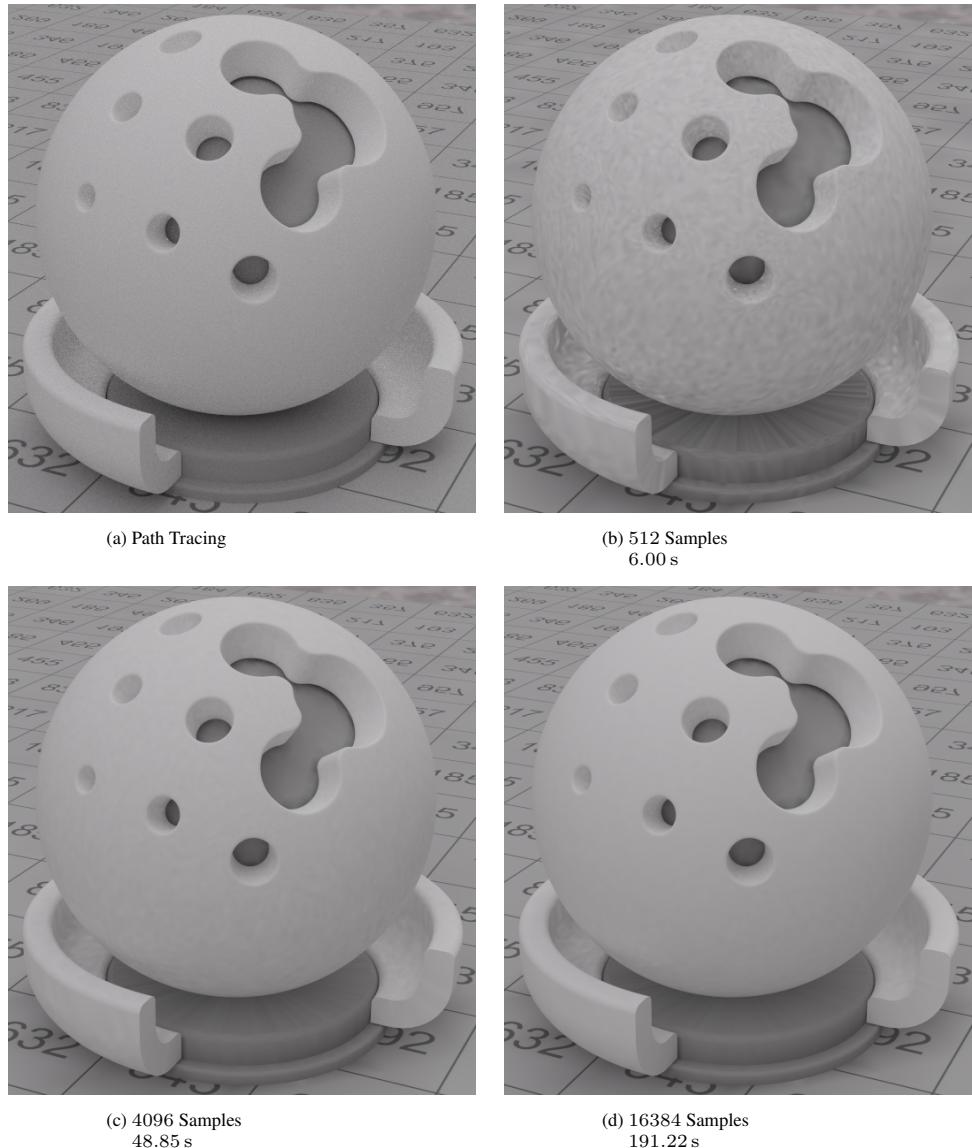


Abbildung 15: Die Bilder zeigen die gerenderte »Shaderball«-Szene entsprechend der Benennung aus Abbildung 14. Die Monte-Carlo-Integration erzeugt in 15b mit nur 2^9 Samples ein fleckiges Muster, welches durch die Verteilung der Zufallsvariable um den Erwartungswert hervorgerufen wird. In 15c und 15d ist erkennbar, dass die Entstehung von Flecken durch eine höhere Sampleanzahl verringert werden kann. Dennoch sind in 15d auch für 2^{14} Samples im unteren Stand der Szene noch Flecken sichtbar.

Die Ergebnisse des Algorithmus variieren stark mit der Szenengeometrie und der Umgebungsbeleuchtung. Während im Bild 14c in der »Dragon«-Szene mit einer einfachen Umgebungsbeleuchtung bereits nach 4096 Messungen an einem Punkt kein Unterschied mehr zum Referenzbild 18a festgestellt werden kann, sind in der »Shaderball«-Szene in Bild 15d mit 16384 Messungen pro Punkt immer noch fleckige Artefakte zu sehen.

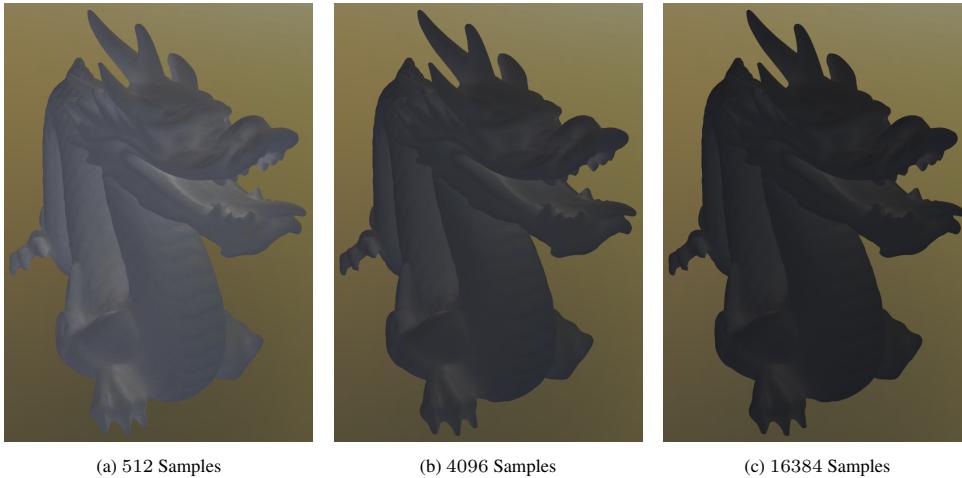


Abbildung 16: Die Bilder zeigen die geschätzten Standardabweichungen der Irradianzen an den Eckpunkten der »Dragon«-Szene analog zu Abbildung 14. Eine hellere Farbe bedeutet dabei eine höhere Standardabweichung. Deutlich erkennbar ist die Abnahme des Fehlers für größere Sampleanzahlen. Dennoch bleiben auch für 2^{14} Samples noch Stellen übrig, die einen vergleichsweise hohen Fehler aufweisen.

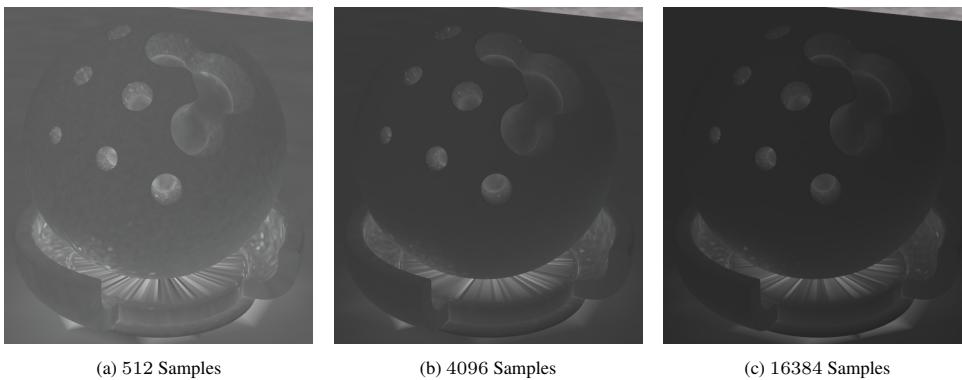


Abbildung 17: Die Bilder zeigen die Standardabweichungen der »Shaderball«-Szene aus Abbildung 15 entsprechend der Benennung aus Abbildung 14. Vor allem Bereiche, die schwach von außen beleuchtet werden oder in Nischen liegen, weisen einen erhöhten Fehler auf. Die regelmäßigen Artefakte entstehen durch geringe Auflösung der Mesh in diesen Bereichen.

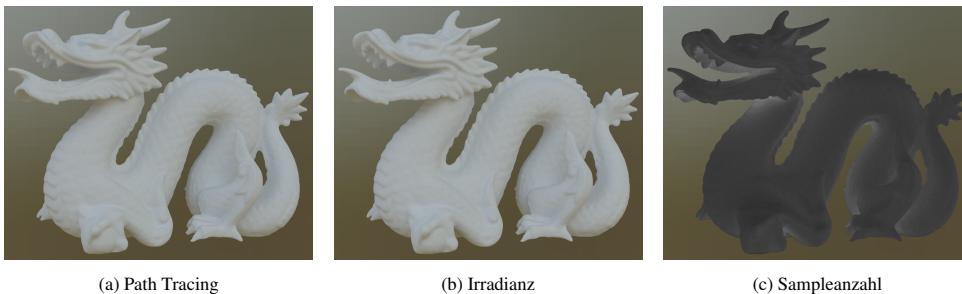


Abbildung 18: Die Bilder zeigen die »Dragon«-Szene mit einer schwach variierenden Umgebungsbeleuchtung. 18a ist das durch Path Tracing erzeugte Referenzbild. 18b zeigt die an den Eckpunkten adaptiv aufgenommenen Irradianzen mit linearer Interpolation. 18c stellt dabei die zugehörigen Sampleanzahlen dar. Hier bedeutet eine hellere Farbe eine höhere Sampleanzahl. Die Berechnungszeit betrug 253 s mit der maximalen Sampleanzahl 2^{16} und einem maximalen relativen Fehler von 3 %.

Die ersten beiden dienen dem Vergleich der Korrektheit des Algorithmus. Das dritte Bild

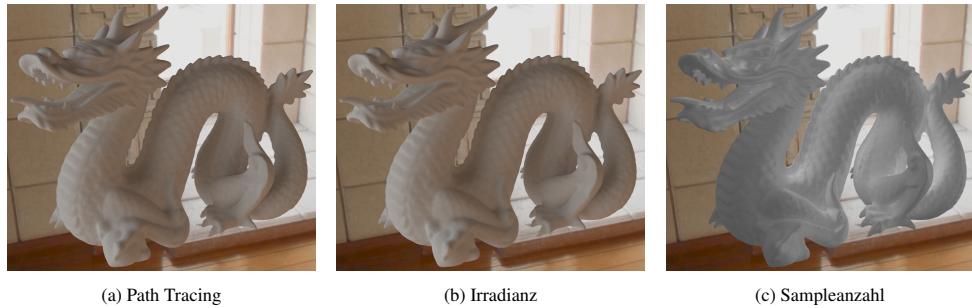


Abbildung 19: Die Bilder zeigen die »Dragon«-Szene mit einer stark variierenden Umgebungsbeleuchtung analog zu Abbildung 18. Die Berechnungszeit betrug 559 s mit der maximalen Sampleanzahl 2^{16} und einem maximalen relativen Fehler von 3 %.

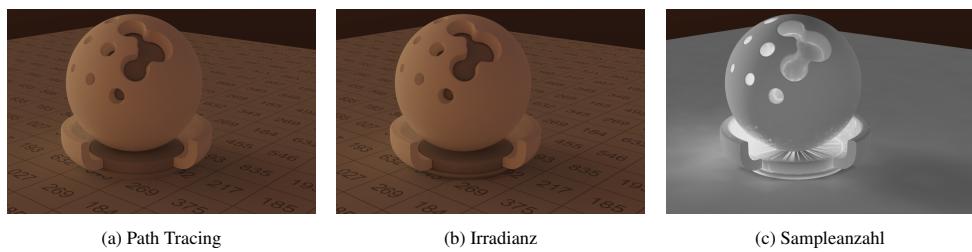


Abbildung 20: Die Bilder zeigen die »Shaderball«-Szene mit einer stark variierenden Umgebungsbeleuchtung analog zu Abbildung 18. Die Berechnungszeit betrug 1170 s mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.



Abbildung 21: Die Bilder zeigen die »Shaderball«-Szene der Abbildung 20 aus einem anderen Blickwinkel. Die aufgenommenen Irradianzen beinhalten im unteren Stand weniger Rauschen als das Referenzbild.

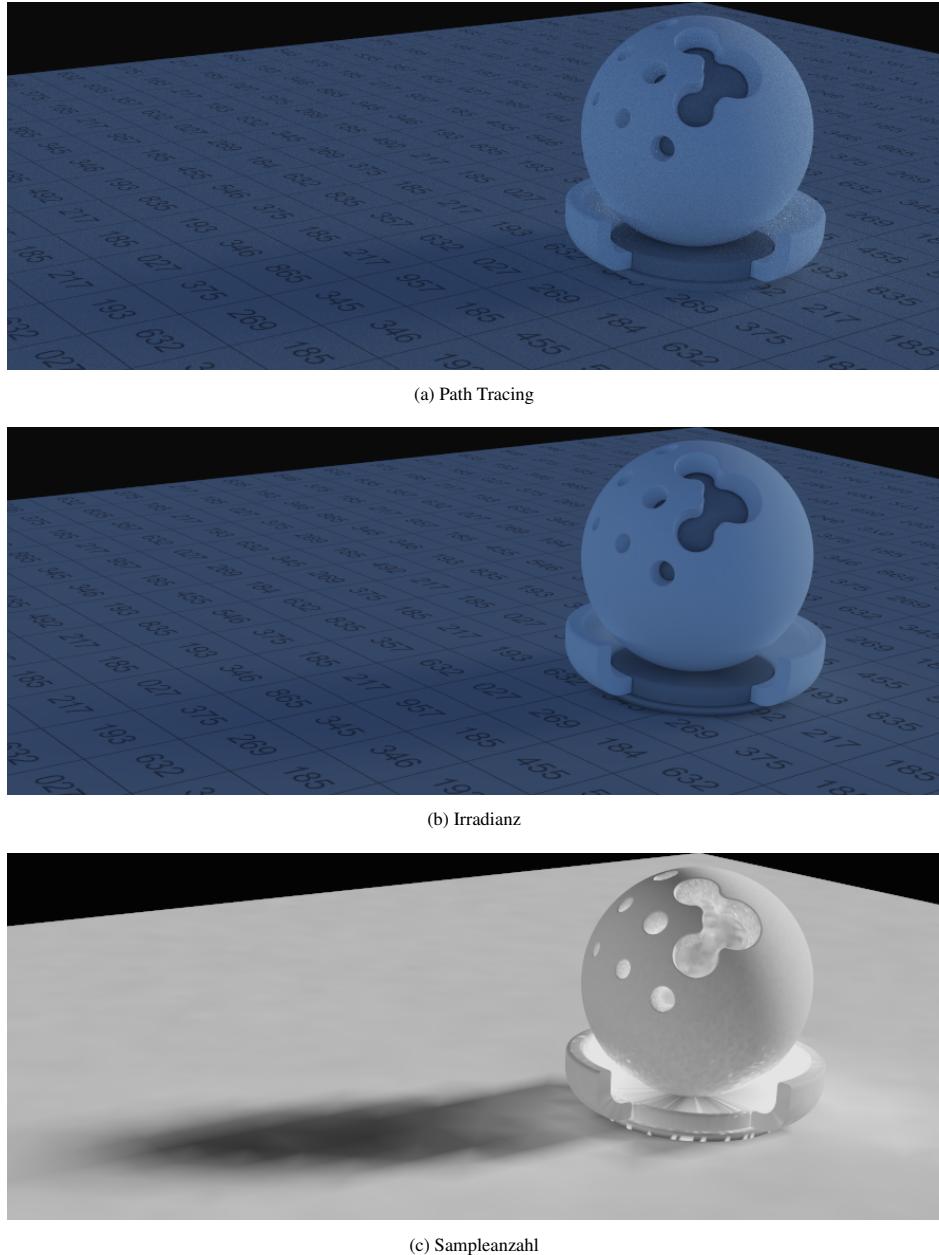
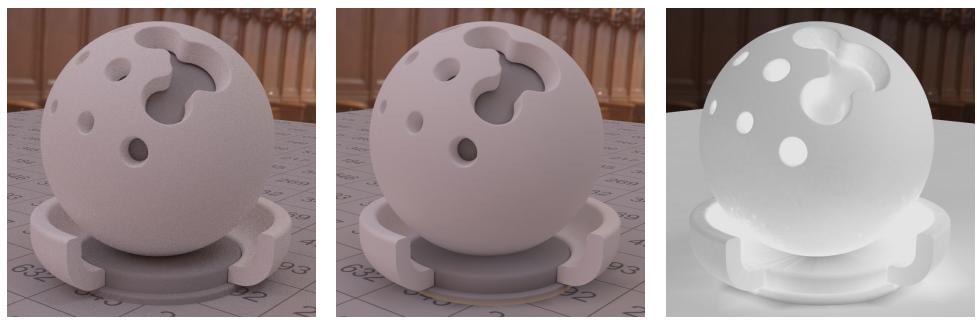


Abbildung 22: Die Bilder zeigen die »Shaderball«-Szene mit einer stark konzentrierten Umgebungsbeleuchtung analog zu Abbildung 18. Die Berechnungszeit betrug 1090 s mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.

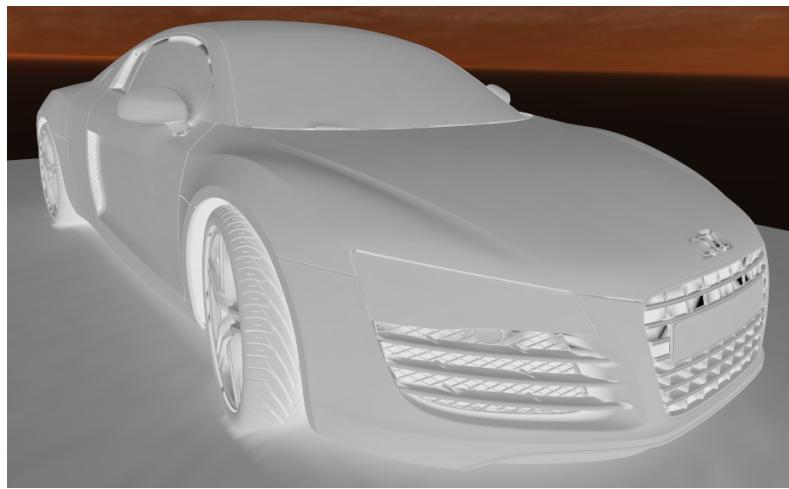




(a) Path Tracing



(b) Irradianz



(c) Sampleanzahl

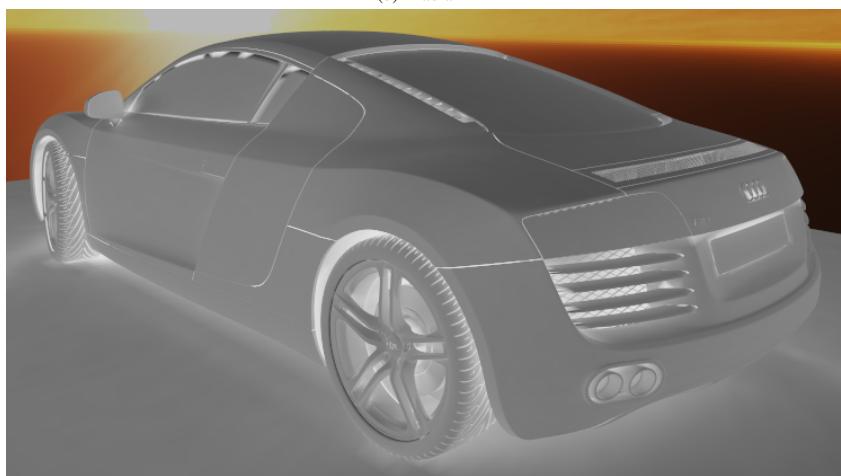
Abbildung 24: Die Bilder zeigen die »Audi R8«-Szene mit der »Sky 20«-Umgebungsbeleuchtung analog zu Abbildung 18. Die Berechnungszeit betrug 17.8 h mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.



(a) Path Tracing



(b) Irradianz

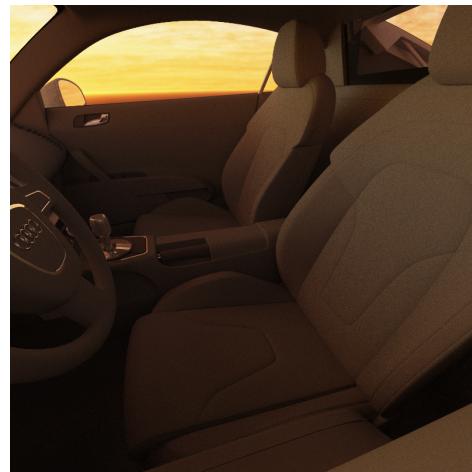


(c) Sampleanzahl

Abbildung 25: Die Bilder zeigen die »Audi R8«-Szene mit der »Sky 20«-Umgebungsbeleuchtung analog zu Abbildung 18. Die Berechnungszeit betrug 17.8 h mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.



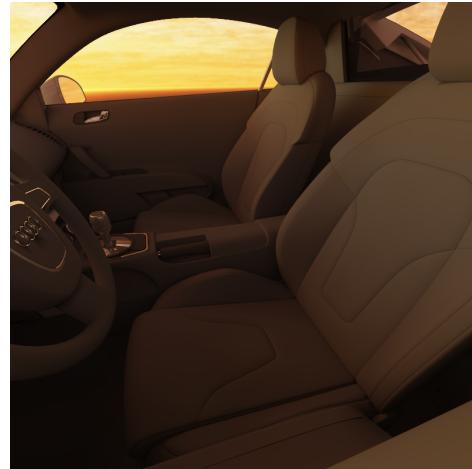
(a) Path Tracing



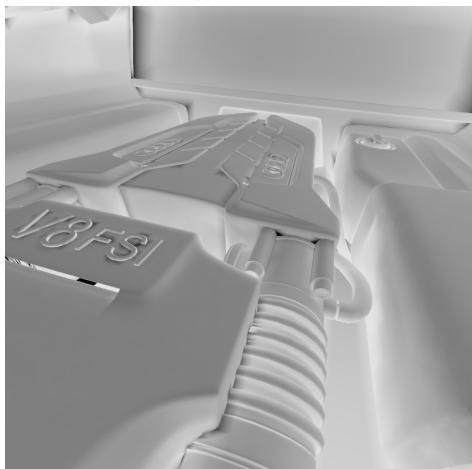
(b) Path Tracing



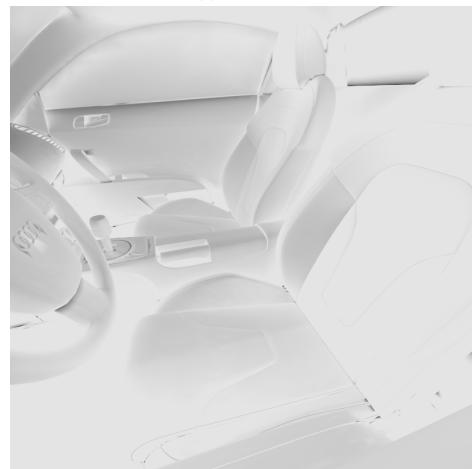
(c) Irradianz



(d) Irradianz



(e) Sampleanzahl



(f) Sampleanzahl

Abbildung 26: Die Bilder zeigen die »Audi R8«-Szene mit der »Sky 20«-Umgebungsbeleuchtung analog zu Abbildung 18. Die Berechnungszeit betrug 17.8 h mit der maximalen Sampleanzahl 2^{18} und einem maximalen relativen Fehler von 1 %.

Quelltext: Adaptive Irradianzschätzung

```

vecf4 est_irr_adap(const vecf4& pos, const vecf4& n, uint& sample_count){
    const uint max_count = max_sample_count;
    const uint min_count = 256;
    const float rel_err_eps = 0.0001f;

    // initialisiere temporaere Variablen
    vecf4 sum(0,0,0);
    vecf4 sum_sq(0,0,0);
    uint count = 0;
    float rel_err = 0.0f;

    // generiere weitere Samples, bis Abbruchbedingung erreicht
    while ((count < max_count) && ((count < min_count) || (rel_err >
        rel_err_bound))){

        // konstruiere Strahl fuer weitere Messung
        ray r;
        r.dir = rnd_dir();
        r.ori = pos;

        float tmp_dot = vecf4::dot(n,r.dir);
        // Strahl muss nach aussen zeigen
        if (tmp_dot < 0.0f){
            tmp_dot = -tmp_dot;
            r.dir = -r.dir;
        }
        // verhindere Schnittpunkt mit sich selbst
        r.ori += ray_eps * r.dir;

        // berechne skalierte einfallende Strahldichte durch Path Tracing
        const vecf4 tmp = tmp_dot * path_trace(r);

        // schaetze fuer Sampleanzahl Mittelwert und Varianz
        sum += tmp;
        sum_sq += tmp * tmp;
        count++;
        const float inv_count = 1.0f / float(count);
        const float inv_count1 = 1.0f / float(count-1);
        const vecf4 mean = sum * inv_count;
        const vecf4 var = ((sum_sq * inv_count1) - (float(count)*inv_count1 *
            mean * mean)) * inv_count;
        // berechne den relativen Fehler
        rel_err = max(sqrt(var) / (mean + rel_err_eps));
    }

    // setze zurueckgegebene Sampleanzahl
    sample_count = count;

    // gebe geschaetzte Irradianz zurueck
    return sum * 2.0f * M_PI / float(count);
}

```


4 Aufbau und Generierung der Irradiance Map

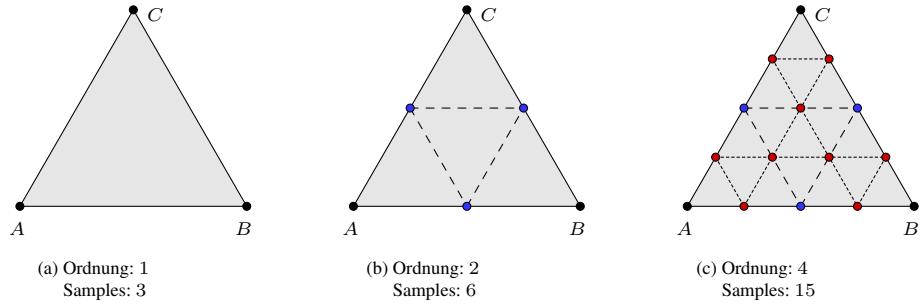


Abbildung 27: Die Abbildung zeigt das Schema der Dreieck Irradianc Map Datenstruktur auf einem Dreieck (A, B, C) für verschiedene Ordnungen.

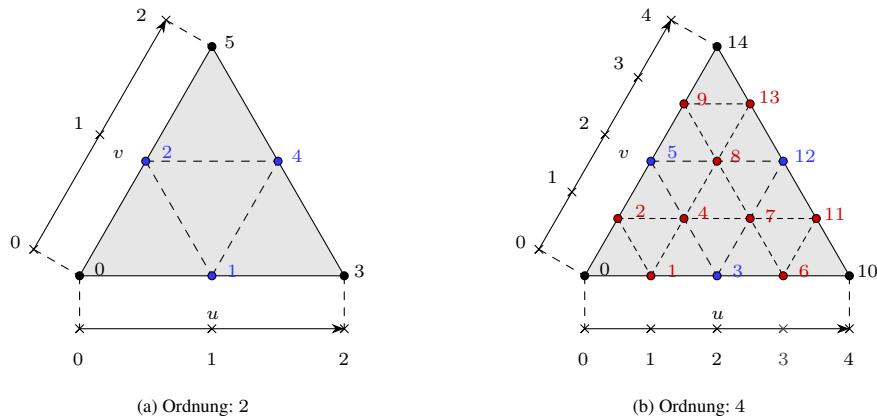


Abbildung 28: Die Abbildung zeigt das Schema der Dreieck Irradianc Map Datenstruktur für verschiedene Ordnungen.

$$s: \mathbb{N} \rightarrow \mathbb{N}, \quad s(n) := \frac{(n+1)(n+2)}{2}$$

$$U_n := \{(u, v) \in \mathbb{N}_0^2 \mid u + v \leq n\}, \quad I_n := \{i \in \mathbb{N}_0 \mid i < s(n)\}$$

$$m_n: U_n \rightarrow I_n, \quad m_n(u, v) := \frac{(u+v)(u+v+1)}{2} + v$$

$$m_{n,p}: U_n \rightarrow I_{n \cdot 2^p}, \quad m_{n,p}(u, v) := m_{n \cdot 2^p}(u \cdot 2^p, v \cdot 2^p)$$

Quelltext: Dreieck Irradiance Map Datenstruktur

```
namespace irr_map{

struct triangle_uvmap{
    uint order;
    vecf4 *data;

    vecf4 operator()(float u, float v) const;
};

}
```

```

inline uint size(uint order){
    return ((order+1)*(order+2))>>1;
}

inline uint memidx(uint uidx, uint vidx){
    const uint sum = uidx + vidx;
    return ((sum * (sum+1)) >> 1) + vidx;
}

inline uint memidx(uint uidx, uint vidx, uint shift){
    const uint suidx = uidx << shift;
    const uint svidx = vidx << shift;
    return memidx(suidx, svidx);
}

inline vec4f triangle_uvmap::operator()(float u, float v) const{
    const float nu = order * u;
    const float nv = order * v;
    const float uidx = floorf(nu);
    const float vidx = floorf(nv);
    const float wu = nu - uidx;
    const float vv = nv - vidx;
    const float sum_w = wu + vv;

    if (sum_w > 1.0f){
        const uint idx = memidx(uidx+1, vidx);
        const uint sum = uidx + vidx + 1;

        return (1.0f-wu) * data[idx + 1] + // memidx(u, v+1)
            (1.0f-vv) * data[idx] + // memidx(u+1, v)
            (sum_w-1.0f) * data[idx + sum + 2]; // memidx(u+1, v+1)
    } else{
        const uint idx = memidx(uidx, vidx);
        const uint sum = uidx + vidx;

        return wu * data[idx + sum + 1] + // memidx(u+1, v)
            vv * data[idx + sum + 2] + // memidx(u, v+1)
            (1.0f-sum_w) * data[idx]; // memidx(u, v)
    }
}

} // irr_map

```

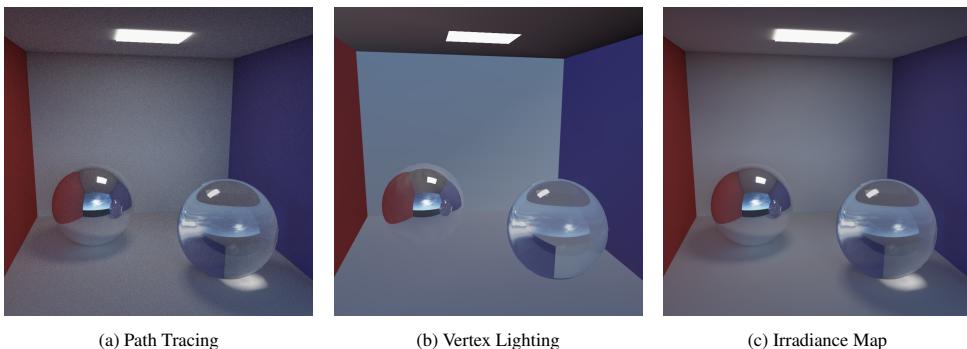


Abbildung 29: Die Bilder zeigen die Standardabweichungen der »Cornell Box«-Szene aus Abbildung 15 entsprechend der Benennung aus Abbildung 14. Vor allem Bereiche, die schwach von außen beleuchtet werden oder in Nischen liegen, weisen einen erhöhten Fehler auf. Die regelmäßigen Artefakte entstehen durch geringe Auflösung der Mesh in diesen Bereichen.

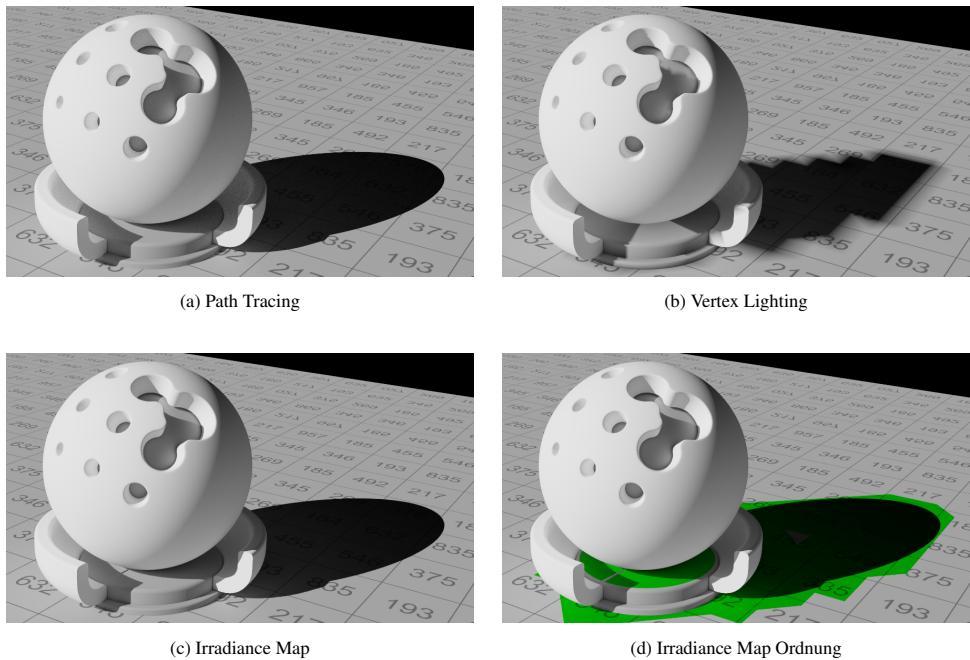


Abbildung 30: Die Bilder zeigen die Standardabweichungen der »Cornell Box«-Szene aus Abbildung 15 entsprechend der Benennung aus Abbildung 14. Vor allem Bereiche, die schwach von außen beleuchtet werden oder in Nischen liegen, weisen einen erhöhten Fehler auf. Die regelmäßigen Artefakte entstehen durch geringe Auflösung der Mesh in diesen Bereichen.

Quelltext: Irradiance Map Generator

```

namespace irr_map{

    vecf4 irr(triangle* prim, float u, float v){
        const float w = 1.0f - u - v;

        pos =
            prim->v0.pos * w +
            prim->v1.pos * u +
            prim->v2.pos * v;

        n = prim->v0.n * w +
            prim->v1.n * u +
            prim->v2.n * v;

        vecf4::normalize(n);

        uint sample_count;
        return est_irr_adap(pos, n, sample_count);
    }

    void gen_irr_map(){
        const float rel_err_eps = 0.0001f;
        const float border_eps = 0.00001f;
        const float rel_err_bound = ctrlWnd->irrmap_eps_dsb->value();

        const float samples_on_edge = ctrlWnd->sample_diag_dsb->value() * prim->edge
            / bvh->edge_mean;
        const float cur_max_order = samples_on_edge+1.0f;
        const uint cur_max_exp = max(ceilf(log2f(cur_max_order)), 1.0f);
    }
}

```

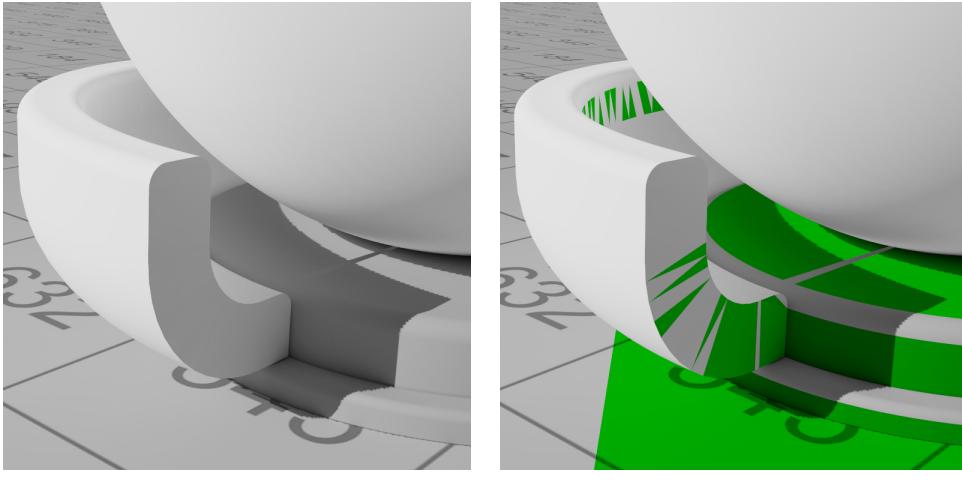


Abbildung 31: Die Bilder zeigen die Standardabweichungen der »Cornell Box«-Szene aus Abbildung 15 entsprechend der Benennung aus Abbildung 14. Vor allem Bereiche, die schwach von außen beleuchtet werden oder in Nischen liegen, weisen einen erhöhten Fehler auf. Die regelmäßigen Artefakte entstehen durch geringe Auflösung der Mesh in diesen Bereichen.

```

// const uint max_exp = 7;
const uint read_max_exp = ctrlWnd->irrmap_max_order_sb->value();
const uint max_exp = (cur_max_exp < read_max_exp)?(cur_max_exp):(read_max_exp
    );
uint max_order = 1 << max_exp;
const uint half_max_order = max_order >> 1;
const uint max_size = size(max_order);
const uint half_max_size = size(half_max_order);

// memory indices of vertices
const uint a = 0;
const uint b = max_size-max_order-1;
const uint c = max_size-1;

// memory indices of half edge samples
const uint eab = half_max_size-half_max_order-1;
const uint ebc = max_size-half_max_order-1;
const uint eca = half_max_size-1;

vecf4 data[max_size];
triangle* prim;

while(prim = next_triangle()){
    uint cur_exp = 1;
    uint cur_order = 1 << cur_exp;

    // vertices
    data[a] = irr(prim, border_eps, border_eps);
    data[c] = irr(prim, border_eps, 1.0f - 2.0f * border_eps);
    data[b] = irr(prim, 1.0f - 2.0f*border_eps, border_eps);

    // edges
    data[eca] = irr(prim, border_eps, 0.5f);
    data[eab] = irr(prim, 0.5f, border_eps);
    data[ebc] = irr(prim, 0.5f - border_eps, 0.5f - border_eps);

    float rel_err = max(fabsf( 0.5f*(data[a] + data[b]) - data[eab] ) / (

```

```

        rel_err_eps + data[eab]));
rel_err = max(rel_err, max(fabsf( 0.5f*(data[b] + data[c]) - data[ebc] )
/ (rel_err_eps + data[ebc])));
rel_err = max(rel_err, max(fabsf( 0.5f*(data[c] + data[a]) - data[eca] )
/ (rel_err_eps + data[eca])));

while (rel_err > rel_err_bound && cur_exp < max_exp){
    cur_exp++;
    cur_order = 1 << cur_exp;
    float inv_cur_order = 1.0f / float(cur_order);

    // compute new samples and residual norm
    const uint shift = max_exp - cur_exp;

    // edges
    for (uint i = 1; i < cur_order; i+=2){
        const float tmp = float(i) * inv_cur_order;

        uint idx = memidx(i,0,shift);
        data[idx] = irr(prim, tmp, border_eps);

        idx = memidx(0,i,shift);
        data[idx] = irr(prim, border_eps, tmp);

        idx = memidx(i,cur_order-i,shift);
        data[idx] = irr(prim, tmp-border_eps, 1.0f-tmp-border_eps);

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,0,shift)
] + data[memidx(i+1,0,shift)]) - data[memidx(i,0,shift)] )
/ (rel_err_eps + data[memidx(i,0,shift)])));

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(0,i-1,shift)
] + data[memidx(0,i+1,shift)]) - data[memidx(0,i,shift)] )
/ (rel_err_eps + data[memidx(0,i,shift)])));

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,
cur_order-i+1,shift)] + data[memidx(i+1,cur_order-i-1,shift)
]) - data[memidx(i,cur_order-i,shift)] ) / (rel_err_eps +
data[memidx(i,cur_order-i,shift)])));
    }

    // inner
    // odd i
    for (uint i = 1; i < cur_order; i+=2){
        // odd j
        for (uint j = 1; i+j < cur_order; j+=2){
            const float u = float(i) * inv_cur_order;
            const float v = float(j) * inv_cur_order;
            data[memidx(i,j,shift)] = irr(prim, u, v);

            rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,j
+1,shift)] + data[memidx(i+1,j-1,shift)]) - data[memidx(
i,j,shift)] ) / (rel_err_eps + data[memidx(i,j,shift)])));
        }

        // even j
        for (uint j = 2; i+j < cur_order; j+=2){
            const float u = float(i) * inv_cur_order;
            const float v = float(j) * inv_cur_order;
            data[memidx(i,j,shift)] = irr(prim, u, v);

            rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i,j-1,
shift)] + data[memidx(i,j+1,shift)]) - data[memidx(i,j,
shift)] ) / (rel_err_eps + data[memidx(i,j,shift)])));
        }
    }
}

```

```

        shift) ] ) / (rel_err_eps + data[memidx(i,j,shift))));

    }

}

// even i
for (uint i = 2; i < cur_order; i+=2){
    // odd j
    for (uint j = 1; i+j < cur_order; j+=2){
        const float u = float(i) * inv_cur_order;
        const float v = float(j) * inv_cur_order;
        data[memidx(i,j,shift)] = irr(prim, u, v);

        rel_err = max(rel_err, max(fabsf( 0.5f * (data[memidx(i-1,j,
            shift)] + data[memidx(i+1,j,shift)]) - data[memidx(i,j,
            shift)] ) / (rel_err_eps + data[memidx(i,j,shift))));

    }
}
}

const uint shift = max_exp - cur_exp;
prim->irr_map.order = cur_order;
prim->irr_map.data = (vecf4*)mem_alloc(sizeof(cur_order) * sizeof(vecf4));

uint idx = 0;
for (uint r = 0; r <= cur_order; r++){
    for (uint j = 0; j <= r; j++){
        const uint i = r - j;
        prim->irr_map.data[idx] = data[memidx(i,j,shift)];
        idx++;
    }
}
}

} // irr_map

```

5 Fazit und Aussicht

Das in der Arbeit angewendete Verfahren zur Schätzung der Irradianz an der Oberfläche der Mesh wurde zuerst nur mithilfe von Vertex Lighting implementiert. Für hochauflöste Meshes ist dieses Verfahren eine gute Approximation und brilliert durch seine Einfachheit. Da die meisten Szenen in dieser Arbeit diese Eigenschaft jedoch nicht erfüllen, musste eine Alternative, wie die Irradiance Map, gefunden werden.

Die adaptive Schätzung der Irradianz führt für eine vorgegebene relative Fehlerschranke und einer maximalen Sampleanzahl zu akzeptablen Ergebnissen. In den meisten Fällen konnten keine Unterschiede zum Referenzbild ausgemacht werden. Insbesondere wurde im Laufe der Untersuchung festgestellt, dass die besten Ergebnisse für eine relative Fehlerschranke von 1 % und einer maximalen Sampleanzahl von 2^{16} bis 2^{18} entstehen. Auftretende Fehler in diesem Bereich sind für das menschliche Auge kaum noch wahrnehmbar. Für sehr dunkle Bereiche der Darstellung sollte allerdings noch eine weitere Fehlermetrik verwendet werden, die die Ausbildung schwarzer Flecken in schwach beleuchteten Bereichen verhindert.

Die Datenstruktur der Irradiance Map ist für einen optimierten Raytracing-Algorithmus sehr gut geeignet, da die Szenen durch Dreiecke gegeben sind und damit eine natürliche Speichermethode für ein schnelles Auslesen der Irradianz-Werte Anwendung findet [22]. Light Leaks werden aufgrund der Speicherung auf der Oberfläche verhindert. Im Gegensatz zum Irradiance Caching lassen sich auch direkte Irradianzen annähern. Zu berücksichtigen bleibt jedoch die Entstehung neuer Artefakte aufgrund möglicher fehlerhafter Geometrien und Aliasing-Effekten.

Die Generierung der Irradiance Map funktioniert sowohl für direkte als auch indirekte Irradianzen. Allerdings ist hierbei eine Anpassung der Parameter notwendig. Bessere Ergebnisse werden im Allgemeinen für indirekte Irradianzen erzielt. In der Mehrzahl der Fälle wird das Rauschen vollständig eliminiert, sodass die entstehenden Bilder im Vergleich zu den Referenzbildern realistischer wirken.

Aufgrund der Tatsache, dass die Geometrie einer Szene im Allgemeinen nicht konvex ist, ist man bei der Berechnung der Irradiance Map dazu gezwungen die Messwerte innerhalb eines Dreiecks aufzunehmen. Diese Einschränkung verhindert die Benutzung von Vertex Lighting als Basis und führt zu einem höheren Rechenzeit. Die geringe Sampleanzahl, die man pro Dreieck verwendet, kann die Irradianzfunktion auf dem Dreieck nicht ausreichend interpolieren, sodass bei einzelnen Dreiecken der Schattenverlauf gar nicht oder falsch reproduziert wird.

Schlussfolgernd bleibt festzuhalten, dass das Verfahren zur Generierung der Irradiance Map zwar eine Möglichkeit der simultanen Berechnung der Strahldichte darstellt, aber aufgrund der großen Berechnungszeit, des hohen Speicheraufwandes, der unzureichenden Interpolation und den daraus resultierenden Beleuchtungsartefakten nur begrenzt effizient in der Praxis anwendbar ist. Dennoch sind wir durch geeignete Algorithmen in der Lage die Funktionsweise des Irradiance Map Generators zu verbessern. In den Quellen ... sind diverse Varianzreduktionsmethoden erläutert, die das Messen der Irradianz unter Anwendung der hier eingeführten Fehlermetrik durch eine geringere Anzahl von Samples ermöglichen. Dies würde zu einer erheblichen Beschleunigung des Verfahrens beitragen. In Quelle [31] wird das Verfahren des sogenannten »Irradiance Caching« beschrieben. Hierbei wird jedem Samplepunkt durch eine Abstandsmetrik eine maximale Interpolationsdistanz zugeordnet. Die Verwendung einer solchen Größe in der Irradiance Map kann zur Verringerung des Speicherverbrauches führen. Fortführend ist in [?] die Methode der »Irradiance Gradients« eingeführt worden, die ein alternatives Interpolationsschema zwischen aufgenommenen Samplepunkten der Irradianz auf Basis der restlichen Szenengeometrie erklären. Eine Abwandlung des Verfahrens für die Irradiance Map könnte in der Lage sein, die Interpolationsartefakte ohne größeren Rechenaufwand zu minimieren und damit auch zu einer Beschleunigung des gesamten Verfahrens beitragen.

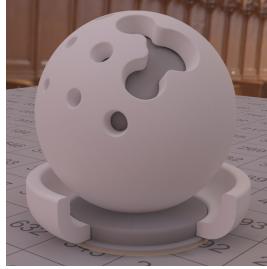
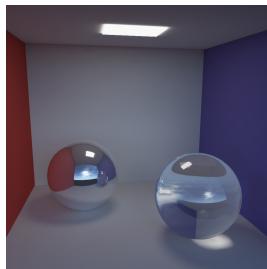
Literatur

- [1] Akenine-Möller, Tomas, Eric Haines, and Naty Hoffman: *Real-Time Rendering*. A K Peters, Ltd., third edition, 2008.
- [2] Blochi: *Basketball Court, sIBL Archive*, Aug 2009. <http://hdrlabs.com/sibl/archive.html>.
- [3] Botsch, Mario, Mark Pauly, and C Rossli: *Geometric modeling based on triangle meshes*. Acm Siggraph 2006 ..., pages 1–148, 2006. <http://dl.acm.org/citation.cfm?id=1185839>.
- [4] Bronstein, Semendjajew, Musiol und Mühlig: *Taschenbuch der Mathematik*. Europa-Lehrmittel, zehnte Auflage, 2016, ISBN 978-3-8085-5789-1.
- [5] Cohen, Michael F. and John R. Wallace: *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1995, ISBN 0-12-059756-X.
- [6] Cohen-Or, Daniel, Yiorgos L. Chrysanthou, Cláudio T. Silva, and Frédo Durand: *A survey of visibility for walkthrough applications*. IEEE Transactions on Visualization and Computer Graphics, 9(3):412–431, 2003, ISSN 10772626.
- [7] Creative Technologies, USC University of Southern California Institute for: *High-Resolution Light Probe Image Gallery*, Jun 2017. <http://vgl.ict.usc.edu/Data/HighResProbes/>.
- [8] Durand, F: *3D Visibility: analytical study and applications*. Université Joseph Fourier, Grenoble, France, 1999. [http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:3D+Visibility++analytical+study+and+applications](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:3D+Visibility++Analytical+Study+and+Applications).
- [9] Elstrodt, Jürgen: *Maß- und Integrationstheorie*. Springer, siebte korrigierte und aktualisierte Auflage, 2011, ISBN 978-3-642-17904-4.
- [10] Gautron, Pascal, Jaroslav Kriva, Sumanta Pattanaik, and Kadi Bouatouch: *Radiance Caching for Efficient Global Illumination Computation*. 11(5):1–11, 2005.
- [11] Grant, Barbara G.: *Field Guide to Radiometry*. SPIE, 2011, ISBN 978-0-8194-8827-5.
- [12] Jensen, Henrik Wann: *A practical guide to global illumination using photon maps*. SIGGRAPH 2000 Course Notes CD-, 38:20–es, 2000. <http://portal.acm.org/citation.cfm?doid=1103900.1103920>.
- [13] Kajiya, James T: *The rendering equation*. In *ACM Siggraph Computer Graphics*, volume 20, pages 143–150. ACM, 1986.
- [14] Kusolitsch, Norbert: *Maß- und Wahrscheinlichkeitstheorie: Eine Einführung*. SpringerWienNew York, 2011, ISBN 978-3-7091-0684-6.
- [15] Kühnel, Wolfgang: *Differentialgeometrie: Kurven - Flächen - Mannigfaltigkeiten*. Springer Spektrum, sechste Auflage, 2013, ISBN 978-3-658-00614-3.
- [16] Laboratory, Stanford University Computer Graphics: *The Stanford 3D Scanning Repository*, Jun 2017. <https://graphics.stanford.edu/data/3Dscanrep/>.

- [17] Lafortune, Eric P and Yves D Willems: *Bi-Directional Path Tracing*.
- [18] LaMothe, André: *Tricks of the 3D Game Programming Gurus: Advanced 3D Graphics and Rasterization*. Sams Publishing, 2003.
- [19] Levoy, Marc and Pat Hanrahan: *Light field rendering*. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96, pages 31–42, 1996, ISSN 00978930. <http://portal.acm.org/citation.cfm?doid=237170.237199>.
- [20] Malacara, Daniel: *Color Vision and Colorimetry: Theory and Applications*. SPIE, 2011, ISBN 978-0-8194-8397-3.
- [21] McGuire, Morgan: *Computer Graphics Archive*, August 2011. <http://graphics.cs.williams.edu/data>.
- [22] Mm Oller, Tomas and Ben Trumbore: *Fast, Minimum Storage Ray/Triangle Intersection*. ACM SIGGRAPH 2005 Courses, (1):1–7, 2005, ISSN 1086-7651.
- [23] Morvan, Jean Marie and Boris Thibert: *Smooth surface and triangular mesh: comparison of the area, the normals and the unfolding*. SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications, pages 147–158, 2002.
- [24] Müller-Gronbach, Thomas, Erich Novak und Klaus Ritter: *Monte Carlo-Algorithmen*. Springer, 2012, ISBN 978-3-540-89140-6.
- [25] Nate just: *20 HDRI Images*, DeviantArt, Jun 2017. <http://just-nate.deviantart.com/art/20-HDRI-Images-109988135>.
- [26] Nicodemus, Fe, Jc Richmond, and Jj Hsia: *Geometrical considerations and nomenclature for reflectance*. Science And Technology, 60(October):1–52, 1977, ISSN 10411135. <http://graphics.stanford.edu/courses/cs448-05-winter/papers/nicodemus-brdf-nist.pdf>.
- [27] Nikodym, Tomas: *Ray Tracing Algorithm For Interactive Applications Tomas Nikodym*. page 76, 2010.
- [28] Nolting, Wolfgang: *Grundkurs Theoretische Physik 3: Elektrodynamik*. Springer, achte Auflage, 2007, ISBN 978-3-540-71251-0.
- [29] Ohta, Noboru and Alan Robertson: *Colorimetry: Fundamentals and Applications*. Wiley, 2005, ISBN 978-0-470-09472-3.
- [30] Parker, Steven, William Martin, Peter Pike J Sloan, Peter Shirley, Brian Smits, and Charles Hansen: *Interactive ray tracing*. Proceedings of the 1999 symposium on Interactive 3D graphics SI3D 99, pages:119–126, 1999. <http://portal.acm.org/citation.cfm?doid=300523.300537>.
- [31] Pharr, M. and G. Humphreys: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, second edition, 2010.
- [32] Pharr, M., W. Jakob, and G. Humphreys: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, third edition, 2017.
- [33] Pipkin, A. C.: *A Course on Integral Equations*. Springer Science+Business Media, LLC, 2013, ISBN 978-1-4612-8773-5.

- [34] Scherzer, Daniel, Chuong H. Nguyen, Tobias Ritschel, and Hans Peter Seidel: *Pre-convolved Radiance Caching*. Computer Graphics Forum, 31(4):1391–1397, 2012, ISSN 01677055.
- [35] Shirley, P., B. Wade, P. M. Hubbard, D. Zareski, B. Walter, and D. P. Greenberg: *Global Illumination via Density-Estimation*. Proceedings of 6th Workshop on Rendering, pages 219–230, 1995. http://link.springer.com/chapter/10.1007/978-3-7091-9430-0{_\}21{\%}5Cnhttp://www.cs.utah.edu/{~}shirley/papers/de95.pdf.
- [36] Veach, Eric: *Robust Monte Carlo Methods for Light Transport Simulation*. Dissertation at the Department of Computer Science of Stanford University, 134(December):759–764, 1997, ISSN 13509462. http://graphics.stanford.edu/papers/veach{_\}thesis/.
- [37] Veach, Eric and Leonidas J. Guibas: *Metropolis light transport*. Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97, pages 65–76, 1997, ISSN 00978930. <http://portal.acm.org/citation.cfm?doid=258734.258775>.
- [38] Wald, Ingo: *Fairy Forest, The Utah 3D Animation Repository*, Jun 2017. <http://www.sci.utah.edu/~wald/animrep/>.
- [39] Walter, Bruce, Philip M Hubbard und Peter Shirley: *Global Illumination Using Local Linear Density Estimation*.
- [40] Ward, Gregory J. und Paul S Heckbert: *Irradiance gradients*. ACM SIGGRAPH 2008 classes on - SIGGRAPH '08, Seite 1, 2008. <http://portal.acm.org/citation.cfm?doid=1401132.1401225>.
- [41] Ward, Gregory J., Francis M. Rubinstein, and Robert D. Clear: *A ray tracing solution for diffuse interreflection*. ACM SIGGRAPH Computer Graphics, 22(4):85–92, 1988, ISSN 00978930. <http://portal.acm.org/citation.cfm?doid=378456.378490>.
- [42] Wolfe, William L.: *Introduction to Radiometry*. SPIE, 1998, ISBN 0-8194-2758-6.
- [43] Yuksel, Cem, John Keyser und Donald H. House: *Mesh colors*. ACM Transactions on Graphics, 29(2):1–11, 2010, ISSN 07300301.

A Verwendete Szenen

Bild	Daten
	<p>»Dragon«-Szene:</p> <p>871306 Dreiecke 439704 Eckpunkte</p> <p>Quelle: [16, 21]</p>
	<p>»Shaderball«-Szene:</p> <p>119250 Dreiecke 61258 Eckpunkte</p> <p>Quelle: vom <i>Fraunhofer ITWM</i> zur Verfügung gestellt</p>
	<p>»Audi R8«-Szene:</p> <p>1749705 Dreiecke 1601106 Eckpunkte</p> <p>Quelle: vom <i>Fraunhofer ITWM</i> zur Verfügung gestellt</p>
	<p>»Cornell Box«-Szene:</p> <p>2188 Dreiecke 1281 Eckpunkte</p> <p>Quelle: [21]</p>
	<p>»Fairy«-Szene:</p> <p>174117 Dreiecke 103523 Eckpunkte</p> <p>Quelle: [38]</p>

B Verwendete HDR Maps

Bild	Daten
	<p>»Uffizi Gallery, Italy«: Quelle: [7]</p>
	<p>»Grace Cathedral, San Francisco, California«: Quelle: [7]</p>
	<p>»Ennis-Brown House, Los Angeles, California«: Quelle: [7]</p>
	<p>»Basketball Court«: Quelle: [2]</p>
	<p>»Sky 10«: Quelle: [25]</p>
	<p>»Sky 20«: Quelle: [25]</p>

Eigenständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel und Quellen angefertigt habe. Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder einer anderen Sprache als Veröffentlichung erschienen.

Seitens des Verfassers bestehen keine Einwände, die vorliegende Bachelorarbeit für die öffentliche Benutzung zur Verfügung zu stellen.

Jena, 16.Juni 2017

Markus Pawellek