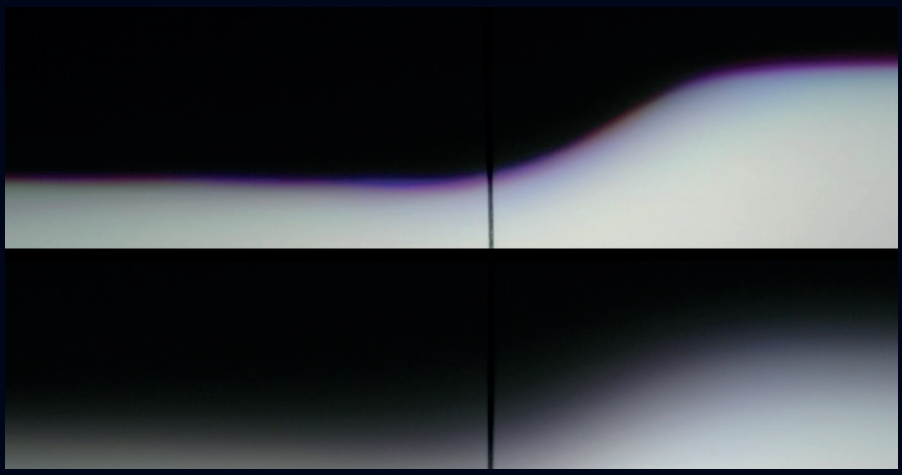
BAND 25 | SPEKTRUM DER LICHTTECHNIK

DENNIS ZIMMERMANN

VERBESSERUNG DER PROZESSKETTE ZUR HERSTELLUNG MIKROSTRUKTURIERTER LINSEN FÜR AUTOMOBILE SCHEINWERFER



KIT Scientific Publishing

Dennis Zimmermann

Verbesserung der Prozesskette zur Herstellung mikrostrukturierter Linsen für automobile Scheinwerfer

Lichttechnisches Institut
Karlsruher Institut für Technologie (KIT)

Verbesserung der Prozesskette zur Herstellung mikrostrukturierter Linsen für automobile Scheinwerfer

von
Dennis Zimmermann

KIT Scientific Publishing

Karlsruher Institut für Technologie
Lichttechnisches Institut

Verbesserung der Prozesskette zur Herstellung
mikrostrukturierter Linsen für automobile Scheinwerfer

Zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
von der KIT-Fakultät für Elektrotechnik und Informationstechnik des
Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von M.Sc. Dennis Zimmermann

Tag der mündlichen Prüfung: 17. Juli 2020
Hauptreferent: Prof. Dr. Cornelius Neumann
Korreferent: Prof. Dr. Wilhelm Stork

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

www.ksp.kit.edu

Print on Demand 2021 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 2195-1152
ISBN 978-3-7315-1058-1
DOI 10.5445/KSP/1000125386

Es wäre so einfach, wenn du einfach machst
Anstatt dich nur zu fragen, wie ein andrer es schafft
Es wäre so einfach, wenn du nicht mehr quatschst
Und anstatt was du sagst lieber ganz einfach machst
Kontra K

DANKSAGUNG

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meiner Promotionszeit begleitet und unterstützt haben.

Zu aller erst möchte ich mich bei Herrn Prof. Dr. Cornelius Neumann für die Möglichkeit zur Promotion bedanken. Vielen Dank für die Betreuung, Unterstützung und vielen hilfreichen Diskussionen und Anmerkungen zu dieser Arbeit.

An zweiter Stelle möchte ich Prof. Dr. Wilhelm Stork dafür danken, dass er sich bereit erklärt hat, das Amt des Zweitgutachters zu übernehmen.

Ein großer Dank geht an Piet Risthaus für die zahlreichen Diskussionen und viel Feedback zu der Auslegung von Mikrostrukturen.

Meinen Kollegen Uwe Haack, Christian Plattfaut, Bernd Rudolf, Florian Stapel und Markus Unland möchte ich für die ständige Unterstützung bedanken, insbesondere Harald Zerhau-Dreihöfer für viele hilfreiche Anmerkungen.

Aniella Thoma hat mir viele Hinweise zum Thema menschliche Wahrnehmung gegeben. Hierfür vielen Dank!

Andre Schellbach, David Duhme, Marc Kaup und Henry Vogeler möchte ich für den täglichen Kaffeeklatsch danken und Markus Giehl für zahlreiche entspannende Mopped-Touren.

Zu guter Letzt möchte ich mich bei meiner Familie für die ständige Unterstützung und sehr viel Geduld über meine ganze Promotionszeit hinweg bedanken.

ABSTRACT

The transition between the light and dark areas of the luminous intensity distribution of a headlamp needs to fulfil statutory regulations. These regulations restrict the sharpness of the transition. Projection headlamps generate a sharp cutoff line. Reducing the sharpness is done by adding a structure to the light exit surface of the lens with an amplitude of a few micrometers. A long and error-prone process was used to design and manufacture structured lenses. The process used so far and the problems associated with it are explained. Afterwards a new, faster and less error-prone process is presented. An algorithm was developed to structure surfaces within the computer-aided design software directly. The lenses were manufactured using injection moulding. The moulds were manufactured by an ultra precision turning machine. A description of the machine and the data interface is given. The process being used until now did not include a qualification of the moulds. An optical measuring instrument was used to scan the structured surface. The result of the scan is a point grid. To verify the structure by a photometric simulation a surface fitting was done. The algorithm used for fitting is presented. Until now, the design of the structure was done manually by a trial-and-error process. The time for a photometric simulation was reduced to make the design by a genetic algorithm possible. Therefore, the calculation of the B-spline basis functions was vectorized. The function of the new process is demonstrated with two examples.

This research and development project was funded by the German Federal Ministry of Education and Research (BMBF) within the “Innovations for Tomorrow’s Production, Services, and Work“ program (funding number 02P14A100ff) and implemented by the Project Management Agency Karlsruhe (PTKA).

KURZFASSUNG

Der Übergang des hellen in den dunklen Bereich einer automobilen Scheinwerfer-Lichtverteilung unterliegt gesetzlichen Auflagen. Die Auflagen beschränken, wie scharf der Übergang sein darf. Scheinwerfer-Projektionssysteme erzeugen zunächst eine zu scharfe Hell-Dunkel-Grenze. Um diese aufzuweichen, wird die Lichtaustrittsseite der Linse mit einer streuenden Struktur versehen, deren Amplitude im Bereich weniger Mikrometer liegt. Die Auslegung und Herstellung der strukturierten Linsen erfolgte bisher durch eine lange und fehleranfällige Prozesskette. Die bisher verwendete Prozesskette und die damit verbundenen Probleme werden kurz erläutert. Anschließend wird eine neue, schnelle und fehlerunanfälligere Prozesskette vorgestellt. Es wurde ein Algorithmus entwickelt, um im Computer-Aided Design Linsenflächen direkt strukturieren zu können. Die Herstellung der Linsen erfolgte durch ein Spritzgussverfahren. Dazu wurden Werkzeugeinsätze benötigt. Die zur Herstellung verwendete Ultrapräzisionsdrehmaschine wird kurz vorgestellt und die Datenschnittstelle erklärt. In der bisher verwendeten Prozesskette existierte keine Möglichkeit die Werkzeugeinsätze zu qualifizieren. Um die lichttechnische Wirkung der eingebrachten Struktur zu überprüfen, wurde eine optische Messmaschine verwendet. Das Ergebnis der Vermessung ist ein Punktgitter der strukturierten Oberfläche. Um eine lichttechnische Simulation durchzuführen, wurde eine Fläche ausgehend von dem Punktgitter berechnet. Der dazu verwendete Algorithmus wird vorgestellt. Bisher erfolgte die Auslegung der Strukturen in einem manuellen „Trial and Error"-Prozess. Um eine Auslegung durch genetische Algorithmen zu ermöglichen, wurde die Zeit für eine lichttechnische Simulation reduziert. Dazu wurde die Berechnung der B-Spline Basis-Funktionen vektorisiert. Die Funktion der neuen

Prozesskette wird anhand von zwei Beispielen demonstriert. Dieses Forschungs- und Entwicklungsprojekt wurde durch das Bundesministerium für Bildung und Forschung (BMBF) im Programm „Innovationen für die Produktion, Dienstleistung und Arbeit von morgen“ (Förderkennzeichen 02P14A100ff) gefördert und vom Projektträger Karlsruhe (PTKA) betreut.

INHALTSVERZEICHNIS

KAPITEL 1

EINLEITUNG

Seit 2007 die ersten Serienfahrzeuge mit LED-Scheinwerfern vom Band liefen, haben diese LED-Scheinwerfer immer mehr Marktanteile gewonnen [1, 2]. Durch den Einsatz von LEDs mussten neue Konzepte für Scheinwerfer erarbeitet werden [3]. Da der Lichtstrom einer einzelnen LED zu gering ist, ist es notwendig, die Straße mit mehreren LEDs zu beleuchten. Dazu werden häufig Projektionssysteme verwendet (vgl. Abb. 1.1). Ein Projektionssystem sammelt über einen Reflektor oder eine Primäroptik das Licht der LEDs und konzentriert es in der Ebene einer Blende [4]. Die Blendenebene wird durch eine Linse auf die Straße abgebildet. Die Verwendung von Freiformlinsen ergeben Freiheiten bei Design und geometrischer Auslegung, wie z. B. die Reduzierung der Baugröße oder die Abbildung mehrerer LEDs durch eine Linse [5]. Durch die Kontur der Blende generieren Projektionssysteme eine sehr scharfe Hell-Dunkel-Grenze (HDG) in der Lichtverteilung, wie in Abb. 1.2 dargestellt [6]. Eine scharfe HDG reduziert den visuellen Komfort für den Fahrer. Um Objekte zu erkennen, sucht der Mensch nach Kanten bzw. Kontrasten [7]. Eine scharfe HDG führt dazu, dass der Fahrer seinen Blick ständig auf die HDG fokussiert. Um die HDG aufzuweichen, wird die Lichtaustrittsseite der Linse mit einer licht-streuenden Struktur überlagert [6]. Da die Amplitude der Strukturen im Bereich weniger Mikrometer ist, werden diese als Mikrostrukturen bezeichnet. Eine weiche HDG ist in Abb. 1.2 dargestellt. Die „Economic Commission for Europe“ (ECE) stellt gesetzliche Vorgaben, damit ein Scheinwerfer zugelassen wird [8]. Unter anderem gibt die Vorschrift an, wie die Schärfe der HDG

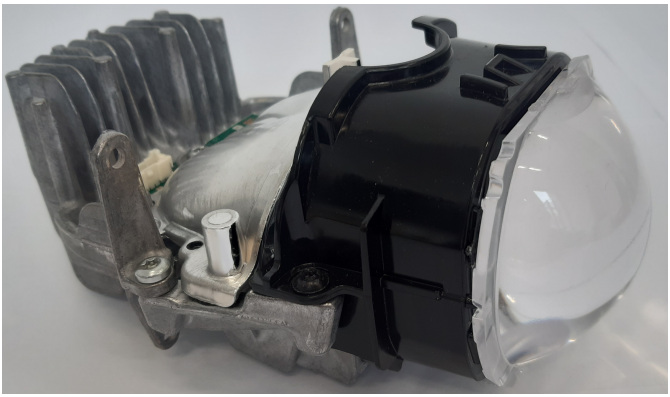


Abbildung 1.1: BiLED-Scheinwerfermodul

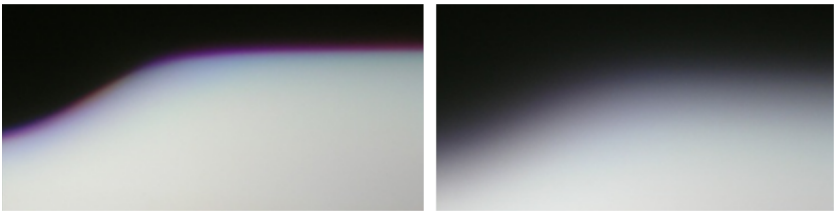


Abbildung 1.2: Vergleich zwischen einer scharfen und einer weichen HDG.

zu bestimmen ist und definiert den legalen Bereich für den Schärfegrad. Bei der Auslegung von Mikrostrukturen muss einerseits die Schärfe der HDG reduziert, andererseits darf durch das gestreute Licht der Gegenverkehr nicht geblendet werden [6]. Daher werden Möglichkeiten benötigt, die Mikrostrukturen gezielt auslegen zu können.

1.1 MOTIVATION UND UMFELD DER ARBEIT

In dieser Arbeit liegt der Fokus auf Kunststofflinsen. Sowohl die Auslegung als auch die Fertigung mikrostrukturierter Linsen erfolgt bei

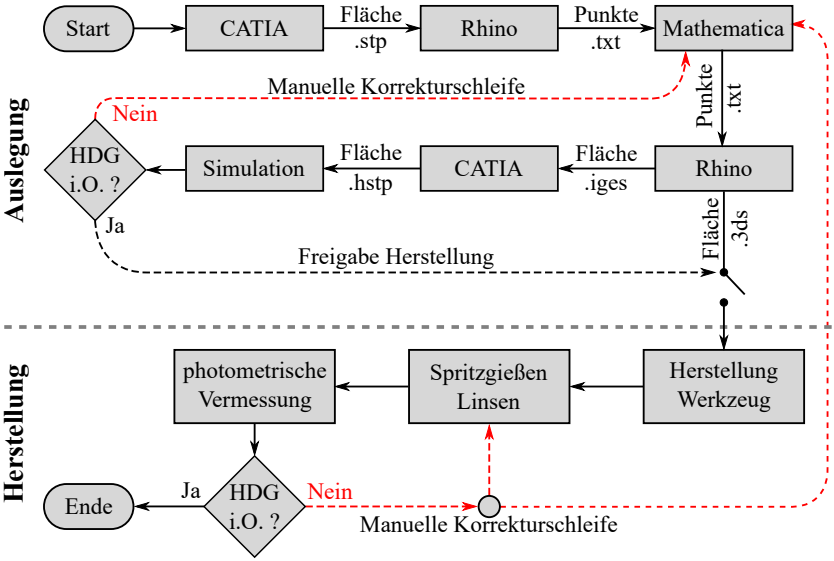


Abbildung 1.3: Bisher verwendete Prozesskette zur Herstellung mikrostrukturierter Linsen.

der Firma *HELLA GmbH & Co. KGaA* durch einen langen und fehleranfälligen Prozess. Bei der Entwicklung eines neuen Scheinwerfers wird die Schärfe der HDG zunächst nicht berücksichtigt. Das System wird mit einer nicht-strukturierten Linse ausgelegt. Wenn die Vorgaben von Kunde und Gesetz erfüllt werden, wird durch Strukturierung der Linse die Schärfe der HDG reduziert. Die Prozesskette, die zur Herstellung der strukturierten Linsen angewandt wird, ist in Abb. 1.3 dargestellt. Die Konstruktion der optischen Flächen, wie die nicht-strukturierte Linse, erfolgt in dem Computer-Aided Design (CAD) Programm CATIA [9]. Die analytisch beschriebene Linsenfläche wird in einer STEP-Datei abgespeichert und in dem Programm Rhinoceros (Rhino) geladen [10, 11]. In Rhino wird auf Basis der Fläche ein Punktgitter berechnet, dass die Fläche repräsentiert und in einer ASCII-Textdatei abgespeichert [12].

Die ASCII-Datei mit dem Punktgitter wird in dem Programm Mathematica eingeladen [13]. Mit Mathematica wird die Struktur hinzugefügt, indem die Punkte um wenige Mikrometer verschoben werden. Die verschobenen Punkte werden in einer ASCII-Datei abgespeichert und in das Programm Rhino geladen. Durch eine Fit-Funktion des Programms Rhino wird eine Fläche auf Basis des Punktgitters berechnet. Die Berechnung dauert ca. 25 min auf derzeit üblichen Prozessoren. Die berechnete Fläche wird in eine IGES-Datei abgespeichert [14]. Die IGES-Datei wird in CATIA geladen. Durch eine vom Unternehmen implementierte Schnittstelle wird die Fläche in ein *HELLA* eigenes STEP-Format abgespeichert. Die STEP-Datei wird in dem unternehmenseigenen lichttechnischen Simulationsprogramm zusammen mit den restlichen optischen Flächen des Scheinwerfermoduls geladen. Das Programm simuliert die Lichtverteilung einschließlich des Schärfegrades der HDG. Liegt der Schärfegrad außerhalb des Soll-Bereiches oder verursacht die Struktur zu viel Blendung, so werden erfahrungsgestützt die Parameter der Struktur in Mathematica angepasst. In manuellen Korrekturschleifen wird die Struktur so angepasst, dass die Vorgaben erfüllt werden. Zur Fertigung wird ausgehend von der berechneten Fläche in Rhino eine Rhino-3ds-Datei exportiert und zur Herstellung von Stahlwerkzeugen verwendet. Die Stahlwerkzeuge dienen als Formen für das Spritzgießen der Linsen [15]. Die abgeformten Linsen werden in einem gefertigten Scheinwerfermodul (vgl. Abb. 1.1) eingebaut. Die Lichtstärkeverteilung (LSV) des Moduls wird vermessen. Wenn die HDG nicht in Ordnung ist und Simulation und Vermessung voneinander abweichen, wird zunächst durch Änderung der Spritzguss-Parameter versucht, eine Verbesserung zu erreichen. Kann keine Verbesserung erreicht werden, werden die Parameter der Struktur in Mathematica verändert und erneut Stahlwerkzeuge gefertigt. Diese Korrekturschritte werden solange wiederholt, bis die HDG der gemessenen LSV in Ordnung ist.

Durch die vielen manuellen Schritte ist der Prozess langwierig. Die

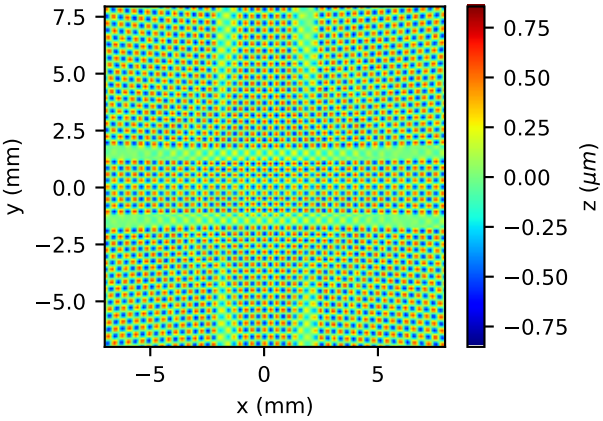


Abbildung 1.4: Die zwei vertikalen und horizontalen Streifen stellen den lokalen Verlust der Mikrostruktur durch die Übertragung von Rhino zu Catia dar.

vielen Konvertierungen und Dateiformate können für eine fehlerhafte Informationsweitergabe sorgen. Bei einer ersten Analyse zeigte sich, dass die Mikrostruktur durch die Übertragung von Rhino zu CATIA lokal verschwindet, wie in Abb. 1.4 dargestellt. Die Verwendung von Mathematica erfordert viel Erfahrung und lässt sich auf Grund der großen Freiheiten, die Mathematica bietet, schlecht standardisieren. Durch die Verwendung des kommerziellen Programms Rhino ist eine „Black Box“ in der Prozesskette vorhanden, dessen Algorithmen nicht bekannt sind. Es existiert keine Möglichkeit zu überprüfen, ob die Struktur erfolgreich in das Stahlwerkzeug eingebracht wird. Dadurch entstehen teure Korrekturschleifen, in denen wiederholt Stahlwerkzeuge gefertigt werden müssen. Für das BiLED-Modul (vgl. Abb. 1.1) wurden insgesamt acht Strukturen ausgelegt und für jede Struktur Stahlwerkzeuge gefertigt, bis die HDG des Moduls in Ordnung war. Durch die hohe Anzahl an Korrekturen kann die Laufzeit der Projekte überschritten werden und es

werden Ressourcen verschwendet.

Im Rahmen des BMBF-Förderprojektes „effiziente Photonikproduktion durch intelligente Technologie“ (ePiTec) ist das Ziel dieser Arbeit die Verbesserung der Prozesskette [16, 17]. Die neue Prozesskette soll die Anzahl an Korrekturschleifen verringern bzw. idealerweise gänzlich verhindern. Dazu sind die folgenden Ziele zu erreichen:

1. Die Auslegung von Mikrostrukturen soll direkt in dem CAD-Programm CATIA oder in dem lichttechnischen Simulationsprogramm ermöglicht werden. Durch einen Projektpartner wird die Fertigungstechnik zur Herstellung der Stahlwerkzeuge verbessert. In Zusammenarbeit mit dem Projektpartner soll eine neue Datenschnittstelle zwischen Auslegung und Herstellung implementiert werden.
2. Ein weiterer Projektpartner ermöglicht die optische Vermessung der Stahlwerkzeuge. Das Ergebnis der Messung ist ein Punktgitter. Um zu überprüfen, ob das Werkzeug in Ordnung ist, soll eine lichttechnische Simulation auf Basis des Punktgitters ermöglicht werden.
3. Die Auslegung der Mikrostrukturen soll durch den Einsatz von Optimierungsalgorithmen automatisiert werden.

1.2 STAND DER TECHNIK

Bezüglich der Auslegung von mikrostrukturierten Linsen zum Aufweichen der HDG existieren kaum wissenschaftliche Veröffentlichungen, jedoch einige Patente. HOLTZ und SCHMIDT beschreiben ein mathematisches Modell, dass einen Zusammenhang zwischen der Rauigkeit der Linsenoberfläche und der Schärfe der HDG bzw. eines Blendwertes enthält [18]. Es werden Herstellungsverfahren genannt, mit denen

die gewünschte Rauigkeit erreicht werden kann. FISCHER stellt ein Verfahren vor, bei dem die Schärfe der HDG durch punktförmige Defekte innerhalb der Linse reduziert wird [19]. Die Defekte werden durch einen Laser eingebracht. Um die Position der Defekte zu bestimmen, wird die Veränderung der HDG nach dem Einbringen eines Defektes vermessen. Mit den Informationen wird ein neuronales Netz trainiert, um die optimalen Positionen der Defekte bestimmen zu können [20]. HAMKENS berichtet über die Herstellung von Linsen durch eine Kombination aus Spritzguss- und Press-Verfahren und erwähnt in diesem Zusammenhang Strukturen, die der Oberfläche eines Golfballs ähnlich sind [21]. BONITZ ET AL. überlagern zwei sinusförmige Strukturen mit Amplituden im Bereich 0,1 µm bis 10 µm [22]. Die Strukturen sind so ausgerichtet, dass die Wellenbäuche und -täler konzentrische Kreise um die optische Achse bilden. KIESEL verwendet regelmäßige Strukturen der Größe 3 µm bis 30 µm und Strukturen, die durch Sand- oder Kugelstrahlen hervorgerufen werden [23]. Von KIESEL ET AL. werden Strukturen erläutert, deren Amplitude z durch Kosinus-Funktionen der Form $z(u) = A \cdot \cos^{\varepsilon}(u)$ gegeben sind [24]. Die Streuwirkung wird durch die Parameter A und ε sowie Überlagerung mehrerer Funktionen eingestellt. Mit diesen Funktionen soll eine exakte Einstellung des Schärfegrades möglich sein.

Wie für Patente üblich, werden die beanspruchten Produkteigenschaften erläutert. Es wird nicht erklärt, wie die Parameter der Eigenschaften für ein konkretes Produkt bestimmt werden.

1.3 STRUKTUR DER ARBEIT

In der Arbeit wird eine neue Prozesskette vorgestellt. Die Kapitel beschäftigen sich mit den einzelnen Schritten zur Herstellung mikrostrukturierter Linsen. In jedem Kapitel wird das notwendige theoretische

Hintergrundwissen erläutert und die Ergebnisse bzw. umgesetzten Verbesserungen dargestellt.

In Kapitel 2 werden die mathematischen Grundlagen beschrieben, mit denen im CAD Flächen dargestellt werden. Es werden Algorithmen entwickelt, um die Flächen auf modernen Prozessoren schnell auswerten zu können. Die Entwicklung von Scheinwerfern umfasst lichttechnische Simulationen. Die verwendete Methodik zur Simulation wird erläutert. Das Kapitel 3 beschreibt die neue Prozesskette. Es wird erklärt, wie eine glatte Fläche strukturiert wird und mit welchen Parametern die Strukturen definiert werden. Simulationsergebnisse, die die Streuwirkung der Strukturen demonstrieren, werden präsentiert. Die verwendete Fertigungs- und Messtechnik wird beschrieben. Bei der Rückführung von Punktegittern in analytisch beschriebene Flächen existieren viele Möglichkeiten. Zunächst wird eine Übersicht über verschiedene Ansätze gegeben und anschließend das Verfahren im Detail erläutert, das in dieser Arbeit verwendet wird. In Kapitel 4 werden die Anforderungen an eine Scheinwerfer-Lichtverteilung, die für diese Arbeit relevant sind, dargestellt. An zwei Beispielen wird die Funktionsweise der neuen Prozesskette demonstriert. Um die Auslegung der Strukturen zu automatisieren wird in Kapitel 5 die Anwendung von genetischen Algorithmen untersucht. In Kapitel 6 werden die Ergebnisse zusammengefasst und weitere Verbesserungsmöglichkeiten aufgezeigt.

KAPITEL 2

RECHNERGESTÜTZTE AUSLEGUNG OPTISCHER FLÄCHEN

Bei der Entwicklung von automobilen Scheinwerfern ist die Erstellung eines virtuellen Prototypen unabdingbar [25, 26]. Um Entwicklungskosten und -zeiten zu reduzieren, muss ein Computermodell des Scheinwerfers erstellt, eine lichttechnische Simulation durchgeführt und bewertet werden, bevor ein erster physischer Prototyp gefertigt wird. Dazu wird eine Möglichkeit benötigt, die Geometrie des Scheinwerfers abzubilden, diese mit optischen Eigenschaften zu versehen und zu simulieren. Die in dieser Arbeit verwendeten Modelle und Methodiken werden im Folgenden erklärt.

2.1 RAY-TRACING

Zur Simulation von Scheinwerfer-Lichtverteilungen wird das sogenannte vorwärts Ray-Tracing verwendet [27]. Das Licht wird in Form von Strahlen modelliert und ausgehend von der Lichtquelle durch den optischen Aufbau verfolgt, bis die Strahlen eine zu beleuchtende Fläche erreichen oder den Simulationsaufbau verlassen [28]. In Abbildung 2.1 sind schematisch verschiedene Vorgänge des Ray-Tracing dargestellt. Abhängig von dem Material der Fläche, auf die ein Strahl trifft, finden unterschiedliche Beeinflussungen statt. Ein Strahl kann unter anderem

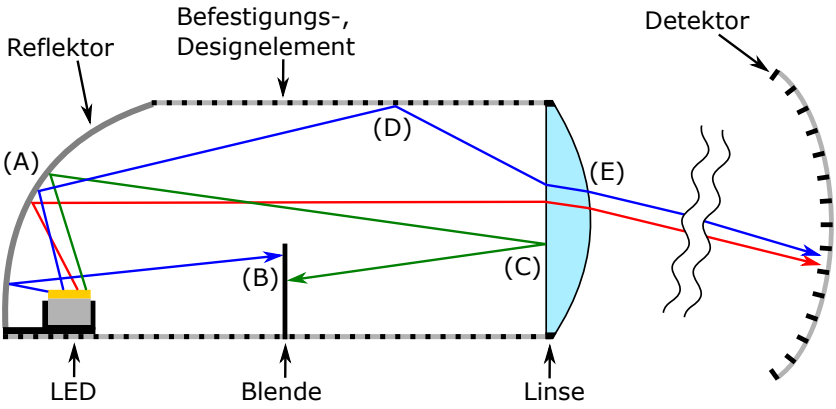


Abbildung 2.1: Schematische Darstellung des Ray-Tracing.

spiegelnd reflektiert (A), absorbiert (B), fresnel reflektiert (C), diffus reflektiert (D) oder gebrochen (E) werden [29].

2.1.1 MONTE CARLO-METHODE

Die Interaktionen zwischen Strahlen und Flächen lassen sich in zwei Kategorien einteilen. Eine Kategorie umfasst die Interaktionen, bei denen es nur einen Ergebnisstrahl gibt. Dazu zählen die spiegelnde Reflexion, die Absorption und die Brechung. Die zweite Kategorie umfasst die Interaktionen, bei denen mehrere Ergebnisstrahlen möglich sind. Bei Fresnel-Relexionen wird in Abhängigkeit der Polarisation und des Auftreffwinkels ein Teil des Lichtes reflektiert und der restliche Teil in das Medium hineingebrochen. Bei diffusen Reflexionen (z. B. nach dem Lambertschen Gesetz [30]) wird das Licht in einen Halbraum gestreut. Der in dieser Arbeit verwendete Ray-Tracer der Firma *HELLA* nutzt die Monte Carlo-Methode, um Strahl-Flächen-Interaktionen mit mehreren Ausgangsstrahlen zu handhaben [31, 32].

Trifft ein Strahl auf eine brechende Fläche, so werden entsprechend der Fresnelschen Formeln der Reflexionsgrad und der Transmissionsgrad bestimmt. Der Strahl wird mit der Wahrscheinlichkeit, die dem Reflexionsgrad entspricht, reflektiert und mit der Wahrscheinlichkeit, die dem Transmissionsgrad entspricht, gebrochen. Bei diffusen Reflexionen, die einem Lambert-Strahler entsprechen, wird die ausgehende Richtung des Strahls aus einer zweidimensionalen Kosinus-Wahrscheinlichkeitsverteilung gezogen.

Ein Nachteil der Monte Carlo-Methode ist das entstehende Simulationsrauschen [33]. Durch die auf Wahrscheinlichkeitsverteilungen basierenden Strahl-Flächen-Interaktionen können einige Detektorzellen zu viel oder zu wenig Licht enthalten. Die Größe des Simulationsrauschens ist näherungsweise antiproportional zur Wurzel der Strahlanzahl. Um das Simulationsrauschen zu halbieren, muss die bei der Simulation verwendete Strahlanzahl vervierfacht werden. Besonders problematisch ist das Rauschen in Bereichen geringer Lichtströme der simulierten Lichtstärkeverteilungen. Diese Bereiche werden im Vergleich zu heller ausgeleuchteten Bereichen von nur wenigen Strahlen getroffen, sodass das Rauschen besonders stark ausfällt. Daher müssen zur Simulation einer auswertbaren HDG, abhängig von dem konkreten Aufbau des Scheinwerfermoduls, mehrere hundert Millionen Strahlen simuliert werden.

2.1.2 SCHNITTPUNKTSBERECHNUNG

Bei dem Ray-Tracing müssen wiederholt die Schnittpunkte zwischen einem Strahl und einer Fläche gefunden werden. Ein Strahl ist gegeben durch [28]

$$\vec{R}(t) = \vec{R}_0 + \vec{R}_\mathrm{d} \cdot t, \tag{2.1}$$

wobei $\vec{R}_0$ den Startpunkt und $\vec{R}_\mathrm{d}$ den Richtungsvektor angibt. Um den Schnittpunkt mit einer Fäche $\vec{S}(u,v)$ zu berechnen, muss die Gleichung

$$\vec{S}(u,v) = \vec{R}(t) \tag{2.2}$$

$$\vec{S}(u,v) = \vec{R}_0 + \vec{R}_\mathrm{d} \cdot t \tag{2.3}$$

gelöst werden. Dazu wird die Funktion

$$\vec{F}(u,v,t) = \vec{S}(u,v) - \vec{R}_0 - \vec{R}_\mathrm{d} \cdot t \tag{2.4}$$

definiert. Um die Nullstelle von $\vec{F}$ zu bestimmen, wird das Newton-Raphson-Verfahren verwendet [34]. Das Verfahren berechnet ausgehend von einem Startwert $\vec{h}_0 = (u_0, v_0, t_0)$ iterativ die Nullstelle. Dazu wird die Jacobi-Matrix

$$\underline{\underline{J}}(u,v) = \begin{pmatrix} \frac{\partial}{\partial u} S_x(u,v) & \frac{\partial}{\partial v} S_x(u,v) & R_{\mathrm{d},x} \\ \frac{\partial}{\partial u} S_y(u,v) & \frac{\partial}{\partial v} S_y(u,v) & R_{\mathrm{d},y} \\ \frac{\partial}{\partial u} S_z(u,v) & \frac{\partial}{\partial v} S_z(u,v) & R_{\mathrm{d},z} \end{pmatrix} \tag{2.5}$$

aufgestellt. Durch Lösen des linearen Gleichungssystems [34]

$$\underline{\underline{J}}(\vec{h}) \cdot \Delta\vec{h} = -\vec{F}(\vec{h}) \tag{2.6}$$

werden die Korrekturen $\Delta\vec{h}$ bestimmt. Der neue Lösungsvektor ist gegeben durch [34]:

$$\vec{h}_\mathrm{neu} = \vec{h}_\mathrm{alt} + \Delta\vec{h} \tag{2.7}$$

Das Verfahren wird iteriert, bis die Konvergenz erreicht ist.

Die Suche nach Schnittpunkten durch das Newton-Raphson-Verfahren kann durch den Einsatz einer Hierarchie von Begrenzungsvolumen beschleunigt werden [33]. In Abbildung 2.2 ist eine Hierarchie dargestellt. Die Fläche wird in kleine Teilflächen unterteilt und für jede Teilfläche ein Begrenzungsvolumen berechnet. Zum Finden der Schnittpunkte

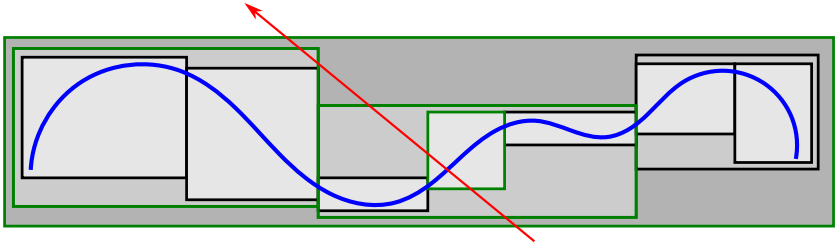


Abbildung 2.2: Schematische Darstellung einer Hierarchie von Begrenzungsrahmen (Begrenzungsvolumen) einer Kurve (blau). Die vom Strahl getroffenen Begrenzungsrahmen sind grün markiert.

wird die Hierarchie der Begrenzungsvolumen von außen nach innen durchquert und nach Schnittpunkten zwischen Strahl und Begrenzungsvolumen gesucht. Wenn ein innen liegendes Begrenzungsvolumen vom Strahl geschnitten wird, wird nach dem Schnittpunkt auf der Fläche selbst gesucht. Durch die Verwendung von Begrenzungsvolumen kann die Verwendung von numerisch aufwendigen Verfahren reduziert werden. Die Berechnung von Schnittpunkten mit Ebenen ist schneller als die Suche nach Schnittpunkten auf Freiformflächen mit, z. B., dem Newton-Raphson-Verfahren.

2.2 FLÄCHENDARSTELLUNG

In der Automobilindustrie sind Freiformflächen nicht mehr wegzudenken. In den 1960er Jahren entwickelte Pierre Bezier die sogenannten Bezier-Kurven und -Flächen, um Freiformflächen modellieren zu können [35, 36]. Nachteil der Bezier-Kurven und -Flächen ist der globale Einfluss der Kurven- bzw. Flächenparameter. Die Änderung eines Parameters ändert den Verlauf der gesamten Geometrie [37]. Um lokale

Änderungen an der Geometrie vornehmen zu können, ist eine Flächenbeschreibung durch „Basis Splines“ (B-Splines) besser geeignet [38, 39].

2.2.1 DAS KOORDINATENSYSTEM

Das in dieser Arbeit verwendete Koordinatensystem ist in Abb. 2.3 dargestellt. Die z-Achse zeigt aus Sicht eines Autofahrers in die Vor-

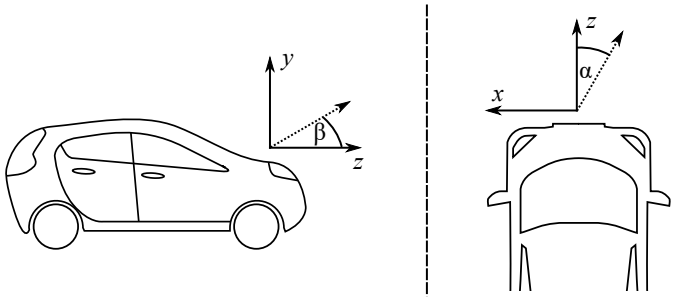


Abbildung 2.3: Ausrichtung des Koordinatensystems.

wärtsrichtung. Die y-Achse zeigt vertikal nach oben. Die x-Achse bildet mit den anderen Achsen ein rechtshändiges Koordinatensystem und zeigt damit aus Fahrersicht von rechts nach links. In einigen Kapiteln wird die Lichtrichtung in Winkeln angegeben. Beide Winkel werden bezüglich der z-Achse angegeben. Der Winkel α verläuft horizontal. Positive Werte zeigen nach rechts. Der Winkel β verläuft vertikal. Positive Werte zeigen nach oben.

2.2.2 BASIS SPLINES

Die Notation zur Beschreibung der Splines ist [40] entnommen. Sei $U = \{u_0, \ldots, u_r\}$ der sogenannte Knotenvektor, bestehend aus einer

monoton steigenden Sequenz reeller Zahlen u_i. Die u_i werden als Knoten bezeichnet. Sind die Abstände zwischen den u_i nicht äquidistant, so wird der Knotenvektor als nicht-uniform bezeichnet. Die B-Spline Basis-Funktionen vom Grad p sind definiert durch [40]:

$$N_{i,0}(u) = \begin{cases} 1 & \text{falls } u_i \leq u < u_{i+1} \\ 0 & \text{ansonsten} \end{cases} \tag{2.8}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.9}$$

Eine B-Spline-Kurve basiert auf $n+1$ Kontrollpunkte $\vec{P}_i \in \mathbb{R}^3$ und ist gegeben durch [40]:

$$\vec{C}(u) = \sum_{i=0}^{n} N_{i,p}(u) \vec{P}_i \qquad \text{mit } u \in [u_0, u_r] \tag{2.10}$$

Zwischen der Anzahl der Knoten $r+1$ und der Anzahl der Kontrollpunkte $n+1$ gilt die Beziehung $r = n + p + 1$. Zur Veranschaulichung von Gl. (2.10) sind in Abb. 2.4 zwei B-Spline-Kurven dargestellt. Die Kontrollpunkte $\vec{P}_i$ spannen ein Polygon auf, dass die B-Spline-Kurve einhüllt. Abhängig von den Knoten wird der Parameterbereich definiert, in dem die Basis-Funktionen ungleich null sind (siehe Gl. (2.9)) und dadurch bestimmt, wie stark sich die einzelnen Kontrollpunkte auf den Verlauf der Kurve auswirken. Ist ein Knoten p-mal vorhanden, so entstehen Knicke in der Kurve (siehe blaue Kurve in Abb. 2.4).

Durch Verwendung eines zweiten Knotenvektors $V = \{v_0, \ldots, v_s\}$ und einer Kontrollpunktmatrix $\vec{P}_{i,j} \in \mathbb{R}^3$ wird eine B-Spline-Fläche definiert durch [40]:

$$\vec{S}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) \vec{P}_{i,j} \tag{2.11}$$

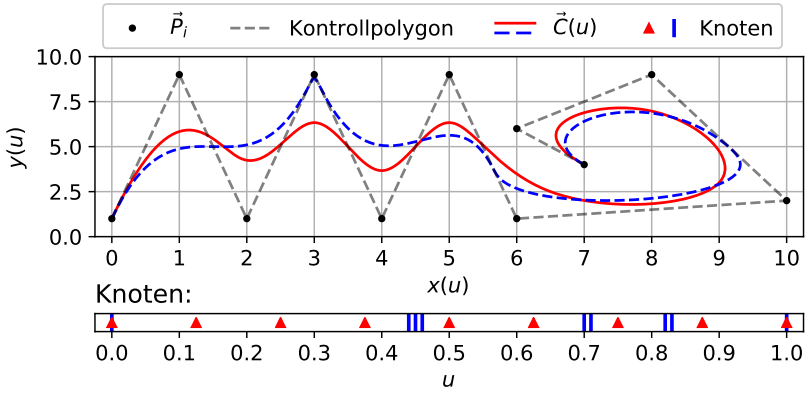


Abbildung 2.4: Vergleich von zwei B-Spline-Kurven mit den gleichen Kontrollpunkten und unterschiedlichen Knotenvektoren. Die äußeren Knoten sind vierfach in dem Knotenvektor vorhanden.

Hierbei gibt p den Grad in u-Richtung und q den Grad in v-Richtung der Fläche an. In Abb. 2.5 ist eine B-Spline-Fläche dargestellt. Im Gegensatz zur B-Spline-Kurve hat eine B-Spline-Fläche eine Kontrollpunktmatrix anstatt eines Vektors. Beide Geometriedarstellungen (Kurve und Fläche) basieren jedoch auf Knotenvektoren. Das hat zur Folge, dass z. B. eine Änderung eines Knotens in dem u-Knotenvektor einer Fläche die Fläche entlang der kompletten v-Richtung ändert.

Die Ableitung in u- und v-Richtung einer B-Spline-Fläche berechnen sich aus [40]

$$\frac{\partial^{k+l}}{\partial^k u\, \partial^l v}\vec{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}^{(k)}(u) N_{j,q}^{(l)}(v)\vec{P}_{i,j}, \tag{2.12}$$

wobei die Ableitungen der Basis-Funktionen bestimmt werden durch [40]:

$$N_{i,p}^{(k)} = p \cdot \left(\frac{N_{i,p-1}^{(k-1)}}{u_{i+p} - u_i} - \frac{N_{i+1,p-1}^{(k-1)}}{u_{i+p+1} - u_{i+1}} \right) \tag{2.13}$$

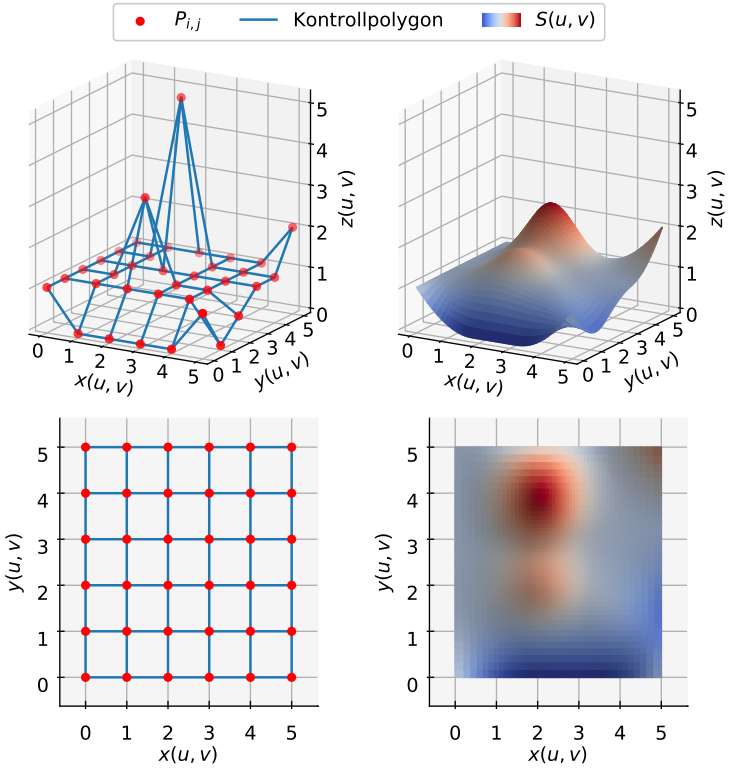


Abbildung 2.5: Darstellung einer B-Spline-Fläche (rechts) und die dazugehörigen Kontrollpunkte und das Kontrollpolygon (links). Die Knotenvektoren in u- und v-Richtung sind [0, 0, 0, 0, ⅓, ⅔ 1, 1, 1, 1].

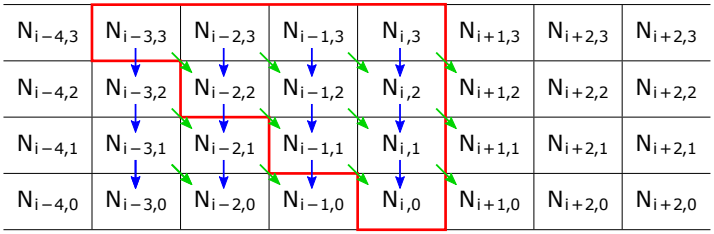


Abbildung 2.6: Grafische Darstellung der Abhängigkeit gegeben durch Gl. (2.8) und Gl. (2.9) für kubische $(p = 3)$ Basis-Funktionen.

2.2.3 KONVENTIONELLE BERECHNUNG DER BASIS-FUNKTIONEN

Konventionelle Berechnung der Basis-Funktionen Zur effizienten Berechnung einer B-Spline-Kurve eignet sich die direkte Auswertung von Gl. (2.10) nicht, da für ein bestimmtes u ein Großteil der Basis-Funktionen null ist. Nach Gl. (2.8) und (2.9) existieren höchstens $(p + 1)$ Basis-Funktionen, die ungleich null sind. Für ein $u \in [u_i, u_{i+1}[$ sind das die Basis-Funktionen $N_{i-p,p}$ bis $N_{i,p}$. Diese Eigenschaft ist in Abb. 2.6 für kubische Basis-Funktionen $(p = 3)$ veranschaulicht. Die grünen und blauen Pfeile deuten die rekursive Abhängigkeit der einzelnen Summanden von Gl. (2.9) an. Die Basis-Funktionen außerhalb der roten Umrandung sind null, dass heißt, es müssen nur die Basis-Funktionen innerhalb berechnet werden. Bei der Bestimmung aller nicht-verschwindenden Basis-Funktionen ergeben sich durch Gl. (2.9) redundante Berechnungen. Um die redundanten Berechnungen zu vermeiden werden die Notationen [41]

$$\text{left}[j] = u - u_{i+1-j} \quad \text{und} \quad \text{right}[j] = u_{i+j} - u \tag{2.14}$$

eingeführt. Damit wird der Algorithmus 2.1 definiert, der in einigen quelloffenen Programmbibliotheken verwendet wird [42–48]. Zur Berechnung der Basis-Funktionen einschließlich deren Ableitungen kann

der Algorithmus 2.2 verwendet werden. Dieser vermeidet ebenfalls redundante Berechnungen und wird von den quelloffenen Programmbibliotheken [44–48] verwendet.

Programmauflistung 2.1: Berechnung der nicht-verschwindenden Basis-Funktionen [40, 41]

```
1  procedure BasisFuns(i, u, p, U)
2    N[0] ← 1
3    for j ← 1, p
4      left[j] ← u − U[i + 1 − j]
5      right[j] ← U[i + j] − u
6      saved ← 0
7      for r ← 0, j − 1
8        temp ← N[r] / (right[r + 1] + left[j − r])
9        N[r] ← saved + right[r + 1] · temp
10       saved ← left[j − r] · temp
11     end for
12     N[j] ← saved
13   end for
14   return N
15 end procedure
```

Programmauflistung 2.2: Berechnung der nicht-verschwindenden Basis-Funktionen und deren n-fachen Ableitungen [40]

```
1  procedure DersBasisFuns(i, u, p, n, U)
2    ndu[0][0] ← 1
3    for j ← 1, p
4      left[j] ← u − U[i + 1 − j]
5      right[j] ← U[i + j] − u
6      saved ← 0
```

```
7       for r ← 0, j − 1
8         ndu[j][r] ← right[r + 1] + left[j − r]
9         temp ← ndu[r][j − 1] / ndu[j][r]
10        ndu[r][j] ← saved + right[r + 1] · temp
11        saved ← left[j − r] · temp
12      end for
13      ndu[j][j] ← saved
14    end for
15    for j ← 0, p // Load basis functions
16      ders[0][j] ← ndu[j][p]
17    end for
18    for r ← 0, p
19      s1 ← 0
20      s2 ← 1
21      a[0][0] ← 1
22      for k ← 1, n // Loop to compute nth derivative
23        d ← 0
24        rk ← r − k
25        pk ← p − k
26        if r ≥ k
27          a[s2][0] ← a[s1][0] / ndu[pk + 1][rk]
28          d ← a[s2][0] · ndu[rk][pk]
29        end if
30        if rk ≥ −1
31          j1 ← 1
32        else
33          j1 ← −rk
34        end if
35        if r − 1 ≤ pk
36          j2 ← k − 1
```

```
37          else
38            j2 ← p − r
39          end if
40          for j ← j1, j2
41            a[s2][j] ← (a[s1][j] − a[s1][j − 1]) / ndu[pk + 1][rk + j]
42            d ← d + a[s2][j] · ndu[rk + j][pk]
43          end for
44          if r ≤ pk
45            a[s2][k] ← −a[s1][k − 1] / ndu[pk + 1][r]
46            d ← d + a[s2][k] · ndu[r][pk]
47          end if
48          ders[k][r] ← d
49          j ← s1; s1 ← s2; s2 ← j
50        end for
51      end for
52      r ← p
53      for k ← k, n
54        for j ← 0, p
55          ders[k][j] ← r · ders[k][j]
56        end for
57        r ← r · (p − k)
58      end for
59      return ders
60  end procedure
```

2.2.4 BERECHNUNG DURCH AVX2-INSTRUKTIONEN

Um eine schnelle Simulation der lichttechnischen Auswirkung von optischen Flächen zu ermöglichen, müssen die B-Spline-Flächen und deren

erste Ableitungen schnell berechnet werden (vgl. Kap. 2.1). Die Algorithmen 2.1 und 2.2 berechnen ausschließlich die nicht-verschwindenden Basis-Funktionen (roter Bereich in Abb. 2.6), jedoch sequentiell. Seit 1966 können bestimmte Prozessoren die gleichen Operationen auf mehreren Daten durch sogenannte „Single Instruction, Multiple Data"-Anweisungen (kurz SIMD) gleichzeitig ausführen [49]. Durch Einführung von „Advanced Vector Extensions"-Instruktionen können Operationen auf vier Gleitkommazahlen mit doppelter Genauigkeit angewendet werden [50]. Diese Anweisungen können die Berechnung der Basis-Funktionen beschleunigen. Anstatt die Basis-Funktionen sequentiell zu berechnen, können mit SIMD-Instruktionen die Basis-Funktionen innerhalb einer Zeile aus Abb.2.6 gleichzeitig berechnet werden. Dadurch werden auch Basis-Funktionen berechnet, die null sind $(N_{i-3,2}\, N_{i-3,1},\ N_{i-2,1})$, wobei sich aufgrund der Gleichzeitigkeit dennoch eine Beschleunigung ergibt.

In dieser Arbeit liegt der Fokus auf kubische B-Spline-Flächen. Kubische Flächen sind krümmungsstetig. Damit lassen sich tangentenstetige Lichtstärkeverteilungen erzeugen. Es wird ein Algorithmus entwickelt, um Gl. (2.11) und (2.12) durch AVX-Instruktionen berechnen zu können. Dazu werden die Notationen [51]

$$c_{1,i,p} = \frac{1}{u_{i+p} - u_i} \tag{2.15}$$

$$c_{2,i,p} = -\frac{1}{u_{i+p+1} - u_{i+1}} \tag{2.16}$$

$$c_{3,i,p} = -u_i \cdot c_{1,i,p} \tag{2.17}$$

$$c_{4,i,p} = -u_{i+p+1} \cdot c_{2,i,p} \tag{2.18}$$

eingeführt und Gl. (2.9) umgeformt zu:

$$N_{i,p}(u) = \left(c_{1,i,p} \cdot u + c_{3,i,p}\right) N_{i,p-1} + \left(c_{2,i,p} \cdot u + c_{4,i,p}\right) N_{i+1,p} \tag{2.19}$$

Das Symbol $\odot$ gibt die elementweise Multiplikation an. Die Berechnung der Basis-Funktionen $N_{i-3,p}$ bis $N_{i,p}$ von Grad $p \leq 3$ ist gegeben durch:

$$\begin{bmatrix} N_{i,p} \\ N_{i-1,p} \\ N_{i-2,p} \\ N_{i-3,p} \end{bmatrix} = \underbrace{\underbrace{\left(\begin{bmatrix} c_{1,i,p} \\ c_{1,i-1,p} \\ c_{1,i-2,p} \\ c_{1,i-3,p} \end{bmatrix} \odot \begin{bmatrix} u \\ u \\ u \\ u \end{bmatrix} + \begin{bmatrix} c_{3,i,p} \\ c_{3,i-1,p} \\ c_{3,i-2,p} \\ c_{3,i-3,p} \end{bmatrix} \right)}_{\text{simd\_fmadd}} \odot \begin{bmatrix} N_{i,p-1} \\ N_{i-1,p-1} \\ N_{i-2,p-1} \\ N_{i-3,p-1} \end{bmatrix}}_{\text{simd\_mul}}$$

$$\underbrace{+}_{\text{simd\_add}} \underbrace{\underbrace{\left(\begin{bmatrix} c_{2,i,p} \\ c_{2,i-1,p} \\ c_{2,i-2,p} \\ c_{2,i-3,p} \end{bmatrix} \odot \begin{bmatrix} u \\ u \\ u \\ u \end{bmatrix} + \begin{bmatrix} c_{4,i,p} \\ c_{4,i-1,p} \\ c_{4,i-2,p} \\ c_{4,i-3,p} \end{bmatrix} \right)}_{\text{simd\_fmadd}} \odot \begin{bmatrix} N_{i+1,p-1} \\ N_{i,p-1} \\ N_{i-1,p-1} \\ N_{i-2,p-1} \end{bmatrix}}_{\text{simd\_mul}} \tag{2.20}$$

Durch Gl. (2.20) wird eine Zeile aus Abb. 2.6 basierend auf der darunter liegenden Zeile berechnet, wobei die Berechnung der einzelnen Elemente jeder Zeile durch SIMD-Instruktionen stattfindet. Die Anweisung *simd_add* steht für die gleichzeitige Addition, die Anweisung *simd_mul* für die gleichzeitige elementweise Multiplikation von zwei Vektoren mit je vier Gleitkommazahlen [50, 52]. Die Anweisung *simd_fmadd* („fused multiply–add“) bezeichnet eine vereinte Multiplikation-Addition [53]. Diese hat den Vorteil, dass, im Gegensatz zur sequentiellen Multiplikation und anschließenden Addition, nur eine Rundung stattfindet und die Berechnung schneller ist [54]. Algorithmus 2.3 verwendet SIMD-Anweisungen zur Berechnung der nicht verschwindenden Basis-Funktionen vom Grad drei.

Programmauflistung 2.3: Berechnung der nicht-verschwindenden kubischen Basis-Funktionen durch SIMD-Anweisungen

```
1  procedure getBasisFuns(i, u, c11, c21, c31, c41,
2                                  c12, c22, c32, c42, c13, c23, c33, c43)
3    s ← i − 3
4    u_AVX ← {u, u, u, u}
5
6    // Contains {N_{i−3,0}, N_{i−2,0}, N_{i−1,0}, N_{i,0}}
7    N_AVX ← {0, 0, 0, 1}
8
9    // Compute {N_{i−3,1}, N_{i−2,1}, N_{i−1,1}, N_{i,1}}
10   N_AVX ←getNextDegreeBasisFuns(s, c11, c21, c31,
11                                          c41, u_AVX, N_AVX)
12   // Compute {N_{i−3,2}, N_{i−2,2}, N_{i−1,2}, N_{i,2}}
13   N_AVX ←getNextDegreeBasisFuns(s, c12, c22, c32,
14                                          c42, u_AVX, N_AVX)
15   // Compute {N_{i−3,3}, N_{i−2,3}, N_{i−1,3}, N_{i,3}}
16   N_AVX ←getNextDegreeBasisFuns(s, c13, c23, c33,
17                                          c43, u_AVX, N_AVX)
18   return N_AVX
19 end procedure
20 procedure getNextDegreeBasisFuns(s, c1, c2, c3
21                                          c4, u_AVX, N_AVX)
22   // Compute {N_{i−2,k}, N_{i−1,k}, N_{i,k}, N_{i−3,k}} by cyclic
         permutation
23   N_{2,AVX} ← permuteLeft(N_AVX)
24   // Compute {N_{i−2,k}, N_{i−1,k}, N_{i,k}, 0}}
25   N_{2,AVX} ← setLastElementZero(N_{2,AVX})
26
27   // Load {c1[s], c1[s + 1], c1[s + 2], c1[s + 3]}
```

```
28    c_{1,AVX} ← loadAVX(c_1, s)
29
30    // Load {c_2[s], c_2[s+1], c_2[s+2], c_2[s+3]}
31    c_{2,AVX} ← loadAVX(c_2, s)
32
33    // Load {c_3[s], c_3[s+1], c_3[s+2], c_3[s+3]}
34    c_{3,AVX} ← loadAVX(c_3, s)
35
36    // Load {c_4[s], c_4[s+1], c_4[s+2], c_4[s+3]}
37    c_{4,AVX} ← loadAVX(c_4, s)
38
39    // Compute c_1 · u + c_3
40    c_Left ← simd_fmadd(u_AVX, c_{1,AVX}, c_{3,AVX})
41    // Compute c_2 · u + c_4
42    c_Right ← simd_fmadd(u_AVX, c_{2,AVX}, c_{4,AVX})
43
44     N_AVX ← simd_add(simd_mul(c_Left, N_AVX),
45                      simd_mul(c_Right, N_{2,AVX}))
46    return N_AVX
47 end procedure
```

Zur Berechnung der Ableitung wird mit Gl. (2.15) bis (2.18) die Gl. (2.13) umformuliert zu:

$$\begin{bmatrix} N_{i,p}^{(1)} \\ N_{i-1,p}^{(1)} \\ N_{i-2,p}^{(1)} \\ N_{i-3,p}^{(1)} \end{bmatrix} = \underbrace{p \cdot \underbrace{\left(\underbrace{\begin{bmatrix} c_{1,i,p} \\ c_{1,i-1,p} \\ c_{1,i-2,p} \\ c_{1,i-3,p} \end{bmatrix} \odot \begin{bmatrix} N_{i,p-1} \\ N_{i-1,p-1} \\ N_{i-2,p-1} \\ N_{i-3,p-1} \end{bmatrix}}_{\text{simd\_mul}} + \underbrace{\begin{bmatrix} c_{2,i,p} \\ c_{2,i-1,p} \\ c_{2,i-2,p} \\ c_{2,i-3,p} \end{bmatrix} \odot \begin{bmatrix} N_{i+1,p-1} \\ N_{i,p-1} \\ N_{i-1,p-1} \\ N_{i-2,p-1} \end{bmatrix}}_{\text{simd\_mul}} \right)}_{\text{simd\_add}}}_{\text{simd\_mul}} \tag{2.21}$$

In Algorithmus 2.4 ist die Berechnung durch SIMD-Anweisungen von Gl. (2.21) für kubische Basis-Funktionen dargestellt.

Programmauflistung 2.4: Berechnung der nicht-verschwindenden kubischen Basis-Funktionen und deren ersten Ableitungen durch SIMD-Anweisungen

```
1  procedure getBasisDerivat(i, u, c11, c21, c31, c41,
2                                   c12, c22, c32, c42, c13, c23, c33, c43)
3    s ← i − 3
4    uAVX ← {u, u, u, u}
5
6    // Contains {Ni−3,0, Ni−2,0, Ni−1,0, Ni,0}
7    NAVX ← {0, 0, 0, 1}
8
9    // Compute {Ni−3,1, Ni−2,1, Ni−1,1, Ni,1}
10   NAVX ←getNextDegreeBasisFuns(s, c11, c21, c31, c41,
11                                          uAVX, NAVX)
12
13   // Compute {Ni−3,2, Ni−2,2, Ni−1,2, Ni,2}
14   NAVX ←getNextDegreeBasisFuns(s, c12, c22, c32, c42,
```

```
15                                          u_AVX, N_AVX)
16
17      // Compute {N_{i-3,3}, N_{i-2,3}, N_{i-1,3}, N_{i,3}}
18      N_AVX, N^(1)_AVX ← getNextDegreeBasisFunsDerivative(
            s, c_13, c_23, c_33, c_43, u_AVX, N_AVX)
19
20  return N_AVX, N^(1)_AVX
21  end procedure
22
23  procedure getNextDegreeBasisFunsDerivative(
        s, c_1, c_2, c_3, c_4, u_AVX, N_AVX)
24      // Compute {N_{i-2,k}, N_{i-1,k}, N_{i,k}, N_{i-3,k}} by cyclic
            permutation
25      N_{2,AVX} ← permuteLeft(N_AVX)
26      // Compute {N_{i-2,k}, N_{i-1,k}, N_{i,k}, 0}
27      N_{2,AVX} ← setLastElementZero(N_{2,AVX})
28
29      // Load {c_1[s], c_1[s+1], c_1[s+2], c_1[s+3]}
30      c_{1,AVX} ← loadAVX(c_1, s)
31
32      // Load {c_2[s], c_2[s+1], c_2[s+2], c_2[s+3]}
33      c_{2,AVX} ← loadAVX(c_2, s)
34
35      // Load {c_3[s], c_3[s+1], c_3[s+2], c_3[s+3]}
36      c_{3,AVX} ← loadAVX(c_3, s)
37
38      // Load {c_4[s], c_4[s+1], c_4[s+2], c_4[s+3]}
39      c_{4,AVX} ← loadAVX(c_4, s)
40
41      // Compute c_1 · u + c_3
```

```
42    c_Left ← simd_fmadd(u_AVX, c_1,AVX, c_3,AVX)
43    // Compute c_2 · u + c_4
44    c_Right ← simd_fmadd(u_AVX, c_2,AVX, c_4,AVX)
45
46    // Compute derivative
47    N^(1)_AVX ← simd_add(simd_mul(c_1,AVX, N_AVX) ,
48                         simd_mul(c_2,AVX, N_2,AVX))
49    N^(1)_AVX ← simd_mul(N^(1)_AVX, {3,3,3,3})
50
51    N_AVX ← simd_add(simd_mul(c_Left, N_AVX),
52                     simd_mul(c_Right, N_2,AVX))
53    return N_AVX, N^(1)_AVX
54 end procedure
```

Um eine B-Spline-Fläche und die ersten Ableitungen in u- und v-Richtung bestimmen zu können, müssen die Gleichungen (2.11) und (2.12) ausgewertet werden. Beide Gleichungen lassen sich als Vektor-Matrix-Vektor-Produkt umschreiben [40]:

$$S_c(u,v) = \vec{N}(u) \cdot \underline{\underline{P_c}} \cdot \vec{N}(v) \tag{2.22}$$

Hierbei steht $c \in \{x, y, z\}$ für eine der drei Raumrichtungen und $\underline{\underline{P_c}}$ für die jeweilige Kontrollpunktmatrix. Die Vektoren sind gegeben durch:

$$\vec{N}(u) = \begin{pmatrix} N_{i-3,3}(u) \\ N_{i-2,3}(u) \\ N_{i-1,3}(u) \\ N_{i,3}(u) \end{pmatrix} \quad \vec{N}(v) = \begin{pmatrix} N_{j-3,3}(v) \\ N_{j-2,3}(v) \\ N_{j-1,3}(v) \\ N_{j,3}(v) \end{pmatrix} \tag{2.23}$$

Bei der Auswertung einer Fläche muss Gl. (2.22) für jede Raumkoordinate berechnet werden. Die Umsetzung von Gl. (2.22) durch SIMD-Anweisungen ist schwierig, da es keine SIMD-Anweisung gibt, die alle

vier Zahlen innerhalb eines AVX-Registers aufsummiert. Diese Art der Addition wird bei dem Produkt $\underline{\underline{P_c}} \cdot \vec{N}(v)$ benötigt. Nach der Multiplikation des Vektors mit einer Zeile der Matrix muss die Addition ausgeführt werden. Schneller ist die Berechnung, in dem Gl. (2.22) umgeschrieben wird zu:

$$S_c(u,v) = \left\langle \vec{N}(u) \cdot \vec{N}^T(v),\ \underline{\underline{P_c}} \right\rangle_{\mathrm{F}} \tag{2.24}$$

Hierbei bezeichnet $\langle , \rangle_{\mathrm{F}}$ das Frobenius-Skalarprodukt [55]:

$$\left\langle \underline{\underline{A}}, \underline{\underline{B}} \right\rangle_{\mathrm{F}} = \sum_i \sum_k a_{ik} b_{ik} \tag{2.25}$$

Nach Gl. (2.24) wird eine Matrix

$$\begin{aligned} \underline{\underline{NN}} &= \vec{N}(u) \cdot \vec{N}^T(v) \\ &= \begin{pmatrix} N_{i-3,3} \cdot N_{j-3,3} & N_{i-3,3} \cdot N_{j-2,3} & N_{i-3,3} \cdot N_{j-1,3} & N_{i-3,3} \cdot N_{j,3} \\ N_{i-2,3} \cdot N_{j-3,3} & N_{i-2,3} \cdot N_{j-2,3} & N_{i-2,3} \cdot N_{j-1,3} & N_{i-2,3} \cdot N_{j,3} \\ N_{i-1,3} \cdot N_{j-3,3} & N_{i-1,3} \cdot N_{j-2,3} & N_{i-1,3} \cdot N_{j-1,3} & N_{i-1,3} \cdot N_{j,3} \\ N_{i,3} \cdot N_{j-3,3} & N_{i,3} \cdot N_{j-2,3} & N_{i,3} \cdot N_{j-1,3} & N_{i,3} \cdot N_{j,3} \end{pmatrix} \end{aligned} \tag{2.26}$$

berechnet. Die Matrix $\underline{\underline{NN}}$ wird für die Berechnung der Fläche in jeder Raumrichtung benötigt. Jede Zeile der Matrix wird in einem AVX-Register gespeichert. Nach der elementweisen Multiplikation mit jeweils einer Kontrollpunktmatrix $\underline{\underline{P_c}}$ ergibt sich die Matrix:

$$\underline{\underline{NN}} \odot \underline{\underline{P_c}} = \\ \begin{pmatrix} NN_{00} \cdot P_{c,i-3,j-3} & NN_{01} \cdot P_{c,i-3,j-2} & NN_{02} \cdot P_{c,i-3,j-1} & NN_{03} \cdot P_{c,i-3,j} \\ NN_{10} \cdot P_{c,i-2,j-3} & NN_{11} \cdot P_{c,i-2,j-2} & NN_{12} \cdot P_{c,i-2,j-1} & NN_{13} \cdot P_{c,i-2,j} \\ NN_{20} \cdot P_{c,i-1,j-3} & NN_{21} \cdot P_{c,i-1,j-2} & NN_{22} \cdot P_{c,i-1,j-1} & NN_{23} \cdot P_{c,i-1,j} \\ NN_{30} \cdot P_{c,i,j-3} & NN_{31} \cdot P_{c,i,j-2} & NN_{32} \cdot P_{c,i,j-1} & NN_{33} \cdot P_{c,i,j} \end{pmatrix} \tag{2.27}$$

Die Multiplikationen innerhalb einer Zeile werden parallel durch SIMD-Anweisungen ausgeführt. Nach der Multiplikation ist jede Zeile aus Matrix (2.27) in einem AVX-Register gespeichert. Nach Gl. (2.24) müssen die Elemente aus (2.27) addiert werden. Dazu werden erst die vier Register, die die Matrix (2.27) enthalten, und anschließend die Einträge des verbleibenden Registers aufsummiert. Eine Pseudocode-Darstellung des Frobenius-Skalarproduktes ist in Algorithmus 2.5 gegeben.

Programmauflistung 2.5: Berechnung des Frobenius-Skalarproduktes durch SIMD-Anweisungen.

```
1  procedure Frobenius(N_{u,AVX}, N_{v,AVX}, P_c=)
2    // N_{u,AVX} contains
         {N_{i-3,3}(u), N_{i-2,3}(u), N_{i-1,3}(u), N_{i,3}(u)}
3    // N_{v,AVX} contains
         {N_{j-3,3}(v), N_{j-2,3}(v), N_{j-1,3}(v), N_{j,3}(v)}
4
5    // Load elements of N_{u,AVX} fourfold in an AVX
         register each
6    N_{i-3,AVX} ← {N_{i-3,3}(u), N_{i-3,3}(u), N_{i-3,3}(u), N_{i-3,3}(u)}
7    N_{i-2,AVX} ← {N_{i-2,3}(u), N_{i-2,3}(u), N_{i-2,3}(u), N_{i-2,3}(u)}
8    N_{i-1,AVX} ← {N_{i-1,3}(u), N_{i-1,3}(u), N_{i-1,3}(u), N_{i-1,3}(u)}
9    N_{i,AVX}   ← {N_{i,3}(u),   N_{i,3}(u),   N_{i,3}(u),   N_{3,3}(u)}
10
11   // Mulitply with N_{v,AVX}, to get NN=
12   N_{i-3,AVX} ← simd_mul(N_{i-3,AVX}, N_{v,AVX})
13   N_{i-2,AVX} ← simd_mul(N_{i-2,AVX}, N_{v,AVX})
14   N_{i-1,AVX} ← simd_mul(N_{i-1AVX}, N_{v,AVX})
15   N_{i,AVX}   ← simd_mul(N_{i,AVX}, N_{v,AVX})
16
17   // Multiply with P_c=
```

```
18   NP_{i-3,AVX} ← simd_mul(N_{i-3,AVX}, {P_{c,i-3,j-3}, P_{c,i-3,j-2},
19                                         P_{c,i-3,j-1}, P_{c,i-3,j}})
20   NP_{i-2,AVX} ← simd_mul(N_{i-3,AVX}, {P_{c,i-2,j-3}, P_{c,i-2,j-2},
21                                         P_{c,i-2,j-1}, P_{c,i-2,j}})
22   NP_{i-1,AVX} ← simd_mul(N_{i-1,AVX}, {P_{c,i-1,j-3}, P_{c,i-1,j-2},
23                                         P_{c,i-3,j-1}, P_{c,i-1,j}})
24   NP_{i,AVX}   ← simd_mul(N_{i,AVX}, {P_{c,i,j-3}, P_{c,i,j-2}, P_{c,i,j-1}, P_{c,i,j}})
25
26   // Sum up registers
27   Sum_AVX ← simd_add (simd_add (NP_{i-3,AVX}, NP_{i-2,AVX}) ,
28                       simd_add (NP_{i-1,AVX}, NP_{i,AVX}))
29   Sum_Skalar ← Sum_AVX[0] + Sum_AVX[1] + Sum_AVX[2] + Sum_AVX[3]
30   return Sum_Skalar
31 end procedure
```

Die Ableitungen der Fläche lassen sich durch Gl. (2.24) berechnen, indem einer der beiden Vektoren durch seine Ableitung ausgetauscht wird:

$$\frac{\partial}{\partial u} S_c(u,v) = \left\langle \frac{\partial}{\partial u} \vec{N}(u) \cdot \vec{N}^T(v),\ \underline{\underline{P_c}} \right\rangle_{\mathrm{F}} \tag{2.28}$$

$$\frac{\partial}{\partial v} S_c(u,v) = \left\langle \vec{N}(u) \cdot \frac{\partial}{\partial v} \vec{N}^T(v),\ \underline{\underline{P_c}} \right\rangle_{\mathrm{F}} \tag{2.29}$$

Zur Auswertung der Fläche sowie deren Ableitungen müssen für jede Raumrichtung die Gleichungen (2.24), (2.28) und (2.29) ausgewertet werden.

2.2.5 VERGLEICH ZWISCHEN KONVENTIONELLER UND AVX-BASIERENDER BERECHNUNG

Zum Testen der erreichten Verbesserung wird ein Computer mit einem *Intel Xeon E5-1650 v3* verwendet [56]. Um die Laufzeit der konventionel-

Tabelle 2.1: Vergleiche der Leistung zwischen konventioneller und SIMD-basierter Auswertung von B-Spline-Flächen. Die Angaben zu den Implementierungen (Imp.) beziehen sich auf den Anhang.

Berechnung	Imp.	Laufzeit (s)	Beschleunigung
Basis-Funktionen	6.1	15,5	3,6
	6.2	4,28	
Basis-Funktionen & erste Ableitung	6.3	42,0	7,3
	6.4	5,8	
Auswertung der Fläche	6.5	113,1	4,6
	6.6	24,4	
Auswertung der Fläche & erste Ableitungen	6.7	248,4	5,9
	6.8	42,0	

len mit den AVX-basierten Algorithmen vergleichen zu können, ist eine konkrete Implementierung notwendig. Der Quellcode der Implementierungen ist im Anhang 6 aufgeführt. Der Quellcode wird mit *Microsoft Visual Studio 2012* kompiliert und zum überprüfen der Verbesserung 10^8 mal aufgerufen. In Tabelle 2.1 sind die Ergebnisse zusammengefasst. Die größte Beschleunigung mit einem Faktor von 7,3 ergab sich bei der Berechnung der Basis-Funktionen einschließlich den ersten Ableitungen. Bei der Auswertung der Fläche einschließlich Ableitungen bleibt ungefähr eine Verbesserung von einem Faktor sechs übrig. Die Funktionen zur Auswertung der Fläche bekommen die u- und v-Parameter übergeben und müssen die dazugehörigen i- und j-Indizes der Knotenvektoren bestimmen. Bei den Berechnungen der Basis-Funktionen wurden die Indizes vorab bestimmt. Daher ergab sich nur ein Faktor sechs bei der Flächen-Auswertung.

Neben der parallelen Berechnung durch SIMD-Anweisungen basieren die Laufzeitunterschiede auf weiteren Punkten. Die Konstanten zur Berechnung der Basis-Funktionen (Gl. (2.15) bis (2.18)) werden bei der

SIMD-Variante vorab, bei der bisher verwendeten Implementierung bei jedem Funktionsaufruf neu berechnet. Die SIMD-Variante ist ausschließlich für kubische Flächen implementiert und kann nur die jeweilige Funktion und ihre erste Ableitung berechnen. Die konventionelle Variante funktioniert für beliebige Grade und beliebige Ableitungen. Das ist möglich, weil vom Grad abhängende Schleifen und Verzweigungen verwendet werden (siehe Algorithmen 2.1 und 2.2 sowie Implementierungen 6.5 und 6.7).

Einige Prozessor-Instruktionen werden vom Prozessor in Teilaufgaben unterteilt. Die Teilaufgaben von aufeinander folgenden Instruktionen können gleichzeitig ausgeführt werden, wenn für die Ausführung unterschiedliche Komponenten des Prozessors verwendet werden (sogenanntes „Pipelining“) [57]. Um den Vorteil der Pipeline nutzen zu können, müssen zu jedem Zeitpunkt die zukünftigen Instruktionen bekannt sein. Enthält das Programm konditionale Sprünge, so versucht der Prozessor zu erraten, ob ein Sprung stattfindet (sogenanntes „Branch Prediction“) [58]. Wurde falsch geraten, enthält die Pipeline ungültige Instruktionen. Die Pipeline muss geleert und mit den korrekten Instruktionen gefüllt werden, wodurch ein Zeitverlust entsteht [57]. Durch Vermeidung von Verzweigungen kann die Laufzeit eines Programmes reduziert werden [59, 60]. Daher wurde bei der Entwicklung der SIMD basierten Auswertung von B-Spline-Flächen auf Schleifen „for“ und Verzweigungen „if“ verzichtet, wie der Vergleich von Algorithmus 2.1 mit 2.3 und Algorithmus 2.2 mit 2.4 zeigt.

KAPITEL 3

PROZESSKETTE ZUR HERSTELLUNG MIKROSTRUKTURIERTER LINSEN

In diesem Kapitel wird erläutert, welche Anforderungen an Scheinwerfer-Lichtverteilungen bestehen, wie mikrostrukturierte Flächen in CAD-Programmen dargestellt, wie die Strukturen in Stahlwerkzeuge eingebracht und wie die dazu benötigten Fertigungsdaten aufbereitet werden. Es wird die Messapparatur vorgestellt, mit denen die Oberfläche der Stahlwerkzeuge flächig vermessen werden und Algorithmen vorgestellt, mit denen Flächen auf Basis der gemessenen Punktwolken bestimmt werden können. Der Inhalt dieses Kapitels stammt zu Teilen aus der Veröffentlichung [61].

3.1 ANFORDERUNG AN LICHTSTÄRKEVERTEILUNGEN

Es existieren zwei Regelungen, die in den meisten Ländern die Anforderungen an Scheinwerfern stellen. Eine Regelung wird von der „Economic Comission for Europe“ (ECE) ausgegeben, die Zweite ist die „Federal Motor Vehicle Safety Standard 108“ (FMVSS) [8, 62]. Beide Regelungen stellen Anforderungen an die LSV eines Scheinwerfers, sodass der Scheinwerfer bestimmte Lichtstärke-Werte in vorgegebenen Bereichen mindestens haben muss, bzw. nicht überschreiten darf. Es wird in beiden Regelungen die gleiche Definition für die Schärfe der HDG, auch genannt Gradient oder AK31, verwendet. Die Gradient-Kurve $G(\beta)$ wird

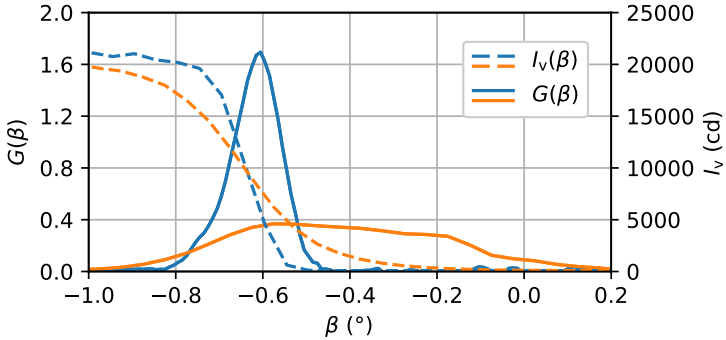


Abbildung 3.1: Darstellung des Zusammenhangs zwischen der Lichtstärke I_v und der Gradienten-Kurve $G(\beta)$ für eine scharfe (blau) und weiche (orange) HDG.

aus der Lichtstärke I_v bei einem vertikalen Winkel β berechnet [8, 62]:

$$G(\beta) = \log(I_v(\beta)) - \log(I_v(\beta + 0{,}1°)) \tag{3.1}$$

Der Maximalwert der Kurve (3.1) wird als „Gradient“ bezeichnet. Der Zusammenhang aus Gl. (3.1) ist in Abb. 3.1 visualisiert.

Beide Vorschriften geben an, wie die Werte der Lichtstärke zu messen sind. Die ECE-Regelung unterscheidet zwischen minimaler und maximaler Schärfe. Beide Werte werden bei einem horizontalen Winkel von $\alpha = -2{,}5°$ gemessen. Die minimale Schärfe ist durch eine Messung der Beleuchtungsstärke in einem Abstand von 10 m mit einem Detektor der ungefähr einen Durchmesser von 10 mm hat oder in einem Abstand von 25 m mit einem Detektor der ungefähr einen Durchmesser von 30 mm hat, zu bestimmen. Die vorgegebenen Kombinationen aus Abstand und Detektorgröße ergeben einen unterschiedlichen Raumwinkelbereich. Daher können die zwei Messkonfigurationen unterschiedliche Ergebnisse liefern. Die maximale Schärfe ist durch eine Messung der Beleuchtungsstärke in einem Abstand von 25 m mit einem Detektor der ungefähr einen Durchmesser von 30 mm hat, zu bestimmen. Beide Messungen

sollen mit einer vertikalen Schrittweite von 0,05° durchgeführt werden. Die minimale Schärfe muss über einen Wert von 0,13 und die maximale Schärfe unter einem Wert von 0,4 liegen.

Die FMVSS fordert eine Messung bei $\alpha = -2{,}5°$ oder $\alpha = 2{,}0°$. Der Detektor muss einen Durchmesser von 10 mm haben und in einem Abstand von 10 m zum Scheinwerfer platziert sein. Der Gradient darf einen Wert von 0,13 nicht unterschreiten. Eine obere Grenze ist in der Regelung nicht gegeben.

Die ECE-Regelung definiert unter anderem, dass die Lichtstärke bei $(\alpha = 0°, \beta = 0°)$, bezeichnet als „HV“, mindestens 50 cd und höchstens 625 cd betragen darf. Bei $(\alpha = 2{,}5°, \beta = 1°)$, bezeichnet als „BR“, müssen mindestens 50 cd und höchstens 1750 cd vorhanden sein.

3.2 DESIGN

Die in dieser Arbeit verwendete CAD-Software und die Ray-Tracing-Software basieren auf „Non-Uniform Rational B-Spline“-Flächen (NURBS) [40]. Daher sollte das Ergebnis der Strukturierung wieder eine NURBS-Fläche sein. Eine kubische B-Spline Fläche (vgl. Kap. 2.2.2) ist ein Spezialfall von NURBS, bei dem nicht alle Freiheitsgrade verwendet werden.

3.2.1 VERFAHREN ZUR STRUKTURIERUNG

Das Ziel der Strukturierung ist eine lokale Verkippung der Flächennormalen, um die Abbildungsqualität der Linse zu reduzieren. Dadurch wird eine scharfe HDG aufgeweicht. Die zu strukturierende Fläche wird

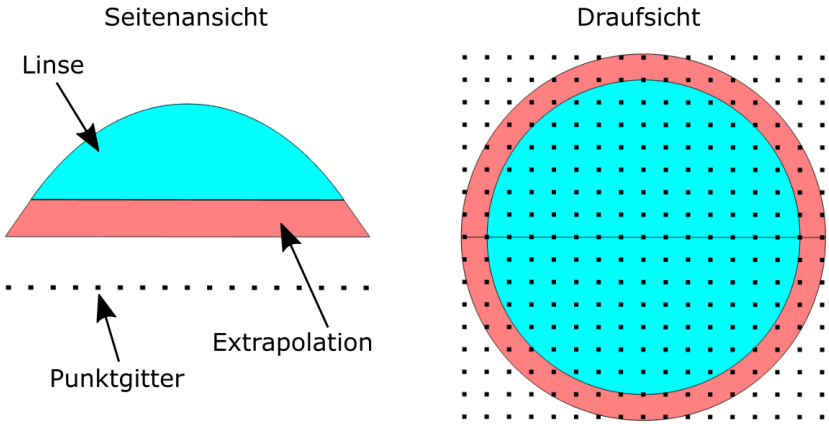


Abbildung 3.2: Schematische Darstellung des ersten Schrittes der Strukturierung: Generierung eines äquidistanten Punktgitters.

um wenige Millimeter extrapoliert (typischerweise einen Millimeter), wie in Abb. 3.2 dargestellt. Die Extrapolation dient dazu, Randeffekte bei der späteren Flächenberechnung unwirksam zu machen (siehe unten). Es wird ein äquidistantes Punktgitter in der Ebene senkrecht zur optischen Achse generiert. Der typische Punktabstand im Gitter beträgt 50 µm bis 200 µm. Das Punktgitter wird in die extrapolierte Fläche projiziert, indem die Gitterpunkte als Startpunkte und die optische Achse als Richtungsvektor von Strahlen verwendet werden. Die Schnittpunkte zwischen Strahlen und Fläche werden, wie in Kap. 2.1.2 beschrieben, berechnet. In Abb. 3.3 ist das projizierte Punktgitter schematisch dargestellt. Ausgehend von dem projizierten Punktgitter werden zwei Verfahren angewendet, um eine Strukturierung der Fläche zu erreichen.

Bei dem ersten Verfahren werden die Punkte entlang der Normalen der Fläche unterschiedlich verschoben. Die Verschiebung ist in Abb. 3.4 schematisch dargestellt. Der individuelle Versatz der Punkte wird in Abhängigkeit der HDG des Scheinwerfermoduls bestimmt. Die detaillierte

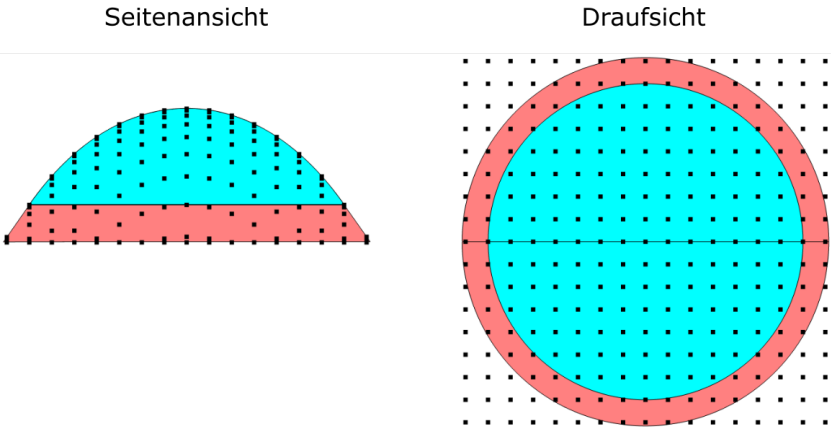


Abbildung 3.3: Schematische Darstellung des zweiten Schrittes der Strukturierung: Projektion des Punktgitters.

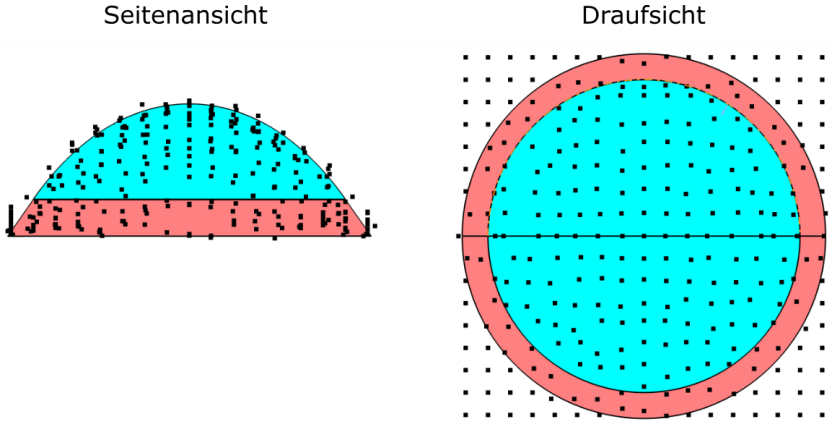


Abbildung 3.4: Schematische Darstellung des dritten Schrittes des ersten Verfahrens der Strukturierung: Verschiebung des Punktgitters entlang der Normalenvektoren.

Erklärung, wie der Versatz bestimmt wird, ist in Kap. 3.2.3 gegeben. Aus dem Punktgitter wird eine Fläche bestimmt, indem eine interpolierende kubische B-Spline-Fläche berechnet wird. Die Berechnung ist in Kap. 3.2.2 erläutert. In Abb. 3.5 ist eine kubische Fläche aufgezeigt, die das Punktgitter aus Abb. 3.4 interpoliert. Die Parameter zur Erstellung der Struktur sind für Darstellungszwecke deutlich übertrieben gewählt. Wie in Abb. 3.6 dargestellt, werden die extrapolierten Bereiche durch zuschneiden entfernt.

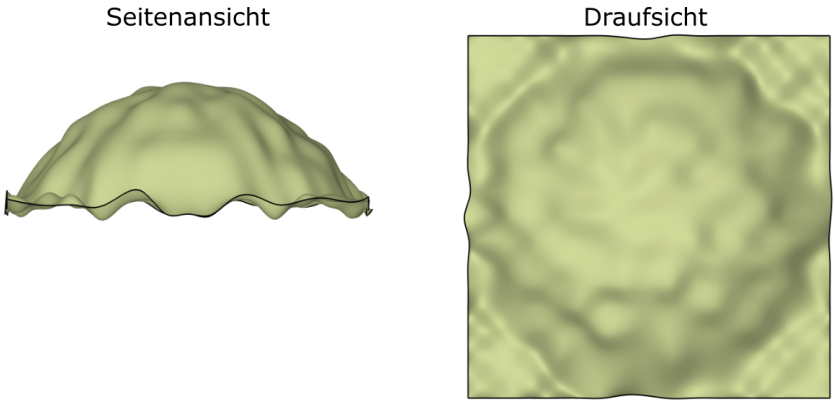


Abbildung 3.5: Schematische Darstellung des vierten Schrittes des ersten Verfahrens der Strukturierung: Berechnung einer interpolierenden Fläche.

Die zweite Variante zur Strukturierung verwendet das in die Fläche projizierte Punktgitter aus Abb. 3.3. Anstatt die Punkte wie bei der ersten Variante zu verschieben, wird eine interpolierende Fläche von dem Punktgitter berechnet. Die berechnete Fläche ist näherungsweise identisch mit der Ausgangsfläche, wird aber mit einer größeren Anzahl an Kontrollpunkten (vgl. Gl. (2.11)) beschrieben. Die Abb. 3.7 zeigt eindimensional und schematisch die Fläche und ihre Kontrollpunkte.

Anstatt die Punkte des Gitters vor der Flächenberechnung zu verschieben, werden direkt die Kontrollpunkte verschoben. Bei dem ersten Ver-

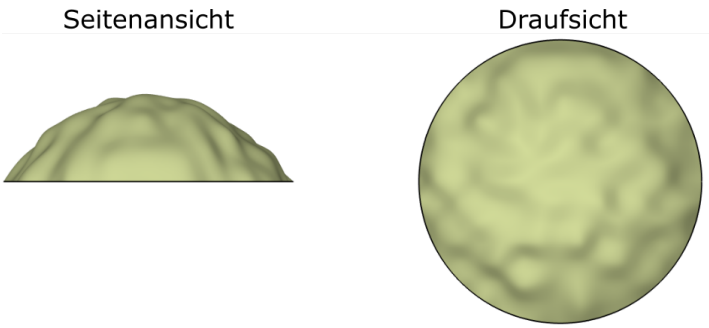


Abbildung 3.6: Schematische Darstellung des fünften Schrittes der Strukturierung: Zuschneiden der Fläche.

fahren besteht ein direkter geometrischer Zusammenhang zwischen der Verschiebung der Gitterpunkte und der Fläche. Durch die Interpolation der verschobenen Punkte, wird die Fläche immer um den gleichen Betrag verändert wie die Punkte. Bei dem zweiten Verfahren ist diese direkte Abhängigkeit nicht gegeben. Durch die Verschiebung der Kontrollpunkte wird ebenfalls eine lokale Änderung verursacht, jedoch wird die Fläche nicht um den gleichen Betrag verschoben wie der Kontrollpunkt. Die zweite Methode hat den Vorteil der schnelleren Strukturierung der Fläche, wenn wiederholt die gleiche Fläche mit unterschiedlichen Strukturen versehen werden soll. Bei der ersten Methode muss für jede Strukturierung ein lineares Gleichungssystem (siehe Kap. 3.2.2) gelöst werden, bei der zweiten Methode nur einmalig. In Kap. 5 wird die Auslegung der Strukturen durch Optimierungsalgorithmen erläutert. Bei dem Optimierungsprozess müssen fortlaufend neue Strukturen bestimmt werden. Die zweite Methode ist ungefähr zwanzig mal schneller bei der Strukturierung als die erste Methode.

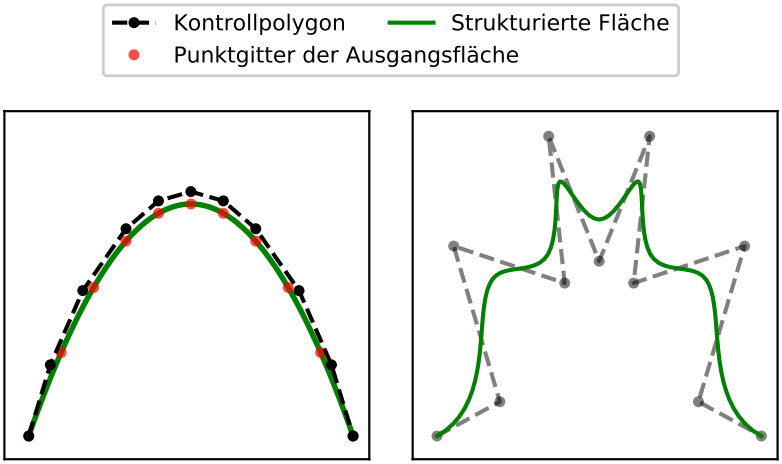


Abbildung 3.7: Schematische Darstellung des zweiten Verfahrens. Links: Interpolierende Fläche des Punktgitters. Rechts: Strukturierung der Fläche durch Verschieben der Kontrollpunkte.

3.2.2 BERECHNUNG EINER INTERPOLIERENDEN B-SPLINE-FLÄCHE

Die zu interpolierenden Punkte eines Gitters der Größe $(n+1) \times (m+1)$ sind durch $\vec{Q}_{k,l}$ gegeben. Um eine interpolierende Fläche $S(u,v)$ zu erhalten, müssen die Knotenvektoren U und V sowie die Kontrollpunkte $\vec{P}_{i,j}$ bestimmt werden. Als Knoten werden die x- und y- Koordinaten der nicht verschobenen Gitterpunkte gewählt (vgl. Abb. 3.3). Dadurch wird die Fläche so parametrisiert, dass Parameterraum und Realraum ähnlich sind, also

$$\vec{S}(x,y) \simeq \begin{pmatrix} x \\ y \\ S_z(x,y) \end{pmatrix} \tag{3.2}$$

Das hat Vorteile bei der Projektion von Punkten auf Flächen, wie in Kap. 3.5 beschrieben. Zwischen der Anzahl an Knoten $(r+1)$ und der Anzahl an Kontrollpunkten $(n+1)$ gilt der Zusammenhang [40]

$$r+1 = (n+1) + p + 1 \tag{3.3}$$

Damit die äußeren Kontrollpunkte die Endpunkte der Fläche sind, müssen die äußeren Knoten $(p+1)$-mal vorhanden sein. Dadurch resultiert, dass nicht jeder Gitterpunkt in dem Knotenvektor vorhanden sein kann. Die Koordinaten des zweiten und des vorletzten Punktes werden nicht in den Knotenvektor übernommen. Der Grad der Fläche wird in beide Richtungen auf Grad drei gesetzt ($p = q = 3$). Dadurch ergibt sich eine krümmungsstetige Fläche. Beispielsweise wird für eine Linse mit einem extrapolierten Durchmesser von 70 mm und einem Gitterabstand von 0,1 mm der Knotenvektor

$$U = \{-35;\ -35;\ -35;\ -35;\ -34,8;\ -34,7;\ -34,6;\ -34,5;\ \ldots,\\ 34,5;\ 34,6;\ 34,7;\ 34,8;\ 35;\ 35;\ 35;\ 35\}$$

verwendet. Für eine Linse mit einem extrapolierten Durchmesser von 80 mm und einem Gitterabstand von 0,2 mm wird der Knotenvektor

$$U = \{-40;\ -40;\ -40;\ -40;\ -39,6;\ -39,4;\ -39,2;\ -39,0;\ \ldots,\\ 39,0;\ 39,2;\ 39,4;\ 39,6;\ 40;\ 40;\ 40;\ 40;\}$$

verwendet. Die äußeren Kontrollpunkte sind identisch mit den Gitterpunkten:

$$\vec{P}_{k,0} = \vec{Q}_{k,0} \qquad \vec{P}_{k,m} = \vec{Q}_{k,m} \qquad \text{für } k = 0,\ldots,n \tag{3.4}$$

$$\vec{P}_{0,l} = \vec{Q}_{0,l} \qquad \vec{P}_{n,l} = \vec{Q}_{n,l} \qquad \text{für } l = 0,\ldots,m \tag{3.5}$$

Zu bestimmen sind die inneren Kontrollpunkte für $k = 1,\ldots,n-1$ und $l = 1,\ldots,m-1$. Damit die Fläche die Punkte interpoliert, müssen die

Kontrollpunkte so bestimmt werden, dass die Gleichung [40]

$$\vec{Q}_{k,l} = \vec{S}(\bar{u}_k, \bar{v}_l) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(\bar{u}_k) N_{j,q}(\bar{v}_l) \vec{P}_{i,j} \tag{3.6}$$

erfüllt wird. Die Symbole $\bar{u}_k$ und $\bar{v}_l$ geben die Parameter an, an denen die Interpolation stattfindet, bzw. an denen die Fläche die Gitterpunkte berührt. Die Parameter werden identisch mit den Knoten $u \in U$ und $v \in V$ gewählt:

$$\bar{u}_k = u_{k+p-1} \text{ für } k = 2, \ldots, n-2 \tag{3.7}$$

$$\bar{v}_l = v_{l+q-1} \text{ für } l = 2, \ldots, m-2, \tag{3.8}$$

wodurch sich ein tridiagonales Gleichungssystem (siehe unten) aufstellen lässt [40]. Die Verschiebung der Indizes um $p-1$ ergibt sich durch die $(p+1)$-fache Multiplizität der äußeren Knoten. Die Parameter für die Gitterpunkte mit $k = 1, n-1$ und $l = 1, n-1$ werden durch den Mittelwert der benachbarten Parameter bestimmt:

$$\bar{u}_k = \frac{\bar{u}_{k+1} + \bar{u}_{k-1}}{2} \text{ für } k = 1, n-1 \tag{3.9}$$

$$\bar{v}_l = \frac{\bar{u}_{l+1} + \bar{u}_{l-1}}{2} \text{ für } l = 1, m-1 \tag{3.10}$$

Um die Kontrollpunkte zu bestimmen, sodass Gl. (3.6) erfüllt wird, wird die Gleichung umgeschrieben zu [40]:

$$\vec{Q}_{k,l} = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k) \left(\sum_{j=0}^{m} N_{j,q}(\bar{v}_l) \vec{P}_{i,j} \right) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k) \vec{R}_{i,l} \tag{3.11}$$

$$\text{mit} \qquad \vec{R}_{i,l} = \sum_{j=0}^{m} N_{j,q}(\bar{v}_l) \vec{P}_{i,j} \tag{3.12}$$

Die Berechnung der interpolierenden Fläche wird durch mehrere Kurven-Interpolationen durchgeführt. Es werden $m+1$ Kurven-Interpolationen durch die Punkte $Q_{0,l}, \ldots, Q_{n,l}$ (für $l = 0, \ldots, m$) ausgeführt, um die $R_{i,l}$ zu erhalten. Weitere $n+1$ Kurven-Interpolationen

durch $R_{i,0}, \ldots, R_{i,m}$ (für $i = 0, \ldots, n$) bestimmen die $P_{i,j}$. Mit (modifiziert von [40]):

$$a_k = N_{k-1,3}\left(\bar{u}_k\right) \quad b_k = N_{k,3}\left(\bar{u}_k\right) \quad c_k = N_{k+1,3}\left(\bar{u}_k\right) \tag{3.13}$$

wird für jede Kurven-Interpolation ein Gleichungssystem aufgestellt:

$$\begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ \vdots \\ Q_{n-2} \\ Q_{n-1} \end{pmatrix} = \underline{\underline{M}} \cdot \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_{n-2} \\ R_{n-1} \\ R_n \end{pmatrix} \tag{3.14}$$

mit

$$\underline{\underline{M}} = \begin{pmatrix} a_1 & b_1 & c_1 & N_{3,3}\left(\bar{u}_1\right) & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_{n-2} & b_{n-2} & c_{n-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & N_{n-3,3}\left(\bar{u}_{n-1}\right) & a_{n-1} & b_{n-1} & c_{n-1} \end{pmatrix} \tag{3.15}$$

Hierbei wird verwendet, dass an einem inneren Knoten nur drei Basis-Funktionen ungleich null sind. Da die Kontrollpunkte am Rand mit den Gitterpunkten übereinstimmen, wird Gl. (3.14) umgeformt zu (modifi-

ziert von [40]):

$$\begin{pmatrix} Q_1 - a_1 R_0 \\ Q_2 \\ Q_3 \\ \vdots \\ Q_{n-2} \\ Q_{n-1} - c_{n-1} R_n \end{pmatrix} = \underline{\underline{M}} \cdot \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ \vdots \\ R_{n-2} \\ R_{n-1} \end{pmatrix} \tag{3.16}$$

mit

$$\underline{\underline{M}} = \begin{pmatrix} b_1 & c_1 & N_{3,3}(\bar{u}_1) & 0 & \cdots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{n-2} & b_{n-2} & c_{n-2} \\ 0 & 0 & 0 & 0 & \cdots & N_{n-3,3}(\bar{u}_{n-1}) & a_{n-1} & b_{n-1} \end{pmatrix} \tag{3.17}$$

Mit zwei Umformungsschritten werden die Elemente $N_{3,3}(\bar{u}_1)$ und $N_{n-3,3}(\bar{u}_{n-1})$ aus der ersten und letzten Zeile der Matrix entfernt. Dadurch ergibt sich ein tridiagonales Gleichungssystem, das mit dem Tridiagonalmatrix-Algorithmus gelöst werden kann [63].

3.2.3 BESTIMMUNG DER AMPLITUDEN

Die Amplituden werden aus Normalverteilungen (Gauß-Kurven) generiert[1]. Dazu wird das Punktgitter (vgl. Abb. 3.2) in Zellen eingeteilt. Die Einteilung ist für eine Zellengröße von 2x2 schematisch in Abb. 3.8 dargestellt. Jedem Punkt aus der Zelle wird eine Normalverteilung

[1] Die Idee entstammt der alten Prozesskette (vgl. Kap. 1).

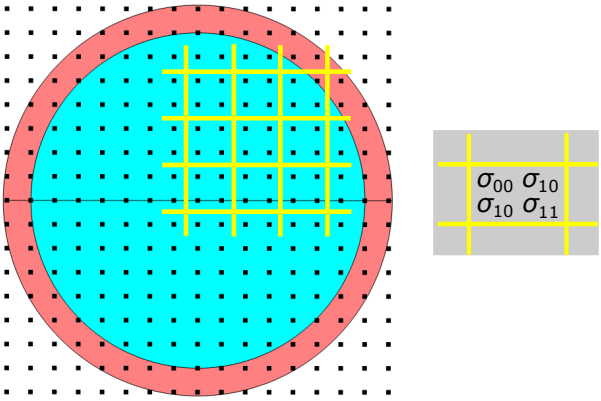
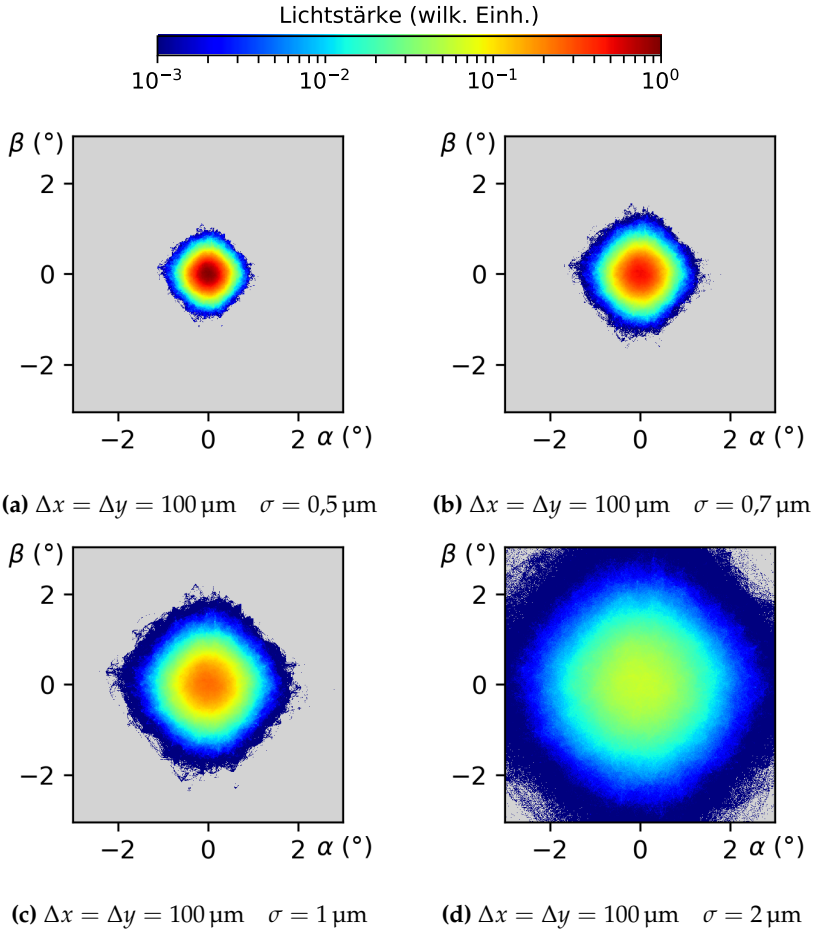


Abbildung 3.8: Einteilung der Gitterpunkte (vgl. Abb. 3.2) in Zellen der Größe 2x2 zur Zuordnung von Normalverteilungen.

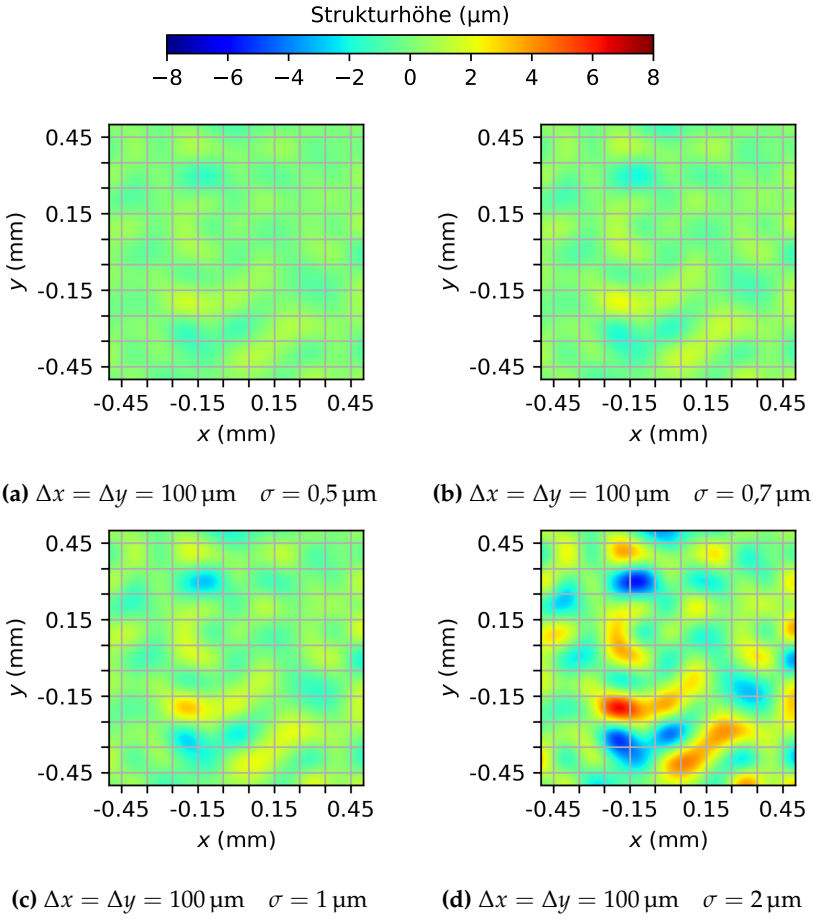
zugeordnet. Die Standardabweichungen der Normalverteilungen sind in jeder Zelle identisch. Der Versatz der Punkte (vgl. Kap. 3.2.1) wird aus den Normalverteilungen bestimmt. Die Parameter, über die das Streuverhalten der Struktur eingestellt wird, sind die Zellengröße, die Standardabweichungen σ der Normalverteilungen und der Gitterabstand. Um die Auswirkung der Parameter zu demonstrieren, wird im CAD eine planare Fläche der Größe 10 mm mal 10 mm strukturiert und in der Simulationsumgebung mit parallelem Licht bestrahlt. Die von den strukturierten Platten ausgehenden LSVen sind in den Abb. 3.9, 3.11 und 3.13 dargestellt. Die Abb. 3.10, 3.12 und 3.14 zeigen ausschnittsweise die Strukturen. Die zum Erstellen der Strukturen verwendeten Parameter sind unter den einzelnen Abbildungen angegeben. Hierbei geben Δx und Δy die Auflösung der Punktgitter an. Die Verwendung von Zellen der Größe 1x1 ergeben Streuverhalten, die Normalverteilungen ähnlich sind (vgl. Abb. 3.9). Durch die Vergrößerung auf Zellen der Größe 2x2 wird das Streuverhalten rechteckiger (vgl. Abb. 3.11). Um die Streuung in vertikaler oder horizontaler Richtung asymmetrischer zu gestalten,



(a) $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 0{,}5\,\mu\text{m}$ **(b)** $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 0{,}7\,\mu\text{m}$

(c) $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 1\,\mu\text{m}$ **(d)** $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 2\,\mu\text{m}$

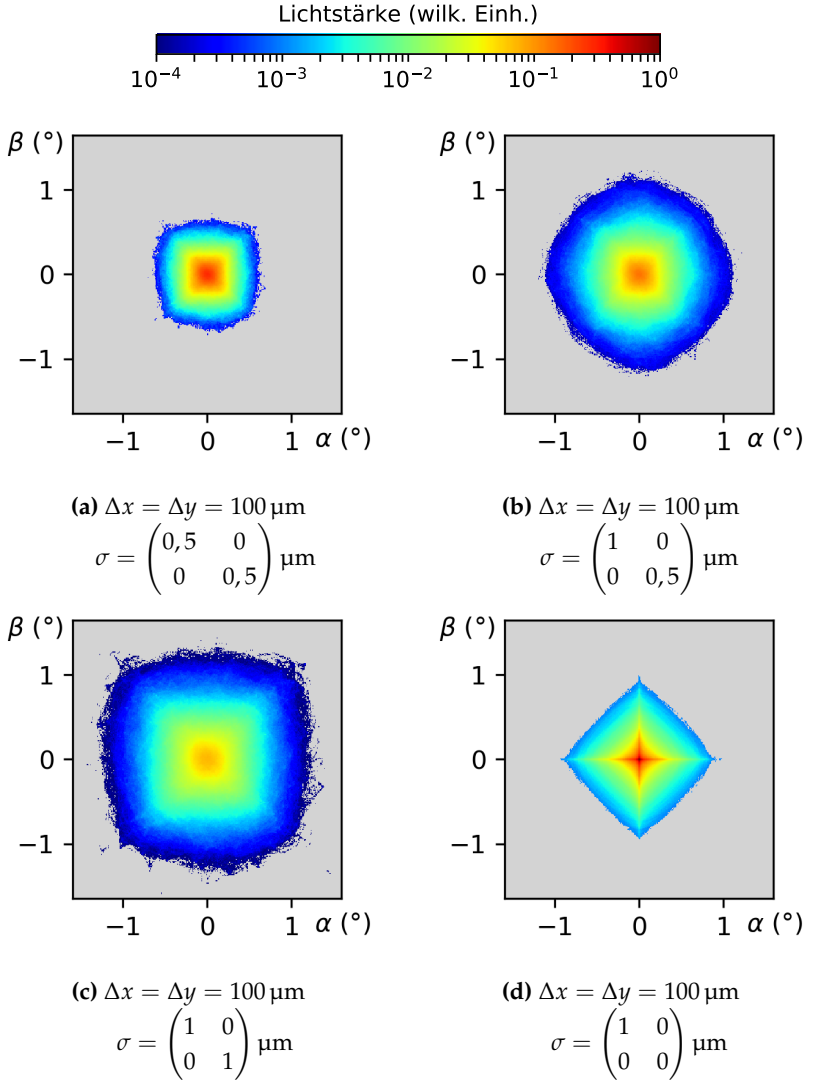
Abbildung 3.9: Streuverhalten strukturierter Platten für Zellen der Größe 1x1.



(a) $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 0{,}5\,\mu\text{m}$ **(b)** $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 0{,}7\,\mu\text{m}$

(c) $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 1\,\mu\text{m}$ **(d)** $\Delta x = \Delta y = 100\,\mu\text{m} \quad \sigma = 2\,\mu\text{m}$

Abbildung 3.10: Strukturhöhen der Platten für Zellen der Größe 1x1. Die grauen Linien kennzeichnen die Zellengrößen.



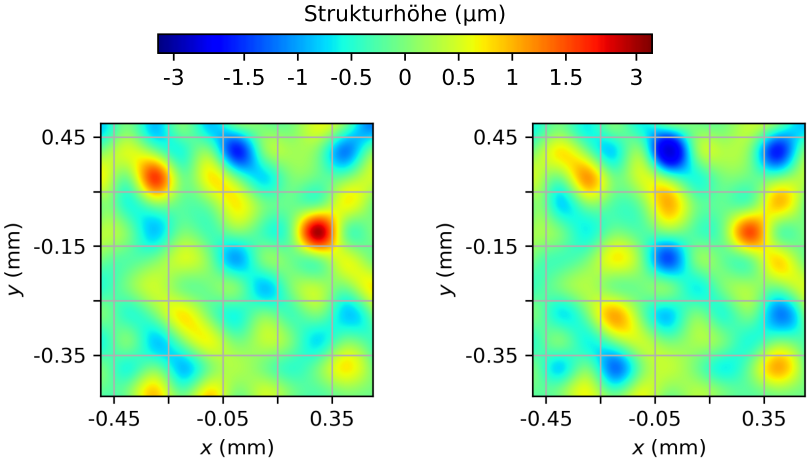
(a) $\Delta x = \Delta y = 100\,\mu m$
$\sigma = \begin{pmatrix} 0,5 & 0 \\ 0 & 0,5 \end{pmatrix} \mu m$

(b) $\Delta x = \Delta y = 100\,\mu m$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 0,5 \end{pmatrix} \mu m$

(c) $\Delta x = \Delta y = 100\,\mu m$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mu m$

(d) $\Delta x = \Delta y = 100\,\mu m$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \mu m$

Abbildung 3.11: Streuverhalten strukturierter Platten für Zellen der Größe 2x2.



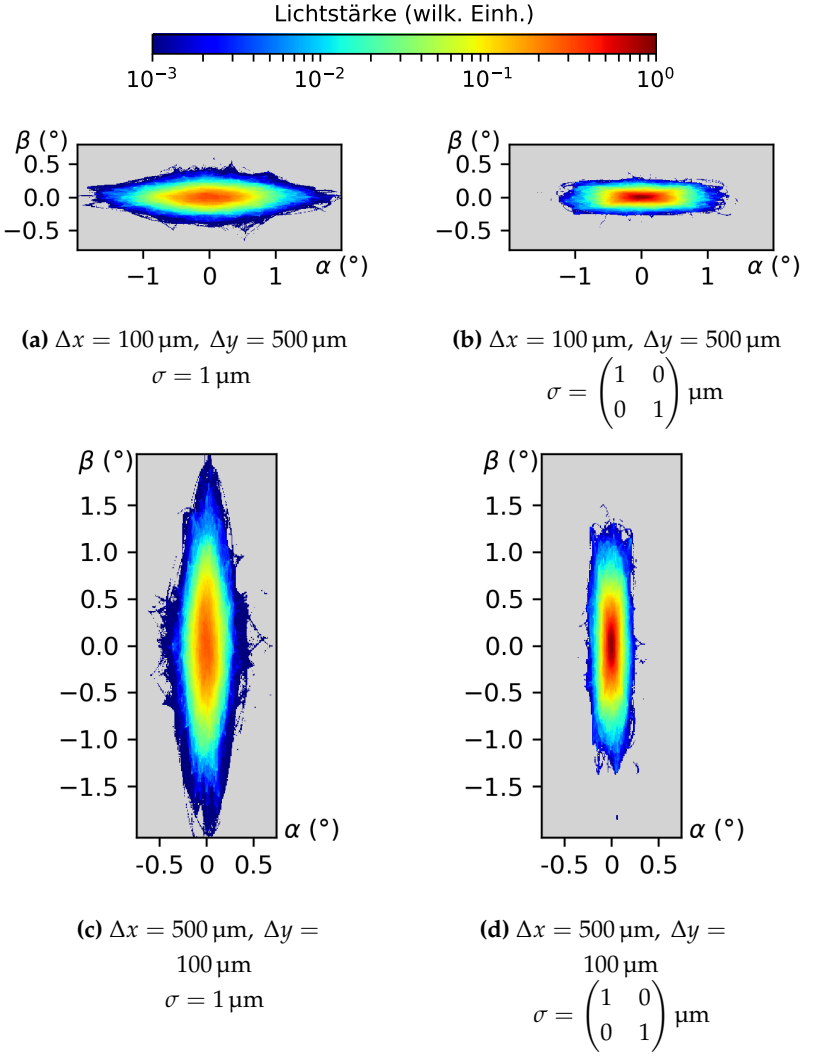
(a) $\Delta x = \Delta y = 100\,\mu\text{m}$
$\sigma = \begin{pmatrix} 0,5 & 0 \\ 0 & 0,5 \end{pmatrix}\,\mu\text{m}$

(b) $\Delta x = \Delta y = 100\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 0,5 \end{pmatrix}\,\mu\text{m}$

(c) $\Delta x = \Delta y = 100\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\,\mu\text{m}$

(d) $\Delta x = \Delta y = 100\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\,\mu\text{m}$

Abbildung 3.12: Strukturhöhen der Platten für Zellen der Größe 2x2. Die grauen Linien kennzeichnen die Zellengrößen. Die Farbskala ist unterhalb von $-1{,}5\,\mu\text{m}$ und oberhalb von $1{,}5\,\mu\text{m}$ logarithmisch.



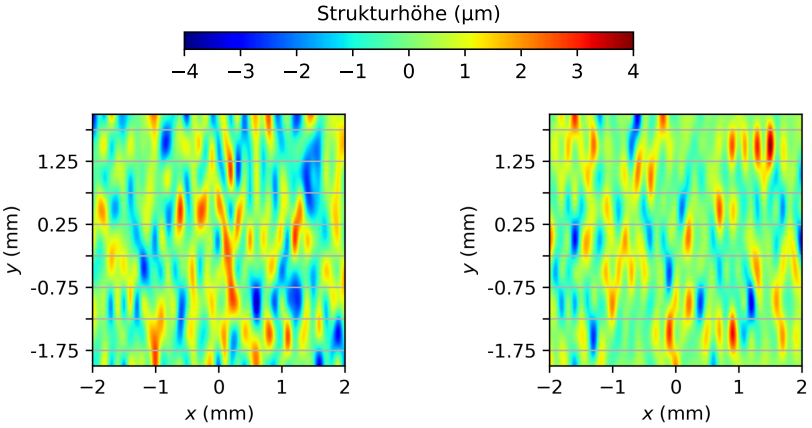
(a) $\Delta x = 100\,\mu\text{m},\ \Delta y = 500\,\mu\text{m}$
$\sigma = 1\,\mu\text{m}$

(b) $\Delta x = 100\,\mu\text{m},\ \Delta y = 500\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\,\mu\text{m}$

(c) $\Delta x = 500\,\mu\text{m},\ \Delta y = 100\,\mu\text{m}$
$\sigma = 1\,\mu\text{m}$

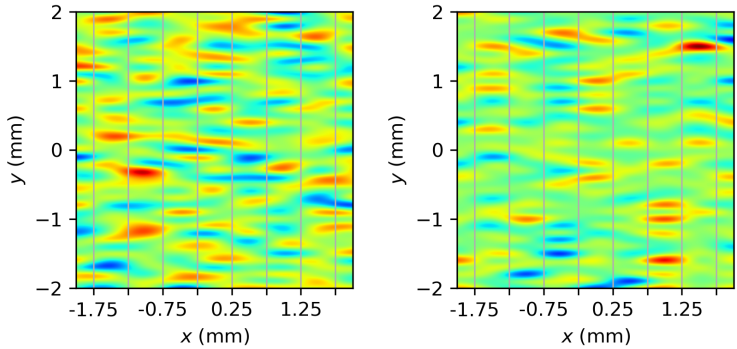
(d) $\Delta x = 500\,\mu\text{m},\ \Delta y = 100\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\,\mu\text{m}$

Abbildung 3.13: Streuverhalten strukturierter Platten für Punktgitter mit unterschiedlichen Auflösungen in x- und y-Richtung.



(a) $\Delta x = 100\,\mu\text{m},\ \Delta y = 500\,\mu\text{m}$
$\sigma = 1\,\mu\text{m}$

(b) $\Delta x = 100\,\mu\text{m},\ \Delta y = 500\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mu\text{m}$



(c) $\Delta x = 500\,\mu\text{m},\ \Delta y = 100\,\mu\text{m}$
$\sigma = 1\,\mu\text{m}$

(d) $\Delta x = 500\,\mu\text{m},\ \Delta y = 100\,\mu\text{m}$
$\sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mu\text{m}$

Abbildung 3.14: Strukturhöhen der Platten für Punktgitter mit unterschiedlichen Auflösungen in x- und y-Richtung. Die grauen Linien kennzeichnen die Zellengrößen. Zur besseren Sichtbarkeit sind nur die Zellengrenzen in der Richtung der größeren Ausdehnung eingezeichnet.

können die Gitterabstände in x- und y-Richtung unterschiedlich gewählt werden (vgl. Abb. 3.13). Lichtstärkeverteilungen von Scheinwerfern mit Matrix-LED-Systemen haben neben horizontalen auch vertikale HDGs [64–66]. Um je nach Anforderungen die HDGs unterschiedlich aufzuweichen, können die Strukturen mit asymmetrischem Streuverhalten verwendet werden.

3.3 FERTIGUNG

Die Linsen werden im Spritzgussverfahren hergestellt [15]. Der thermoplastische Kunststoff PMMA wird erhitzt und in den Hohlraum eines Stahlwerkzeuges gespritzt [67]. Nach dem Abkühlen nimmt das Material einen festen Zustand an. Durch die Form des Werkzeuges bzw. des Werkzeugeinsatzes wird die Linse und ihre Mikrostruktur vorgegeben. In Abb. 3.15 sind die in dieser Arbeit verwendeten Werkzeughälften dargestellt. Die Werkzeugeinsätze mit der strukturierten Lichtaustritts-

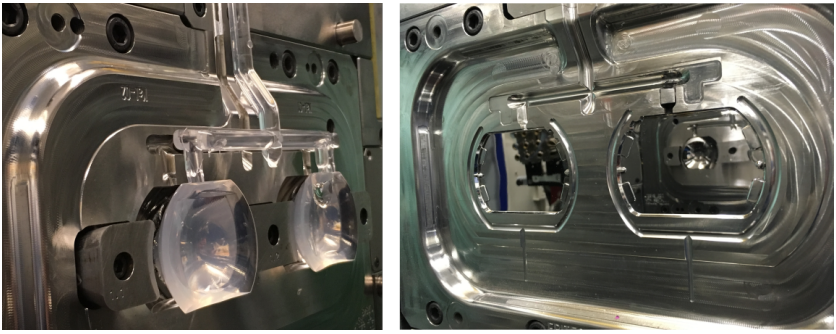


Abbildung 3.15: Links: Werkzeughälfte mit Einsatz für die Lichtaustrittsseite der Linse sowie spritzgegossene Linsen. Rechts: Werkzeughälfte mit Lichteintrittseite der Linse.

seite können gewechselt werden. Ein Werkzeugeinsatz ist in Abb. 3.16 dargestellt. Die Einsätze werden durch konventionelle Verfahren ohne

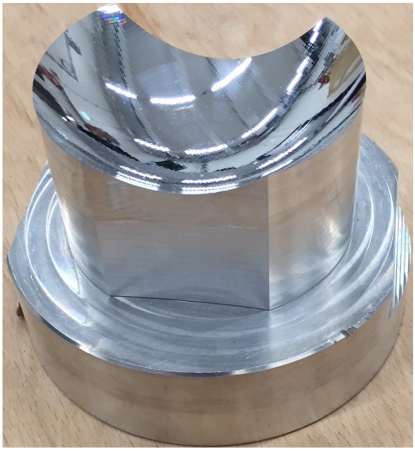


Abbildung 3.16: Werkzeugeinsatz mit Mikrostruktur.

Struktur und mit 50 µm Aufmaß hergestellt. Die Struktur wird durch eine Drehmaschine mit einem Diamantschneidewerkzeug in den Einsatz eingebracht [68]. Um die Bearbeitung eines Stahleinsatzes durch ein Diamantschneidewerkzeug zu ermöglichen, muss der Diamant durch Ultraschall in Schwingung versetzt werden [69]. Die Schwingung unterbricht mit einer Frequenz von der Größenordnung im 100 Kilohertz-Bereich den Schneidvorgang, wodurch der Verschleiß des Schneidwerkzeugs soweit reduziert wird, dass eine Bearbeitung eines Stahlbauteils erst ermöglicht wird [70–72]. In Abb. 3.17 ist die im Förderprojekt verwendete Drehmaschine und ein Diamantschneidwerkzeug dargestellt.

Im Rahmen des Förderprojektes wurde von der Firma Innolite ein neuartiger Steuerungsansatz für Drehmaschinen entwickelt, um eine höhere Präzision bei der Fertigung zu erreichen [74, 75]. Die Drehmaschine fertigt dabei auf Basis eines polaren Punktgitters, wie für eine kleine Punktanzahl schematisch in Abb. 3.18 dargestellt. Eine typische Auflösung des Punktgitters für eine mikrostrukturierte Linse mit einem

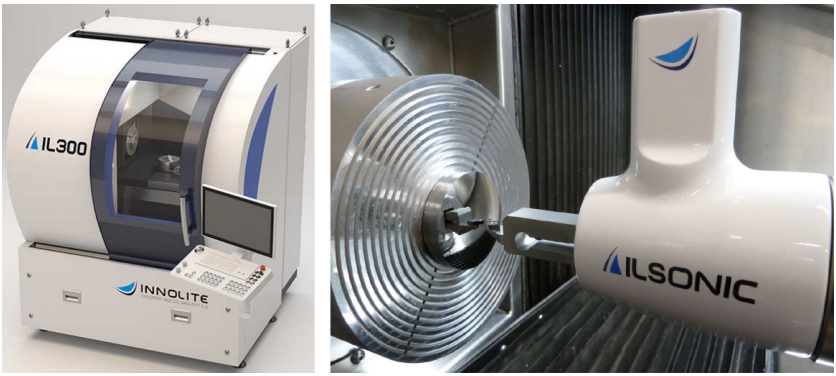


Abbildung 3.17: Links: Im Projekt verwendete Drehmaschine „Innolite IL 300“ [73]. Rechts: Bauteil und Diamantwerkzeug auf der Drehmaschine [74].

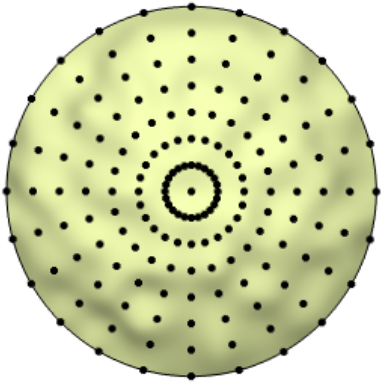


Abbildung 3.18: Schematische Darstellung eines polaren Punktgitters.

Abbildung 3.19: Links: In dem Projekt verwendete Messmaschine Mahr MFU200 [77]. Rechts: Werkzeugeinsatz auf Messmaschine [61].

Durchmesser von 70 mm beträgt 0,05° und 5 µm. Dadurch ergeben sich ca. 50 Millionen Punkte. Diese Punkte werden aus der strukturierten Linsenfläche generiert und in dem Dateiformat „Hierarchical Data Format 5“ (HDF5) an die Fertigung übergeben [76].

3.4 VERMESSUNG

Um zu überprüfen, ob die Fertigung der mikrostrukturierten Werkzeugeinsätze erfolgreich ist, werden die Einsätze optisch vermessen. Für die Messung der Einsätze wird das System MFU200 der Firma Mahr verwendet, wie in Abb. 3.19 dargestellt [77]. Das Messsystem ist eine zylindrische Koordinaten-Messmaschine, bestehend aus zwei linearen (X, Z) und einer drehbaren Messachse (C). Die Klemmvorrichtung zur

Aufnahme des Werkzeugeinsatzes kann durch vier Achsen verschoben und gekippt werden, um den Einsatz zentral und orthogonal zum Messsensor zu platzieren. Zur Bestimmung des Zenits kann eine zusätzliche lineare Achse (Y) verwendet werden. Das Messverfahren basiert auf Weißlichtinterferometrie. Der Messsensor hat einen Arbeitsbereich von $\pm 20°$ zur Flächennormalen bei einem Abstand von (300 ± 5) µm. Um die äußeren Bereiche der Einsätze, mit einer Steilheit größer 20° zu vermessen, kann der Messsensor gekippt werden (B-Achse). Im Fokus des optischen Systems beträgt der Strahldurchmesser 2 µm.

Zur Zentrierung des Einsatzes wird entlang zweier Linien gemessen, wie in Abb. 3.20 dargestellt. Der Scheitelpunkt wird über Ausgleichspolyno-

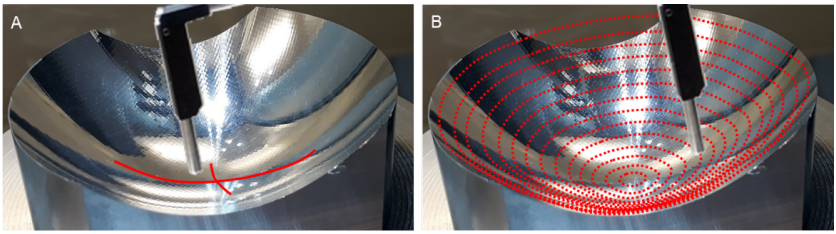


Abbildung 3.20: Bewegung des Messkopfes zur Justierung (A) und Messung (B) [61].

me bestimmt. Die Vermessung des Einsatzes erfolgt durch die Messung von konzentrischen Kreisen. Die X- und Z-Achsen fahren nach Soll-Daten eine Position an, die zu einem vorgegebenen Radius gehört. An der Stelle wird um die C-Achse mit konstanter Geschwindigkeit gedreht und währenddessen mit konstantem Winkelabstand Messwerte aufgenommen. Die in dieser Arbeit verwendete Schrittweite beträgt 0,1° und 25 µm. Bei den verwendeten Linsen mit 70 mm Durchmesser ergeben sich Punktgitter mit ca. 5 Millionen Punkten. Diese Messauflösung stellt einen Kompromiss zwischen Detailgrad und Messzeit, die hierbei bei mehreren Stunden lag.

3.5 FLÄCHENRÜCKFÜHRUNG

Um die Verwendbarkeit der gefertigten Werkzeugeinsätze zu überprüfen, ist ein Soll-Ist-Vergleich zwischen Messdaten und Fertigungsdaten nicht sinnvoll [78, 79]. Bei dem Vergleich können zwar grobe Unterschiede ausgemacht und daraus eventuelle Probleme bei der Fertigung abgeleitet werden, jedoch ist das photometrische Ergebnis der abgeformten Linse entscheidend. Daher muss auf Basis des gemessenen Punktgitters eine lichttechnische Simulation durchgeführt werden. Damit kann die LSV der abgeformten Linse bewertet werden. Zur Bestimmung der LSV durch eine Ray-Tracing-Simulation wird Information über die Fläche zwischen den Punkten benötigt. Daher muss eine Flächendarstellung des Punktgitters bestimmt werden.

Eine Interpolation der Punkte, wie in Kap. 3.2 beschrieben, ist ungeeignet. Die Interpolation würde das Messrauschen mit in die Fläche übernehmen und benötigt für jeden Messpunkt einen Kontrollpunkt (vgl. Kap. 3.2.2). Durch die Übernahme des Messrauschens würde die Abbildungsqualität der Linse in der Simulation schlechter sein als sie tatsächlich ist. Eine Fläche mit ca. 5 Millionen Kontrollpunkten ist im Ray-Tracing nicht gut handhabbar. Zum Beispiel würde die Berechnung einer Hierarchie von Begrenzungsvolumen (vgl. Kap. 2.1.2) und das anschließende Ray-Tracing vergleichsweise lange dauern. Daher muss eine bessere Darstellung der Fläche bestimmt werden, die das Punktgitter bestmöglich approximiert.

3.5.1 ÜBERSICHT VERSCHIEDENER VERFAHREN ZUR RÜCKFÜHRUNG

Wie bei der Interpolation müssen Kontrollpunkte, Knoten und eventuelle Parameter für die Messpunkte bestimmt werden. Für das Anpassen

einer Fläche an ein Punktgitter existieren verschiedene Ansätze. PIEGEL und TILLER berechnen eine Fläche, indem entlang zweier Richtungen mehrere Kurvenanpassungen durchgeführt werden. Das entspricht dem Vorgehen aus Kapitel 3.2.2, jedoch mit Methode der kleinsten Quadrate [40, 80]. Die Knoten und Parameter werden vorab berechnet. Für den Algorithmus müssen die Messdaten gitterbasiert vorliegen. Diese Methode wurde durch HEMMERICH im Bezug auf mikrostrukturierte Flächen hin untersucht [81]. In der Arbeit von Hemmerich wurde die Flächenrückführung erfolgreich für kartesische Punktgitter durchgeführt. Eine Anwendung auf polare Gitter ist möglich, indem die Kurvenanpassungen in radialer und azimuthaler Richtung verlaufen. Entsprechend verlaufen die u- und v- Richtungen der resultierenden Fläche bzw. des Kontrollpolygons radial und azimuthal, wie in Abb. 3.21 dargestellt. Bei dem Importieren trennt CATIA eine so parametrisierte Fläche in zwei Flächen auf. Das ist zwar kein direktes Problem, jedoch zeigt sich,

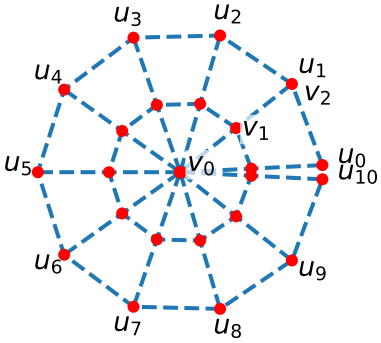


Abbildung 3.21: Kontrollpunkte (rot) und Kontrollpolygon (blau) einer B-Spline Fläche. Die Lücke zwischen u_0 und u_{10} dient der Veranschaulichung.

dass CATIA polar-parametrisierte Flächen beim Einladen umrechnet. Beim Umrechnen besteht die Möglichkeit der Datenverluste. Ein weiterer Nachteil ist die ungleiche Verteilung der Kontrollpunkte. Durch

das polare Gitter sind im Inneren der Fläche die Abstände der Kontrollpunkte geringer als im Randbereich. Eine ausreichende Anzahl an Kontrollpunkten im Randbereich erfordert eine unnötig hohe Anzahl an Kontrollpunkten im Inneren.

BECKER ET AL. führen eine zweistufige Flächenanpassung durch, die für nicht-strukturierte Gitter verwendet werden kann [82]. In der ersten Stufe wird im Gegensatz zu der Methode von PIEGEL und TILLER ein komplexerer Algorithmus zur Bestimmung der Parameter verwendet. Durch Aufstellen und Lösen eines Gleichungssystems werden die Positionen der Kontrollpunkte bestimmt. In dem zweiten Schritt wird als Flächendarstellung „NURBS“ (Non-Uniform Rational B-Spline) verwendet [40]. Durch NURBS ergeben sich weitere Freiheitsgrade, sogenannte Gewichte. Mit einem Gradientenverfahren werden die Gewichte und die Positionen der Kontrollpunkte optimiert. Die Methode ist für nominelle Daten ausgelegt, die nicht durch ein Messrauschen verunreinigt sind.

ZHANG ET AL. wählen aus den Messdaten einige Punkte aus [83]. Die Punkte dienen als Stützstellen zur Bestimmung einer interpolierenden B-Spline-Fläche. Um die Fläche an die Messdaten anzupassen, wird das Newton-Raphson-Verfahren angewendet. Bei dem Verfahren werden die Positionen der Stützstellen verschoben, bis die Differenz zwischen Fläche und Messdaten minimal ist. Es werden keine Informationen gegeben, wie die Stützstellen bzw. ihre Anzahl gewählt werden sollen.

In der Arbeit von CARLSSON wird eine NURBS-Flächenrückführung durch die Gauss-Newton-Methode untersucht [84]. Variiert werden Parameter, Kontrollpunkte und Gewichte. Die Anzahl der Messpunkte beträgt ca. 500 und die Größe der Kontrollpunktmatrix der Fläche 7x7. Bei dem Verfahren wird die Jacobi-Matrix aufgestellt. Die Größe der Matrix wächst in beiden Dimensionen linear mit der Anzahl an Messpunkten. Da bei der Rückführung der vermessenen Werkzeugeinsätze

die Anzahl an Messpunkten im Bereich mehrerer Millionen liegt, ist die Methode nicht anwendbar.

RANDRIANARIVONY und BRUNNEN nutzen beschränkte nichtlineare Optimierung, um neben den Positionen der Kontrollpunkte und der Gewichte auch die Knoten einer NURBS-Fläche zu optimieren [85, 86]. Es werden keine Informationen gegeben, wie die Parameter der Messpunkte bestimmt oder die Anzahl der Kontrollpunkte gewählt werden soll.

BRUJIC ET AL. stellen eine Methode vor, um eine bereits angepasste Fläche noch besser anzupassen [87]. Zur Berechnung der neuen Kontrollpunkte wird die Funktion

$$f = \sum_{k=0}^{M-1} \left(\vec{Q}_k - \vec{S}(\bar{u}_k, \bar{v}_k) \right)^2 + \gamma \sum_{i=0}^{n} \sum_{j=0}^{m} \left(\vec{P}_{ij} - \vec{S}(\bar{u}_{ij}, \bar{v}_{ij}) \right)^2 + \delta \sum_{i=0}^{n} \sum_{j=0}^{m} \left(\vec{P}_{ij} - \vec{P}_{ij}^{\,0} \right)^2 \tag{3.18}$$

bezüglich den neuen Kontrollpunkten $\vec{P}_{ij}$ minimiert. Der erste Term bestraft den Abstand zwischen der neuen Fläche $\vec{S}$, beschrieben durch die $\vec{P}_{ij}$, und den Messpunkten $\vec{Q}_k$. Die Parameter $\bar{u}_k, \bar{v}_k$ ergeben sich durch Projektion der Messpunkte auf die Startfläche, beschrieben durch die Kontrollpunkte $\vec{P}_{ij}^{\,0}$. Der zweite Term bestraft den Abstand zwischen Fläche und den neuen Kontrollpunkten, wodurch die neue Fläche glatter verläuft. Die Stärke der Glättung kann durch den Parameter γ eingestellt werden. In Situationen, wo die Form der Startfläche in Bereichen einer dünnen Messpunktdichte erhalten bleiben soll, kann der dritte Term verwendet werden. Der dritte Term bestraft den Abstand zwischen den Kontrollpunkten der neuen Fläche zu denen der Startfläche. Wie stark sich Start und Endfläche ähnlich seien sollen, kann durch den Parameter δ eingestellt werden.

3.5.2 ITERATIVE GEOMETRISCHE APPROXIMATION

Zur Rückführung wird der „Iterative Geometrische Approximations"-Algorithmus verwendet [88]. Der Algorithmus hat die Vorteile, dass er ohne Aufstellen bzw. Lösen eines Gleichungssystems auskommt, keinen zusätzlichen Term zum Glätten (z. B. Biegeenergie) benötigt und ermöglicht, die Approximations-Genauigkeit lokal beeinflussen zu können. Zur Flächenrückführung wird eine Ausgangsfläche, beschrieben durch B-Splines, benötigt. Der Algorithmus berechnet den Abstand zwischen Messdaten und Fläche und verschiebt die Kontrollpunkte entsprechend des Differenzvektors.

Die Ausgangsfläche ist durch $\vec{S}^{(0)}(u,v)$ und die Fläche nach jedem Iterationsschritt durch $\vec{S}^{(\alpha)}(u,v)$ beschrieben. Der Index α gibt die Anzahl der Iterationsschritte an. Die Messdaten werden durch $\vec{Q}_k,\ k = 0,\ \ldots, N$ beschrieben. Für jeden Messpunkt wird die orthogonale Projektion auf die Fläche $\vec{S}^{(\alpha)}(u,v)$ bestimmt, bzw. die Parameter $\bar{u}_k^{(\alpha)}, \bar{v}_k^{(\alpha)}$, die den projizierten Punkt beschreiben. Wie die orthogonale Projektion bestimmt wird, ist in Kap. 3.5.3 gegeben. Wie in Kap. 2.2.2 geben u_s und v_t die Elemente der Knotenvektoren der Fläche an. Zu den Parametern $\bar{u}_k^{(\alpha)}, \bar{v}_k^{(\alpha)}$ werden die Indizes s und t so bestimmt, dass $u_s \leq \bar{u}_k^{(\alpha)} \leq u_{s+1}$ und $v_t \leq \bar{v}_k^{(\alpha)} \leq v_{t+1}$ gilt. Der projizierte Punkt $\vec{S}^{(\alpha)}(\bar{u}_k^{(\alpha)}, \bar{v}_k^{(\alpha)})$ berechnet sich aus den nicht verschwindenden Basis-Funktionen $N_{i,p}(\bar{u}_k^{(\alpha)}), i = s-p, \ldots, s$ und $N_{j,q}(\bar{v}_k^{(\alpha)}), j = t-q, \ldots, t$ und damit aus den Kontrollpunkten $P_{ij}^{(\alpha)},\ i = s-p, \ldots, s,\ j = t-q, \ldots, t$. Die Abhängigkeit ist in Abb. 3.22 veranschaulicht. Zur Anpassung der Fläche an die Punkte werden die Differenzvektoren $Q_k - \vec{S}^{(\alpha)}(\bar{u}_k^{(\alpha)}, \bar{v}_k^{(\alpha)})$ bei jedem Iterationsschritt berechnet. Die Differenzvektoren werden mit den Basis-Funktionen gewichtet und zu den Kontrollpunkten hinzu addiert. Dadurch wird die Fläche iterativ näher an die Punkte herangezogen. Ein Iterationsschritt ist in der Auflistung 3.1 beschrieben.

Programmauflistung 3.1: Iterative Geometrische Approximation [88]

```
1  for k ← 0, N
2    ū_k, v̄_k ← getProjectedPoint(Q⃗[k])
3    s, t ← getSpan(ū_k, v̄_k)
4    for i ← s − p, s
5      for j ← t − q, t
6        W⃗_num[i][j] ← W⃗_num[i][j] + N_{i,p}(ū_k) N_{j,q}(v̄_k) ·
             ( Q⃗[k] − S⃗^(α)(ū_k, v̄_k) )
7        W_den[i][j] ← W_den[i][j] + N_{i,p}(ū_k^(α)) N_{j,q}(v̄_k^(α))
8      end for
9    end for
10 end for
11 for i ← 0, n
12   for j ← 0, m
13     P⃗_{i,j}^(α+1) ← P⃗_{i,j}^(α) + W⃗_num[i][j] / W_den[i][j]
14   end for
15 end for
```

Die Iteration wird so lange durchgeführt, bis die Änderung der mittleren Abweichung zwischen Fläche und Punktgitter kleiner als 1 nm beträgt. Es wird angenommen, dass ab einem Unterschied von 1 nm die Abweichungen keine lichttechnische Auswirkung mehr haben. Wie beschrieben, benötigt der Algorithmus eine Ausgangsfläche. In dieser Arbeit wird dafür die hoch-aufgelöste unstrukturierte Fläche (vgl. Kap. 3.2.1 „zweite Variante“) verwendet.

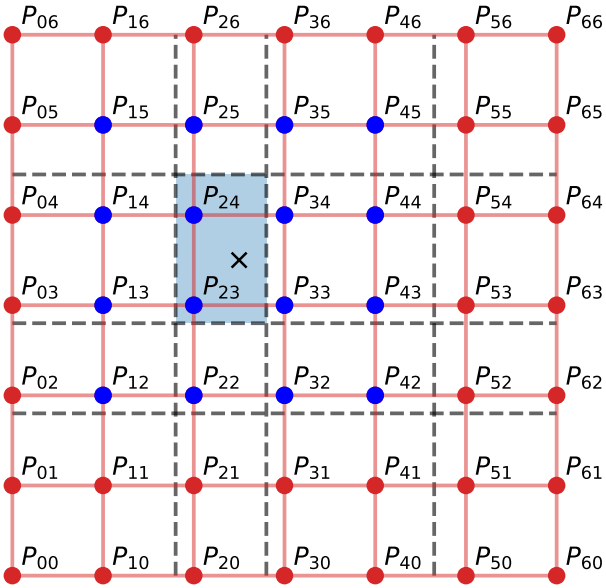


Abbildung 3.22: Ein bikubisches Kontrollpolygon [88]. Die schwarzen unterbrochenen Linien geben die Positionen der inneren Knoten an. Das schwarze Kreuz markiert den projizierten Punkt. Die blau markierten Kontrollpunkte beeinflussen die Fläche an der Stelle mit dem schwarzen Kreuz.

3.5.3 ORTHOGONALE PROJEKTION AUF FLÄCHEN

Zur Projektion der Punkte auf die Fläche wird ein Algorithmus verwendet, der die Krümmung der Fläche nutzt, um die Projektionspunkte zu bestimmen [89, 90]. Der Punkt, dessen Projektion auf die Fläche $\vec{S}(u,v)$ gesucht wird, ist durch $\vec{x}$ gegeben. Der Algorithmus benötigt einen Startpunkt $\vec{p}_0$. Der Startpunkt $\vec{p}_0$ stellt eine erste Vermutung des Projektionspunktes dar und ist im Parameterraum durch die Koordinaten $(\bar{u},\bar{v})$ gegeben. Die Tangentialvektoren von der Fläche an der Stelle $\vec{p}_0$ sind $\vec{S}_1$

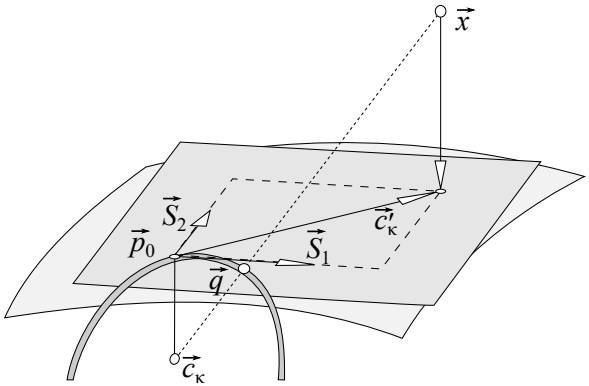


Abbildung 3.23: Schematische Darstellung der orthogonalen Projektion (modifiziert von [89]).

und $\vec{S}_2$. In Abb. 3.23 ist der Aufbau veranschaulicht. Der Normalenvektor ist $\vec{n} = \vec{S}_1 \times \vec{S}_2$. Um den geratenen Projektionspunkt $\vec{p}_0$ genauer zu bestimmen, wird der Differenzvektor $\vec{x} - \vec{p}_0$ als Linearkombination aus Tangentialvektoren und Normalenvektor dargestellt [89]:

$$\vec{x} - \vec{p}_0 = \lambda_1 \vec{S}_1 + \lambda_2 \vec{S}_2 + \nu \vec{n} \tag{3.19}$$

Durch das Lösen des Gleichungssystems (3.19) ergeben sich die Parameter λ_1, λ_2 und ν. Die zweifachen Ableitungen der Fläche sind gegeben durch

$$\vec{S}_{11} = \frac{\partial^2}{\partial u^2}\vec{S}(u,v) \qquad \vec{S}_{12} = \vec{S}_{21} = \frac{\partial^2}{\partial u \partial v}\vec{S}(u,v) \qquad \vec{S}_{22} = \frac{\partial^2}{\partial v^2}\vec{S}(u,v) \tag{3.20}$$

Mit den Koeffizienten der ersten und zweiten Fundamentalform [91]

$$g_{ij} = \left\langle \vec{S}_i, \vec{S}_j \right\rangle \quad \text{und} \quad h_{ij} = \left\langle \vec{S}_{ij}, \vec{n} \right\rangle \tag{3.21}$$

berechnet sich die Krümmung κ in Richtung des Tangentialvektors $\vec{c}'_\kappa = \lambda_1 \vec{S}_1 + \lambda_2 \vec{S}_2$ aus [89]

$$\kappa = \frac{\sum_{i,j=1}^{2} \left(h_{ij} \lambda_i \lambda_j \right)}{\sum_{i,j=1}^{2} \left(g_{ij} \lambda_i \lambda_j \right)}. \tag{3.22}$$

In der Ebene, die die Punkte $\vec{p}_0$, $\vec{x}$ und den Vektor $\vec{n}$ enthält, befindet sich der Krümmungskreis mit Krümmung κ und dem Mittelpunkt [89]

$$\vec{c}_\kappa = \vec{p}_0 + \frac{\vec{n}}{\kappa}. \tag{3.23}$$

Die orthogonale Projektion des Punktes $\vec{x}$ auf den Krümmungskreis ergibt den Punkt [89]

$$\vec{q} = \vec{c}_\kappa + \frac{\vec{x} - \vec{c}_\kappa}{\| \kappa \left(\vec{x} - \vec{c}_\kappa \right) \|}. \tag{3.24}$$

Mit dem Projektionspunkt $\vec{q}$ wird die Änderung Δh der Parameter berechnet aus [89]:

$$\Delta h = \operatorname{sgn} \left(\langle \vec{c}'_\kappa, \vec{q} - \vec{p}_0 \rangle \right) \cdot \sqrt{2 \frac{\| \vec{c}'_\kappa \times (\vec{q} - \vec{p}_0) \|}{|\kappa| \| \vec{c}'_\kappa \|^3}} \tag{3.25}$$

Hierbei gibt $\operatorname{sgn}()$ die Vorzeichenfunktion an. Die Parameter werden aktualisiert durch [89]:

$$\bar{u} \leftarrow \bar{u} + \lambda_1 \Delta h \tag{3.26}$$

$$\bar{v} \leftarrow \bar{v} + \lambda_2 \Delta h \tag{3.27}$$

$$\vec{p}_0 \leftarrow \vec{S}(\bar{u}, \bar{v}) \tag{3.28}$$

Die Schritte (3.19) bis (3.28) werden solange wiederholt, bis die gewünschte Genauigkeit erreicht ist. Als Konvergenzkriterium wird verwendet, dass der Kosinus des Winkels zwischen $\vec{n}$ und $\vec{x} - \vec{p}_0$ kleiner als 10^{-8} oder der Abstand zwischen $\vec{x}$ und $\vec{p}_0$ kleiner als 1×10^{-10} mm ist. Die Werte wurden empirisch gefunden. Bei größeren Werten endete die Konvergenz des Algorithmus vermutlich auf Grund der endlichen Rechengenauigkeit.

3.5.4 UNTERSUCHUNG DER KONVERGENZ

Zur Überprüfung der Konvergenz wird eine Linse mit einem Radius von 35 mm mit einer Mikrostruktur versehen. Für die Erstellung der Mikrostruktur wird ein quadratisches Gitter mit einem Punktabstand von 100 µm verwendet. Die Amplituden werden aus einer Normalverteilung mit einer Standardabweichung von 0,5 µm bestimmt. Von der strukturierten Linsenfläche wird ein polares Punktgitter mit einer Auflösung von 0,1° und 25 µm abgeleitet. Diese Auflösung entspricht der Messauflösung (vgl. Kap. 3.4). Zum Starten der Flächenrückführung wird eine unstrukturierte Linsenfläche verwendet, deren Kontrollpunkte einen lateralen Abstand von $\Delta P_{x,y} = 100\,\mu\text{m}$ haben und damit der Auflösung der strukturierten Linse entsprechen. In Abb. 3.24 (blaue Kurve) ist die mittlere orthogonale Abweichung zwischen Punktgitter und Fläche in Abhängigkeit der Iterationsschritte dargestellt. Der Algo-

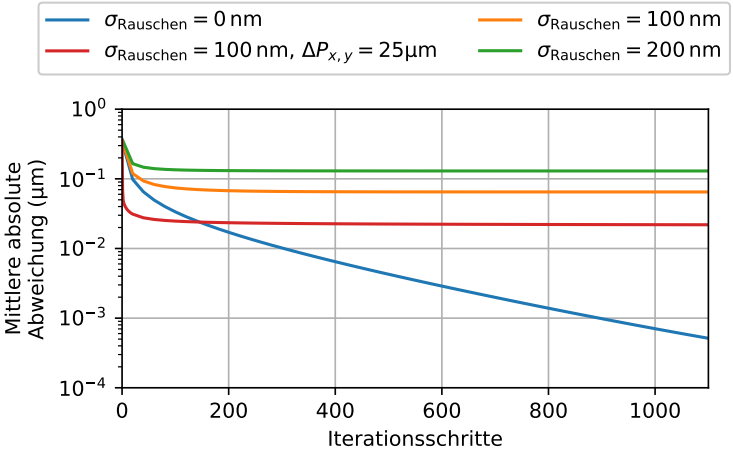


Abbildung 3.24: Konvergenzverhalten des Algorithmus 3.1 für eine mikrostrukturierte Linse.

rithmus erreicht nach 1100 Iterationsschritten eine mittlere Abweichung von 0,5 nm, wobei die Konvergenz noch nicht erreicht wird. Theoretisch ist eine Konvergenz bis zur Rechnergenauigkeit möglich, da es sich hier um nominelle Daten handelt. Für weitere Tests wird das Punktgitter mit einem Rauschen versehen, indem jeder Punkt entlang der optischen Achse der Linse verschoben wird. Die Verschiebung wird dabei aus einer Normalverteilung generiert. Die Standardabweichung der Verteilung beträgt bei einem Versuch 100 nm und bei einem anderen Versuch 200 nm. Das Konvergenzverhalten bei der Rückführung ist in Abb. 3.24 dargestellt. Der Algorithmus konvergierte gegen eine mittlere orthogonale Abweichung von 65 nm bzw. 130 nm. Wird die Auflösung der Ausgangsfläche auf einen Abstand von 25 µm erhöht, so reduzierte sich die mittlere orthogonale Abweichung für das Punktgitter mit 100 nm Rauschen auf 22 nm (rote Kurve in Abb. 3.24). Das bedeutet, dass ein Teil des Rauschens durch den Algorithmus in die Fläche übernommen wurde. Durch den geringeren Abstand der Kontrollpunkte von 25 µm anstatt 100 µm sind sechzehn Mal so viele Freiheitsgrade vorhanden. Um die Auswirkungen von einer Überanpassung zu demonstrieren, wird die nominelle strukturierte und alle zurückgeführten Linsenflächen in einer lichttechnischen Simulation für ein Scheinwerfermodul (vgl. Kap. 4) verwendet. Die simulierten Gradienten-Kurven (vgl. Kap. 3.1) sind in Abb. 3.25 dargestellt. Die Gradienten-Kurve der nominellen Linsenfläche stimmt im Rahmen des Simulationsrauschens mit den Gradienten-Kurven der Rückführungen überein, die auf Basis der Fläche mit einer Auflösung von 100 µm durchgeführt wird. Die Simulation mit der 25 µm-Ausgangsfläche zeigt auf Grund der Überanpassung einen weicheren Gradienten. Daher ist es wichtig, die Auflösung der Ausgangsfläche so gering wie möglich zu halten, um kein potentielles Messrauschen mit in die Fläche zu übernehmen.

Bei den Rückführungen von realen Messdaten wird die Auflösung der nominellen Fläche verwendet. Wie in Kap. 4 gezeigt wird, ist die Auf-

lösung im Rahmen dieser Arbeit immer ausreichend. Sollte bei einer Rückführung die Auflösung nicht ausreichend sein, so kann durch Hinzufügen von weiteren Spalten oder Zeilen (bzw. hinzufügen weiterer Knoten) in der Kontrollpunktmatrix die Anzahl an Freiheitsgrade der Fläche und damit die Genauigkeit der Rückführung erhöht werden.

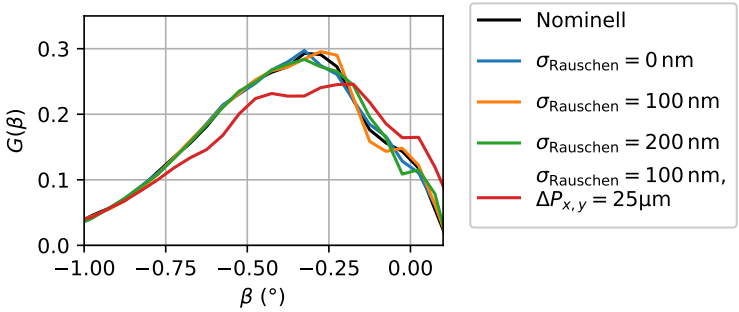


Abbildung 3.25: Simulierte Gradienten-Kurven in Abhängigkeit eines künstlichen Messrauschens und der Auflösung der Ausgangsfläche.

KAPITEL 4

ANWENDUNG DER PROZESSKETTE

In Abb. 4.1 ist eine Übersicht über die neue Prozesskette dargestellt. Anhand von zwei verschieden Mikrostrukturen wird die Funktion der neuen Prozesskette demonstriert. Eine Mikrostruktur wurde für den ECE-, die zweite für den FMVSS-Bereich ausgelegt. Üblicherweise sind die HDGs von Scheinwerfersysten für den FMVSS-Bereich deutlich weicher. Die Mikrostrukturen wurden manuell, d. h. in einem „trial-and-error"-Verfahren, von einem Optikingeniuer ausgelegt. Die Funktionweise der automatischen Korrekturschleife (vgl. Abb. 4.1) wird in Kap. 5 erläutert und demonstriert. Der Inhalt dieses Kapitels stammt zu Teilen aus der Veröffentlichung [61]. Es werden die zwei Mikrostrukturen dargestellt, zu denen jeweils zwei Werkzeugeinsätze gefertigt wurden. Die Ergebnisse der Flächenrückführungen und die darauf basierenden Simulationen werden präsentiert.

4.1 VERMESSUNG VON LICHTSTÄRKEVERTEILUNGEN

Zur Vermessung von Lichtstärkeverteilungen existieren mehrere Möglichkeiten [92]. Eine vergleichsweise schnelle Variante ist die indirekte Messung durch eine Bildverarbeitungs-Anlage (vgl. Abb. 4.2). Der Scheinwerfer wird in einem abgedunkelten Raum in 10 m Entfernung vor einer Messwand platziert. Das Scheinwerferlicht verursacht eine Leuchtdichte auf der diffus reflektierenden Messwand. Mit einer Leuchtdichtemesskamera wird ein Leuchtdichtebild aufgenommen. Über die

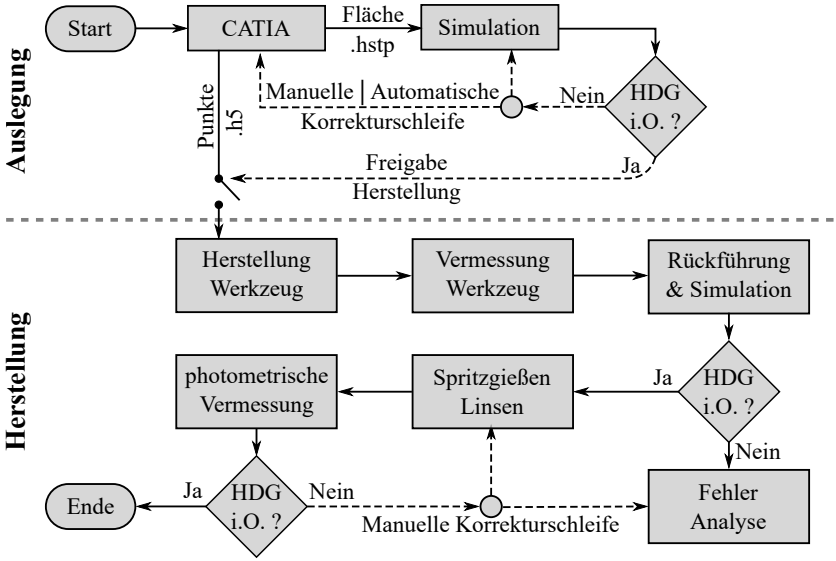


Abbildung 4.1: Neue Prozesskette zur Herstellung mikrostrukturierter Linsen.

Leuchtdichte kann auf die LSV des Scheinwerfers zurückgerechnet werden.

4.2 FÜR ECE-GEBIETE

Zur Überprüfung der neuen, in Kap. 3 vorgestellten Prozesskette wird das in Abb. 4.3 dargestellte Scheinwerfermodul *BiLED* verwendet. Das Modul besteht aus insgesamt fünf LED-Lichtquellen, die das Abblendlicht erzeugen. Der aus drei Kammern bestehende Reflektor reflektiert das Licht über eine Blende in Richtung der Linse. Die Linse bildet die Blendenebene auf die Straße ab und erzeugt hierdurch die HDG. Durch Vermessungen des BiLed-Moduls mit einer nicht-strukturierten Linse auf einer BV-Anlage, zeigte sich, dass oberhalb der HDG Streulicht von

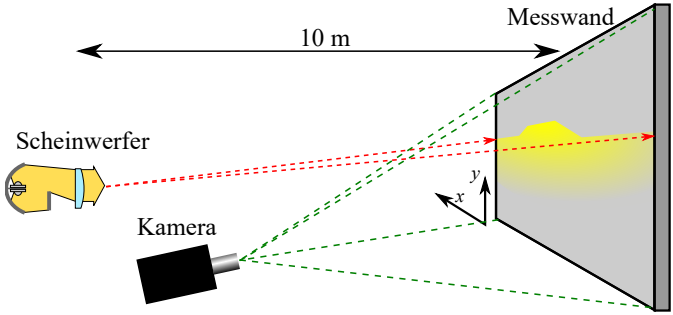


Abbildung 4.2: Schematische Darstellung einer Bildverarbeitungs-Anlage.

ca. 75 cd auftritt. Daher werden zu allen Simulationsergebnissen 75 cd hinzugefügt. Die von dem Modul ohne strukturierte Linse simulierte LSV sowie die Gradienten-Kurve sind in Abb. 4.4 dargestellt.

Die ausgelegte Mikrostruktur ist in Abb. 4.5 dargestellt. Wie in Kap. 3.2.1 beschrieben, basiert die Struktur auf einem Punktgitter, wobei hier eine Auflösung von 100 µm verwendet wird. Das BiLed-Modul erzeugt durch zusätzliche LEDs ein Fernlicht. Die Fernlicht-LEDs sind in Abb. 4.3 nicht dargestellt. Das Fernlicht wird wie das Abblendlicht durch die Linse gebrochen. Wie in Abb. 4.5 erkennbar, hat die Struktur in einem horizontalen Band eine größere Amplitude. Die unterschiedliche Ausprägung der Struktur ist nötig, um die HDG ausreichend aufzuweichen, ohne das Fernlicht des Systems zu stark zu beeinflussen. Die simulierte LSV und die Gradienten-Kurve sind in Abb. 4.6a dargestellt. Wie in Kap. 3 beschrieben, wird ein polares Punktgitter mit ca. 50,4 Millionen Punkten exportiert und zum Herstellen von zwei Werkzeugeinsätzen durch eine Diamantdrehmaschine verwendet. Die Werkzeugeinsätze werden vermessen, wodurch sich zwei polare Punktgitter mit je ca. 5 Millionen Punkten ergeben. Für jedes gemessene Punktgitter wird eine Flächenrückführung (vgl. Kap. 3.5.2) durchgeführt. Die Abweichungen der rückgeführten Fläche zu den Messdaten ist in Abb. 4.7 dargestellt.

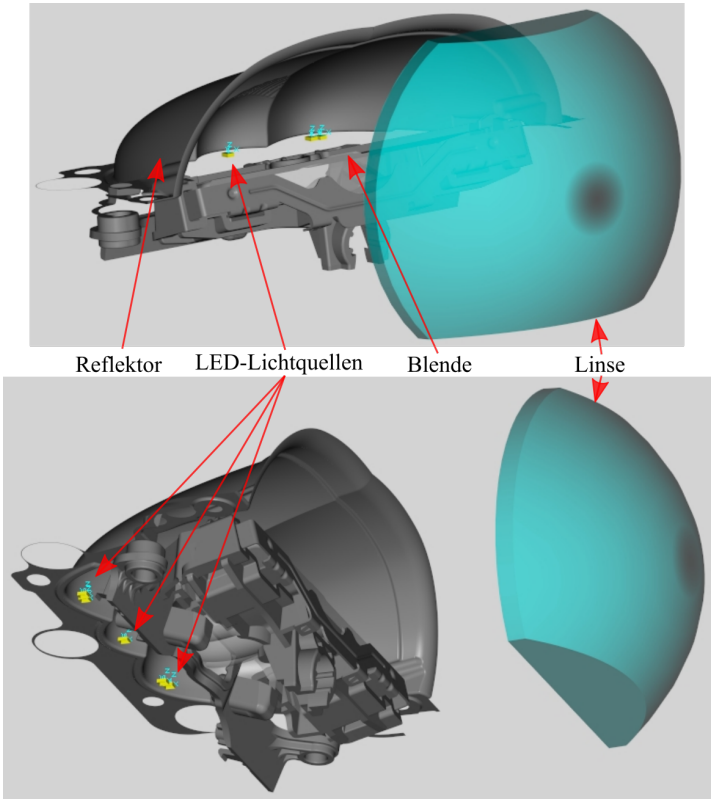


Abbildung 4.3: Aufbau des *BiLED*-Moduls, das zum Testen der neuen Prozesskette verwendet wird.

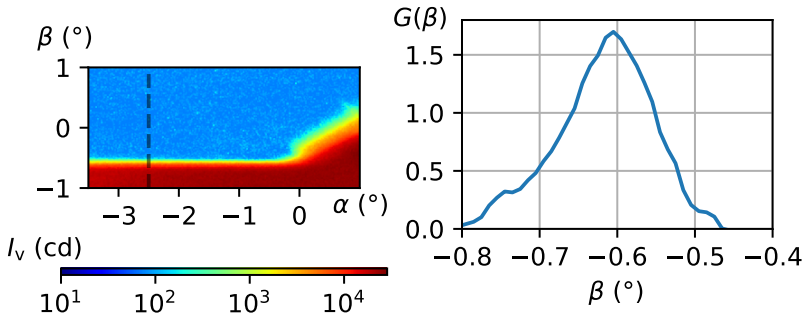


Abbildung 4.4: Simulation der LSV und der Gradienten-Kurve des *BiLED*-Moduls mit nicht-strukturierter Linse. Die gestrichelte Linie in der LSV gibt die Position der Gradienten-Kurve an.

Die mittlere Abweichung zwischen Fläche und Punktgitter beträgt 32 nm für Werkzeugeinsatz 1 und 47 nm für Werkzeugeinsatz 2. Wie in Abb. 4.7 zu erkennen ist, sind in den Messdaten von Werkzeugeinsatz 2 kreisförmige Abweichungen mit einer Größe von ca. 0,1 µm enthalten. Diese können entweder durch einen Fertigungs- oder Messfehler entstehen. Die Farbskala von Abb. 4.7 hat einen maximalen Wert von 10 µm, da vereinzelt Ausreißer in der Größenordnung von Mikrometern in den Messdaten vorhanden sind. Die Ausreißer können durch Verunreinigungen (z. B. Staub) des Werkzeugeinsatzes beim Messvorgang verursacht worden sein.

Die simulierte LSV der rückgeführten Flächen sowie die dazugehörigen Gradienten-Kurven sind in Abb. 4.6b und Abb. 4.6c dargestellt. Bei beiden Werkzeugeinsätzen ist der Gradient von der rückgeführten Fläche um $0,06$ größer als nominell ausgelegt. Aus beiden Werkzeugeinsätzen wurden Linsen im Spritzguss-Verfahren abgeformt, in einem Scheinwerfermodul verbaut und die erzeugte LSV photometrisch auf einer Bildverarbeitungsanlage vermessen. Die Gradienten-Kurven der

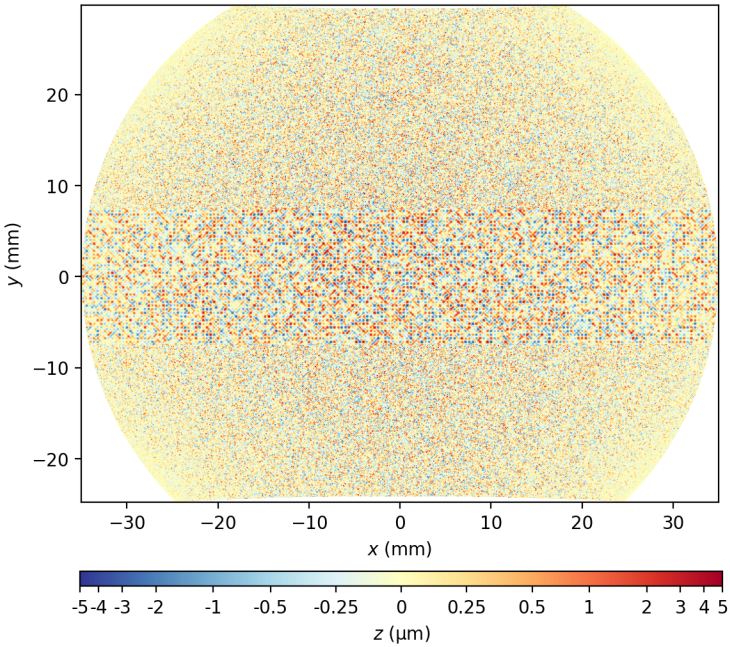
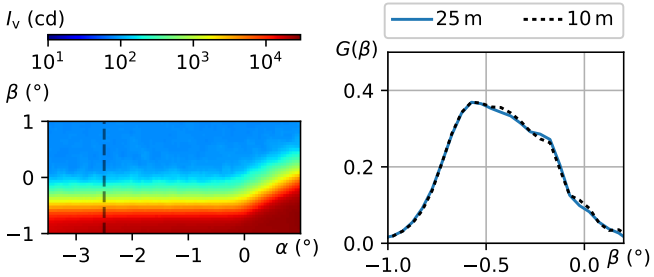
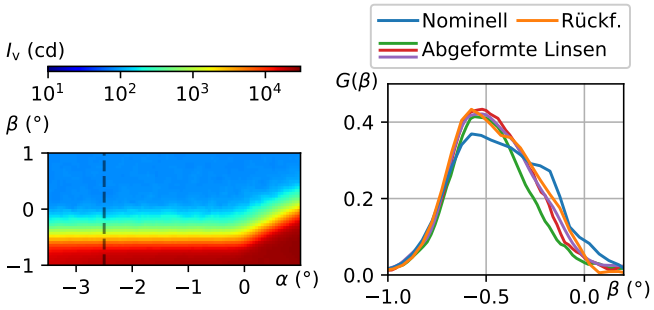


Abbildung 4.5: Nominelle Mikrostruktur zur Aufweichung der HDG für den ECE-Bereich. Die Farbskala ist unter $-0{,}5\,\mu m$ und über $0{,}5\,\mu m$ logarithmisch.

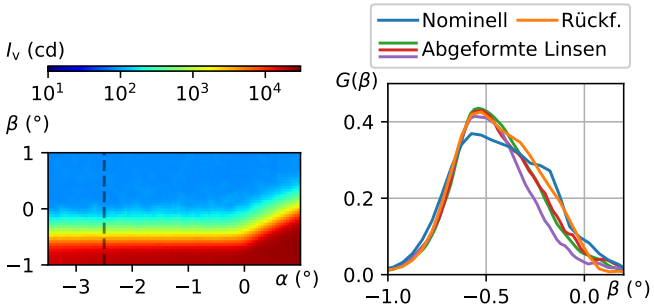
gemessenen LSVen sind in den Abb. 4.6b und Abb. 4.6c dargestellt. Die Gradienten-Kurven der gemessenen LSVen passen sehr gut zu denen der rückgeführten Flächen. Die Abweichung der Gradienten-Kurven zu der nominellen Simulation kann durch eine nicht korrekte Abformung der Struktur beim Spritzgussprozess oder eine zu geringe Strukturhöhe im Werkzeugeinsatz verursacht sein. In Abb. 4.8 werden nominelle und gemessene Strukturhöhe des Werkzeugeinsatzes 1 verglichen. Die Strukturhöhe entspricht in der Nähe des Zenits sehr gut den Vorgaben. In der Nähe der horizontalen Beschnittkanten ist die gemessene Strukturhöhe geringer als vorgegeben. Durch die geringere Strukturhöhe wird



(a) Nominelle Simulation.



(b) Werkzeugeinsatz 1.



(c) Werkzeugeinsatz 2.

Abbildung 4.6: Simulation der LSV und der Gradienten-Kurve des *BiLED*-Moduls mit einer strukturierten Linse ausgelegt für den ECE-Bereich im Vergleich zu gemessenen Gradienten-Kurven von abgeformten Linsen. Die gestrichelte Linie in den LSVen gibt die Position der Gradienten-Kurve an.

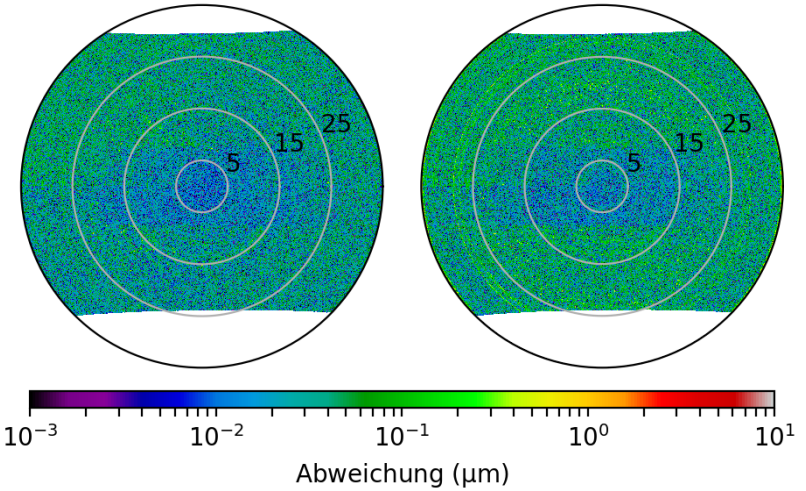


Abbildung 4.7: Abweichung zwischen Flächenrückführung und Messdaten für die zwei Werkzeugeinsätze mit Struktur für ECE-Gebiete. Links: Werkzeugeinsatz Nr. 1, Rechts: Werkzeugeinsatz Nr. 2.

weniger Licht gestreut, sodass der Gradient einen größeren Wert hat. Ähnliche Unterschiede zeigen sich auch bei dem Werkzeugeinsatz 2. Die Abb. 4.9 zeigt ein Foto einer strukturierten Linse.

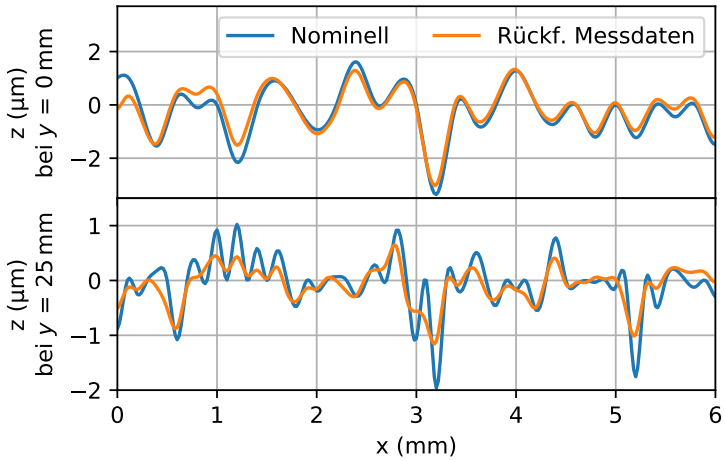


Abbildung 4.8: Vergleich der Strukturhöhen zwischen den nominellen und den gemessenen Daten für Werkzeugeinsatz 1.

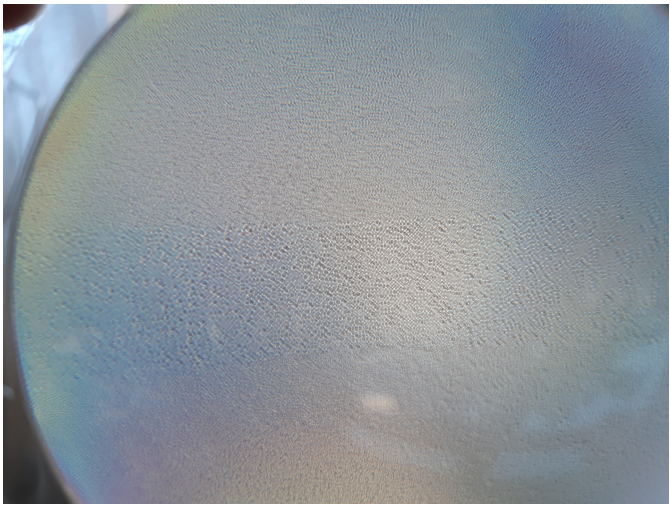


Abbildung 4.9: Foto einer Linse mit der ECE-Struktur.

4.3 FÜR FMVSS-GEBIETE

Die für den FMVSS-Bereich ausgelegte Mikrostruktur ist in Abb. 4.10 dargestellt. Die Struktur basiert auf einem Gitter mit einer Auflösung von 100 µm. Wie bei der ECE-Struktur ist die Strukturierung auf der Linsenfläche inhomogen. Zwei Bereiche der Linse sind nicht strukturiert, um

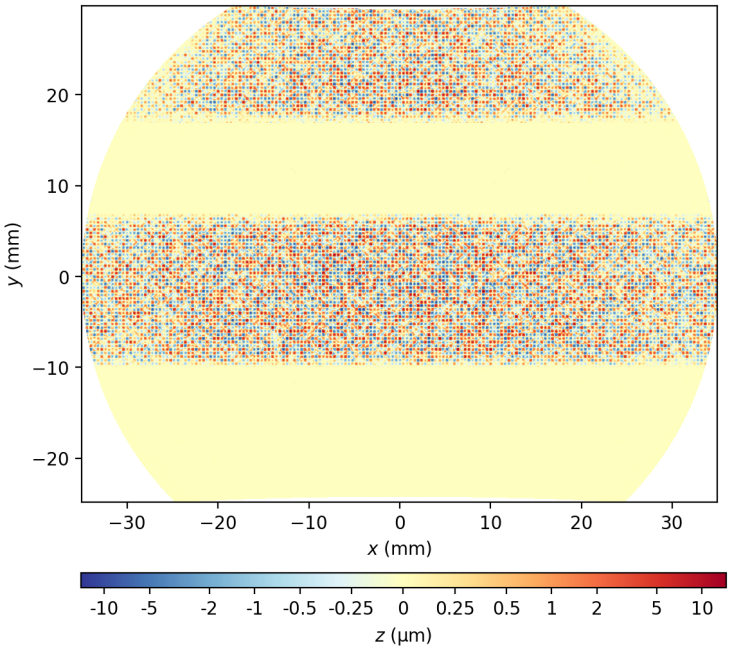
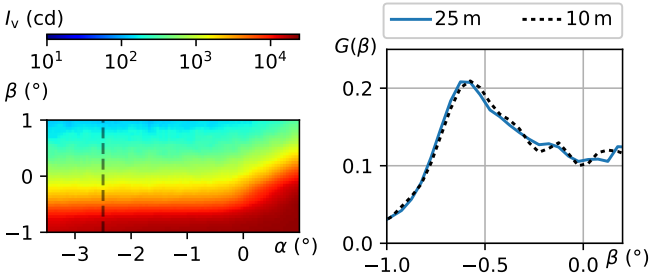


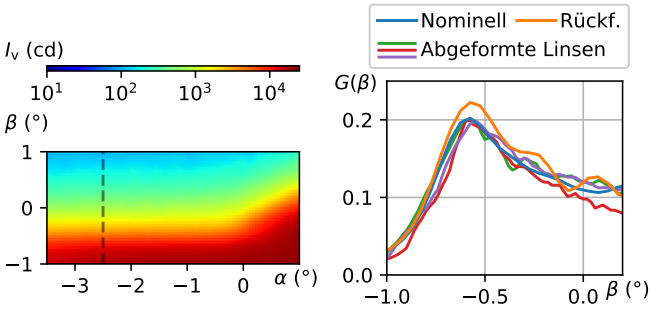
Abbildung 4.10: Nominelle Mikrostruktur zur Aufweichung der HDG für den FMVSS-Bereich. Die Farbskala ist unter $-0{,}5$ µm und über 0,5 µm logarithmisch.

die Beeinflussung des Fernlichtes gering zu halten. In den Bereichen mit Struktur ist die Amplitude der Struktur größer als bei der ECE-Struktur. Die größere Amplitude ist notwendig, um einen geringeren Gradient mit einem Wert von $0,2$ zu erreichen. Die nominelle simulierte LSV und

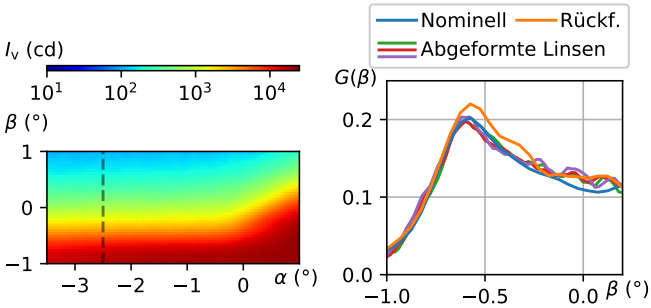
die Gradienten-Kurve sind in Abb. 4.11a dargestellt. Analog zur Herstellung der Linsen mit ECE-Struktur, werden zwei Werkzeugeinsätze (Nr. 3 & 4) gefertigt und vermessen. Das Ergebnis der Flächenrückführung bzw. die Abweichung zwischen Punktgitter und Fläche sind in Abb. 4.12 dargestellt. Für die Werkzeugeinsätze 3 und 4 beträgt die mittlere orthogonale Abweichung zwischen Fläche und Punktgitter 35 nm und 37 nm. Im Bereich des mittleren Streifens der Struktur zeigen sich lokal größerer Abweichungen als bei der ECE-Struktur, wobei die Strukturhöhe selbst größer ist. In Abb. 4.11b und Abb. 4.11c sind die simulierten LSVen und Gradienten-Kurven auf Basis der rückgeführten Flächen dargestellt. In den gleichen Abbildungen sind die Gradienten-Kurven von gemessenen LSVen basierend auf abgeformten Linsen aufgezeigt. Die Gradienten-Kurven der abgeformten Linsen passen sehr gut zu der nominellen Simulation. Die Gradienten-Kurve der Flächenrückführungen weichen mit einem etwas größeren Gradienten ab. In Abb. 4.13 sind die nominellen und gemessenen Strukturhöhen des Werkzeugeinsatzes 3 verglichen. Im Bereich um den Zenit der Linse wird die nominelle Strukturvorgabe gut erfüllt. In dem Bereich nahe der Schnittkante der Linse zeigen sich starke Abweichungen, bzw. eine Korrelation zwischen Soll- und Ist-Struktur ist kaum noch feststellbar. Da die Strukturhöhe am Randbereich um ca. einen Faktor 10 geringer ist als im Inneren, sind die Auswirkungen der Abweichung auf die Gradienten-Kurve gering. Die abgeformten Linsen zeigen einen Einfall in der Lichteintrittsseite der Linse, visualisiert in Abb. 4.14. Der Einfall hat eine defokussierende Wirkung auf das Licht, wodurch der Gradient der HDG verringert werden kann. Durch eine zu geringe Strukturhöhe, wird der Gradient größer, wie in den simulierten Gradienten-Kurven der rückgeführten Flächen dargestellt. Der Einfall der Lichteintrittsseite kann der zu geringen Strukturhöhe durch die Defokussierung entgegenwirken, weshalb die Gradienten-Kurven der abgeformten Linsen mit denen der nominellen Simulation übereinstimmen.



(a) Nominelle Simulation.



(b) Werkzeugeinsatz 3.



(c) Werkzeugeinsatz 4.

Abbildung 4.11: Simulation der LSV und der Gradienten-Kurve des *BiLED*-Moduls mit einer strukturierten Linse, ausgelegt für den FMVSS-Bereich im Vergleich zu gemessenen Gradienten-Kurven von abgeformten Linsen. Die gestrichelte Linie in den LSVen gibt die Position der Gradienten-Kurve an.

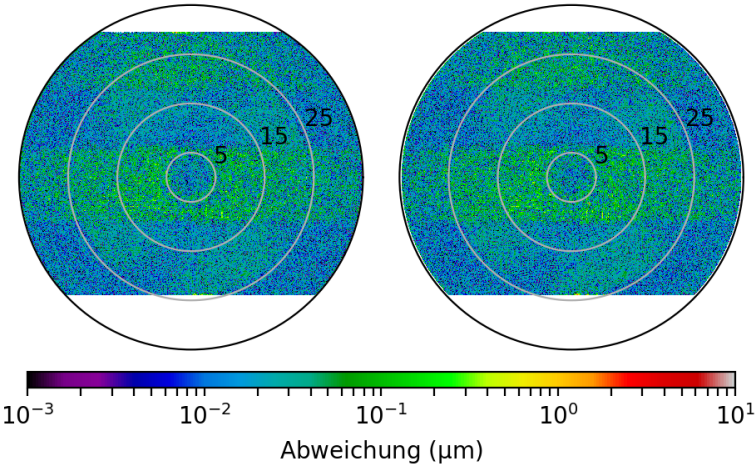


Abbildung 4.12: Abweichung zwischen Flächenrückführung und Messdaten für die zwei Werkzeugeinsätze mit Struktur für FMVSS-Gebiete. Links: Werkzeugeinsatz Nr. 3, Rechts: Werkzeugeinsatz Nr. 4.

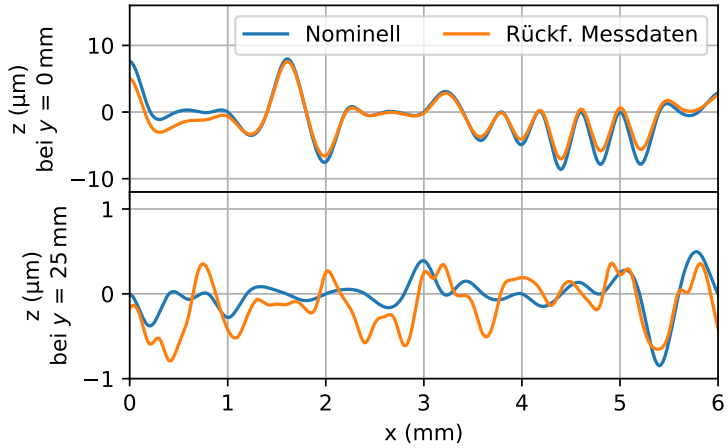


Abbildung 4.13: Vergleich der Strukturhöhen zwischen den nominellen und den gemessenen Daten für Werkzeugeinsatz 3.

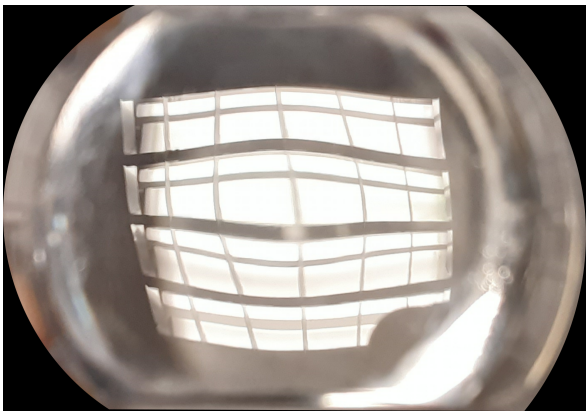


Abbildung 4.14: Foto einer Fresnel-Reflexion einer Deckenleuchte an der Lichteintrittsseite der Linse. Die Verzerrung der rechteckigen Leuchte entsteht durch den Einfall der nominell planaren Fläche.

KAPITEL 5

OPTIMIERUNG DER AUSLEGUNG DER MIKROSTRUKTUREN

Die Auslegung der Strukturen bisher, einschließlich denen aus Kapitel 4, erfolgt in manuellen Korrekturschleifen. Ein Optikingenieur erstellt eine Struktur, führt eine lichttechnische Simulation durch und bewertet die Gradienten-Kurve. Je nach resultierender Gradienten-Kurve wird z. B. die Amplitude der Struktur vergrößert oder verringert. Um eine Struktur in angemessener Zeit auszulegen, wird viel Erfahrung benötigt. Dieses Kapitel befasst sich mit der Reduzeriung der Simulationszeit von Scheinwefersystemen mit strukturierten Linsen und der Auslegung von solchen Strukturen durch Anwendung von Optimierungsalgorithmen. Der Inhalt dieses Kapitels stammt zu großen Teilen aus der Veröffentlichung [93].

5.1 REDUZIERUNG DER SIMULATIONSSZEIT

Um eine auswertbare, also nicht zu verrauschte Gradienten-Kurve zu erhalten, muss mit mehreren hundert Millionen Strahlen simuliert werden. Eine Simulation mit 500 Millionen Strahlen des *BiLED*-Moduls (vgl. Abb. 4.3) dauerte auf dem Test-Computer mit einem *Intel Xeon E5-1650 v3* ca. eine Stunde [56]. Für einen Optimierungsprozess, bei dem ständig neue Strukturen generiert und simuliert werden, ist eine Stunde

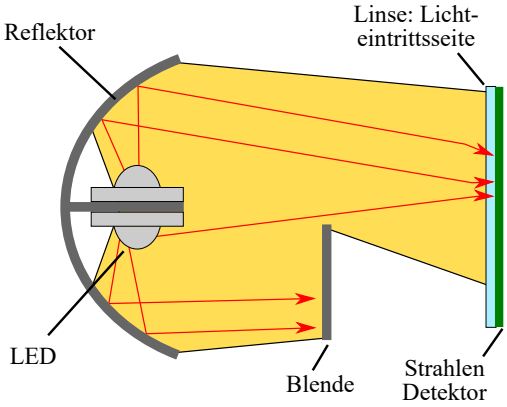


Abbildung 5.1: Schematische Darstellung des Aufbaus zur Vorbereitung des Optimierungsprozesses

zu lang. Zum Beispiel würde die Simulation von 1000 verschiedenen Strukturen ca. 40 Tage dauern. Zur Reduzierung der Simulationszeit wird der Aufbau aus Abb. 5.1 verwendet. Der Aufbau entspricht dem kompletten Scheinwerfermodul, ohne die Lichtaustrittsfläche der Linse. Ein Strahlendetektor ist hinter der Lichteintrittsseite der Linse platziert. Der Detektor speichert die Informationen der Strahlen die ihn treffen. Zu den Informationen gehören Schnittpunkt, Richtungsvektor, Lichtstrom und Wellenlänge. Die abgespeicherten Strahlen werden als Lichtquelle für den zweiten Schritt der Vorbereitung verwendet. Die Strahlen-Lichtquelle modelliert das Scheinwerfermodul ohne Lichtaustrittsseite der Linse.

Bei der Simulation des *BiLED*-Moduls mit 500 Millionen Strahlen erreichen ca. 400 Millionen Strahlen den Detektor. Wird die Strahlen-Lichtquelle zur Simulation des Moduls verwendet und nur der Einfluss der Lichtaustrittsseite simuliert, so reduziert sich die Simulationszeit mit den 400 Millionen Strahlen auf ca. 10 Minuten. Im zweiten Schritt werden die für die HDG relevanten Strahlen herausgefiltert. Dazu wird eine

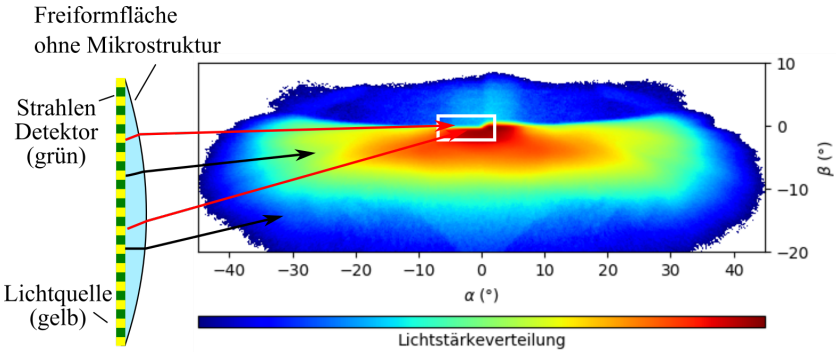


Abbildung 5.2: Filterung der Strahlen: Es werden nur die Strahlen behalten, die in einen Bereich in der Nähe der HDG treffen (weißes Rechteck).

Simulation mit der unstrukturierten Linsenfläche durchgeführt. Es wird die Linsenfläche verwendet, die die gleiche Kontrollpunktdichte hat wie die später erstellte strukturierte Fläche (vgl. Kap 3.2.1 „zweites Verfahren“). Dadurch haben unstrukturierte und strukturierte Linsenflächen eine ähnliche Parametrisierung. Werden die Strahlen durch die unstrukturierte Linsenfläche in einen vorher definierten Bereich in der Nähe der HDG gebrochen (vgl. Abb. 5.2), so werden die Strahl-Informationen und die *uv*-Koordinaten der Schnittpunkte gespeichert. Das Ergebnis der Filterung ist eine Strahlen-Lichtquelle, die nur die Strahlen enthält, die die HDG erzeugen bzw. die Gradienten-Kurve beeinflussen. Ein Versuch zeigt, dass nach der Filterung ca. 10 Millionen Strahlen übrig bleiben. Die Simulation des *BiLED*-Moduls mit der gefilterten Strahlen-Lichtquelle benötigt ca. 18 s.

Bei einer Standard-Simulation wird zu jeder Zeit die Information gespeichert, in welchem Material sich der jeweilige Strahl befindet. Durch numerische Ungenauigkeiten kann es passieren, dass ein Strahl, der sich in einem Material A befindet auf eine Fläche trifft, die die Materialien B und C voneinander trennt. Dieser Fall stellt einen Simulationsfehler da,

sodass die weitere Verfolgung des Strahls verworfen wird. Da nur eine optische Fläche in dem Simulationsaufbau während der Optimierung vorhanden ist, kann auf die Überprüfung verzichtet werden. Um die Schnittpunkte bei einer Simulation schnell berechnen zu können, wird vor dem Ray-Tracing eine Hierarchie von Begrenzungsvolumen aufgebaut (vgl. Kap. 2.1.2). Die Berechnung der Volumen dauern für eine strukturierte Fläche einige Sekunden. Wie oben beschrieben, werden bei der Filterung die *uv*-Koordinaten der Schnittpunkte gespeichert. Anstatt die Begrenzungsrahmen zu verwenden, wird bei der Optimierung direkt auf der Fläche mit dem Newton-Verfahren (vgl. Kap. 2.1.2) nach den Schnittpunkten gesucht. Die *uv*-Koordinaten werden als Startpunkte für die Iteration verwendet.

Bei dem Newton-Verfahren müssen die B-Spline-Flächen und deren Ableitungen ausgewertet werden. Während der Optimierung werden dafür die SIMD-Implementierungen (vgl. Kap. 2.2.4) verwendet. Durch die Verwendung des „zweiten Verfahrens “ aus Kap. 3.2.1 zur Strukturierung der Flächen reduziert sich die Strukturierung auf eine Verschiebung der Kontrollpunkte. Es wird kein Gleichungssystem gelöst, wodurch Zeit eingespart wird. Bei der Optimierung müssen kontinuierlich Strukturen erstellt, simuliert und die Gradienten-Kurven bewertet werden. Diese drei Schritte werden zwölf-fach parallel ausgeführt, sodass auf jedem logischen Prozessorkern je eine Struktur bewertet wird. Durch die genannten Maßnahmen können, unter Verwendung einer Strahlen-Lichtquelle mit 10 Millionen Strahlen, zwölf Strukturen in 24 s erstellt und bewertet werden. Das entspricht umgerechnet 2 s pro Struktur.

5.2 GENETISCHE ALGORITHMEN

Um eine Optimierung durch Algorithmen zu ermöglichen, muss definiert werden, was eine gute von einer schlechten Mikrostruktur un-

terscheidet. Dazu wird eine Soll-Gradienten-Kurve definiert. In dieser Arbeit werden die simulierten Gradienten-Kurven der Strukturen, ausgelegt durch Optikingenieure, als Soll-Kurven verwendet. Als Qualitätsmaß wird die quadratische Abweichung zwischen der vorgegebenen und den simulierten Gradienten-Kurven verwendet:

$$\Phi = -\sum_{i=1}^{N} \left(G_{\text{Ist}}(\beta_i) - G_{\text{Soll}}(\beta_i)\right)^2 \tag{5.1}$$

Da das Ergebnis einer Simulation eine diskretisierte Gradienten-Kurve ist, gibt N die Anzahl an Stützstellen an. Im Zusammenhang mit genetischen Algorithmen beschreibt Gl. (5.1) die Qualitäts- bzw. Fitness-Funktion, die überlicherweise maximiert wird. Daher wird als Qualitätsmaß das Negative der Summe aus Gl. (5.1) verwendet. Ziel der Optimierung ist es, eine Struktur zu finden, die Φ maximiert. Die ideale Struktur würde $\Phi = 0$ erfüllen. Je nach Anzahl an Parametern, die bei der Optimierung verwendet werden, sind auf Grund der stochastischen Strukturen, lokale Minima in der Funktion Φ zu erwarten. Die Ableitung der Qualitätsfunktion (5.1) nach den Parametern der Mikrostruktur (z. B. Gitterabstand oder Amplitude) ist schwierig zu bestimmen. Daher wird ein Algorithmus benötigt, der global nach Minima sucht und unabhängig von den Ableitungen der Qualitätsfunktion ist. Sogenannte genetische Algorithmen erfüllen diese Kriterien [94, 95].

Die genetischen Algorithmen basieren auf Phänomenen, wie sie in der Natur vorkommen. Dazu gehören die natürliche Selektion und die genetische Evolution. In Abb. 5.3 ist schematisch der Ablauf eines genetischen Algorithmus dargestellt. Zu Beginn wird eine Sammlung (Population) von verschiedenen mikrostrukturierten Linsenflächen (Individuen) bestimmt, deren Parameter (Gene) zufällig gewählt werden. Es wird der Fitness-Wert Φ jedes Individuums bestimmt. Für die Mikrostrukturen wird dazu die Gl. (5.1) ausgewertet. Die Fitness wird so interpretiert, dass ein größeres Φ eine bessere Fitness darstellt. Wird ein Abbruch-

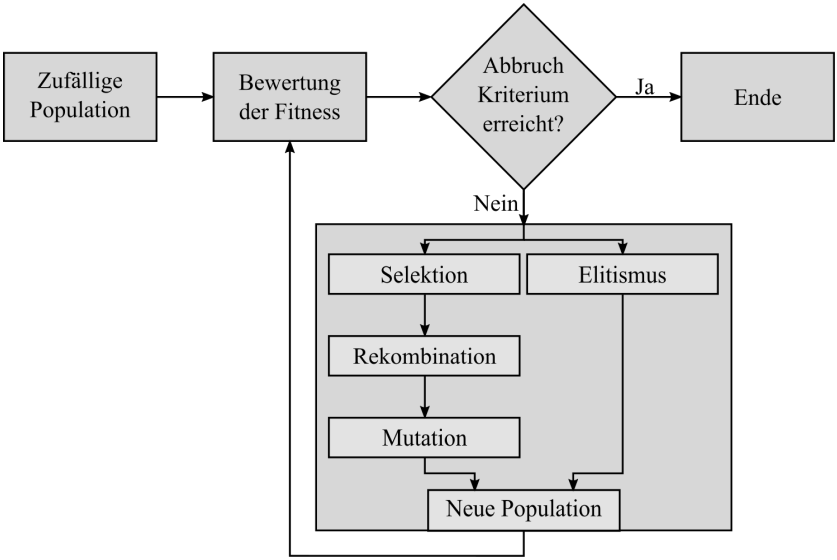


Abbildung 5.3: Schematische Darstellung des Ablaufes eines genetischen Algorithmus.

kriterium erfüllt, so endet der Algorithmus. Abbruchkriterien können eine maximale Anzahl an Iterationen (Generationen) oder ein erreichter Fitness-Wert sein.

Ist kein Abbruchkriterium erfüllt, so wird eine neue Population bestimmt. Zur Bestimmung der neuen Population werden die Individuen, ihrem Fitness-Wert entsprechend, sortiert. Eine vorher definierte Anzahl an Individuen mit den besten Fitness-Werten werden unverändert in die neue Population übernommen (sog. Elitismus). Die restlichen neuen Individuen werden auf Basis der bestehenden Population bestimmt. Für die sog. Turnier-Auswahl (Selektion) werden zwei Individuen zufällig aus der bestehenden Population, einschließlich der bereits übernommenen Elite, gewählt [96]. Die Fitness beider Individuen wird verglichen. Das Individuum mit der besseren Fitness „gewinnt“ das Turnier, wird aus der alten Population entfernt und wird für die Rekombination ver-

wendet. Über eine zweite Turnier-Auswahl wird ein weiteres Individuum für die Rekombination bestimmt. Bei der Rekombination werden zwei neue Individuen (Nachkommen) aus den zwei zuvor gewählten Individuen bestimmt. Hier wird die sog. „Intermediate Recombination“ angewendet [96]. Bei dieser Methode werden die neuen Parameter durch gewichtete Linearkombination der alten Parameter bestimmt. Seien $\vec{\nu} = \{\nu_0, \nu_1, \ldots, \nu_l\}$ und $\vec{\omega} = \{\omega_0, \omega_1, \ldots, \omega_l\}$ die Parameter der beiden gewählten Mikrostrukturen (Individuen). Bei der Rekombination werden die neuen Parameter wie folgt bestimmt:

Programmauflistung 5.1: „Intermediate Recombination“ [96]

```
1  for i ← 0, l
2    do
3      a ← randomValue(−p, 1 + p)
4      b ← randomValue(−p, 1 + p)
5      t ← aν_i + (1 − a)ω_i
6      s ← bω_i + (1 − b)ν_i
7    while outsideBounds(t, s)
8    v_i ← t
9    w_i ← s
10 end for
```

Es werden zwei Zufallszahlen zwischen $-p$ und $1 + p$ bestimmt. Durch lineare Kombination werden die neuen Parameter t und s bestimmt. Sind die Parameter außerhalb eventuell gesetzter Schranken, wird der Vorgang wiederholt, bis die neuen Parameter innerhalb der Schranken liegen. Wird $p = 0$ verwendet, werden die neuen Parameter nur innerhalb des Intervalls $[\nu_i, \omega_i]$ der alten Parameter bestimmt. Dadurch kann eine vorzeitige Konvergenz in Bereichen lokaler Minima stattfinden. Um das zu verhindern, wird $p = 0,5$ verwendet, sodass die neuen Parameter auch außerhalb des Intervalls $[\nu_i, \omega_i]$ gewählt werden können.

Die Parameter der neuen Mikrostrukturen (Nachkommen) werden mutiert. Durch Mutation wird eine vorzeitige Konvergenz des Algorithmus zusätzich verhindert und die Suche nach globalen Minima ermöglicht. Mutation bedeutet, dass zufällig ausgewählte Parameter eines Individuums um zufällig gewählte Werte verschoben werden [97]. Die Mutation die hier verwendet wird, wählt von jeder Mikrostruktur einen Parameter aus und verschiebt diesen durch

$$\omega_i \leftarrow \omega_i + s \cdot r \cdot (\omega_{\max,i} - \omega_{\min,i}) \,. \tag{5.2}$$

Die Symbole $\omega_{\max,i}$ und $\omega_{\min,i}$ geben die Schranken für den Parameter ω_i an. Das Symbol r gibt den Mutationsbereich an. Es wird $r = 0,1$ verwendet [97]. Das Vorzeichen des Versatzes ist durch s gegeben, wobei $s \in \{-1, 1\}$ zufällig bestimmt wird. Sollte das neue ω_i außerhalb der Schranken liegen, so wird ω_i auf den Wert der jeweiligen Schranke gesetzt. Die Schritte Selektion, Rekombination und Mutation werden wiederholt, bis die neue Population mit Individuen gefüllt ist. Es werden die Fitness-Werte der neuen Population bestimmt und das Abbruch-Kriterium überprüft. Die Iteration wird durchgeführt, bis das Abbruch-Kriterium erreicht ist.

5.3 STOCHASTISCHE STRUKTUREN

5.3.1 BILED-MODUL

Zum Testen der Optimierung wird das *BiLED*-Modul verwendet. Es wird die erste Methode aus Kap. 3.2.1 zur Strukturierung angewendet. Die Mikrostruktur basiert auf einem Punktgitter mit einem Abstand von 100 µm. Die Amplituden werden aus Zellen der Größe 2x2 gezogen (vgl. Kap. 3.2.3). Die Einträge der Hauptdiagonalen sind die Parameter, die

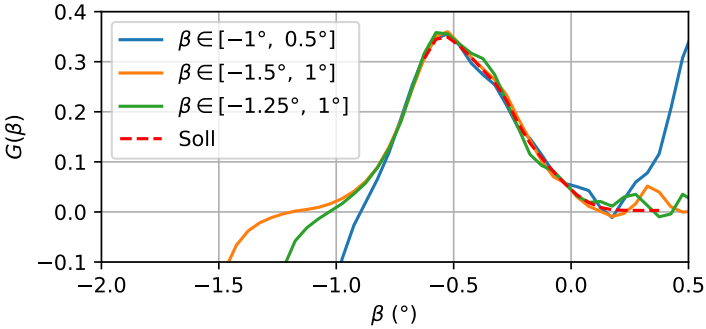


Abbildung 5.4: Vergleich zwischen der Soll- und den resultierenden Gradienten-Kurven für die drei Optimierungen mit unterschiedlich gefilterten Strahlen-Lichtquellen.

der Algorithmus optimieren soll. Die Einträge der Nebendiagonalen werden auf null gesetzt, also:

$$\sigma = \begin{pmatrix} \sigma_{00} & 0 \\ 0 & \sigma_{11} \end{pmatrix} \tag{5.3}$$

Bei der Optimierung beträgt die Populationsgröße 36, wobei eine Elite, die 12 Individuen umfasste, verwendet wird. Die Soll-Gradienten-Kurve ist in Abb. 5.4 dargestellt und entspricht der simulierten Gradienten-Kurve des *BiLED*-Moduls mit ECE-Struktur. Es werden drei Optimierungen durchgeführt, um die Auswirkungen des Filterbereiches (vgl. Kap. 5.1 und Abb. 5.2) und der Strahlanzahl zu untersuchen. Die Filterbereiche in vertikaler Richtung sind $[-1°,\ 0,5°]$ mit 6 Millionen Strahlen, $[-1,25°,\ 1°]$ mit 8 Millionen Strahlen und $[-1,5°,\ 1.5°]$ mit 10 Millionen Strahlen. Der horizontale Bereich ist $[-5°,\ 1°]$. Die Auflösung des Detektors, der in der Simulation verwendet wird, beträgt 0,05° vertikal und 0,25° horizontal. Die Auflösung in horizontaler Richtung wird niedrig gewählt, um das Simulationsrauschen zu verringern. Das ist möglich, da benachbarte horizontale Zellen in der Nähe der HDG einen sehr

ähnlichen Wert haben. Die nach 20 Generationen erzielten Gradienten-Kurven der jeweiligen Strukturen mit der maximalen Fitness sind in Abb. 5.4 dargestellt. Alle drei Kurven passen gut zur Zielvorgabe. Die Abweichung der blauen Kurve in dem Bereich $\beta > 0°$ entsteht durch die gefilterte Strahlen-Lichtquelle. Bei der blauen Kurve werden nur die Strahlen verwendet, die auf einen Bereich unterhalb von 0,5° auftreffen. Da die Struktur das Licht auch in negativer Richtung ablenkt, kann die Gradienten-Kurve in diesem Bereich nicht korrekt simuliert werden. In Abb. 5.5 ist die maximale und die durchschnittliche Fitness pro Generation aufgetragen. Die Kurven der maximalen Fitness erreichten einen

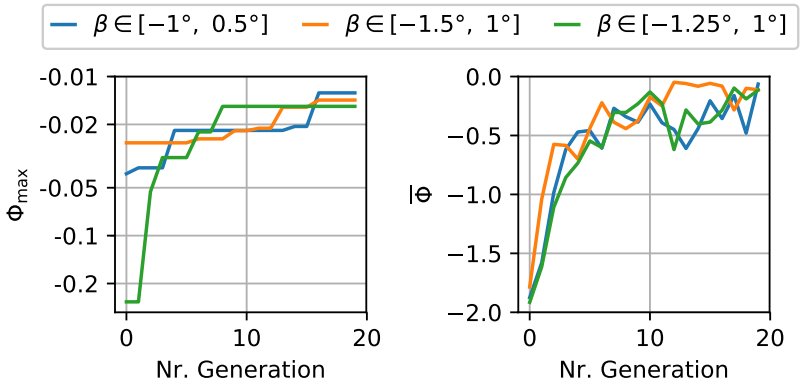


Abbildung 5.5: Konvergenzverhalten des genetischen Algorithmus bei der Optimierung von stochastischen Strukturen für das *BiLED*-Modul. Links: Maximale Fitness pro Generation. Rechts: Durchschnittliche Fitness pro Generation.

Wert zwischen $-0,014$ bis $-0,012$. Die Kurven der durchschnittlichen Fitness erreichten einen Wert von ca. $-0,1$. Die Gradienten-Kurven sind durch die Monte Carlo-Simulation immer verrauscht, weshalb eine exakte Anpassung sehr unwahrscheinlich ist. Der Unterschied zwischen maximaler und durchschnittlicher Fitness bedeutet, dass die Population noch nicht komplett konvergiert ist. Da die Gradienten-Kurven jedoch

sehr gut zu den Vorgaben passen, hätten weitere Iterationen keine merkliche Verbesserung mehr gebracht. Die simulierten LSVen sind in der Abb. 5.6 dargestellt. Die Farbskala zu den LSVen ist in willkürlichen

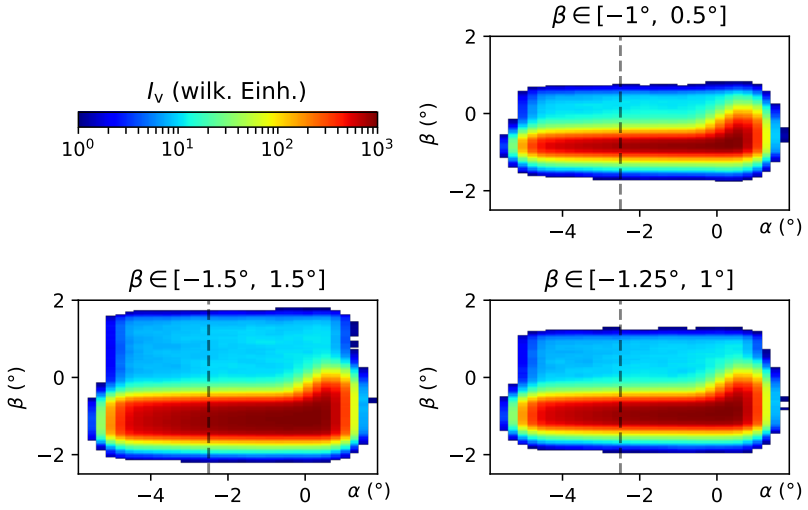


Abbildung 5.6: Simulierte LSVen für das *BiLED*-Modul mit jeweils der stochastischen Struktur mit der maximalen Fitness. Die gestrichelte Linie zeigt die Position, an der die Gradienten-Kurve bestimmt wurde.

Einheiten angegeben, da bei der Generierung der gefilterten Strahlen-Lichtquelle ein Skalierungsfaktor verloren geht. Da die Berechnung der Gradienten-Kurve invariant bezüglich der Skalierung der LSV ist, werden keine absoluten Lichtstärkewerte benötigt. Die Parameter der

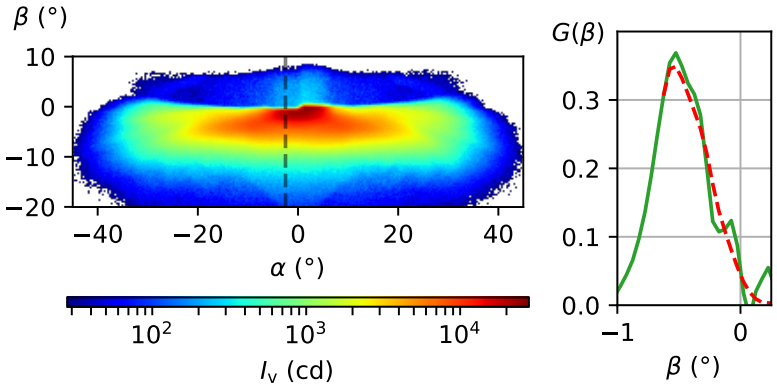


Abbildung 5.7: Simulation des kompletten *BiLED*-Moduls mit einer Mikrostruktur erstellt mit den Parametern $\sigma_{[-1.25,1]}$. Die gestrichelte Linie markiert die Position zur Auswertung der Gradienten-Kurve. Rechts: Vergleich zwischen Soll- und Ist-Gradienten-Kurve.

Strukturen mit dem besten Fitness-Wert sind:

$$\sigma_{[-1,0.5]} = \begin{pmatrix} 0,67 & 0 \\ 0 & 0,12 \end{pmatrix} \mu m \quad \sigma_{[-1.5,1]} = \begin{pmatrix} 0,66 & 0 \\ 0 & 0,19 \end{pmatrix} \mu m$$

$$\sigma_{[-1.25,1]} = \begin{pmatrix} 0,66 & 0 \\ 0 & 0,16 \end{pmatrix} \mu m$$

Die Parameter der Strukturen sind nahezu identisch. Da die Optimierung nur auf zwei Parameter basiert, war eine Ähnlichkeit der Ergebnisse zu erwarten. Bei zwei Parametern ist die Dimension des Parameterraums so klein, dass die Wahrscheinlichkeit, das der Optimierungsalgorithmus bei unterschiedlichen Läufen unterschiedliche Minima findet, sehr gering ist. Die Simulation der Mikrostruktur mit dem kompletten Scheinwerfer-Modul ist in Abb. 5.7 dargestellt. Die Simulation des kompletten Moduls reproduziert sehr gut die Gradienten-Kurve der Optimierung. Dieser Versuch zeigt, dass Mikrostrukturen durch Optimierungsalgorithmen ausgelegt werden können. Bei der Aus-

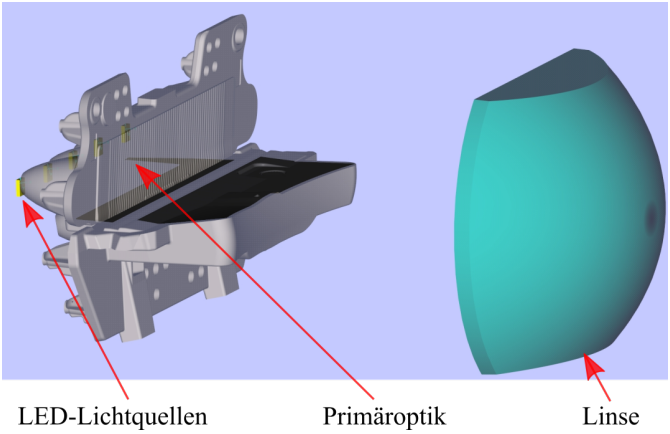


Abbildung 5.8: Aufbau des *MLS-BiLED*-Scheinwerfermoduls.

wahl des Filterbereiches zur Generierung der Strahlen-Lichtquelle muss darauf geachtet werden, dass der Bereich um ca. 0,5° größer ist als der Bereich der Soll-Gradienten-Kurve, wie Abb. 5.4 zeigt. Bei einer flacheren Soll-Gradienten-Kurve müsste der Bereich eventuell größer gewählt werden, da bei einer flacheren Kurve mehr Streuung benötigt wird.

5.3.2 MLS-BiLED-Modul

Für einen zweiten Test wird das *MLS-BiLED*-Scheinwerfermodul verwendet. Die Abb. 5.8 zeigt den Aufbau des Moduls. Für das Modul wurde eine Mikrostruktur von einem Optikingenieur unter Anwendung der alten Prozesskette (vgl. Abb. 1.3) ausgelegt. Die simulierte LSV und Gradienten-Kurve ist in Abb. 5.9 dargestellt. Durch den genetischen Algorithmus wird versucht, eine Mikrostruktur zu finden. Die Mikrostruktur soll eine Gradienten-Kurve erzeugen, die der Nominellen, vom Optikingenieur ausgelegten, möglichst ähnlich ist. Dazu wird

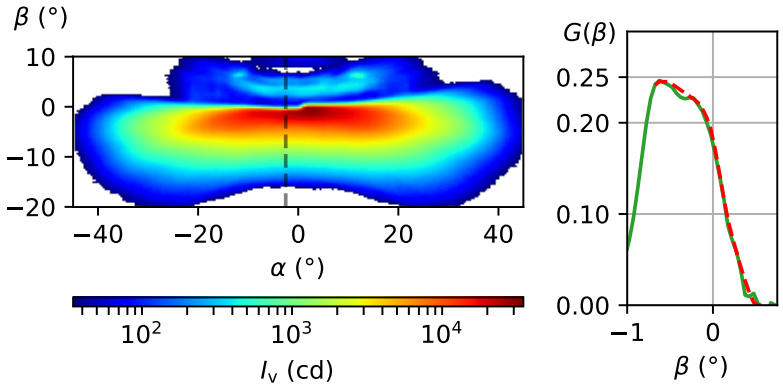


Abbildung 5.9: Simulation des kompletten *MLS-BiLED*-Moduls mit einer Mikrostruktur ausgelegt durch einen Optikingenieur. Rechts: Vergleich zwischen der simulierten (grün) und der für die Optimierung verwendeten Soll-Gradienten-Kurve (rot).

eine Soll-Gradienten-Kurve definiert (rote Kurve in Abb. 5.9), die der nominellen geglätteten Kurve entspricht. Im Vergleich zu der Struktur für das *BiLED*-Modul ist der Gradient geringer und die Gradienten-Kurve fällt erst bei höheren Winkeln auf Null ab. Durch die flachere Kurve wirkt sich das Simulationsrauschen stärker aus. Daher werden zwei Optimierungen durchgeführt. Bei der ersten Optimierung werden 10 Millionen Strahlen in dem Bereich $[-1.5°,\ 1.5°]$ und bei der zweiten Optimierung 40 Millionen Strahlen in dem Bereich $[-2.5°,\ 2.5°]$ verwendet. Die Größe der Population beträgt 36 Individuen, die Größe der Elite beträgt 12 Individuen. Die erzielten Gradienten-Kurven sind in Abb. 5.10 dargestellt. Beide Kurven passen gut zur Zielvorgabe. Die blaue Kurve, basierend auf der Simulation mit 10 Millionen Strahlen, zeigt ein größeres Simulationsrauschen auf Grund der geringeren Strahlanzahl. Das Rauschen zeigt sich auch in einer etwas schlechteren Fitness (vgl. Abb. 5.11). Wie bei dem *BiLED*-Modul, ist die erreichte durchschnittliche Fitness der Population um ungefähr einen Faktor 10 schlechter als die

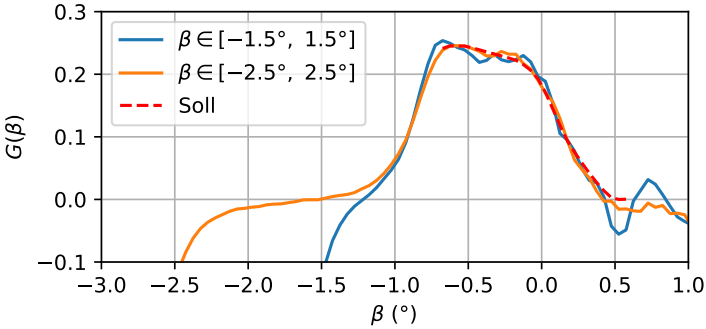


Abbildung 5.10: Vergleich zwischen der Soll- und den resultierenden Gradienten-Kurven für die zwei Optimierungen mit unterschiedlich gefilterten Strahlen-Lichtquellen.

maximale Fitness. Weitere Iterationen hätten keine merkliche Verbesserung gebracht, da Soll- und Ergebniskurve sehr ähnlich sind. In den Abb. 5.12 sind die simulierten LSVen dargestellt.

Die Parameter der Strukturen sind:

$$\sigma_{[-1.5,\,1.5]} = \begin{pmatrix} 1,04 & 0 \\ 0 & 0,19 \end{pmatrix} \mu m \quad \sigma_{[-2.5,\,2.5]} = \begin{pmatrix} 1,03 & 0 \\ 0 & 0,32 \end{pmatrix} \mu m$$

Die Simulation des kompletten Moduls mit der Struktur mit den Parametern $\sigma_{[-2.5,\,2.5]}$ ist in Abb. 5.13 dargestellt. Die Gradienten-Kurve der simulierten LSV ist stärker verrauscht als die der Optimierungen, da hier eine horizontale Detektorauflösung von 0,05° verwendet wird. Trotz des Rauschens ist erkennbar, dass simulierte und vorgegebene Gradienten-Kurven gut übereinstimmen.

Die Auslegung der Strukturen für das *MLS-BiLED*-Modul bestätigen, dass eine Auslegung durch Optimierungsalgorithmen möglich ist. Beide Versuche verwendeten zwei Optimierungsparameter. Durch die geringe Anzahl an Parametern und die gesetzten Schranken wäre der Einsatz

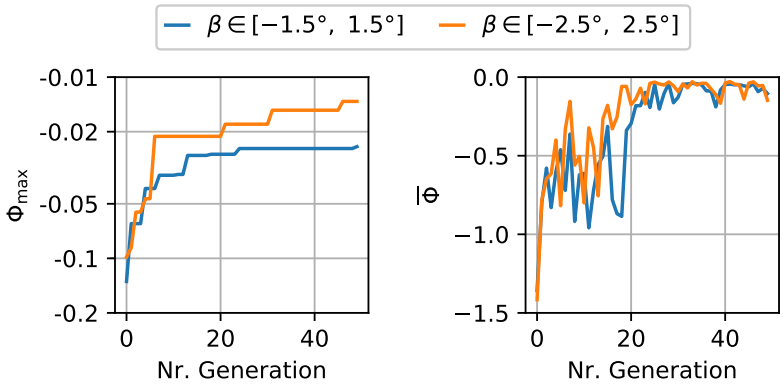


Abbildung 5.11: Konvergenzverhalten des genetischen Algorithmus bei der Optimierung von stochastischen Strukturen für das *MLS-BiLED*-Modul. Links: Maximale Fitness pro Generation. Rechts: Durchschnittliche Fitness pro Generation.

eines genetischen Algorithmus eventuell nicht nötig gewesen. Vermutlich hätten durch ein lineares Absuchen des Parameterraums geeignete Strukturen gefunden werden können. Die Parameter, die zur Optimierung verwendet werden, stellen eine Art Streumaß dar. Je größer die Parameter gewählt werden, umso stärker wird die Flächennormale der Linse um ihre ursprüngliche Ausrichtung verkippt. Es findet eine zufällige Streuung des Lichtes statt, wodurch der Gradient verringert wird. In weiteren Versuchen wird nach Strukturen gesucht, die keine zufällige Verkippung der Flächennormalen haben, sondern bei denen die Verkippung direkter der Soll-Gradienten-Kurve angepasst werden.

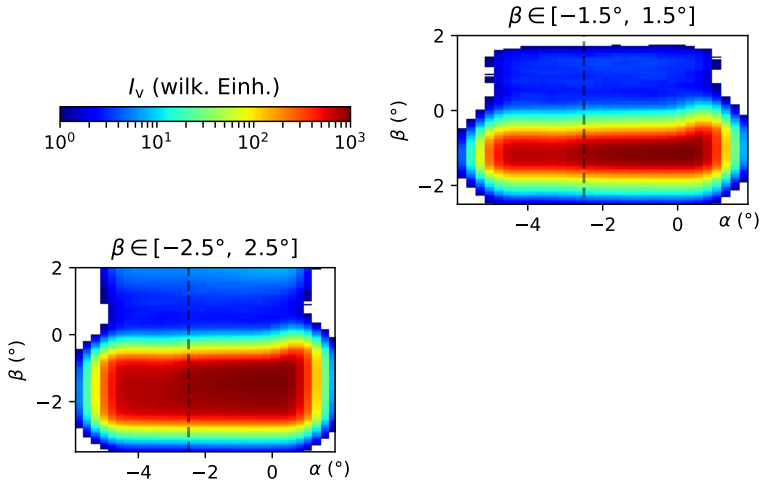


Abbildung 5.12: Simulierte LSVen für das *MLS-BiLED*-Modul mit jeweils der stochastischen Struktur mit der maximalen Fitness. Die gestrichelte Linie zeigt die Position, an der die Gradienten-Kurve bestimmt wurde.

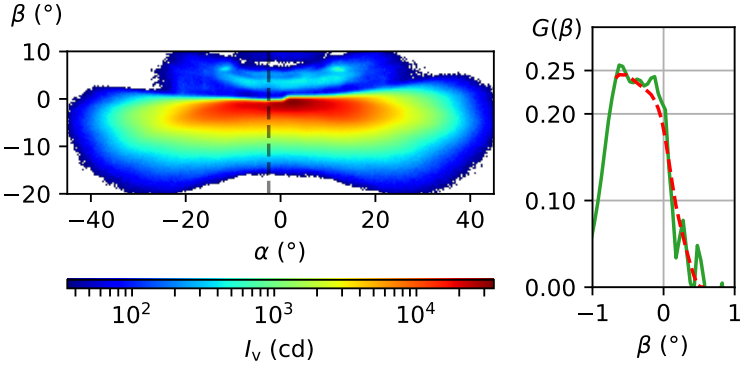


Abbildung 5.13: Simulation des kompletten *MLS-BiLED*-Moduls mit einer Mikrostruktur erstellt mit den Parametern $\sigma_{[-2.5°,\ 2.5°]}$. Rechts: Vergleich zwischen Ziel- und Ist-Gradienten-Kurve.

5.4 PERIODISCHE FREIFORM-STRUKTUREN

Bei den folgenden Versuchen wird die zweite Methode (Verschiebung der Kontrollpunkte, vgl. Kap. 3.2.1) angewendet, um für das *BiLED*-Modul geeignete nicht-stochastische Strukturen zu finden. Anstatt das Punktgitter in Zellen einzuteilen, wird die Kontrollpunktmatrix der zu strukturierenden Fläche eingeteilt. Bei einem Versuch beträgt die Größe der Zellen 1x10 (horizontal x vertikal), bei einem zweitem Versuch 2x6. Die Parameter, die dem Optimierungsalgorithmus zur Verfügung stehen, sind Verschiebungen der Kontrollpunkte in vertikaler (y) und longitudinaler (z) Richtung. Die Verschiebung in y-Richtung gibt die Möglichkeit, eine Struktur mit steileren Flanken (größere Ablenkung) zu generieren, ohne die Amplitude der Strukturen vergrößern zu müssen. Es wird eine Fläche verwendet, deren Kontrollpunkte einen Abstand von 200 µm in vertikaler und horizontaler Richtung haben. Der Optimierungsalgorithmus darf die Kontrollpunkte in vertikaler Richtung um ± 95 µm und in longitudinaler Richtung um ± 1 µm verschieben. Es wird eine Strahlenlichtquelle mit 10 Millionen Strahlen aus dem Bereich $[-3°,\ 1°]$ horizontal und $[-5°,\ 1°]$ vertikal verwendet. Die Größe der Population beträgt 48 und die Größe der Elite 24. Die Gradienten-Kurven der jeweiligen Mikrostruktur mit dem höchsten Fitness-Wert sind in Abb. 5.14 dargestellt. Die vorgegebene Gradienten-Kurve wird sehr gut reproduziert. Auf Grund der höheren Anzahl an Parametern, wird eine Konvergenz erst nach ca. 200 Generationen erreicht (vgl. Abb. 5.15). In Abb 5.16 sind die simulierten LSVen dargestellt. Im Vergleich zu den simulierten LSVen der stochastischen Struktur (vgl. Abb. 5.6) fällt auf, dass die Lichtstärke in dem Bereich $(\alpha \geq 0°, \beta \geq 0°)$ größer ist. Die Parameter, mit denen die Strukturen erstellt worden sind, sind:

$$\Delta P_{y,1x10} = (2,9;\ -2,7;\ -13,6;\ -1,4;\ -3,1;\ 18,9;\ -19,2;\ -14,7;\ 7,1;\ -1,7)^{\mathrm{T}}\ \mu\mathrm{m}$$

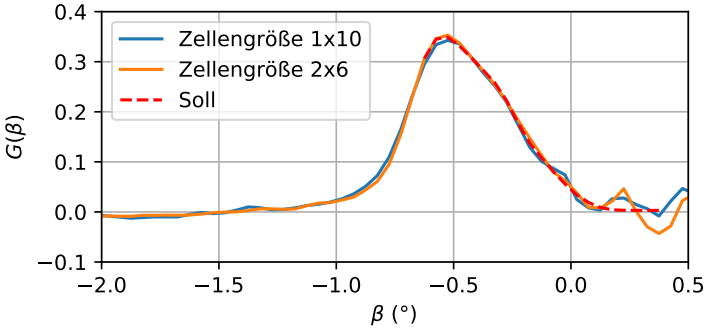


Abbildung 5.14: Vergleich zwischen der Soll- und den resultierenden Gradienten-Kurven für die zwei Optimierungen mit unterschiedlichen Freiheitsgraden der Mikrostrukturen.

$$\Delta P_{z,1x10} = (-0,31;\ 0,58;\ 0,49;\ 0,41;\ 0,47;\ -0,10;\ -0,18;\ 0,14;\ -0,22;\ 0,81)^{\mathrm{T}}\ \mu\mathrm{m}$$

$$\Delta P_{y,2x6} = \begin{pmatrix} 37,6 & -3,7 \\ 44,2 & -22,6 \\ -17,9 & -52,2 \\ 1,2 & 13,7 \\ 41,8 & -42,3 \\ 10,2 & 34,6 \end{pmatrix} \mu\mathrm{m} \quad \Delta P_{z,2x6} = \begin{pmatrix} -0,05 & -0,14 \\ -0,82 & 0,46 \\ -0,65 & -0,60 \\ -0,37 & 0,85 \\ 0,59 & 0,89 \\ -0,24 & -0,36 \end{pmatrix} \mu\mathrm{m}$$

Die simulierten LSVen und Gradienten-Kurven des kompletten Moduls sind in Abb. 5.17 dargestellt. Die Gradienten-Kurven passen sehr gut zu den Vorgaben. Die simulierten LSVen zeigen wellenförmige Modulationen in dem Bereich $\beta < -10°$. Die Modulationen sind bei der Struktur mit der Zellengröße 1x10 stärker ausgeprägt als bei der Struktur mit der Zellengröße 2x6. Die Verwendung von Zellen, die in einer Richtung eine Größe von 1 haben, erzeugen eine eindimensionale Struktur, die das Licht nur in vertikaler Richtung ablenkt (vgl. Abb. 5.17). Zweidimensio-

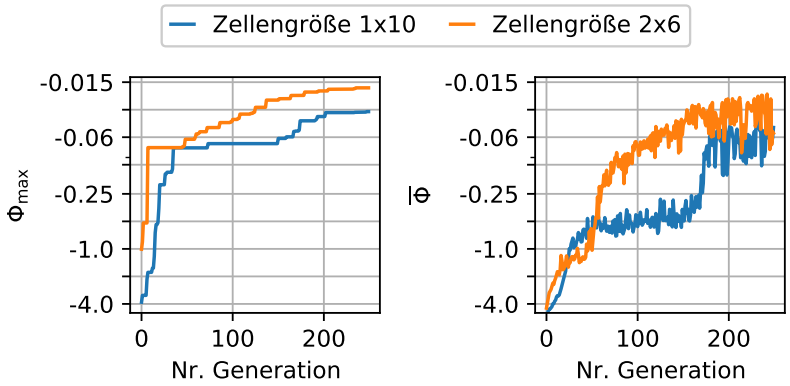


Abbildung 5.15: Konvergenzverhalten des genetischen Algorithmus bei der Optimierung von periodischen Freiform-Strukturen. Links: Maximale Fitness pro Generation. Rechts: Durchschnittliche Fitness pro Generation.

nale Zellen verursachen eine zweidimensionale Streuung. Dadurch wird die Ausprägung der Modulation verringert.

Wie bei den Simulationen der Optimierung zeigt sich auch bei der Simulation des gesamten Moduls eine höhere Lichtstärke in dem Bereich ($\alpha \geq 0°, \beta \geq 0°$). Die Lichtstärke übersteigt in diesem Bereich teilweise die gesetzlichen Höchstgrenzen. In Tab. 5.1 sind die Werte der Messpunkte aufgezeigt, die überschritten werden. Die Tabelle vergleicht die gesetzlichen Höchstwerte mit den Simulationen aus den Kap. 4.2, 5.3.1 und 5.4. Sowohl die stochastische Struktur, ausgelegt von einem Optikingenieur zum Testen der neuen Prozesskette, als auch die stochastische Struktur, bestimmt durch den genetischen Algorithmus, erfüllen die Vorgaben. Die periodische Freiform-Struktur mit der Zellengröße 1x10 überschreitet beide Werte, die mit der Zellengröße 2x6 den HV-Wert.

Dieses Kapitel zeigt, dass es möglich ist, Strukturen durch Optimierungsalgorithmen zu finden, die eine vorgegebene Gradienten-Kurve reproduzieren und deren Amplituden nicht auf Normalverteilungen

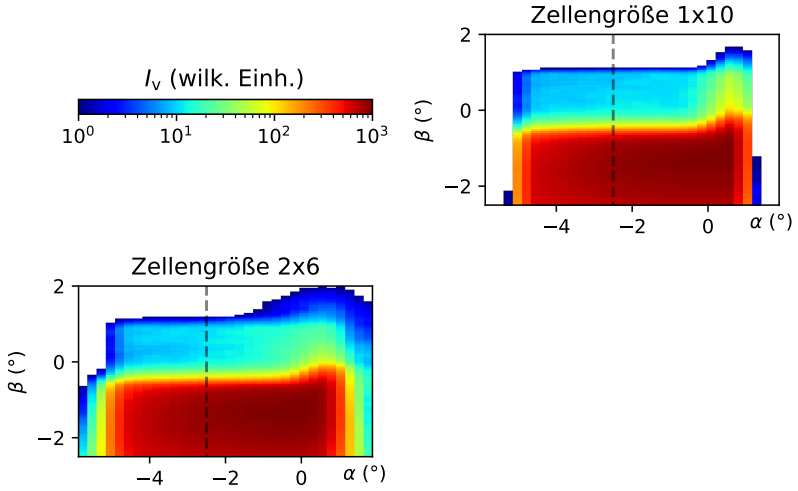


Abbildung 5.16: Simulierte LSVen mit jeweils der periodischen Freiform-Struktur mit der maximalen Fitness. Die gestrichelte Linie zeigt die Position, an der die Gradienten-Kurve bestimmt wurde.

(vgl. Kap.5.3.1) basieren. Die Strukturen erzeugen jedoch wellenförmige Modulationen der LSV oder streuen soviel Licht über die HDG, dass gesetzliche Werte überschritten werden. Daher kann die Gradienten-Kurve nicht unabhängig als einzelnes Kriterium verwendet werden, um eine Struktur als gut oder schlecht zu bewerten. Um eine Auslegung zu ermöglichen, muss die Bewertungsfunktion (5.1) angepasst werden, sodass diese die gesetzlichen Grenzen beachtet. Das Verhindern der wellenförmigen Modulationen ist schwieriger. Um diese zu verhindern, muss entweder die Bewertungsfunktion angepasst oder die Parameter der Strukturen so gewählt werden, dass diese nicht entstehen können. Für die Anpassung der Bewertungsfunktion muss die Modulation mathematisch beschrieben werden, was keine triviale Pro-

Tabelle 5.1: Vergleich von gesetzlichen Vorgaben für die LSV für ein Abblendlicht mit den Simulationsergebnissen. Die Angabe $\sigma_{[-1.25,1]}$ bezieht sich auf Abb. 5.6.

Bez.	α (°)	β (°)	gesetzliches Max. (cd)	
HV	0	0	625	
BR	2,5	1,0	1750	
Bez.	stochastisch ECE (cd)	$\sigma_{[-1.25,1]}$ (cd)	1x10 (cd)	2x6 (cd)
HV	598	578	724	1374
BR	308	310	2096	1394

blemstellung ist. Vermutlich ist das Verhindern der Modulation durch die Verwendung von zweidimensionalen Zellen einfacher. Wie der Vergleich zwischen den LSVen der Zellengröße 1x10 und 2x6 zeigt, können die Modulationen durch die Wahl der Zellengröße verringert werden. Bei eindimensionalen Zellen wird das Licht nur entlang einer Achse gestreut, wodurch die Modulationen entstehen. Durch die Verwendung von zweidimensionalen Zellen wird das Licht zweidimensional gestreut womit sich die Modulationen verhindern lassen.

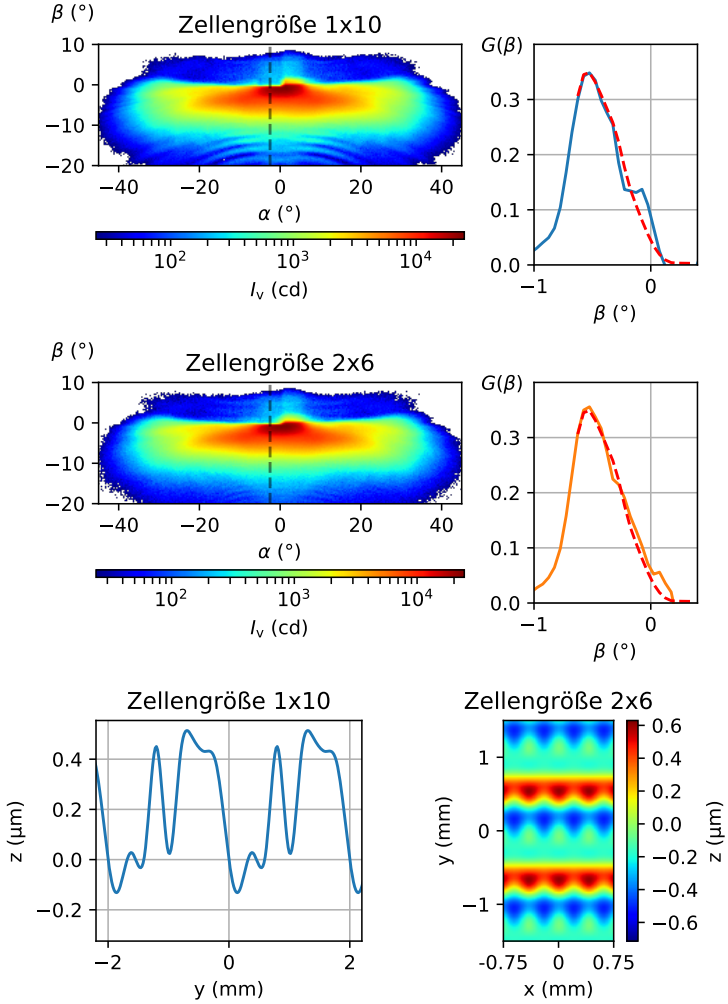


Abbildung 5.17: Oben & Mitte: Simulation des kompletten *BiLED*-Moduls mit den nicht-stochastischen Mikrostrukturen. Die gestrichelten Linien in den LSVen geben die Position des Auswertebereiches für die Gradienten-Kurve an. Unten: Darstellung der Strukturen.

KAPITEL 6

ZUSAMMENFASSUNG UND AUSBLICK

In dieser Arbeit wurde die Prozesskette zur Auslegung und Herstellung von mikrostrukturierten Linsen verbessert. Die neue Prozesskette ist erneut in Abb. 6.1 dargestellt. Durch Entwicklung eines Strukturierungs-Algorithmus konnten die kommerziellen Programme Rhino und Mathematica entfernt werden (vgl. Kap. 1). Die Anzahl der zur Beschreibung

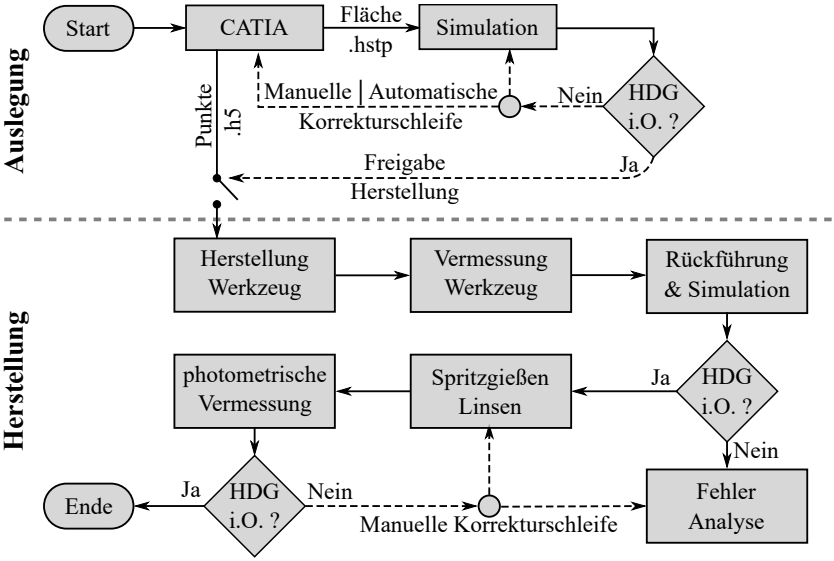


Abbildung 6.1: Neue Prozesskette zur Herstellung mikrostrukturierter Linsen.

der Fläche verwendeten Datenformate konnte von fünf auf zwei reduziert werden. Dadurch reduziert sich neben der Prozesszeit auch die

Gefahr an Datenverluste durch Konvertierungsfehler. Einerseits kosteten die vielen manuellen Schritte Zeit, andererseits benötigte Rhino ca. 25 min um eine Fläche zu berechnen. Die Zeit für die Strukturierung einer Fläche konnte auf wenige Sekunden reduziert werden. Der Algorithmus zur Strukturierung wurde in das CAD-Programm CATIA integriert. Für die Optikingenieure unterscheidet sich die Funktion nicht von anderen CATIA-Funktionen. Die alte Prozesskette erforderte viel Erfahrung, um alle notwendigen Programme bedienen zu können. Für die neue Prozesskette reichen CATIA-Kenntnisse aus. Die Funktionsweise wurde an zwei Beispielen erfolgreich demonstriert.

Um eine automatisierte Auslegung der Strukturen zu ermöglichen, wurde die Simulationszeit stark reduziert. Die Reduzierung basiert auf der Entwicklung eines auf moderne Prozessoren angepassten Algorithmus zu Berechnung der B-Spline Basis-Funktionen sowie der Vermeidung unnötiger Strahlverfolgungen bei der Simulation. Durch Anwendung von genetischen Algorithmen wurde die Auslegung der Mikrostrukturen automatisiert. An zwei Beispielen wurde gezeigt, dass vorgegebene Gradienten-Kurven reproduziert werden können. Bisher basierten die Amplituden der Strukturen auf Normalverteilungen. Mit dem Optimierungsalgorithmus konnten Strukturen gefunden werden, indem direkt die Kontrollpunkte der Flächen verschoben wurden. Die Strukturen erzeugen die vorgegebenen Gradienten-Kurven, überschreiten jedoch die gesetzlichen Blendwerte. Zukünftig könnten durch die Freiform-Strukturen eventuell die HDG bzw. der Verlauf der Gradienten-Kurve mit mehr Freiheitsgraden gestaltet werden.

Die Prozesskette der Herstellung wurde um die Vermessung, Rückführung und lichttechnische Simulation der Werkzeugeinsätze erweitert. Dadurch wurde eine neue Analysemöglichkeit geschaffen, um eventuell auftretende Fehler frühzeitig zu erkennen. Damit kann die Anzahl an Werkzeugkorrekturen reduziert werden, wodurch Zeit und Ressourcen

eingespart werden. An vier Werkzeugeinsätzen wurde die Funktionsweise erfolgreich demonstriert.

Ausblick

Bei der Flächenrückführung der vermessenen Werkzeugeinsätze wurden die Positionen der Kontrollpunkte als Freiheitsgrade verwendet, wobei die Anzahl bei der Rückführung festgehalten wurde. Sollten zukünftig Abweichungen auftreten, die eine höhere Anzahl an Freiheitsgraden benötigt, so kann der Algorithmus angepasst werden. Zunächst würde die bestehende Anzahl verwendet werden. Finden sich Bereiche, in denen die Anpassung nicht genau genug ist, so kann durch das Hinzufügen weiterer Zeilen bzw. Spalten in der Kontrollpunktmatrix die Flexibilität der Fläche lokal erhöht werden.

Um eine noch bessere Kontrolle über den Herstellungsprozess zu bekommen, wäre eine Vermessung der strukturierten Linsenoberfläche wünschenswert. Zum Zeitpunkt dieser Arbeit konnte seitens der Messtechnik keine Linsenoberfläche komplett vermessen werden. Durch langwellige Abweichungen in der Lichtaustrittsseite der Linse sowie eine Verkippung zwischen Lichteintritts- zu Lichtaustrittsseite konnten die Linsen auf der Messmaschine nicht ausgerichtet werden. Die Vermessung, Rückführung und Simulation würde die Kontrolle des Spritzgussprozesses ermöglichen. Wird die Vermessung des Werkzeugeinsatzes verglichen mit der Vermessung der Linsenoberfläche, so lässt sich beurteilen, ob die Strukturen ordnungsgemäß aus dem Einsatz abgeformt wurden.

Die Freiform-Strukturen erzeugen zu viel Blendung. Um das zu verhindern, muss eine neue Bewertungsfunktion gefunden werden, die sowohl die Gradienten-Kurve, als auch die gesetzlichen Blendwerte beachtet. Durch eine größere Versuchsreihe kann herausgefunden werden, wie viele Parameter minimal notwendig sind, um gewünschte Gradienten-Kurven zu generieren. Das Herabsenken der Anzahl an Parametern

reduziert die Optimierungszeit. Moderne Grafikkarten sind für schnelle Ray-Tracing-Berechnungen ausgelegt [98, 99]. Eventuell kann die Optimierungszeit durch die Berechnung auf Grafikkarten anstatt des Prozessors reduziert werden.

Bei Strukturierung der Linsen durch den Optimierungsalgorithmus wird die ganze Linse mit der Struktur versehen. Die Optikingenieure strukturieren z. B. in Abhängigkeit des Bereiches, durch den das Fernlicht die Linse passiert. Denkbar ist, vor oder während des Optimierungsprozesses zu analysieren, durch welchen Bereich der Linse das meiste Licht auf die HDG trifft. Damit kann die Struktur nur in dem Bereich aufgetragen werden, in dem sie notwendig ist. Weitergehend könnte die Struktur in Form von Markenzeichen (z. B. Mercedes-Stern) aufgebracht werden, wodurch sich neue Design-Möglichkeiten ergeben.

LITERATUR

[1] E. F. Schubert. *Light-Emitting Diodes*. 2. Aufl. Cambridge University Press, 2006.

[2] R. Schedel. "LED-Scheinwerfer gewinnen Marktanteile". *ATZ - Automobiltechnische Zeitschrift* 109.11 (2007), S. 1010–1013.

[3] M. Götz, M. Kleinkes, S. Pietzonka und N. Schiermeister. "Künftige Entwicklungen von LED-Scheinwerfern". *ATZ - Automobiltechnische Zeitschrift* 111.9 (2009), S. 620–627.

[4] M. Maier, J. Moisel und F. Herold. "Multibeam-Scheinwerfer in der Mercedes-Benz CLS-Klasse". *ATZ - Automobiltechnische Zeitschrift* 117.2 (2015), S. 16–21.

[5] R. Lachmayer, M. Götz, M. Kleinkes und W. Pohlmann. "LED-Technik im Scheinwerfer". *ATZ - Automobiltechnische Zeitschrift* 108.11 (2006), S. 956–961.

[6] B. Wördenweber, J. Wallaschek, P. Boyce und D. Hoffman. *Automotive Lighting and Human Vision*. Springer-Verlab Berlin Heidelberg, 2007.

[7] E. B. Goldstein. *Wahrnehmungspsychologie: Eine Einführung*. Spektrum Akademischer Verlag, 1997.

[8] Economic Comission for Europe. *Regulation No. 123 - Uniform provisions concerning the approval of adaptive front-lighting systems (AFS) for motor vehicles*. 2013.

[9] Dassault Systemes. *CATIA V5-R2016*. URL: https://www.3ds.com/de/produkte-und-services/catia/produkte/.

[10] R. Anderl und D. Trippner. *STEP STandard for the Exchange of Product Model Data*. Vieweg+Teubner Verlag, 2000.

[11] Robert McNeel & Associates. *Rhinoceros 3D V5*. URL: https://www.rhino3d.com/.

[12] B. Fortner. *The Data Handbook*. Springer-Verlag New York, 1995.

[13] Wolfram Research. *Mathematica*. URL: https://www.wolfram.com/mathematica/.

[14] U. S. Product Data Association. *Initial Graphics Exchange Spcification IGES 5.3*. 1996.

[15] J. Bliedtner und G. Gräfe. *Optiktechnologie: Grundlagen - Verfahren - Anwendungen - Beispiele*. Fachbuchverl. Leipzig im Carl-Hanser-Verlag, 2008.

[16] Bundesministerium für Bildung und Forschung. *Industrie 4.0 - Innovationen für die Produktion von morgen*. 2017.

[17] C. Wenzel, A. Beutler, D. Zimmermann, A. Stockfisch und S. Lowis. *Abschlussbericht zum Verbundvorhaben: Effiziente Photonikproduktion durch intelligente Technologie (ePiTec).* 2019.

[18] S. Holtz und J. Schmidt. "Schweinwerferlinse für einen Kraftfahrzeugscheinwerfer". Pat. DE 10 2005 009 556 A1. 2005.

[19] J. Fischer. "Verfahren zum Herstellen einer Scheinwerferlinse für einen Kraftfahrzeugscheinwerfer". Pat. WO 2009 012 735 A1. 2009.

[20] F. Hutter, L. Kotthoff und J. Vanschoren. *Automated Machine Learning*. Springer International Publishing, 2019.

[21] J. H. Hamkens. "Verfahren zum Herstellen eines optischen Linsenelementes, insbesondere einer Scheinwerferlinse für einen Kraftfahrzeugscheinwerfer". Pat. W0 2009 018 798 A2. 2009.

[22] R. Bonitz, R. Biertümpfel, W. Semar und M. Redey. "Optische Linse mit Weichzeicheneffekt". Pat. EP 1 514 148 B1. 2008.

[23] M. Kiesel. “Zur Erzeugung einer definierten Overhead-Beleuchtung eingerichteter Fahrzeugscheinwerfer”. Pat. DE 10 2009 020 593 B4. 2017.

[24] M. Kiesel, B. v. Blanckenhagen und E.-O. Rosenhahn. “Beleuchtungseinrichtung in Form eines Projektionsscheinwerfers für Kraftfahrzeuge”. Pat. DE 10 2008 023 551 B4. 2019.

[25] G. G. Wang. “Definition and Review of Virtual Prototyping”. *Journal of Computing and Information Science in Engineering* 2.3 (2002), S. 232–236.

[26] J. C. Schaaf Jr. und F. L. Thompson. “System Concept Development with Virtual Prototyping”. *Proceedings of the 29th Conference on Winter Simulation*. 1997, S. 941–947.

[27] J. D. Foley, A. van Dam, S. K. Feiner und J. F. Hughes. *Computer Graphics: Principles and Practice.* 2. Aufl. Addison-Wesley, 1996.

[28] A. S. Glassner, Hrsg. *An Introduction to Ray Tracing*. Academic Press Ltd., 1989.

[29] F. Pedrotti, L. Pedrotti, W. Bausch und H. Schmidt. *Optik für Ingenieure: Grundlagen*. Springer Berlin Heidelberg, 2007.

[30] O. Reeb. *Grundlagen der Photometrie*. G. Braun Karlsruhe, 1962.

[31] N. Metropolis und S. Ulam. “The Monte Carlo Method”. *Journal of the American Statistical Association* 44.247 (1949), S. 335–341.

[32] N. Metropolis. “THE BEGINNING of the MONTE CARLO METHOD”. *Los Alamos Science Special Issue* 15 (1987), S. 125–130.

[33] M. Pharr und G. Humphreys. *Physically Based Rendering*. 2. Aufl. Morgan Kaufmann Publishers Inc., 2010.

[34] W. H. Press, S. A. Teukolsky, W. T. Vetterling und B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. 2. Aufl. Cambridge University Press, 1992.

[35] P. E. Bezier. “Example of an Existing System in the Motor Industry: The Unisurf System”. *Proceedings of the Royal Society of London. Series A* 321 (1971), S. 207–218.

[36] C. Rabut. “On Pierre Bézier’s life and motivations”. *Computer-Aided Design* 34 (2002), S. 493–510.

[37] D. F. Rogers. *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann Publishers Inc., 2001.

[38] I. J. Schoenberg. “Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions”. *Quarterly of Applied Mathematics* 4.1 (1946), S. 45–99.

[39] R. F. Riesenfeld. “Applications of B-spline Approximation to Geometric Problems of Computer-aided Design.” Diss. Syracuse University, 1973.

[40] L. Piegl und W. Tiller. *The NURBS Book*. 2. Aufl. Springer-Verlag, 1996.

[41] C. De Boor. *A Practical Guide to Splines; rev. ed.* Applied Mathematical Sciences. Springer, 2001.

[42] *GNU Scientific Library (GSL)*. Version 2.5. 19. Juli 2019. URL: http://www.gnu.org/software/gsl/.

[43] M. Galassi, J. Davies, J. Theiler, B. Gough und G. Jungman. *GNU Scientific Library - Reference Manual, Third Edition, for GSL Version 1.12 (3. ed.)*. 2009.

[44] *geomdl*. Version 5.2.8. 19. Juli 2019. URL: https://pypi.org/project/geomdl/.

[45] O. R. Bingol und A. Krishnamurthy. “NURBS-Python: An open-source object-oriented NURBS modeling framework in Python”. *SoftwareX* 9 (2019), S. 85–94.

[46] *Object Oriented Finite Element Solver (OOFEM)*. Version 2.5. 19. Juli 2019. URL: http://www.oofem.org.

[47] B. Patzák. "OOFEM - an object-oriented simulation tool for advanced modeling of materials and structures". *Acta Polytechnica* 52.6 (2012), S. 59–66.

[48] *Overture - An Object-Oriented Toolkit for Solving Partial Differential Equations in Complex Geometry*. Version 25. 19. Juli 2019. URL: http://www.overtureframework.org/.

[49] R. M. Hord. *The Illiac IV - The First Supercomputer*. Springer-Verlag Berlin Heidelberg, 1982.

[50] C. Lomont. *Introduction to Intel Advanced Vector Extensions*. 2011.

[51] O. Abert, M. Geimer und S. A. Muller. "Direct and Fast Ray Tracing of NURBS Surfaces". *IEEE Symposium on Interactive Ray Tracing* (2006), S. 161–168.

[52] *Intel Intrinsics Guide*. 19. Juli 2019. URL: https://software.intel.com/sites/landingpage/IntrinsicsGuide/.

[53] Microprocessor Standards Committee. *754-2019 - IEEE Standard for Floating-Point Arithmetic*. IEEE, 2019.

[54] P. Gepner. "Using AVX2 Instruction Set to Increase Performance of High Performance Computing Code". *Computing and Informatics* 36 (2017), S. 1001–1018.

[55] S. Dreiseitl. *Mathematik für Software Engineering*. Springer Vieweg, 2018.

[56] *Intel® Xeon® Prozessor E5-1650 v3*. 28. März 2020. URL: https://ark.intel.com/content/www/de/de/ark/products/82765/intel-xeon-processor-e5-1650-v3-15m-cache-3-50-ghz.html.

[57] G. Hager und G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, Inc., 2010.

[58] J. E. Smith. “A Study of Branch Prediction Strategies”. *Proceedings of the 8th Annual Symposium on Computer Architecture*. 1981, S. 135–148.

[59] Intel. *Intel 64 and IA-32 Architectures Optimization Reference Manual*. 2016.

[60] R. Kapoor. *Avoiding the Cost of Branch Misprediction*. 2009. URL: https://software.intel.com/en-us/articles/avoiding-the-cost-of-branch-misprediction.

[61] D. Zimmermann u. a. “A procedure for designing and manufacturing microstructured lenses used in automotive headlamps”. *Advanced Optical Technologies* 8.6 (2019), 483–489.

[62] National Highway Traffic Safety Administration. *Federal Motor Vehicle Safety Standard 108*. 2007.

[63] S. D. Conte und C. W. D. Boor. *Elementary Numerical Analysis: An Algorithmic Approach*. 3. Aufl. McGraw-Hill Higher Education, 1980.

[64] C. Schmidt und F.-J. Kalze. “Dynamic Cut-Off-Line geometry as the next step in forward lighting beyond AFS”. *International Symposium on Automotive Lighting*. 2007.

[65] C. Schmidt, F.-J. Kalze und T. Irmscher. “Illumination Strategies for Dynamic Headlamp Functions like Adaptive and Vertical Cut-Off-Line”. *International Symposium on Automotive Lighting*. 2009.

[66] J. Moisel, R. Ackermann und M. Griesinger. “Adaptive Headlights utilizing LED-Arrays”. *International Symposium on Automotive Lighting*. 2009.

[67] C. Brecher, C. Baum, B. Meiers, D. De Simone und R. Krappig. *Kunststoffkomponenten für LED-Beleuchtungsanwendungen*. Springer Vieweg, 2016.

[68] R. L. Rhorer und C. J. Evans. “Fabrication of Optics by Diamond Turning”. In: *Handbook of Optics*. Hrsg. von M. Bass. New York: McGraw-Hill, Inc., 1995. Kap. 41, S. 1–13.

[69] T. Moriwaki und E. Shamoto. “Ultraprecision Diamond Turning of Stainless Steel by Applying Ultrasonic Vibration”. *CIRP Annals* 40.1 (1991), S. 559 –562.

[70] C. Brecher, F. Klocke, M. Winterschladen und M. Heselhaus. “Ultraschallunterstütztes Hartdrehen für die Fertigung von gehärteten Präzisionsstahlbauteilen”. *wt Werkstatttechnik online* 4 (2006), S. 396–401.

[71] F. Schneider, J. Das, B. Kirsch, B. Linke und J. C. Aurich. “Sustainability in Ultra Precision and Micro Machining: A Review”. *International Journal of Precision Engineering and Manufacturing-Green Technology* 6.3 (2019), S. 601–610.

[72] E. Brinksmeier und R. Gläbe. “Elliptical Vibration Cutting Steel with Diamond Tools.” *Proceedings of 14th Annual ASPE Meeting* (1999).

[73] Innolite GmbH. *IL 300*. URL: https://innolite.de/wp-content/uploads/2018/pdf/Innolite_IL300_Brochure_en.pdf.

[74] C. Wenzel, A. Stockfisch, T. Weske und D. Zontar. “Effiziente Produktion von komplexen Kunststoffoptiken”. *wt Werkstatttechnik online* 4 (2019), S. 388–393.

[75] M. Brozio, D. Sensen, C. Wenzel, S. Freutel und C. Brecher. “Surface adaptive fast axis ultra precision turning”. *Proceedings of SPIE*. 2018, 10544:1–6.

[76] The HDF Group. *Hierarchical Data Format, version 5*. 2019. URL: http://www.hdfgroup.org/HDF5/.

[77] A. Beutler. “Flexible, non-contact and high-precision measurements of optical components”. *Surface Topography: Metrology and Properties* 4.2 (2016), 024011:1–12.

[78] A. Weckenmann und W. Hartmann. “Function-oriented method for the definition and verification of microstructured surfaces”. *Precision Engineering* 37.3 (2013), S. 684 –693.

[79] J. Stock, M. Beier, J. Hartung, S. Merx und H. Gross. “Simulation and analysis of optical imaging systems including real freeform components”. *Advanced Optical Technologies* 8.2 (2019), S. 111–117.

[80] F. Bornemann. *Numerische lineare Algebra*. Springer Spektrum, 2018.

[81] M. Hemmerich. “Evaluation und Implementierung von Algorithmen zur Rückführug von Punktwolken in NURBS-Flächen zur Verwendung in lichttechnischen Simulationsprogrammen”. Masterarbeit. Hochschule Bochum, 2018.

[82] G. Becker, M. Schäfer und A. Jameson. “An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations”. *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition* (2011).

[83] Y. Zhang, H.-N. Cheng, R. Wu und R. Liang. “Data processing for point-based in situ metrology of freeform optical surface”. *Optics. Express* 25.12 (2017), S. 13414–13424.

[84] N. Carlsson. “NURBS Surface Fitting with Gauss-Newton”. Masterarbeit. Luleå University of Technology, 2009.

[85] M. Randrianarivony und G. Brunnett. “Parallel Implementation of Surface Reconstruction From Noisy Samples”. *TU Chemnitz - Sonderforschungsbereich 393* (2002).

[86] M. Randrianarivony und G. Brunnen. “Approximation by NURBS curves with free knots”. *Proceedings of the Vision, Modeling, and Visualization Conference*. 2002, S. 195–201.

[87] D. Brujic, I. Ainsworth und M. Ristic. “Fast and accurate NURBS fitting for reverse engineering”. *The International Journal of Advanced Manufacturing Technology* 54 (2011), S. 691–700.

[88] Y. Kineri, M. Wang, H. Lin und T. Maekawa. “B-spline surface fitting by iterative geometric interpolation/approximation algorithms”. *Computer-Aided Design* 44.7 (2012), S. 697 –708.

[89] S. Hu und J. Wallner. “A second order algorithm for orthogonal projection onto curves and surfaces”. *Computer Aided Geometric Design* 22 (2005), S. 251–260.

[90] X. Li u. a. “Convergence Analysis on a Second Order Algorithm for Orthogonal Projection onto Curves”. *Symmetry* 9 (2017), 210:1–13.

[91] I. N. Bronstein, K. A. Semendjajew, G. Musiol und H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 2001.

[92] C. Schwanengel. *Comparison of techniques for measuring luminous intensity distribution overall and across segments*. TechnoTeam Bildverarbeitung GmbH, 2010.

[93] D. Zimmermann und C. Neumann. “Simulation und Design mikrostrukturierter Linsen für automobile Scheinwerfer”. *Lux junior*. 2019.

[94] K. Höschel und V. Lakshminarayanan. “Genetic algorithms for lens design: a review”. *Journal of Optics* 48.1 (2018), 134–144.

[95] R. L. Haupt und S. E. Haupt. *Practical Genetic Algorithms*. Wiley-Interscience, 2004.

[96] S. Luke. *Essentials of Metaheuristics*. 2. Aufl. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/. Lulu, 2013.

[97] H. Mühlenbein und D. Schlierkamp-Voosen. “Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization”. *Evolutionary Computation* 1.1 (1993), S. 25–49.

[98] V. Sanzharov, A. Gorbonosov, V. Frolov und A. Voloboy. “Examination of the Nvidia RTX”. *Proceedings of the 29th International Conference on Computer Graphics and Vision*. 2019, S. 7–12.

[99] NVIDIA. *NVIDIA Turing GPU Architecture (WP-09183-001_v01)*. 2018.

Eigene Veröffentlichungen

[61] D. Zimmermann u. a. “A procedure for designing and manufacturing microstructured lenses used in automotive headlamps”. *Advanced Optical Technologies* 8.6 (2019), 483–489.

[93] D. Zimmermann und C. Neumann. “Simulation und Design mikrostrukturierter Linsen für automobile Scheinwerfer”. *Lux junior*. 2019.

Betreute studentische Arbeiten

[81] M. Hemmerich. “Evaluation und Implementierung von Algorithmen zur Rückführug von Punktwolken in NURBS-Flächen zur Verwendung in lichttechnischen Simulationsprogrammen”. Masterarbeit. Hochschule Bochum, 2018.

TABELLENVERZEICHNIS

ABBILDUNGSVERZEICHNIS

ABKÜRZUNGEN UND SYMBOLE

ABKÜRZUNGEN

AK31	ArbeitsKreis 31 (alternative Bezeichnung für „Gradient“)
AVX	Advanced Vector Extensions
ASCII	American Standard Code for Information Interchange
BMBF	Bundesministerium für Bildung und Forschung
CAD	Computer-Aided Design
ECE	Economic Comission for Europe
ePiTec	effiziente Photonikproduktion durch intelligente Technologie
FMVSS	Federal Motor Vehicle Safety Standards
HDF5	Hierarchical Data Format 5
HDG	Hell-Dunkel-Grenze
IGES	Initial Graphics Exchange Specification
LED	Light-Emitting Diode
LSV	LichtStärkeVerteilung
NURBS	Non-Uniform Rational Basis-Spline
SIMD	Single Instruction, Multiple Data
STEP	STandard for the Exchange of Product model data

LATEINISCHE SYMBOLE UND VARIABLEN

$\vec{C}(u)$	B-Spline-Kurve
G	Gradient bzw. AK31-Wert
$I_{\mathbf{v}}$	Lichtstärke
$\underline{\underline{J}}$	Jacobi-Matrix
n	Index des letzten Kontrollpunktes einer B-Spline-Fläche bzw. Kurve in u-Richtung
$N_{i,p}$	B-Spline-Basisfunktion vom Grad p
p	Grad der B-Spline-Kurve bzw. -Fläche in u-Richtung
$P_{i,j}$	Kontrollpunkt einer B-Spline-Fläche bzw. -Kurve
q	Grad der B-Spline-Fläche in v-Richtung
$\vec{Q}_{i,j}$	Gitterpunkt
$\vec{S}(u,v)$	B-Spline-Fläche
$\vec{R}$	Richtungsvektor eines Lichtstrahls
$S_c(u,v)$	Eine Koordinate der B-Spline-Fläche mit $c \in x,y,z$
u_i	Knoten einer B-Spline-Kurve bzw. -Fläche
U	Erster Knotenvektor einer B-Spline-Kurve bzw. -Fläche
v_i	Knoten einer B-Spline-Fläche
V	Zweiter Knotenvektor einer B-Spline-Fläche

GRIECHISCHE SYMBOLE UND VARIABLEN

α	Horizontaler Winkel
β	Vertikaler Winkel
σ	Standardabweichung einer Normalverteilung
Φ	Qualitätsfunktion bzw. Fitness-Wert
$\vec{v}$	Parameter einer Mikrostruktur
$\vec{\omega}$	Parameter einer Mikrostruktur

OPERATOREN UND MATHEMATISCHE SYMBOLE

$\odot$	Elementweise Multiplikation
$\left\langle \underline{\underline{A}}, \underline{\underline{B}} \right\rangle_{\mathbf{F}}$	Frobenius-Skalarprodukt

ANHANG

Programmauflistung 6.1: C++ Implementierung des Algorithmus 2.1.

```
void NubsConventionalDiss::compUBasis(int span, double param,
    VectorN& basis){
    int i, j;
    double temp, saved;

    basis[0] = 1.0;
    for ( i = 1; i < ndu; i++ ) {
        m_pLeft[i]  = param - (*knot_u_ptr)[span+1-i];
        m_pRight[i] = (*knot_u_ptr)[span+i] - param;
        saved = 0.0;
        for ( j = 0; j < i; j++ ) {
            temp = basis[j] / ( m_pRight[j+1] + m_pLeft[i-j] )
                ;
            basis[j] = saved + m_pRight[j+1]*temp;
            saved = m_pLeft[i-j]*temp;
        }
        basis[i] = saved;
    }
}
```

Programmauflistung 6.2: C++ Implementierung des Algorithmus 2.3.

```
__m256d CubicNubsSurface::getBasisFunsU(int span, double param
    ) const{
  const int spanMinusDegree = span-m_degree;
  const __m256d uVec = _mm256_set1_pd(param);

  // Contains [N_{i-3, 0}, N_{i-2, 0}, N_{i-1, 0}, N_{i, 0}]
  __m256d Nim3 = _mm256_setr_pd(0., 0., 0., 1.);

  // Calculate [N_{i-3, 1}, N_{i-2, 1}, N_{i-1, 1}, N_{i, 1}]
```

```
9     this->getNextDegreeBasisFuns(spanMinusDegree, m_c11u, m_c21u
          , m_c31u, m_c41u, uVec, Nim3);
10
11    // Calculate [N_{i-3, 2}, N_{i-2, 2}, N_{i-1, 2}, N_{i, 2}]
12    this->getNextDegreeBasisFuns(spanMinusDegree, m_c12u, m_c22u
          , m_c32u, m_c42u, uVec, Nim3);
13
14    // Calculate [N_{i-3, 3}, N_{i-2, 3}, N_{i-1, 3}, N_{i, 3}]
15    this->getNextDegreeBasisFuns(spanMinusDegree, m_c13u, m_c23u
          , m_c33u, m_c43u, uVec, Nim3);
16
17    return Nim3;
18  }
19
20  void CubicNubsSurface::getNextDegreeBasisFuns(
21    int spanMinusDegree, const std::vector<double>& c1,
22    const std::vector<double>& c2, const std::vector<double>& c3
          ,
23    const std::vector<double>& c4, const __m256d& paramVec,
24    __m256d &basisFuns) const{
25
26    // Calculate [N_{i-2, p}, N_{i-1, p}, N_{i, p}, N_{i-3, p}]
27    __m256d Nim2 = _mm256_permute4x64_pd(basisFuns, 57);
28    // Calculate [N_{i-2, p}, N_{i-1, p}, N_{i, p}, 0]
29    Nim2 = _mm256_blend_pd(Nim2, _mm256_setzero_pd(), 8);
30
31    // load constants
32    const __m256d c1Vec = _mm256_loadu_pd(&(c1[spanMinusDegree])
          );
33    const __m256d c3Vec = _mm256_loadu_pd(&(c3[spanMinusDegree])
          );
34    const __m256d c2Vec = _mm256_loadu_pd(&(c2[spanMinusDegree])
          );
35    const __m256d c4Vec = _mm256_loadu_pd(&(c4[spanMinusDegree])
          );
36
37    // c1 multiply u add c3
38    const __m256d c1mupc3 = _mm256_fmadd_pd(paramVec, c1Vec,
39      c3Vec);
```

```
40   // c2 multiply u add c4
41   const __m256d c2mupc4 = _mm256_fmadd_pd(paramVec, c2Vec,
42     c4Vec);
43
44   // Calculate [N_{i-3, 1}, N_{i-2, 1}, N_{i-1, 1}, N_{i, 1}]
45   basisFuns = _mm256_add_pd(_mm256_mul_pd(c1mupc3, basisFuns),
46     _mm256_mul_pd(c2mupc4, Nim2));
47 }
```

Programmauflistung 6.3: C++ Implementierung des Algorithmus 2.2.

```
1  void NubsConventionalDiss::compUDers( int span, double param,
       int n, Matrix& derU ){
2    int i,j,k;
3    int s1,s2;
4    int jk,uk;
5    double temp,saved;
6    double d;
7    int i1,i2;
8
9    (*m_pBasisTmp)[0][0] = 1.0;
10   for ( i = 1; i < ndu; i++ ) {
11     m_pLeft[i]  = param - (*knot_u_ptr)[span+1-i];
12     m_pRight[i] = (*knot_u_ptr)[span+i] - param;
13     saved = 0.0;
14     for ( j = 0; j < i; j++ ) {
15       (*m_pBasisTmp)[i][j] = m_pRight[j+1] + m_pLeft[i-j];
16       temp = (*m_pBasisTmp)[j][i-1] / (*m_pBasisTmp)[i][j];
17       (*m_pBasisTmp)[j][i] = saved + m_pRight[j+1]*temp;
18       saved = m_pLeft[i-j]*temp;
19     }
20     (*m_pBasisTmp)[i][i] = saved;
21   }
22
23   for ( i = 0; i < ndu; i++ ) {
24     derU[0][i] = (*m_pBasisTmp)[i][ndu-1];
25   }
26
27   for ( j = 0; j < ndu; j++ ) {
```

```
28      s1 = 0;
29      s2 = 1;
30      (*m_pColsTmp)[0][0] = 1.0;
31
32      for ( k = 1; k <= n; k++ ) {
33        d = 0.0;
34        jk = j - k;
35        uk = ndu -1 - k;
36        if ( j >= k ) {
37          (*m_pColsTmp)[s2][0] = (*m_pColsTmp)[s1][0] / (*
              m_pBasisTmp)[uk+1][jk];
38          d = (*m_pColsTmp)[s2][0] * (*m_pBasisTmp)[jk][uk];
39        }
40        if ( jk >= -1 )  i1 = 1;
41        else             i1 = -jk;
42        if ( j-1 <= uk ) i2 = k;
43        else             i2 = ndu - j;
44        for ( i = i1; i < i2; i++ ) {
45          (*m_pColsTmp)[s2][i] = ( (*m_pColsTmp)[s1][i] - (*
              m_pColsTmp)[s1][i-1] )
46                                 / (*m_pBasisTmp)[uk+1][jk+i];
47          d += (*m_pColsTmp)[s2][i] * (*m_pBasisTmp)[jk+i][uk];
48        }
49        if ( j <= uk ) {
50          (*m_pColsTmp)[s2][k] = -(*m_pColsTmp)[s1][k-1] / (*
              m_pBasisTmp)[uk+1][j];
51          d += (*m_pColsTmp)[s2][k] * (*m_pBasisTmp)[j][uk];
52        }
53        derU[k][j] = d;
54        i  = s1;
55        s1 = s2;
56        s2 = i;
57      }
58    }
59    j = ndu - 1;
60    for ( k = 1; k <= n; k++ ) {
61      for ( i = 0; i < ndu; i++ ) {
62        derU[k][i] *= j;
63      }
```

```
64      j *= ndu - 1 - k;
65    }
66  }
```

Programmauflistung 6.4: C++ Implementierung des Algorithmus 2.4.

```
1  __m256d CubicNubsSurface::getBasisDerivat(int span,
2      double param, __m256d &firstDerivatives) const
3  {
4    const int spanMinusDegree = span-m_degree;
5    const __m256d uVec = _mm256_set1_pd(param);
6
7    // Contains [N_{i-3, 0}, N_{i-2, 0}, N_{i-1, 0}, N_{i, 0}]
8    __m256d Nim3 = _mm256_setr_pd(0., 0., 0., 1.);
9
10   // Calculate [N_{i-3, 1}, N_{i-2, 1}, N_{i-1, 1}, N_{i, 1}]
11   this->getNextDegreeBasisFuns(spanMinusDegree, m_c11u, m_c21u
         , m_c31u, m_c41u, uVec, Nim3);
12
13   // Calculate [N_{i-3, 2}, N_{i-2, 2}, N_{i-1, 2}, N_{i, 2}]
14   this->getNextDegreeBasisFuns(spanMinusDegree, m_c12u, m_c22u
         , m_c32u, m_c42u, uVec, Nim3);
15
16   // Calculate [N_{i-3, 3}, N_{i-2, 3}, N_{i-1, 3}, N_{i, 3}]
17   // and [N^{(1)}_{i-3, 3}, N^{(1)}_{i-2, 3}, N^{(1)}_{i-1,
         3}, N^{(1)}_{i, 3}]
18   this->getNextDegreeBasisFunsDerivative(spanMinusDegree,
19     m_c13u, m_c23u, m_c33u, m_c43u, uVec, Nim3,
20     firstDerivatives);
21
22   return Nim3;
23 }
24
25 void CubicNubsSurface::getNextDegreeBasisFunsDerivative(
26   int spanMinusDegree,  const std::vector<double>& c1,
27   const std::vector<double>& c2,  const std::vector<double>&
         c3,
28   const std::vector<double>& c4, const __m256d& paramVec,
29   __m256d& basisFuns, __m256d& basisDerivative) const
```

```
30 {
31
32   // Calculate [N_{i-2, p}, N_{i-1, p}, N_{i, p}, N_{i-3, p}]
33   __m256d Nim2 = _mm256_permute4x64_pd(basisFuns, 57);
34   // Calculate [N_{i-2, p}, N_{i-1, p}, N_{i, p}, 0]
35   Nim2 = _mm256_blend_pd(Nim2, _mm256_setzero_pd(), 8);
36
37   // load constants
38   const __m256d c1Vec = _mm256_loadu_pd(&(c1[spanMinusDegree])
         );
39   const __m256d c3Vec = _mm256_loadu_pd(&(c3[spanMinusDegree])
         );
40   const __m256d c2Vec = _mm256_loadu_pd(&(c2[spanMinusDegree])
         );
41   const __m256d c4Vec = _mm256_loadu_pd(&(c4[spanMinusDegree])
         );
42
43   // c1 multiply u add c3
44   const __m256d c1mupc3 = _mm256_fmadd_pd(paramVec, c1Vec,
45     c3Vec);
46   // c2 multiply u add c4
47   const __m256d c2mupc4 = _mm256_fmadd_pd(paramVec, c2Vec,
48     c4Vec);
49
50   // We calculate the derivatives before the next degree basis
         functions, because we need the previous basis functions
         whose registers are overwritten later
51   basisDerivative = _mm256_add_pd(_mm256_mul_pd(c1Vec,
52     basisFuns), _mm256_mul_pd(c2Vec, Nim2));
53   basisDerivative = _mm256_mul_pd(basisDerivative,
         _mm256_set1_pd(m_degree));
54
55   // Calculate [N_{i-3, p+1}, N_{i-2, p+1}, N_{i-1, p+1}, N_{i
         , p+1}]
56   basisFuns = _mm256_add_pd(_mm256_mul_pd(c1mupc3, basisFuns),
57                             _mm256_mul_pd(c2mupc4, Nim2));
58 }
```

Programmauflistung 6.5: Bisher verwendete C++ Implementierung der Gl. (2.22) zur Auswertung einer B-Spline-Fläche.

```
1  HPoint NubsConventionalDiss::getPointFromUV(
2    const UVPoint& uv ){
3    int l,k,indU,indV;
4    int spanU,spanV;
5    HPoint pt;
6
7    getSpanUV ( uv, spanU, spanV );
8    compUBasis ( spanU, uv.u(), *m_pBasisU );
9    compVBasis ( spanV, uv.v(), *m_pBasisV );
10
11   for ( l = 0; l < ndv; l++ ) {
12     indV = spanV - ndv + 1 + l;
13
14     (*m_pMatrixTmp)[l][0] = 0.0;
15     (*m_pMatrixTmp)[l][1] = 0.0;
16     (*m_pMatrixTmp)[l][2] = 0.0;
17
18     for ( k = 0; k < ndu; k++ ) {
19       indU = spanU - ndu + 1 + k;
20       (*m_pMatrixTmp)[l][0] += (*m_pBasisU)[k] * (*
             c_points_x_ptr)[indU][indV];
21       (*m_pMatrixTmp)[l][1] += (*m_pBasisU)[k] * (*
             c_points_y_ptr)[indU][indV];
22       (*m_pMatrixTmp)[l][2] += (*m_pBasisU)[k] * (*
             c_points_z_ptr)[indU][indV];
23       }
24   }
25   for ( l = 0; l < ndv; l++ ) {
26     pt[0] += (*m_pBasisV)[l] * (*m_pMatrixTmp)[l][0];
27     pt[1] += (*m_pBasisV)[l] * (*m_pMatrixTmp)[l][1];
28     pt[2] += (*m_pBasisV)[l] * (*m_pMatrixTmp)[l][2];
29   }
30
31   return pt;
32 }
```

Programmauflistung 6.6: C++ Implementierung des Algorithmus 2.5 zur Auswertung einer B-Spline-Fläche.

```
1  void CubicNubsSurface::getPointFromUV(const UVPoint& uv,
       HPoint& retPoint) const{
2      int spanU = -1;
3      int spanV = -1;
4      __m256d m256_basisU, m256_basisV;
5      double result[3] = { {0} };
6
7      this->getSpanUV(uv, spanU, spanV);
8      m256_basisU = this->getBasisFunsU(spanU, uv.u());
9      m256_basisV = this->getBasisFunsV(spanV, uv.v());
10
11     const Matrix& xMatrix = *c_points_x_ptr;
12     const Matrix& yMatrix = *c_points_y_ptr;
13     const Matrix& zMatrix = *c_points_z_ptr;
14
15     int indV = -1;
16     int indU = -1;
17     indU = spanU - 3;
18     indV = spanV - 3;
19
20     double resultUnpacked[4] = { {0} };
21
22     __m256d N3j = _mm256_permute4x64_pd(m256_basisU,
           _MM_SHUFFLE(0, 0, 0, 0));
23     __m256d N2j = _mm256_permute4x64_pd(m256_basisU,
           _MM_SHUFFLE(1, 1, 1, 1));
24     __m256d N1j = _mm256_permute4x64_pd(m256_basisU,
           _MM_SHUFFLE(2, 2, 2, 2));
25     __m256d N0j = _mm256_permute4x64_pd(m256_basisU,
           _MM_SHUFFLE(3, 3, 3, 3));
26
27     N3j = _mm256_mul_pd(N3j, m256_basisV);
28     N2j = _mm256_mul_pd(N2j, m256_basisV);
29     N1j = _mm256_mul_pd(N1j, m256_basisV);
30     N0j = _mm256_mul_pd(N0j, m256_basisV);
31
32     // Calculate X
```

```
33      __m256d N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(
            xMatrix[spanU-3][indV])));
34      __m256d N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(
            xMatrix[spanU-2][indV])));
35      __m256d N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(
            xMatrix[spanU-1][indV])));
36      __m256d N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(
            xMatrix[spanU-0][indV])));
37
38      __m256d resultX = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
             _mm256_add_pd(N1jP, N0jP));
39      resultX = _mm256_hadd_pd(resultX, resultX);
40
41      _mm256_storeu_pd(resultUnpacked, resultX);
42      result[0] = resultUnpacked[0] + resultUnpacked[2];
43
44      // Calculate Y
45      N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(yMatrix[spanU
            -3][indV])));
46      N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(yMatrix[spanU
            -2][indV])));
47      N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(yMatrix[spanU
            -1][indV])));
48      N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(yMatrix[spanU
            -0][indV])));
49
50      resultX = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
            _mm256_add_pd(N1jP, N0jP));
51      resultX = _mm256_hadd_pd(resultX, resultX);
52
53      _mm256_storeu_pd(resultUnpacked, resultX);
54      result[1] = resultUnpacked[0] + resultUnpacked[2];
55
56      // Calculate Z
57      N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(zMatrix[spanU
            -3][indV])));
58      N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(zMatrix[spanU
            -2][indV])));
```

```
59      N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(zMatrix[spanU
            -1][indV])));
60      N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(zMatrix[spanU
            -0][indV])));
61
62      resultX = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
            _mm256_add_pd(N1jP, N0jP));
63      resultX = _mm256_hadd_pd(resultX, resultX);
64
65      _mm256_storeu_pd(resultUnpacked, resultX);
66      result[2] = resultUnpacked[0] + resultUnpacked[2];
67
68      retPoint.x() = result[0];
69      retPoint.y() = result[1];
70      retPoint.z() = result[2];
71  }
```

Programmauflistung 6.7: Bisher verwendete C++ Implementierung der Gl. (2.22) zur Auswertung einer B-Spline-Fläche und Berechnung der ersten Ableitungen.

```
1  void NubsConventionalDiss::getPointFromUV ( double uParameter,
2    double vParameter, double& pointX, double& pointY,
3    double& pointZ, double& uTangentX, double& uTangentY,
4    double& uTangentZ, double& vTangentX, double& vTangentY,
5    double& vTangentZ )
6  {
7
8      int           uSpan, vSpan;
9      UVPoint       uv (uParameter,vParameter);
10     int r,s;
11     int i1,i2;
12
13     getSpanUV(uv, uSpan, vSpan);
14     compUDers(uSpan, uParameter, 1, *m_pDerivatU);
15     compVDers(vSpan, vParameter, 1, *m_pDerivatV);
16
17   i2 = vSpan - (ndv-1);
18   for (s=0; s<ndv; s++) {
19     (*m_pMatrixTmp)[s][0] = 0.0;
```

```
20      (*m_pMatrixTmp)[s][1] = 0.0;
21      (*m_pMatrixTmp)[s][2] = 0.0;
22
23      i1 = uSpan - (ndu-1);
24      for (r=0; r<ndu; r++) {
25        (*m_pMatrixTmp)[s][0] += (*m_pDerivatU)[0][r] *
26          (*c_points_x_ptr)[i1][i2];
27        (*m_pMatrixTmp)[s][1] += (*m_pDerivatU)[0][r] *
28          (*c_points_y_ptr)[i1][i2];
29        (*m_pMatrixTmp)[s][2] += (*m_pDerivatU)[0][r] *
30          (*c_points_z_ptr)[i1][i2];
31        i1++;
32      }
33      i2++;
34    }
35
36    pointX = 0.0; pointY = 0.0; pointZ = 0.0;
37
38    for (s=0; s<ndv; s++) {
39      pointX += (*m_pDerivatV)[0][s] * (*m_pMatrixTmp)[s][0];
40      pointY += (*m_pDerivatV)[0][s] * (*m_pMatrixTmp)[s][1];
41      pointZ += (*m_pDerivatV)[0][s] * (*m_pMatrixTmp)[s][2];
42    }
43
44    vTangentX = 0.0; vTangentY = 0.0; vTangentZ = 0.0;
45
46    if ( ndv > 1 ) {
47      for (s=0; s<ndv; s++) {
48        vTangentX += (*m_pDerivatV)[1][s] * (*m_pMatrixTmp)[s
              ][0];
49        vTangentY += (*m_pDerivatV)[1][s] * (*m_pMatrixTmp)[s
              ][1];
50        vTangentZ += (*m_pDerivatV)[1][s] * (*m_pMatrixTmp)[s
              ][2];
51      }
52    }
53
54    uTangentX = 0.0; uTangentY = 0.0; uTangentZ = 0.0;
55
```

```
56    if ( ndu > 1 ) {
57      i2 = vSpan - (ndv-1);
58      for (s=0; s<ndv; s++) {
59        (*m_pMatrixTmp)[s][0] = 0.0;
60        (*m_pMatrixTmp)[s][1] = 0.0;
61        (*m_pMatrixTmp)[s][2] = 0.0;
62
63        i1 = uSpan - (ndu-1);
64        for (r=0; r<ndu; r++) {
65          (*m_pMatrixTmp)[s][0] += (*m_pDerivatU)[1][r] *
66            (*c_points_x_ptr)[i1][i2];
67          (*m_pMatrixTmp)[s][1] += (*m_pDerivatU)[1][r] *
68            (*c_points_y_ptr)[i1][i2];
69          (*m_pMatrixTmp)[s][2] += (*m_pDerivatU)[1][r] *
70            (*c_points_z_ptr)[i1][i2];
71          i1++;
72        }
73        i2++;
74      }
75      for (s=0; s<ndv; s++) {
76        uTangentX += (*m_pDerivatV)[0][s] * (*m_pMatrixTmp)[s
              ][0];
77        uTangentY += (*m_pDerivatV)[0][s] * (*m_pMatrixTmp)[s
              ][1];
78        uTangentZ += (*m_pDerivatV)[0][s] * (*m_pMatrixTmp)[s
              ][2];
79      }
80    }
81  }
```

Programmauflistung 6.8: C++ Implementierung des Algorithmus 2.5 zur Auswertung einer B-Spline-Fläche und Berechnung der ersten Ableitungen.

```
1  void CubicNubsSurface::getPointFromUV(double uParameter,
2    double vParameter, double& pointX, double& pointY,
3    double& pointZ, double& uTangentX, double& uTangentY,
4    double& uTangentZ, double& vTangentX, double& vTangentY,
5    double& vTangentZ)
6  {
```

```
7   int spanU = -1;
8   int spanV = -1;
9   __m256d m256_basisU, m256_basisV, m256_basisDerivativU,
        m256_basisDerivativV;
10  double resultUnpacked[4] = { {0} };
11
12  this->getSpanUV(UVPoint(uParameter, vParameter), spanU,
        spanV);
13  m256_basisU = this->getBasisDerivatU(spanU, uParameter,
        m256_basisDerivativU);
14  m256_basisV = this->getBasisDerivatV(spanV, vParameter,
        m256_basisDerivativV);
15
16  const Matrix& xMatrix = *c_points_x_ptr;
17  const Matrix& yMatrix = *c_points_y_ptr;
18  const Matrix& zMatrix = *c_points_z_ptr;
19
20  int indV = -1;
21  indV = spanV - 3;
22
23  // ========================================
24  // Point evaluation
25  // ========================================
26  __m256d N3j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE
        (0, 0, 0, 0));
27  __m256d N2j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE
        (1, 1, 1, 1));
28  __m256d N1j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE
        (2, 2, 2, 2));
29  __m256d N0j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE
        (3, 3, 3, 3));
30
31  N3j = _mm256_mul_pd(N3j, m256_basisV);
32  N2j = _mm256_mul_pd(N2j, m256_basisV);
33  N1j = _mm256_mul_pd(N1j, m256_basisV);
34  N0j = _mm256_mul_pd(N0j, m256_basisV);
35
36  // Calculate X
```

```
37  __m256d N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(xMatrix[
        spanU-3][indV])));
38  __m256d N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(xMatrix[
        spanU-2][indV])));
39  __m256d N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(xMatrix[
        spanU-1][indV])));
40  __m256d N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(xMatrix[
        spanU-0][indV])));
41
42  __m256d result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP
        ), _mm256_add_pd(N1jP, N0jP));
43  result_m256 = _mm256_hadd_pd(result_m256, result_m256);
44
45  _mm256_storeu_pd(resultUnpacked, result_m256);
46  pointX = resultUnpacked[0] + resultUnpacked[2];
47
48    // Calculate Y
49  N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(yMatrix[spanU
        -3][indV])));
50  N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(yMatrix[spanU
        -2][indV])));
51  N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(yMatrix[spanU
        -1][indV])));
52  N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(yMatrix[spanU
        -0][indV])));
53
54  result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
        _mm256_add_pd(N1jP, N0jP));
55  result_m256 = _mm256_hadd_pd(result_m256, result_m256);
56
57  _mm256_storeu_pd(resultUnpacked, result_m256);
58  pointY = resultUnpacked[0] + resultUnpacked[2];
59
60  // Calculate Z
61  N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(zMatrix[spanU
        -3][indV])));
62  N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(zMatrix[spanU
        -2][indV])));
```

```
63  N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(zMatrix[spanU
        -1][indV])));
64  N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(zMatrix[spanU
        -0][indV])));
65
66  result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
        _mm256_add_pd(N1jP, N0jP));
67  result_m256 = _mm256_hadd_pd(result_m256, result_m256);
68
69  _mm256_storeu_pd(resultUnpacked, result_m256);
70  pointZ = resultUnpacked[0] + resultUnpacked[2];
71
72  // ========================================
73  // uTangent Evaluation
74  // ========================================
75  N3j = _mm256_permute4x64_pd(m256_basisDerivativU,
        _MM_SHUFFLE(0, 0, 0, 0));
76  N2j = _mm256_permute4x64_pd(m256_basisDerivativU,
        _MM_SHUFFLE(1, 1, 1, 1));
77  N1j = _mm256_permute4x64_pd(m256_basisDerivativU,
        _MM_SHUFFLE(2, 2, 2, 2));
78  N0j = _mm256_permute4x64_pd(m256_basisDerivativU,
        _MM_SHUFFLE(3, 3, 3, 3));
79
80  N3j = _mm256_mul_pd(N3j, m256_basisV);
81  N2j = _mm256_mul_pd(N2j, m256_basisV);
82  N1j = _mm256_mul_pd(N1j, m256_basisV);
83  N0j = _mm256_mul_pd(N0j, m256_basisV);
84
85  // Calculate X
86  N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(xMatrix[spanU
        -3][indV])));
87  N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(xMatrix[spanU
        -2][indV])));
88  N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(xMatrix[spanU
        -1][indV])));
89  N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(xMatrix[spanU
        -0][indV])));
90
```

```
91  result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
        _mm256_add_pd(N1jP, N0jP));
92  result_m256 = _mm256_hadd_pd(result_m256, result_m256);
93
94  _mm256_storeu_pd(resultUnpacked, result_m256);
95  uTangentX = resultUnpacked[0] + resultUnpacked[2];
96
97  // Calculate Y
98  N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(yMatrix[spanU
        -3][indV])));
99  N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(yMatrix[spanU
        -2][indV])));
100 N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(yMatrix[spanU
        -1][indV])));
101 N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(yMatrix[spanU
        -0][indV])));
102
103 result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
        _mm256_add_pd(N1jP, N0jP));
104 result_m256 = _mm256_hadd_pd(result_m256, result_m256);
105
106 _mm256_storeu_pd(resultUnpacked, result_m256);
107 uTangentY = resultUnpacked[0] + resultUnpacked[2];
108
109 // Calculate Z
110 N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(zMatrix[spanU
        -3][indV])));
111 N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(zMatrix[spanU
        -2][indV])));
112 N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(zMatrix[spanU
        -1][indV])));
113 N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(zMatrix[spanU
        -0][indV])));
114
115 result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
        _mm256_add_pd(N1jP, N0jP));
116 result_m256 = _mm256_hadd_pd(result_m256, result_m256);
117
118 _mm256_storeu_pd(resultUnpacked, result_m256);
```

```
119 uTangentZ = resultUnpacked[0] + resultUnpacked[2];
120
121 //=======================================
122 // vTangent Evaluation
123 // =======================================
124 N3j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE(0, 0,
        0, 0));
125 N2j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE(1, 1,
        1, 1));
126 N1j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE(2, 2,
        2, 2));
127 N0j = _mm256_permute4x64_pd(m256_basisU, _MM_SHUFFLE(3, 3,
        3, 3));
128
129 N3j = _mm256_mul_pd(N3j, m256_basisDerivativV);
130 N2j = _mm256_mul_pd(N2j, m256_basisDerivativV);
131 N1j = _mm256_mul_pd(N1j, m256_basisDerivativV);
132 N0j = _mm256_mul_pd(N0j, m256_basisDerivativV);
133
134 // Calculate X
135 N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(xMatrix[spanU
        -3][indV])));
136 N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(xMatrix[spanU
        -2][indV])));
137 N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(xMatrix[spanU
        -1][indV])));
138 N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(xMatrix[spanU
        -0][indV])));
139
140 result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
        _mm256_add_pd(N1jP, N0jP));
141 result_m256 = _mm256_hadd_pd(result_m256, result_m256);
142
143 _mm256_storeu_pd(resultUnpacked, result_m256);
144 vTangentX = resultUnpacked[0] + resultUnpacked[2];
145
146 // Calculate Y
147 N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(yMatrix[spanU
        -3][indV])));
```

```
148   N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(yMatrix[spanU
          -2][indV])));
149   N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(yMatrix[spanU
          -1][indV])));
150   N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(yMatrix[spanU
          -0][indV])));
151
152   result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
          _mm256_add_pd(N1jP, N0jP));
153   result_m256 = _mm256_hadd_pd(result_m256, result_m256);
154
155   _mm256_storeu_pd(resultUnpacked, result_m256);
156   vTangentY = resultUnpacked[0] + resultUnpacked[2];
157
158   // Calculate Z
159   N3jP = _mm256_mul_pd(N3j, _mm256_loadu_pd(&(zMatrix[spanU
          -3][indV])));
160   N2jP = _mm256_mul_pd(N2j, _mm256_loadu_pd(&(zMatrix[spanU
          -2][indV])));
161   N1jP = _mm256_mul_pd(N1j, _mm256_loadu_pd(&(zMatrix[spanU
          -1][indV])));
162   N0jP = _mm256_mul_pd(N0j, _mm256_loadu_pd(&(zMatrix[spanU
          -0][indV])));
163
164   result_m256 = _mm256_add_pd(_mm256_add_pd(N3jP, N2jP),
          _mm256_add_pd(N1jP, N0jP));
165   result_m256 = _mm256_hadd_pd(result_m256, result_m256);
166
167   _mm256_storeu_pd(resultUnpacked, result_m256);
168   vTangentZ = resultUnpacked[0] + resultUnpacked[2];
169 }
```

SPEKTRUM DER LICHTTECHNIK

Lichttechnisches Institut Karlsruher Institut für Technologie (KIT)

ISSN 2195-1152

Band 1 Christian Jebas
Physiologische Bewertung aktiver und passiver Lichtsysteme im Automobil. 2012
ISBN 978-3-86644-937-4

Band 2 Jan Bauer
Effiziente und optimierte Darstellungen von Informationen auf Grafikanzeigen im Fahrzeug. 2013
ISBN 978-3-86644-961-9

Band 3 Christoph Kaiser
Mikrowellenangeregte quecksilberfreie Hochdruckgasentladungslampen. 2013
ISBN 978-3-7315-0039-1

Band 4 Manfred Scholdt
Temperaturbasierte Methoden zur Bestimmung der Lebensdauer und Stabilisierung von LEDs im System. 2013
ISBN 978-3-7315-0044-5

Band 5 André Domhardt
Analytisches Design von Freiformoptiken für Punktlichtquellen. 2013
ISBN 978-3-7315-0054-4

Band 6 Franziska Herrmann
Farbmessung an LED-Systemen. 2014
ISBN 978-3-7315-0173-2

Band 7 Simon Wendel
Freiform-Optiken im Nahfeld von LEDs. 2014
ISBN 978-3-7315-0251-7

Band 8 Carmen Kettwich
Ablenkung im Straßenverkehr und deren Einfluss auf das Fahrverhalten. 2014
ISBN 978-3-7315-0288-3

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 9 Steffen Michenfelder
Konzeption, Realisierung und Verifikation eines automobilen Forschungsscheinwerfers auf Basis von Digitalprojektoren. 2015
ISBN 978-3-7315-0301-9

Band 10 Celal Mohan Ögün
Surface wave driven molecular low pressure plasmas for general lighting. 2016
ISBN 978-3-7315-0464-1

Band 11 Theresa Bonenberger
LED Farbmischung mit chaotischen Lichtleitern. 2016
ISBN 978-3-7315-0480-1

Band 12 Michael Schöne
Diffraktive Optiken im Automobil: Achromatisierung, Athermalisierung, Formung von Scheinwerferlichtverteilungen. 2017
ISBN 978-3-7315-0613-3

Band 13 Tobias Werner
Simulation, Aufbau und Charakterisierung von autostereoskopischen Display-Systemen im Fahrzeugbereich. 2017
ISBN 978-3-7315-0617-1

Band 14 Christian Herbold
Entwicklung und Herstellung naturähnlich verzweigter Kühlkörper für LED-Systeme. 2017
ISBN 978-3-7315-0635-5

Band 15 Carsten Gut
Laserbasierte hochauflösende Pixellichtsysteme. 2018
ISBN 978-3-7315-0710-9

Band 16 Annie Shalom Samji Isaac Chandra
Intelligent Freeform Deformation for LED Illumination Optics. 2018
ISBN 978-3-7315-0741-3

Band 17 Ingo Rotscholl
Spectral near field data of LED systems for optical simulations. 2018
ISBN 978-3-7315-0750-5

Band 18 Inca Leopoldo Sayanca
Sensorfusion zur Kompensation von Messfehlern bei kamerabasierter Farbverteilungsmessung. 2018
ISBN 978-3-7315-0830-4

Band 19 Benjamin Schulz
Weiterentwicklung der Beleuchtungseinheit LED-basierter Projektionssysteme. 2019
ISBN 978-3-7315-0865-6

Band 20 Said Omerbegovic
Prädiktive Lichtfunktionen für volladaptive Scheinwerfersysteme. 2019
ISBN 978-3-7315-0875-5

Band 21 Patric Jahn
Bewertungsmodell zur Evaluation hochauflösender, lichtbasierter Fahrerassistenzsysteme. 2020
ISBN 978-3-7315-1009-3

Band 22 Maximilian Barthel
Aufmerksamkeitslenkung mithilfe Innenraumbeleuchtung im Automobil. 2020
ISBN 978-3-7315-1011-6

Band 23 Melanie Helmer
Methode zur Messung des Einflusses von Lichtimpulsen auf die visuelle Leistungsfähigkeit. 2021
ISBN 978-3-7315-1013-0

Band 24 Marina Budanow
Entwicklung eines lichtbasierten Fahrerassistenzsystems. 2020
ISBN 978-3-7315-1018-5

SPEKTRUM DER LICHTTECHNIK ISSN 2195-1152

Band 25 Dennis Zimmermann
Verbesserung der Prozesskette zur Herstellung mikrostrukturierter Linsen für automobile Scheinwerfer. 2021
ISBN 978-3-7315-1058-1

Lichttechnisches Institut
Karlsruher Institut für Technologie (KIT)

SPEKTRUM DER LICHTTECHNIK

Der Übergang des hellen in den dunklen Bereich einer automobilen Scheinwerfer-Lichtverteilung unterliegt gesetzlichen Auflagen. Die Auflagen beschränken, wie scharf der Übergang sein darf. Scheinwerfer-Projektionssysteme erzeugen zunächst eine zu scharfe Hell-Dunkel-Grenze. Um diese aufzuweichen, wird die Lichtaustrittsseite der Linse mit einer streuenden Struktur versehen, deren Amplitude im Bereich weniger Mikrometer liegt. Diese Arbeit befasst sich mit der Auslegung, Herstellung und Qualitätskontrolle mikrostrukturierter Linsen. Es wird ein Algorithmus beschrieben, um glatte Linsenflächen zu strukturieren. Die Datenweitergabe zur Herstellung von Stahlwerkzeugen für den Spritzgussprozess wird erläutert. Um die Werkzeuge zu qualifizieren, wird ein „Reverse Engineering"-Prozess vorgestellt.

ISSN 2195-1152
ISBN 978-3-7315-1058-1