

Master-Arbeit

Rendering-gestütztes Deep
Learning zur Rekonstruktion
physikalisch-konformer Szenen

Abdullah Haris Yalcinkaya

11. November 2019

Gutachter:

Dr. Frank Weichert

Prof. Dr. Heinrich Müller

Lehrstuhl Informatik VII
Graphische Systeme
TU Dortmund

Inhaltsverzeichnis

Mathematische Notation	1
1 Einleitung	3
1.1 Motivation	3
1.2 Gegenstand der Arbeit	4
1.3 Übersicht über die Arbeit	5
2 Generierung physikalisch-konformer Szenen	7
2.1 Rendering	7
2.1.1 Licht, Farbe und radiometrische Größen	8
2.1.2 Bidirektionale Reflexionsdistributionsfunktion	9
2.1.3 Beleuchtungsmodelle	11
2.1.4 Bilderzeugung	16
2.1.5 Repräsentation der beleuchteten Bilder	18
2.1.6 Inverses Rendering	19
2.2 Verwandte Arbeiten	19
2.3 Konzeptionelle Übersicht	22
3 Inverses Rendering für planare Oberflächen	23
3.1 Definition	23
3.2 Lösungsansätze und weiteres Vorgehen	25
3.3 Netzarchitektur und Training	28
3.3.1 Grundarchitektur	28
3.3.2 Globales Merkmalsnetzwerk	32
3.3.3 Datenanpassungen	38
3.3.4 Training	40
3.4 Rendering-gestütztes Fehlermaß	41
3.4.1 Differenzierbare Renderingkomponente	44
3.4.2 Fehlerbestimmung mittels multiplen Renderings	47
3.4.3 Regularisierung	50

3.5	Datenaugmentationen	51
3.5.1	Prozedurale Synthese von Trainingsdaten	51
3.5.2	Selbstaugmentation	52
3.5.3	Online-Datengenerierung	53
3.5.4	Generative Datenaugmentation	54
3.6	Multiple Eingabefotos	55
3.6.1	Parallele U-Nets	56
3.6.2	Autoencoder-gestützte inverse Renderingfunktion	58
3.7	Realisierung	59
3.8	Evaluierung	61
3.8.1	Evaluierungsszenarien	61
3.8.2	Vergleich der U-Net-Varianten	63
3.8.3	SVBRDF-Akquise realer Materialien	68
3.8.4	Quantitative Analyse der naiven Datenaugmentation	71
3.8.5	Qualitativer Vergleich mit der Originalarbeit	74
4	Inverses Rendering für räumliche Szenen	81
4.1	Definition	81
4.2	Lösungsansatz	84
4.3	Netzarchitektur	85
4.4	Differenzierbarer Renderer für räumliche Szenen	88
4.5	Stochastisches Rendering-gestütztes Fehlermaß	89
4.6	Regularisierung der Fehlerfunktion	92
4.7	Realisierung	94
4.8	Evaluierung	96
4.8.1	Evaluierungsszenarien	96
4.8.2	Analyse der inversen Renderingfunktion für räumliche Szenen	98
4.8.3	Analyse der Regularisierungsverfahren	105
4.8.4	Anwendung der räumlichen inversen Renderingfunktion	112
5	Zusammenfassung und Ausblick	115
5.1	Zusammenfassung	115
5.2	Ausblick	116
A	Externes GPU-gestütztes Rendering der SVBRDF-Schätzungen	119
	Abbildungsverzeichnis	122

Algorithmenverzeichnis	123
Literaturverzeichnis	125



Mathematische Notation

Notation	Bedeutung
$\mathbb{N}$	Menge der natürlichen Zahlen $1, 2, 3, \dots$
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^d$	d -dimensionaler Raum
$\mathcal{M} = \{m_1, \dots, m_N\}$	ungeordnete Menge $\mathcal{M}$ von N Elementen m_i
$\mathcal{M} = \langle m_1, \dots, m_N \rangle$	geordnete Menge $\mathcal{M}$ von N Elementen m_i
$\mathbf{v}$	Vektor $\mathbf{v} = (v_1, \dots, v_n)^T$ mit N Elementen v_i
$v_i^{(j)}$	i -tes Element des j -ten Vektors
$\mathbf{u} \cdot \mathbf{v}$	Skalarprodukt der Vektoren $\mathbf{u}$ und $\mathbf{v}$
$ \mathbf{v} $	Betrag des Vektors $\mathbf{v}$
$\mathbf{A}$	Matrix $\mathbf{A}$ mit Einträgen $a_{i,j}$
$\mathbf{J}_{m,n}$	$m \times n$ große Matrix mit allen Einträgen gleich eins
$\mathbf{j}_m$	$m \times 1$ großer Vektor mit allen Elementen gleich eins
$\mathbf{K} ** \mathbf{A}$	Faltung der Matrix $\mathbf{A}$ mit dem Kernel $\mathbf{K}$

1 Einleitung

1.1 Motivation

In einem Artikel spricht Ken Perlin¹ über seine Vision von der computergestützten Augmentation des menschlichen Sehsystems; dabei soll jeder Aspekt unserer sichtbaren Umgebung visuell durch computergrafische Ansätze erweitert werden, einschließlich unseres eigenen Aussehens, der Art, wie wir mit Menschen kommunizieren, der Interaktion mit realen und der Generierung von virtuellen Objekten. Darauf aufbauend steht uns seiner Meinung nach das Zeitalter der Computergrafik erst noch bevor [53].

Die Kombination von virtuellen Inhalten mit realen Umgebungen ermöglicht eine scheinbar unendliche Anzahl an Anwendungen. Diese reichen von medizinischen Applikationen [3], der erweiterten Kommunikation [53], der Navigation [47], über architektonische Visualisierungen [10], Visualisierungen von Innenarchitekturen [10] und Spezialeffekten [10]. So kann beispielsweise im Rahmen einer medizinischen Anwendung durch die Überlagerung von Ultraschalldaten auf Kamerabilder aus der Perspektive des Betrachters in den Patienten „hineingesehen“ werden [3]. In einer anderen Anwendung können durch das Echtzeitrendern von virtuellen Inhalten auf reale Objekte Spezialeffekte in Echtzeit dargestellt und die Realisierung von interaktiven Filmen ermöglicht werden [75].

Für die realitätsgetreue Einführung virtueller Inhalte in die uns umgebende Wirklichkeit wird eine einheitliche Beleuchtung der Objekte und der umliegenden Oberflächen benötigt; zudem muss das Lichtspiel zwischen den Objekten und ihren Umgebungen korrekt simuliert werden [10]. Durch eine computergrafische Repräsentation der Szene, die über die bloße Geometrieerfassung hinausgeht, können synthetische Objekte korrekt in die Szene hinein gerendert werden; diese zusätzlichen Informationen werden beispielsweise durch die Reflexionseigenschaften der Oberflächen dargestellt [10]. Die Rekonstruktion des Reflexionsverhaltens, der sogenannten bidirek-

¹Erfinder des *Academy Award* ausgezeichneten Perlin-Noise-Algorithmus, siehe <https://mrl.nyu.edu/~perlin/doc/oscar.html> – Zugriffsdatum: 14. Oktober 2019

tionalen Reflexionsdistributionsfunktion (BRDF) [21], wird hierbei im Rahmen des inversen Renderings betrachtet [59].

In den vergangenen Jahren konnten mittels der Technologie des Deep Learnings in vielen visuellen Gebieten revolutionäre Erfolge erzielt werden [66]. So wurden Methoden des Deep Learnings eingesetzt, um Atari-Spiele ausgehend von den Pixelwerten des Bildschirms zu meistern; dabei konnte die Stärke von sämtlichen vorherigen Algorithmen überboten und dem professionellen menschlichen Spieler vergleichbare Leistungen erbracht werden [45]. Zudem konnten durch maschinelles Lernen große Erfolge im Bereich des Echtzeit- und Offline-Renderings erzielt werden, beispielsweise durch die Reduktion von Rauschen oder beim Anti-Aliasing [28].

Ein vielversprechender Ansatz wird durch die Kombination von lernbasierten und modellgestützten Ansätzen im visuellen Bereich dargestellt; dabei können physikalische Modelle in Convolutional Neural Networks (CNNs) durch zusätzliche Schichten integriert werden [81]. Eine Reihe an erst kürzlich erschienenen Arbeiten behandelt das Thema der Deep-Learning-gestützten BRDF-Akquise ausgehend von einem [11, 39] oder mehreren Eingabebildern [11, 16]; hierbei wird das physikalische Modell durch eine differenzierbare Renderingkomponente dargestellt, die im Rahmen der Fehlerfunktion für das Training der Deep Neural Networks eingesetzt wird.

1.2 Gegenstand der Arbeit

Der Gegenstand dieser Arbeit besteht darin, die inverse Renderingfunktion – primär dargestellt durch die Akquisition von Reflexionsinformationen – zu untersuchen und weiterzuentwickeln. Motiviert durch die Entwicklungen im Bereich des Deep Learnings, dem Rendering und der Vision einer Computergrafik-gestützten Zukunft wird somit die inverse Renderingfunktion auf räumliche Szenen erweitert. Dazu wird zunächst der Stand der Technik der Deep-Learning-gestützten inversen Renderingfunktion für die BRDF-Akquise von planaren Oberflächen zusammengefasst, repliziert und anschließend untersucht (siehe Abbildung 1.1). Dies bildet im zweiten Schritt die Grundlage für die erstmalige Übertragung dieser Methoden auf räumliche Szenen. Hierbei wird eine Methode vorgestellt, die eine pixelweise Akquise der Reflexionseigenschaften anhand eines Fotos und der dazu korrespondierenden Positionskarte durchführt. Aufbauend auf diesen Resultaten wird eine Reihe von interessanten Anwendungen ermöglicht, die sich beispielsweise durch die Neubeleuchtung, Einführung zusätzlicher Lichtquellen und der künstlerischen Reinszenierung der räumlichen Szene zusammensetzen (siehe Abbildung 1.2). Entsprechend Perlins

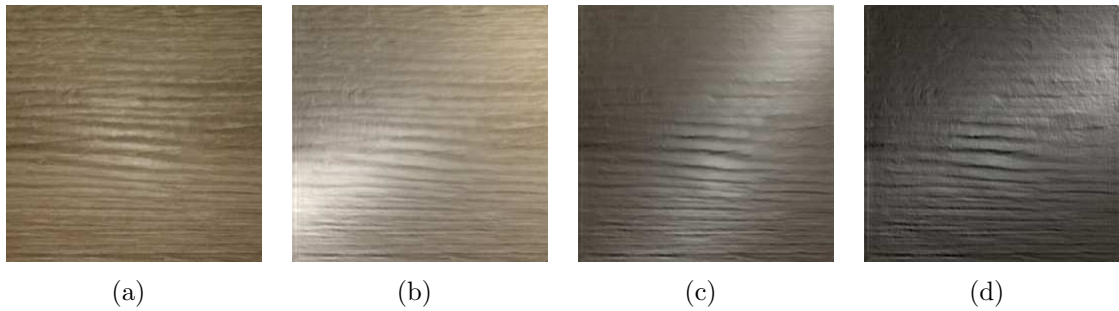


Abbildung 1.1: Neubeleuchtungen einer Holztischplatte. Abbildung (a) stellt die Eingabeprobe dar und Abbildungen (b) bis (d) sind verschiedene Neubeleuchtungen basierend auf den Ausgaben der inversen Renderingfunktion planarer Oberflächen. Die Abbildung stellt Ergebnisse dar, die im Rahmen dieser Arbeit entstanden sind.

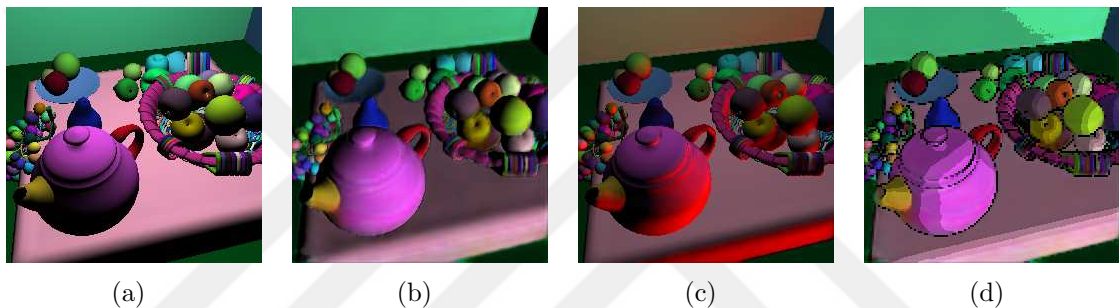


Abbildung 1.2: Anwendungen der räumlichen inversen Renderingfunktion. Abbildung (a) stellt das Eingabebild, (b) die Neubeleuchtung basierend auf den Ausgaben der inversen Renderingfunktion, (c) die Einführung einer zusätzlichen Lichtquelle durch eine Überblendung des Originalbildes und einem Neurendering und (d) eine komplett neuen Reinszinierung der Szene dar. Die Abbildung stellt Ergebnisse dar, die im Rahmen dieser Arbeit entstanden sind.

Vision [53] wird somit ein weiterer Schritt in die Richtung der Fusion virtueller Inhalte mit der realen Welt getätigt.

1.3 Übersicht über die Arbeit

Die für diese Arbeit relevanten Grundlagen werden in Kapitel 2 eingeführt; diese beziehen sich primär auf das Konzept des Renderings. Darauf aufbauend wird eine Definition des inversen Renderings vorgestellt und verwandte Arbeiten dargelegt. Des Weiteren wird das Konzept dieser Arbeit erläutert.

In Kapitel 3 wird die inverse Renderingfunktion für planare Oberflächen definiert und der verfolgte Lösungsansatz vorgestellt; daraufhin wird die Netzarchitektur und

das Training der Deep Neural Networks beschrieben. Anschließend wird das eingesetzte stochastische Rendering-gestützte Fehlermaß dargelegt, welches einen Vergleich der gerenderten Resultate basierend auf den BRDFs anstatt der direkten Repräsentationsformen durchführt. Diverse Datenaugmentationsmethoden und die Verfahren zur Einbindung mehrerer Eingabebilder werden betrachtet. Danach wird die Realisierung der jeweiligen Deep Neural Networks implementierungsbezogen aufgegriffen. Abschließend werden die umgesetzten Methoden in einer ausgiebigen Evaluierung analysiert.

Das Kapitel 4 stellt eine Übertragung der Methoden des inversen Renderings für planare Oberflächen auf das inverse Rendering räumlicher Szenen dar. Hierbei wird zunächst eine Definition der räumlichen inversen Renderingfunktion aufgestellt und eine Relation zur planaren Variante aufgebaut. Daraufhin wird der angepasste Lösungsansatz dargelegt und die Modifikationen an der Netzarchitektur elaboriert. Ein neues stochastisches Rendering-gestütztes Fehlermaß für die Bewertung der geschätzten Reflexionseigenschaften von räumlichen Szenen wird vorgestellt und zwei mögliche Regularisierungen der Fehlerfunktion erläutert; dabei stellt die Filter-gestützte Variante eine anwendungsspezifische Entwicklung im Rahmen dieser Arbeit dar. Auch hier wird die Realisierung der umgesetzten Methoden implementierungsbezogen dargelegt. Abschließend werden die realisierten Ansätze evaluiert.

Die Arbeit findet mit einer Zusammenfassung und einem zukunftsorientierten Ausblick im Rahmen des Kapitels 5 einen Abschluss.

2 Generierung physikalisch-konformer Szenen

In diesem Kapitel werden die relevanten Grundlagen des Renderings, die Definition des inversen Renderings, verwandte Arbeiten und das Konzept dieser Arbeit vorgestellt.

In Abschnitt 2.1 wird das Konzept des Renderings vorgestellt und eine detaillierte Übersicht über physikalische Grundlagen gegeben; darauf aufbauend wird zudem eine einleitende Definition des inversen Renderings eingeführt. Abschnitt 2.2 stellt die zu dieser Arbeit verwandten Arbeiten, Methoden und Lösungsansätze vor. Abschließend wird in Abschnitt 2.3 eine Übersicht über das Konzept und die Methoden dieser Arbeit gegeben.

2.1 Rendering

Das Rendering ist der Prozess der Generierung von Bildern basierend auf einer Beschreibung einer dreidimensionalen Szene. Physikalisch-basierte Ansätze versuchen dabei die Realität durch die Simulation von physikalischen Prinzipien bei der Interaktion von Licht und Materie nachzubilden [54].

Das fotorealistische Rendering ist für viele Anwendungen von großer Relevanz, z.B. für Spezialeffekte, dem Architekturdesign, Computerspielen, dem Automobildesign, Werbungen und der Unterhaltungsindustrie [7, 21].

In den folgenden Unterabschnitten werden Schritt für Schritt die relevanten Grundlagen für das physikalisch-basierte Rendering vorgestellt. Abschnitt 2.1.1 führt eine Definition von Licht, Farbe und den zentralen radiometrischen Größen ein. In Abschnitt 2.1.2 wird die bidirektionale Reflexionsdistributionsfunktion (BRDF) vorgestellt. Verschiedene BRDF-Modelle für die computergrafische Beleuchtung von Szenen werden in Abschnitt 2.1.3 betrachtet. Konzepte und Verfahren für den eigentlichen Beleuchtungsprozess, dem Rendering, werden in Abschnitt 2.1.4 thematisiert. Die Repräsentation der beleuchteten Ergebnisse wird in Abschnitt 2.1.5 dargelegt.

Abschließend wird in Abschnitt 2.1.6 eine kurze, einleitende Definition des inversen Renderings eingeführt; die Approximation der inversen Renderingfunktion stellt dabei das Ziel dieser Arbeit dar.

2.1.1 Licht, Farbe und radiometrische Größen

In diesem Teilabschnitt werden relevante Begriffe wie die des Lichts, der Farbe und die diesbezüglich relevanten radiometrischen Größen vorgestellt. Hierbei bilden die radiometrischen Größen eine physikalische Grundlage für spätere Bilderzeugungsgleichungen.

Licht ist Teil des elektromagnetischen Spektrums. Der für den Menschen sichtbare Teil liegt bei ca. 380 bis 700 Nanometern. Lichtempfindliche Elemente im Auge des Menschen, dargestellt durch die sogenannten Zapfen und Stäbchen, sind dabei empfänglich für diesen Bereich des elektromagnetischen Spektrums [71]. Das Phänomen der Farbe kann dabei sowohl als reines Signal aus dem elektromagnetischen Spektrum oder in vielen Fällen auch mittels additiver Überlagerung von Strahlen mehrerer Wellenlängen entstehen [70].

Relevante Größen bei der Erfassung und Wiedergabe von elektromagnetischen Strahlen werden durch radiometrische Größen festgehalten (siehe Tabelle 2.1). Lichtquellen emittieren Photonen, die eine gewisse Menge an Energie besitzen; diese Energie wird in Joule gemessen. Die Energie misst allerdings Arbeit, die über Zeit verrichtet wird, wobei sich in der Computergrafik der Fokus der Messungen bezüglich des Lichts oft auf einen bestimmten Zeitpunkt bezieht. Der *Strahlungsfluss* Φ ist die gesamte Menge an Energie, die sich durch eine Fläche oder einen Raum pro Zeiteinheit bewegt; diese Größe wird in Watt gemessen [55].

Radiometrie		Fotometrie	
Begriff	Einheit	Begriff	Einheit
Strahlungsfluss Φ	Watt (W)	Lichtstrom Φ	Lumen (lm)
Spezif. Ausstrahlung	$\text{W} \cdot \text{m}^{-2}$	Spezif. Lichtausstrahl.	$\text{lm} \cdot \text{m}^{-2}$
Strahlungsstärke I	$\text{W} \cdot \text{sr}^{-1}$	Lichtstärke I	Candela (cd)
Bestrahlungsstärke E	$\text{W} \cdot \text{m}^{-2}$	Beleuchtungsichte E	Lux (lx)
Strahlungsdichte L	$\text{W} \cdot \text{m}^{-2} \cdot \text{sr}^{-1}$	Leuchtdichte L	$\text{cd} \cdot \text{m}^{-2}$

Tabelle 2.1: Radiometrische und fotometrische Größen mit ihren Einheiten. Die linke Spalte stellt die radiometrischen Größen und die rechte Spalte die dazu analogen fotometrischen Größen samt ihren Einheiten dar [44, 71]. Die Einheit Lux (lx) entspricht dabei $\text{lm} \cdot \text{m}^{-2}$ und die Einheit Candela (cd) $\text{lm} \cdot \text{sr}^{-1}$ [44].

Der Strahlungsfluss benötigt eine Fläche A , über die Photonen über die Zeit gemessen werden. Durch das Teilen des Strahlungsflusses Φ durch die Fläche A wird entweder die *spezifische Ausstrahlung*, welche die Flächendichte des ausgehenden Strahlungsflusses angibt, oder die *Bestrahlungsstärke* E , welche die Flächendichte des eingehenden Strahlungsflusses angibt, erhalten; beide Größen werden in Watt pro Quadratmeter angegeben [55].

Die *Strahlungsstärke* I gibt die Dichte über einen Raumwinkel des Strahlungsflusses an und wird in Watt pro Steradian gemessen. Der Steradian gibt hierbei den Raumwinkel an; dieser wird analog zum Radianten des Einheitskreises definiert, wobei hier eine Einheitskugel betrachtet wird. So gibt $2\pi sr$ den Raumwinkel einer Halbkugel und $4\pi sr$ den der gesamten Einheitskugel an [55].

Die für die Computergrafik wichtigste radiometrische Größe wird durch die *Strahlungsdichte* L dargestellt. Die spezifische Ausstrahlung bzw. die Bestrahlungsstärke E gibt die Differenzialleistung pro Differenzialfläche an; bei dieser Größe wird jedoch die richtungsabhängige Verteilung der Leistung nicht miteinbezogen. So gibt die Strahlungsdichte L die spezifische Ausstrahlung bzw. die Bestrahlungsstärke mit Bezug auf den Raumwinkel an; somit ist die Strahlungsdichte der Strahlungsfluss pro Flächenenheit pro Raumwinkel und wird folglich in der Einheit Watt pro Quadratmeter pro Steradian angegeben [55].

Analog zu den radiometrischen Größen gibt es die fotometrischen Größen, welche essentiell dieselben Begebenheiten erfassen (siehe Tabelle 2.1); während radiometrische Einheiten für die Messung von elektromagnetischen Strahlungen genutzt werden, finden fotometrische Größen ihren Einsatz durch ihre relative Gewichtung anhand der spektralen Antwort des menschlichen Auges [71]; sie sind somit auf das menschliche Sehsystem ausgelegt [55]. Im Rahmen dieser Arbeit wird primär Bezug auf die radiometrischen Größen genommen.

Im nächsten Abschnitt wird die bidirektionale Reflexionsdistributionsfunktion vorgestellt, die eine radiometrische Funktion darstellt und angibt, wie eingehende Energie in alle Richtungen der oberen Hemisphäre einer Oberfläche verteilt wird [21].

2.1.2 Bidirektionale Reflexionsdistributionsfunktion

Die bidirektionale Reflexionsdistributionsfunktion (BRDF) ist eine vierdimensionale radiometrische Verteilungsfunktion und gibt die Verteilung von eingehender Energie in ausgehende Richtungen in einer Hemisphäre oberhalb einer Oberfläche an; sie

wird in inversen Steradians gemessen [21, 49]. Die BRDF ist gegeben durch

$$f_r(\mathbf{v}_i, \mathbf{v}_r) = \frac{dL_r(\mathbf{v}_r)}{dE_i(\mathbf{v}_i)}, \quad (2.1)$$

wobei $\mathbf{v}_i$ die Eingangsrichtung und $\mathbf{v}_r$ die Ausgangsrichtung beschreibt; sie stellt das Verhältnis der eingehenden Bestrahlungsstärke E_i zur ausgehenden Strahlungsdichte L_r dar [21].

Eine valide BRDF muss dabei die Eigenschaften

- der Nicht-Negativität, formalisiert durch $f_r(\mathbf{v}_i, \mathbf{v}_r) \geq 0$,
- der Helmholtz-Reziprozität, formalisiert durch $f_r(\mathbf{v}_i, \mathbf{v}_r) = f_r(\mathbf{v}_r, \mathbf{v}_i)$, und die
- der Energieerhaltung

erfüllen. Dabei ist die Energieerhaltung gegeben durch

$$\int_{\Omega_+} f_r(\mathbf{v}_i, \mathbf{v}_r)(\mathbf{v}_r \cdot \mathbf{n}) d\mathbf{v}_r \leq 1, \quad (2.2)$$

wobei Ω_+ die Einheitshemispäre oberhalb der Oberfläche angibt und die Ungleichung für jedes $\mathbf{v}_i$ gelten muss [21].

Da bei der BRDF die Annahme gemacht wird, dass an einem Punkt einer Oberfläche auftreffendes Licht diese ebenfalls an demselben Punkt verlässt, schlägt die Approximation der BRDF für transluzente Materialien fehl. Des Weiteren können Phänomene wie die der Volumenstreuung, welche im Rahmen der generelleren bidirektionale Oberflächenstreuungsverteilungsfunktion (BSSRDF) thematisiert werden, nicht durch die BRDF rekonstruiert werden [24]. Diese Phänomene stellen im weiteren Verlauf der Arbeit keine Relevanz dar.

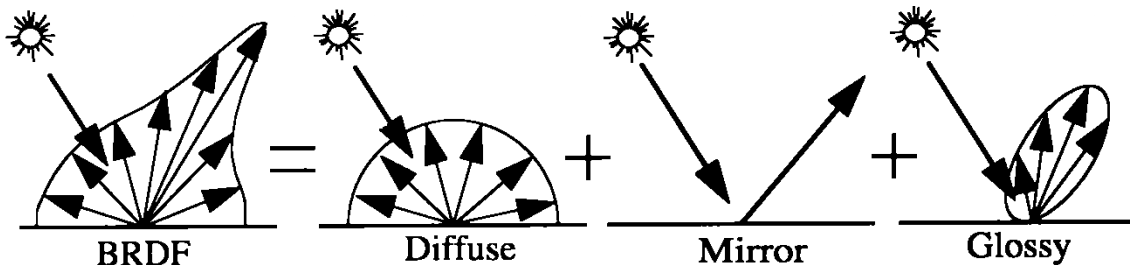


Abbildung 2.1: Komponenten der Reflexion [7]. Die BRDF kann in der Praxis als eine Summe von Reflexionskomponenten dargestellt werden; diese beziehen sich auf die diffuse, perfekt gespiegelte und Glanzreflexionskomponente [7].

In der Anwendung wird die BRDF häufig als eine Summe der diffusen Reflexion, der Spiegelreflexion und der Glanzreflexion dargestellt (siehe Abbildung 2.1). Bei der diffusen Reflexion wird angenommen, dass das Licht in alle Richtungen gleichmäßig reflektiert wird. Bei der perfekten Spiegelreflexion entspricht der Eingangswinkel gleich dem Ausgangswinkel bezüglich des Normalenvektors der Oberfläche. Reale Materialien sind jedoch weder perfekt diffus, noch stellen sie einen perfekten Spiegel dar; stattdessen gibt es eine Art von Reflexion zwischen den beiden Phänomenen, die als Glanzreflexion bezeichnet wird [7].

Während die BRDF für uniforme Materialien genutzt wird, ist die räumlich variierende BRDF (SVBRDF) zusätzlich abhängig von einem gegebenen Punkt auf der Oberfläche und dient somit der Repräsentation nicht-homogener Materialien; die SVBRDF erfordert somit zwei weitere Parameter, welche die Position des Punktes auf der Oberfläche angeben [21].

Verschiedene BRDF-Modelle wurden über die Jahre vorgeschlagen [21]; von diesen werden im nächsten Abschnitt das Phong- [58], das Blinn-Phong- [5] und das Cook-Torrance-Beleuchtungsmodell [8] näher erläutert, da sie im weiteren Verlauf der Arbeit Einsatz im Rahmen verschiedener Renderingkomponenten finden.

2.1.3 Beleuchtungsmodelle

Verschiedene Modelle basierend auf unterschiedlichen Ansätzen wurden für die Repräsentation der BRDF vorgeschlagen; diese lassen sich grundsätzlich in phänomenologische, physikalisch-basierte und datenbasierte Modelle unterteilen [21].

Phänomenologische Modelle werden durch analytische Formeln dargestellt, welche auf Reflexionsdaten angepasst sind und dadurch die Reflektanz realer Materialien approximieren [21].

Physikalisch-basierte Modelle basieren auf physikalischen Gegebenheiten und dem Forschungsfeld der Optik. Hierbei wird die Annahme gemacht, dass die Oberfläche eine Rauheit auf einem feinen Level aufweist, die durch Mikrofacetten beschrieben werden kann [21].

Datenbasierte BRDF-Modelle nutzen gemessene BRDF-Daten, die beispielsweise in einer Tabelle abgespeichert werden können. Mittels Interpolationsverfahren können somit große Look-Up-Tabellen produziert werden, welche die notwendigen Daten beinhalten. Diese Methode ist einerseits simpel, andererseits ineffizient bezüglich des verbrauchten Speichers. Des Weiteren sind die gemessenen Rohdaten oft mit einem Rauschen behaftet, weswegen die gerenderten Materialien ebenfalls oft ein Rauschen aufweisen. Eine gemessene BRDF kann durch ein analytisches Modell

interpoliert und somit rekonstruiert werden; hierbei wird auch der verbrauchte Speicher stark reduziert. Der Nachteil dieser Strategie ist mit der Rigidität von vielen Modellen verbunden, die nur schwer bearbeitet und angepasst werden können; oft kann so nur eine limitierte Klasse an Materialien repräsentiert werden. Eine weitere Möglichkeit besteht darin, gemessene Daten mittels passender Funktionsräume, wie sie beispielsweise durch *Spherical Harmonics* oder *Wavelets* dargestellt werden, zu approximieren [21].

Des Weiteren lassen sich BRDFs in isotrope und anisotrope Modelle einteilen. Die Reflexionen in isotropen BRDFs sind unabhängig von der Orientierung der Oberfläche um die Normale $\mathbf{n}$ herum. Dies stellt einen Gegensatz zu anisotropen BRDFs dar [21]. In dieser Arbeit werden ausschließlich isotrope BRDFs betrachtet, wobei einige verwandte Arbeiten (siehe Abschnitt 2.2) auch das Phänomen der Anisotropie mit einbeziehen, beispielsweise mittels des BRDF-Modells von Ward [77].

Für diese Arbeit stellt im Rahmen der phänomenologischen Modelle das Phong-Modell [58] und dessen Weiterentwicklung, das Blinn-Phong-Modell [5], Relevanz dar; diese Modelle werden im Anschluss erläutert und finden Anwendung in Kapitel 4. Zudem kommt auch das den physikalisch-basierten Modellen zuzuordnende Cook-Torrance-Modell zum Einsatz [8]; dieses wird ebenfalls im Folgenden näher betrachtet und repräsentiert das im Kapitel 3 eingesetzte Beleuchtungsmodell.

2.1.3.1 Phong- und Blinn-Phong-Modell

In diesem Abschnitt wird zunächst das Phong-Modell [58] und darauf aufbauend das Blinn-Phong-Modell [5] vorgestellt.

Das Phong-Modell berechnet die reflektierte Strahlungsdichte L an einem Punkt $\mathbf{x}$ der Oberfläche in Abhängigkeit der Lichtquellenrichtung $\mathbf{l}$, Augenpunkttrichtung $\mathbf{v}$ und der Normalen $\mathbf{n}$ der Oberfläche (siehe Abbildung 2.2(a)); hierbei werden alle Richtungen durch Einheitsvektoren dargestellt [72].

Das reflektierte Licht berechnet sich als eine lineare Kombination der ambienten, diffusen und Glanzlichtkomponenten. Die ambiente Komponente ist eine Konstante, die schwaches, in der Szene gestreutes Licht repräsentiert. Die diffuse Komponente wird durch das Lambert-Modell dargestellt; sie repräsentiert das Licht, welches von der Oberfläche uniform in alle Richtungen reflektiert wird. Im Gegensatz zu der diffusen Komponente beschreibt die Glanzlichtkomponente reflektiertes Licht, welches sich oft in einem geringen Winkel im Raum konzentriert und zusätzlich von der Blickrichtung $\mathbf{v}$ abhängt [72].

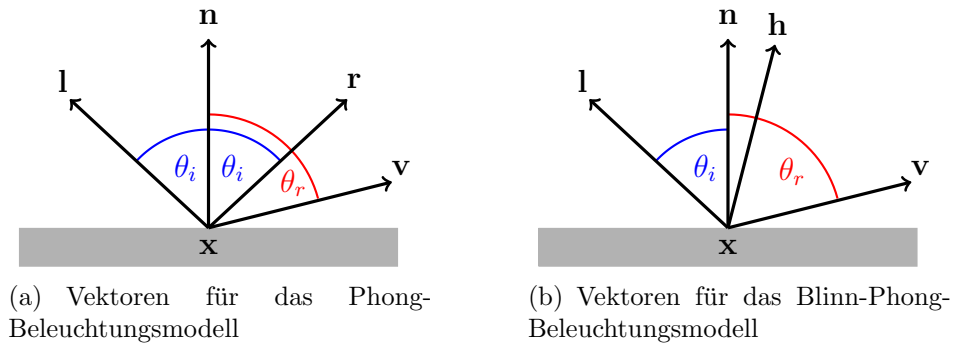


Abbildung 2.2: Relevante Vektoren für die Beleuchtungsberechnung. Abbildung (a) stellt die relevanten Vektoren für das Phong-Modell [58], Abbildung (b) die Vektoren für das Blinn-Phong-Modell dar. Der Vektor $\mathbf{n}$ steht für die Normale der Oberfläche an dem Punkt $\mathbf{x}$, $\mathbf{v}$ für die Richtung zum Augenpunkt, $\mathbf{l}$ für die Richtung zur Lichtquelle, $\mathbf{r}$ für den an der Normale $\mathbf{n}$ gespiegelten Lichtquellenrichtungsvektor $\mathbf{l}$ und $\mathbf{h}$ für den Halbvektor. Die Winkel θ_i und θ_r geben den Eingangs- und Ausgangswinkel in dieser Reihenfolge an [72].

Die reflektierte Strahlungsdichte berechnet sich somit auf

$$I = k_a I_a + k_d I_d \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I_s (\max(0, \mathbf{v} \cdot \mathbf{r}))^\gamma, \quad (2.3)$$

wobei $\mathbf{r}$ sich auf den an der Normale $\mathbf{n}$ reflektierten Lichtquellenrichtungsvektor $\mathbf{l}$ bezieht. Die Parameter I_a , I_d und I_s geben die Leuchtkraft der ambienten, diffusen und Glanzlichtkomponenten in dieser Reihenfolge an. Die skalaren Werte k_a , k_d und k_s bestimmen die relative Stärke dieser drei Komponenten [72]; sie geben die Reflexionskoeffizienten des Materials an [58]. Der Parameter γ gibt die Reflexionscharakteristik des Glanzlichtes an [58].

Eine Weiterentwicklung des Phong-Modells wird durch das Blinn-Phong-Modell [5] dargestellt. In diesem Modell wird der Term $\mathbf{v} \cdot \mathbf{r}$ des Glanzteils durch den Term $\mathbf{n} \cdot \mathbf{h}$ ersetzt (siehe Abbildung 2.2(b)), wobei $\mathbf{h}$ durch

$$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{\|\mathbf{l} + \mathbf{v}\|_2} \quad (2.4)$$

gegeben ist [72]. Dies reduziert die Laufzeitkosten für das Beleuchtungsmodell, da der Reflexionsvektor $\mathbf{r}$ nicht länger berechnet werden muss [21].

Obwohl das Blinn-Phong-Modell aus Performanzgründen entwickelt wurde, hat sich in einer späteren experimentellen Auswertung [48] gezeigt, dass das Blinn-Phong-Modell Resultate liefert, die reale Daten genauer repräsentieren als das originale

Phong-Modell [72]. Dies wird damit begründet, dass die Formulierung von $\mathbf{n} \cdot \mathbf{h}$ natürlich zu dem Mikrofacettenmodell korrespondiert; das Mikrofacettenmodell wird im nächsten Abschnitt näher erläutert. Modelle, die den Term $\mathbf{v} \cdot \mathbf{r}$ nutzen, produzieren inakkurate Ergebnisse in den gerenderten Szenen [48].

Sowohl das Phong-, als auch das Blinn-Phong-Modell erfüllen die Eigenschaften der Helmholtz-Reziprozität als auch der Energieerhaltung nicht. Die Modelle sind den phänomenologischen, isotropen BRDF-Modellen zuzuordnen [21]. Im folgenden Abschnitt 2.1.3.2 wird das Cook-Torrance-Modell vorgestellt.

2.1.3.2 Cook-Torrance-Modell

In diesem Abschnitt wird das Cook-Torrance-Beleuchtungsmodell [8] eingeführt, welches ein analytisches Modell beschreibt, das die wellenlängenabhängige Reflexionseigenschaft einer Oberfläche angibt [18].

Das Cook-Torrance-Modell ist den isotropen, physikalisch-basierten BRDF-Modellen zuzuordnen; es basiert auf dem Mikrofacettenmodell und stellt eine Erweiterung des Torrance-Sparrow-Modells [74] dar, welches dem Anwendungsfeld der Optik entstammt [18]. Das Torrance-Sparrow-Modell nimmt dabei an, dass die Oberfläche aus kleinen, zufällig angeordneten, spiegelähnlichen Facetten besteht [74].

Für eine gegebene Lichtquelle, eine Oberfläche und einen Augenpunkt beschreibt das Reflexionsmodell die Intensität und spektrale Komposition des zum Betrachter reflektierten Lichts; von dieser spektralen Energieverteilung aus können RGB-Farbwerte abgeleitet werden [8].

Das Beleuchtungsmodell ist in Glanzlicht-, diffuse und ambiente Komponenten aufgeteilt. Die Glanzlichtkomponente repräsentiert Licht, welches von der Oberfläche des Materials reflektiert wird. Die diffuse Komponente wird durch interne Streuungen, bei der einfallendes Licht die Oberfläche durchdringt und durch multiple Reflexionen auf der rauen Oberfläche dargestellt. Die Glanzlicht- und diffuse Komponente können verschiedenen Farben besitzen, sofern es sich bei der Oberfläche um kein homogenes Material handelt. Die BRDF ergibt sich somit als

$$R = sR_s + dR_d \quad (2.5)$$

mit $s + d = 1$, wobei s den Glanzlicht- und d den diffusen Reflexionsanteil angibt [8].

Zusätzlich zu den direkten Beleuchtungen durch individuelle Lichtquellen kann ein Objekt durch ambientes Licht beleuchtet werden. Die Beleuchtungsformel des Cook-

Torrance-Modells ergibt sich somit aus der Summe der reflektierten Intensität der ambienten Beleuchtung und den reflektierten Intensitäten aller Lichtquellen und beläuft sich auf

$$I_r = I_{ia}R_a + \sum_l I_{il}(\mathbf{n} \cdot \mathbf{l}_l) d\omega_{il}(sR_s + dR_d), \quad (2.6)$$

wobei I_{ia} die Intensität der ambienten Komponente und R_a den ambienten Reflexionsgrad angibt; zusätzlich wird die reflektierte Intensität über alle Lichtquellen aufsummiert, wobei I_{il} die Intensität der l -ten Lichtquelle, $d\omega_{il}$ den Raumwinkel der jeweiligen Lichtquelle, $\mathbf{l}_l$ den Einheitsvektor in Richtung der jeweiligen Lichtquelle und $\mathbf{n}$ den Normalenvektor der Oberfläche an dem zu beleuchtenden Punkt angibt (siehe Abbildung 2.2(b)) [8].

Im Rahmen dieser Arbeit wird die Glanzlichtkomponente durch

$$R_s = \frac{D(r; \mathbf{n}, \mathbf{h}) \cdot G(r; \mathbf{n}, \mathbf{l}, \mathbf{v}) \cdot F(s; \mathbf{h}, \mathbf{v})}{4(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})} \quad (2.7)$$

gegeben [6, 26], wobei G den geometrischen Dämpfungsterm, D die Mikrofacettensteigungsverteilungsfunktion und F den Fresnel-Term angibt [8].

Der geometrische Dämpfungsterm wird hier durch

$$G(r; \mathbf{n}, \mathbf{l}, \mathbf{v}) = \frac{1}{(\mathbf{n} \cdot \mathbf{l})(1 - k) + k} \cdot \frac{1}{(\mathbf{n} \cdot \mathbf{v})(1 - k) + k} \quad (2.8)$$

repräsentiert [6, 26], wobei der Wert k für die Hälfte der quadrierten Rauheit steht; die Oberflächenrauheit ist dabei ein skalarer Parameter und wird hier durch r angegeben. Der geometrische Dämpfungsterm gibt die Verdeckung des reflektierten Lichts und die Schattierung des ankommenden Lichts durch benachbarte Facetten an [8, 74].

Die Mikrofacettensteigungsverteilungsfunktion ist im Rahmen dieser Arbeit durch

$$D(r; \mathbf{n}, \mathbf{h}) = \frac{r^4}{\pi((\mathbf{n} \cdot \mathbf{h})^2(r^4 - 1) + 1)^2} \quad (2.9)$$

gegeben [6, 26] und beschreibt den Anteil der Facetten, die in Richtung des Halbvektors $\mathbf{h}$ orientiert sind [8].

Der Fresnel-Term, welcher hier durch

$$F(s; \mathbf{h}, \mathbf{v}) = F_0 + (1 - F_0) 2^{(-5.55473(\mathbf{v} \cdot \mathbf{h}) - 6.98316)(\mathbf{v} \cdot \mathbf{h})} \quad (2.10)$$

gegeben ist, wobei F_0 dem Glanzlichtreflexionsfaktor s entspricht [6, 26], beschreibt die anteilige Reflexion des Lichts von den glatten Mikrofacetten [8]. Im Rahmen dieser Arbeit besitzt der Fresnel-Term eine RGB-Repräsentation und wird somit im Folgenden durch einen Vektor beschrieben.

Das Modell erfüllt die Eigenschaften der Helmholtz-Reziprozität, jedoch nicht der Energieerhaltung [21].

Die sich aus den Formeln ergebenden Materialparameter belaufen sich auf die diffuse Grundfarbe $\mathbf{d}$, die Glanzfarbe $\mathbf{s}$ und den skalaren Rauheitswert r ; die Koeffizienten der diffusen Grundfarbe und der Glanzfarbe werden dabei im RGB-Format angegeben. Des Weiteren kann an jedem Punkt der Oberfläche ein Normalenvektor $\mathbf{n}$ angegeben werden; gemeinsam stellen $\mathbf{n}$, $\mathbf{d}$, r und $\mathbf{s}$ Materialparameter dar, die es im Rahmen der inversen Renderingfunktion an jedem Punkt der Oberfläche zu ermitteln gilt.

2.1.4 Bilderzeugung

In diesem Abschnitt wird die für das Rendering zentrale Rendering-Gleichung vorgestellt, die eine Generalisierung bekannter Bilderzeugungsgleichungen darstellt [25]. Zudem werden Methoden der Bilderzeugung aufgegriffen.

2.1.4.1 Rendering-Gleichung

Die Rendering-Gleichung

$$L(\mathbf{x}, \mathbf{x}') = L_e(\mathbf{x}, \mathbf{x}') + \int_{\mathcal{S}} f_r(\mathbf{v}_i, \mathbf{v}_r) L(\mathbf{x}', \mathbf{x}'') G(\mathbf{x}', \mathbf{x}'') V(\mathbf{x}', \mathbf{x}'') dA \quad (2.11)$$

gibt die Strahlungsdichte $L(\mathbf{x}, \mathbf{x}')$ des Lichts an dem Punkt $\mathbf{x}$ ausgehend von dem Punkt $\mathbf{x}'$ an, wobei $L_e(\mathbf{x}, \mathbf{x}')$ die Intensität der emittierten Strahlung vom Punkt $\mathbf{x}'$ in Richtung $\mathbf{x}$ beschreibt und $\mathcal{S} = \bigcup S_i$ die Menge aller Oberflächen darstellt. Das Integral wird über alle Oberflächenpunkte $\mathbf{x}''$ gebildet und berechnet die von diesen Punkten ausgehende Intensität, die über den Punkt $\mathbf{x}'$ reflektiert am Punkt $\mathbf{x}$ ankommt. Dazu wird der Reflexionskoeffizient $f_r(\mathbf{v}_i, \mathbf{v}_r)$ der Oberfläche mit der Intensität $L(\mathbf{x}', \mathbf{x}'')$, welche die Strahlungsdichte am Punkt $\mathbf{x}'$ ausgehend vom Punkt $\mathbf{x}''$ beschreibt, multipliziert [7, 25]. Der Vektor $\mathbf{v}_i$ ist der normierte Richtungsvektor ausgehend von dem Punkt $\mathbf{x}'$ in Richtung $\mathbf{x}''$ und analog dazu ist $\mathbf{v}_r$ der normierte Richtungsvektor ausgehend von $\mathbf{x}'$ in Richtung $\mathbf{x}$. Die Sichtbarkeitsfunktion $V(\mathbf{x}, \mathbf{x}')$ gibt den Wert 1 zurück, falls der Punkt $\mathbf{x}$ und der Punkt $\mathbf{x}'$ wechselseitig sichtbar sind [7]. Der Term $G(\mathbf{x}, \mathbf{x}')$ beschreibt im Rahmen des Integrals die Projektion des

Raumwinkels einer Quelle auf die entsprechende Fläche; die Herleitung dieses Terms ist in der Arbeit von Cohen et al. [7] nachzuschlagen und wird hier nicht weiter behandelt.

Die Rendering-Gleichung analytisch zu lösen ist im Allgemeinen nicht möglich; durch Monte-Carlo-Verfahren und Ray-Tracing-Algorithmen kann das Integral der Rendering-Gleichung ausgewertet werden [7, 57]. In dieser Arbeit werden derartige Verfahren nicht eingesetzt; stattdessen wird im Rahmen dieser Arbeit die lokale Beleuchtung einer Szene vorgenommen. Im nächsten Abschnitt wird das Konzept der lokalen Beleuchtung vorgestellt.

2.1.4.2 Lokale Beleuchtung

Im Folgenden wird das Konzept der lokalen Beleuchtung aufbauend auf der Reflektanz-Gleichung vorgestellt; die Reflektanz-Gleichung lässt sich dabei aus der Rendering-Gleichung ableiten und ist gegeben durch

$$L(\mathbf{x}, \mathbf{x}') = \int_{\Omega_i} f_r(\mathbf{v}_i, \mathbf{v}_r) L(\mathbf{x}', \mathbf{x}'') \cos \theta_i d\omega_i. \quad (2.12)$$

Sie beschreibt die reflektierte Strahlungsdichte ausgehend von dem Punkt $\mathbf{x}'$ in Richtung des Punktes $\mathbf{x}$. Die Strahlungsdichte berechnet sich durch die eingehende Strahlungsdichte $L(\mathbf{x}', \mathbf{x}'')$ am Punkt $\mathbf{x}'$ aus der Richtung des Punktes $\mathbf{x}''$ aus allen Richtungen der Hemispähre Ω_i des Punktes $\mathbf{x}'$; die eingehende Strahlungsdichte wird zudem durch die BRDF f_r und den projizierten Raumwinkel $\cos \theta_i d\omega_i$ gewichtet [7]. Die Vektoren $\mathbf{v}_i$ und $\mathbf{v}_r$ sind dabei normierte Richtungsvektoren ausgehend von dem Punkt $\mathbf{x}'$ in Richtung $\mathbf{x}''$ und $\mathbf{x}$ in dieser Reihenfolge.

Zudem werden in dieser Arbeit ausschließlich Punktlichtquellen betrachtet; wird die Formel der Strahlungsdichte bezüglich einer Punktlichtquelle in die Reflektanz-Gleichung eingesetzt, so resultiert daraus die Gleichung

$$L(\mathbf{x}, \mathbf{x}') = \frac{\Phi}{4\pi \|\mathbf{x}' - \mathbf{x}''\|_2^2} f_r(\mathbf{v}_i, \mathbf{v}_r) \cos \theta_i, \quad (2.13)$$

wobei $\mathbf{x}''$ den Punkt der Lichtquelle, $\mathbf{x}'$ den beleuchteten Oberflächenpunkt, $\mathbf{x}$ den Augenpunkt und Φ den Strahlungsfluss der Punktlichtquelle darstellt [7] und $\mathbf{v}_i$ und $\mathbf{v}_r$ wie zuvor definiert werden. Der Winkel θ_i beschreibt hierbei den Winkel zwischen der Lichtquellenrichtung und der Normalen am Punkt $\mathbf{x}'$.

Für die Berechnung der Strahlungsdichte in einer Szene mit mehreren Lichtquellen wird die vorausgehende Gleichung über alle Punktlichtquellen summiert [7].

2.1.4.3 G-Buffer-Verfahren

Im Rahmen dieser Arbeit wird statt der klassischen Bilderzeugungspipeline das G-Buffer-Verfahren [65] eingesetzt. Klassisches Rendering beschreibt hierbei den Prozess, bei dem eine 3D-Szene in Form von Objektbeschreibungen eine Pipeline durchläuft und so am Ende ein beleuchtetes Bild synthetisiert wird; solche Pipelines beinhalten oft Prozesse wie die geometrische Transformation von Objekten und Beleuchtungsberechnungen [79]. Moderne Grafik-APIs besitzen Pipelines mit über zehn verschiedenen Schritten; viele dieser Schritte führen dabei frei programmierbaren Code, die sogenannten *Shader*, aus [73].

Bei dem G-Buffer-Verfahren werden geometrische Eigenschaften in geometrischen Puffern (G-Buffer) abgespeichert; diese werden durch Bilder dargestellt, die Zwischenresultate des Renderingprozesses festhalten [65]. In dieser Arbeit speichern die G-Buffer die pixelweisen Positionen, Normalenvektoren und weiteren Materialeigenschaften für die Beleuchtungsberechnung.

Die lokale Beleuchtung, die im vorausgehenden Abschnitt beschrieben wurde, wird im Rahmen dieser Arbeit auf den G-Buffern durchgeführt; das Resultat ist ein beleuchtetes Bild. Der Vorteil für diese Arbeit besteht darin, dass sich der gesamte Renderingprozess bei gegebenen G-Buffern auf eine pixelweise Evaluierung reduziert; insbesondere müssen an dieser Stelle keine geometrischen Transformationen von 3D-Szenenbeschreibungen durchgeführt werden. Auch wird hier kein erweitertes Kameramodell benötigt, welches die Grundlage solcher geometrischen Transformationen bildet und oft auf dem Lochkameramodell aufbaut [56].

Im nächsten Abschnitt wird die Repräsentationsform der diesem Verfahren entsprechend resultierenden beleuchteten Szenen erläutert.

2.1.5 Repräsentation der beleuchteten Bilder

Die Repräsentation der synthetischen Beleuchtungsergebnisse findet in Form von RGB-Farbbildern statt. Eine RGB-Farbe stellt sich aus der gewichteten Summe der roten, grünen und blauen Farbkomponente dar [78]. Bei der Darstellung ist sie das Resultat der gewichteten Summe der zu diesen Farben korrespondierenden spektralen Kurven des Darstellungsgerätes und steht somit in Abhängigkeit zu demselben [55]. Die korrekte Farbdarstellung stellt ein komplexes Thema dar; an dieser Stelle wird auf die entsprechende Literatur verwiesen [55, 69, 70, 78]. Im Rahmen dieser Arbeit wird das RGB-Farbmodell unabhängig von dem Darstellungsgerät für die internen Berechnungen während der Bildsynthese und auch zur Abspeicherung der

Ergebnisse eingesetzt.

Bilder, die pixelweise Intensitäten im richtigen Verhältnis zu der Szene darstellen, besitzen einen hohen Dynamikumfang, die sogenannte *High Dynamic Range* (HDR). Bilder hingegen, die für die Darstellung angepasst sind, besitzen einen niedrigen Dynamikumfang, die sogenannte *Low Dynamic Range* (LDR) [60]. Entsprechend dieser Definition werden im Rahmen dieser Arbeit jegliche Bilder, die Intensitäten im Intervall $[0, 1]$ besitzen, mit niedrigem Dynamikumfang und Bilder, die Intensitäten außerhalb dieses Intervalls besitzen mit hohem Dynamikumfang angesehen.

Die Transformation aus dem HDR in das LDR mit akkurater Repräsentation der aufgenommenen Szene stellt das *HDR tone-mapping* Problem dar [61]. Im Rahmen dieser Arbeit wird ein Clamping der gerenderten Ergebnisse vorgenommen; dabei liegen diese zunächst im HDR-Format mit korrekten Intensitäten vor und werden durch das Abschneiden von Werten größer als eins in das LDR-Format gebracht.

2.1.6 Inverses Rendering

Basierend auf einem detaillierten Modell der Szene samt Lichtquelleninformationen kann ein physikalisch-basierter Renderingalgorithmus realistische Bilder erzeugen. Dazu werden neben der Geometrie von den Objekten der Szene auch deren Materialeigenschaften, beispielsweise in Form einer BRDF, benötigt [59].

Das *inverse Rendering* beschreibt den Prozess der Akquise der BRDF und weiteren für das Rendering relevanten Informationen, die beispielsweise durch die Beleuchtungsumstände und die Texturen der Szene dargestellt werden [59].

Im Rahmen dieser Arbeit bezieht sich der Begriff des inversen Renderings primär auf die pixelweise Rekonstruktion der BRDF für ein gegebenes Beleuchtungsmodell. Eine ausgiebige Definition für das inverse Rendering für planare Oberflächen wird in Abschnitt 3.1 vorgenommen; die Definition des inversen Renderings für räumliche Szenen findet in Abschnitt 4.1 statt.

Im folgenden Abschnitt werden für diese Arbeit relevante verwandte Arbeiten bezüglich des inversen Renderings vorgestellt.

2.2 Verwandte Arbeiten

Das inverse Rendering beschreibt im Rahmen dieser Arbeit primär den Prozess der BRDF-Akquise; die Berechnung von den geometrischen Formen [40], der Tiefe [36] oder der Beleuchtung [35] einer Szene sind hierbei nicht von Relevanz.

Des Weiteren wird hier eine Unterscheidung zwischen leichtgewichtigen BRDF-Akquisemethoden und klassischen Messverfahren [21] gemacht; Messungen mittels Apparaturen, wie sie beispielsweise durch den Gonioreflectometer [21, 34] dargestellt werden, sind an dieser Stelle nicht von Interesse. Stattdessen werden Verfahren betrachtet, die mittels einer einfachen Digitalkamera, z.B. der Smartphonekamera, imstande sind, eine BRDF-Akquise durchzuführen [11].

Zudem wird zwischen Deep-Learning-gestützten Verfahren und analytischen Methoden differenziert. Analytische Methoden versuchen dabei mittels eines Optimierungsverfahrens eine Fehlerfunktion bezogen auf die Zielparameter zu reduzieren. Beispielsweise führt eine auf einem solchen Ansatz basierende Arbeit eine SVBRDF-Rekonstruktion anhand von zwei Fotos planarer Oberflächenmaterialien durch [2]; dabei wird ein Foto ohne und ein Foto mit eingeschaltetem Blitzlicht aufgenommen. Bei diesem exemplarischen Ansatz wird eine Einschränkung auf texturartige Materialien durchgeführt; die Selbstähnlichkeit von solchen Materialien wird im Rahmen der zielparameterbezogenen Optimierung ausgenutzt [2]. Ein weiteres Verfahren fertigt eine BRDF-Schätzung ausgehend von zwei Fotos eines homogenen planaren Oberflächenmaterials an [80]; dieser Ansatz nutzt dabei eine BRDF-Datenbank von 100 gemessenen Materialien [43] und optimiert basierend auf den Eingabebildern einen Gewichtungsvektor für Hauptkomponenten dieser Datenbank, um so die BRDF des dargestellten Materials zu approximieren. Ein anderer Ansatz versucht die Geometrie und Grundfarbe einer Szene mittels einer RGBD-Kamera durch die Optimierung einer Energiefunktion über eine Sequenz von Bildern zu rekonstruieren [22].

Inspiziert durch den Erfolg von Deep Neural Networks im Bereich des Renderings [28, 46] und der BRDF-Akquise [11, 12, 38] wird jedoch in dieser Arbeit der Fokus auf lernbasierte Verfahren gelegt. Diese datenbasierten Verfahren versuchen dabei a priori Annahmen über den möglichen Lösungsraum implizit über einen Datensatz zu lernen, statt sich auf eine manuelle Konstruktion dieser Annahmen zu stützen [11]. An dieser Stelle wird der Fokus zunächst auf die BRDF-Akquise von planaren Oberflächenmaterialien gelegt, was zugleich das Ziel des Kapitels 3 beschreibt.

Eine weitere Unterscheidung wird bei der Anzahl der Eingabefotos vorgenommen; so konzentriert sich diese Arbeit zunächst auf die SVBRDF-Akquise ausgehend von einem einzelnen Foto. Eine Vielzahl an Arbeiten beschäftigt sich mit der SVBRDF-Approximation ausgehend von einem beleuchteten Foto [11, 38, 39, 40]. Ausgehend von einem kleinen Datensatz ermöglicht ein Ansatz mittels der sogenannten Selbstaugmentierung das Training eines Deep Neural Networks für die pixelweise BRDF-

Bestimmung [38]; dieser Ansatz nutzt das BRDF-Modell von Ward [77]. Ein anderer Ansatz trainiert ebenfalls ein Convolutional Neural Network mittels eines großen SVBRDF-Datensatzes und einer Fehlerregularisierung basierend auf den tatsächlichen Materialkarten [39]. Die Grundlage des Kapitels 3 bildet dabei eine Arbeit, die ein Rendering-gestütztes Fehlermaß für das Training eines Deep Neural Networks einsetzt [11]; zudem macht dieser Ansatz Gebrauch von einem noch größeren Datensatz als die vorausgehenden zwei Methoden. Zusätzlich erweitert dieser Ansatz die initiale Architektur mittels eines anwendungsspezifischen parallelen Netzwerkes, um Flecken, die auch in den SVBRDF-Schätzungen des vorausgehenden Ansatzes entstehen, zu eliminieren. Alle drei Ansätze machen dabei Gebrauch von Encoder-Decoder-Architekturen.

Kürzlich erschienene Arbeiten konnten durch den Einsatz von mehreren Fotos des selben planaren Oberflächenmaterials unter verschiedenen Beleuchtungsbedingungen wesentlich bessere Ergebnisse als ähnliche Verfahren mit lediglich einem Foto als Eingabe erzielen [12, 16]. So kann durch die parallele Instanziierung mehrerer Deep Neural Networks für jeweils ein Eingabefoto die resultierende Information jedes dieser Netzwerke in Kombination genutzt werden, um eine deutlich bessere SVBRDF-Schätzung zu generieren [12]. Ein alternatives Verfahren für die Verarbeitung von multiplen Eingabefotos versucht eine anhand einer initialen Schätzung entstandene latente Repräsentation der SVBRDF basierend auf den weiteren Eingabefotos zu optimieren und so die initiale Schätzung erheblich zu verbessern [16]. Beide Verfahren werden im Rahmen des Kapitels 3 näher erläutert.

Das Kapitel 4 beschäftigt sich mit der Deep-Learning-gestützten Rekonstruktion von einer pixelweisen BRDF einer räumlichen Szene ausgehend von einem einzelnen Foto und einer Positionsangabe der Pixel relativ zur Kamera. Ein Verfahren für die Approximation der BRDF eines Objektes mit homogenem Oberflächenmaterial beschäftigt sich mit einer ähnlichen Zielsetzung [17]; jedoch ist das Ziel des Kapitels 4 eine pixelweise SVBRDF der gesamten Szene zu rekonstruieren, was ein weitaus komplexeres Problem darstellt. Die Deep-Learning-gestützte Beleuchtungsberechnung räumlicher Szenen ausgehend von Materialkarten einer Szene [46] stellt eine exakte inverse Aufgabenstellung zu diesem Vorhaben dar.

Abschließend wird auf den Aspekt des Rendering-gestützten Fehlermaßes eingegangen; so ein Fehlermaß baut dabei auf einer differenzierbaren Renderingkomponente auf [11]. Verschiedene Arbeiten haben sich mit differenzierbaren Renderingkomponenten beschäftigt [11, 37, 41, 81]. Das konkrete Konzept dieser Arbeit wird im folgenden Abschnitt 2.3 vorgestellt.

2.3 Konzeptionelle Übersicht

In dieser Arbeit wird die inverse Renderingfunktion mit dem Bezug auf zwei verschiedene Zielsetzungen unterschiedlich approximiert. Zunächst wird das Konzept der inversen Renderingfunktion für planare Oberflächenmaterialien vorgestellt; anschließend wird die Übertragung der Methoden auf räumliche Szenen erläutert. In beiden Fällen bezieht sich die inverse Renderingfunktion auf die Rekonstruktion von pixelweisen Beleuchtungsparametern und den Normalen jedes Pixels.

Im Rahmen des Kapitels 3 wird der Stand der Wissenschaft bezüglich der leichtgewichtigen pixelweisen SVBRDF-Akquise von Materialien planarer Oberflächen ausgehend von einem einzelnen Blitzlichtfoto untersucht; die Grundlage bildet dabei die Arbeit von Deschaintre et al. [11]. Hierbei wird ein Deep-Learning-gestützter Ansatz mittels einer Encoder-Decoder-Architektur und einer anwendungsspezifischen Erweiterung verfolgt; das Deep Neural Network wird mittels einer differenzierbaren Renderingkomponente im Rahmen eines stochastischen Rendering-gestützten Fehlermaßes trainiert; das eingesetzte Beleuchtungsmodell bezieht sich dabei auf das Cook-Torrance-Modell (siehe Abschnitt 2.1.3.2). Die differenzierbare Renderingkomponente setzt ein lokales Beleuchtungsverfahren basierend auf der G-Buffer-Methode um. Der zentrale Datensatz beläuft sich auf SVBRDF-Materialkarten samt einem darauf basierend gerenderten Blitzlichtfoto, wobei das Blitzlichtfoto als Eingabe und die dazu korrespondierenden Materialkarten als gewünschte Ausgabe fungieren. Des Weiteren werden Verfahren für multiple Eingabefotos [12, 16] und verschiedene Datenaugmentationsverfahren für SVBRDF-Datensätze vorgestellt.

Das Kapitel 4 überträgt aufbauend auf dem vorausgehenden Konzept die Methode der inversen Renderingfunktion planarer Oberflächenmaterialien auf Szenen mit räumlicher Geometrie. Insbesondere werden auch bei diesem Ansatz Deep-Learning-gestützte Verfahren im Rahmen einer Encoder-Decoder-Architektur, einer differenzierbaren Renderingkomponente und einem stochastischen Rendering-gestützten Fehlermaß eingesetzt. Hierbei wird der differenzierbare Renderer für das Blinn-Phong-Modell (siehe Abschnitt 2.1.3.1) ausgelegt; dieser setzt ebenfalls auf lokale Beleuchtungsmethoden aufbauend auf dem G-Buffer-Verfahren. Der zentrale Datensatz dieses Ansatzes beläuft sich auf HDR-Materialkarten räumlicher Szenen samt der pixelweisen Positionsinformation relativ zur Kamera [46]; die Eingabe wird durch ein ad-hoc beleuchtetes Foto einer 3D-Szene und der dazu korrespondierenden Positionskarte dargestellt; die Ausgabe beläuft sich auf die pixelweise diffuse Grundfarbe und die Normale jedes Pixels.

3 Inverses Rendering für planare Oberflächen

In diesem Kapitel wird ein Deep-Learning-gestütztes Verfahren für das inverse Rendering planarer Oberflächen vorgestellt. Das inverse Rendering bezieht sich im Rahmen dieses Kapitels auf die Approximation der SVBRDF in Form von Materialkarten des dargestellten Materials basierend auf einem einzelnen Blitzlichtfoto.

Abschnitt 3.1 führt eine formale Definition der Problemstellung ein. In Abschnitt 3.2 wird ein Überblick über die umgesetzten Lösungsansätze und die relevanten Verfahren gegeben, die den Fokus dieses Kapitels darstellen. Der Abschnitt 3.3 stellt die Architektur und das Training des eingesetzten Deep Neural Networks vor. Ein Rendering-gestütztes Fehlermaß für das Training eines solchen Netzes wird in Abschnitt 3.4 thematisiert. Verschiedene Datenaugmentationsverfahren werden in Abschnitt 3.5 behandelt. Verfahren, die mehrere Eingabebilder desselben Materials unter unterschiedlichen Beleuchtungsvoraussetzungen für das inverse Rendering nutzen, werden in Abschnitt 3.6 dargelegt. Die Realisierung der vorgestellten Verfahren wird in Abschnitt 3.7 diskutiert. Abschließend wird in Abschnitt 3.8 eine Evaluierung der realisierten Methoden vorgenommen.

3.1 Definition

Das Ziel dieses Kapitels besteht darin, eine inverse Renderingfunktion für planare Oberflächen zu untersuchen und umzusetzen. Der Begriff des inversen Renderings bezieht sich im Rahmen dieses Kapitels primär auf die Rekonstruktion bzw. Approximation der räumlich variierenden bidirektionalen Reflexionsdistributionsfunktion (SVBRDF) [21] ausgehend von einem einzelnen Eingabefoto des zu approximierenden Materials. Die SVBRDF gibt das variierende Reflexionsverhalten eines Materials für jeden Punkt der Oberfläche an. Sie wird durch einen Satz an Materialkarten in Kombination mit einem entsprechenden Beleuchtungsmodell repräsentiert. Materialkarten halten hierbei für jedes Pixel des Eingabebildes Materialinformationen fest,

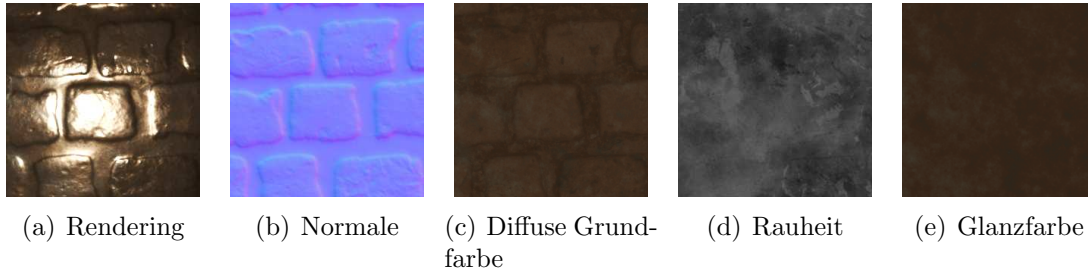


Abbildung 3.1: Datum bestehend aus dem Rendering und der entsprechenden SVBRDF [11]. Abbildung (a) stellt das beleuchtete Bild, welches das Resultat eines physikalisch-basierten Renderings ist, dar. Abbildungen (b) bis (e) stellen in Assoziation mit dem Cook-Torrance-Beleuchtungsmodell [8] die SVBRDF des Materials dar. Diese halten für jedes Pixel (b) die Normale, (c) die diffuse Grundfarbe, (d) den skalaren Rauheitswert und (e) die Glanzfarbe fest.

die in einem späteren Renderingprozess für eine Neubeleuchtung des Materials eingesetzt werden können (siehe Abbildung 3.1).

Sei $\mathbf{I} \in \mathcal{I}$ mit $\mathcal{I} \subset \mathbb{R}^{m \times n \times 3}$ ein Bild der Größe $m \times n$ eines planaren Oberflächenmaterials und $\mathbf{M} \in \mathcal{M}$ mit $\mathcal{M} \subset \mathbb{R}^{m \times n \times 10}$ der korrespondierenden Satz an Materialkarten, wobei $m, n \in \mathbb{N}$ gilt. Hierbei liegen die Materialinformationen für ein Pixel $\mathbf{i}_{i,j} \in \mathbf{I}$ an der Position (i, j) des Eingabefotos in der Form $\mathbf{m}_{i,j} \in \mathbf{M}$ mit $\mathbf{m}_{i,j} = (\mathbf{m}_{i,j}^{(n)}, \mathbf{m}_{i,j}^{(d)}, m_{i,j}^{(r)}, \mathbf{m}_{i,j}^{(s)})$ vor, wobei $\mathbf{m}_{i,j}^{(n)} \in \mathbb{R}^3$ den Normalenvektor, $\mathbf{m}_{i,j}^{(d)} \in \mathbb{R}^3$ die diffuse Grundfarbe, $m_{i,j}^{(r)} \in \mathbb{R}$ den skalaren Rauheitswert und $\mathbf{m}_{i,j}^{(s)} \in \mathbb{R}^3$ die Glanzfarbe des jeweiligen Pixels festhält. Des Weiteren sei eine Renderingfunktion, die einen solchen Satz an Materialkarten für eine physikalisch-basierte Beleuchtung verwenden kann, gegeben durch $R : \mathcal{M} \times \mathcal{L} \times \mathcal{V} \rightarrow \mathcal{I}$, wobei $\mathbf{l} \in \mathcal{L}$ mit $\mathcal{L} \subset \mathbb{R}^3$ eine Lichtquellenposition und $\mathbf{v} \in \mathcal{V}$ mit $\mathcal{V} \subset \mathbb{R}^3$ einen Augenpunkt darstellen; hierbei ist die Geometrie der Szene dem Renderer bekannt und bezieht sich auf eine fest positionierte rechteckige planare Oberfläche. Die ideale inverse Renderingfunktion $Q : \mathcal{I} \rightarrow \mathcal{M}$ für planare Oberflächen ist definiert als eine Abbildung, die ein solches Eingabebild $\mathbf{I}$ eines planaren Oberflächenmaterials auf ihre Materialkarten $\mathbf{M}$ abbildet.

Im Rahmen dieses Kapitels wird eine inverse Renderingfunktion Q als eine Funktion definiert, welche die Zielfunktion

$$\min_Q \mathbb{E}_{(\mathbf{l}, \mathbf{v})} \delta(R(\mathbf{M}, \mathbf{l}, \mathbf{v}), R(Q(\mathbf{I}), \mathbf{l}, \mathbf{v})) \quad (3.1)$$

minimiert. Hierbei wird eine inverse Renderingfunktion Q gesucht, die den Erwartungswert bezüglich der Fehlerfunktion δ zweier Renderings minimiert, wobei eines

der Renderings basierend auf den tatsächlichen Materialkarten $\mathbf{M}$ und das andere Rendering basierend auf den geschätzten Materialkarten $Q(\mathbf{I})$ synthetisiert wird; die Schätzung wird dabei mittels einer inversen Renderingfunktion Q ausgehend von dem zu den Materialkarten $\mathbf{M}$ korrespondierenden Blitzlichtfoto $\mathbf{I}$ angefertigt. Die Vektoren $\mathbf{l}$ und $\mathbf{v}$ stellen dabei jeweils die Lichtquellen- und Augenpunktposition in dieser Reihenfolge dar.

Die Approximation der inversen Renderingfunktion stellt mathematisch gesehen ein schlecht gestelltes Problem dar [11, 38, 39]. Im folgenden Abschnitt werden daher Lösungsansätze, die eine Approximation einer solchen inversen Renderingfunktion ermöglichen, vorgestellt.

3.2 Lösungsansätze und weiteres Vorgehen

In diesem Abschnitt wird eine Übersicht über das zentrale Verfahren dieses Kapitels gegeben, welches sich mit der inversen Renderingfunktion für planare Oberflächen beschäftigt und sich an der Arbeit von Deschaintre et al. [11] orientiert. Zudem werden weitere untersuchte und relevante Verfahren zusammenhängend erläutert.

In diesem Kapitel ist eine inverse Renderingfunktion Q gesucht, welche die Zielfunktion (3.1) erfüllt. Die Herausforderung bei der Approximation einer solchen inversen Renderingfunktion besteht darin, dass mehrere Kombinationen von Materialinformation dieselben Pixelwerte als Ergebnis eines Renderings liefern können. Somit stellt die inverse Renderingfunktion mathematisch gesehen eine schlecht gestellte Aufgabe dar. Eine mögliche Lösung besteht darin, mehrere Aufnahmen desselben Materials unter verschiedenen Beleuchtungsbedingungen zu nutzen um Ambiguitäten aufzulösen. Alternativ können a priori Annahmen über den möglichen Lösungsraum getroffen werden [11].

Um die Aufgabe des inversen Renderings im Rahmen dieses Kapitels zu lösen, wird ein datenbasierter Ansatz eingesetzt; Abbildung 3.2 gibt eine Übersicht über eingesetzte Methoden für die Umsetzung eines solchen Ansatzes. Dieser lässt sich dabei in einen Datengenerierungsteil und einen Trainingsteil aufteilen. Die zentralen Daten dieses Ansatzes belaufen sich hierbei auf SVBRDFs planarer Materialien; diese können zum einen durch die Aufnahme realer Fotos akquiriert werden [21]. Einen solchen Akquisitionsprozess zu entwickeln stellt jedoch genau das Ziel dieses Kapitels dar. Alternativ können manuell generierte SVBRDFs – beispielsweise dargestellt durch künstlergenerierte Materialien – eingesetzt werden [11]. Durch Datenaugmentationsverfahren kann ein initial kleiner Satz solch künstlergenerierter Materialien um ein

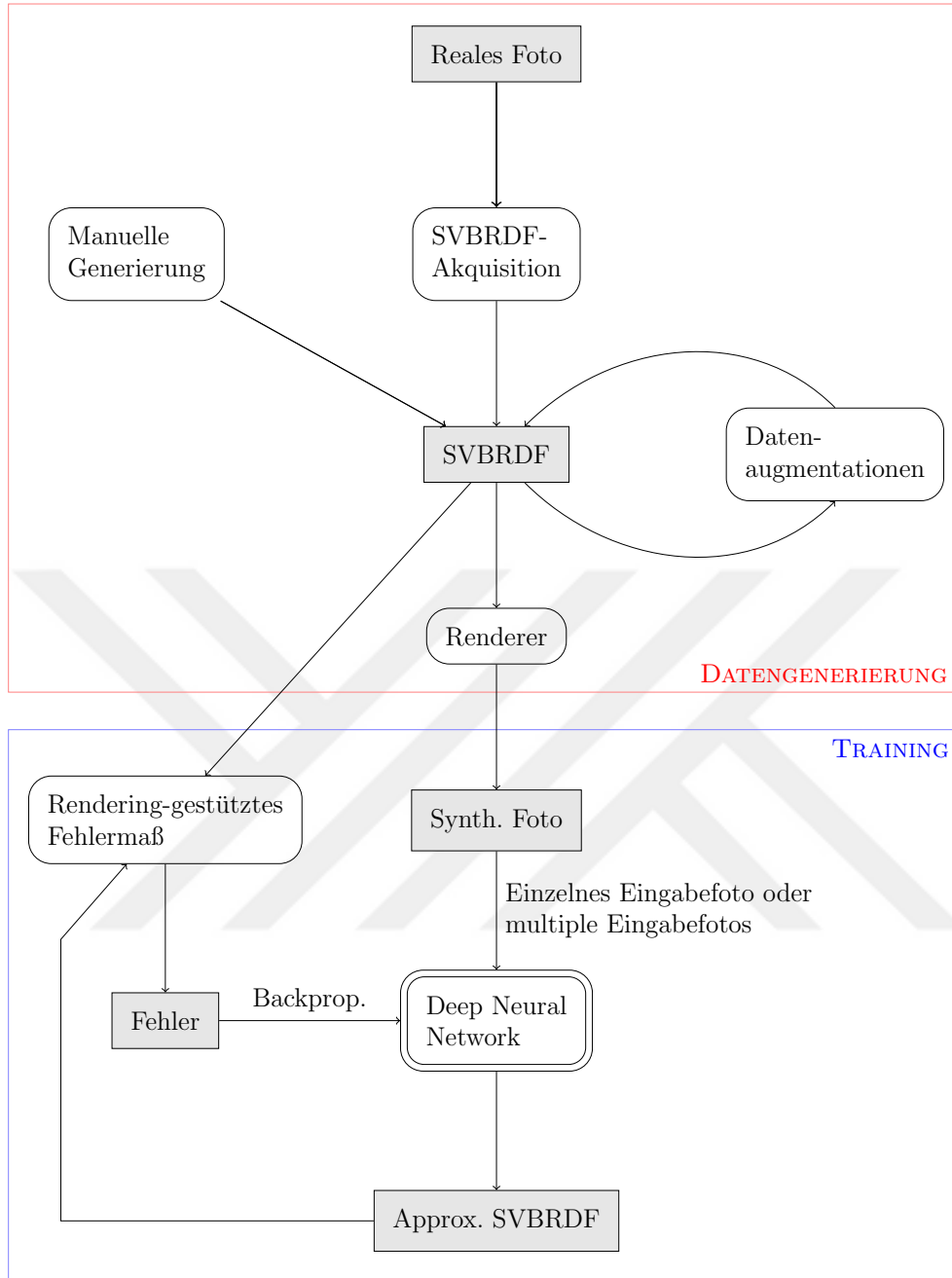


Abbildung 3.2: Übersicht über die zentralen Verfahren des dritten Kapitels. Die Ansätze lassen sich in die Datengenerierung (rot) und das Training (blau) aufteilen. Die SVBRDFs stellen die zentralen Daten dieses Ansatzes dar; diese können ausgehend von realen Fotos akquiriert oder durch manuelle Generierung erstellt werden. Mittels Datenaugmentationsverfahren kann ein initialer Datensatz um ein vielfaches vergrößert werden. Durch das Rendering eines SVBRDF-Datums kann ein synthetisches Foto erstellt werden. Dies dient als Eingabe für ein Deep Neural Network, welches eine Schätzung der zugrundeliegenden SVBRDF ausgibt. Alternativ können auch mehrere Fotos für eine akkuratere Approximation der SVBRDF eingesetzt werden. Der Fehler der Approximation wird mittels des Rendering-gestützten Fehlermaßes berechnet und an das Netzwerk zurück propagiert. Das trainierte Deep Neural Network stellt die approximierte inverse Renderingfunktion und somit die relevante Ausgabe des Verfahrens dar [11, 12].

Vielfaches vergrößert werden [11]; diverse Datenaugmentationsverfahren werden im Abschnitt 3.5 vorgestellt.

Eine SVBRDF wird im Rahmen dieser Arbeit durch einen Satz an Materialkarten und einem entsprechenden Beleuchtungsmodell dargestellt. Die Materialkarten sind dabei für das Cook-Torrance-Beleuchtungsverfahren [8] (siehe Abschnitt 2.1.3.2) ausgelegt und belaufen sich für jedes Pixel auf die jeweilige Normale, die diffuse Grundfarbe, den skalaren Rauheitswert und auf die Glanzfarbe. Für die SVBRDFs, die den zentralen Datensatz des datenbasierten Ansatzes dieses Kapitels darstellen, wird mittels eines physikalisch-basierten Renderingverfahrens ein beleuchtetes Blitzlichtfoto synthetisiert; hierbei stellt ein Paar, bestehend aus einer SVBRDF, dargestellt durch die Materialkarten, und dem darauf basierend gerenderten Blitzlichtfoto, jeweils ein Datum dar (siehe Abbildung 3.1) [11].

Ein solcher Datensatz kann als Grundlage für das Training eines Deep Neural Networks dienen, welches einen lernbasierten Ansatz darstellt; dieser approximiert eine Funktion f^* , in diesem Fall die inverse Renderingfunktion Q , ausgehend von einer Menge von Eingabe- und Ausgabepaaren $(\mathbf{x}, \mathbf{y})$ der Funktion mit $\mathbf{y} = f^*(\mathbf{x})$ durch die Optimierung der Parameter $\boldsymbol{\theta}$ eines solchen Netzes mit dem Ziel die Funktion $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ zu approximieren [19]. Hierbei stellt das synthetische Foto die Eingabe und die dazu korrespondierenden Materialkarten die erwartete Ausgabe dar [11]. Die Struktur und das Training eines solchen Deep Neural Networks wird in Abschnitt 3.3 thematisiert.

Ein einzelnes Eingabefoto beinhaltet oft nicht ausreichend Informationen, um die SVBRDF ausreichend detailliert zu rekonstruieren [12]; Ansätze, die imstande sind, mehrere Eingabefotos desselben Materials unter verschiedenen Beleuchtungsbedingungen als Eingabe zu verarbeiten, werden im Abschnitt 3.6 thematisiert.

Ein Vergleich der ausgegebenen SVBRDF-Schätzung mit der tatsächlichen SVBRDF bildet den Fehler, der an das Deep Neural Network propagiert wird. Hierbei findet jedoch kein direkter Vergleich zwischen den jeweiligen Materialkarten statt; stattdessen kommt ein Rendering-gestütztes Fehlermaß zum Einsatz (siehe Abschnitt 3.4) [11]. Dies hat den Hintergrund, dass die wechselseitigen Einwirkungen der Materialparameter bezüglich der resultierenden Renderings in einem direkten Vergleich nicht mit berücksichtigt werden. Visuell ähnliche Materialkarten müssen deswegen nicht zwangsweise zu visuell ähnlichen Renderings führen. Des Weiteren korrelieren die numerischen Differenzen in den Materialkarten nicht konsistent mit denen der resultierenden Pixelwerte nach der Beleuchtung. Das Rendering-gestützte Fehlermaß dagegen führt keinen direkten Vergleich der Materialkarten durch, sondern rendert

die jeweiligen SVBRDFs unter gegebenen Beleuchtungseinstellungen mittels einer differenzierbaren Renderingkomponente, die imstande ist, den resultierenden Fehler der jeweiligen Renderings an die geschätzte SVBRDF und somit an das Deep Neural Network zu propagieren [11].

3.3 Netzarchitektur und Training

Dieser Abschnitt befasst sich mit der Netzarchitektur und dem Training des Deep Neural Networks für die Aufgabe des inversen Renderings planarer Oberflächen.

Abschnitt 3.3.1 stellt die Grundarchitektur eines solchen Netzes vor, die dem Konzept des U-Nets [63] entspricht. Abschnitt 3.3.2 stellt eine anwendungsspezifische Erweiterung vor, die das Auftreten von unerwünschten Flecken in den Schätzungen des U-Nets vermeidet. Im Abschnitt 3.3.3 wird die Vorbereitung der Eingabe und Nachbearbeitung der Ausgabe des Deep Neural Networks während des Trainings vorgestellt. Abschließend wird die Grundidee des Trainings eines solchen Deep Neural Networks in Abschnitt 3.3.4 erläutert.

3.3.1 Grundarchitektur

In diesem Abschnitt wird die grundlegende Architektur des Deep Neural Networks für die Aufgabe des inversen Renderings vorgestellt.

Das Ziel besteht darin, für ein eingegebenes Blitzlichtfoto eines planaren Oberflächenmaterials pixelweise Materialinformationen zu schätzen. Hierbei beläuft sich die Eingabe auf ein RGB-Foto mit niedrigem Dynamikumfang. Dieses kann sowohl ein synthetisches Material, beispielsweise als Resultat eines physikalisch-basierten Renderingprozesses, als auch eine Aufnahme eines realen Materials darstellen. Die erwartete Ausgabe ist die SVBRDF des eingegebenen Materials in Form von Materialkarten, die für jedes Pixel der Eingabe Materialinformationen festhalten.

Die Approximation von einer SVBRDF basierend auf einem einzelnen Foto ist eine schlecht gestellte Aufgabe. Ein datenbasiertes Verfahren mittels eines Deep Neural Network ist in der Lage eine implizite a priori Annahme über den möglichen Lösungsraum aus einem ausreichend großen Datensatz zu lernen und stellt so einen vielversprechenden Lösungsansatz dar [11]. Convolutional Neural Networks (CNNs) haben sich als eine mächtige Methode für das Lernen dieser a priori Annahmen herausgestellt [11, 23].

Weit voneinander entfernte Regionen in einem Eingabebild können relevante Informationen zueinander besitzen [2, 11]. Um diese globalen Effekte über das gesamte

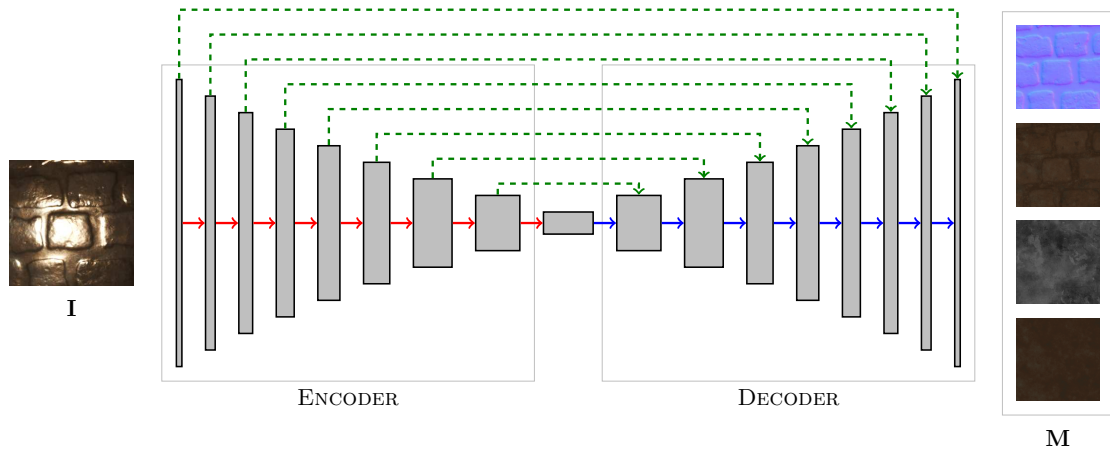


Abbildung 3.3: Die U-Net-Architektur. Die Eingabe (links) wird durch ein gerendertes Oberflächenmaterial **I** und die Ausgabe (rechts) durch einen Satz an Materialkarten **M** dargestellt. Das U-Net besteht aus einem Encoder und einem Decoder und sogenannten Skipconnections (dargestellt durch grüne Pfeile), die Zwischenergebnisse (grau) der Encoderschichten mit den entsprechenden Schichten im Decoder konkatenieren. Dies ermöglicht eine Übertragung feiner Details in die Ausgabe [11, 23]. Die roten Pfeile deuten hierbei Encoderoperationen und die blauen Pfeile Operationen des Decoders an, die entsprechende Zwischenergebnisse (grau) produzieren.

Bild zu modellieren, wird ein ausreichend großes rezeptives Feld benötigt; das rezeptive Feld bezieht sich dabei auf die Fläche auf einem Eingabebild, die von einem Pixel der Ausgabe abgedeckt werden kann [39]. Aus diesem Grund wird eine Netzstruktur eingesetzt, die aus einem Encoder und einem Decoder besteht [11]. Eine solche Struktur wird in Abbildung 3.3 schematisch dargestellt. Hierbei wird durch die Encoderkomponenten die Auflösung des Bildes in jeder Schicht verkleinert, während die Anzahl der Kanäle und somit die Anzahl erfassbarer Merkmale der Zwischenergebnisse vergrößert wird; das Resultat des Encoders ist dabei am Ende nur noch einen Pixel groß [11]. Dies weitet das rezeptive Feld für jedes Pixel der Ausgabe auf das gesamte Eingabebild aus [39] und zwingt die Encoderkomponenten dazu, relevante Informationen in einen globalen Merkmalsvektor zu komprimieren [11].

Die Aufgabe des Decoders besteht darin, ausgehend von diesem Vektor das gewünschte Ausgabebild, in diesem Fall die Materialkarten, zu erzeugen. Obwohl dieser Bottleneck, dargestellt durch den globalen Merkmalsvektor, notwendig für die Aggregation von Informationen weit voneinander entfernter Regionen ist, verhindert es die Reproduktion hochfrequenter Details im Ausgabebild. Um diese Details in die Ausgabe zu reintegrieren, werden sogenannte *Skipconnections* eingesetzt, die entsprechende Schichten des Encoders mittels einer Konkatenation an eine korrespon-

dierende Schicht des Decoders weiterleiten (siehe Abbildung 3.3); dies ermöglicht es dem Decoder relevante Details an den entsprechenden Stellen der Ausgabe zu synthetisieren [11]. Diese Architektur ist auch als U-Net bekannt [63].

Im Folgenden wird der detaillierte Aufbau des eingesetzten U-Nets aufgezeigt; dieser orientiert sich an der Arbeit von Deschaintre et al. [11], welche sich wiederum auf die Arbeit von Isola et al. [23] stützt. Hierbei wird die Notation $256 \sim 3$ genutzt, um für ein gegebenes Bild eine Breite und Höhe von 256 Pixeln und eine Anzahl von drei Kanälen anzugeben. Als Aktivierungsfunktion für die Encoder- und Decoderschichten wird die Leaky-ReLU-Funktion (LReLU) [42] mit einem Wert von 0.2 für den negativen Teil eingesetzt. Die LReLU-Aktivierungsfunktion besitzt für jede Eingabe einen Gradienten ungleich null; dies steht im Kontrast zu der normalen ReLU-Funktion, die einen Gradienten von null im nicht aktivierten Zustand besitzt [42]. Die im Folgenden genutzte Normalisierung bezieht sich dabei auf die Instanznormalisierung nach Ulyanov et al. [76], die für eine Stabilisierung des Trainings sorgt [11]. Hierbei wird eine Normalisierung auf einem einzelnen Bild statt auf dem gesamten Batch vorgenommen [76].

Zunächst wird der Encoder der Encoder-Decoder-Architektur vorgestellt. Dieser beginnt mit einer Faltung, die das Eingabebild von $256 \sim 3$ auf $128 \sim 64$ bringt; die Faltungen der Encoderkomponenten besitzen dabei alle eine Kernelgröße von vier, einen Stride von zwei und einen Padding von eins¹. Das Ergebnis wird mittels einer Skipconnection an die entsprechende Schicht des Decoders weitergeleitet. Anschließend wird die Aktivierungsfunktion angewendet und eine weitere Faltung durchgeführt, sodass das Bild in der Form $64 \sim 128$ vorliegt.

Die folgenden Encoderschichten sind in ihrem Aufbau identisch und werden in der Abbildung 3.4 schematisch dargestellt. Die Operationen zwischen zwei Zwischenergebnissen entsprechen dabei den roten Pfeilen in der Abbildung 3.3. Zunächst wird eine Instanznormalisierung durchgeführt, gefolgt von der Aktivierungsfunktion und abschließend einer Faltung, sodass das Bild die Formen $32 \sim 256$, $16 \sim 512$, $8 \sim 512$, $4 \sim 512$, $2 \sim 512$ und $1 \sim 512$ annimmt. Alle Zwischenergebnisse mit Ausnahme des letzten Bildes werden an korrespondierende Schichten des Decoders mittels einer Konkatination angefügt. Das letzte Bild mit einer Pixel Auflösung und 512 Kanälen kann als globaler Merkmalsvektor mit 512 Komponenten verstanden werden und bildet das Resultat des Encoders.

Der Aufbau des Decoders ist symmetrisch zum Encoder gehalten (siehe Abbildung

¹Siehe *A guide to convolution arithmetic for deep learning* von Dumoulin et al. [13] für die Erklärung der Begriffe Kernelgröße, Stride, Padding usw.

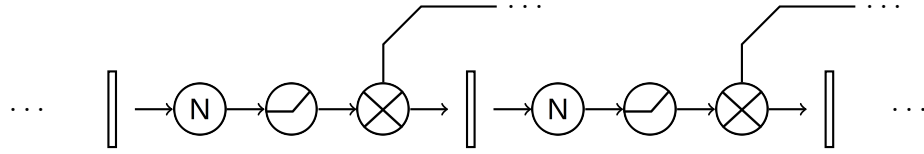


Abbildung 3.4: Detaillierte Struktur zweier aufeinanderfolgender Encoderschichten des U-Nets. Die Rechtecke stellen die Zwischenergebnisse, die in Form von Bildern vorliegen, dar. Eine Eingabe einer Encoderschicht durchläuft der Reihe nach zunächst eine Instanznormalisierung (N), dann die Aktivierungsfunktion und abschließend eine Faltung; dabei wird das resultierende Ergebnis der Faltung zusätzlich über eine Skipconnection – angedeutet durch die ausgehende obere Verbindung – an die korrespondierende Schicht im Decoder weitergeleitet.

3.3), sodass für jede Schicht des Encoders eine korrespondierende Schicht im Decoder vorliegt. Der Decoder erhält den globalen Merkmalsvektor als Eingabe und führt auf diesen zunächst eine Aktivierungsfunktion durch. Anschließend wird eine Doppelfaltung durchlaufen; diese besteht aus zwei aufeinanderfolgenden Faltungen mit einer Kernelgröße von fünf und einem Padding von zwei. Diese Konfiguration verändert die Dimension des Eingabebildes nicht, weswegen zuvor eine Upsampling-Operation eingesetzt wird, die das Bild mittels des Nearest-Neighbour-Verfahrens [11] auf die doppelte Größe skaliert. So liegt nach der ersten Schicht des Decoders das Bild in der Form $2 \sim 512$ vor.

Abbildung 3.5 stellt den Aufbau der verbleibenden Decoderschichten schematisch dar. Die Operationen zwischen zwei Zwischenergebnissen entsprechen dabei den blauen Pfeilen in Abbildung 3.3. In einer gegebenen Schicht wird zunächst eine Instanznormalisierung angewendet; daraufhin wird eine Dropout-Schicht mit einer Wahrscheinlichkeit von 50% eingesetzt. Die Dropout-Schicht liegt dabei nur in den

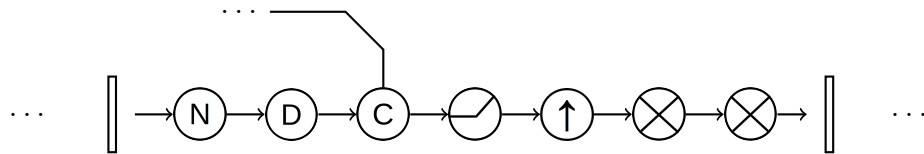


Abbildung 3.5: Detaillierte Struktur einer Decoderschicht. Das Eingabebild wird zunächst mittels einer Instanznormalisierung (N) normalisiert, dann folgt eine optionale Dropout-Schicht (D) mit einer Wahrscheinlichkeit von 0.5, anschließend wird das Bild aus der Skipconnection empfangen und mittels einer Konkatenation (C) der Kanäle an das Bild angefügt. Als Nächstes wird die Aktivierungsfunktion angewendet, woraufhin das resultierenden Bild mittels eines Upscalings auf die doppelte Größe skaliert und abschließend durch eine Doppelfaltung verarbeitet wird.

ersten drei Decoderschichten vor und ist dafür ausgelegt, ein Overfitting zu verhindern [67]. Nach dem Dropout wird eine Konkatenation mit dem Ergebnis der Faltung der korrespondierenden Encoderschicht mittels der Skipconnections vorgenommen. Daraufhin wird die Aktivierungsfunktion angewendet, gefolgt von einer Upsampling-Operation, die das Bild auf die doppelte Auflösung bringt. Abschließend wird eine Doppelfaltung angewendet.

So besitzen die Bilder am Ende der jeweiligen Decoderschicht die Form $4 \sim 512$, $8 \sim 512$, $16 \sim 512$, $32 \sim 256$, $64 \sim 128$, $128 \sim 64$ und $256 \sim 10$. Die Wahl der zehn Kanäle des letzten Bildes entspricht hierbei den benötigten zehn Kanälen der Materialkarten (siehe Abschnitt 3.1). Um die resultierenden Materialkarten in das Intervall $[0, 1]$ zu transformieren, wird die Sigmoid-Funktion angewendet [11].

3.3.2 Globales Merkmalsnetzwerk

In diesem Abschnitt wird eine anwendungsspezifische Erweiterung der U-Net-Architektur des letzten Abschnittes vorgestellt. Diese bezieht sich auf das Auftreten unerwünschter Flecken in den Ausgaben des trainierten U-Nets, die in fehlerhaften Renderings resultieren (siehe Abbildung 3.6); die Erweiterung hebt dieses Defizit größtenteils auf.

Weit auseinander liegende Regionen einer Materialprobe besitzen oft ergänzende Informationen zueinander, welche von Relevanz für den Prozess der SVBRDF-Approximation sind [2, 11]. Der Encoder der vorausgehenden Architektur ist so konstruiert, dass das rezeptive Feld jedes Ausgabepixels das gesamte Eingabebild umfasst [39]. Somit sollte theoretisch jedes Ausgabepixel in der Lage sein, Abhängigkeiten zu weit entfernten Pixeln aufzubauen [11].

Die Flecken auf den Materialkarten deuten jedoch daraufhin, dass diese globalen Abhängigkeiten nicht vollständig aufgebaut werden können. Diese Annahme wurde in einem Experiment getestet, bei dem für eine beliebige Eingabe die globale Durchschnittsfarbe dieses Bildes an jedem Pixel der Ausgabe ermittelt werden sollte [11]. Abbildung 3.7 bildet die Ergebnisse dieses Experiments ab. Hierbei ist zu sehen, dass die einfache U-Net-Architektur selbst eine scheinbar triviale Aufgabe wie das Ermitteln des Durchschnittsfarbwerts nicht ohne resultierende Flecken in der Ausgabe bewältigen konnte. Das Experiment bestätigt die Vermutung, dass Abhängigkeiten bezüglich globaler Informationen, dargestellt durch die Durchschnittsfarbe der Eingabe, nicht an jedem Pixel der Ausgabe aufgebaut werden können [11].

Zusätzlich wird in der Arbeit von Deschaintre et al. die Vermutung aufgestellt, dass die eingesetzte Instanznormalisierung die Fähigkeit des Netzwerks einschränkt glo-

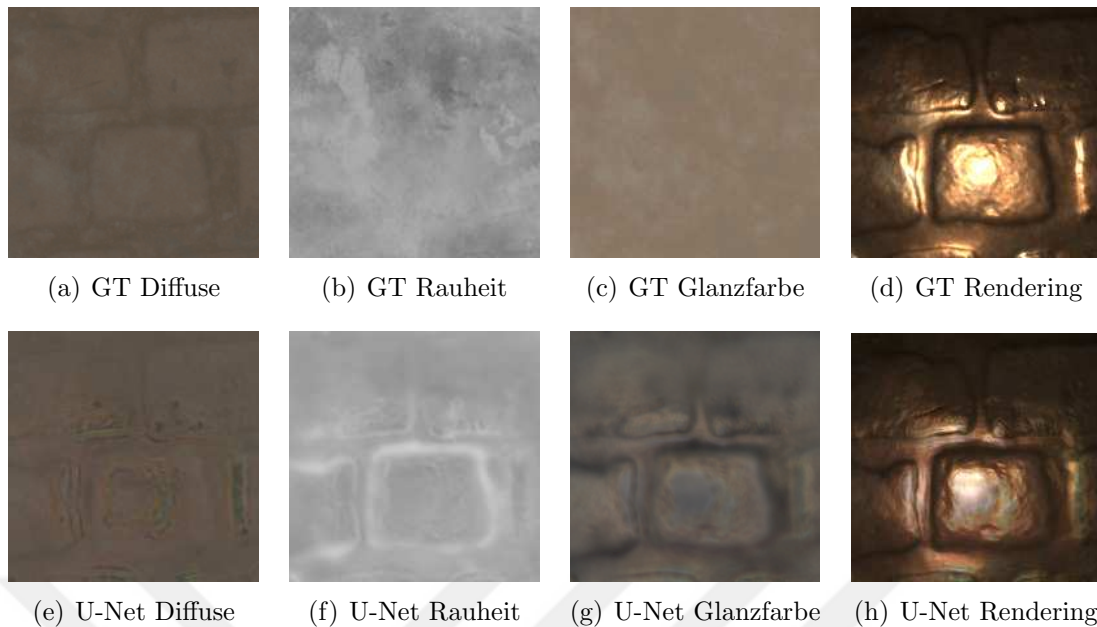


Abbildung 3.6: Flecken auf den Ausgabekarten. Abbildungen (a) bis (c) stellen einen Teil der tatsächlichen Karten [11] und Abbildungen (e) bis (g) die vom U-Net geschätzten entsprechenden Karten dar. Die Schätzung beruht auf einem Rendering basierend auf den tatsächlichen Karten. Die Karten beziehen sich jeweils auf die diffuse Grundfarbe, die Rauheit und die Glanzfarbe jedes Pixels, in dieser Reihenfolge. Basierend auf den jeweiligen Karten sind die Renderings (d) und (h) unter denselben Beleuchtungsbedingungen entstanden. Für eine bessere Visualisierung besitzen die Farbwerte der Materialkarten einen Exponenten von 0.4 und die der Renderings einen Exponenten von 1.125. In den vom U-Net geschätzten Karten sind unerwünschte Flecken nachzuweisen, die in einem fehlerhaften Rendering resultieren.

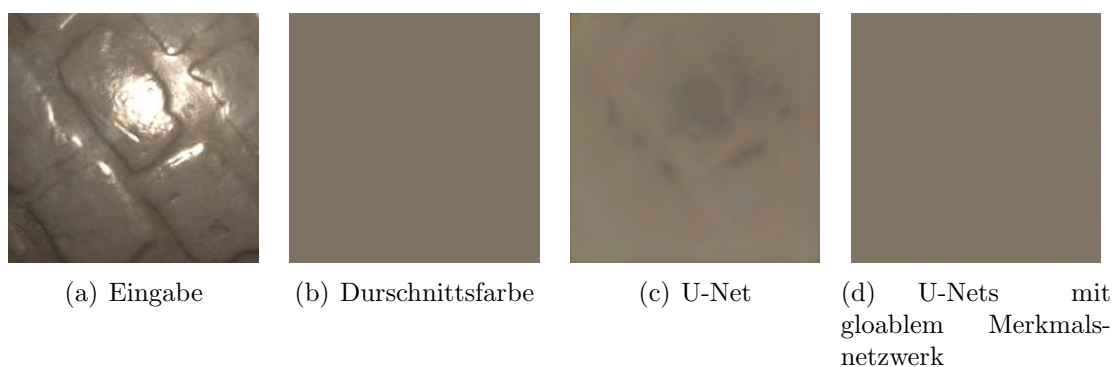


Abbildung 3.7: Ermitteln der Durchschnittsfarbe [11]. Abbildung (a) ist das Eingabebild, Abbildung (b) die Durchschnittsfarbe und somit das erwünschte Resultat. Abbildung (c) stellt die mit Flecken behaftete Schätzung der einfachen U-Net-Architektur dar. In Abbildung (d) wird eine größtenteils uniforme Schätzung der erweiterten U-Net-Architektur basierend auf einem globalen Merkmalsnetzwerk dargestellt [11].

bale Informationen zu ermitteln, da die gelernten Merkmale nach jeder Faltungsschicht durch eine Anpassung des Durchschnittswertes und der Standardabweichung standardisiert werden; obwohl die Normalisierung für die Stabilisierung des Trainings notwendig ist, verhindert sie somit die Propagierung globaler Informationen [11]. Ein Versuch, das Netzwerk ohne eine Normalisierung oder mit einer Variante der Instanznormalisierung ohne Subtraktion des Durchschnittswertes zu trainieren, resultierte in unerwünschten Beleuchtungsrückständen; Beleuchtungsrückstände stellen dabei Artefakte einer vorherigen Beleuchtung dar [11].

Im Folgenden wird eine Erweiterung der vorausgehenden Architektur vorgestellt, die in der Lage ist, sowohl globale Informationen an jedes Pixel weiterzuleiten als auch Durchschnittswerte vor einer Normalisierung zwischenspeichern, sodass diese nicht verworfen werden; diese Architektur orientiert sich an der Arbeit von Deschaintre et al. [11].

Die Erweiterung besteht aus einem globalen Merkmalsnetzwerk, welches bezüglich der Schichten analog zum Aufbau des vorgestellten U-Nets des letzten Abschnitts strukturiert ist; die Struktur der Erweiterung ist schematisch in Abbildung 3.8 dargestellt und wird durch die grünen Komponenten repräsentiert. Hierbei werden keine Bilder, sondern globale Merkmalsvektoren verarbeitet und folglich keine Faltungsschichten, sondern *Fully Connected Layer* eingesetzt. Die Aktivierungsfunktion beläuft sich im globalen Merkmalsnetzwerk stets auf die *Scaled Exponential Linear Units* (SELU) [33], welche dafür ausgelegt ist, das Training von Fully Connected

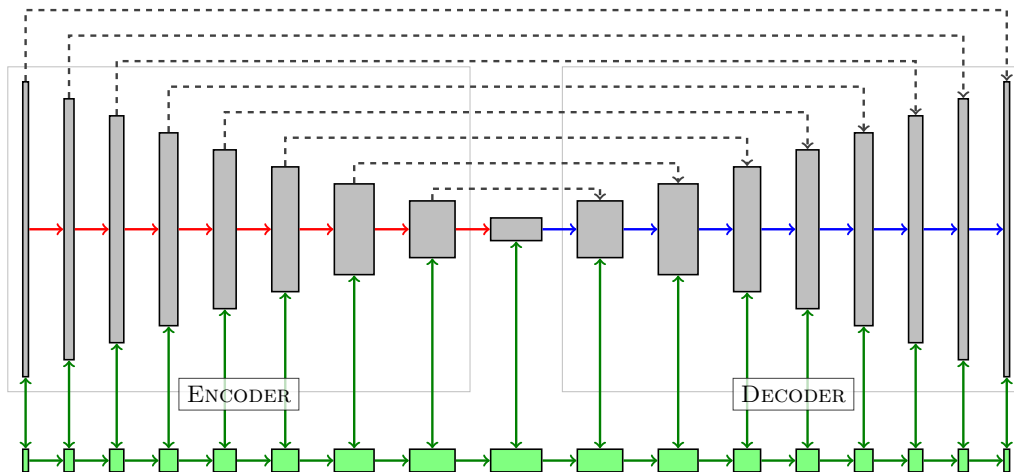


Abbildung 3.8: U-Net mit parallelem globalen Merkmalsnetzwerk. An jeder Schicht werden globale Informationen vor der Normalisierung extrahiert und an ein paralleles globales Merkmalsnetzwerk, welches aus Fully Connected Layern besteht, weitergeleitet. Gleichzeitig werden so Informationen vorausgehender Schichten in die aktuelle Schicht eingeführt [11].

Networks zu stabilisieren [11]. Des Weiteren wird die Funktion der Skipconnections nicht in das globale Merkmalsnetzwerk übernommen. Die somit resultierende Erweiterung stellt ein paralleles *Fully Connected Network* dar, welches in jeder Schicht Informationen mit der korrespondierenden Schicht des U-Nets austauscht.

Für den Informationsaustausch werden zwei Änderungen an den jeweiligen Schichten des U-Nets vorgenommen. Zum einen werden vor der Instanznormalisierung die Durchschnittswerte aller Kanäle extrahiert und an den Merkmalsvektor vorausgehender Schichten konkateniert. Zum anderen werden die zur aktuellen Schicht korrespondierenden globalen Merkmalsvektoren an jedes Pixel der jeweiligen U-Net-Schicht durch eine Addition angefügt.

Abbildung 3.9 stellt die Struktur des globalen Merkmalsnetzwerks als auch die angepasste Struktur der U-Net-Schichten in zwei aufeinanderfolgenden Encoderkompo-

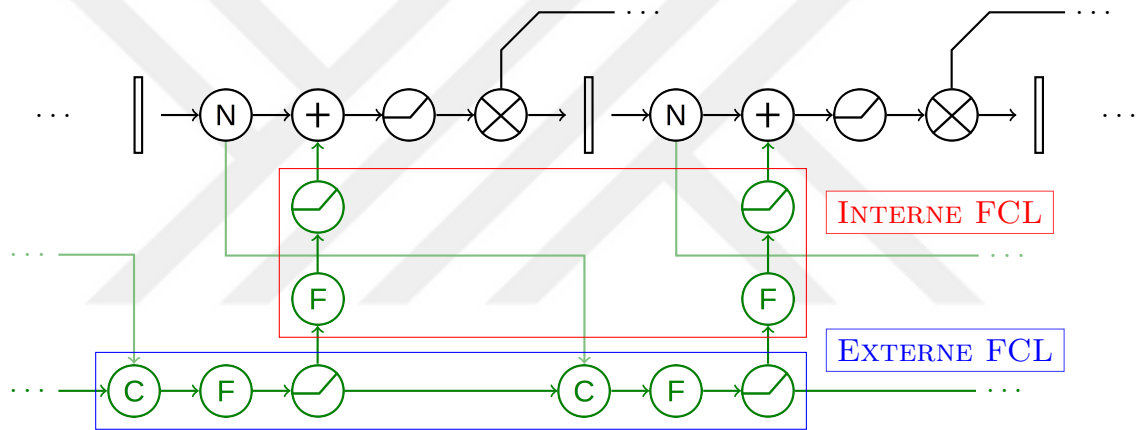


Abbildung 3.9: Detaillierte Struktur zweier aufeinanderfolgender Encoderschichten mit parallelem globalem Merkmalsnetzwerk. Aufbauend auf der Architektur des U-Nets werden bei der Normalisierung (N) die Durchschnittswerte extrahiert und an das parallele Netzwerk mittels einer Konkatenation (C) an die vorausgehenden Informationen angeknüpft. Diese konkatenierten Vektoren dienen als Eingabe für ein Fully Connected Layer (F), welches anschließend eine Aktivierungsschicht (SELU) durchläuft. Die Ergebnisse werden einerseits an die nächste Schicht des globalen Merkmalsnetzwerks konkateniert (C), andererseits durchlaufen sie einen weiteren Fully Connected Layer (F) und eine Aktivierungsfunktion, um anschließend auf jedes Pixel jeden Kanals der nächsten U-Net-Schicht addiert (+) werden zu können. Auf diese Weise werden in jedem Schritt globale Informationen zwischen dem U-Net und dem parallelen globalen Merkmalsnetzwerk ausgetauscht [11]. Im Rahmen dieser Arbeit werden die für die Injektion der globalen Informationen genutzten Fully Connected Layer als interne FCLs (rot) bezeichnet; analog dazu werden die für die Extraktion und Weiterleitung genutzten Fully Connected Layer externe FCLs (blau) genannt.

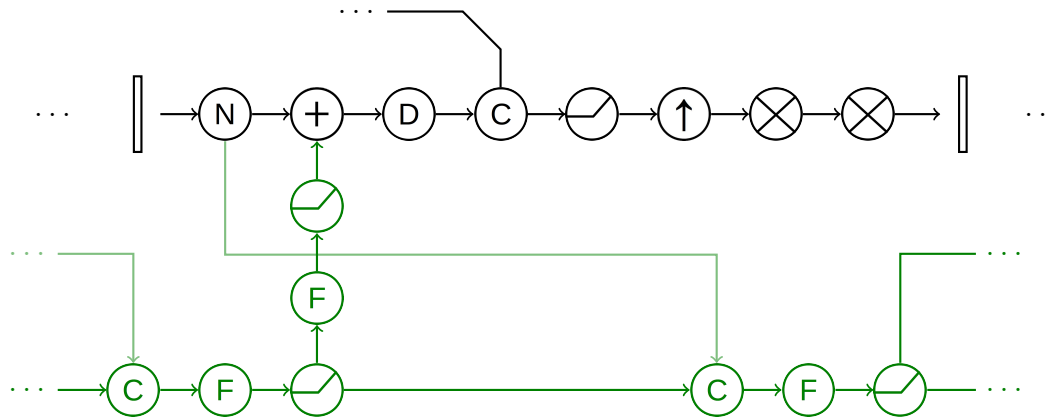


Abbildung 3.10: Detaillierte Struktur einer Decoderschicht mit parallelem globalem Merkmalsnetzwerk. Ähnlich der Encoderschicht werden während der Normalisierung (N) Durchschnittswerte extrahiert, an das parallele globale Merkmalsnetzwerk konkateniert (C), welches die Informationen durch ein Fully Connected Network (F) verarbeitet und auf den Wert jedes Pixels auf jedem Kanal der nächsten U-Net-Schicht addiert (+) [11].

nenten schematisch dar. Globale Informationen aus den vorausgehenden Schichten des U-Nets werden hierbei mittels einer Konkatenation dem globalen Merkmalsnetzwerk hinzugefügt. Ein Fully Connected Layer transformiert diese globalen Informationen, die daraufhin eine durch die SELU-Funktion dargestellte Aktivierungsschicht durchlaufen; dieses Fully Connected Layer wird im Rahmen dieser Arbeit als externes Layer bezeichnet. Von dort aus werden die Informationen mittels einer weiteren Fully-Connected-Transformation gefolgt von einer weiteren SELU-Aktivierung an jedes Pixel jeden Kanals der entsprechenden U-Net-Schicht angefügt. Diese Schicht wird als interner Fully Connected Layer bezeichnet. Zudem werden die Informationen an die nächste Schicht des globalen Merkmalsnetzwerk weitergeleitet. Nach der Weiterleitung werden die globalen Informationen der korrespondierenden U-Net-Schicht durch eine Konkatenation in das globale Merkmalsnetzwerk eingeführt und der Prozess für die nächste Schicht wiederholt.

Abbildung 3.10 stellt die angepasste Struktur der Decoderkomponente dar. Die Anpassungen sind hierbei Analog zu denen der Encoderkomponenten. Dabei werden auch zunächst globale Merkmalsvektoren vor der Normalisierung extrahiert und für die nächste Schicht mittels des globalen Merkmalsnetzwerks vorverarbeitet. Anschließend werden die globalen Informationen der vorausgehenden Schichten auch hier mittels einer Addition in das U-Net eingeführt. Dies geschieht noch vor einer möglichen Dropout-Schicht, die auch hier nur in den ersten drei Decoderschichten vorliegt.

Komponente	Schicht	Internes Format	Externes Format
Encoder	1	—	3×128
	2	128×128	256×256
	3	256×256	512×512
	4	512×512	1024×512
	5	512×512	1024×512
	6	512×512	1024×512
	7	512×512	1024×512
	8	512×512	1024×512
Decoder	1	512×512	1024×512
	2	512×512	1024×512
	3	512×512	1024×512
	4	512×512	1024×512
	5	512×256	1024×256
	6	512×128	384×128
	7	128×64	192×64
	8	64×10	—

Tabelle 3.1: Dimensionen der externen und internen Fully Connected Layer des globalen Merkmalsnetzwerks. Die Werte wurden dabei dem Zusatzmaterial der Arbeit von Deschaintre et al. [11] entnommen. Das Format 3×128 beschreibt beispielsweise ein Fully Connected Layer mit drei Eingabe- und 128 Ausgabeneuronen.

Ein wichtiger Aspekt bei den Fully Connected Layern besteht darin, dass diese den Kanälen der jeweiligen U-Net-Schichten entsprechen müssen. Tabelle 3.1 stellt die jeweiligen Formate der internen und externen Schichten dar. Das Format beschreibt dabei die Anzahl der Neuronen in der Eingabe- und Ausgabeschicht, die zusammen einen Fully Connected Layer darstellen. Ausnahmen bezüglich des Formats bilden hierbei die erste Schicht des Encoders und die letzte Schicht des Decoders. Die erste Schicht besitzt kein internes Fully Connected Layer, da noch keine vorausgehenden Merkmalsvektoren existieren. Auch existiert kein externes Fully Connected Layer für die letzte Schicht, da diese keine Merkmalsvektoren an weitere Schichten weiterleiten muss. Vor dem Anwenden der Sigmoid-Funktion in der letzten Schicht wird hier ein letztes mal eine Injektion globaler Informationen aus dem parallelen Fully Connected Network in das U-Net vorgenommen. Ein weiterer Spezialfall muss in der letzten Schicht des Encoders bzw. ersten Schicht des Decoders behandelt werden. Hierbei wird nach der Injektion globaler Informationen in das U-Net direkt eine Extraktion der Durchschnittswerte des resultierenden Bildes vorgenommen, da dort keine weitere Normalisierungsschicht vorliegt.

Die durch das globale Merkmalsnetzwerk dargestellte Erweiterung ist imstande,

Flecken auf den resultierenden Materialkarten beinahe restlos zu entfernen. Auch im Rahmen des anfangs erwähnten Experiments werden wesentlich bessere Ergebnisse erzielt (siehe Abbildung 3.7) [11]. Im Abschnitt 3.8.2 wird ein sowohl qualitativer als auch quantitativer Vergleich der vorausgehenden und im Rahmen dieses Abschnittes vorgestellten Architekturen vorgenommen.

3.3.3 Datenanpassungen

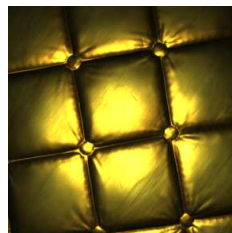
In diesem Abschnitt werden Vorbereitungen der Daten für eine Eingabe an das Deep Neural Network und Nachbearbeitungen der Ausgabe für ein mögliches Rendering vorgestellt.

Die Eingabe und erwünschte Ausgabe des Deep Neural Networks stellen ein Datenpaar dar. Dieses besteht aus einem gerenderten Blitzlichtfoto $\mathbf{I}$ eines planaren Oberflächenmaterials, welches in das Netz eingegeben wird und der zugrunde liegenden tatsächlichen SVBRDF dieses Materials, die durch einen Satz an Materialkarten $\mathbf{M}$ dargestellt wird; Letzteres stellt die zu erzielende Ausgabe dar. Ein solches Datenpaar wird in Abbildung 3.1 abgebildet.

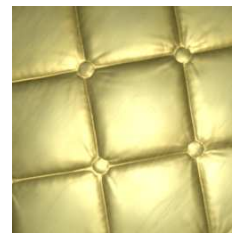
Das gerenderte Eingabebild liegt im niedrigen Dynamikumfang vor. Trotzdem kann das synthetische Blitzlichtfoto einen zu großen Wertebereich spannen; deswegen wird vor der ersten Faltung des Encoders eine Glättung des Eingabebildes mittels

$$\mathbf{I}' = \frac{\log(\mathbf{I} + \varepsilon \mathbf{J}_{m,n}) - \log(\varepsilon \mathbf{J}_{m,n})}{\log(1 + \varepsilon) - \log(\varepsilon)} \quad (3.2)$$

vorgenommen, was den Dynamikumfang des Bildes abflacht und das Bild in den logarithmischen Raum bringt, wobei hier für ε der Wert 0.01 gewählt wird [11]; die Funktion $\log$ wird dabei komponentenweise auf die Matrizen angewandt. Die



(a) Synthetisches Blitzlichtfoto



(b) Ergebnis der Glättung

Abbildung 3.11: Glättung eines synthetischen Blitzlichtfotos [11]. Abbildung (a) stellt ein synthetisches Blitzlichtfoto und Abbildung (b) das Ergebnis der Glättung dieses Fotos dar.

Matrix $\mathbf{I}$ besitzt hierbei die Dimensionen $m \times n$ und entspricht somit der Größe der Matrix $\mathbf{J}_{m,n}$, bei der jeder Eintrag den Wert eins besitzt. Abbildung 3.11 stellt eine solche Glättung eines synthetischen Blitzlichtfotos anhand der Gleichung (3.2) exemplarisch dar.

Ausgehend von dem geglätteten Blitzlichtfoto erstellt das Deep Neural Network eine Schätzung für die SVBRDF des eingegebenen Materials. Der Satz an geschätzten Materialkarten besitzt für jedes Pixel an der Position (i, j) einen Vektor $\mathbf{m}_{i,j}^{(n)'}$, der die Normale der Oberfläche an dieser Pixelposition festhält. Hierbei ist nur die vom Netz geschätzte Richtung des Vektors von Relevanz, nicht jedoch der Betrag. Deswegen wird die Normale jedes Pixels zusätzlich mittels

$$\mathbf{m}_{i,j}^{(n)} = \frac{\mathbf{m}_{i,j}^{(n)'} - 0.5 \mathbf{j}_3}{2 \|\mathbf{m}_{i,j}^{(n)'} - 0.5 \mathbf{j}_3\|_2} + 0.5 \mathbf{j}_3 \quad (3.3)$$

im Rahmen der encodierten Form normalisiert. Da der Renderer in jedem Fall eine zusätzliche Normalisierung der eingegebenen Vektoren vornimmt, hat dieser Prozess auf das gerenderte Bild keinen Einfluss. Stattdessen wird mittels dieses Schrittes ein quantitativer Direktvergleich mit der tatsächlichen Normalenkarte ermöglicht. Abbildung 3.12 bildet sowohl die initiale als auch die mittels dieser Formel renormalisierte Schätzung ab. Zusätzlich wird die absolute Differenz jedes Farbkanals der geschätzten Normalenkarten mit der tatsächlichen Normalenkarte dargestellt. In diesen Abbildungen wird deutlich, dass ein direkter Vergleich mit der initialen

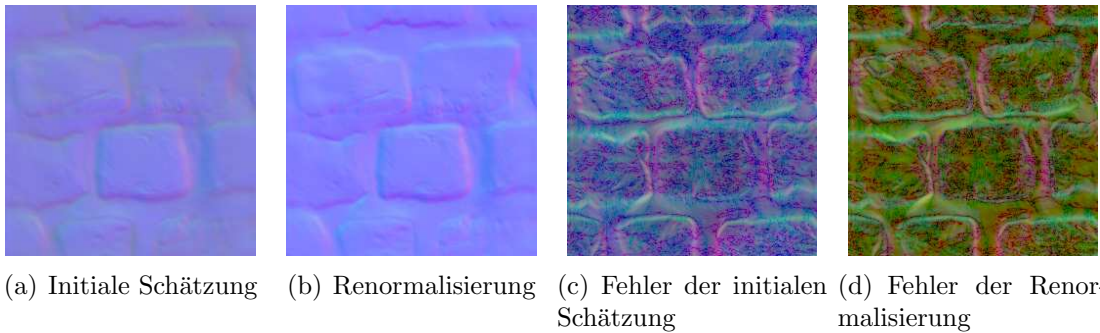


Abbildung 3.12: Renormalisierung der geschätzten Normalenkarte. Abbildung (a) stellt die initiale Schätzung des Deep Neural Networks und Abbildung (b) die normalisierte Version dieser Schätzung dar. Die Abbildungen (c) und (d) zeigen die absolute Differenz jedes Farbkanals zwischen der tatsächlichen Normalenkarte und der initial geschätzten bzw. der renormalisierten Normalenkarte in dieser Reihenfolge. Die Farbwerte dieser Fehlerbilder wurden für eine bessere Visualisierung mit einem Exponenten von 0.25 versehen.

Schätzung zu verfälschten Fehlerwerten führen würde und die Renormalisierung somit für eine korrekte Fehlerberechnung notwendig wird.

3.3.4 Training

Die Intention des Trainings des Deep Neural Networks besteht darin, einen Performanzwert P zu optimieren [19]. Im Rahmen dieses Kapitels bezieht sich der Performanzwert auf die anfangs vorgestellte Formel (3.1) und somit auf die Approximation einer idealen inversen Renderingfunktion, wie sie im Abschnitt 3.1 eingeführt wurde. Dieser Performanzwert wird nur indirekt optimiert; stattdessen wird ein Satz an Parametern θ für das Deep Neural Network gesucht, welcher eine Kostenfunktion $J(\theta)$ minimiert. Eine solche Kostenfunktion kann durch

$$J(\theta) = \mathbb{E}_{(\mathbf{I}, \mathbf{M}) \sim \hat{p}_{\text{data}}} L(f(\mathbf{I}; \theta), \mathbf{M}) \quad (3.4)$$

formalisiert werden, wobei L eine Fehlerfunktion, $f(\mathbf{I}; \theta)$ die approximierte Ausgabe der Deep-Learning-gestützten inversen Renderingfunktion ausgehend von dem Eingabebild $\mathbf{I}$ und den zu optimierenden Parametern θ darstellt und $\hat{p}_{\text{data}}$ eine empirische Verteilung symbolisiert. Da es sich hierbei um ein beaufsichtigtes Lernen, dem sogenannten *Supervised Learning* handelt, wird der Fehler zusätzlich über die tatsächliche Ausgabe $\mathbf{M}$ gebildet [19]; in diesem Kapitel wird so eine Ausgabe durch einen Satz tatsächlicher Materialkarten für das Eingabebild $\mathbf{I}$ eines planaren Oberflächenmaterials dargestellt.

Das Training des Deep Neural Networks findet durch die Minimierung des empirischen Risikos

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{I}^{(i)}; \theta), \mathbf{M}^{(i)}) \quad (3.5)$$

statt, wobei m die Anzahl der Trainingsdaten darstellt. Hierbei wird die tatsächliche Verteilung, die im Kontext dieser Anwendung alle möglichen Oberflächenmaterialien unter allen möglichen Beleuchtungsbedingungen beinhaltet, durch eine empirische Verteilung ersetzt, die durch das Trainingsdatenset dargestellt wird. Die Minimierung des empirischen Risikos besitzt dabei die Intention, das tatsächliche Risiko zu minimieren [19]. Ein solcher empirischer Ansatz ist für das Problem des *Overfittings* anfällig, bei dem das Modell den Trainingsdatensatz auswendig lernt. Durch den Einsatz eines großen Trainingsdatensatzes, wie es im Rahmen dieses Kapitels der Fall ist, kann ein solches Overfitting umgangen werden [19].

Eine Reihe von Optimierungsalgorithmen benötigen den Gradienten $\nabla_{\theta} J(\theta)$ über den

Trainingsdatensatz [19], so auch der im Rahmen dieses Kapitels eingesetzte Adam-Algorithmus [31]. Für die Berechnung solcher Gradienten werden die folgenden zwei Aspekte berücksichtigt. Zum einen ist eine durch den Gradienten dargestellte Eigenschaft über das gesamte Trainingsdatenset aufwendig zu berechnen. Dies wird durch das Aufteilen des Trainingsdatensets in sogenannte *Minibatches* umgangen; eine solche Methode wird auch als stochastische Methode bezeichnet [19]. Ein Batch im Rahmen dieses Kapitels beschreibt somit eine Menge an Datenpaaren der Form $(\mathbf{I}, \mathbf{M})$, wobei $\mathbf{I}$ ein Blitzlichtfoto und $\mathbf{M}$ den dazu korrespondierenden Satz an tatsächlichen Materialkarten darstellt. Der Gradient wird somit jeweils bezüglich eines Batches ausgerechnet. Zum anderen wird eine Fehlerfunktion benötigt, die einen solchen Gradienten für den Einsatz des genutzten Optimierungsalgorithmus bilden kann. Ein solches differenzierbares Fehlermaß wird im nächsten Abschnitt vorgestellt.

Der hierbei eingesetzte Adam-Optimierungsalgorithmus stellt eine effiziente adaptive Variante einer stochastischen Optimierungsmethode basierend auf den Gradienten bezüglich der Parameter θ des Deep Neural Networks dar [31]. Im nächsten Abschnitt wird ein stochastisches Rendering-gestütztes Fehlermaß vorgestellt, welches ein Training für gradientenbasierte Optimierungsmethoden, wie den hier eingesetzten Adam-Algorithmus [31], ermöglicht.

3.4 Rendering-gestütztes Fehlermaß

In diesem Abschnitt wird das Rendering-gestützte Fehlermaß vorgestellt, welches eine Bewertung der geschätzten SVBRDF ermöglicht und für das Training des in den letzten Abschnitten vorgestellten Deep Neural Networks eingesetzt werden kann. Ein direkter Vergleich zwischen den geschätzten und den tatsächlichen Materialkarten stellt keinen direkten Zusammenhang zwischen dem dadurch ermittelten Fehler und dem gerenderten Erscheinen des Materials her. Dies hat den Hintergrund, dass numerische Differenzen in den Materialkarten nicht konsistent mit dem Erscheinen der darauf basierenden Renderings korrelieren, da die Interaktionen der jeweiligen Variablen in den Materialkarten untereinander nicht berücksichtigt werden [11].

Ein Vergleich zwischen den Renderings der geschätzten und tatsächlichen Materialkarten, die unter gleichen Beleuchtungsvoraussetzungen synthetisiert werden, ermöglicht eine bessere Bewertung der Ergebnisse [11]. Abbildung 3.13 stellt die Idee eines solchen Rendering-gestützten Fehlermaßes schematisch dar.

Die in dieser Arbeit eingesetzte Trainingsmethode (siehe Abschnitt 3.3.4) erfordert

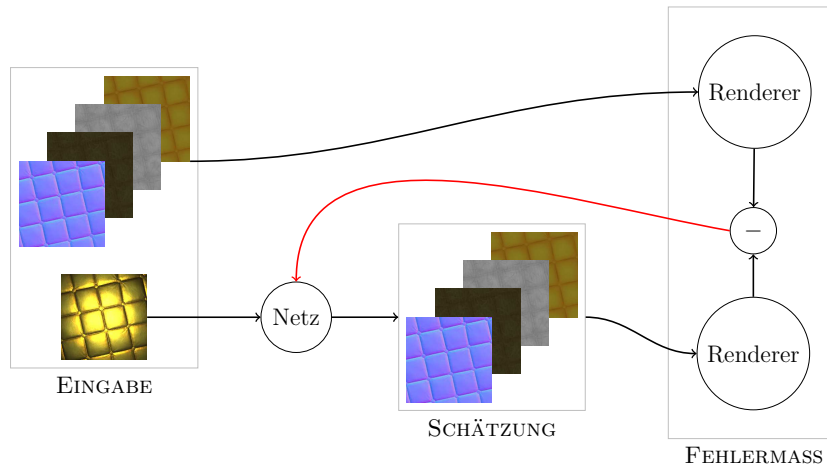


Abbildung 3.13: Der Renderingfehler. Das Eingabedatum wird durch ein Paar an Materialkarten und einem darauf basierend synthetisiertem Blitzlichtfoto dargestellt [11]. Ein Deep Neural Network bekommt das gerenderte Blitzlichtfoto als Eingabe und gibt eine SVBRDF-Schätzung als Ausgabe aus. Die tatsächlichen Materialkarten und die Schätzung werden beide unter denselben Bedingungen mittels einer Reihe an Licht- und Kamerapositionen mehrfach gerendert und anschließend mit einem pixelweisen Fehlermaß verglichen. Dies bildet im Durchschnitt den Fehler, der an das Netz zurück propagiert wird (roter Pfeil).

die Berechenbarkeit von Gradienten für die Parameter des Deep Neural Networks bezüglich des Fehlers, der zwischen der geschätzten Ausgabe und den tatsächlichen Materialkarten gebildet wird [31]. Folglich wird eine differenzierbare Renderingkomponente benötigt, die in der Lage ist, physikalisch-basierte Renderings anhand der Materialkarten anzufertigen und einen Fehler zwischen diesen Renderings an die entsprechenden Materialkarten zurück zu propagieren. Eine solche Komponente wird in Abschnitt 3.4.1 vorgestellt. Die geschätzten und tatsächlichen Materialkarten müssen bei einem gegebenen Vergleich unter denselben Beleuchtungsbedingungen gerendert werden, da sonst ein direkter Vergleich der Renderings nicht möglich wäre. Eine Menge fester Betrachtungs- und Beleuchtungswinkel würde das Fehlermaß jedoch fehleranfällig machen, da nicht alle Winkel berücksichtigt werden. Stattdessen wird ein stochastischer Ansatz verfolgt, der eine Auswertung über alle möglichen Winkel durchführt und das Problem somit umgeht. Das stochastische Verfahren wird in Abschnitt 3.4.2 vorgestellt. Abschließend wird in Abschnitt 3.4.3 ein Regularisierungsverfahren für das finale Rendering-gestützte Fehlermaß vorgestellt.

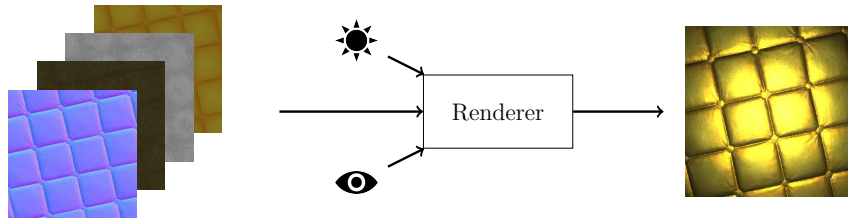


Abbildung 3.14: Renderer für planare Oberflächen. Basierend auf den Materialinformationen (links) [11] und einer zusätzlichen Lichtquelle sowie eines festpositionierten Augenpunktes evaluiert die differenzierbare Renderingkomponente jedes Pixel, sodass das beleuchtete Ergebnis (rechts) basierend auf dem Cook-Torrance-Beleuchtungsmodell [8] resultiert.

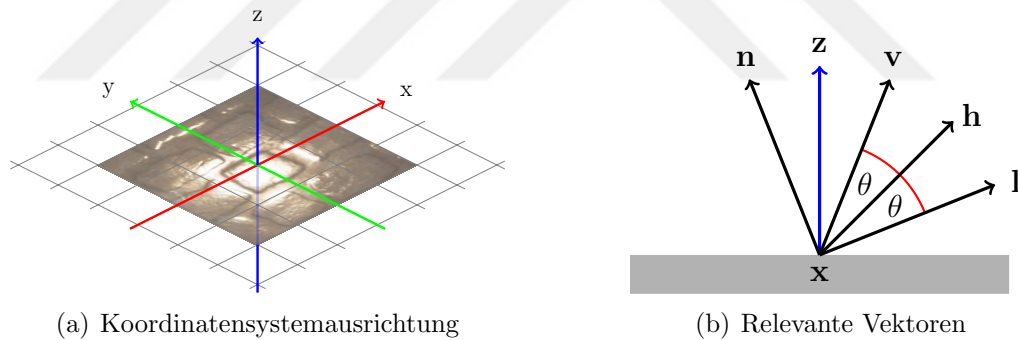


Abbildung 3.15: Koordinatensystem und relevante Vektoren der planaren Oberfläche. Abbildung (a) stellt die Koordinatensystemausrichtung und Platzierung der planaren Oberflächen im dreidimensionalen Raum dar. Das planare Material liegt auf der xy -Ebene und besitzt somit für jeden Punkt der Oberfläche einen z -Wert von null. Abbildung (b) skizziert relevante Vektoren für einen Punkt $\mathbf{x}$ der Oberfläche, die für die Beleuchtungsberechnung benötigt werden. Die Normale $\mathbf{n}$ wird durch die Normalenkarte bestimmt und kann vom Einheitsvektor $\mathbf{z}$, der in die z -Richtung zeigt, abweichen. Der Halbvektor $\mathbf{h}$ liegt zwischen der Augenrichtung $\mathbf{v}$ und der Richtung zur Lichtquelle $\mathbf{l}$ [8].

3.4.1 Differenzierbare Renderingkomponente

In diesem Abschnitt wird die für das Rendering-gestützte Fehlermaß benötigte differenzierbare Renderingkomponente vorgestellt. Diese ist instand, physikalisch-basierte Renderings basierend auf den SVBRDF-Schätzungen des erweiterten Deep Neural Networks (siehe Abschnitt 3.3.2) zu synthetisieren. Ein Fehler, der auf diesen Renderings berechnet wird, kann dabei aufgrund der Differenzierbarkeit dieser Komponente an das Netz zurück propagiert werden; dies ermöglicht ein Training des Deep Neural Networks basierend auf dem Fehler der Renderings dieser Komponente. Die differenzierbare Renderingkomponente kann als ein Reflexionsevaluierungsnetzwerk verstanden werden, das im Rahmen eines Deep Neural Networks ein Subnetzwerk darstellt und die reflektierte Strahlungsdichte L in Richtung des Betrachters berechnet [81]. Das umgesetzte Beleuchtungsmodell bezieht sich auf das Cook-Torrance-Modell (siehe Abschnitt 2.1.3.2).

Die Eingabe für die Synthese eines physikalisch-basierten Renderings wird neben den Materialkarten durch eine Menge an Punktlichtquellen und einem Augenpunkt dargestellt. Die Ausgabe ist ein beleuchtetes Bild, welches das Resultat einer pixelweisen Beleuchtungsevaluierung darstellt (siehe Abbildung 3.14).

Das Ausgabebild besitzt dieselbe Größe wie die Materialkarten; die Größe beläuft sich in dieser Arbeit auf 256×256 Pixel. Für jedes Pixel der Ausgabe halten die korrespondierenden Pixel in den Materialkarten der Eingabe alle benötigten Materialinformationen fest. Diese beziehen sich pro Pixel auf die freien Parameter des Cook-Torrance-Modells (siehe Abschnitt 2.1.3.2) und werden durch die Normale $\mathbf{n}$, die diffuse Grundfarbe $\mathbf{d}$, den skalaren Rauheitswert r und durch die Glanzfarbe $\mathbf{s}$ dargestellt. Durch die Hinzunahme der Pixelpositionen, der Menge von Lichtquellen der Szene und der Position des Augenpunktes kann die Strahlungsdichte jedes Pixels unabhängig voneinander evaluiert werden. Speziell müssen an dieser Stelle keine geometrischen Berechnungen – beispielsweise dargestellt durch eine Kameraprojektion – für das Rendering durchgeführt werden, da die Materialkarten zusammen mit den pixelweisen Positionsinformationen als geometrische Puffer (G-Buffer) verstanden werden können, die alle relevanten Informationen für eine entsprechende Beleuchtung des Bildes beinhalten [11, 65].

Die Lichtquellen und Augenpunkte werden im dreidimensionalen Raum positioniert. Folglich muss für eine Beleuchtungsberechnung die zweidimensionale planare Oberfläche des zu beleuchtenden Materials ebenfalls in diesem dreidimensionalen Raum positioniert werden, sodass eine Lichtquellenrichtung und Augenpositionsrichtung korrespondierend zu jedem Pixel der Oberfläche bestimmt werden kann. Abbildung

3.15(a) stellt die Positionierung der planaren Oberfläche im dreidimensionalen Raum schematisch dar. Die planare Oberfläche wird hier auf die xy -Ebene mit $x, y \in [-1, 1]$ und $z = 0$ positioniert. So lässt sich für ein gegebenes Pixel an der Position (i, j) ein korrespondierender Oberflächenpunkt $\mathbf{x} \in \mathbb{R}^3$ bestimmen; darauf basierend können die für die Beleuchtungsberechnung relevanten Vektoren ermittelt werden (siehe Abbildung 3.15(b)).

Für eine stetige Beleuchtungsumgebung berechnet sich die reflektierte Strahlungsdichte $L(\mathbf{x}, \mathbf{x}'')$ entsprechend der Rendering-Gleichung [25], wobei $\mathbf{x}$ den Augenpunkt und $\mathbf{x}'$ den betrachteten Oberflächenpunkt darstellt (siehe Abschnitt 2.1.4.1). In der Anwendung wird jedoch über eine Menge von Punktlichtquellen diskretisiert [81]. Hierbei kann die lokale Beleuchtungsberechnung eingesetzt werden (siehe Abschnitt 2.1.4.2). Die pixelweise Strahlungsdichte des Ausgabebildes lässt sich durch

$$\mathbf{i}_{i,j} = \sum_{k=0}^l f_r(i, j, \mathbf{n}, \mathbf{l}_k, \mathbf{v}) \mathbf{c}_k^{\mathbf{x}} \max(0, \mathbf{n} \cdot \mathbf{l}_k) \quad (3.6)$$

berechnen [81], wobei $\mathbf{i}_{i,j} \in \mathbb{R}^3$ die RGB-Farbe des Ausgabepixels an der Position (i, j) festhält. Die reflektierte Strahlungsdichte berechnet sich über die Summe der Beiträge der l verschiedenen Punktlichtquellen. Hierbei wird die Eigenschaft der Punktlichtquellen ausgenutzt; wird der Term für die Berechnung der eingehenden Strahlungsdichte einer Punktlichtquelle in die Reflexionsgleichung substituiert, so reduziert sich das Integral auf ein Produkt (siehe Abschnitt 2.1.4.2). Die gesamte reflektierte Strahlungsdichte aller Punktlichtquellen lässt sich somit über die Summe der einzelnen Beiträge berechnen [7]. Bezüglich des zum Pixel an der Position (i, j) korrespondierenden Oberflächenpunktes $\mathbf{x} \in \mathbb{R}^3$ stellt $\mathbf{l}_k$ die k -te Lichtquellenrichtung, $\mathbf{v}$ die Richtung zum Augenpunkt und $\mathbf{n}$ die Normale dar. Die Funktion f_r repräsentiert die SVBRDF in Abhängigkeit von der Pixelposition (i, j) . Das Produkt $\mathbf{c}_k^{\mathbf{x}} \cdot \max(0, \mathbf{n} \cdot \mathbf{l}_k)$ stellt den verbleibenden Term der Substitution dar, wobei $\mathbf{c}_k^{\mathbf{x}}$ die in Abhängigkeit zum Oberflächenpunkt $\mathbf{x}$ relevanten Lichteigenschaften wie den Strahlungsfluss Φ und die Abdämpfung durch das Abstandsgesetz beinhaltet [7], und $\max(0, \mathbf{n} \cdot \mathbf{l}_k)$ die Anwendung des Kosinusetzes darstellt [58]. Die SVBRDF f_r wird durch einen Satz an Materialkarten in Assoziation mit einem Beleuchtungsmodell definiert. Im Rahmen dieses Kapitels wird das physikalisch-basierte Cook-Torrance-Modell [8] eingesetzt. Diese Entscheidung beruht auf zwei Aspekten. Zum einen ist der eingesetzte Datensatz für dieses Beleuchtungsmodell ausgelegt. Zum anderen sollen neben den synthetischen Renderings auch Fotos realer Materialien in die inverse Renderingfunktion eingegeben werden können; der physikalisch-basierte

Aspekt des ausgewählten Beleuchtungsverfahrens hat im Gegensatz zu empirischen Modellen den Vorteil, dass dieser auf physischen und optisch korrekten Annahmen basiert [21].

Sei $\mathbf{m}_{i,j}$ der Satz an Materialinformationen für das Pixel an der Position (i, j) und SHADE die Cook-Torrance-Beleuchtungsfunktion, so lässt sich die vorausgehende Gleichung entsprechend

$$\mathbf{i}_{i,j} = \sum_{k=0}^l \text{SHADE}(\mathbf{n}, \mathbf{d}, r, \mathbf{s}, \mathbf{l}_k, \mathbf{v}) \mathbf{c}_k^{\mathbf{x}} \max(0, \mathbf{n} \cdot \mathbf{l}_k) \quad (3.7)$$

umformulieren, wobei die Normale $\mathbf{n}$ durch $\mathbf{m}_{i,j}^{(n)}$, die diffuse RGB-Grundfarbe $\mathbf{d} \in \mathbb{R}^3$ durch $\mathbf{m}_{i,j}^{(d)}$, der skalare Rauheitswert $r \in \mathbb{R}$ durch $m_{i,j}^{(r)}$ und die Glanzfarbe $\mathbf{s} \in \mathbb{R}^3$ durch $\mathbf{m}_{i,j}^{(s)}$ aus den entsprechenden Materialkarten an der entsprechenden Stelle gegeben ist.

```

procedure SHADE( $\mathbf{n}, \mathbf{d}, r, \mathbf{s}, \mathbf{l}, \mathbf{v}$ )
   $\mathbf{h} \leftarrow \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}$ 
   $\mathbf{d} \leftarrow \frac{\mathbf{d}(1-\mathbf{s})}{\pi}$ 
   $D \leftarrow \left[ \frac{1}{\pi} \frac{r^2}{(\mathbf{n} \cdot \mathbf{h})^2 (r^4 - 1) + 1 + \varepsilon} \right]^2$ 
   $G \leftarrow \frac{1}{(\mathbf{n} \cdot \mathbf{l})(1 - \frac{r^2}{2}) + \frac{r^2}{2} + \varepsilon} \cdot \frac{1}{(\mathbf{n} \cdot \mathbf{v})(1 - \frac{r^2}{2}) + \frac{r^2}{2} + \varepsilon}$ 
   $\mathbf{F} \leftarrow \mathbf{s} + (1 - \mathbf{s}) 2^{((-5.55473(\mathbf{v} \cdot \mathbf{h}) - 6.98316)(\mathbf{v} \cdot \mathbf{h}))}$ 
  return  $(\frac{\mathbf{F}GD}{4} + \mathbf{d})$ 
end procedure

```

Algorithmus 3.1: Der Cook-Torrance-Shader [11]. Der Shader basiert auf dem Cook-Torrance-Modell [8]. Der Wert $\varepsilon > 0$ wurde für die numerische Konsistenz eingeführt, die eine Differenzierbarkeit des Beleuchtungsmodells garantieren soll.

Der Algorithmus 3.1 bildet das eingesetzte Beleuchtungsprogramm, den sogenannten *Shader*, [11] ab. Dieses wird für jedes Pixel unabhängig von den Ausführungen der anderen Pixel durchlaufen, um den finalen Farbwert zu berechnen. Der Algorithmus basiert dabei auf dem Cook-Torrance-Beleuchtungsverfahren [8], welches im Abschnitt 2.1.3.2 detailliert vorgestellt wurde. Hierbei wird zunächst der Halbvektor $\mathbf{h}$ berechnet, der einen normalisierten Vektor darstellt, der genau zwischen der Augenpunkttrichtung $\mathbf{v}$ und der Lichtquellenrichtung $\mathbf{l}$ liegt (siehe Abbildung 3.15(b)). Die Variablen D , G und $\mathbf{F}$ stehen für die Mikrofacettensteigungsverteilungsfunktion, den geometrischen Dämpfungsterm und den Fresnel-Faktor in dieser Reihenfolge und belaufen sich auf die Ergebnisse der im Abschnitt 2.1.3.2 vorgestellten Formeln. Der

Fresnel-Faktor besitzt im Gegensatz zu D und G für jeden Farbkanal einen eigenen Koeffizienten. Der ausgegebene Farbwert setzt sich aus der berechneten Glanzfarbe und dem diffusen Anteil zusammen.

Abschließend wird die Differenzierbarkeit der Komponente erläutert. Wie in Abschnitt 3.3.4 bereits erwähnt wurde, werden für das Training des Deep Neural Networks im Rahmen dieser Arbeit die Gradienten für die Parameter θ des Netzes benötigt [19]. Für die differenzierbare Renderingkomponente sind die Gradienten bezüglich der Szeneninformationen, welche hier für ein Pixel $\mathbf{i}_{i,j}$ durch die entsprechenden Materialinformationen $\mathbf{m}_{i,j}$ dargestellt werden, von Interesse [37]. Sei $f(\mathbf{m}_{i,j}, \mathbf{l}, \mathbf{c}, \mathbf{v})$ die zusammengefasste Beleuchtungsberechnungsfunktion innerhalb der Summe der Gleichung (3.7), so wird eine Differenzierbarkeit bezüglich $\mathbf{m}_{i,j}$ gefordert, sodass sich die Gradienten

$$\nabla \mathbf{i}_{i,j} = \nabla \sum_{k=0}^l f(\mathbf{m}_{i,j}, \mathbf{l}_k, \mathbf{c}_k, \mathbf{v}) \quad (3.8)$$

für das Pixel (i, j) bezüglich der Materialkarten berechnen lassen [37], wobei die Lichtquelleneigenschaften sowie die Lichtquellen- und Augenpunktpositionen nicht von Interesse sind.

Die geforderte Differenzierbarkeit wird mittels zweier Aspekte erreicht. Zum einen werden ausschließlich differenzierbare Operatoren eingesetzt, die relevante Gradienten nicht abschneiden. Insbesondere eine Begrenzungsfunktion – das sogenannte Clamping – auf den berechneten finalen Pixelwert darf nicht angewandt werden, um beispielsweise das Bild in den niedrigen Dynamikumfang (LDR) zu transformieren. Stattdessen müssen zwei Renderings im hohen Dynamikumfang (HDR) miteinander verglichen werden, da sonst Gradienten speziell an den Glanzstellen bzw. bei zu hellem Licht abgeschnitten werden. Der zweite Aspekt beläuft sich auf die numerische Konsistenz; hierbei darf keine Division durch null stattfinden, da sonst ungültige Gradienten im Laufe des Trainings entstehen können. Dies wird durch einen Wert $\varepsilon > 0$ garantiert, der an den relevanten Stellen im Algorithmus 3.1 eingesetzt wird.

3.4.2 Fehlerbestimmung mittels multiplen Renderings

In diesem Abschnitt wird das stochastische Fehlermaß vorgestellt, welches basierend auf der differenzierbaren Renderingkomponente des letzten Abschnitts den Fehler zwischen den geschätzten und tatsächlichen SVBRDFs mittels eines stochastischen Verfahrens bildet.

Der Vorteil eines Rendering-gestützten Fehlermaßes besteht darin, dass das beleuchtete Resultat basierend auf den geschätzten Materialkarten bewertet wird. Dies steht

im Gegensatz zu einem direkten Vergleich der geschätzten und tatsächlichen Materialkarten – beispielsweise basierend auf dem L1-Fehlermaß –, der keinen direkten Zusammenhang zwischen dem so ermittelten Fehler und dem beleuchteten Resultat herstellen kann [11].

Das stochastische Rendering-gestützte Fehlermaß fertigt mittels der differenzierbaren Renderingkomponente mehrere Renderings an. Diese entstehen unter einer Vielzahl verschiedener Positionierungen der Lichtquelle und des Augenpunktes, wobei ein zu vergleichendes Paar an geschätzter und tatsächlicher SVBRDF unter identischen Beleuchtungsbedingungen evaluiert werden muss, um einen Direktvergleich zwischen den Renderings zu ermöglichen [11].

procedure GENERATECOSINEVECTOR

$u_1 \leftarrow \text{Uniform}(0.001, 0.95)$

$u_2 \leftarrow \text{Uniform}(0, 1)$

$r \leftarrow \sqrt{u_1}$

$\phi \leftarrow 2\pi u_2$

$x \leftarrow r \cos(\phi)$

$y \leftarrow r \sin(\phi)$

$z \leftarrow \sqrt{1 - r^2}$

return $[x, y, z]$

end procedure

Algorithmus 3.2: Generierung eines Vektors aus der Kosinusverteilung. Der Algorithmus entstammt dem Zusatzmaterial der Arbeit von Deschaintre et al. [11]. Die Funktion $\text{Uniform}(a, b)$ generiert einen uniform verteilten zufälligen Wert aus dem Intervall $[a, b]$. u_2 ist für den xy-Winkel ϕ verantwortlich. u_1 sorgt implizit für den Winkel in die z-Richtung.

Eine fest gewählte Menge an endlich vielen Lichtquellen- und Augenpunktpositionen würde einen Großteil des Winkelraums ignorieren. Stattdessen wird ein stochastischer Ansatz verfolgt; für jedes Trainingsdatum wird ein zufälliger Winkel aus einer Kosinusverteilung gewählt. Der Algorithmus 3.2 gibt eine Methode an, einen Vektor aus einer solchen Kosinusverteilung zu generieren. Die Kosinusverteilung weist flacheren Winkeln eine geringere Wahrscheinlichkeit zu. Die Wahl der Verteilung hat den Hintergrund, dass flachere Winkel in Bildern aufgrund der perspektivischen Projektion seltener auftreten [11].

Obwohl diese Methode jegliche Winkel abdeckt, erschwert sie das Entstehen von Spiegelreflexionen, welche für das Auftreten des Glanzlichtes verantwortlich sind. Eine Spiegelreflexion beschreibt hierbei eine Konstellation, bei der der Eingangsvektor gespiegelt an der Normalen den Ausgangsvektor approximiert. Das Phänomen des Glanzlichtes beinhaltet jedoch essentielle Informationen über das vorliegende

Material. Aus diesem Grund werden Spiegelreflexionen in dem Fehlermaß speziell berücksichtigt [11].

Das resultierende Fehlermaß fertigt $n_S = 6$ Glanzlicht- und $n_D = 3$ diffuse Renderings an, die mittels des L1-Fehlermaßes verglichen werden; der durchschnittliche Fehler dieser Vergleiche bildet den finalen Fehler des stochastischen Fehlermaßes [11]. Hierbei wird der L1-Fehler zwischen den Logarithmen der resultierenden Bilder gebildet, da potentiell hohe dynamische Bereiche extreme Differenzen an Spiegelreflexionspunkten aufweisen könnten und an dieser Stelle der relative statt absolute Fehler von Relevanz ist [11]. Abbildung 3.16 stellt dabei vier solcher entstehenden Renderings im Rahmen der Fehlerberechnung exemplarisch dar, wobei das erste ein diffuses und die restlichen drei ein Glanzsetup darstellen.

Für die diffusen Renderings werden die Vektoren für die Lichtquellenposition $\mathbf{l}$ und die Augenpunktposition $\mathbf{v}$ ausgehend vom Mittelpunkt der planaren Oberfläche aus der Kosinusverteilung gewählt. Für die Glanzlichtrenderings wird lediglich eine Richtung für $\mathbf{l}$ entsprechend der Kosinusverteilung generiert. Diese wird am Vektor $\mathbf{z}$, der in die z-Richtung zeigt (siehe Abbildung 3.15(b)), gespiegelt um den Vektor $\mathbf{v}$ zu erzeugen. Die Richtungsvektoren $\mathbf{l}$ und $\mathbf{v}$ werden zusätzlich mit dem Wert $\exp(d_l)$ bzw. $\exp(d_v)$ multipliziert, der einen zufälligen Abstand zum Ursprung herstellt und so die Position der Lichtquelle bzw. des Augenpunktes festlegt. Die Werte d_l und d_v entstammen dabei jeweils aus der Normalverteilung mit $\mu = 0.5$ und $\sigma = 0.75$. Zum Vergleich beträgt die Größe der planaren Oberfläche in der Anwendung 2×2 Einheiten [11].

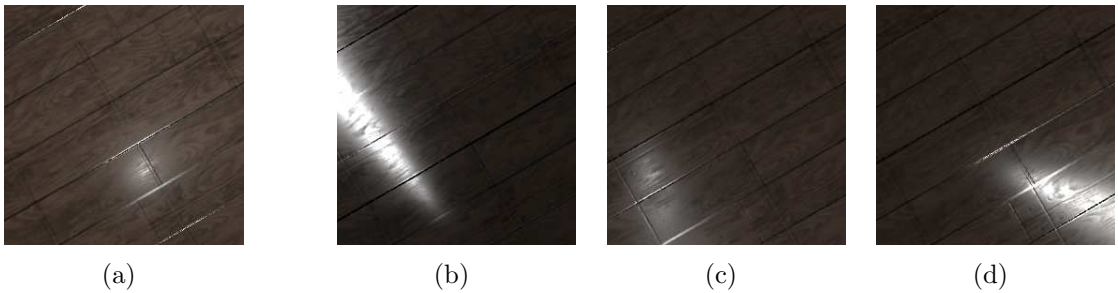


Abbildung 3.16: Beleuchtung eines Datums im Rahmen des Trainings in unterschiedlichen Konfigurationen für das Rendering-gestützte Fehlermaß. Abbildung (a) stellt eine diffuse Lichtkonfiguration dar. Die Abbildungen (b), (c) und (d) bilden Ergebnisse von Glanzlichtsetups ab. Die zugrundeliegende SVBRDF entstammt der Arbeit von Deschaintre et al. [11].

Sei

$$L_{\text{RenderDiffuse}} = \sum_{i=1}^{n_D} \delta(R(\mathbf{M}, \mathbf{l}_{D,i}, \mathbf{v}_{D,i}), R(\mathbf{M}', \mathbf{l}_{D,i}, \mathbf{v}_{D,i})) \quad (3.9)$$

der Fehler aller diffusen Renderings und

$$L_{\text{RenderSpecular}} = \sum_{i=1}^{n_S} \delta(R(\mathbf{M}, \mathbf{l}_{S,i}, \mathbf{v}_{S,i}), R(\mathbf{M}', \mathbf{l}_{S,i}, \mathbf{v}_{S,i})) \quad (3.10)$$

der Fehler aller Glanzlichtrenderings, wobei

$$\delta(\mathbf{x}, \mathbf{x}') = \|\log(\mathbf{x} + \varepsilon \mathbf{j}) - \log(\mathbf{x}' + \varepsilon \mathbf{j})\|_1 \quad (3.11)$$

gilt und für ε der Wert 0.01 gewählt wird, die Dimensionen von $\mathbf{j}$ denen von $\mathbf{x}$ entsprechen und jedes Element von $\mathbf{j}$ einen Wert gleich eins besitzt, so beläuft sich der finale Fehler auf

$$L_{\text{Render}} = \frac{L_{\text{RenderDiffuse}} + L_{\text{RenderSpecular}}}{n_D + n_S}, \quad (3.12)$$

wobei die Funktion $R(\mathbf{M}, \mathbf{l}, \mathbf{v})$ ein Rendering der Materialkarten $\mathbf{M}$ mit Lichtquellenposition $\mathbf{l}$ und Augenpunkt $\mathbf{v}$ erzeugt, $\mathbf{M}$ die tatsächlichen und $\mathbf{M}'$ die geschätzten Materialkarten darstellen und n_D und n_S die Anzahl der diffusen und Glanzlichtrenderings in dieser Reihenfolge angeben. Die Vektoren $\mathbf{l}_{D,i}$ und $\mathbf{v}_{D,i}$ geben die i -te Lichtquellen- und Augenpunktposition für ein diffuses Renderingpaar an; diese werden nach der vorausgehenden Beschreibung entsprechend der Kosinusverteilung generiert. Analog dazu stellen $\mathbf{l}_{S,i}$ und $\mathbf{v}_{S,i}$ die Lichtquellen- und Augenpunktpositionen für die jeweiligen Glanzlichtrenderings dar. Der summierte Fehler wird durch die Anzahl der angefertigten Renderings geteilt und bildet den finalen Fehler.

Da die Renderings mittels der differenzierbaren Renderingkomponente des letzten Abschnittes angefertigt werden, ist auch dieses stochastische Fehlermaß imstande, die relevanten Gradienten für das Training des Deep Neural Networks zu erzeugen und somit als Ganzes ebenfalls differenzierbar.

3.4.3 Regularisierung

Sei L_{Render} der durch das Rendering-gestützte Fehlermaß bestimmte Fehler und L_{Normal} , L_{Diffuse} , $L_{\text{Roughness}}$ und L_{Specular} der durchschnittliche L1-Fehler der Normalen, der diffusen Grundfarbe, dem skalaren Rauheitswert und der Glanzfarbe, so

kann mittels

$$L = L_{\text{Render}} + \lambda (L_{\text{Normal}} + L_{\text{Diffuse}} + L_{\text{Roughness}} + L_{\text{Specular}}) \quad (3.13)$$

ein regularisierter Fehler bestimmt werden, der imstande ist, ein darauf basiertes Training zu stabilisieren [12]. Der Fehler wird hierbei beispielsweise anhand der geschätzten und tatsächlichen Materialkarten bestimmt.

In der Arbeit von Deschaintre et al. [12] wird eine Gewichtung von $\lambda = 0.1$ gewählt. Alternativ können die Komponenten auch individuell gewichtet werden und statt des L1- das L2-Fehlermaß eingesetzt werden [39, 40]. Li et al. [39] regularisieren zudem zusätzlich mittels des Cross-Entropy-Fehlers, den sie mit einem Wert $\lambda_{CE} = 0.0005$ gewichtet auf den Fehler aufsummieren.

Eine ähnliche Form der Regularisierung wird im nächsten Kapitel eingesetzt; in diesem Kapitel dagegen wird auf das vorgestellte Rendering-gestützte Fehlermaß entsprechend des letzten Abschnittes zurückgegriffen.

3.5 Datenaugmentationen

Die Akquise von großen Mengen von gelabelten Trainingsdatenpaaren bestehend aus einem Foto und den korrespondierenden Reflexionsparametern stellt eine schwere Herausforderung dar [38]. Datenaugmentationsverfahren können aus einem kleinen Satz initialer Daten diesen für das Training um ein Vielfaches vergrößern [11].

In den folgenden Abschnitten werden eine Reihe an Datenaugmentationsverfahren vorgestellt. Abschnitt 3.5.1 behandelt die prozedurale Synthese von Trainingsdaten, die im Rahmen der Arbeit von Deschaintre et al. [11] genutzt wird. Abschnitt 3.5.2 stellt das Verfahren der Selbstaugmentation von Li et al. vor [38]. Ein Ansatz für die ad-hoc Datenaugmentation im Rahmen des Trainings wird in Abschnitt 3.5.3 thematisiert; dabei wird zunächst auf einen naiven Ansatz eingegangen und anschließend ein erweitertes Verfahren aus der Arbeit von Deschaintre et al. [12] vorgestellt, welches imstande ist, ad-hoc neue SVBRDFs zu generieren. Abschließend werden in Abschnitt 3.5.4 Deep-Learning-gestützte generative Ansätze thematisiert.

3.5.1 Prozedurale Synthese von Trainingsdaten

In diesem Abschnitt wird das Datenaugmentationsverfahren von Deschaintre et al. [11] vorgestellt; das Verfahren nutzt dabei einen initialen Datensatz von 155 SVBRDFs, der aus ausgesuchten künstlergenerierten Materialien besteht. Diese setzen

sich aus Materialien wie beispielsweise Plastik, Leder, Metall und Holz zusammen und entstammen einer Datenbank, welche aus mehr als 800 prozeduralen SVBRDFs zusammengesetzt ist; die ausgewählten Materialien lassen sich über einen Satz an Parametern prozedural anpassen.

Durch die Wahl zufälliger Parameter werden aus dem initialen Datensatz zunächst 1.850 SVBRDFs prozedural generiert. Mittels einem Blendingverfahren werden diese auf 20.000 ausgeweitet. Diese werden mit zehn zufälligen Konfigurationen bezüglich der Lichtquellenposition, Skalierung und Rotation gerendert. Abschließend werden die entstandenen Renderings zufällig zugeschnitten; das Resultat ist ein Datensatz von 200.000 SVBRDFs, der gleichzeitig den Datensatz dieser Arbeit bildet.

3.5.2 Selbstaugmentation

Im Folgenden wird ein Ansatz für das teilweise unbeaufsichtigte Lernen basierend auf der Idee der Selbstaugmentation vorgestellt. Dieser Ansatz beruht auf der Arbeit von Li et al. [38].

Beim Trainieren eines Netzes für die Schätzung von SVBRDFs basierend auf einem einzelnen Foto werden gelabelte Datensätze bestehend aus einem beleuchteten Foto und den dazu korrespondierenden Materialkarten benötigt. Die Akquise ungelabelter Daten, beispielsweise dargestellt durch ein reales Foto eines planaren Oberflächenmaterials, ist um ein Vielfaches leichter als eine zusätzliche Akquise der dazugehörigen SVBRDF und somit einem vollständigen Datum für das Training der Deep-Learning-gestützten inversen Renderingfunktion. Durch eine bereits existierende inverse Renderingfunktion, dem sogenannten Orakel, könnte die SVBRDF des ungelabelten akquirierten Datums ermittelt und somit dieses zu einem gelabelten Datum transformiert werden. Die Idee der Selbstaugmentation besteht darin, dass dieses Orakel durch ein teilweise vortrainiertes Netz basierend auf einem kleinen initialen Satz an gelabelten Daten dargestellt werden kann [38].

Abbildung 3.17 stellt die Idee der Selbstaugmentation schematisch dar. Hierbei wird ein teilweise vortrainiertes Deep Neural Network genutzt, um aus einem ungelabelten Datum eine SVBRDF zu schätzen, darauf basierend ein Rendering anzufertigen und so ein gelabeltes Datenpaar zu erstellen. Das erstellte Rendering kann anschließend als Eingabe für genau dieses Deep Neural Network dienen, wobei der Fehler basierend auf der initialen SVBRDF-Schätzung und der neuen SVBRDF-Schätzung für das Rendering gebildet und an das Deep Neural Network propagiert werden kann. So stellen SVBRDFs der initialen Datenmenge Stützstellen im Lösungsraum dar, die mittels der Selbstaugmentation im Lösungsraum korrekt interpoliert werden können

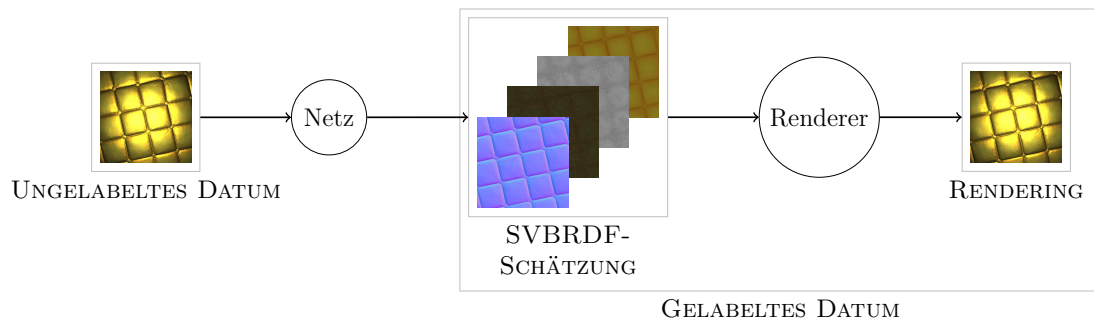


Abbildung 3.17: Generierung eines gelabelten Datums. Ausgehend von einem ungelabelten Datum – beispielsweise ein Blitzlichtfoto einer planaren Oberfläche – wird zunächst eine SVBRDF-Schätzung [11] mittels eines teilweise vortrainierten Netzes vorgenommen. Basierend auf dieser Schätzung wird ein beleuchtetes Bild generiert. Dieses Bild zusammen mit der Schätzung bildet das gelabelte Datum [38].

[38].

Da im Rahmen dieser Arbeit ein Datensatz von 200.000 SVBRDF zum Einsatz kam, musste die Selbstaugmentation für ein erfolgreiches Training nicht eingesetzt werden.

3.5.3 Online-Datengenerierung

In diesem Abschnitt werden zwei Ansätze für die Online-Datenaugmentation – der Datenaugmentation, die während des Trainings stattfindet – vorgestellt.

Das erste Verfahren nutzt einen initialen Datensatz, wie beispielsweise die 200.000 SVBRDFs aus der Arbeit von Deschaintre et al. [11], um während des Trainings ein zufälliges Blitzlichtfoto für die Eingabe zu generieren. Hierbei wird für ein Datum bestehend aus der SVBRDF und dem korrespondierenden Rendering nur die SVBRDF genutzt, wobei das Rendering verworfen wird. Stattdessen wird unter zufälligen Beleuchtungsbedingungen ein neues Bild mittels der differenzierbaren Renderingkomponente synthetisiert und als Eingabe für das Netz genutzt. Das Verfahren wird im Abschnitt 3.8.4 näher untersucht. Der Ansatz stellt jedoch eine naive Augmentationsmethode dar, da die zugrundeliegende Datenmenge an SVBRDFs nicht augmentiert wird.

Eine vollständige Online-Datengenerierung wurde in der Arbeit von Deschaintre et al. [12] vorgeschlagen, welche imstande ist, mittels einer integrierten Komponente in jeder Trainingsiteration basierend auf einem verhältnismäßig kleinen Satz von 1.850 initialen SVBRDFs ein neues Material zu synthetisieren. Hierbei wird ein dem Abschnitt 3.5.1 ähnliches Verfahren genutzt, welches ad-hoc ein Materialblending vornimmt. Zusätzlich kommen weitere Augmentationen zum Einsatz; diese werden

beispielsweise durch zufällige Lichtintensitäten und -farben und Rotationen dargestellt. Mittels dieser Methode können in der Praxis bis zu zwei Millionen Trainingsdaten generiert werden [12].

Es hat sich gezeigt, dass der Aufwand für die Generierung von Daten während des Trainings durch die entfallenden Kosten für das Einlesen von offline-generierten Daten amortisiert wird [12].

3.5.4 Generative Datenaugmentation

In diesem Abschnitt werden generative Methoden basierend auf modernen Deep-Learning-Ansätzen für eine mögliche Datenaugmentation vorgestellt; da diese Methoden nicht den Fokus dieser Arbeit darstellen, wird hier lediglich ein zusammenfassender Überblick über eine solche Idee gegeben.

Generative Methoden, die neue Proben aus einer hochdimensionalen Datenverteilung generieren können, finden Anwendung in vielen Bereichen; eine solche hochdimensionale Datenverteilung kann sich hier beispielsweise auf Bilder beziehen [27].

Die drei prominentesten generativen Ansätze werden durch die autoregressiven Modelle, wie sie beispielsweise durch das PixelCNN dargestellt werden [51], die Variational Autoencoder (VAEs) [32] und durch die Generative Adversarial Networks (GANs) [20] repräsentiert [27]. Im Folgenden wird ein Datenaugmentationsansatz anhand der Funktionsweise der GANs näher erläutert.

Generative Adversarial Networks bestehen aus einem Generatornetzwerk G und einem Diskriminatornetzwerk D , wobei der Generator eine Probe aus einer Datendistribution zu generieren versucht und der Diskriminator zwischen generierten und tatsächlichen Proben differenzieren muss, was in einem Minimax-Spiel mit zwei Spielern resultiert [20]. Beide Netze werden hierbei durch Fully Connected Layer dargestellt [20]. Der Generator bekommt dabei einen latenten Vektor $\mathbf{z}$ als Eingabe und generiert ein Datum $\mathbf{x}$. Der Diskriminator bekommt ein Datum $\mathbf{x}$ als Eingabe und gibt einen Konfidenzwert $D(\mathbf{x}) \in [0, 1]$ aus, der die geschätzte Wahrscheinlichkeit des Netzwerkes angibt, dass es sich beim Eingabedatum um eine Probe aus der tatsächlichen Datendistribution handelt, wobei in diesem Fall ein Wert von 1 approximiert werden soll [20].

Die Funktion

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (3.14)$$

stellt die Wertefunktion eines solchen Spiels dar, wobei der Generator die Summe

zu minimieren und der Diskriminator die Summe zu maximieren versucht; der erste Summand stellt dabei den Konfidenzwert des Diskriminators für ein tatsächliches Datum $\mathbf{x}$ aus der entsprechenden Datenverteilung dar und der zweite Summand gibt den invertierten Konfidenzwert des Diskriminators für ein durch den Generator synthetisiertes Datum an, wobei beide Summanden durch die Logarithmus-Funktion in den logarithmischen Raum transformiert werden [20].

Im Rahmen dieser Anwendung stellt ein tatsächliches Datum $\mathbf{x}$ einen Satz an Materialkarten aus einer initialen SVBRDF-Datenmenge dar, die der Diskriminator von generierten Materialkarten des Generators differenzieren muss. Das Ziel besteht darin, dass in einem trainierten Zustand ein solcher Generator ausgehend von einem beliebigen Vektor $\mathbf{z}$ einen neuen Satz valider Materialkarten aus der ursprünglichen Datenverteilung generieren soll. So ein Generator könnte ad-hoc Trainingsdaten für das Training einer inversen Renderingfunktion, wie sie in diesem Kapitel vorgestellt wurde, generieren. Basierend auf den ad-hoc generierten SVBRDFs kann folglich ein Rendering durchgeführt werden, was in einem synthetischen Blitzlichtfoto und somit zusammen mit der generierten SVBRDF in einem gelabelten Datenpaar resultiert. Das Training solcher GANs stellt eine Herausforderung dar; zum einen generieren GANs in der Regel kleine Bilder mit wenigen Variationen, zum anderen ist ein Trainingsprozess für GANs oft instabil [27]. Insbesondere können durch ein unausgeglichenes Training von G und D die Abbildungen des Generators zusammenfallen, wobei als Folge viele latente Vektoren dasselbe Datum generieren [20]. In der Arbeit von Karras et al. [27] wird eine Methode für das Training solcher GANs vorgestellt, die eine progressive Vergrößerung der generierten Bilder vornehmen und somit sowohl in der Lage ist, größere Bilder zu generieren als auch das Training stark zu stabilisieren. Ein solches Verfahren kann in zukünftigen Arbeiten als möglicher Lösungsansatz genutzt werden, um eine Datenaugmentation ausgehend von einer kleinen Menge an SVBRDFs vorzunehmen.

3.6 Multiple Eingabefotos

Im Gegensatz zu traditionellen optimierungsbasierten BRDF-Akquisitionsverfahren, die in der Regel hunderte Bilder eines Materials benötigen, wird lediglich ein einzelnes Foto von dem in diesem Kapitel vorgestellten Deep-Learning-basierten Verfahren benötigt. Oft reicht ein einzelnes Foto jedoch nicht aus, um die umfangreiche Erscheinung eines Materials akkurat festzuhalten, da das Aussehen von realen Materialien sowohl von der Blick- als auch Lichtquellenrichtung abhängig ist. Durch den Ein-

satz von mehreren Fotos desselben Materials unter verschiedenen Beleuchtungs- und Kamerakonfigurationen kann ein breiteres Spektrum an Reflexionsinformationen im Rahmen der Eingabe abgedeckt werden [12].

Im Folgenden werden zwei Ansätze vorgestellt, die eine Eingabe von mehr als einem Foto für das inverse Rendering planarer Oberflächen zulassen und somit relevante Erweiterungen für die Arbeit von Deschaintre et al. [11] darstellen.

Abschnitt 3.6.1 nutzt dabei pro Eingabefoto eine Instanz eines parallel geschalteten Encoder-Decoder-Netzwerks, um eine Eingabe mehrerer Fotos zu ermöglichen [12]. In Abschnitt 3.6.2 wird ein Verfahren vorgestellt, welches anhand multipler Eingabefotos eine Suche im latenten Raum eines trainierten Autoencoders nach einem möglichst ähnlichen Material vornimmt [16].

Der Akquisitionsprozess solcher Fotos, die das Erscheinen eines Materials unter variierenden Kamera- und Lichtquellenpositionen festhalten, steht hierbei nicht im Fokus dieses Abschnittes; Akquisemethoden bzw. Vorgaben für einen Satz multipler Eingabefotos sind in den Arbeiten von Deschaintre et al. [12] und Gao et al. [16] nachzulesen.

3.6.1 Parallele U-Nets

In diesem Abschnitt wird der Ansatz von Deschaintre et al. [12] für das Verarbeiten mehrerer Eingabefotos für die SVBRDF-Akquisition planarer Oberflächenmaterialien vorgestellt. Abbildung 3.18 stellt den Aufbau der für diesen Ansatz eingesetzten Architektur schematisch dar. Für jedes Eingabefoto wird hierbei eine eigene modifizierte Instanz der im Abschnitt 3.3.2 vorgestellten erweiterten U-Net-Architektur erstellt. Anstatt einer finalen SVBRDF-Schätzung mit den entsprechenden zehn Kanälen produzieren die jeweiligen Instanzen jeweils ein Zwischenergebnis mit 64 Kanälen. Durch ein *Max Pooling Layer* werden diese Zwischenergebnisse unabhängig von der Anzahl und Reihenfolge der jeweiligen Instanzen auf ein Bild der Form $256 \sim 64$ reduziert. Dieses durchläuft drei weitere Faltungen, wobei die letzte Faltung die finale SVBRDF-Schätzung als Ergebnis liefert.

Jede Instanz besitzt zudem ein eigenes globales Merkmalsnetzwerk mit eigenen globalen Merkmalsvektoren. Die Vektoren der letzten Schicht jeder Instanz werden hierbei ebenfalls mittels eines Max-Poolings auf einen einzigen Satz an Merkmalsvektoren reduziert. Diese Vektoren werden einem globalen Merkmalsnetzwerk übergeben, welches parallel zu den letzten drei Faltungen läuft. Hierbei findet ebenfalls ein Informationsaustausch zwischen diesen Faltungsschichten und dem parallelen Merkmalsnetzwerk statt.

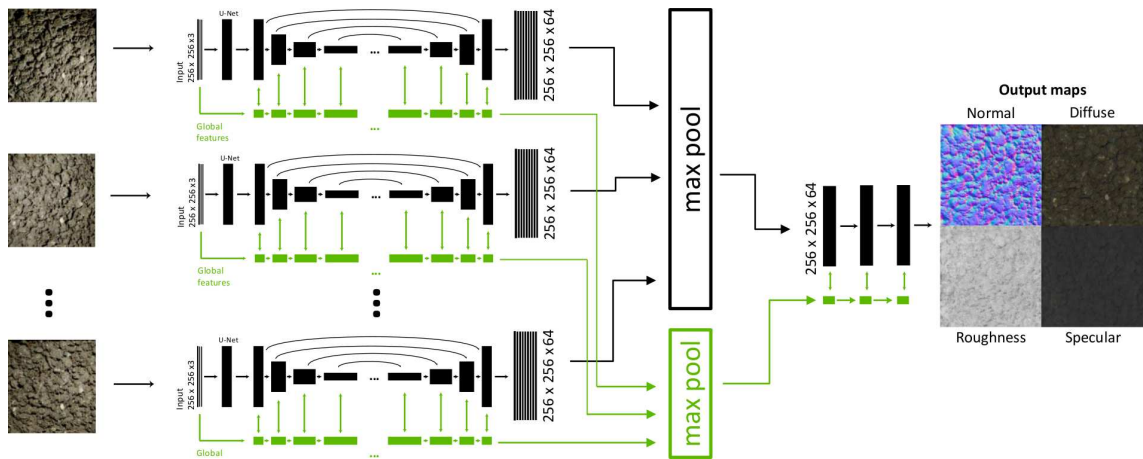


Abbildung 3.18: Architektur paralleler U-Nets mit globalen Merkmalsnetzen [12]. Das Material für die SVBRDF-Akquise wird unter verschiedenen Beleuchtungsbedingungen aufgenommen; für jedes Eingabefoto wird eine Instanz der U-Net-Architektur mit globalem Merkmalsnetzwerk erzeugt, die dieses Foto verarbeitet. Nach der letzten Faltung liegt hierbei nicht die finale SVBRDF, sondern ein Zwischenergebnis mit 64 Kanälen vor. Die Ergebnisse der jeweiligen Instanzen werden durch eine Max-Pooling-Operation auf ein einzelnes Bild reduziert, welches nach drei weiteren Faltungen die finale SVBRDF repräsentiert. Parallel dazu werden auch die globalen Merkmalsvektoren der jeweiligen Instanzen mittels eines Max-Poolings zusammengefasst und an ein einziges paralleles Merkmalsnetzwerk übergeben, welches parallel zu den letzten drei Faltungsschichten läuft [12].

Während des Trainings wird das Netz mit einer Anzahl zwischen einem und fünf Eingabebildern trainiert, kann jedoch im trainierten Zustand eine beliebige Anzahl an Eingabebildern verarbeiten.

Im Vergleich zu Methoden, die lediglich ein Foto als Eingabe akzeptieren, liefert diese Methode bei bereits wenigen zusätzlichen Eingabebildern wesentlich bessere Ergebnisse [12].

3.6.2 Autoencoder-gestützte inverse Renderingfunktion

Im Folgenden wird ein Ansatz aus der Arbeit von Gao et al. [16] vorgestellt, der mithilfe eines Autoencoders eine genauere SVBRDF-Approximation basierend auf mehreren Eingabebildern erzielen kann. Eine Voraussetzung bei diesem Ansatz besteht darin, dass sich die Lichtquelle und die Kameraposition approximativ an der selben Position befinden und darüberhinaus diese durch die extrinsischen und intrinsischen Kameraparameter des i -ten Bildes durch $\mathbf{C}_i$ gegeben und somit bekannt sind. Seien somit $\mathbf{l}_i$ und $\mathbf{v}_i$ die i -te Lichtquellen- und Kameraposition in dieser Reihenfolge und die korrespondierenden Bilder $\mathbf{I}_i$ radiometrisch linearisiert und auf die frontale Ansicht korrigiert. Des Weiteren wird für das erste der aufgenommenen Fotos $\mathbf{I}_0$ des planaren Oberflächenmaterials gefordert, dass die Richtung zur Kamera die Normale der aufgenommenen Oberfläche approximiert.

Im Gegensatz zu dem Ansatz des vorausgehenden Abschnittes wird das Erscheinen des Materials anhand der Eingabefotos direkt im latenten Raum optimisiert. Dieser latente Raum wird durch einen auf Faltungen basierenden Autoencoder konstruiert. Hierbei überträgt der Encoder E durch

$$\mathbf{z} = E(\mathbf{M}) \quad (3.15)$$

eine durch Materialkarten repräsentierte SVBRDF $\mathbf{M}$ in die Repräsentationsform des latenten Vektors $\mathbf{z}$; der Decoder D rekonstruiert mittels

$$\mathbf{M} = D(\mathbf{z}) \quad (3.16)$$

eben diese Materialkarten $\mathbf{M}$ anhand des entsprechenden latenten Vektors $\mathbf{z}$.

Im Folgenden wird der Ablauf der inversen Renderingfunktion basierend auf dem vorgestellten Autoencoder beschrieben. Zunächst wird das erste Foto $\mathbf{I}_0$ des planaren Oberflächenmaterials in die trainierte Deep-Learning-gestützte inverse Renderingfunktion mit globalem Merkmalsnetzwerk (siehe Abschnitt 3.3.2) eingegeben. Dies

resultiert in einer initialen SVBRDF-Schätzung, die anschließend in den trainierten Autoencoder eingegeben wird, um den initialen latenten Vektor $\mathbf{z}$ zu erhalten.

Darauf basierend wird dieser latente Vektor dahingehend optimiert, dass dieser die Zielfunktion

$$\min_{\mathbf{z}} \sum_i \delta(\mathbf{I}_i, R(D(\mathbf{z}), \mathbf{l}_i, \mathbf{v}_i)) \quad (3.17)$$

erfüllt, wobei δ sich auf den L1-Fehler der Logarithmen, welcher im Abschnitt 3.4.2 vorgestellt wurde, bezieht. Hierbei wird anhand des zu optimierenden latenten Vektors $\mathbf{z}$ ein Decoding $D(\mathbf{z})$ vorgenommen, um einen Satz an Materialkarten zu erhalten und unter den approximativ bekannten Lichtquellen- und Kamerapositionen des jeweiligen Eingabefotos zu rendern. Der Fehler zwischen dem i -ten Eingabefoto $\mathbf{I}_i$ und dem dazu korrespondierenden Rendering wird optimiert, um den finalen latenten Vektor $\mathbf{z}$ zu erhalten. Ausgehend von diesem Vektor bilden die Materialkarten $D(\mathbf{z})$ die finale SVBRDF-Schätzung.

Das in diesem Abschnitt vorgestellte Verfahren konnte im Vergleich mit verwandten Arbeiten, die lediglich ein einzelnes Eingabefoto für eine SVBRDF-Akquise verarbeiten, bessere Ergebnisse erzielen [16].

3.7 Realisierung

Im Folgenden werden die Implementierungsdetails der Realisierung bezüglich der in diesem Kapitel vorgestellten inversen Renderingfunktion für planare Oberflächen erläutert.

Framework Die Implementierung wurde in der Programmiersprache Python durchgeführt. Als Framework für die Methoden des maschinellen Lernens wurde die PyTorch-Bibliothek eingesetzt².

Datenlader Für das Laden des SVBRDF-Datensatzes samt physikalisch-basiert gerendertem Blitzlichtfoto [11] wurde ein angepasster PyTorch-Datenlader implementiert; mittels der *Python Image Library* (PIL)³ konnten die Daten eingelesen und in das entsprechende Format zugeschnitten werden. Mithilfe von Funktionen der PyTorch-Bibliothek ließen sich die Bilder in Tensoren transformieren, die somit als Eingabe für ein Deep Neural Network fungieren konnten.

²PyTorch <https://pytorch.org/> – Zugriffsdatum: 26. September 2019

³Python Image Library (PIL) <https://pypi.org/project/Pillow/> – Zugriffsdatum: 26. September 2019

U-Net Das einfache U-Net (siehe Abschnitt 3.3.1) wurde in Modulen implementiert; dabei wurde jede Schicht in ein eigenes parametrisierbares Modul integriert. Sämtliche Operationen, wie sie beispielsweise durch die Convolutional-Schichten, die LeakyReLU-Schichten und der Sigmoid-Funktion dargestellt werden, wurden der PyTorch-Bibliothek entnommen. Die Skipconnections wurden mittels Konkatenationen umgesetzt. Für die Transformation der Eingabebilder in den logarithmischen Raum wurde ein Glättungsmodul implementiert, was dem Deep Neural Network als erste Schicht zugewiesen wird.

U-Net+ Die Erweiterung der U-Net-Architektur durch das parallele globale Merkmalsnetzwerk wurde mittels Fully-Connected- und SELU-Schichten vorgenommen. Beide Funktionalitäten sind ebenfalls im PyTorch-Framework enthalten.

Differenzierbare Renderingkomponente Die Komponente für die Synthese von beleuchteten Blitzlichtfotos setzt das vorgestellte physikalisch-basierte Cook-Torrance-Modell (siehe Abschnitt 2.1.3.2) um; sämtliche Operationen wurden dabei mithilfe von Tensor-Operationen der PyTorch-Bibliothek durchgeführt, um eine vollständige GPU-beschleunigte Implementierung und die automatische Differenzierbarkeit [52] zu ermöglichen. Die Renderingkomponente wurde als PyTorch-Modul implementiert und kann als eine Deep-Neural-Network-Schicht verstanden werden.

Stochastisches Rendering-gestütztes Fehlermaß Die stochastische Rendering-gestützte Fehlerfunktion instanziiert die differenzierbare Renderingkomponente und fertigt mittels dieser eine Reihe an diffusen und Glanzlichtrenderings entsprechend des Abschnitts 3.4.2 an. Die resultierenden Renderings werden in den logarithmischen Raum transformiert und mittels des L1-Fehlermaßes miteinander verglichen. Der durchschnittliche Fehler aller Renderings wird als Renderingfehler bezeichnet. Den Lichtquellen- und Augenpunktpositionen werden dabei zufällig generierte Vektoren der Kosinusverteilung zugewiesen; diese generiert Tensoren mit drei Komponenten, die einen Ortsvektor für die Renderingkomponente darstellen.

Logging Während des Trainings werden sämtliche Fortschritte im Rahmen eines Loggings dokumentiert. Hierbei werden neben allen Fehlerwerten auch Ausgabebilder jeder Epoche stichprobenweise im vom Benutzer angegebenen Ausgabeordner abgespeichert; dies ermöglicht eine direkte Einsicht in die qualitative Trainingsperformance des Deep Neural Networks ohne das Training zu unterbrechen.

Training Das Training wird in mehreren Epochen durchgeführt; die genaue Anzahl ist dabei frei parametrisierbar und kann vom Benutzer angegeben werden. Im Rahmen des Trainings wird das Deep Neural Network, der Datenlader und das stochastische Rendering-gestützte Fehlermaß instanziiert. Zudem wird ein Optimizer basierend auf dem Adam-Algorithmus [31] eingesetzt, der die vom Renderingfehler entnommenen Gradienten für eine Verbesserung der Parameter des Deep Neural Networks nutzt. Nach der letzten Epoche wird das Programm terminiert; zuvor wird der letzte Zustand des Deep Neural Networks in Form der Parameter für eine spätere Evaluierung abgespeichert.

Naive Datenaugmentation Die naive Datenaugmentation nutzt die SVBRDF-Materialkarten aus dem Datenlader. Das eigentliche Blitzlichtfoto wird verworfen; stattdessen wird mittels der differenzierbaren Renderingkomponente ad-hoc ein neues Blitzlichtfoto erzeugt, welches als Eingabe für das Deep Neural Network diene.

3.8 Evaluierung

In diesem Abschnitt werden die umgesetzten Methoden sowohl qualitativ als auch quantitativ analysiert und bewertet. Abschnitt 3.8.1 stellt dabei die Evaluierungsszenarien vor. In den darauf folgenden Abschnitten wird die Evaluierung vorgenommen.

3.8.1 Evaluierungsszenarien

Im Folgenden werden die Rahmenbedingungen der Evaluierung vorgestellt. Dabei werden zunächst die untersuchten Aspekte basierend auf den umgesetzten und zur Verfügung stehenden Verfahren dargelegt und anschließend der eingesetzte Datensatz und das Referenzsystem erläutert.

Im Rahmen dieser Arbeit wurden die Deep-Neural-Network-Varianten des ursprünglichen U-Nets (siehe Abschnitt 3.3.1) und der erweiterten Version mit dem globalen Merkmalsnetzwerk (siehe Abschnitt 3.3.2) umgesetzt. Zudem wurde die naive Datenaugmentation durch die ad-hoc Generierung von Blitzlichtfotos (siehe Abschnitt 3.5.3) basierend auf dem eingesetzten SVBRDF-Datensatz realisiert. Für das Training wurde das stochastische Rendering-gestützte Fehlermaß samt differenzierbarer physikalisch-basierter Renderingkomponente implementiert (siehe Abschnitt 3.4). Des Weiteren steht das trainierte Deep Neural Network der Arbeit von Deschaintre et al. [11] für eine qualitative Analyse von SVBRDF-Schätzungen als Referenzimplementierung zur Verfügung. Die untersuchten Aspekte im Rahmen dieses Abschnittes

beziehen sich auf die Punkte

- des qualitativen und quantitativen Vergleichs der umgesetzten U-Net-Varianten und somit dem Vorteil der Erweiterung, wobei die ursprüngliche Variante im Folgenden durch den Begriff *U-Net* und die erweiterte Version durch die Bezeichnung *U-Net+* beschrieben wird,
- der qualitativen Untersuchung der Praktikabilität durch die Anwendung der entwickelten inversen Renderingfunktion auf selbstakquirierte reale planare Materialproben und eine darauf aufbauende Neubeleuchtung basierend auf den resultierenden SVBRDF-Schätzungen,
- der quantitativen Analyse der naiven Datenaugmentation durch einen Vergleich mit den U-Net- und U-Net+-Varianten und
- dem qualitativen Vergleich der SVBRDF-Schätzungen und den darauf basierenden Neubeleuchtungen der umgesetzten Verfahren mit der Referenzimplementierung von Deschaintre et al. [11].

Der grundlegende Datensatz wird durch 200.000 SVBRDFs samt physikalisch-basiert gerenderten Beleuchtungen der Materialien dargestellt [11]. Die Evaluierung wurde auf einem Linux-System mit dem *Ubuntu 18.04 LTS* Betriebssystem durch die GPU *NVIDIA GeForce GTX 1080 Ti* mit 12 GB VRAM beschleunigt durchgeführt. Dabei stand dem System 128 GB Hauptspeicher auf einer *Intel Core i7-6850K* CPU mit sechs Kernen und einer Taktfrequenz von 3.6 GHz zur Verfügung.

Das Training der Deep Neural Networks wurde dabei unter den selben Bedingungen entsprechend der Arbeit von Deschaintre et al. [11] durchgeführt. Dabei wurden die Netze mit einer Batchgröße von 8 für 4000 Epochen trainiert. Als Optimierungsverfahren wurde der Adam-Algorithmus [31] eingesetzt und für die Bestimmung der Gradienten das stochastische Rendering-gestützte Fehlermaß (siehe Abschnitt 3.4) ausgewählt. Abweichend von den Parametern der Originalarbeit [11] wurde für diese Evaluierung jedoch eine Lernrate von 0.00001 gewählt; dieser Wert hat sich im Rahmen einer Reihe von kleinen Experimenten als minimal besser herausgestellt. Das Training wurde dabei auf dem bereits vorgestellten Referenzsystem durchgeführt. Für die Neubeleuchtung der SVBRDF-Resultate wurde ein Beleuchtungs-Shader eingesetzt (siehe Anhang A).

Im folgenden Abschnitt 3.8.2 werden die U-Net-Varianten sowohl qualitativ als auch quantitativ miteinander verglichen.

3.8.2 Vergleich der U-Net-Varianten

Im Rahmen der folgenden Analyse werden die jeweiligen U-Net-Varianten sowohl qualitativ als auch quantitativ miteinander verglichen und somit der Vorteil des parallelen globalen Merkmalsnetzwerks untersucht.

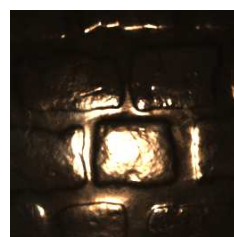
Dazu wird in Abschnitt 3.8.2.1 die SVBRDF-Akquise einer synthetisch generierten Materialprobe durch die jeweiligen Deep Neural Networks durchgeführt und eine anschließende Neubeleuchtung für einen qualitativen Vergleich vorgenommen. Im Abschnitt 3.8.2.2 werden die im Rahmen des Trainings entstandenen Fehlerkurven bezüglich des stochastischen Renderingfehlers und der Konvergenz der geschätzten auf die tatsächlichen Materialkarten untersucht.

3.8.2.1 Qualitative Analyse

Die anwendungsspezifische Erweiterung durch das parallele globale Merkmalsnetzwerk wurde als Gegenmaßnahme für das Entstehen von Flecken auf den geschätzten Materialkarten eingeführt (siehe Abschnitt 3.3.2). Der Effekt der Erweiterung wird im Rahmen dieser qualitativen Analyse der SVBRDF-Schätzungen und der darauf basierenden Neubeleuchtung untersucht.

Dazu wird ein synthetisches Blitzlichtfoto als exemplarische Eingabe für die Deep Neural Networks gewählt (siehe Abbildung 3.19). Dies ermöglicht die Gegenüberstellung der tatsächlichen Materialkarten mit denen der SVBRDF-Schätzungen des U-Nets und U-Net+; zudem bildet dies eine Grundlage für die Neubeleuchtung aller drei SVBRDFs mit exakt gleichem Beleuchtungssetup.

Das synthetische Blitzlichtfoto wird in die entsprechend des Abschnitts 3.8.1 trainierten Deep Neural Networks eingegeben und die resultierenden SVBRDFs durch



(a) Eingabe



(b) Geglättete Eingabe

Abbildung 3.19: Eingabedatum für die qualitative Analyse der U-Net-Varianten [11]. Abbildung (a) stellt dabei ein synthetisches Blitzlichtfoto eines planaren Oberflächenmaterials dar. Abbildung (b) stellt die bezüglich des dynamischen Bereichs geglättete Version im logarithmischen Raum dar.

ein externes Beleuchtungstools mittels des Cook-Torrance-Modells unter denselben Bedingungen beleuchtet.

Abbildung 3.20 stellt die Resultate tabellarisch dar. Die jeweiligen Spalten stellen von links nach rechts die Normalenkarte, die diffuse Grundfarbe, die Rauheitskarte und die Glanzlichtfarbe für das Cook-Torrance-Beleuchtungsmodell dar. Die letzte Spalte bildet die Neubeleuchtungen unter denselben Beleuchtungsbedingungen ab. Die Farbwerte der Karten der diffusen Grundfarbe und der Glanzfarbe wurden hierbei für eine bessere Visualisierung mit einem Exponenten von 0.5 versehen.

Die SVBRDF-Schätzungen des ursprünglichen U-Nets weisen sowohl große Flecken, die sich auf Beleuchtungsrückstände zurückführen lassen, als auch Artefakte in den Materialkarten auf. Im Vergleich dazu sind nur minimale Beleuchtungsrückstände in den Schätzungen der erweiterten Architektur nachzuweisen. Die resultierenden Renderings unterstreichen das Defizit der ursprünglichen U-Net-Architektur; das beleuchtete Foto basierend auf den Materialkarten der U-Net-Schätzung weist Verfärbungen und Flecken auf. Im Gegensatz dazu ist das synthetische Foto der erweiterten Architektur dem Ground-Truth-Rendering wesentlich ähnlicher.

Die Defizite der U-Net-Architektur lassen sich auf die mangelhafte Propagierung globaler Informationen, wie sie beispielsweise durch die durchschnittliche Pixelfarbe dargestellt werden, zurückführen. Durch das zusätzliche parallele Merkmalsnetzwerk ermöglicht das U-Net+ die Weiterleitung globaler Informationen an jedes Pixel und hebt somit das vorliegende Defizit auf.

Somit kann qualitativ festgehalten werden, dass das parallele globale Merkmalsnetzwerk eine sinnvolle anwendungsspezifische Erweiterung darstellt, die zu besseren Ergebnissen sowohl in den Materialkarten, als auch in den darauf basierend resultierenden Renderings führt.

3.8.2.2 Quantitative Analyse

Zusätzlich zu der qualitativen Analyse des vorausgehenden Abschnitts wird im Folgenden eine quantitative Auswertung der U-Net-Varianten durchgeführt. Hierbei wird die Entwicklung sämtlicher Fehlerwerte über die gesamte Trainingszeit, die in Epochen angegeben wird, betrachtet und daraus ein Fazit gezogen.

Für die quantitative Analyse der Deep Neural Networks müssen die Fehlerwerte in jeder Epoche zwischengespeichert und der Durchschnitt dieser Fehlerwerte ermittelt werden.

Dazu werden Logging-Mechanismen in den Ablauf des Trainings integriert und für jede Epoche der durchschnittliche Fehlerwert des stochastischen Rendering-gestützten

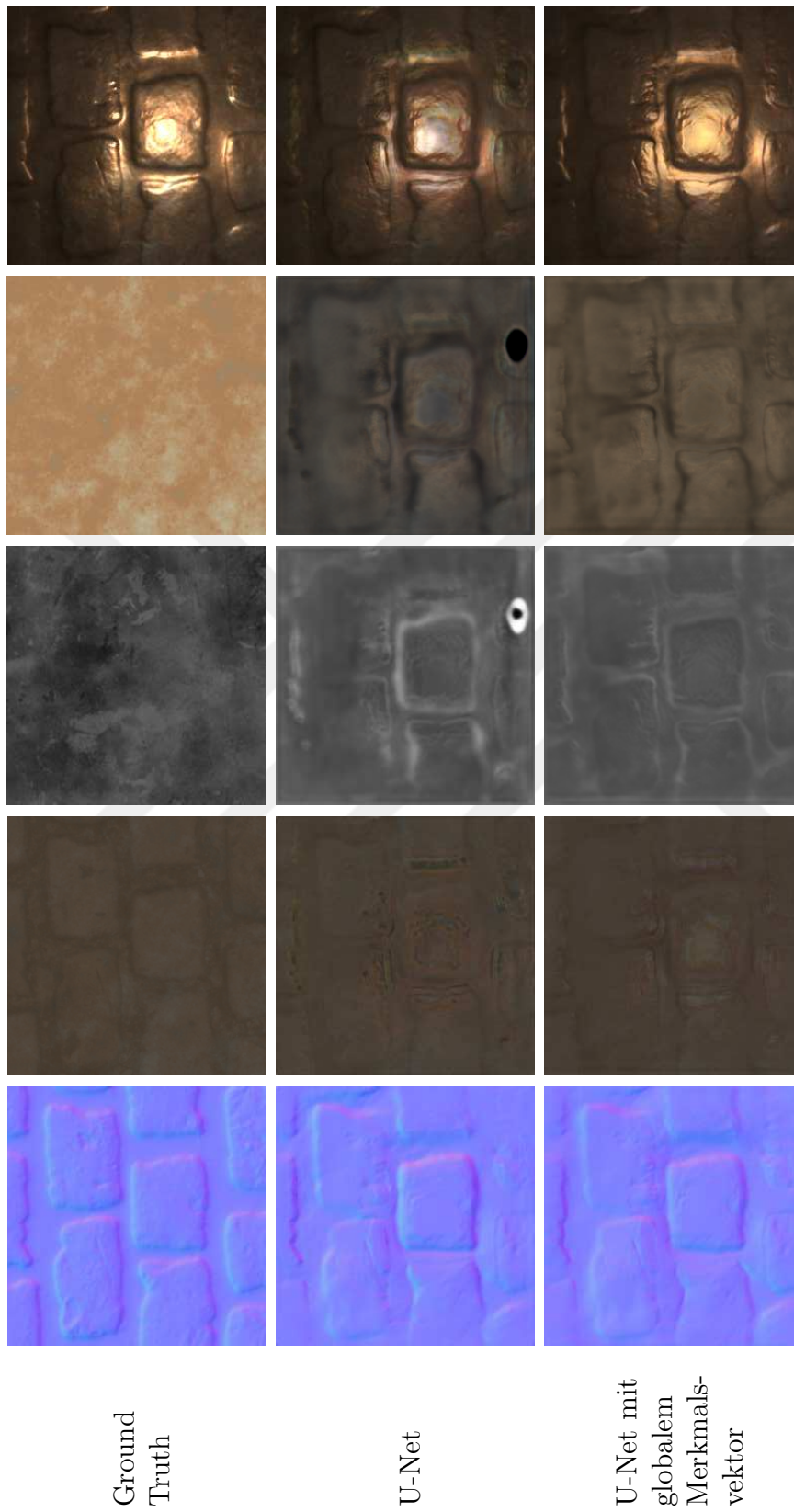


Abbildung 3.20: Qualitativer Vergleich der Ergebnisse der jeweiligen Architekturen. Die Spalten stellen von links nach rechts die Normalkarte, die diffuse Grundfarbe, die Rauheitskarte, die Glanzfarbe und Renderings basierend auf diesen Karten unter selben Beleuchtungsbedingungen dar. Die erste Reihe stellt die tatsächlichen Daten [11], die zweite Reihe die Schätzungen des U-Nets und die letzte Reihe die Schätzungen der erweiterten Architektur dar. Die U-Net-Schätzungen weisen sowohl Beleuchtungsrückstände als auch Artefakte auf, die in fehlerhaften Renderings resultieren. Für die Evaluierung wurden die Abbildungen der diffusen Grundfarbe und Glanzfarbe mit einem Exponenten von 0.5 versehen.

Fehlermaes und die durchschnittlichen L1-Fehler der jeweiligen Materialkarten aufgezeichnet.

Die Abbildung 3.21 stellt den durchschnittlichen Renderingfehler und die Abbildung 3.22 die durchschnittlichen L1-Fehler der Materialkarten über die 4000 trainierten Epochen dar. Die roten Kurven entsprechen den Fehlerwerten des U-Nets und die blauen Kurven den Fehlerwerten des U-Net+. Die x-Achse gibt dabei die Epochen des Trainings und die y-Achse die jeweiligen durchschnittlichen Fehlerwerte an. Dabei sind die Fehlerkurven der U-Net+-Varianten durchweg niedriger als

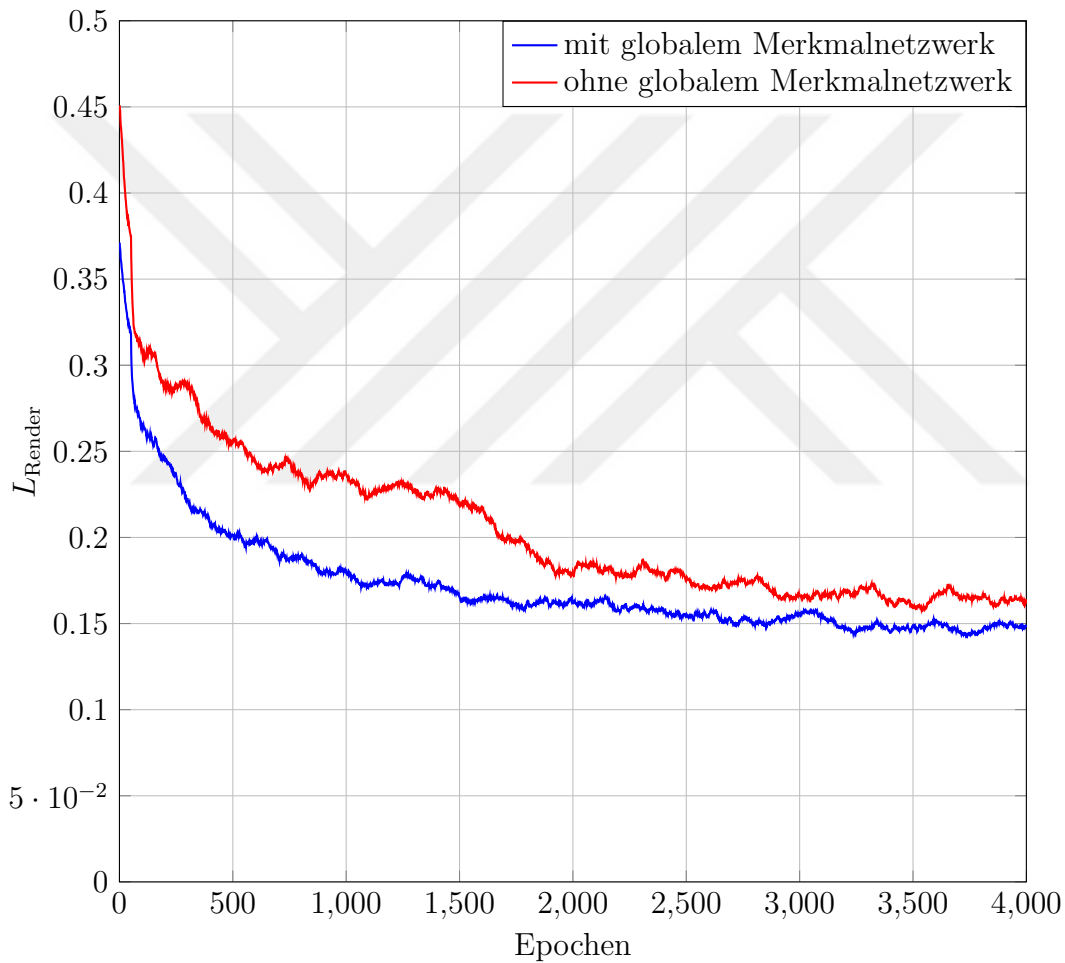


Abbildung 3.21: Vergleich der Renderingfehler während des Trainings. Die x-Achse stellt die Epochen, die y-Achse den durchschnittlichen Renderingfehler dar. Die rote Kurve spiegelt den Fehler der ursprünglichen U-Net-Architektur, die blaue Kurve den Fehler der erweiterten Architektur mit globalem Merkmalsnetzwerk wider. Die Kurven wurden mit einer Kernelgröße von 101 geglättet, um das Rauschen des stochastischen Fehlermaßes zu reduzieren. Die erweiterte Architektur erzielt dabei durchweg bessere quantitative Ergebnisse.

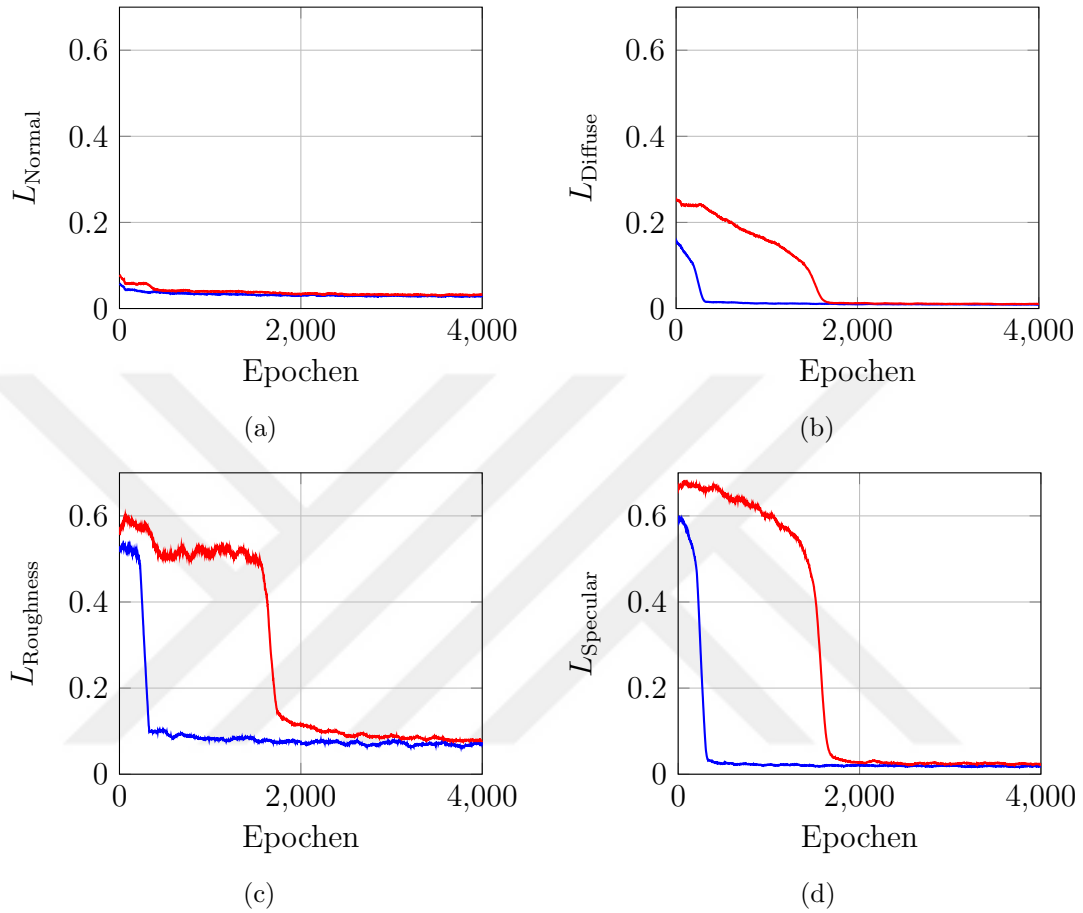


Abbildung 3.22: Vergleich der durchschnittlichen L1-Fehler während des Trainings. Hierbei stellt die x-Achse die Anzahl der Epochen und die y-Achse den durchschnittlichen L1-Fehler (a) der Normalenkarte, (b) der diffusen Grundfarbe, (c) der Rauheitskarte und (d) der Glanzfarbe dar. Die rote Kurve stellt die Architektur ohne, die blaue dagegen die Architektur mit dem globalen Merkmalsnetzwerk dar. Die Kurven wurden mit einer Kernelgröße von 101 geglättet. Bemerkenswert ist, dass der Fehler der Normalenkarte durchweg niedrig ist, die anderen Fehlerwerte dagegen bei der erweiterten Architektur bei ca. der 500. Epoche, bei der ursprünglichen Architektur bei ca. der 2000. Epoche schlagartig abfallen.

die des U-Nets. Des Weiteren weisen auch die Fehlerkurven der Materialkarten der U-Net+-Schätzungen geringere Fehlerwerte auf.

Da das Rendering-gestützte Fehlermaß das Aussehen der Renderings in die Bewertung miteinbezieht, kann geschlussfolgert werden, dass die erweiterte Architektur – dargestellt durch die blaue Kurve – einen Vorteil für die inverse Renderingfunktion bietet. Darüber hinaus sind auch die durchschnittlichen L1-Fehler der Materialkarten durchgängig niedriger; das signalisiert, dass das normale U-Net Materialkarten mit größerem Fehler in den Schätzungen produziert.

Somit lässt sich ein durchweg quantitativer Vorteil der erweiterten Architektur gegenüber der einfachen U-Net-Variante festhalten und der Einsatz der Erweiterung rechtfertigen.

3.8.3 SVBRDF-Akquise realer Materialien

In diesem Abschnitt wird die Praktikabilität der umgesetzten inversen Renderingfunktion für planare Oberflächen mittels einer qualitativen Analyse der SVBRDF-Akquise realer Materialien untersucht. Dazu werden Neubeleuchtungen basierend auf den geschätzten SVBRDFs angefertigt und diese qualitativ bewertet. Ein wichtiger Aspekt bei der Analyse besteht darin, dass keine tatsächlichen Materialkarten der realen Materialproben vorliegen und somit keine parallele quantitative Analyse durchgeführt werden kann.

Für eine umfassende qualitative Untersuchung wird eine Menge an Proben realer Materialien mit grundsätzlich unterschiedlichen Eigenschaften bezüglich ihres Lichtverhaltens akquiriert. Die Akquise wird mittels eines Smartphones vom Modell *OnePlus One* durchgeführt; die Fotos werden mit eingeschaltetem Blitzlicht aufgenommen und im LDR-Format abgespeichert. Abbildung 3.23 stellt die vier aufgenommenen

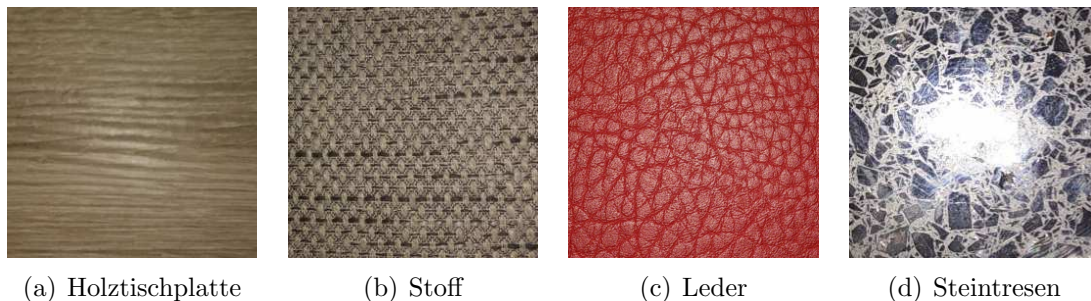


Abbildung 3.23: Blitzlichtaufnahmen realer Materialien. Abbildung (a) stellt eine Holztafelplatte, (b) einen Stoffbezug, (c) eine Wandbekleidung aus Leder und (d) einen polierten Steintresen dar.

Materialproben für die Untersuchung dar. Diese bestehen aus

- einer Holztischplatte (siehe Abbildung 3.23(a)), die sowohl diffuse als auch Glanzreflexionen aufweist und leichte geometrische Details besitzt,
- einem Stoffbezug (siehe Abbildung 3.23(b)), der vollständig diffus reflektiert und zudem starke geometrische Strukturen aufweist,
- einer Lederwandbekleidung (siehe Abbildung 3.23(c)), die diffuse und Glanzreflexionen besitzt, wobei die glänzenden Bereiche von den Konturen des Leders getrennt werden und
- einer polierten Steintischplatte (siehe Abbildung 3.23(d)), die starke Glanzreflexionen aufweist, jedoch keine geometrischen Details besitzt.

Die aufgenommenen Fotos werden durch eine Skalierung und dem Zuschneiden auf eine Größe von 256×256 Pixeln gebracht. Diese werden durch eine Glättung mittels der in Abschnitt 3.3.3 beschriebenen Methode in den logarithmischen Raum transformiert und anschließend an das Deep Neural Network übergeben. Dieses bezieht sich auf das entsprechend des Abschnitts 3.8.1 trainierte U-Net mit parallelem globalem Merkmalsnetzwerk. Das Deep Neural Network fertigt somit SVBRDF-Schätzungen für alle vier vorliegenden realen Materialien an. Anhand der resultierenden Materialkarten wird eine Reihe an physikalisch-basierten Neubeleuchtungen vorgenommen. Abbildung 3.24 bildet diese Neubeleuchtungen tabellarisch ab. Hierbei stellt die erste Spalte die in das Netz eingegebenen Materialproben dar. Die verbleibenden Spalten stellen die mittels unterschiedlichen Beleuchtungssetups entstandenen Renderings dar, wobei für eine gegebene Spalte exakt gleiche Beleuchtungsbedingungen für die entsprechenden Materialien vorliegen. Der Fokus der zweiten und dritten Spalte liegt auf dem diffusen Erscheinungsbild, wobei die dritte Spalte ein stark beleuchtetes Szenario darstellen soll. Mit den Beleuchtungsbedingungen der letzten drei Spalten wird das Glanzreflexionsverhalten der jeweiligen Materialien analysiert; gleichzeitig können durch den flachen Betrachtungswinkel, welcher in der letzten Spalte am ausgeprägtesten vorliegt, geometrische Details gut visualisiert werden.

Die Ergebnisse zeigen eine Reihe an realitätskonformen Neubeleuchtungen der ursprünglichen Materialien, wobei das Verhalten des zugrunde liegenden Materials sich entsprechend der Beleuchtungsbedingungen den Erwartungen entsprechend anpasst. So wird in der zweiten Neubeleuchtung der Ledermaterialprobe nicht bloß die Helligkeit des Bildes erhöht; stattdessen ist eine Glanzreflexion in den durch die Konturen

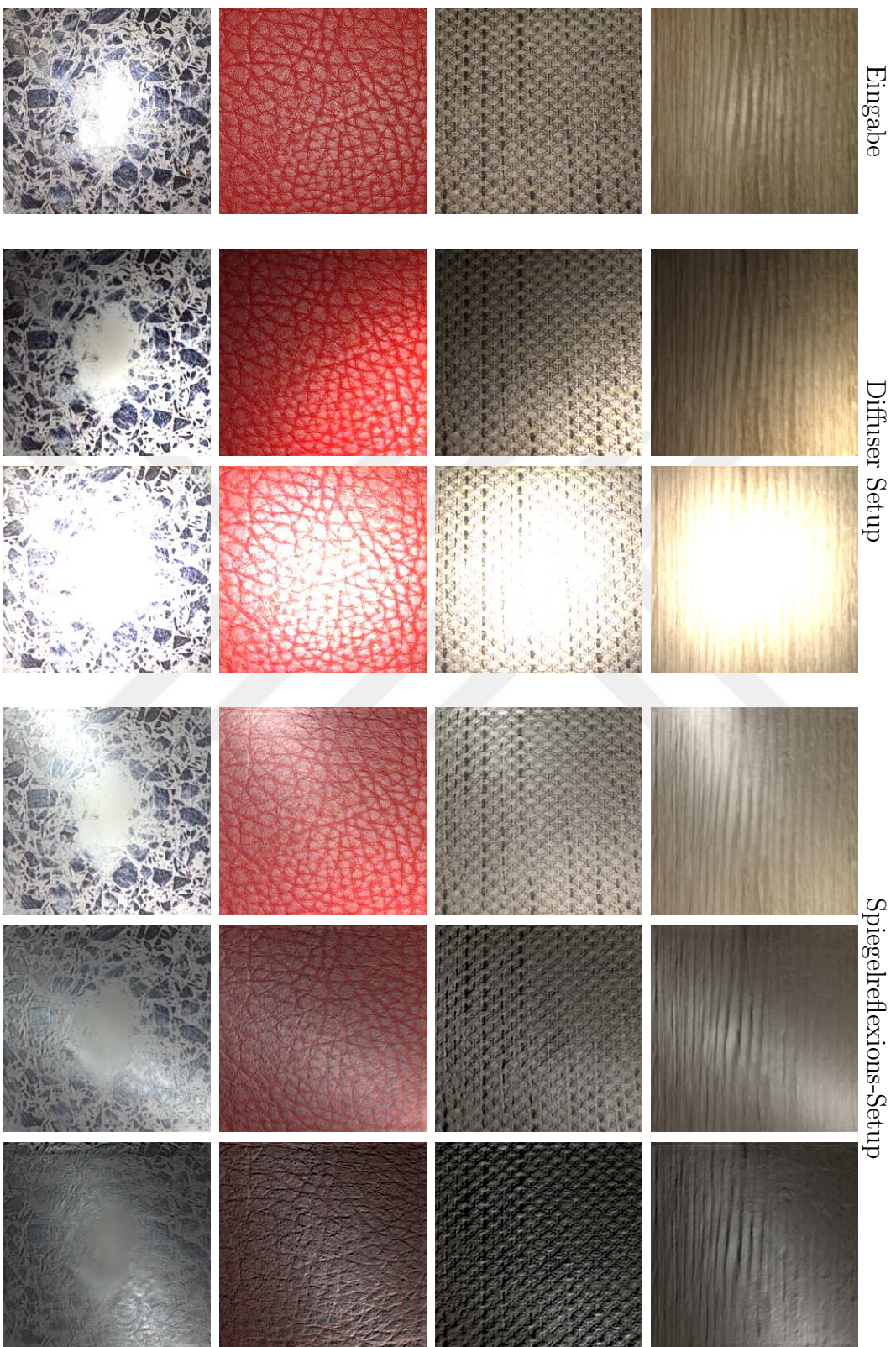


Abbildung 3.24: Neubelichtungen realer Materialien. Die erste Spalte stellt die Eingabe, die Verbleibenden Neubelichtungen dar.

getrennten Bereichen festzustellen, wie sie unter den starken Beleuchtungsbedingungen auftreten sollten. Des Weiteren sind starke geometrische Details in der letzten Neubeleuchtung des Leders besonders gut nachzuvollziehen. Auch die Konturen der Holztischplatte und des Stoffbezuges werden unter diesen Beleuchtungsbedingungen stark sichtbar und weisen realistische Charakteristika auf. Schwierigkeiten treten bei der polierten Steintischplatte auf. Hier wird die Rekonstruktion der Materialinformationen durch das verstärkte Auftreten des reflektierten Blitzlichtes erschwert. Trotz dieser starken Einschränkung ist das Netz in der Lage, das Glanzreflexionsverhalten der Tischplatte zu approximieren, was in den letzten Glanzlichtrenderings besonders gut nachzuvollziehen ist. Des Weiteren werden die Muster der Steintischplatte mit leichten geometrischen Ausprägungen in Verbindung gebracht, die in der letzten Beleuchtungskonfiguration deutlich zu sehen sind; solche geometrischen Details liegen jedoch nicht vor, da es sich bei dieser Materialprobe um eine polierte Oberfläche handelt.

Generell lässt sich festhalten, dass das Deep Neural Network in der Lage ist, grundsätzlich physikalisch-konforme Materialkarten zu produzieren. Basierend auf diesen konnten im Rahmen dieser Analyse eine Reihe an realistischen Neubeleuchtungen angefertigt werden, die das erwartete Materialverhalten realitätskonform nachstellen konnten. Somit ist die Umsetzung der inversen Renderingfunktion, wie sie zu Anfang dieses Kapitels definiert wurde, größtenteils erfolgreich erfolgt. Defizite in stark reflektierenden Materialien lassen sich möglicherweise durch Methoden, die mehrere Eingabefotos für eine SVBRDF-Akquise nutzen, umgehen. Zwei solcher Methoden wurden im Abschnitt 3.6 vorgestellt.

3.8.4 Quantitative Analyse der naiven Datenaugmentation

In diesem Abschnitt wird der quantitative Vorteil eines Trainings mittels einer Datenaugmentation untersucht. Die Augmentationsmethode bezieht sich dabei auf die naive Datenaugmentation (siehe Abschnitt 3.5.3).

Hierbei wird zwischen zwei Datensätzen unterschieden; zum einen gibt es den unaugmentierten Datensatz, der aus den ursprünglichen 200.000 Daten besteht. Zum anderen wird ein augmentierter Datensatz implizit durch die naive Datenaugmentation dargestellt. Untersucht wird die quantitative Leistung des U-Nets und U-Net+ gegenüber dem U-Net+, welches auf dem augmentierten Datensatz trainiert wurde. Die Analyse findet dabei sowohl auf dem unaugmentierten Datensatz, als auch auf dem augmentierten Datensatz statt.

Die jeweiligen Deep Neural Networks werden entsprechend Abschnitt 3.8.1 für 4000

Unaugmentierter Datensatz			
	U-Net	U-Net+	U-Net+ & NDA
L_{Render}	0.16293 ± 0.04	0.14664 ± 0.04	0.27075 ± 0.08
L_{Normal}	0.03088 ± 0.01	0.02822 ± 0.01	0.03443 ± 0.01
L_{Diffuse}	0.01018 ± 0.00	0.00903 ± 0.00	0.01744 ± 0.01
$L_{\text{Roughness}}$	0.07690 ± 0.03	0.07352 ± 0.05	0.09163 ± 0.03
L_{Specular}	0.02183 ± 0.01	0.01781 ± 0.01	0.05338 ± 0.03
Augmentierter Datensatz			
	U-Net	U-Net+	U-Net+ & NDA
L_{Render}	0.41509 ± 0.12	0.40571 ± 0.13	0.17006 ± 0.04
L_{Normal}	0.04383 ± 0.02	0.04402 ± 0.02	0.03086 ± 0.01
L_{Diffuse}	0.03275 ± 0.02	0.02992 ± 0.02	0.01169 ± 0.01
$L_{\text{Roughness}}$	0.20736 ± 0.07	0.23569 ± 0.09	0.08618 ± 0.04
L_{Specular}	0.04191 ± 0.03	0.03952 ± 0.03	0.02537 ± 0.02

Tabelle 3.2: Quantitativer Vergleich der naiven Datenaugmentation mit bisherigen Methoden. Hierbei werden der Renderingfehler L_{Render} und die L1-Fehlerwerte der Materialkarten L_{Normal} , L_{Diffuse} , $L_{\text{Roughness}}$ und L_{Specular} des U-Nets, des U-Net+ und des U-Net+ mit naiver Datenaugmentation gegenübergestellt. Die obere Tabelle stellt resultierende Fehlerwerte auf dem unaugmentierten Datensatz, die untere Tabelle dagegen Fehlerwerte auf dem augmentierten Datensatz dar. Die fett markierten Werte zeigen die jeweils niedrigsten Fehlerwerte an.

Epochen trainiert; das U-Net und U-Net+ werden auf den unaugmentierten 200.000 SVBRDFs samt den im Voraus gerenderten Blitzlichtfotos trainiert. Das Deep Neural Network, welches im Folgenden durch *U-Net+ & NDA* referenziert wird, nutzt die U-Net-Architektur mit globalem Merkmalsnetzwerk; dieses Netzwerk wurde jedoch auf dem mittels der naiven Datenaugmentation augmentierten Datensatz trainiert. Die Evaluierung wird sowohl auf dem unaugmentierten als auch auf dem augmentierten Datensatz durchgeführt. Dabei wird der durchschnittliche Fehlerwert und die Standardabweichung des stochastischen Renderingfehlers und den L1-Fehlern der jeweiligen Materialkarten mit den tatsächlichen Daten festgehalten. Insgesamt wird jedes Deep Neural Network über 1000 SVBRDFs und gerenderten Blitzlichtfotos pro Datensatz ausgeführt.

In der Tabelle 3.2 werden die quantitativen Fehlerwerte festgehalten. Zunächst werden die Fehlerwerte auf dem unaugmentierten Datensatz betrachtet. Das U-Net mit globalem Merkmalsnetzwerk kann hierbei in jedem Testfall die niedrigsten Fehlerwerte aufzeigen, dicht gefolgt von den Werten des U-Nets; die auf dem augmentierten Datensatz trainierte U-Net+-Variante besitzt hierbei einen fast doppelt so hohen Fehlerwert bei dem Renderingfehler und vergleichbare Fehlerwerte in den L1-Fehlern der Materialkarten. Auf dem augmentierten Datensatz dagegen besitzt die augmentierte U-Net-Variante die niedrigsten Fehlerwerte; die unaugmentierten Deep Neural Networks besitzen einen mehr als doppelt so hohen Renderingfehler und wesentlich höhere Fehlerwerte in den Materialkarten, in einigen Fällen ebenfalls um mehr als das Doppelte der augmentierten Variante.

Aus dieser Beobachtung wird die Schlussfolgerung gezogen, dass die Übertragung eines augmentierten U-Nets auf einen unaugmentierten Datensatz mit wesentlich niedrigeren Fehlerwerten durchgeführt werden kann als es bei der Rückrichtung der Fall ist. Diese Schlussfolgerung wird vor allem aus den geringen Fehlerwerten auf den Materialkarten der augmentierten Netzvariante auf dem unaugmentierten Datensatz gezogen und aus den besonders hohen Renderingfehlerwerten der unaugmentierten Netze auf dem augmentierten Datensatz.

Die Diskrepanz der Fehlerwerte zwischen den augmentierten und unaugmentierten Netzvarianten wird auf die unterschiedlichen Renderingkomponenten für die synthetische Erzeugung der Blitzlichtfotos zurückgeführt; in der Arbeit von Deschaintre et al. [11] wird angegeben, dass der *Mitsuba-Renderer*⁴ für das Offline-Rendering der Blitzlichtfotos genutzt wurde. Im Rahmen der naiven Datenaugmentation dagegen wird die differenzierbare Renderingkomponente (siehe Abschnitt 3.4.1) für die ad-

⁴Mitsuba-Renderer <http://www.mitsuba-renderer.org/> – Zugriffsdatum: 4. Oktober 2019

hoc Erzeugung der Blitzlichtfotos eingesetzt. Ein weiterer Aspekt der mangelhaften Übertragbarkeit der unaugmentierten Netzvarianten auf augmentierten Datensätze besteht möglicherweise in einem leichten Overfitting während des Trainings bezüglich des unaugmentierten Datensatzes, das auf einem durch Datenaugmentationsmethoden hinreichend erweiterten Datensatz nicht möglich ist [19].

Abschließend lässt sich festhalten, dass die mittels der naiven Datenaugmentation trainierten U-Nets sich besser auf andere Datensätze übertragen lassen als unaugmentierte Deep Neural Networks auf augmentierte Datensätze. Die naive Datenaugmentation stellt somit einen möglichen ersten Schritt in die Richtung eines robusteren Rendering-gestützten Trainings von Deep Neural Networks dar. Des Weiteren ist die naive Datenaugmentation eine mögliche Lösung für das leichte Overfitting der Netze auf unaugmentierten Datensätzen.

3.8.5 Qualitativer Vergleich mit der Originalarbeit

In diesem Abschnitt wird ein qualitativer Vergleich des im Rahmen dieser Arbeit entwickelten Deep Neural Networks mit dem Netz der Originalarbeit [11] durchgeführt; dazu steht das Deep Neural Network der Arbeit von Deschaintre et al. [11] mit trainierten Gewichten für eine qualitative Evaluierung zur Verfügung.

Die Auswertung wird in zwei Varianten durchgeführt. Zunächst wird die Qualität der Materialkartenschätzungen für ein synthetisches Blitzlichtfoto und eine darauf basierende Neubeleuchtung untersucht; hierbei liegen die tatsächlichen Materialkarten des Blitzlichtfotos vor, was einen solchen Vergleich ermöglicht. Im darauffolgenden Abschnitt wird die Qualität beider Deep Neural Networks durch die Neubeleuchtung realer Materialproben untersucht.

Das Deep Neural Network dieser Arbeit bezieht sich dabei auf das U-Net mit globalem Merkmalsvektor, welches auf den unaugmentierten Daten bestehend aus 200.000 SVBRDFs samt synthetisch erzeugten Blitzlichtfotos trainiert wurde.

3.8.5.1 Vergleich auf einem synthetischen Datum

Im Folgenden wird ein qualitativer Vergleich des U-Nets mit globalem Merkmalsvektor dieser Arbeit und dem trainierten Deep Neural Network der Arbeit von Deschaintre et al. [11] auf einem synthetischen Datum durchgeführt.

Das Training des Deep Neural Networks dieser Arbeit wurde entsprechend des Abschnitts 3.8.1 durchgeführt. Im Vergleich dazu wurde das Deep Neural Network der Arbeit von Deschaintre et al. für 400.000 Iterationen auf einer *TitanX* GPU mit

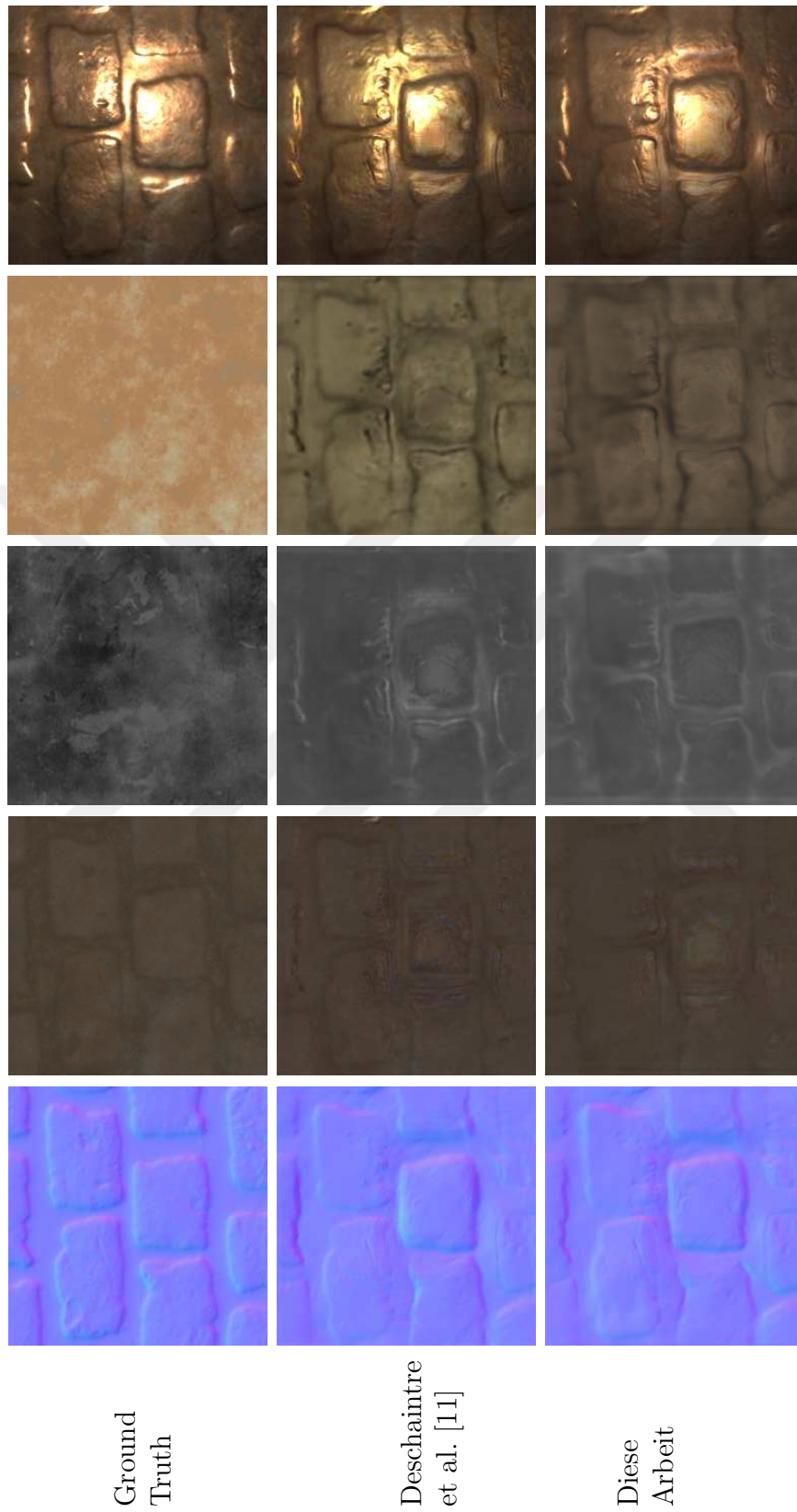


Abbildung 3.25: Qualitativer Vergleich mit der Originalarbeit. Von links nach rechts werden die Karten der Normalen, der diffusen Grundfarbe, des skalaren Rauheitswerts und der Glanzfarbe abgebildet. Die erste Reihe stellt die tatsächlichen Daten, die zweite Reihe die Schätzungen des trainierten Deep Neural Networks von Deschaintre et al. [11] und die letzte Reihe Schätzungen des Netzes dieser Arbeit dar.

einer Lernrate von 0.00002 mittels der Adam-Optimierung [31] durchgeführt; dabei betrug die Trainingszeit bei einer Batchgröße von 8 ca. eine Woche [11].

Als Eingabe für die jeweiligen Deep Neural Networks diente das synthetische Blitzlichtfoto aus dem Abschnitt 3.8.2.1 (siehe Abbildung 3.19). Die SVBRDF-Schätzungen der Deep Neural Networks werden dabei für eine physikalisch-basierte Neubeleuchtung unter denselben Beleuchtungsbedingungen eingesetzt; hierbei wird auch basierend auf den tatsächlichen Materialkarten eine Neubeleuchtung vorgenommen, um eine gerenderte Referenz für den qualitativen Vergleich der Neubeleuchtungen zu generieren. Für eine bessere Visualisierung wurden die Pixelwerte der Karten der diffusen Grundfarbe und der Glanzfarbe mit einem Exponenten von 0.5 versehen.

In Abbildung 3.25 werden die Ergebnisse der SVBRDF-Schätzungen und den darauf basierenden Neubeleuchtungen abgebildet. Die erste Reihe stellt dabei die tatsächlichen Daten, die zweite Reihe die Ergebnisse der Arbeit von Deschaintre et al. [11] und die letzte Reihe die Ergebnisse dieser Arbeit dar. Die Spalten zeigen von links nach rechts die Normalenkarte, die Karte der diffusen Grundfarbe, die Rauheitskarte, die Glanzfarbentkarte und die Neubeleuchtung. Insgesamt weisen alle geschätzten Karten bis auf die Normalenkarte leichte Beleuchtungsrückstände in Form von Flecken auf. Zudem weisen die Schätzungen beider Deep Neural Networks in den Karten des skalaren Rauheitswertes und der Glanzfarbe in den tatsächlichen Daten nicht existente Konturen auf.

Abbildung 3.26 bildet den direkten Vergleich der Neubeleuchtungen ab; hierbei wird

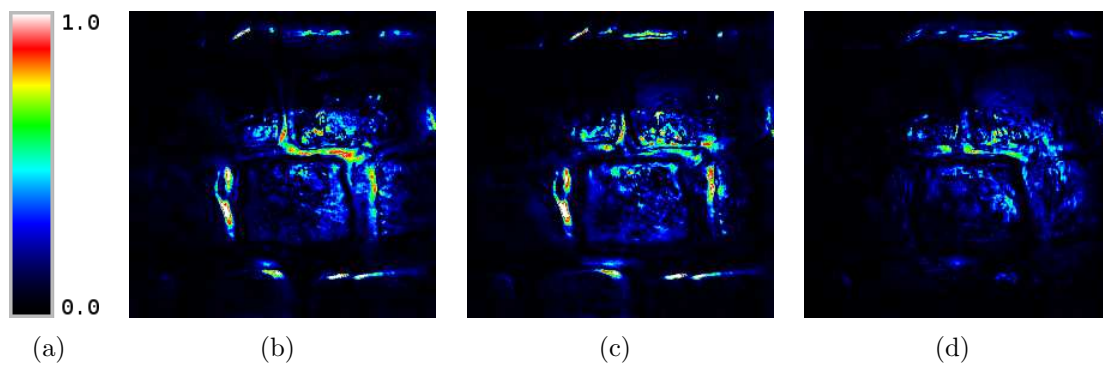


Abbildung 3.26: Direkter Vergleich der Neubeleuchtungen. Abbildung (a) bildet die eingesetzte Farbskala ab. Abbildungen (b) bis (d) stellen die dementsprechend farblich kodierten Vektordistanzen der Farbwerte dar; Abbildung (b) vergleicht die Referenzneubeleuchtung mit der dieser Arbeit, Abbildung (c) die Referenzneubeleuchtung mit der der Originalarbeit und Abbildung (d) die Neubeleuchtungen dieser Arbeit und der Originalarbeit.

die Vektordistanz der Pixelwerte entsprechend der Skala in Abbildung 3.26(a) farblich kodiert. In Abbildung 3.26(b) wird die tatsächliche Neubeleuchtung mit der Neubeleuchtung dieser Arbeit verglichen; die Abweichungen ähneln denen des Vergleiches der tatsächlichen Neubeleuchtung mit der Neubeleuchtung basierend auf den Schätzungen des Deep Neural Networks der Arbeit von Deschaintre et al. [11] (siehe Abbildung 3.26(c)). Die geringsten Abweichungen sind jedoch im Vergleich der Neubeleuchtungen dieser Arbeit und der Originalarbeit festzustellen (siehe Abbildung 3.26(d)).

Generell produzieren beide Deep Neural Networks qualitativ hochwertige Schätzungen und somit realitätsgetreue Neubeleuchtungen. Die Ergebnisse der Deep Neural Networks sind sich dabei im Direktvergleich visuell ähnlicher als zu den tatsächlichen Daten.

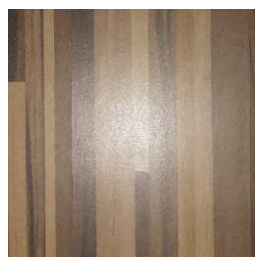
Somit kann festgehalten werden, dass im Rahmen dieses Kapitels die Methoden der Arbeit von Deschaintre et al. [11] erfolgreich nachgebildet werden konnten.

Im nächsten Abschnitt wird ein Vergleich auf realen Materialproben durchgeführt.

3.8.5.2 Vergleich auf realen Materialproben

In diesem Abschnitt wird ein qualitativer Vergleich der Umsetzung dieser Arbeit mit der von Deschaintre et al. [11] bezüglich realer Materialproben durchgeführt.

Dazu werden zwei grundsätzlich unterschiedliche Materialproben entsprechend des Abschnitts 3.8.3 mittels der Smartphonekamera akquiriert, skaliert und zurechtgeschnitten, sodass die Bilder am Ende eine Größe von 256×256 Pixeln besitzen.



(a) Laminat mit Holzmuster



(b) Ledereinband mit Stoffverschlussband

Abbildung 3.27: Blitzlichtaufnahmen realer Materialien für den qualitativen Vergleich. Die Abbildung (a) stellt einen Laminatboden mit Holzmuster dar. Abbildung (b) zeigt die Vorderseite eines Notizbuchs mit Ledereinband und einem Stoffverschlussband. Die Materialien besitzen grundsätzlich unterschiedliche Reflexionseigenschaften; die letztere Materialprobe besteht aus zwei unterschiedlichen Materialien.

Abbildung 3.27 stellt die zwei akquirierten realen Materialien dar. Das erste Material zeigt einen Laminatboden mit einem Holzmuster, das zweite Material dagegen einen Ledereinband eines Notizbuchs mit einem Stoffverschlussband in der Mitte; die letztere Probe setzt sich aus zwei unterschiedlichen Materialien zusammen. Die Materialproben weisen dabei grundsätzlich unterschiedliche Lichtreflexionseigenschaften auf. Das Laminat besitzt einen geringeren Rauheitsgrad und weist zudem wenig geometrische Details auf. Der Ledereinband besitzt eine detailreiche glänzende Oberfläche, die im Kontrast zu dem Stoffverschlussband steht, welches eine schwache Glanzlichtkomponente aufweist.

Die zugeschnittenen Materialproben werden den trainierten Deep Neural Networks dieser Arbeit und der von Deschaintre et al. [11] als Eingabe übergeben. Die resultierenden SVBRDF-Schätzungen werden unter verschiedenen Beleuchtungsbedingungen, die sowohl das diffuse Reflexionsverhalten als auch das Glanzreflexionsverhalten über Spiegelreflexions-Setups festzuhalten abzielen, neu beleuchtet.

Abbildung 3.28 zeigt die Ergebnisse dieser Neubeleuchtungen. Dabei stellt die erste Spalte die Eingaben, die restlichen Spalten die synthetisierten Bilder mit unterschiedlichen Beleuchtungssetups dar; für je eine solche Spalte herrscht dabei das exakt selbe Beleuchtungssetup vor. Ausgehend von der zweiten Spalte stellen die erste und die dritte Reihe die Ergebnisse des Deep Neural Networks der Arbeit von Deschaintre et al. [11] und die zweite und vierte Spalte die Ergebnisse des im Rahmen dieser Arbeit entstandenen Netzes dar.

Die Neubeleuchtungen spiegeln ein generell realistisches Lichtreflexionsverhalten wider. So besitzt das Laminat eine größtenteils glatte Oberfläche und der Ledereinband und das Stoffverschlussband werden als grundsätzlich unterschiedliche Materialien erkannt und mit unterschiedlichen Reflexionsverhalten geschätzt. Jedoch sind in einigen Szenarien klare Beleuchtungsrückstände in Form von Flecken auf den Neubeleuchtungen des Laminats zu erkennen, wobei die Verfärbungen bei den Ergebnissen der Originalarbeit wesentlich deutlicher zu sehen sind (siehe dritte, vierte und fünfte Spalte). Zudem zeigt das Laminat im letzten Beleuchtungssetup starke geometrische Konturen auf, die in dem realen Material nicht existieren. Dieser Effekt ist bei den Schätzungen des Netzes dieser Arbeit kaum zu sehen. Die Neubeleuchtungen des Leders sind in den Ergebnissen der Originalarbeit zum oberen Rand hin stark verschwommen (siehe vierte und fünfte Spalte); auch dieser Effekt tritt in den Ergebnissen dieser Arbeit kaum auf. Im letzten Beleuchtungsszenario ist zu sehen, dass die Geometrie des Stoffverschlussbands als in den Ledereinband eingeprägt geschätzt wird; diese Fehleinschätzung ist in beiden Ergebnissen, jedoch in den Ergebnissen

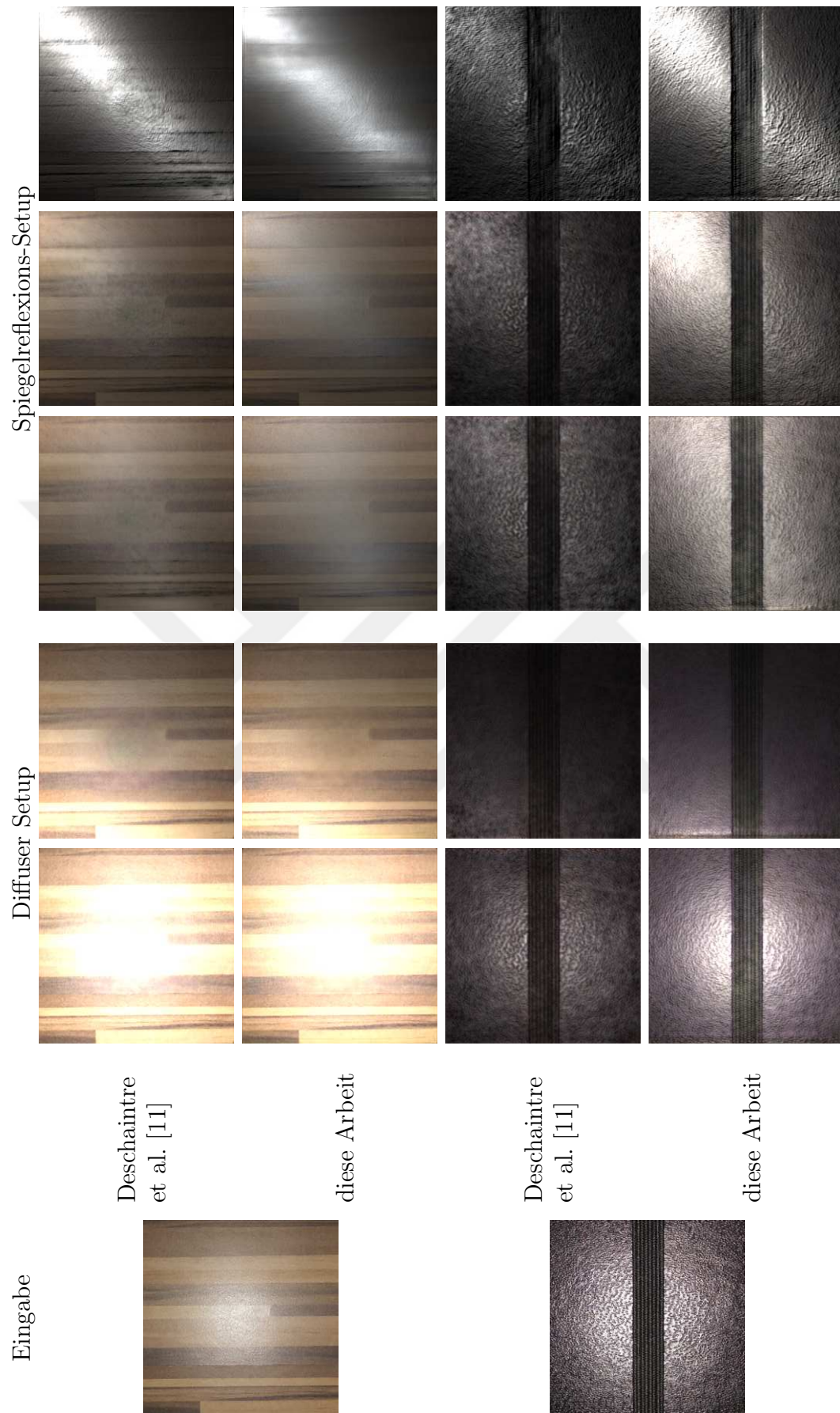


Abbildung 3.28: Neubeleuchtungen realer Materialien im Vergleich

dieser Arbeit verstärkt aufzuweisen.

Insgesamt lässt sich festhalten, dass das Deep Neural Network dieser Arbeit die SV-BRDFs der Materialproben hinreichend approximiert und so realitätsgetreue Neubeleuchtungen ähnlich der Arbeit von Deschaintre et al. [11] zulässt. In den meisten Fällen ist diese Neubeleuchtung qualitativ besser und weniger durch Artefakte geprägt als die Originalarbeit. Speziell Unschärfe und Beleuchtungsrückstände treten in den Ergebnissen dieser Arbeit mit geringerer Intensität auf.

Diese Unterschiede lassen sich möglicherweise auf minimale Parameterunterschiede während des Trainings – beispielsweise der unterschiedlichen Lernrate von 0.00001 statt 0.00002 – und andere minimale Implementierungsdetails zurückführen. Zudem ist das Training eines Deep Neural Networks selbst bei exakt gleicher Architektur nicht zwangsweise deterministisch. Auch wurde in dieser Arbeit ein anderes Framework eingesetzt; so setzte die Arbeit von Deschaintre et al. [11] das TensorFlow-Framework [1] ein, wohingegen diese Arbeit auf das PyTorch-Framework⁵ setzte (siehe Abschnitt 3.7).

Abschließend lässt sich festhalten, dass diese Arbeit die Verfahren der zugrunde liegenden Originalarbeit von Deschaintre et al. [11] nachbilden und qualitativ in einigen Aspekten übertreffen konnte.

⁵PyTorch <https://pytorch.org/> – Zugriffsdatum: 7. Oktober 2019

4 Inverses Rendering für räumliche Szenen

In diesem Kapitel wird die inverse Renderingfunktion für räumliche Szenen vorgestellt und ein Lösungsansatz für das inverse Rendering solcher Szenen verfolgt. Das Kapitel baut dabei auf den Verfahren des letzten Kapitels bezüglich der inversen Renderingfunktion für planare Oberflächen auf und passt vorgestellte Verfahren entsprechend an.

Der Abschnitt 4.1 gibt dabei eine formale Definition der Problemstellung dieses Kapitels an. Abschnitt 4.2 stellt eine Übersicht des entwickelten Lösungsansatzes vor. Im Abschnitt 4.3 wird die angepasste Netzarchitektur des Deep Neural Networks für die Approximation der inversen Renderingfunktion für räumliche Szenen erläutert. Der Abschnitt 4.4 stellt die differenzierbare Renderingkomponente vor, die das Rendern räumlicher Szenen ermöglicht und die Grundlage für das Rendering-gestützte Fehlermaß des darauffolgenden Abschnitts 4.5 darstellt. Regularisierungsmethoden für diese Fehlerfunktionen werden in Abschnitt 4.6 eingeführt. Abschließend wird in Abschnitt 4.8 eine Evaluierung der vorgestellten Verfahren durchgeführt.

4.1 Definition

In diesem Abschnitt wird aufbauend auf der formalen Definition der Problemstellung des vorausgehenden Kapitels die zu lösende Problemstellung dieses Kapitels formuliert und eine Relation zu der vorausgehenden Problemstellung aufgebaut.

Bei planaren Oberflächen liegen alle Oberflächenpunkte auf einer Ebene und geometrische Details werden anhand der Normalenkarte nachgebildet. Im Gegensatz dazu besitzt jedes Pixel eines Bildes, welches eine räumliche Szene darstellt, eine korrespondierende Position im Raum, die somit die Geometrie der Szene repräsentiert. Zusätzlich können Oberflächendetails weiterhin über die Normalenkarte definiert werden. Der fundamentale Unterschied zwischen räumlichen Szenen und planaren Oberflächen besteht für diese Arbeit darin, dass durch die mögliche unterschiedliche

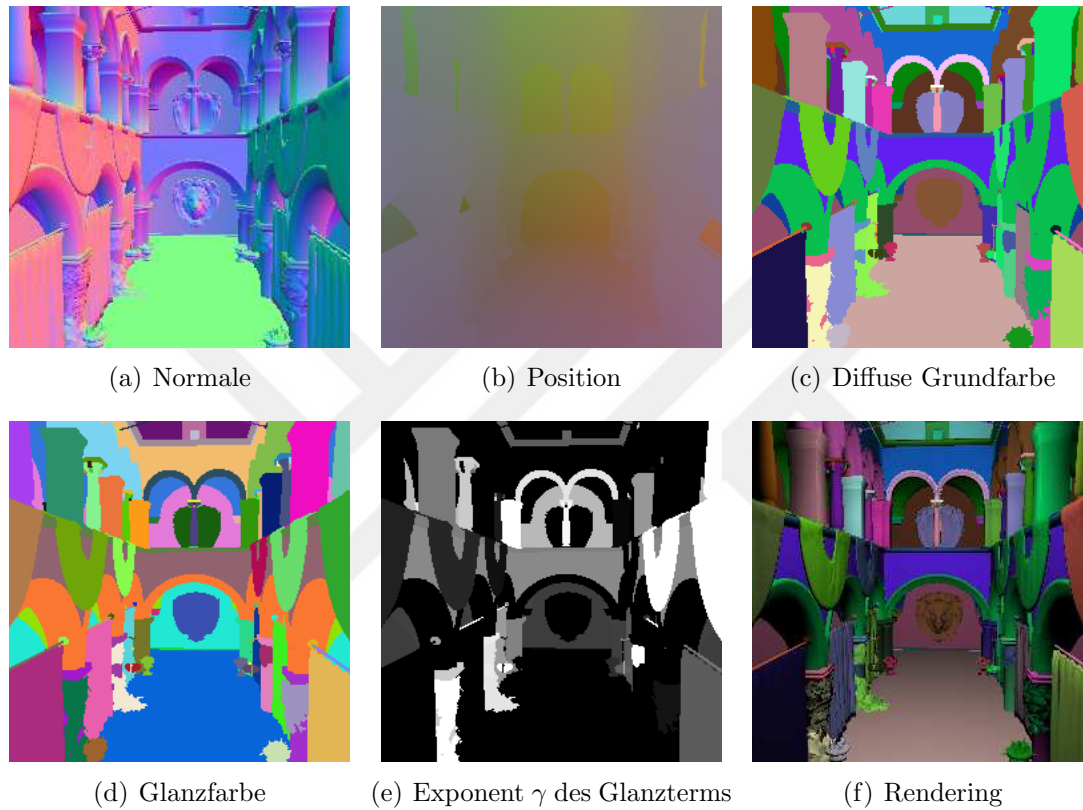


Abbildung 4.1: Datum einer räumlichen Szene [46]. Die Daten sind dabei für das Phong-Beleuchtungsmodell [58] ausgelegt und beinhalten neben (a) der Normale und (b) der Pixelposition im Raum (c) die diffuse Grundfarbe, (d) die Glanzfarbe und (e) den Exponenten γ des Glanzterms. Abbildung (f) stellt ein Rendering basierend auf dem diffusen Term des Phong-Modells dar.

Pixelpositionierung von benachbarten Pixeln der Effekt des Lichtabfalls basierend auf dem Abstandsgesetz [7] natürlich erzeugt werden und so der Eindruck von räumlicher Tiefe konsistent entstehen kann.

Sei $\mathbf{M} \in \mathcal{M}$ mit $\mathcal{M} \subset \mathbb{R}^{m \times n \times 10}$ ein Satz an Materialkarten (siehe Abbildung 4.1), der für jedes Pixel an der Position (i, j) die Materialinformationen $\mathbf{m}_{i,j} \in \mathbf{M}$ mit $\mathbf{m}_{i,j} = (\mathbf{m}_{i,j}^{(n)}, \mathbf{m}_{i,j}^{(d)}, \mathbf{m}_{i,j}^{(s)}, m_{i,j}^{(g)})$ festhält, wobei $m \times n$ mit $m, n \in \mathbb{N}$ die Größe der Materialkarten beschreibt und $\mathbf{m}_{i,j}^{(n)} \in \mathbb{R}^3$ die Normale, $\mathbf{m}_{i,j}^{(d)} \in \mathbb{R}^3$ die diffuse Grundfarbe, $\mathbf{m}_{i,j}^{(s)} \in \mathbb{R}^3$ die Glanzfarbe und $m_{i,j}^{(g)} \in \mathbb{R}$ den Exponenten des Glanzterms [58] des Pixels darstellen. Des Weiteren sei $\mathbf{P} \in \mathcal{P}$ mit $\mathcal{P} \subset \mathbb{R}^{m \times n \times 3}$ die zu der Szene korrespondierende Positionskarte, welche die zur Kameraposition relativen Pixelpositionen $\mathbf{p}_{i,j} \in \mathbf{P}$ im Raum festhält. Die Renderingfunktion $R : \mathcal{P} \times \mathcal{M} \times \mathcal{L} \times \mathcal{V} \rightarrow \mathcal{I}$ bildet darauf basierend ein beleuchtetes Bild $\mathbf{I} \in \mathcal{I}$ mit $\mathcal{I} \subset \mathbb{R}^{m \times n \times 3}$ ab, wobei $\mathbf{l} \in \mathcal{L}$ mit $\mathcal{L} \subset \mathbb{R}^3$ eine Lichtquellenposition und $\mathbf{v} \in \mathcal{V}$ mit $\mathcal{V} \subset \mathbb{R}^3$ einen Augenpunkt darstellen. Die ideale inverse Renderingfunktion wird im Rahmen dieses Kapitels als $Q : \mathcal{I} \times \mathcal{P} \rightarrow \mathcal{M}$ definiert, die ausgehend von dem beleuchteten Bild $\mathbf{I}$ und der dazu korrespondierenden Positionskarte $\mathbf{P}$ auf einen Satz an Materialkarten $\mathbf{M}$ abbildet, wobei die Lichtquellen- und Augenpunktpositionen $\mathbf{l}$ und $\mathbf{v}$ nicht von Interesse sind. Entsprechend des vorausgehenden Kapitels ist eine Funktion Q , welche die Zielfunktion

$$\min_Q \mathbb{E}_{(\mathbf{l}, \mathbf{v})} \delta(R(\mathbf{P}, \mathbf{M}, \mathbf{l}, \mathbf{v}), R(\mathbf{P}, Q(\mathbf{I}, \mathbf{P}), \mathbf{l}, \mathbf{v})) \quad (4.1)$$

optimiert, eine inverse Renderingfunktion.

Im Folgenden wird die Relation des inversen Renderings räumlicher Szenen zum inversen Rendering planarer Oberflächen und analog dazu die Relation der Renderingfunktionen beider Varianten vorgestellt. Dafür wird für den restlichen Teil dieses Abschnitts die folgende Notation übernommen;

- die inverse Renderingfunktion planarer Oberflächen wird fortan mit dem Bezeichner Q_{2D} referenziert,
- die inverse Renderingfunktion räumlicher Szenen wird dementsprechend Q_{3D} genannt,
- die Renderingfunktion planarer Oberflächen wird mittels R_{2D} annotiert und
- die Renderingfunktion räumlicher Szenen wird durch R_{3D} repräsentiert.

Es gilt

$$R_{3D}(\mathbf{P}_{2D}, \mathbf{M}, \mathbf{l}, \mathbf{v}) = R_{2D}(\mathbf{M}, \mathbf{l}, \mathbf{v}), \quad (4.2)$$

wobei $\mathbf{P}_{2D}$ eine konstante Positionskarte einer planaren Oberfläche darstellt und somit die Funktion R_{2D} einen Spezialfall von R_{3D} beschreibt. Des Weiteren gilt

$$Q_{3D}(\mathbf{P}_{2D}, \mathbf{I}) = Q_{2D}(\mathbf{I}), \quad (4.3)$$

wobei auch hier Q_{2D} einem Spezialfall von Q_{3D} entspricht. Eine Voraussetzung besteht hierbei darin, dass alle Varianten sich auf dasselbe Beleuchtungsverfahren und somit auf dieselbe Struktur der Materialkarten beziehen.

An dieser Stelle wird nochmals darauf hingewiesen, dass die Approximation von Q_{2D} eine mathematisch gesehen schlecht gestellte Aufgabe darstellt [11, 38, 39]. Folglich stellt auch die Approximation von Q_{3D} eine schlecht gestellte Aufgabe dar.

Ein Lösungsansatz, der die inverse Renderingfunktion für räumliche Szenen approximiert, wird aufbauend auf dem Ansatz des Kapitels 3 im nächsten Abschnitt vorgestellt.

4.2 Lösungsansatz

In diesem Abschnitt wird ein Überblick über den Lösungsansatz für das inverse Rendering räumlicher Szenen gegeben. Der Lösungsansatz orientiert sich am vorausgehenden Kapitel und stellt somit eine Übertragung der Verfahren auf räumliche Szenen dar.

Wie im letzten Abschnitt bereits erwähnt wurde, stellt das inverse Rendering für räumliche Szenen ebenfalls eine mathematisch schlecht gestellte Aufgabe dar. Aufbauend auf dem datenbasierten Ansatz des vorausgehenden Kapitels können auch hier *a priori* Annahmen durch ein Deep Neural Network implizit gelernt werden. In diesem Kapitel wird die U-Net-Architektur ohne globalem Merkmalsnetzwerk (siehe Abschnitt 3.3.1) mit minimalen Anpassungen für die Approximation der inversen Renderingfunktion für räumliche Szenen eingesetzt; die anwendungsspezifische Erweiterung des letzten Kapitels hat dabei für diesen Lösungsansatz keine Vorteile aufzeigen können und wurde folglich in diesem Kapitel nicht eingesetzt. Die finale Architektur für dieses Kapitel wird in Abschnitt 4.3 vorgestellt.

Zusätzlich werden zwei Einschränkungen bezüglich der Aufgabenstellung eingeführt. Zum einen wird die Positionskarte $\mathbf{P}$ als gegeben betrachtet; die Rekonstruktion einer Positionskarte bzw. Tiefenkarte aus einem Foto [36] ist nicht Fokus dieser Arbeit. Zum anderen wird hier ausschließlich die Approximation des diffusen Reflexionsverhaltens einer räumlichen Szene betrachtet. Die Eingabe in die inverse Renderingfunk-

tion besteht somit aus einem Bild $\mathbf{I}$ und der dazu korrespondierenden Positionskarte $\mathbf{P}$; eine Schätzung der Materialkarten $\mathbf{M}$ bezüglich der Normalen und der diffusen Grundfarbe stellt die Ausgabe dar. Somit beziehen sich die Materialeigenschaften eines Pixels an der Position (i, j) im Folgenden auf $\mathbf{m}_{i,j} = (\mathbf{m}_{i,j}^{(n)}, \mathbf{m}_{i,j}^{(d)})$, wobei $\mathbf{m}_{i,j}^{(n)}$ die Normale und $\mathbf{m}_{i,j}^{(d)}$ die diffuse Grundfarbe des entsprechenden Pixels darstellt.

Der zentrale Datensatz dieses Kapitels bezieht sich hierbei auf räumliche Szenen [46], die aus einer Positionskarte $\mathbf{P}$ und einer SVBRDF, dargestellt durch einen Satz an Materialkarten $\mathbf{M}$, bestehen. Basierend auf diesen Informationen kann ein Renderer ein beleuchtetes Bild $\mathbf{I}$ synthetisieren; gemeinsam stellt $\mathbf{I}$ mit der Positionskarte $\mathbf{P}$ und den Materialkarten $\mathbf{M}$ ein Datum dar. Abbildung 4.1 bildet ein solches Datum ab. Die Daten sind dabei für das Phong-Beleuchtungsmodell ausgelegt [46]. Das Eingabebild $\mathbf{I}$ wird dabei mittels einer fest positionierten Lichtquelle und einem ebenfalls fest positionierten Augenpunkt ad-hoc online generiert, da keine tatsächlichen Renderings im entsprechenden Format vorliegen.

Auch in diesem Kapitel wird ein Rendering-gestütztes Fehlermaß eingesetzt. Aufbauend auf dem Fehlermaß des letzten Kapitels bezüglich planarer Oberflächen werden für dieses Kapitel einige Anpassungen spezifisch für die Evaluierung von räumlichen Szenen vorgenommen. Zum einen wird eine differenzierbare Renderingkomponente für räumliche Szenen benötigt; diese muss zudem die SVBRDF des Datensatzes unterstützen. Eine solche Komponente wird in Abschnitt 4.4 eingeführt. Zum anderen muss eine neue Positionierungsregel bezüglich der Lichtquellen im Raum für den stochastischen Aspekt des Fehlermaßes entwickelt werden. Ein solches Verfahren wird in Abschnitt 4.5 vorgestellt. Des Weiteren wird in Abschnitt 4.6 eine anwendungsspezifische Regularisierungsmethode erläutert. Gemeinsam ermöglicht es dieser Ansatz, die vorgestellte inverse Renderingfunktion für räumliche Szenen zu approximieren (siehe Abschnitt 4.8). Im folgenden Abschnitt wird die angepasste Netzarchitektur vorgestellt.

4.3 Netzarchitektur

In diesem Abschnitt werden die Anforderungen an eine Architektur sowie die umgesetzte Struktur einer solchen Architektur eines Deep Neural Networks für die Approximation der inversen Renderingfunktion für räumliche Szenen vorgestellt. Diese baut auf der Architektur des letzten Kapitels aus dem Abschnitt 3.3.1 auf.

Ein Datum einer räumlichen Szene ist in Abbildung 4.1 abgebildet. Dabei besteht eine räumliche Szene aus mehreren uniform gefärbten geometrischen Objekten.

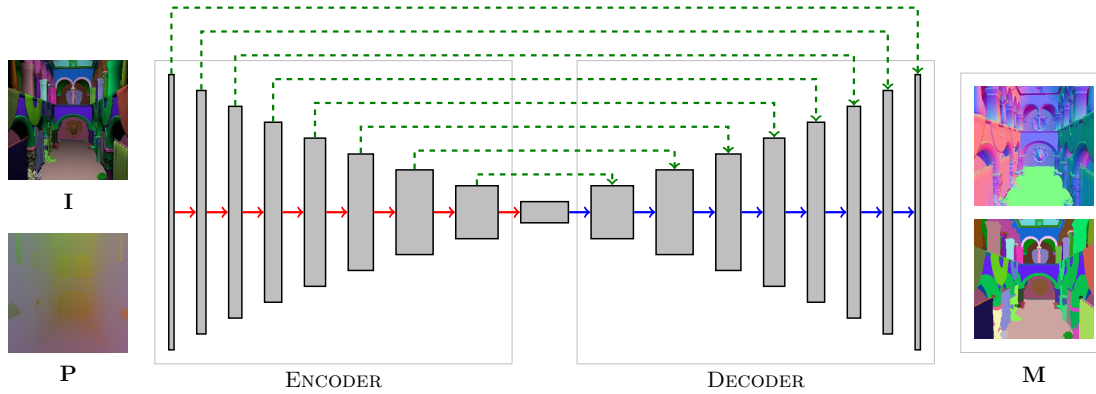


Abbildung 4.2: Angepasste U-Net-Architektur für die inverse Renderingfunktion für räumliche Szenen. Das U-Net [63] besteht aus einem Encoder und einem Decoder mit Skipconnections, die korrespondierende Schichten des Encoders mit denen des Decoders mittels einer Konkatination verbinden. Die Eingabe besteht aus dem beleuchteten Rendering **I** und der dazu korrespondierenden Positionskarte **P**, die Ausgabe aus einem Satz **M** an Materialkarten bestehend aus der Normalenkarte und der diffusen Grundfarbe.

Sei $\mathbf{i}_{i,j}$ der Farbwert eines Pixels des Eingabebildes **I** an der Position (i, j) und $\mathbf{p}_{i,j}$ die dazu korrespondierende Position des Pixels im Raum relativ zu der Positionierung der Kamera. Hierbei ist die positive z-Richtung ausgehend von der Projektionsfläche zur Kamera ausgerichtet, die x-Achse horizontal von links nach rechts und die y-Achse vertikal von unten nach oben. Ausgehend von diesen Eingaben soll das Deep Neural Network für jedes Pixel der Eingabe sowohl die Normale $\mathbf{m}_{i,j}^{(n)}$ als auch die diffuse Grundfarbe $\mathbf{m}_{i,j}^{(d)}$ als Ausgabe liefern.

Die Dimension der Eingabefotos sowie der zu ausgebenden Materialkarten sind dabei identisch und betragen 256 Pixel, sowohl in die Höhe als auch in die Breite. Es wird folglich eine Architektur benötigt, die eine Übersetzung von einem Bild auf ein anderes mit denselben Dimensionen, einer sogenannten *Image-to-Image Translation* [23] durchführen kann. Des Weiteren wird hier die Annahme aufgestellt, dass benachbarte Pixel gemeinsam Informationen über die diffuse Grundfarbe eines Objektes und weit voneinander entfernte Regionen gemeinsam Informationen über die Lichtquelle und somit deren Intensität an einem gegebenen Pixel liefern können. Die Verarbeitung solcher Informationen könnte von einem Deep Neural Network implizit gelernt und für die Approximation der Normalenkarte und diffusen Grundfarbe genutzt werden [11]. Dazu wird eine Architektur benötigt, die das rezeptive Feld, welches den sichtbaren Bereich der Eingabe für ein Pixel der Ausgabe beschreibt, auf das gesamte Eingabebild ausbreitet [39].

Eine Encoder-Decoder-Architektur kann sowohl die benötigte Bild-zu-Bild-Über-

setzung [23] als auch die notwendige Größe des rezeptiven Felds [39] umsetzen. Folglich wird auch in diesem Kapitel eine U-Net-Architektur [63] für die inverse Renderingfunktion eingesetzt. Die Parameter und der Aufbau der jeweiligen Schichten entsprechen dabei der Architektur der Arbeit von Deschaintre et al. [11]; diese wurde bereits detailliert in dieser Arbeit beschrieben (siehe Abschnitt 3.3.1). Hierbei wird die Variante ohne dem globalen Merkmalsnetzwerk als Grundlage für das Deep Neural Network dieses Kapitels gewählt.

Im Folgenden werden die Anpassungen, die für das hier definierte inverse Rendering räumlicher Szenen benötigt werden, vorgestellt. Abbildung 4.2 bildet dabei die Architektur sowie die Eingaben und Ausgabe des Deep Neural Networks schematisch ab.

Zunächst wird die erste Schicht der Architektur für die Eingabe angepasst. Dabei wird auch hier die Notation $256 \sim 64$ genutzt, um die Form eines Bildes mit der Breite und Höhe von 256 Pixeln und 64 Kanälen anzugeben. Die Eingabe besteht aus einem beleuchteten RGB-Bild $\mathbf{I}$ mit drei Kanälen und einer Positionskarte, welche die Position eines gegebenen Pixels relativ zur Kameraposition festhält und somit ebenfalls drei Kanäle besitzt. Die erste Faltung der ersten Schicht wird dementsprechend so angepasst, dass diese ein Bild der Form $256 \sim 6$ auf die Form $128 \sim 64$ abbildet. Das resultierende Bild besitzt somit die Form der entsprechenden Schicht in der ursprünglichen U-Net-Architektur, weswegen weitere Anpassungen an dieser Stelle nicht notwendig sind.

Die Ausgabe besteht im Rahmen dieses Kapitels lediglich aus der Normalenkarte und der Materialkarte für die diffuse Grundfarbe; im Gegensatz zu der ursprünglichen U-Net-Architektur werden hier keine Glanzfarbenseite und Rauheitskarte ausgegeben. Folglich reduziert sich die Anzahl der Kanäle der Ausgabe von zehn auf sechs. Die letzte Faltungsschicht der Architektur produziert somit die finale Ausgabe der Form $256 \sim 6$; die Parameter der Faltungsoperation werden dementsprechend angepasst.

Das so resultierende U-Net bildet die Architektur, die im Rahmen dieses Kapitels für die Approximation der inversen Renderingfunktion für räumliche Szenen genutzt wird. Im nächsten Abschnitt wird die differenzierbare Renderingkomponente vorgestellt, welche die Ausgaben dieses Deep Neural Networks annehmen und darauf basierend ein beleuchtetes Bild synthetisieren kann.

4.4 Differenzierbarer Renderer für räumliche Szenen

In diesem Abschnitt wird die differenzierbare Renderingkomponente des Abschnitts 3.4.1 für das Rendern räumlicher Szenen angepasst. Da die Differenzierbarkeit sowie das Beleuchtungsverfahren für eine Menge von Punktlichtquellen bereits detailliert beschrieben wurde, liegt der Fokus dieses Abschnitts auf dem angepassten Beleuchtungsmodell.

Auch in diesem Kapitel wird auf das Verfahren der G-Buffer [65] zurückgegriffen, bei denen die für die Beleuchtung relevanten Informationen wie die Normalen oder die Position der Pixel in Puffern gespeichert werden. Diese G-Buffer werden in diesem Abschnitt durch die Positionskarte $\mathbf{P}$ und den Satz $\mathbf{M}$ an Materialkarten, bestehend aus der Normalenkarte und der Karte für die diffuse Grundfarbe, dargestellt.

Ein besonderer Aspekt besteht darin, dass im vorausgehenden Kapitel für jedes Pixel der planaren Oberfläche ein korrespondierender Punkt im Raum bestimmt werden musste. Ausgehend von diesem berechneten Punkt, der sich auf derselben Ebene wie alle anderen Punkte der planaren Oberfläche befindet, werden die Vektoren für die relative Lichtquellenrichtung $\mathbf{l}$ und die relative Augenpunkttrichtung $\mathbf{v}$ berechnet und an die Beleuchtungsfunktion übergeben. Da für jedes Pixel der räumlichen Szene bereits eine Position $\mathbf{p}_{i,j} \in \mathbf{P}$ existiert, müssen hier keine weiteren Anpassungen für das Beleuchtungsverfahren gemacht werden.

Die Materialkarten sind für das Phong-Beleuchtungsmodell [58] ausgelegt [46]. In diesem Kapitel wird das Blinn-Phong-Modell [5] eingesetzt; dieses stellt eine Erweiterung des Phong-Modells dar, bei dem der Vektor $\mathbf{h}$ statt dem Reflexionsvektor zum Einsatz kommt und somit die Laufzeitkosten gesenkt werden [21] (siehe Abschnitt 2.1.3.1).

Da hier jedoch aufgrund der anfangs erwähnten Einschränkungen nur das diffuse Erscheinungsbild von Interesse ist, wird der Glanzterm für das Training deaktiviert und das eingesetzte Beleuchtungsmodell reduziert sich auf den diffusen Term entsprechend

$$f_r(i, j, \mathbf{n}, \mathbf{l}, \mathbf{v}) = \mathbf{d} \max(0, \mathbf{n} \cdot \mathbf{l}), \quad (4.4)$$

wobei für ein Pixel an der Position (i, j) der Vektor $\mathbf{i}_{i,j}$ die resultierende Farbe, $\mathbf{d}$ die diffuse Grundfarbe entsprechend $\mathbf{m}_{i,j}^{(d)}$, $\mathbf{n}$ den Normalenvektor entsprechend $\mathbf{m}_{i,j}^{(n)}$ und $\mathbf{l}$ die Lichtquellenrichtung ausgehend von der zum Pixel korrespondierenden Position $\mathbf{p}_{i,j}$ im Raum angibt [5]. Der Parameter $\mathbf{v}$, der die relative Richtung zum Augenpunkt beschreibt, wird hierbei ignoriert.

Die max-Operation mit null als erstem Operanden entspricht dabei exakt der ReLU-

Funktion [81]; da sonst nur differenzierbare Operatoren eingesetzt werden, ist die so resultierende Komponente als ganzes differenzierbar, sofern sich die Lichtquellenposition vor der Oberfläche eines Pixel befindet; ansonsten wird der Gradient auf null gesetzt, da die ReLU-Funktion in diesem Fall eine Eingabe kleiner null bekommt und somit stets null zurück gibt. Insbesondere können die Werte im Gegensatz zu der unangepassten Version der differenzierbaren Renderingkomponente des vorausgehenden Kapitels keine ungültigen Werte annehmen, weswegen an dieser Stelle keine zusätzlichen Anpassungen benötigt werden. Ungültige Werte beziehen sich in diesem Kontext auf Ergebnisse einer Nullteilung.

Im nächsten Abschnitt wird ein stochastisches Fehlermaß basierend auf der hier angepassten differenzierbaren Renderingkomponente vorgestellt.

4.5 Stochastisches Rendering-gestütztes Fehlermaß

Dieser Abschnitt stellt die Anpassungen des stochastischen Fehlermaßes aus dem Abschnitt 3.4.2 vor, die in einem Rendering-gestützten Fehlermaß für räumliche Szenen resultieren. Die Anpassungen beziehen sich dabei auf die Positionierungen der Lichtquellen und Augenpunkte im Raum, sodass die Szene für ein effektives Training ausgeleuchtet werden kann.

Ein direkter Vergleich der vom Deep Neural Network geschätzten und den aus dem Datensatz entstammenden tatsächlichen Materialkarten mittels des L1-Fehlermaßes bezieht die Interaktion der Materialinformationen im Rahmen des Renderingprozesses nicht mit ein; des Weiteren korrelieren numerische Differenzen in den Materialkarten nicht direkt mit den Farbwerten im gerenderten Erscheinungsbild, weswegen die L1-Norm ein schlechtes Fehlermaß für den Vergleich der Materialkarten darstellt [11]. Um die geschätzten Materialkarten mit den tatsächlichen Materialkarten der Szene effektiv vergleichen zu können, werden deswegen beide Szenen zunächst unter denselben Beleuchtungsbedingungen gerendert und anschließend der durchschnittliche L1-Fehler der Logarithmen zwischen den Renderings gebildet. Dieses Fehlermaß erlaubt die Evaluierung der Materialkarten anhand des Reflexionsverhaltens und ermöglicht eine bessere Bewertung der Schätzungen des Deep Neural Networks [11]. Im vorausgehenden Kapitel wurden die Lichtquellen mittels der Kosinusverteilung positioniert, um das Beleuchtungsvorkommen bei der Aufnahme realer Materialien auf planaren Oberflächen nachzubilden [11]. Zusätzlich wurden einige Renderings speziell für die Analyse des Glanzlichtverhaltens im Rahmen des stochastischen Fehlermaßes des letzten Kapitels angefertigt. Beide Aspekte stellen im Rahmen dieses

Kapitels keine Relevanz mehr dar; zum einen handelt es sich hierbei nicht länger um planare Oberflächen, weswegen die Kosinusverteilung den Raum nicht länger ausreichend ausleuchten kann. Zum anderen ist eine Konfiguration für die Untersuchung des Glanzlichtverhaltens nicht nötig, da aufgrund der anfangs erwähnten Einschränkungen kein Glanzlicht in die Beleuchtungsberechnung inkludiert wird.

Da hier im Gegensatz zu der Aufnahme realer Materialien keine Annahme über die Verteilung der Lichtquellenpositionen bezüglich der Kameraposition gemacht werden kann, wird im Rahmen dieses Kapitels die folgende Voraussetzung angenommen. Zunächst wird angenommen, dass Lichtquellenpositionen sich approximativ mittig vor dem Augenpunkt befinden; um eine Variation dieser Annahme einzuführen, wird die initiale Position $\mathbf{l} = (0.5, 0.5, 0.5)^T$ mittels der Addition eines zufälligen Vektors $a \in \mathbb{R}^3$, dessen Komponenten aus der uniformen Verteilung im Intervall $[-0.1, 0.1]$ gewählt werden, in alle Richtungen variiert.

Da einige Szenen mit einer solchen Konfiguration nicht vollständig ausgeleuchtet werden können, wird zusätzlich eine zweite Konfiguration eingeführt. Hierbei wird die Lichtquellenposition am durchschnittlichen Positionswert der Positionskarte $\mathbf{P}$ positioniert; diese Position wird ebenfalls durch die Addition eines zufälligen Vektors $a \in \mathbb{R}^3$ variiert, dessen Komponenten aus der uniformen Verteilung im Intervall $[-0.5, 0.5]$ gewählt werden.

Die Augenpunktposition $\mathbf{v}$ bleibt während des Beleuchtungsprozesses für jedes Rendering konstant an der Position $(0.5, 0.5, 1.0)^T$ positioniert. Abbildung 4.3 stellt einen solchen Szenenaufbau schematisch dar.

Insgesamt werden $n_1 + n_2$ Renderings für die Bestimmung eines Fehlerwertes angefertigt, wobei sich der Fehlerwert auf den durchschnittlichen L1-Fehler der Logarithmen aller Renderings bezüglich der Renderings ausgehend von den tatsächlichen Materialkarten bezieht. Die erste Konfiguration, bei der die initiale Lichtquellenposition vor den Augenpunkt positioniert wird, wird n_1 -fach instanziiert und gerendert; die zweite Konfiguration, bei der die durchschnittliche Position der Positionskarte $\mathbf{P}$ als initialer Ausgangspunkt der Lichtquellen gewählt wird, wird n_2 -fach instanziiert und gerendert. Die Werte n_1 und n_2 stellen dabei frei wählbare Parameter dar; im Rahmen der Evaluierung werden $n_1 = 2$ und $n_2 = 4$ in Abhängigkeit vom verfügbaren Speicher des Referenzsystems gewählt.

So beschreibt

$$L_{\text{Render}_1} = \sum_{i=1}^{n_1} \delta(R(\mathbf{P}, \mathbf{M}, \mathbf{l}_{1,i}, \mathbf{v}), R(\mathbf{P}, \mathbf{M}', \mathbf{l}_{1,i}, \mathbf{v})) \quad (4.5)$$

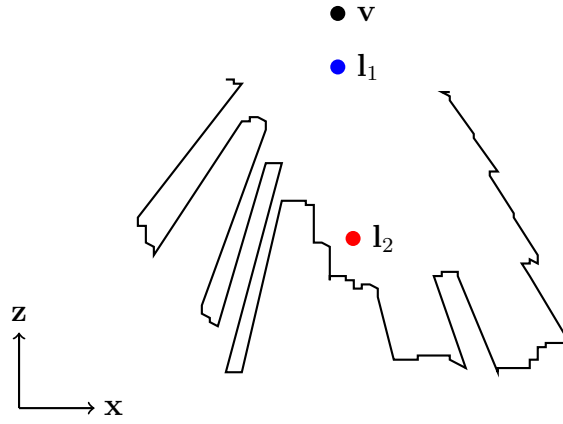


Abbildung 4.3: Schematische Darstellung des Szenenaufbaus einer räumlichen Szene. Die Abbildung stellt eine Ansicht der xz -Ebene dar. Die schwarzen Konturen stellen die projizierten Pixel einer Zeile der Positionskarte $\mathbf{P}$ dar. Der blaue Punkt repräsentiert den initialen Vektor der Lichtquellenposition $\mathbf{l}_1$, die sich an der Position $(0.5, 0.5, 0.5)$ befindet und die Grundlage für die erste Konfiguration bildet. Der rote Punkt repräsentiert die Lichtquellenposition $\mathbf{l}_2$, die sich an der durchschnittlichen Position aller in den Raum projizierten Pixel befindet und die Grundlage für die zweite Konfiguration bildet. Der Vektor $\mathbf{v}$ stellt die konstante Position des Augenpunkts dar und wird durch den schwarzen Punkt symbolisiert.

den resultierenden Fehler basierend auf der ersten Konfiguration mit n_1 Renderings und

$$L_{\text{Render}_2} = \sum_{i=1}^{n_2} \delta(R(\mathbf{P}, \mathbf{M}, \mathbf{l}_{2,i}, \mathbf{v}), R(\mathbf{P}, \mathbf{M}', \mathbf{l}_{2,i}, \mathbf{v})) \quad (4.6)$$

den Fehler der zweiten Konfiguration mit n_2 Renderings, wobei $\mathbf{P}$ eine vorgegebene Positionskarte, $\mathbf{M}$ die tatsächlichen Materialkarten und $\mathbf{M}'$ die geschätzten Materialkarten darstellen. Der Vektor $\mathbf{l}_{1,i}$ beschreibt die i -te Lichtquelle, die entsprechend der ersten Konfiguration und $\mathbf{l}_{2,i}$ analog dazu die i -te Lichtquelle entsprechend der zweiten Konfiguration. Der Augenpunkt $\mathbf{v}$ ist im Gegensatz zu dem stochastischen Fehlermaß des letzten Kapitels statisch platziert und bleibt konstant für alle Renderings. Das Fehlermaß δ für zwei Renderings entspricht dabei dem Fehlermaß aus dem Abschnitt 3.4.2. Der finale Fehlerwert bezieht sich auf

$$L_{\text{Render}} = \frac{L_{\text{Render}_1} + L_{\text{Render}_2}}{n_1 + n_2}. \quad (4.7)$$

Abbildung 4.4 stellt eine für eine solche Fehlerberechnung ausgeleuchtete Szene exemplarisch dar. Dabei stellt die erste Abbildung eine Instanz der ersten Konfiguration, die restlichen Abbildungen dagegen Instanzen der zweiten Konfiguration dar.

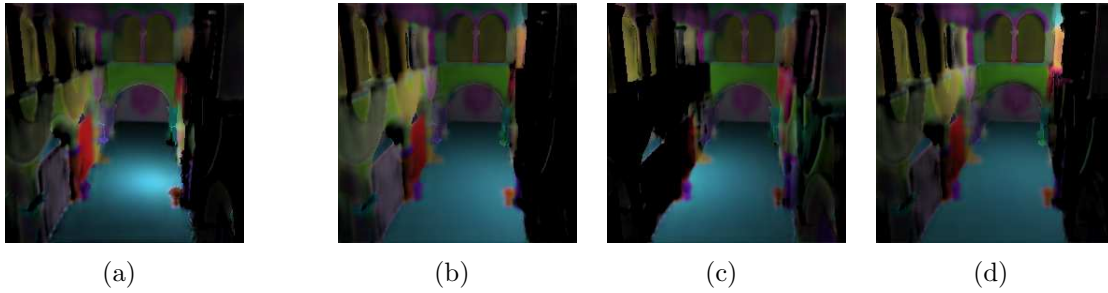


Abbildung 4.4: Beleuchtung einer Szene im Rahmen des Trainings in unterschiedlichen Konfigurationen für das räumliche Rendering-gestützte Fehlermaß. Die Renderings zeigen dieselbe Szene, die im Rahmen eines Trainingsprozesses anhand der geschätzten Materialkarten in unterschiedlichen Konfigurationen gerendert wurde. Abbildung (a) stellt dabei eine Instanz der ersten Konfiguration dar, die Abbildungen (b), (c) und (d) bilden dagegen Instanzen der zweiten Konfiguration ab. Das Training findet auf einem Datensatz bestehend aus räumlichen Szenen statt [46].

4.6 Regularisierung der Fehlerfunktion

Dieser Abschnitt stellt Regularisierungsmethoden für das stochastische Rendering-gestützte Fehlermaß des letzten Abschnittes vor. Die Evaluierung dieser Methoden findet im Abschnitt 4.8.3 statt.

Die aus der aktuellen Netzarchitektur und Fehlerfunktion resultierenden Schätzungen der Materialkarten weisen zwei Defizite auf; diese werden im Rahmen der Evaluierung an einem Beispiel in Abbildung 4.10 dargestellt. Zum einen besitzt die Normalenkarte eine Unschärfe entlang der Kanten der Objekte. Zum anderen weist die Karte für die diffuse Grundfarbe im Gegensatz zu der tatsächlichen Karte Flecken in Bereichen auf, die uniforme Farbinformationen besitzen sollten. Da eine Szene mehrere unterschiedlich uniform gefärbte geometrische Objekte besitzen kann, hat sich die anwendungsspezifische Erweiterung des letzten Kapitels, dargestellt durch das globale Merkmalsnetzwerk, nicht als nützlich erwiesen. An dieser Stelle wird die Vermutung aufgestellt, dass das globale Merkmalsnetzwerk durch die Injektion der selben globalen Informationen an jedes Pixel bei diesen Informationen nicht zwischen den jeweiligen geometrischen Objekten differenzieren kann und somit nicht in der Lage ist, objektspezifische Informationen aggregieren und entsprechend weiterleiten zu können.

Ausgehend von diesen zwei Defiziten wird eine neue Regularisierungsmethode vorgeschlagen; hierbei werden die zwei Defizite separat voneinander behandelt. Für die Normalenkarte wird der Sobel-Operator eingesetzt, der die Pixel entlang der Kanten verstärkt und einen Vergleich der Kanten der geschätzten und tatsächli-

chen Normalenkarte ermöglicht [64]. Für die Karte der diffusen Grundfarbe wird der Gauß-Operator eingesetzt, der das Bild weichzeichnet und somit die Farbwerte der Nachbarschaft gewichtet an einem Pixel festhält [50]; ein Vergleich der weichgezeichneten geschätzten und tatsächlichen Karten der diffusen Grundfarbe bezieht somit die Farbwerte der Nachbarschaft direkt mit in die Berechnung ein. Die Summe dieser zwei unabhängigen Berechnungen wird gewichtet auf den aus dem Fehlermaß des letzten Abschnitts resultierenden Fehler addiert; dies bildet somit die Regularisierung des vorausgehenden stochastischen Fehlermaßes.

Im Folgenden werden die zwei Verfahren detailliert dargestellt. Sei $\mathbf{M}^{(n)}$ die tatsächliche und $\mathbf{M}'^{(n)}$ die geschätzte Normalenkarte. Des Weiteren sei

$$\mathbf{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (4.8)$$

der horizontale Sobel-Operator und

$$\mathbf{S}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.9)$$

der vertikale Sobel-Operator [9], so berechnet sich der erste Term für die Regularisierung mittels der Normalenkarte auf

$$L_{\text{Normals}} = \|\mathbf{S}_x ** \mathbf{M}^{(n)} - \mathbf{S}_x ** \mathbf{M}'^{(n)}\|_1 + \|\mathbf{S}_y ** \mathbf{M}^{(n)} - \mathbf{S}_y ** \mathbf{M}'^{(n)}\|_1. \quad (4.10)$$

Hierbei werden zunächst die zwei Sobel-Operatoren auf die geschätzte und tatsächliche Normalenkarte separat angewandt, um die horizontalen und vertikalen Kanten beider Karten zu ermitteln; anschließend wird der durchschnittliche L1-Fehler der horizontalen Kanten beider Karten berechnet und auf den durchschnittlichen L1-Fehler der vertikalen Kanten beider Karten addiert; die Summe stellt den Fehlerwert des ersten Terms dar.

Sei $\mathbf{M}^{(d)}$ die tatsächliche und $\mathbf{M}'^{(d)}$ die geschätzte Karte der diffusen Grundfarbe.

Die Gauß-Funktion

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \quad (4.11)$$

bildet die Grundlage für die Berechnung der Koeffizienten des Gauß-Kernels abhängig von der Varianz σ^2 [50], wobei die Koordinaten $x = 0$ und $y = 0$ den betrachteten

Pixel darstellen. Im Rahmen dieser Arbeit wird ein Kernel der Größe 3×3 für das Weichzeichnen der Karte der diffusen Grundfarbe genutzt; sei $\mathbf{G}_{3 \times 3}$ ein solcher Kernel mit $\sigma = 1$.

Der zweite Term der Regularisierung beläuft sich somit auf

$$L_{\text{Diffuse}} = \|\mathbf{G}_{3 \times 3} * \mathbf{M}^{(d)} - \mathbf{G}_{3 \times 3} * \mathbf{M}'^{(d)}\|_1 \quad (4.12)$$

und ergibt sich aus dem durchschnittlichen L1-Fehler der weichgezeichneten tatsächlichen und geschätzten Karten der diffusen Grundfarbe.

Sei L_{Render} der Fehler des stochastischen Rendering-gestützten Fehlermaßes, so ergibt sich der finale Fehler nach der Regularisierung

$$L = L_{\text{Render}} + \lambda(L_{\text{Normals}} + L_{\text{Diffuse}}), \quad (4.13)$$

wobei der Wert $\lambda \geq 0$ die Gewichtung der Regularisierung angibt; verschiedene Gewichtungen werden im Rahmen der Evaluierung untersucht (siehe Abschnitt 4.8.3). Auch im Rahmen dieses Kapitels kann eine Regularisierung mittels des durchschnittlichen L1-Fehlers der geschätzten und tatsächlichen Materialkarten vorgenommen werden; diese Methode entspricht dem Regularisierungsverfahren des letzten Kapitels (siehe Abschnitt 3.4.3). Ein Vergleich der Filter-basierten und L1-basierten Methoden findet ebenfalls im Rahmen der Evaluierung statt.

4.7 Realisierung

In diesem Abschnitt wird die Realisierung der vorgestellten inversen Renderingfunktion für räumliche Szenen implementierungsbezogen dargelegt; die Erläuterung wird dabei analog zu dem Realisierungsabschnitt des vorausgehenden Kapitels gehalten (siehe Abschnitt 3.7).

Framework Die Implementierung bezüglich der räumlichen inversen Renderingfunktion wurde ebenfalls in der Python-Programmiersprache realisiert; hierbei wurde auch auf das PyTorch-Framework zurückgegriffen.

Datencompiler Die SVBRDFs der räumlichen Szenen entstammen dem Datensatz der Arbeit von Nalbach et al. [46], die ursprünglich für das Training eines Deep-Neural-Network-gestützten Renderers ausgelegt waren. Die benötigten Material- und Positionskarten sind dabei über mehrere Datensätze und Szenen zerstreut; zudem

liegen die Karten im HDR-Format vor, die in EXR-Dateien abgespeichert wurden. Für die Generierung des benötigten Datensatzes musste so zunächst ein Datencompiler geschrieben werden, der für eine gegebene Szene die Informationen aus verschiedenen Datensätzen in eine einzelne EXR-Datei transformiert. Der Datencompiler wurden ebenfalls in der Programmiersprache Python realisiert und nutzt die OpenEXR-Bibliothek¹ für das Verarbeiten von HDR-Bildern. Die Zusammenstellung des gesamten Datensatzes dauerte einige Minuten und wurde vorab offline realisiert.

Datenlader Die generierten Daten des Datencompilers liegen am Ende in EXR-Dateien vor. Für das Laden und Zuschneiden der Daten wurde ebenfalls die OpenEXR-Bibliothek für die Programmiersprache Python eingesetzt. Der Datenlader transformiert die Material- und Positionskarten in PyTorch-Tensoren, die im weiteren Verlauf im Rahmen des Trainings und der Evaluierung eingesetzt werden können.

U-Net Die Implementierung des U-Nets entspricht bis auf einigen wenigen Änderungen der U-Net-Implementierung des vorausgehenden Abschnitts; die anwendungsspezifische Erweiterung dargestellt durch das globale Merkmalsnetzwerk wurde hierbei nicht übernommen. Die Anpassungen des U-Nets beziehen sich auf die Anzahl der Eingabe- und Ausgabekanäle (siehe Abschnitt 4.3).

Differenzierbare Renderingkomponente Die differenzierbare Renderingkomponente für räumliche Szenen basiert auf der vorausgehenden Komponente für planare Oberflächenmaterialien; hierbei wird die Eingabe einer Positionskarte ermöglicht, die jedem Pixel eine relative Positionierung zur Kamera assoziiert. Die Renderingkomponente wurde ebenfalls mittels PyTorch-Module realisiert, um eine GPU-beschleunigte Bildsynthese und vollständige Differenzierbarkeit zu ermöglichen; sie setzt das Blinn-Phong-Beleuchtungsmodell (siehe Abschnitt 2.1.3.1) um; das Training wird dabei mit deaktivierter Glanzlichtkomponente durchgeführt.

Stochastisches Rendering-gestütztes Fehlermaß Im Rahmen des stochastischen Rendering-gestützten Fehlermaßes wird die differenzierbare Renderingkomponente für räumliche Szenen instanziiert. Entsprechend der Lichtquellen- und Augenpunktpositionierung, welche im Abschnitt 4.5 detailliert beschrieben wurde, werden eine Reihe an Renderings angefertigt; diese werden in den logarithmischen Raum

¹OpenEXR-Bibliothek für Python <https://pypi.org/project/OpenEXR/> – Zugriffsdatum: 26. September 2019

transformiert und mithilfe des L1-Fehlermaßes verglichen. Der Durchschnitt des so entstehenden Fehlers ist das Resultat des stochastischen Rendering-gestützten Fehlermaßes.

L1-basierte Regularisierung Für das Training der inversen Renderingfunktion räumlicher Szenen wurden zwei Regularisierungsverfahren präsentiert (siehe Abschnitt 4.6). Die L1-basierte Regularisierung nutzt dabei die vom Netz geschätzten und tatsächlichen Materialkarten und bildet den L1-Fehler der jeweiligen Karten, der auf den Renderingfehler addiert wird; eine Methode zur Berechnung des L1-Fehlers ist dabei Teil der PyTorch-Bibliothek. Da PyTorch die Gradienten automatisch generiert [52], wird hier keine weitere Anpassung benötigt.

Filter-basierte Regularisierung Für die Filter-basierte Regularisierung wurde ein zusätzliches PyTorch-Modul implementiert, welches die Hoch- und Tiefpassfilterung der jeweiligen Materialkarten vornimmt. Zusätzlich instanziiert dieses Modul die stochastische Rendering-gestützte Fehlermetrik, um auf dem resultierenden Fehlerwert eine Regularisierung durchzuführen.

Logging Für die nachträgliche Evaluierung der inversen Renderingfunktion wurde ein Logging-Mechanismus zum Abspeichern sämtlicher Fehlerwerte jeder Epoche realisiert. Zusätzlich wurde in jeder Epoche die Entwicklung des Deep Neural Networks anhand der Ausgabe einer Schätzung von einem festen Satz an Szenen in Form von Bildern ausgegeben.

4.8 Evaluierung

In diesem Abschnitt werden die umgesetzten Methoden dieses Kapitels evaluiert. Dazu werden zunächst die Evaluierungsszenarien im Abschnitt 4.8.1 vorgestellt und in den darauffolgenden Abschnitten die Analyse durchgeführt.

4.8.1 Evaluierungsszenarien

Im Folgenden werden die umgesetzten Methoden, der vorhandene Datensatz und das Referenzsystem skizziert und darauf aufbauend eine Reihe an zu untersuchenden Aspekten festgehalten.

In diesem Kapitel wurden die Methoden des vorausgehenden Kapitels auf räumliche Szenen übertragen. Insbesondere wurde dafür das U-Net des letzten Kapitels für die

inverse Renderingfunktion räumlicher Szenen angepasst (siehe Abschnitt 4.3). Des Weiteren wurde ein differenzierbarer Renderer und ein neues stochastisches Fehlermaß entworfen. Zudem wurden zwei Regularisierungsmethoden vorgeschlagen; diese beziehen sich auf die L1-basierte und auf die Filter-basierte Regularisierungsmethoden.

Für die Evaluierung wurde der Datensatz [46] in drei Datensätze aufgeteilt. Hierbei wird zwischen Welten und Szenen unterschieden. Ein einzelnes Datum stellt dabei eine Szene dar; dieses besteht aus einer Positionskarte $\mathbf{P}$ und einem Satz $\mathbf{M}$ an Materialkarte, die durch eine Normalenkarte und einer Karte für die diffuse Grundfarbe dargestellt werden. Eine Welt beschreibt eine dreidimensionale Umgebung; die Kamera wird dabei an unterschiedlichen Positionen mit unterschiedlichen Ausrichtungen in dieser Welt positioniert, um eine Szene aufzunehmen.

Insgesamt besteht der Datensatz aus 11 Welten; diese wurden in sieben Trainingswelten und vier Evaluierungswelten eingeteilt. Szenen aus den Trainingswelten wurden weiterhin in Trainingsszenen und Evaluierungsszenen aufgeteilt. Somit entstehen drei Datensätze, die sich auf einen Trainingsdatensatz und zwei Evaluierungsdatensätze beziehen. Der Trainingsdatensatz besteht dabei aus 6979 Szenen, die ausschließlich aus Trainingswelten entstammen, wobei ca. 1000 Szenen pro Welt eingesetzt werden. Die verbleibenden Szenen aus den Trainingswelten bilden den ersten Evaluierungsdatensatz. Der zweite Evaluierungsdatensatz besteht ausschließlich aus Szenen, die Evaluierungswelten entstammen. Pro Evaluierungsdatensatz liegen dabei 14 Szenen vor.

Sofern eine Szene bzw. eine Welt im Rahmen des Trainingsprozesses eingesetzt wurde, wird diese als „gesehen“ bezeichnet. So lässt sich zusammenfassend festhalten, dass die drei Datensätze sich in gesehene Szenen gesehener Welten, ungesehene Szenen gesehener Welten und ungesehene Szenen ungesehener Welten unterteilen lassen. Diese Aufteilung ermöglicht es sowohl ein Overfitting [19] bezüglich der Szenen als auch der Welten zu detektieren.

Das für das Training eingesetzte Referenzsystem entspricht dabei dem System aus dem vorausgehenden Kapitel (siehe Abschnitt 3.8.1); die Neubeleuchtung basierend auf den SVBRDFs wurde ebenfalls analog zum vorausgehenden Kapitel mittels eines Beleuchtungs-Shaders durchgeführt (siehe Anhang A).

Die zu untersuchenden Aspekte beziehen sich auf

- die qualitative und quantitative Analyse der im Rahmen dieses Kapitels entwickelten inversen Renderingfunktion für räumliche Szenen,
- die Praktikabilität der umgesetzten Methoden in der Anwendung,

- die quantitative Analyse der vorgestellten Regularisierungsverfahren und
- dem quantitativen und qualitativen Vergleich der Regularisierungsverfahren untereinander.

Im folgenden Abschnitt 4.8.2 wird die Analyse der räumlichen inversen Renderingfunktion vorgenommen.

4.8.2 Analyse der inversen Renderingfunktion für räumliche Szenen

In diesem Abschnitt wird die inverse Renderingfunktion für räumliche Szenen sowohl quantitativ als auch qualitativ analysiert. Die quantitative Analyse untersucht dabei die Fehlerkurven des Trainings bezüglich des Evaluierungs- als auch des Trainingsdatensatzes, um eine Übertragung und somit ein erfolgreiches Training feststellen zu können. Die qualitative Analyse dagegen beschäftigt sich mit der Neubeleuchtung und der Praktikabilität der umgesetzten inversen Renderingfunktion. Zunächst wird die quantitative und anschließend die qualitative Analyse durchgeführt.

4.8.2.1 Quantitative Analyse des Trainings

Im Folgenden wird eine quantitative Analyse des Trainings des Deep Neural Networks für die räumliche inverse Renderingfunktion durchgeführt. Sofern das Netz während des Trainings den Fehler sowohl auf dem Trainings- als auch auf dem Evaluierungsdatensatz erheblich reduzieren konnte, wird das Training als erfolgreich angesehen.

Das Deep Neural Network bezieht sich auf das angepasste U-Net (siehe Abschnitt 4.3) mit dem neuen stochastischen Rendering-gestützten Fehlermaß (siehe Abschnitt 4.5) und dem Filter-basierten Regularisierungsverfahren mit einer Gewichtung von $\lambda = 0.2$ (siehe Abschnitt 4.6). Die Filter-basierte Regularisierung hat sich als sowohl qualitativ als auch quantitativ bestes Verfahren herausgestellt (siehe Abschnitt 4.8.3).

Das Training wurde für 1000 Epochen mit dem Adam-Optimierungsalgorithmus [31] mit einer Lernrate von 0.00001 und einer Batchgröße von 16 durchgeführt; die Batchgröße wurde dabei größtmöglich gewählt und die Lernrate entspricht der des letzten Kapitels. Die Batchgröße bezieht sich im Rahmen dieses Kapitels auf die Anzahl an Daten, die jeweils aus einer Positionskarte $\mathbf{P}$ und den dazu korrespondierenden Materialkarten $\mathbf{M}$ bestehen; zudem wird ad-hoc ein beleuchtetes Bild $\mathbf{I}$ erzeugt, welches

die Eingabe an das Deep Neural Network darstellt. In jeder Epoche wurde sowohl der durchschnittliche Renderingfehler als auch die durchschnittlichen L1-Fehler der Materialkarten für alle drei anfangs erwähnten Datensätze (siehe Abschnitt 4.8.1) aufgezeichnet.

Abbildung 4.5 stellt den durchschnittlichen Renderingfehler über 1000 Epochen dar. Die x-Achse zeigt dabei die bis dahin durchlaufenen Epochen während des Trainings und die y-Achse bildet den durchschnittlichen Fehlerwert des stochastischen Rendering-gestützten Fehlermaßes ab. Erwartungsgemäß besitzt die Fehlerkurve für bereits gesehene Szenen (schwarz) den niedrigsten Fehlerwert, gefolgt von der Feh-

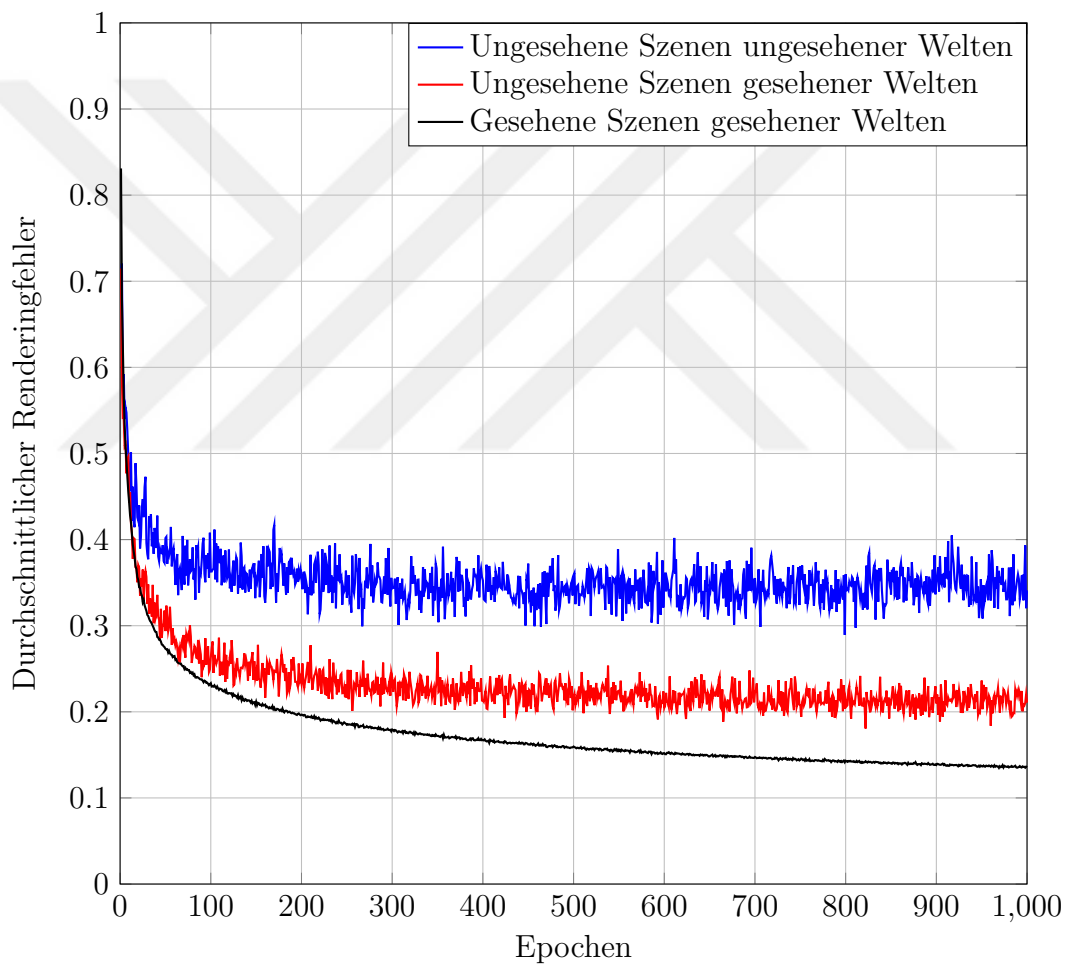


Abbildung 4.5: Durchschnittlicher Renderingfehler der räumlichen inversen Renderingfunktion. Die Kurven zeigen den durchschnittlichen Renderingfehler während des Trainings für 1000 Epochen für jeweils alle drei Datensätze. Die schwarze Kurve reflektiert den Fehler auf dem Trainingsdatensatz; die rote und die blaue Kurve zeigen den Fehler für die Datensätze ungesehener Szenen gesehener Welten und ungesehener Szenen ungesehener Welten in dieser Reihenfolge.

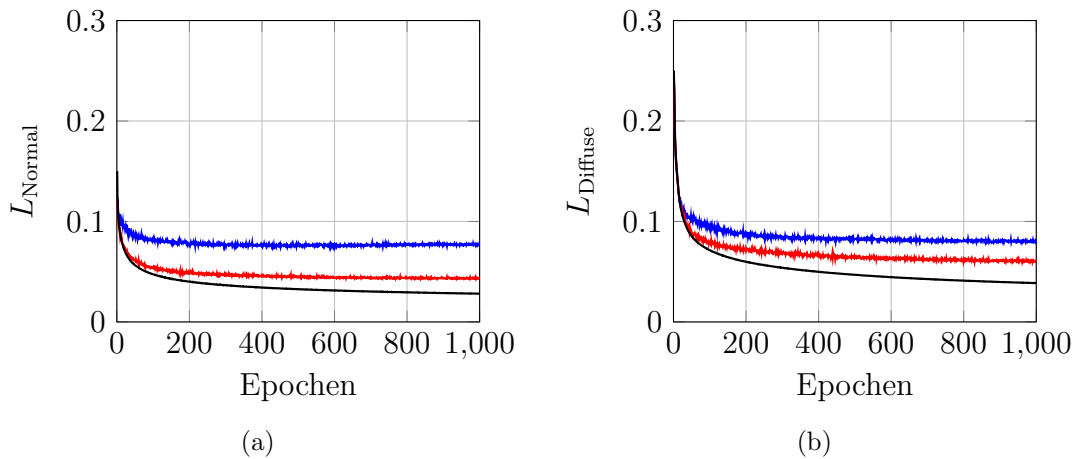


Abbildung 4.6: Durchschnittliche L1-Fehler der Materialkarten während des Trainings. Die Abbildung (a) zeigt den L1-Fehler der Normalenkarte; Abbildung (b) bildet den L1-Fehler der Karte der diffusen Grundfarbe ab. Die schwarze Kurve zeigt den Fehler auf gesehenen Szenen gesehener Welten, die rote Kurve den Fehler auf ungesehenen Szenen gesehener Welten und die blaue Kurve den Fehler auf ungesehenen Szenen ungesehener Welten.

lerkurve für unbekannte Szenen bekannter Welten (rot) und abschließend der Fehlerkurve für unbekannte Szenen unbekannter Welten (blau). Alle drei Fehlerkurven konnten im Laufe des Trainings optimiert werden. Das Rauschen der roten und blauen Kurve lässt sich auf das eingesetzte stochastische Fehlermaß gekoppelt mit der geringen Anzahl an Szenen in den jeweiligen Evaluierungsdatensätzen zurückführen; diese belaufen sich auf 14 Szenen pro Datensatz, was im starken Kontrast zu den ca. 7000 Szenen des Trainingsdatensatzes steht.

Die Abbildung 4.6 hält analog dazu die Fehlerkurven der Materialkarten fest. Die x-Achse zeigt auch hier die jeweilige Trainingsepoche an. Die y-Achse bildet hierbei den durchschnittlichen L1-Fehler der Materialkarten ab; Abbildung 4.6(a) zeigt dabei den Fehler auf den Normalenkarten und Abbildung 4.6(b) den Fehler auf den Karten der diffusen Grundfarbe. Alle L1-Fehlerkurven, sowohl der Evaluierungsdatensätze als auch des Trainingsdatensatzes, besitzen hierbei einen Fehlerwert kleiner 0.1 ab spätestens der 200. Epoche.

Die niedrigen Fehlerkurven der Evaluierungsdatensätze signalisieren eine Übertragung der gelernten *a priori* Annahmen auf neue Datensätze; dies schließt eine Übertragung auf sowohl neue Szenen als auch komplett neue Welten mit ein.

Somit kann an dieser Stelle festgehalten werden, dass ein erfolgreiches Training aus einer quantitativen Sicht erfolgt ist.

4.8.2.2 Qualitative Analyse

Im Rahmen dieser Analyse werden die Schätzungen des trainierten Deep Neural Networks qualitativ untersucht.

Dazu werden drei Daten aus den entsprechenden Datensätzen für die Evaluierung ausgewählt; die Datensätze beziehen sich auch hier auf die gesehenen Szenen gesehener Welten, ungesehenen Szenen gesehener Welten und ungesehenen Szenen ungesehener Welten. Bei dem zu untersuchenden Netz handelt es sich um das trainierte Deep Neural Network, welches entsprechend des letzten Abschnitts trainiert wurde. Basierend auf den Daten wird ein Rendering unter festen Beleuchtungsgegebenheiten angefertigt; dieses dient als Eingabe für das Deep Neural Network. Die daraus resultierenden Schätzungen werden zwischengespeichert. Zusätzlich wird ein Rendering unter denselben Beleuchtungsbedingungen synthetisiert, um einen direkten Vergleich der beleuchteten Szenen zu ermöglichen. Darüber hinaus werden alle Ergebnisse mit den tatsächlichen Daten direkt verglichen und visualisiert; dazu werden die Vektordistanzen der Farbwerte mittels einer Skala farblich gekennzeichnet.

In Abbildung 4.7 werden die Ergebnisse des Deep Neural Networks und den daraus resultierenden Neubeleuchtungen tabellarisch abgebildet. Diese sind in drei Gruppen entsprechend den drei Daten aufgeteilt. Die jeweils obere Reihe einer gegebenen Gruppe stellt die tatsächlichen, die untere die geschätzten Daten dar. Von links nach rechts wird das Rendering, die Normalenkarte und die Karte der diffusen Grundfarbe abgebildet. Die Positionskarte, die neben dem dargestellten tatsächlichen Rendering als zusätzliches Eingabedatum für die Anfertigung der Schätzungen genutzt wird, ist der Übersicht halber nicht abgebildet.

Analog zu dieser Abbildung stellt Abbildung 4.8 die Fehler der jeweiligen Schätzungen und Renderings farblich dar. Die Vektordistanzen der Ergebnisse mit den tatsächlichen Daten werden dabei entsprechend den abgebildeten Farbskalen eingefärbt.

In den Ergebnissen in Abbildung 4.7 kann eine konsistente Qualität in allen Schätzungen durch alle drei Daten hinweg aufgezeigt werden. Die dargestellten Fehler in Abbildung 4.8 sind zudem im Durchschnitt im unteren Bereich der Fehlerskala; eine quantitative Analyse, die diesen Aspekt unterstreicht, wurde im vorausgehenden Abschnitt durchgeführt.

Die Normalenkarte, dargestellt in Abbildung 4.7, kann neben der Struktur der groben Szene auch Details wie Kanten geometrischer Objekte (obere Szene) oder Konturen von Gardinen (mittlere Szene) qualitativ hochwertig approximieren. Auch in den Karten der diffusen Grundfarbe konnten bis auf wenige Beleuchtungsrückstände die

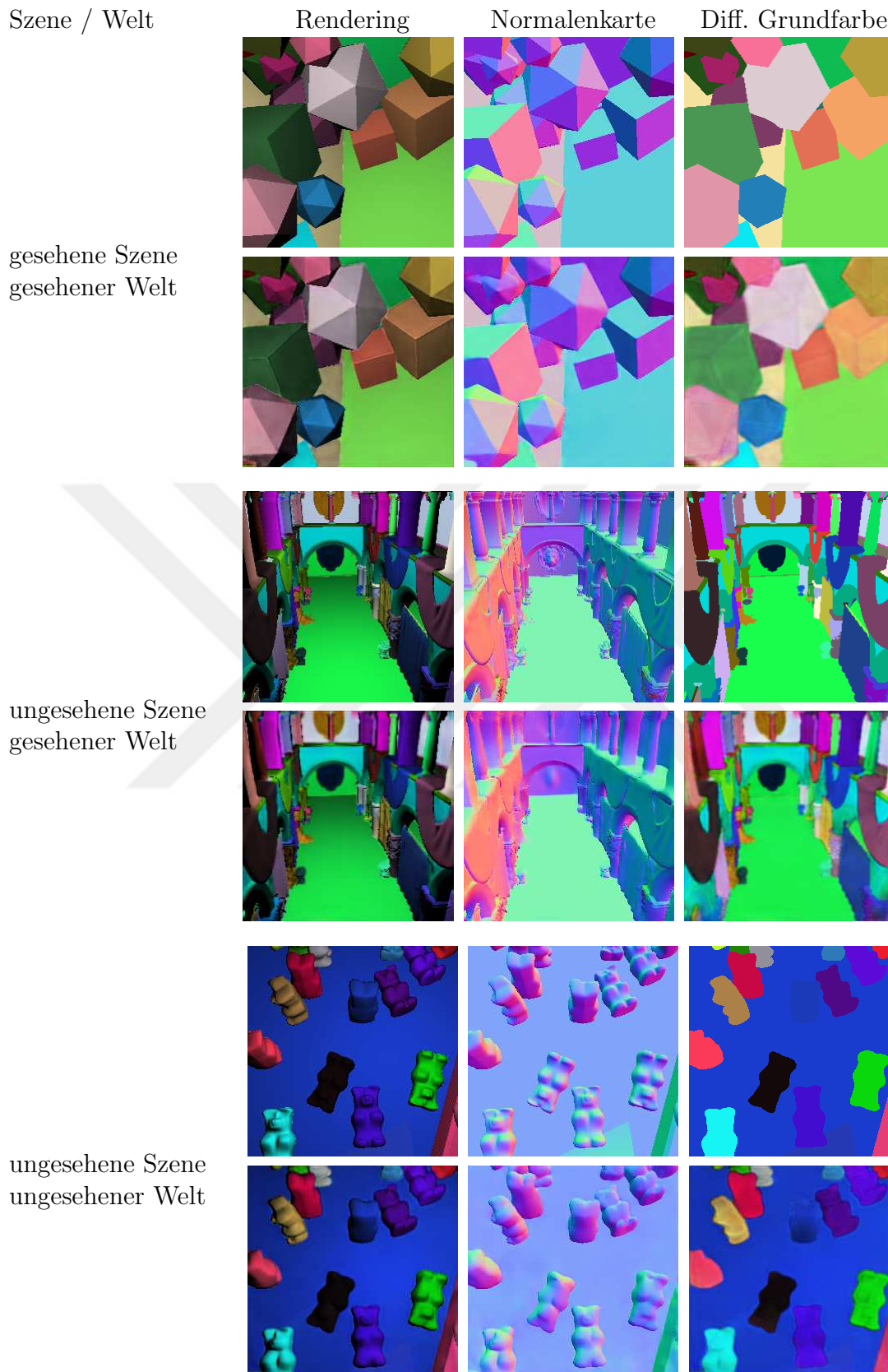


Abbildung 4.7: Qualitative Ergebnisse der räumlichen inversen Renderingfunktion. Die drei Gruppen zeigen unterschiedliche Szenen unterschiedlicher Welten; die Schätzungen in der jeweils unteren Reihe approximieren überzeugend die tatsächlichen Daten, die in den jeweiligen oberen Reihen abgebildet sind.

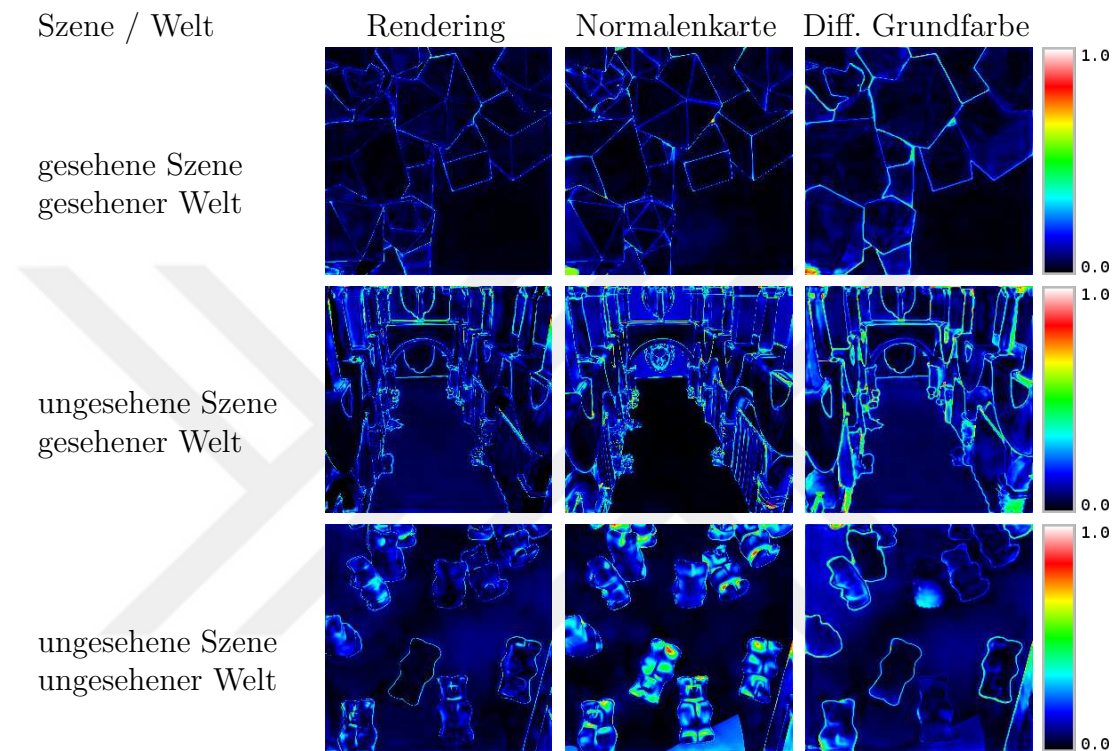


Abbildung 4.8: Qualitativer Vergleich der Ergebnisse der räumlichen inversen Renderingfunktion. Dargestellt werden die Vektordistanzen der Farbwerte der tatsächlichen und geschätzten Daten der Abbildung 4.7. Die eingesetzte Farbskala ist jeweils rechts abgebildet. Die obere Reihe stellt eine gesehene Szene einer gesehenen Welt, die mittlere eine ungesehene Szene einer gesehenen Welt und die untere Reihe eine ungesehene Szene einer ungesehenen Welt dar. Von links nach rechts werden die Renderings, die Normalenkarten und die Karten der diffusen Grundfarbe miteinander verglichen.

tatsächlichen Daten approximiert werden. Die daraus resultierenden Renderings sind schwer von den tatsächlichen Renderings zu unterscheiden und weisen in den Fehlerbildern in Abbildung 4.8 geringe Fehlerwerte auf. Speziell die ungesehene Szene aus einer ungesesehenen Welt (unterste Reihe) zeigt im Rendering basierend auf den geschätzten Materialkarten starke Ähnlichkeiten zu dem tatsächlichen Rendering.

Aus den Fehlerbildern wird ersichtlich, dass bezüglich der inversen Renderingfunktion für räumliche Szenen Schwierigkeiten besonders an den Kanten auftreten. Zudem treten große Abweichungen in der Normalenkarte der Gummibärchen-Szene an den Rundungen der Beine und Ohren auf.

Die auftretenden Probleme in den Normalenkarten lassen sich auf ungünstige Beleuchtungsbedingungen zurückführen; diese treten auf, wenn in dem Eingaberendering das Licht aufgrund einer annähernd schwarzen diffusen Grundfarbe keine kontrastreichen Konturen aufzeigen kann; dieser Effekt ist beispielsweise in der mittleren Szene am Ende der Halle bei der dunklen Löwenstatue oder auch an dem schwarzen Gummibärchen in der unteren Szene zu sehen. Beide Beispiele führen zu ungenaueren Schätzungen (siehe Abbildung 4.8, mittlere Spalte), die sich jedoch in den Renderings aufgrund der geringen Kontraste nicht bemerkbar machen; dieser Aspekt wird in den Fehlerbildern durch geringe Fehlerwerte in den Renderings ersichtlich, bei dem sowohl der Löwenkopf, als auch das schwarze Gummibärchen sehr geringe Fehlerwerte aufweisen. Zudem führen besonders starke Kontraste, wie sie beispielsweise bei dem orangenen Gummibärchen in der unteren Szene auftreten, zu starken Beleuchtungsrückständen in der diffusen Grundfarbe. Auch Kanten in der oberen Szene führen in einigen Fällen zu leichten Verfärbungen in der Karte der diffusen Grundfarbe.

Ein möglicher Grund für die hohen Fehlerwerte entlang der Kanten besteht in dem eingesetzten Datensatz [46]. Dabei lagen die Positionskarten und die Normalenkarten nicht in denselben Dimensionen vor und eine manuelle Skalierung mittels der Nächsten-Nachbar-Methode, die ungenaue Resultate besonders entlang der Kanten produziert, wurde eingesetzt.

Insgesamt lässt sich festhalten, dass die SVBRDF-Schätzungen den tatsächlichen Materialkarten stark ähneln; folglich approximieren die darauf basierenden Renderings die tatsächlichen Renderings ebenfalls stark. Besonders signifikant sind die beinahe identischen Renderings der ungesesehenen Szene aus einer ungesesehenen Welt, was eine erfolgreiche Übertragung der gelernten a priori Annahmen auf komplett neue Szenarien darlegt.

4.8.3 Analyse der Regularisierungsverfahren

Die untersuchten Regularisierungsverfahren beziehen sich auf die L1-basierte und auf die Filter-basierte Regularisierung; beide Verfahren wurden im Rahmen des Abschnitts 4.6 vorgestellt. Die Analyse ist in zwei Aspekte unterteilt. Zum einen werden in Abschnitt 4.8.3.1 die Regularisierungsverfahren quantitativ zunächst getrennt von einander untersucht. Anschließend wird ein quantitativer Vergleich der jeweils besten Parametrisierung des jeweiligen Regularisierungsverfahren durchgeführt. Zum anderen werden in Abschnitt 4.8.3.2 die Regularisierungsverfahren zusätzlich qualitativ gegenübergestellt.

4.8.3.1 Quantitative Auswertung

Im Folgenden werden die L1-basierten und Filter-basierten Regularisierungsverfahren getrennt voneinander quantitativ untersucht. Dabei wird das Deep Neural Network mit unterschiedlichen Gewichtungen bezüglich der jeweiligen Regularisierung trainiert und tabellarisch verglichen. Die Gewichtungen beziehen sich auf $\lambda \in \{0.0, 0.2, 0.5, 1.0\}$. Die Varianten des Trainings werden basierend auf dem Rendering-gestützten Fehlermaß mit der jeweiligen Regularisierung, einer Lernrate von 0.00001, einer Batchgröße von 16 und dem Adam-Optimierungsalgorithmus [31] für jeweils 200 Epochen durchgeführt. Abschließend werden die Ergebnisse der besten Parametrisierungen beider Varianten gegenübergestellt, um das quantitativ überlegene Verfahren herauszustellen.

Die trainierten Deep Neural Networks wurden auf den drei Datensätzen (siehe Abschnitt 4.8.1) ausgewertet. Dazu wurde das Rendering-gestützte Fehlermaß bezüglich der Deep-Neural-Network-Schätzungen für 7000 Szenen auf den jeweiligen Datensätzen ausgeführt. Für die jeweils 14 Szenen der beiden Evaluierungsdatensätze wurden die Datensätze 500 mal durchlaufen; diese hohe Anzahl an zu evaluierenden Szenen hat den Hintergrund, dass es sich bei dem Rendering-gestützten Fehlermaß um einen stochastischen Ansatz handelt und dieser bei zu geringen Datenmengen ein großes Rauschen aufweist; durch die multiplen Durchläufe wird somit eine höhere Konvergenz bezüglich des tatsächlichen Renderingfehlers erreicht. Zusätzlich wird der Vollständigkeit halber der L1-Fehler der Materialkarten festgehalten. Für alle Fehlerwerte wird hierbei der Durchschnitt bezüglich einer Szene gebildet, um einen Vergleich zu vereinfachen.

L1-basierte Regularisierung Tabelle 4.1 stellt die resultierenden Fehlerwerte für das L1-gestützte Regularisierungsverfahren dar. Die Tabelle unterteilt die Datensät-

Bekannte Szenen				
	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1.0$
L_{Render}	0.20807 ± 0.06	0.19697 ± 0.06	0.19400 ± 0.06	0.19312 ± 0.06
L_{Normal}	0.04261 ± 0.01	0.04055 ± 0.01	0.04021 ± 0.01	0.03983 ± 0.01
L_{Diffuse}	0.06738 ± 0.02	0.05943 ± 0.02	0.05487 ± 0.02	0.04952 ± 0.02
Unbekannte Szenen bekannter Welten				
	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1.0$
L_{Render}	0.24422 ± 0.09	0.23933 ± 0.09	0.23952 ± 0.09	0.24209 ± 0.10
L_{Normal}	0.05063 ± 0.02	0.04934 ± 0.02	0.04940 ± 0.02	0.04889 ± 0.02
L_{Diffuse}	0.07319 ± 0.02	0.07001 ± 0.03	0.07063 ± 0.03	0.07035 ± 0.03
Unbekannte Szenen unbekannter Welten				
	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1.0$
L_{Render}	0.36870 ± 0.14	0.36659 ± 0.14	0.36772 ± 0.15	0.37099 ± 0.14
L_{Normal}	0.08124 ± 0.03	0.08157 ± 0.03	0.08232 ± 0.03	0.08231 ± 0.03
L_{Diffuse}	0.09088 ± 0.04	0.08919 ± 0.04	0.09086 ± 0.04	0.08368 ± 0.03

Tabelle 4.1: Gegenüberstellung verschiedener Gewichtungen der L1-basierten Regularisierung. Es wird zwischen bekannten Szenen, unbekannten Szenen bekannter Welten und unbekannten Szenen unbekannter Welten unterschieden. Hierbei wird der stochastische Renderingfehler L_{Render} , der L1-Fehler der Normalenkarten L_{Normal} und der L1-Fehler der diffusen Grundfarbenseiten L_{Diffuse} im Durchschnitt zusammen mit der Standardabweichung festgehalten. Die fett markierten Werte zeigen die jeweils niedrigsten Fehlerwerte an. Die Variante mit einer Gewichtung von $\lambda = 0.2$ zeigt die beste Übertragung auf die Evaluierungsdatensätze und stellt sich somit als beste Variante heraus.

ze dabei in bekannte Szenen, unbekannte Szenen bekannter Welten und unbekannte Szenen unbekannter Welten. Die niedrigsten Fehlerwerte sind dabei durch einen Fettdruck gekennzeichnet. Bei den bekannten Szenen besitzt die Regularisierung mit $\lambda = 1.0$ den durchweg geringsten Fehler; in den Evaluierungsdatensätzen dagegen kann das Verfahren mit einer Gewichtung von $\lambda = 0.2$ den geringsten Renderingfehler aufweisen.

Der geringe Fehler der Regularisierung mit $\lambda = 1.0$ lässt sich auf ein mögliches Overfitting [19] zurückführen, welches besonders durch den geringen Fehlerwert der diffusen Grundfarbe im Vergleich zu den anderen Gewichtungen im Rahmen der bekannten Szenen nachzuweisen ist.

Da der Renderingfehler an dieser Stelle die wichtigste Aussage bezüglich der Qualität angibt und dieser bei einer Gewichtung von $\lambda = 0.2$ bei den Evaluierungsdatensätzen am geringsten ist, wird hier diese Gewichtung als beste Variante festgehalten.

Filter-basierte Regularisierung In Tabelle 4.2 werden die Ergebnisse der Auswertung der unterschiedlich gewichteten Filter-basierten Regularisierungen festgehalten. Bei bekannten Szenen besitzt das Verfahren mit der Gewichtung $\lambda = 1.0$ die durchweg geringsten Fehlerwerte. Auch lassen sich diese geringen Fehlerwerte der Materialkarten auf Evaluierungsdatensätze übertragen mit der Ausnahme der Normalenkarte, die bei unbekannten Welten bei einer Gewichtung von $\lambda = 0.2$ den geringsten Fehlerwert aufweist. Die Renderingfehler dagegen sind bei unbekannten Szenen bekannter Welten bei einer Gewichtung von $\lambda = 0.0$ und bei unbekannten Szenen unbekannter Welten bei einer Gewichtung von $\lambda = 0.2$ am geringsten.

Auch hier lässt sich eine hohe Diskrepanz bei den L1-Fehlerwerten der diffusen Grundfarbe bei einer Gewichtung von $\lambda = 1.0$ im Vergleich zu den anderen Gewichtungen feststellen, was zunächst auf ein mögliches Overfitting [19] deuten könnte. Dies bestätigt sich auch durch den höheren Renderingfehler in den darauffolgenden Szenarien.

Insgesamt findet bei dem Filter-basierten Verfahren eine bessere Übertragung der geringen Fehlerwerte bei den Materialkarten als bei dem L1-basierten Verfahren statt. Auch an dieser Stelle stellt sich das Verfahren mit einer Gewichtung von $\lambda = 0.2$ als bestes Verfahren bezüglich unbekannter Szenen unbekannter Welten heraus.

Im folgenden Abschnitt werden die Resultate der L1- und Filter-basierten Verfahren quantitativ verglichen.

Bekannte Szenen				
	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1.0$
L_{Render}	0.20807 ± 0.06	0.22102 ± 0.06	0.21157 ± 0.06	0.19511 ± 0.06
L_{Normal}	0.04261 ± 0.01	0.04506 ± 0.01	0.04348 ± 0.01	0.04016 ± 0.01
L_{Diffuse}	0.06738 ± 0.02	0.06894 ± 0.02	0.06034 ± 0.02	0.05099 ± 0.02
Unbekannte Szenen bekannter Welten				
	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1.0$
L_{Render}	0.24422 ± 0.09	0.25801 ± 0.10	0.24703 ± 0.09	0.24908 ± 0.10
L_{Normal}	0.05063 ± 0.02	0.05237 ± 0.02	0.05037 ± 0.02	0.05009 ± 0.02
L_{Diffuse}	0.07319 ± 0.02	0.07850 ± 0.03	0.06970 ± 0.03	0.06842 ± 0.03
Unbekannte Szenen unbekannter Welten				
	$\lambda = 0.0$	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 1.0$
L_{Render}	0.36870 ± 0.14	0.36583 ± 0.14	0.36854 ± 0.14	0.37859 ± 0.14
L_{Normal}	0.08124 ± 0.03	0.08054 ± 0.03	0.08225 ± 0.03	0.08239 ± 0.03
L_{Diffuse}	0.09088 ± 0.04	0.09395 ± 0.04	0.08902 ± 0.04	0.08601 ± 0.03

Tabelle 4.2: Gegenüberstellung verschiedener Gewichtungen der Filter-basierten Regularisierung. Abgebildet wird der stochastische Renderingfehler L_{Render} , der L1-Fehler der Normalenkarte L_{Normal} und der L1-Fehler der diffusen Grundfarbenkarte L_{Diffuse} im Durchschnitt zusammen mit der Standardabweichung. Die unterschiedlichen Gewichtungen der Filter-basierten Regularisierung werden auf bekannten Szenen, unbekannten Szenen bekannter Welten und unbekannten Szenen unbekannter Welten untersucht. Die fett markierten Werte zeigen die jeweils niedrigsten Fehlerwerte an. Die Gewichtung $\lambda = 0.2$ besitzt die beste Übertragung auf unbekannte Szenen unbekannter Welten.

	$\lambda = 0.0$	$\lambda_{\text{L1}} = 0.2$	$\lambda_{\text{Filter}} = 0.2$
L_{Render}	0.36870 ± 0.14	0.36659 ± 0.14	0.36583 ± 0.14
L_{Normal}	0.08124 ± 0.03	0.08157 ± 0.03	0.08054 ± 0.03
L_{Diffuse}	0.09088 ± 0.04	0.08919 ± 0.04	0.09395 ± 0.04

Tabelle 4.3: Quantitativer Vergleich der unterschiedlichen Regularisierungsverfahren. Der Vergleich wird auf den Szenen der ungesesehenen Evaluierungswelten durchgeführt. Von links nach rechts werden die durchschnittlichen Fehlerwerte und die Standardabweichungen der Verfahren ohne Regularisierung ($\lambda = 0.0$), mit L1-basierter Regularisierung ($\lambda_{\text{L1}} = 0.2$) und Filter-basierter Regularisierung ($\lambda_{\text{Filter}} = 0.2$) dargestellt. Die Fehler beziehen sich auf den stochastischen Renderingfehler L_{Render} , den L1-Fehler der Normalenkarten L_{Normal} und den L1-Fehler der Karten der diffusen Grundfarbe L_{Diffuse} . Die fett markierten Werte zeigen die jeweils niedrigsten Fehlerwerte an. Die Filter-basierte Regularisierung stellt sich als quantitativ bestes Verfahren heraus.

Quantitativer Vergleich der Verfahren Die Tabelle 4.3 stellt die besten Gewichtungen bezüglich des Evaluierungsdatensatzes unbekannter Welten für einen qualitativen Vergleich gegenüber; zusätzlich werden die Resultate ohne eingesetzte Regularisierung dargestellt. Die besten Gewichtungen der jeweiligen Verfahren beziehen sich dabei entsprechend der letzten Abschnitte in beiden Fällen auf $\lambda = 0.2$. Den insgesamt geringsten Fehlerwert bezüglich des Rendering-gestützten Fehlermaßes kann dabei das Filter-basierte Verfahren mit einer Gewichtung von $\lambda = 0.2$ aufweisen. Auch den geringsten L1-Fehler auf den Normalenkarten besitzt die Filter-basierte Variante; der geringste L1-Fehler der diffusen Grundfarbenkarte wird dabei von dem L1-basierten Verfahren erreicht. Insgesamt sind beide Varianten bezüglich des Renderingfehlers besser als ein Training ohne Regularisierung.

Da der Renderingfehler die Auswirkungen der Materialkarten auf die beleuchteten Resultate festhält, wird an dieser Stelle das Filter-basierte Verfahren als bestes Verfahren herausgestellt.

Eine qualitative Analyse der Regularisierungsmethoden findet im nächsten Abschnitt statt.

4.8.3.2 Qualitativer Vergleich

Im Folgenden werden die Regularisierungsmethoden auf einer ungesesehenen Szene einer ungesesehenen Welt qualitativ evaluiert und gegenübergestellt.

Dazu werden drei verschiedenen trainierte Deep Neural Networks untersucht; das erste Netz wird dabei ohne Regularisierungsverfahren trainiert. Das zweite Netz wird mittels der L1-Regularisierung mit einer Gewichtung von $\lambda = 1.0$ und das dritte Netz mit der Filter-basierten Regularisierung ebenfalls mit einer Gewichtung von $\lambda = 1.0$ trainiert. Die Idee hinter diesen Gewichtungen besteht darin, dass die maximale Auswirkung des Regularisierungsverfahren in den Ergebnissen verdeutlicht werden soll. Das Training findet dabei auf dem anfangs erwähnten Trainingsdatensatz mit ca. 7000 Szenen aus Trainingswelten statt und wird mit denselben Trainingsparametern des Abschnittes 4.8.3 für 200 Epochen durchgeführt.

Im Folgenden werden die Materialkartenschätzungen der trainierten Netze und die daraus resultierenden Renderings qualitativ bewertet. Dazu werden die Schätzungen der Normalenkarten, der Karten der diffusen Grundfarbe und die daraus resultierenden Renderings für einen Ausschnitt einer Szene aus einer Evaluierungswelt miteinander verglichen. Abbildung 4.9 stellt die untersuchte Szene dar; der rote Rahmen kennzeichnet den untersuchten Bereich.

Abbildung 4.10 stellt die Schätzungen tabellarisch dar. Hierbei stellen die Reihen

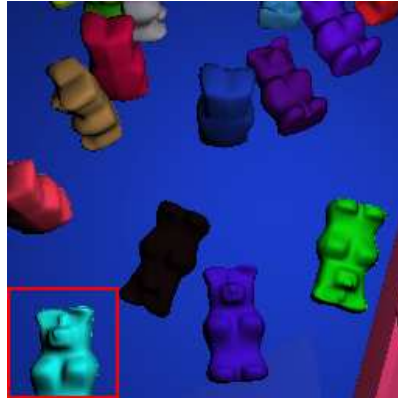


Abbildung 4.9: Gerenderte Szene aus einer Evaluierungswelt. Der rote Rahmen (untere linke Ecke) kennzeichnet den zu untersuchenden Bildausschnitt. Das Bild wurde basierend auf den Materialkarten des Datensatzes dieses Kapitels [46] diffus gerendert und dient als Eingabe für die Deep Neural Networks für den qualitativen Vergleich der Regularisierungsmethoden.

von oben nach unten die Renderings, die Normalenkarten und die Karten der diffusen Grundfarbe dar. Die erste Spalte stellt die tatsächlichen Materialkarten der Szene, die zweite Spalte die Schätzungen des Deep Neural Networks ohne Regularisierung, die dritte Spalte die Schätzungen des trainierten Netzes mit L1-basierter Regularisierung und die letzte Spalte die Schätzungen des Netzes mit Filter-basierter Regularisierung dar.

Die Ergebnisse des Netzes ohne Regularisierung beinhalten Artefakte am unteren linken Bildrand; zudem weist die Karte der diffusen Grundfarbe starke Beleuchtungsrückstände auf. Die Schätzungen des mittels der L1-basierten Regularisierung trainierten Netzes unterscheiden sich qualitativ durch eine artefaktfreie Normalenkarte und einer größtenteils uniform eingefärbten Karte der diffusen Grundfarbe, welche das Erscheinungsbild der tatsächlichen Karte besser approximiert; dennoch lassen sich schwache Beleuchtungsrückstände auch in dieser Karte nachweisen. Auch das Netz, welches mittels der Filter-basierten Regularisierung trainiert wurde, produziert eine qualitativ hochwertigere Normalenkartenschätzung; die Karte der diffusen Grundfarbe weist zudem nur sehr schwache Beleuchtungsrückstände auf und kommt somit der tatsächlichen Karte qualitativ am nächsten.

Durch den qualitativen Vergleich lässt sich herausstellen, dass trotz der quantitativ geringen Unterschiede der Regularisierungsverfahren (siehe Abschnitt 4.8.3.1) ein großer qualitativer Vorteil durch eine Regularisierung erzielt werden konnte. Die beste Qualität konnte in diesem Beispiel durch die Filter-basierte Regularisierung erreicht werden, gefolgt von dem L1-basierten Verfahren. Das Verfahren ohne Regularisierung führte in diesem Beispiel zu Artefakten in den Schätzungen.

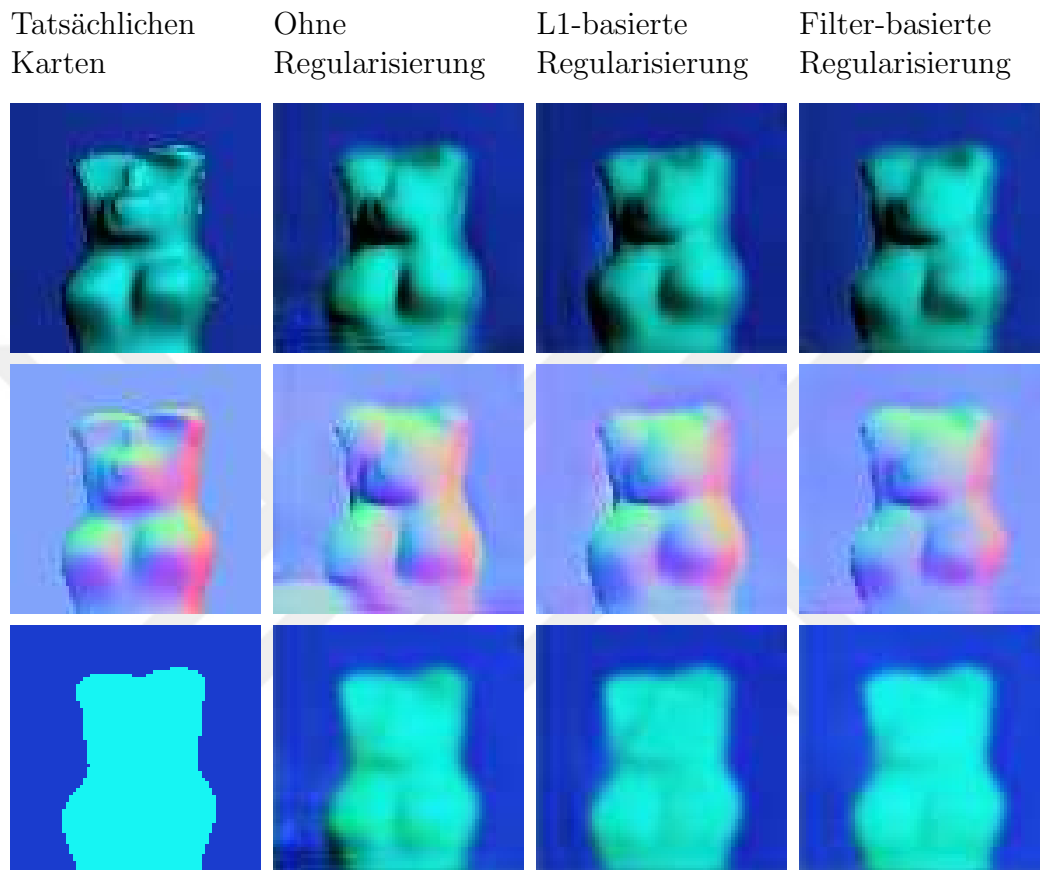


Abbildung 4.10: Qualitativer Vergleich der Regularisierungsverfahren. Der Vergleich wird auf einem Ausschnitt einer Szene einer Evaluierungswelt [46] durchgeführt. Die oberste Reihe stellt das jeweilige Rendering, die Reihe darunter die Normalenkarte und die unterste Reihe die Karte der diffusen Grundfarbe dar. Von links nach rechts werden zunächst die tatsächlichen Daten, die Resultate des Netzes ohne Regularisierung, mit L1-basierter Regularisierung und schließlich mit Filter-basierter Regularisierung dargestellt. Es zeigt sich, dass die Qualität bei dem Filter-basierten Verfahren am höchsten ist, gefolgt von der L1-basierten Regularisierung.

Somit wird die im Rahmen dieser Arbeit entwickelte Filter-basierte Regularisierung als sowohl qualitativ als auch quantitativ überlegenere Methode bezüglich der Verfahren ohne Regularisierung und der L1-basierten Regularisierung angesehen.

4.8.4 Anwendung der räumlichen inversen Renderingfunktion

Abschließend soll in diesem Abschnitt der Evaluierung die Praktikabilität der umgesetzten inversen Renderingfunktion für räumliche Szenen im Rahmen von beispielhaften Anwendungen untersucht werden.

Dazu wird das entsprechend des Abschnitts 4.8.2 trainierte Deep Neural Network eingesetzt; das Training wurde dabei für 1000 Epochen mit Filter-basierter Regularisierung mit einer Gewichtung von $\lambda = 0.2$ durchgeführt. Zusätzlich wird ein Datum einer ungesehenen Szene einer bereits gesehenen Welt als Eingabe gewählt.

Abbildung 4.11(a) stellt die diffus beleuchtete Eingabe für das Deep Neural Network dar. Basierend auf den resultierenden SVBRDF-Schätzungen des Netzes wird zunächst eine Neubeleuchtung mit einer neuen Beleuchtungskonfiguration synthetisiert; diese soll die Szene aus einem explizit anderen Winkel ausleuchten. Ein Szenario einer Neubeleuchtung mit denselben Beleuchtungsparametern wurde bereits zuvor in Abschnitt 4.8.2.2 untersucht. In einem weiteren Szenario wird dem Originaldatum eine zusätzliche Lichtquelle hinzugefügt; dazu wird mittels einer additiven Überlagerung einer Neubeleuchtung der Szene auf die abgedunkelte Eingabeszene ein neues Bild synthetisiert. Die dritte zu untersuchende Anwendung ist eine Neubeleuchtung basierend auf den geschätzten Materialkarten mittels eines Toon-Shadings und soll eine künstlerische Neuinszenierung darstellen; bei dem Toon-Shading wird

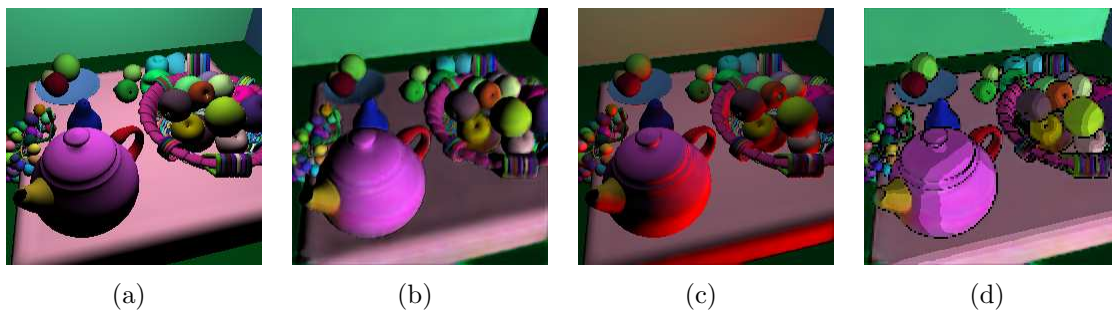


Abbildung 4.11: Verschiedene Anwendungen der räumlichen inversen Renderingfunktion. Abbildung (a) stellt das Eingabedatum dar. Basierend auf einer Materialkartenschätzung des trainierten Deep Neural Networks wird in Abbildung (b) eine einfache Neubeleuchtung, in Abbildung (b) eine additive Überblendung für die Einführung einer neuen Lichtquelle und in Abbildung (c) eine künstlerische Reinszenierung vorgenommen.

mittels der klassischen Lambert-Beleuchtung in eine Abstufungstextur indiziert, um ein cartoonhaftes Aussehen zu generieren [4]. Zudem wird hier ein schwarze Rand entlang der Kanten generiert, um diesen Eindruck weiter zu unterstreichen.

Die Abbildungen 4.11 (b) bis (d) stellen die Ergebnisse der drei Anwendungsfälle dar. Abbildung 4.11(b) zeigt dabei die Neubeleuchtung der Eingabeszene mit unterschiedlicher Positionierung der Lichtquelle; die neubeleuchtete Szene ähnelt bezüglich der Lichtverhältnisse und den Schattierungen der Originalszene sehr gering und stellt somit eine überzeugende Neubeleuchtung der Szene mit unterschiedlich positionierter Lichtquelle dar. Abbildung 4.11(c) zeigt die Eingabeszene mit einer zusätzlichen roten Lichtquelle, welche die Szene von vorne bestrahlt; der mittels der Überlagerung entstehende Effekt zeigt somit eine effektive und qualitativ hochwertige Einführung einer neuen Lichtquelle in eine bereits im Vorfeld beleuchtete Szene. Abschließend stellt Abbildung 4.11(d) die künstlerische Reinszenierung der Eingabeszene durch das Toon-Shading dar; dabei wurde auch in dieser Szene eine Neubeleuchtung mittels einer neu positionierten Lichtquelle vorgenommen und eine Abstufung der Intensitäten durchgeführt, sodass ein cartoonartiges Aussehen hervorgerufen wird. Gemeinsam mit den schwarzen Rändern wird somit ein überzeugendes cartoonhaftiges Aussehen generiert.

Insgesamt produzieren alle drei Anwendungen qualitativ hochwertige Resultate. Die Neubeleuchtung der Szene mit korrekten Beleuchtungseffekten stellt die erfolgreich erlernte Schätzung der SVBRDF einer räumlichen Szene dar. Die Augmentation einer Eingabeszene durch eine zusätzliche Lichtquelle (siehe Abbildung 4.11(c)) zeigt die effektive Interaktion bereits vorhandener Szenen mit zusätzlichen virtuellen Inhalten. Die beispielhafte künstlerische Reinszenierung der Szene durch das Toon-Shading zeigt nur eine von vielen möglichen Anwendungen der inversen Renderingfunktion für räumliche Szenen.

Abschließend lässt sich somit die erfolgreiche Umsetzung der inversen Renderingfunktion für räumliche Szenen festhalten und die Praktikabilität durch beispielhaften Anwendungen unterstreichen. Somit wird auch in diesem Kapitel die zum Anfang gesetzte Aufgabenstellung als erreicht angesehen.

5 Zusammenfassung und Ausblick

In diesem Kapitel werden zunächst die Aspekte dieser Arbeit zusammengefasst und bewertet; dies geschieht in Abschnitt 5.1. Anschließend wird in Abschnitt 5.2 ein Ausblick bezogen auf Weiterentwicklungen und mögliche Einsatzgebiete dieser Arbeit gegeben.

5.1 Zusammenfassung

Motiviert durch die Vision einer computergrafikgestützten Zukunft und aufbauend auf einer Darlegung der relevanten Grundlagen und den verwandten Arbeiten, hat diese Arbeit zwei Hauptbeiträge zum Forschungsfeld der inversen Renderingfunktion geleistet.

Zum einen hat diese Arbeit den Stand der Wissenschaft der leichtgewichtigen Deep-Learning-gestützten SVBRDF-Akquise ausgehend von einem Blitzlichtfoto von einem planaren Oberflächenmaterial untersucht. Dabei wurde aufbauend auf der Arbeit von Deschaintre et al. [11] im Rahmen dieser Arbeit eine inverse Renderingfunktion für planare Oberflächen entwickelt. So wurden die Architekturen des U-Nets und der U-Net-Variante mit der anwendungsspezifischen Erweiterung rekonstruiert und ein Rendering-gestütztes Fehlermaß für das PyTorch-Framework entwickelt. Dieses baut dabei auf einer differenzierbaren Renderingkomponente auf, die das G-Buffer-Verfahren für eine physikalisch-basierte Beleuchtung mittels des Cook-Torrance-Shaders einsetzt, um eine Bildsynthese durchzuführen; auch diese Komponente wurde in dieser Arbeit im Rahmen eines PyTorch-Moduls implementiert. Darüber hinaus wurden verschiedene Datenaugmentationsmethoden thematisiert; diese Arbeit hat sowohl die Deep-Learning-gestützte generative Datenaugmentation als auch eine naive Datenaugmentationsmethode für SVBRDFs planarer Oberflächenmaterialien vorgeschlagen. Letztere wurde erfolgreich implementiert und konnte in einigen Szenarien den Stand der Wissenschaft übertreffen. Erst kürzlich erschienene Arbeiten bezüglich einer Deep-Learning-gestützten inversen Renderingfunktion für multiple Eingaben desselben Oberflächenmaterials wurden detailliert bezüglich ih-

rer Funktionsweise vorgestellt. Abschließend wurde in einer ausgiebigen Evaluierung festgestellt, dass die im Rahmen dieser Arbeit entwickelte inverse Renderingfunktion realistische Neubeleuchtungen von selbstakquirierten realen Materialien ermöglicht; die Neubeleuchtungen finden dabei aufbauend auf den jeweiligen SVBRDF-Schätzungen der trainierten Deep Neural Networks statt. Einschränkungen dabei wurden speziell bei polierten und glänzenden Oberflächen festgestellt. Zudem konnten die in dieser Arbeit entwickelten Methoden vergleichbare und z.T. bessere Ergebnisse als die der Originalarbeit [11] liefern. Ein Vergleich der U-Net-Varianten konnte darlegen, dass die anwendungsspezifische Erweiterung der U-Net-Architektur, dargestellt durch das globale Merkmalsnetzwerk, durchaus gerechtfertigt ist.

Der zweite Hauptbeitrag wird durch die Übertragung der Methoden zur Approximation der inversen Renderingfunktion für räumliche Szenen dargestellt. Hierbei wurde zunächst ein Datensatz für das Rendering räumlicher Szenen [46] restrukturiert, um ein Training der inversen Funktion zuzulassen. So wurden neben einer differenzierbaren Renderingkomponente aufbauend auf dem Blinn-Phong-Beleuchtungsmodell auch ein stochastisches Rendering-gestütztes Fehlermaß entwickelt, welches das Training einer inversen Renderingfunktion für räumliche Szenen ermöglicht. Dabei auftretende Defizite in den geschätzten Materialkarten konnten mittels einer anwendungsspezifischen Regularisierungsmethode, die im Rahmen dieser Arbeit entwickelt wurde, größtenteils beseitigt werden. Diese Filter-basierte Regularisierungsmethode konnte gängige L1-Regularisierungsverfahren sowohl quantitativ als auch qualitativ übertreffen. Des Weiteren konnte die Praktikabilität der inversen Renderingfunktion im Rahmen einer qualitativen Evaluierung dargestellt werden; dabei wurde ausgehend von der SVBRDF-Akquise einer beleuchteten Szene eine Neubeleuchtung vorgenommen; zudem wurde eine Augmentation der ursprünglichen Szene mit einer zusätzlichen Lichtquelle dargestellt und ein komplett anderes Beleuchtungsverfahren eingesetzt, um eine künstlerische Reinszenierung der ursprünglichen Szene zu ermöglichen.

Insgesamt konnte mit diesen Errungenschaften ein weiterer Schritt in Richtung Ken Perlin's Vision [53] getätigt und eine Reihe an möglichen Weiterentwicklungen ermöglicht werden.

5.2 Ausblick

In dieser Arbeit wurde die inverse Renderingfunktion für planare Oberflächen ausgehend von einem einzelnen Blitzlichtfoto untersucht. Die umgesetzten Methoden

konnten qualitativ hochwertige Resultate liefern; Probleme traten jedoch speziell bei polierten Oberflächen auf. Möglicherweise können dabei auftretende Ambiguitäten durch die vorgestellten Methoden anhand von multiplen Eingabefotos aufgelöst und solche Defizite aufgehoben werden.

Ein wichtiger Aspekt beim Training wurde durch das Rendering-gestützte Fehlermaß dargestellt. Verschiedene Arbeiten beschäftigen sich mit Problemstellungen, bei denen der Einsatz eines solchen Fehlermaßes untersucht werden kann. Beispielsweise könnte ein Deep Neural Network für die Transformation einer SVBRDF-Parametrisierung für ein anderes Beleuchtungsmodell mittels des Rendering-gestützten Fehlermaßes trainiert und mit verwandten Arbeiten [68] verglichen werden.

Für die räumliche inverse Renderingfunktion wurden einige Einschränkungen vorgenommen. Zum einen wurde ausgehend von dem verfügbaren Datensatz das Blinn-Phong-Modell eingesetzt; dies könnte die robuste SVBRDF-Akquise von realen räumlichen Szenen einschränken. Durch die Akquise eines physikalisch-basierten Datensatzes kann beispielsweise das Cook-Torrance-Beleuchtungsmodell eingesetzt werden, welches sich für planare Oberflächen bewährt hat. Des Weiteren wurde der Fokus in dieser Arbeit auf die diffuse Materialrekonstruktion gelegt; weitere Arbeiten könnten die Rekonstruktion auf eine vollständigen SVBRDF ausweiten und evaluieren. Aktuell wird dem Deep Neural Network zudem die Positionskarte als unterstützende Eingabe beigelegt; verwandte Arbeiten führen eine Deep-Learning-gestützte Rekonstruktion der pixelweisen Tiefe durch [36]. Die Kombination solcher Verfahren und der Rekonstruktion der SVBRDF gemeinsam in einem Deep Neural Network könnte die Voraussetzung einer genauen Positionskarte möglicherweise aufheben.

Die räumliche differenzierbare Renderingkomponente berechnet aktuell keine Schatten oder weitere komplexe Lichtphänomene. Auch wird durch den Einsatz einer BRDF die Einschränkung auf undurchsichtige Objekte vorgenommen [21]; zudem kann durch die BRDF auch keine Volumenstreuung rekonstruiert werden [24]. Zusammen mit der aktuellen Einschränkung von einem Tiefenwert pro Pixel können somit Fensterscheiben, halbdurchsichtige Objekte und weitere Phänomene, die auf ähnlichen Aspekten beruhen, nicht bei der inversen Renderingfunktion betrachtet werden. Eine Lösung dieser Herausforderungen setzt eine weitaus komplexere physikalisch-basierte differenzierbare Renderingkomponente voraus, deren Resultate in zukünftigen Arbeiten möglicherweise durch eine BSSRDF [21] repräsentiert werden können.

Ein weiterer Aspekt besteht darin, die Methoden des *Simultaneous Localization and*

Mapping (SLAM) [14] in Kombination mit dem Deep Learning auf den daraus resultierenden Graphstrukturen zu nutzen, um so SVBRDFs für Punktemengen berechnen zu können; dabei gibt es bereits sowohl Ansätze für das Lernen auf Graphstrukturen [15] als auch für die BRDF-Schätzung auf Punktemengen [30].

Aktuell konnte aufgrund des eingesetzten Datensatzes [46] kein Training auf hochfrequenten Texturen für die diffuse Grundfarbe eingesetzt werden, da der Datensatz aus uniform gefärbten Objekten besteht. Moderne Videospiele besitzen mittlerweile hochrealistische Umgebungen; für einen gegebenen Frame eines Videospiele können pixelgenaue Ground-Truth-Daten akquiriert [62] und für das Training einer räumlichen inversen Renderingfunktion eingesetzt werden.

Die tatsächliche Relevanz der inversen Renderingfunktion wird sich in den kommenden Jahren herausstellen; durch diese Arbeit konnte die Praktikabilität der räumlichen inversen Renderingfunktion dargelegt werden. Folglich wird hier die Vermutung angestellt, dass die computergrafischen Anwendungen der Zukunft für eine realistische Interaktion von virtuellen und realen Objekten zwangsweise auf die hier vorgestellten Methoden bzw. diesbezüglich verwandte Verfahren zurückgreifen müssen.

A Externes GPU-gestütztes Rendering der SVBRDF-Schätzungen

Im Folgenden wird die Umsetzung der GPU-gestützten Synthese von beleuchteten Bildern zur qualitativen Untersuchung der SVBRDF-Schätzungen der jeweiligen Deep Neural Networks vorgestellt.

Hierzu wurde das Online-Tool Shadertoy¹ eingesetzt, welches die Bearbeitung von benutzerdefinierten Beleuchtungsprogrammen – die sogenannten Fragmentshader – direkt im Browser zulässt. Die Implementierung der Shader wurde dabei in der *OpenGL Shading Language* (GLSL) [29] vorgenommen. Für den Einsatz von benutzerdefinierten Texturen wurde das Browser-Plugin *Shadertoy Custom Textures*² eingesetzt.

Hierbei wurde sowohl der physikalisch-basierte Cook-Torrance-Shader für die Evaluierung der SVBRDF-Schätzungen der inversen Renderingfunktion für planare Oberflächen als auch das Blinn-Phong-Modell umgesetzt; Letzteres kam im Rahmen der Evaluierung der Schätzungen der inversen Renderingfunktion für räumliche Szenen zum Einsatz.

Abbildung A.1 stellt die Benutzeroberfläche des Online-Tools dar. Auf der rechten Seite befindet sich ein Editor, mittels dessen der GLSL-Code des Shaders direkt im Browser bearbeitet werden kann; hierbei wird der Zugriff auf bis zu vier verschiedene Texturen ermöglicht. Die Texturen können unterhalb des Editors festgelegt werden; durch das Plugin *Shadertoy Custom Textures* können hierbei mittels Drag-and-Drop sämtliche Texturen – insbesondere die SVBRDF-Schätzungen – eingesetzt werden. Auf der linken Seite wird das beleuchtete Resultat angezeigt.

Das Online-Tool ermöglicht den Zugriff auf die Mauszeigerposition im Rahmen des GLSL-Shaders. Für die Evaluierung wurde die Zeigerposition für die Bestim-

¹Shadertoy <https://www.shadertoy.com/> – Zugriffsdatum: 26. September 2019

²Shadertoy Custom Textures <https://github.com/andrewhills/ShadertoyCustomTextures> – Zugriffsdatum: 26. September 2019

mung der Lichtquellenposition eingesetzt, um eine interaktive Analyse der SVBRDF-Schätzungen aus verschiedenen Winkeln zu ermöglichen.

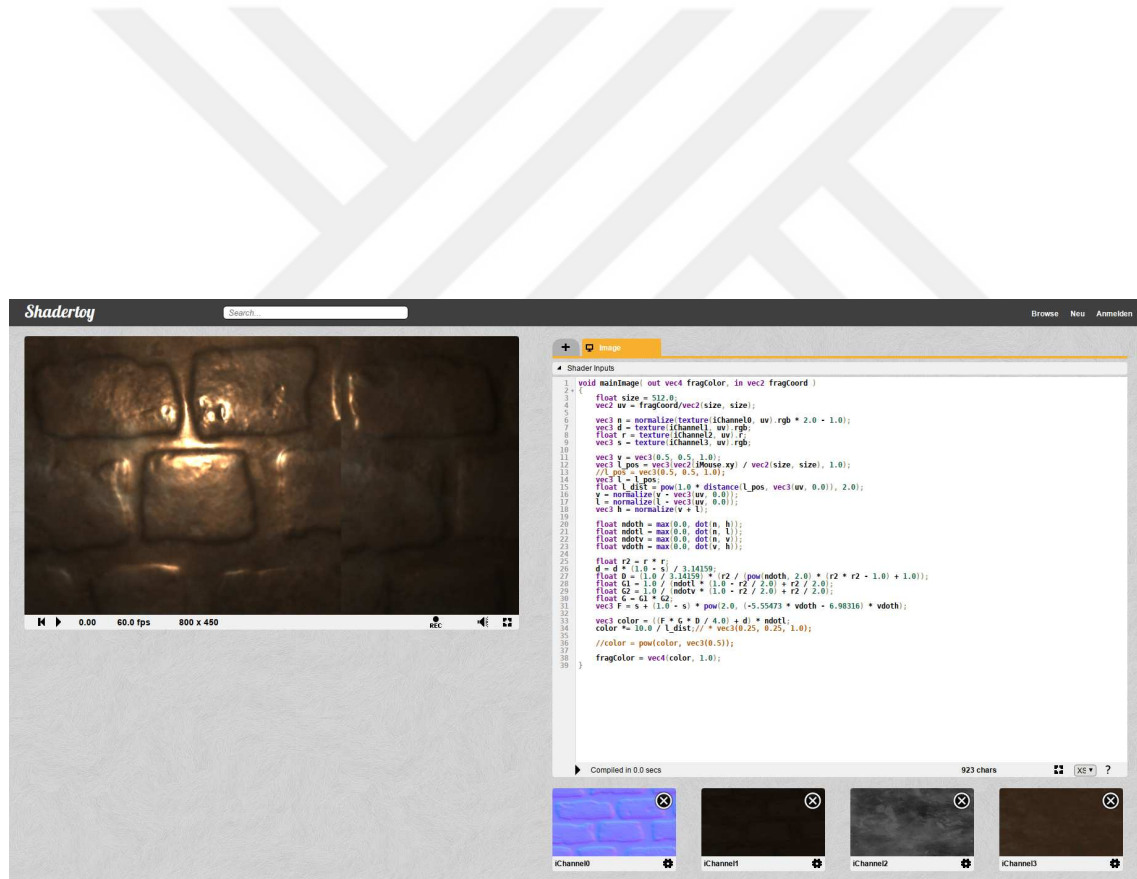


Abbildung A.1: Interaktive qualitative Evaluierung der SVBRDF-Schätzungen mittels Shadertoy³. Rechts wird der GLSL-Quellcode für die Beleuchtungsbe-
rechnung eingegeben. Darunter können die Texturen, in diesem Fall die
SVBRDF eines planaren Oberflächenmaterials, festgelegt werden. Auf
der linken Seite wird das beleuchtete Resultat dargestellt.

³Shadertoy <https://www.shadertoy.com/new> – Zugriffsdatum: 11. November 2019

Abbildungsverzeichnis

1.1	Neubeleuchtungen einer Holztischplatte	5
1.2	Anwendungen der räumlichen inversen Renderingfunktion	5
2.1	Komponenten der Reflexion	10
2.2	Relevante Vektoren für die Beleuchtungsberechnung	13
3.1	Datum bestehend aus dem Rendering und der entsprechenden SVBRDF	24
3.2	Übersicht über die zentralen Verfahren des dritten Kapitels	26
3.3	Die U-Net-Architektur	29
3.4	Detaillierte Struktur zweier aufeinanderfolgender Encoderschichten des U-Nets	31
3.5	Detaillierte Struktur einer Decoderschicht	31
3.6	Flecken auf den Ausgabekarten	33
3.7	Ermitteln der Durchschnittsfarbe	33
3.8	U-Net mit parallelem globalem Merkmalsnetzwerk	34
3.9	Detaillierte Struktur zweier aufeinanderfolgender Encoderschichten mit parallelem globalem Merkmalsnetzwerk	35
3.10	Detaillierte Struktur einer Decoderschicht mit parallelem globalem Merkmalsnetzwerk	36
3.11	Glättung eines synthetischen Blitzlichtfotos	38
3.12	Renormalisierung der geschätzten Normalenkarte	39
3.13	Der Renderingsfehler	42
3.14	Renderer für planare Oberflächen	43
3.15	Koordinatensystem und relevante Vektoren der planaren Oberfläche .	43
3.16	Beleuchtung eines Datums im Rahmen des Trainings in unterschied- lichen Konfigurationen für das Rendering-gestützte Fehlermaß	49
3.17	Generierung eines gelabelten Datums.	53
3.18	Architektur paralleler U-Nets mit globalen Merkmalsnetzen	57
3.19	Eingabedatum für die qualitative Analyse der U-Net-Varianten	63
3.20	Qualitativer Vergleich der Ergebnisse der jeweiligen Architekturen . .	65

3.21	Vergleich der Renderingfehler während des Trainings	66
3.22	Vergleich der durchschnittlichen L1-Fehler während des Trainings . .	67
3.23	Blitzlichtaufnahmen realer Materialien	68
3.24	Neubeleuchtungen realer Materialien	70
3.25	Qualitativer Vergleich mit der Originalarbeit	75
3.26	Direkter Vergleich der Neubeleuchtungen	76
3.27	Blitzlichtaufnahmen realer Materialien für den qualitativen Vergleich	77
3.28	Neubeleuchtungen realer Materialien im Vergleich	79
4.1	Datum einer räumlichen Szene	82
4.2	Angepasste U-Net-Architektur für die inverse Renderingfunktion für räumliche Szenen	86
4.3	Schematische Darstellung des Szenenaufbaus einer räumlichen Szene .	91
4.4	Beleuchtung einer Szene im Rahmen des Trainings in unterschiedli- chen Konfigurationen für das räumliche Rendering-gestützte Fehlermaß	92
4.5	Durchschnittlicher Renderingfehler der räumlichen inversen Rende- ringfunktion	99
4.7	Qualitative Ergebnisse der räumlichen inversen Renderingfunktion . .	102
4.8	Qualitativer Vergleich der Ergebnisse der räumlichen inversen Rende- ringfunktion	103
4.9	Gerenderte Szene aus einer Evaluierungswelt	110
4.10	Qualitativer Vergleich der Regularisierungsverfahren	111
4.11	Verschiedene Anwendungen der räumlichen inversen Renderingfunktion	112
A.1	Interaktive qualitative Evaluierung der SVBRDF-Schätzungen mit- tels Shadertoy	120

Algorithmenverzeichnis

3.1	Der Cook-Torrance-Shader	46
3.2	Generierung eines Vektors aus der Kosinusverteilung	48



Literaturverzeichnis

- [1] ABADI, Martín; AGARWAL, Ashish; BARHAM, Paul; BREVDO, Eugene; CHEN, Zhifeng; CITRO, Craig; CORRADO, Gregory S.; DAVIS, Andy; DEAN, Jeffrey; DEVIN, Matthieu; GHEMAWAT, Sanjay; GOODFELLOW, Ian J.; HARP, Andrew; IRVING, Geoffrey; ISARD, Michael; JIA, Yangqing; JÓZEFOWICZ, Rafal; KAISER, Lukasz; KUDLUR, Manjunath; LEVENBERG, Josh; MANÉ, Dan; MONGA, Rajat; MOORE, Sherry; MURRAY, Derek G.; OLAH, Chris; SCHUSTER, Mike; SHLENS, Jonathon; STEINER, Benoit; SUTSKEVER, Ilya; TALWAR, Kunal; TUCKER, Paul A.; VANHOUCKE, Vincent; VASUDEVAN, Vijay; VIÉGAS, Fernanda B.; VINYALS, Oriol; WARDEN, Pete; WATTENBERG, Martin; WICKE, Martin; YU, Yuan; ZHENG, Xiaoqiang: TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. In: *CoRR* abs/1603.04467 (2016)
- [2] AITTALA, Miika; WEYRICH, Tim; LEHTINEN, Jaakko: Two-shot SVBRDF Capture for Stationary Materials. In: *ACM Trans. Graph.* 34 (2015), Juli, Nr. 4, S. 110:1–110:13
- [3] BAJURA, Michael; FUCHS, Henry; OHBUCHI, Ryutarou: Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery Within the Patient. In: *SIGGRAPH Comput. Graph.* 26 (1992), Juli, Nr. 2, S. 203–210
- [4] BARLA, Pascal; THOLLOT, Joëlle; MARKOSIAN, Lee: X-Toon: An extended toon shader. In: DECARLO, Douglas (Hrsg.); MARKOSIAN, Lee (Hrsg.): *International Symposium on Non-Photorealistic Animation and Rendering (NPAR'06)*. Annecy, France: ACM, Juni 2006
- [5] BLINN, James F.: Models of Light Reflection for Computer Synthesized Pictures. In: *SIGGRAPH Comput. Graph.* 11 (1977), Juli, Nr. 2, S. 192–198
- [6] BURLEY, Brent; WALT DISNEY ANIMATION STUDIOS: Physically-Based Shading at Disney / Walt Disney Animation Studios. URL https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf, 08 2012. – Forschungsbericht

- [7] COHEN, Michael F.; WALLACE, John; HANRAHAN, Pat: *Radiosity and Realistic Image Synthesis*. San Diego, CA, USA: Academic Press Professional, Inc., 1993
- [8] COOK, R. L.; TORRANCE, K. E.: A Reflectance Model for Computer Graphics. In: *ACM Trans. Graph.* 1 (1982), Januar, Nr. 1, S. 7–24
- [9] DANIELSSON, Per-Erik; SEGER, Olle: Generalized and Separable Sobel Operators. In: FREEMAN, Herbert (Hrsg.): *Machine Vision for Three-Dimensional Scenes*. Academic Press, 1990, S. 347 – 379
- [10] DEBEVEC, Paul: Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. In: *ACM SIGGRAPH 2008 Classes*. New York, NY, USA: ACM, 2008 (SIGGRAPH '08), S. 32:1–32:10
- [11] DESCHAINTE, Valentin; AITTALA, Miika; DURAND, Fredo; DRETTAKIS, George; BOUSSEAU, Adrien: Single-Image SVBRDF Capture with a Rendering-Aware Deep Network. In: *ACM Trans. Graph.* 37 (2018), Juli, Nr. 4, S. 128:1–128:15
- [12] DESCHAINTE, Valentin; AITTALA, Miika; DURAND, Frédo; DRETTAKIS, George; BOUSSEAU, Adrien: Flexible SVBRDF Capture with a Multi-Image Deep Network. In: *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 38 (2019), July, Nr. 4
- [13] DUMOULIN, Vincent; VISIN, Francesco: *A guide to convolution arithmetic for deep learning*. 2016
- [14] DURRANT-WHYTE, H.; BAILEY, T.: Simultaneous localization and mapping: part I. In: *IEEE Robotics Automation Magazine* 13 (2006), June, Nr. 2, S. 99–110
- [15] FEY, Matthias; LENSSEN, Jan E.: Fast Graph Representation Learning with PyTorch Geometric. In: *CoRR* abs/1903.02428 (2019)
- [16] GAO, Duan; LI, Xiao; DONG, Yue; PEERS, Pieter; XU, Kun; TONG, Xin: Deep Inverse Rendering for High-resolution SVBRDF Estimation from an Arbitrary Number of Images. In: *ACM Trans. Graph.* 38 (2019), Juli, Nr. 4, S. 134:1–134:15

- [17] GEORGOULIS, Stamatis; REMATAS, Konstantinos; RITSCHER, Tobias; GAVVES, Efstratios; FRITZ, Mario; VAN GOOL, Luc; TUYTELAARS, Tinne: Reflectance and natural illumination from single-material specular objects using deep learning. In: *IEEE transactions on pattern analysis and machine intelligence* 40 (2017), Nr. 8, S. 1932–1947
- [18] GHOSH, Abhijeet: *Cook-Torrance Model*. S. 146–152. In: IKEUCHI, Katsushi (Hrsg.): *Computer Vision: A Reference Guide*. Boston, MA: Springer US, 2014
- [19] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron: *Deep Learning*. MIT Press, 2016. – <http://www.deeplearningbook.org>
- [20] GOODFELLOW, Ian; POUGET-ABADIE, Jean; MIRZA, Mehdi; XU, Bing; WARDE-FARLEY, David; OZAIR, Sherjil; COURVILLE, Aaron; BENGIO, Yoshua: Generative Adversarial Nets. In: GHAHRAMANI, Z. (Hrsg.); WELLING, M. (Hrsg.); CORTES, C. (Hrsg.); LAWRENCE, N. D. (Hrsg.); WEINBERGER, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, S. 2672–2680
- [21] GUARNERA, D.; GUARNERA, G. C.; GHOSH, A.; DENK, C.; GLENCROSS, M.: BRDF Representation and Acquisition. In: *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: State of the Art Reports*. Goslar Germany, Germany: Eurographics Association, 2016 (EG '16), S. 625–650
- [22] GUO, Kaiwen; XU, Feng; YU, Tao; LIU, Xiaoyang; DAI, Qionghai; LIU, Yebin: Real-Time Geometry, Albedo, and Motion Reconstruction Using a Single RGB-D Camera. In: *ACM Trans. Graph.* 36 (2017), Juni, Nr. 4
- [23] ISOLA, Phillip; ZHU, Jun-Yan; ZHOU, Tinghui; EFROS, Alexei A.: Image-to-Image Translation with Conditional Adversarial Networks. In: *CoRR* abs/1611.07004 (2016)
- [24] JENSEN, Henrik W.; MARSCHNER, Stephen R.; LEVOY, Marc; HANRAHAN, Pat: A Practical Model for Subsurface Light Transport. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 2001 (SIGGRAPH '01), S. 511–518
- [25] KAJIYA, James T.: The Rendering Equation. In: *SIGGRAPH Comput. Graph.* 20 (1986), August, Nr. 4, S. 143–150

- [26] KARIS, Brian: Real Shading in Unreal Engine 4 / Epic Games. URL https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_slides.pdf, 2013. – Forschungsbericht
- [27] KARRAS, Tero; AILA, Timo; LAINE, Samuli; LEHTINEN, Jaakko: Progressive Growing of GANs for Improved Quality, Stability, and Variation. In: *CoRR* abs/1710.10196 (2017)
- [28] KELLER, Alexander; KŘIVÁNEK, Jaroslav; NOVÁK, Jan; KAPLANYAN, Anton; SALVI, Marco: Machine Learning and Rendering. In: *ACM SIGGRAPH 2018 Courses*. New York, NY, USA: ACM, 2018 (SIGGRAPH '18), S. 19:1–19:2
- [29] KESSENICH, John; BALDWIN, Dave; ROST, Randi: *The OpenGL® Shading Language, Version 4.60.7*. Khronos Group, Inc. (Veranst.), Juli 2019. – URL <https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.60.pdf>
- [30] KIM, Kihwan; GU, Jinwei; TYREE, Stephen; MOLCHANOV, Pavlo; NIESSNER, Matthias; KAUTZ, Jan: A Lightweight Approach for On-the-Fly Reflectance Estimation. In: *CoRR* abs/1705.07162 (2017)
- [31] KINGMA, Diederik; BA, Jimmy: Adam: A Method for Stochastic Optimization. In: *International Conference on Learning Representations* (2014), 12
- [32] KINGMA, Diederik P.; WELLING, Max: Auto-Encoding Variational Bayes. In: *CoRR* abs/1312.6114 (2013)
- [33] KLAMBAUER, Günter; UNTERTHINER, Thomas; MAYR, Andreas; HOCHREITER, Sepp: Self-Normalizing Neural Networks. In: *CoRR* abs/1706.02515 (2017)
- [34] LAWRENCE, Jason: *Recovery of Reflectance Properties*. S. 663–669. In: IKEUCHI, Katsushi (Hrsg.): *Computer Vision: A Reference Guide*. Boston, MA: Springer US, 2014
- [35] LEGENDRE, Chloe; MA, Wan-Chun; FYFFE, Graham; FLYNN, John; CHARBONNEL, Laurent; BUSCH, Jay; DEBEVEC, Paul E.: DeepLight: Learning Illumination for Unconstrained Mobile Mixed Reality. In: *CoRR* abs/1904.01175 (2019)
- [36] LI, Bo; SHEN, Chunhua; DAI, Yuchao; VAN DEN HENGEL, A.; HE, Mingyi: Depth and surface normal estimation from monocular images using regression

- on deep features and hierarchical CRFs. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, S. 1119–1127
- [37] LI, Tzu-Mao; AITTALA, Miika; DURAND, Frédo; LEHTINEN, Jaakko: Differentiable Monte Carlo Ray Tracing Through Edge Sampling. In: *ACM Trans. Graph.* 37 (2018), Dezember, Nr. 6, S. 222:1–222:11
- [38] LI, Xiao; DONG, Yue; PEERS, Pieter; TONG, Xin: Modeling Surface Appearance from a Single Photograph Using Self-augmented Convolutional Neural Networks. In: *ACM Trans. Graph.* 36 (2017), Juli, Nr. 4, S. 45:1–45:11
- [39] LI, Zhengqin; SUNKAVALLI, Kalyan; CHANDRAKER, Manmohan: Materials for Masses: SVBRDF Acquisition with a Single Mobile Phone Image. In: *CoRR* abs/1804.05790 (2018)
- [40] LI, Zhengqin; XU, Zexiang; RAMAMOORTHY, Ravi; SUNKAVALLI, Kalyan; CHANDRAKER, Manmohan: Learning to Reconstruct Shape and Spatially-varying Reflectance from a Single Image. In: *ACM Trans. Graph.* 37 (2018), Dezember, Nr. 6, S. 269:1–269:11
- [41] LIU, Guilin; CEYLAN, Duygu; YUMER, Ersin; YANG, Jimei; LIEN, Jyh-Ming: Material Editing Using a Physically Based Rendering Network. In: *CoRR* abs/1708.00106 (2017)
- [42] MAAS, Andrew L.; HANNUN, Awni Y.; NG, Andrew Y.: Rectifier nonlinearities improve neural network acoustic models. In: *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013
- [43] MATUSIK, Wojciech; PFISTER, Hanspeter; BRAND, Matt; MCMILLAN, Leonard: A Data-driven Reflectance Model. In: *ACM Trans. Graph.* 22 (2003), Juli, Nr. 3, S. 759–769
- [44] MESCHÉDE, Dieter: *Gerthsen Physik*. Kap. Strahlungsfelder, S. 585–615, Springer Spektrum, 2015
- [45] MNIH, Volodymyr; KAVUKCUOGLU, Koray; SILVER, David; RUSU, Andrei A.; VENESS, Joel; BELLEMARE, Marc G.; GRAVES, Alex; RIEDMILLER, Martin; FIDJELAND, Andreas K.; OSTROVSKI, Georg u. a.: Human-level control through deep reinforcement learning. In: *Nature* 518 (2015), Nr. 7540, S. 529

- [46] NALBACH, Oliver; ARABADZHIYSKA, Elena; MEHTA, Dushyant; SEIDEL, Hans-Peter; RITSCHER, Tobias: Deep Shading: Convolutional Neural Networks for Screen-Space Shading. 36 (2017), Nr. 4
- [47] NARZT, Wolfgang; POMBERGER, Gustav; FERSCHA, Alois; KOLB, Dieter; MÜLLER, Reiner; WIEGHARDT, Jan; HÖRTNER, Horst; LINDINGER, Christopher: Augmented reality navigation systems. In: *Universal Access in the Information Society* 4 (2006), Mar, Nr. 3, S. 177–187
- [48] NGAN, Addy; DURAND, Frédo; MATUSIK, Wojciech: Experimental Analysis of BRDF Models. In: *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2005 (EGSR '05), S. 117–126
- [49] NICODEMUS, Fred E.; RICHMOND, Joseph C.; HSIA, Jack J.; GINSBERG, Irving W.; LIMPERIS, Thomas: Radiometry. USA: Jones and Bartlett Publishers, Inc., 1992, Kap. Geometrical Considerations and Nomenclature for Reflectance, S. 94–145
- [50] NIXON, Mark S.; AGUADO, Alberto S.: Chapter 3 - Basic image processing operations. In: NIXON, Mark S. (Hrsg.); AGUADO, Alberto S. (Hrsg.): *Feature Extraction & Image Processing for Computer Vision*. Third Edition. Oxford: Academic Press, 2012, S. 83 – 136
- [51] OORD, Aäron van den; KALCHBRENNER, Ilya; VINYALS, Oriol; ESPEHOLT, Lasse; GRAVES, Alex; KAVUKCUOGLU, Koray: Conditional Image Generation with PixelCNN Decoders. In: *CoRR* abs/1606.05328 (2016)
- [52] PASZKE, Adam; GROSS, Sam; CHINTALA, Soumith; CHANAN, Gregory; YANG, Edward; DEVITO, Zachary; LIN, Zeming; DESMAISON, Alban; ANTIGA, Luca; LERER, Adam: Automatic differentiation in PyTorch. In: *NIPS 2017 Auto-diff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 12 2017
- [53] PERLIN, Ken: The Coming Age of Computer Graphics and the Evolution of Language. In: *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*. New York, NY, USA: ACM, 2014 (SUI '14), S. 1–1
- [54] PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg: 01 - Introduction. In: PHARR, Matt (Hrsg.); JAKOB, Wenzel (Hrsg.); HUMPHREYS, Greg (Hrsg.):

- Physically Based Rendering*. Third Edition. Boston: Morgan Kaufmann, 2017, S. 1 – 55
- [55] PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg: 05 - Color and Radiometry. In: PHARR, Matt (Hrsg.); JAKOB, Wenzel (Hrsg.); HUMPHREYS, Greg (Hrsg.): *Physically Based Rendering*. Third Edition. Boston: Morgan Kaufmann, 2017, S. 313 – 353
- [56] PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg: 06 - Camera Models. In: PHARR, Matt (Hrsg.); JAKOB, Wenzel (Hrsg.); HUMPHREYS, Greg (Hrsg.): *Physically Based Rendering*. Third Edition. Boston: Morgan Kaufmann, 2017, S. 355 – 399
- [57] PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg: 14 - Light Transport I: Surface Reflection. In: PHARR, Matt (Hrsg.); JAKOB, Wenzel (Hrsg.); HUMPHREYS, Greg (Hrsg.): *Physically Based Rendering*. Third Edition. Boston: Morgan Kaufmann, 2017, S. 805 – 885
- [58] PHONG, Bui T.: Illumination for Computer Generated Pictures. In: *Commun. ACM* 18 (1975), Juni, Nr. 6, S. 311–317
- [59] RAMAMOORTHY, Ravi; HANRAHAN, Pat: A Signal-processing Framework for Inverse Rendering. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 2001 (SIGGRAPH '01), S. 117–128
- [60] REINHARD, Erik; WARD, Greg; PATTANAİK, Sumanta; DEBEVEC, Paul: 01 - Introduction. In: REINHARD, Erik (Hrsg.); WARD, Greg (Hrsg.); PATTANAİK, Sumanta (Hrsg.); DEBEVEC, Paul (Hrsg.): *High Dynamic Range Imaging*. San Francisco: Morgan Kaufmann, 2006 (The Morgan Kaufmann Series in Computer Graphics), S. 1 – 17
- [61] REINHARD, Erik; WARD, Greg; PATTANAİK, Sumanta; DEBEVEC, Paul: 06 - The Human Visual System and HDR Tone Mapping. In: REINHARD, Erik (Hrsg.); WARD, Greg (Hrsg.); PATTANAİK, Sumanta (Hrsg.); DEBEVEC, Paul (Hrsg.): *High Dynamic Range Imaging*. San Francisco: Morgan Kaufmann, 2006 (The Morgan Kaufmann Series in Computer Graphics), S. 187 – 221
- [62] RICHTER, Stephan R.; VINEET, Vibhav; ROTH, Stefan; KOLTUN, Vladlen: Playing for Data: Ground Truth from Computer Games. In: LEIBE, Bastian (Hrsg.); MATAS, Jiri (Hrsg.); SEBE, Nicu (Hrsg.); WELLING, Max (Hrsg.):

- European Conference on Computer Vision (ECCV)* Bd. 9906, Springer International Publishing, 2016, S. 102–118
- [63] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *CoRR* abs/1505.04597 (2015)
- [64] ROSENTHALER, Lukas: Digital Image Processing. In: PERES, Michael R. (Hrsg.): *The Focal Encyclopedia of Photography*. Fourth Edition. Boston: Focal Press, 2007, S. 403 – 411
- [65] SAITO, Takafumi; TAKAHASHI, Tokiichiro: Comprehensible Rendering of 3-D Shapes. In: *SIGGRAPH Comput. Graph.* 24 (1990), September, Nr. 4, S. 197–206
- [66] SILVER, David; HUANG, Aja; MADDISON, Christopher J.; GUEZ, Arthur; SIFRE, Laurent; DRIESSCHE, George van den; SCHRITTWIESER, Julian; ANTONOGLOU, Ioannis; PANNEERSHELVAM, Veda; LANCTOT, Marc; DIELEMAN, Sander; GREWE, Dominik; NHAM, John; KALCHBRENNER, Nal; SUTSKEVER, Ilya; LILLICRAP, Timothy; LEACH, Madeleine; KAVUKCUOGLU, Koray; GRAEPEL, Thore; HASSABIS, Demis: Mastering the game of Go with deep neural networks and tree search. In: *Nature* 529 (2016), S. 484–503
- [67] SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* 15 (2014), S. 1929–1958
- [68] SZTRAJMAN, Alejandro; KRIVÁNEK, Jaroslav; WILKIE, Alexander; WEYRICH, Tim: Image-based remapping of spatially-varying material appearance. In: *CoRR* abs/1808.06715 (2018)
- [69] SÜSSTRUNK, Sabine: Color Spaces, Color Encodings, and Color Image Encodings. In: PERES, Michael R. (Hrsg.): *The Focal Encyclopedia of Photography*. Fourth Edition. Boston: Focal Press, 2007, S. 394 – 398
- [70] SÜSSTRUNK, Sabine: Colorimetry. In: PERES, Michael R. (Hrsg.): *The Focal Encyclopedia of Photography*. Fourth Edition. Boston: Focal Press, 2007, S. 388 – 393

- [71] SÜSSTRUNK, Sabine: Image Formation. In: PERES, Michael R. (Hrsg.): *The Focal Encyclopedia of Photography*. Fourth Edition. Boston: Focal Press, 2007, S. 382 – 388
- [72] TAN, Ping: *Phong Reflectance Model*. S. 592–594. In: IKEUCHI, Katsushi (Hrsg.): *Computer Vision: A Reference Guide*. Boston, MA: Springer US, 2014
- [73] THE KHRONOS® VULKAN WORKING GROUP: *Vulkan® - 1.1.123 - A Specification*. The Khronos Group Inc. (Veranst.), 09 2019. – URL <https://www.khronos.org/registry/vulkan/specs/1.1/pdf/vkspec.pdf>
- [74] TORRANCE, Kenneth E.; SPARROW, Ephraim M.: Theory for off-specular reflection from roughened surfaces. In: *Josa* 57 (1967), Nr. 9, S. 1105–1114
- [75] TSURUGA, Joji; RENAUD-HOUDE, Eric: The Human Race. In: *ACM SIGGRAPH 2017 Real Time Live!* New York, NY, USA: ACM, 2017 (SIGGRAPH '17), S. 26–26
- [76] ULYANOV, Dmitry; VEDALDI, Andrea; LEMPITSKY, Victor S.: Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis. In: *CoRR* abs/1701.02096 (2017)
- [77] WARD, Gregory J.: Measuring and Modeling Anisotropic Reflection. In: *SIGGRAPH Comput. Graph.* 26 (1992), Juli, Nr. 2, S. 265–272
- [78] WESTLAND, Stephen; CHEUNG, Vien: *RGB Systems*. S. 171–177. In: CHEN, Janglin (Hrsg.); CRANTON, Wayne (Hrsg.); FIHN, Mark (Hrsg.): *Handbook of Visual Display Technology*. Cham: Springer International Publishing, 2016
- [79] WHITTED, Turner; WEIMER, David M.: A Software Testbed for the Development of 3D Raster Graphics Systems. In: *ACM Trans. Graph.* 1 (1982), Januar, Nr. 1, S. 43–58
- [80] XU, Zexiang; NIELSEN, Jannik B.; YU, Jiyang; JENSEN, Henrik W.; RAMAMOORTHY, Ravi: Minimal BRDF Sampling for Two-shot Near-field Reflectance Acquisition. In: *ACM Trans. Graph.* 35 (2016), November, Nr. 6, S. 188:1–188:12
- [81] YU, Ye; SMITH, William A. P.: PVNN: A Neural Network Library for Photometric Vision. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Oct 2017, S. 526–535