Kopplung von Raytracern mit Spezialsoftware

Bernhard Michel, Monika Kroneberger, Robert Hermann,
Hembach Photonik GmbH, Rednitzhembach

Die Entwicklung optischer Systeme basiert heute oft auf Strahlverfolgung mit kommerziellen Raytracing-Programmen. Über ihre grafischen Benutzeroberflächen können Standardaufgaben einfach und effizient gelöst werden. Bei komplexeren Aufgabestellungen kann der Nutzer auf integrierte Skriptsprachen zurückgreifen. Maximale Flexibilität für kreative Raytracing-Lösungen sowie für die Automatisierung von Design und Analyse erhält man jedoch, indem man die Software über die dafür vorgesehenen Schnittstellen mit spezialisierten Programmen, Skript-Routinen und dynamischen Link-Bibliotheken koppelt.

1 Marktumfeld

Für die computergestützte Entwicklung optischer Systeme hat sich Raytracing (Strahlverfolgung) als die Standardmethode durchgesetzt. Während so genannte sequenzielle Raytracer für das Design von Abbildungssystemen eingesetzt werden, eignen sich nichtsequenzielle Raytracer für allgemeinere Aufgaben, in denen die physikalisch korrekte Modellierung im Vordergrund steht. Leistungsfähige kommerzielle Software-Pakete unterstützen den Optikentwickler bei seiner Arbeit, z.B. ASAP, APEX, FRED, LightTools, LucidShape, SPEOS und ZEMAX.

Der Markt solcher nichtsequenzieller Raytracer hat sich in den letzten zwanzig Jahren erheblich ausgedehnt, von der eigentlichen optischen Industrie, Luft-, Raumfahrt und Automobilindustrie ausgehend bis zu Branchen, die der eingefleischte Optiker gar nicht erwartet – eben überall dorthin, wo Licht manipuliert werden muss. Beispielsweise ist „weiße Ware" (Waschmaschinen, Spülmaschinen etc.) mit immer aufwendigeren Anzeigeelementen und Warnleuchten ausgestattet, die gemäß Spezifikation beleuchtet werden müssen. Erreicht wird dies durch Hinterleuchtungen und Lichtleiter – bei beidem handelt es sich oft um anspruchsvolle Entwicklungsaufgaben.

Die kommerziellen Raytracer haben sich diesem weiten Markt angepasst und bieten meistintuitive grafische Benutzeroberflächen mit Dialogfeldern, Assistenten und anderen interaktiven Elementen. Das setzt aber voraus, dass der Hersteller der Software ein Stück weit die mögliche Anwendung vorausdenken und daher die Bedienung im Wesentlichen auf Standardaufgaben einschränken muss.

2 Flexibilität durch Spezialprogramme

Speziellere Aufgaben, die vom Hersteller nicht vorhergesehen wurden, erfordern sowohl mehr Flexibilität bei der Softwarebedienung als auch mehr Optik-Knowhow beim Nutzer. Zu diesem Zweck stellen viele kommerzielle Raytracer eine Skriptsprache zur Verfügung. Skripte bieten Vorteile wie Wiederverwendbarkeit, Modularisierung von Projekten und – extrem wichtig bei komplizierteren Projekten – gute Dokumentationsmöglichkeiten und Nachvollziehbarkeit.

Idealerweise ist eine Skriptsprache einfach zu erlernen, bietet alle typischen Strukturelemente einer Programmiersprache und erlaubt den Zugriff auf die eigentliche Softwarefunktionalität auf mehreren Ebenen: auf dem einfachsten Niveau können z.B. Geometrieelemente aufgebaut und manipuliert sowie einzelne Strahlen verfolgt werden, während auf höherem Niveau komplette Simulationen gesteuert und verwaltet werden können. In der Realität kommen viele Skriptsprachen diesem Ideal nicht nahe: entweder sie sind umständlich und „kryptisch", oder sie erlauben nur eingeschränkten Zugriff auf die Kernfunktionalität der Software.

Maximale Flexibilität und zugleich gute Bedienbarkeit bei der Nutzung von Raytracern erhält man aber, indem man sie über maßgeschneiderte Programme koppelt – soweit geeignete Schnittstellen überhaupt verfügbar sind. Grundsätzlich kann dies auf verschiedene Arten geschehen:

- Die Vorverarbeitung, z.B. die Erstellung der Geometrie des Systems, Definition und Erstellung der Lichtquellen etc., geschieht außerhalb der kommerziellen Raytracing Software. Bei sehr komplexen Modellen mit tausenden von Objekten oder speziellen Lichtquellen ist dies oft die einzig akzeptable Weise, einen Raytracer zu „füttern".
- Spezielle Skripte übernehmen die Nachverarbeitung der oft gigantische Datenmengen, die von Raytracing-Programmen schnell produziert werden, und die dann z.B. für die Toleranz- oder Störlichtanalyse durchforstet werden müssen, um schließlich zu einem kompakten Ergebnis zu kommen. Eine Gut/Schlecht-Analyse ist ein Extremfall davon: aufwendige Simulationen mit vielen Gigabyte von Daten münden schließlich in einem einzigen Bit: Spezifikationen „erfüllt" oder „nicht erfüllt". Ein weiteres Argument für externe Nachverarbeitung sind Auswerteverfahren, die vom Raytracer selbst nicht unterstützt werden.
- Sowohl die Vor- als auch die Nachverarbeitung wird durch Skripte gesteuert, die damit im IT-Sprachgebrauch als Client wirken, der den Raytracer als Server ansteuert. Eine solche Architektur bietet sich z.B. bei der Optimierung an, wenn die integrierten Optimierungs-Werkzeuge der Raytracer ungeeignet sind.
- Oder man dreht den Spieß um: Der Raytracer ruft für Aufgaben, die er nicht beherrscht, darauf spezialisierte Pro-

gramme auf. Ein Musterbeispiel hierfür ist ZEMAX: bei ausreichender Programmiererfahrung kann der Nutzer selbst dynamische Link-Bibliotheken (DLLs) in der Programmiersprache C erstellen und an ZEMAX anbinden, was die Funktionalität des Raytracers erheblich ausweitet. Auch andere Programme (z.B. ASAP) ermöglichen eine solche Funktionserweiterung.

Wie die meisten Software-Anbieter neigen auch Hersteller von Raytracern generell dazu, in ihre neuen Produktversionen immer mehr Funktionalität zu integrieren, um immer mehr (vorgefertigte) Lösungen anbieten zu können. Das ist sicher auch sinnvoll.

Insbesondere bei anspruchsvollen Projekten wird aber der oben vorgeschlagene Weg, die Funktionalität von Raytracern durch spezialisierte Programme zu erweitern, zunehmend attraktiv und effizient. Wenn einmal die Struktur eines kombinierten Ablaufes erstellt ist, ergeben sich erhebliche Vorteile. Software-Anpassungen können dann stattfinden, wenn der Nutzer sie braucht, und nicht erst dann, wenn es die Entwicklungsplanung des Herstellers vorsieht. Außerdem kann man bei der Erstellung der Skripte oder Programme in der Regel auf einen weit größeren Erfahrungsschatz an Lösungen zurückgreifen, als dies ein Software-Hersteller je bieten kann, denn für alle Standardprogrammiersprachen stehen umfangreiche, teilweise über Jahrzehnte gepflegte Software-Bibliotheken mit leistungsfähigen numerischen Algorithmen zur Verfügung, z.B. der GAMS-Index (*gams.nist.gov*), das Netlib Repository (*www.netlib.org*) oder SciPy (*www.scipy.org*).

3 Softwareauswahl und Anwendung

Welche externe Software eignet sich besonders? Zunächst könnte man hierzu eigene Software (z.B. C++-Programme) entwickeln, hat dann aber wieder das Problem mangelnder Flexibilität. Meist ist es günstiger, eine Standard-Skriptsprache zu nutzen, oder Standard-Software, die eine Skriptsprache unterstützt. Gerne wird hierzu Mathlab oder Mathematica eingesetzt. Im Sinne einer unternehmensinternen Lösung ist dies gut und sinnvoll, da es sich in beiden Fällen um leistungsfähige Programme handelt. Gerade bei Kooperationsprojekten können sich jedoch Probleme mit der Kompatibilität ergeben, da die Partner ja nun mehrere kommerzielle Softwareprodukte gemeinsam verwenden müssen. Unserer Meinung nach ist es hier günstiger, auf frei verfügbare Skriptsprachen zurückzugreifen. Wir schlagen aus mehreren Gründen die Skriptsprache Python [1] vor:

- Python ist "Freeware".
- Python ist sehr einfach zu lernen, bietet aber alle Strukturelemente moderner Programmiersprachen.
- Python bietet von vorneherein umfangreiche Softwarebibliotheken. Die Zusatz-Routinen NumPy und SciPy bieten eine hochwertige Bibliothek numerischer Algorithmen – z.B. auch für Optimierung, statistische Datenauswertung und Vieles mehr.
- Das Internet ist eine wahre Fundgrube für in Python implementierte Speziallösungen, da viele Wissenschaftler ihre Arbeiten auf diese Weise veröffentlichen.

Im Folgenden stellen wir zwei Anwendungen vor, bei denen die Kopplung von Raytracern mit spezialisierten Programmen Vorteile bringt. Die Beispiele sind bewusst einfach gehalten, zeigen aber das Wesentliche des Ansatzes. Als Raytracer verwenden wir ASAP und einen selbst entwickelten Raytracer. Die Grundidee ist aber gerade, dass die Methode auch bei den meisten anderen Raytracern funktioniert.

3.1 Beispiel Ringlichtleiter

Insbesondere in der Automobilbranche werden gerne Lichtleiter mit Auskoppelflächen nahezu beliebiger Form als ästhetische Designelemente eingesetzt. Den hier vorgestellten Ringlichtleiter haben wir einem aus dem Automobilbereich bekannten Vorbild nachempfunden, das dem Leser sicher schon öfters in Form eines Tagfahrlichts auf der Straße begegnet ist. Der Lichtleiter besteht aus einem Torus mit abgebogenen Enden, über die Licht von Leuchtdioden eingekoppelt wird (**Bild 1**). An der vom Betrachter abgewandten Seite (in Bild 1 unten) sorgt ein Ring mit prismenförmigen Einkerbungen für die Lichtauskopplung. Abstände und Anstellwinkel können für jedes Prisma individuell vorgegeben werden, sie beeinflussen die Stärke und Richtung der Lichtauskopplung. Die Aufgabe ist es, diese Struktur (Bild 1 oben) auf eine möglichst hohe Effizienz und homogene Leuchtdichte zu optimieren, indem sowohl die Prismenabstände als auch die -winkel als Funktion der Position auf dem Lichtleiter variiert werden (Bild 1 unten). Die Geometrie des Systems ist zwar sehr einfach, muss aber vollständig parametrisiert werden, um eine automatische Optimierung zu ermöglichen.

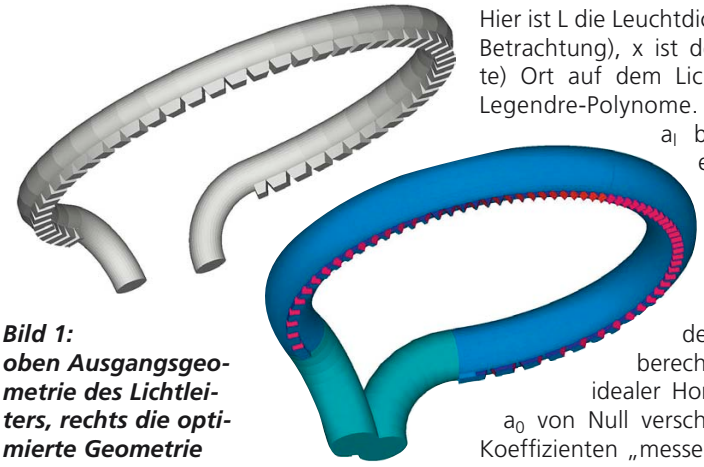

Bild 1: oben Ausgangsgeometrie des Lichtleiters, rechts die optimierte Geometrie

Bei der Simulation des Ringlichtleiters wird die Optimierung durch ein Python-Skript durchgeführt. Der erste Schritt ist die Erstellung eines parametrisierten Modells des Lichtleiters in ASAP. Aus Symmetriegründen reicht es, eine Hälfte des Lichtleiters zu modellieren. Unserer Erfahrung nach kann man den Prismenwinkel und den Prismenabstand als Funktion der Position gut durch Bezier-Polynome darstellen, deren Koeffizienten wir dann bei der Optimierung anpassen. Zur Bewertung der Lichtleiterqualität definiert man nun eine Gütefunktion, welche die ausgekoppelte Lichtmenge sowie die Homogenität der Lichtverteilung bewertet.

Da die Simulation mit Raytracing durchgeführt wird, sind die Ergebnisse immer statistisch verrauscht; außerdem haben die Prismen eine endliche Größe. Beides bewirkt feinskalige Inhomogenitäten der Lichtverteilung, allerdings ist das statistische Rauschen ein reines Artefakt der Simulation, und die Prismen-„Körnigkeit" interessiert uns nicht, da das Auge des Betrachters die feinen Strukturen aus der Ferne gar nicht auflösen kann. Um die interessierende grobskalige Homogenität zu bewerten, entwickeln wir die Lichtverteilung entlang des Lichtleiters nach Legendre-Polynomen [2]:

$$L(x) = \sum_{l=0}^{N} a_l P_l(x) \qquad \text{(Gl. 1)}$$

Hier ist L die Leuchtdichte (bei senkrechter Betrachtung), x ist der (geeignet skalierte) Ort auf dem Lichtleiter und P_l sind Legendre-Polynome. Die Koeffizienten a_l bestimmt man sehr effizient mit dem modifizierten Monte-Carlo-Simulationsverfahren "Importance Sampling" [3] direkt aus den mit dem Raytracer berechneten Strahlen. Bei idealer Homogenität wäre nur a_0 von Null verschieden; alle anderen Koeffizienten „messen" somit die Abwei-

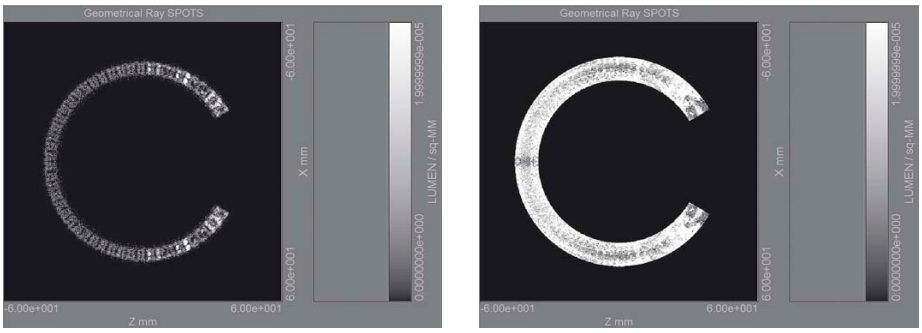

Bild 2: Beleuchtungsstärkeverteilung des Lichtleiters, a) Ausgangsgeometrie, b) optimierte Geometrie

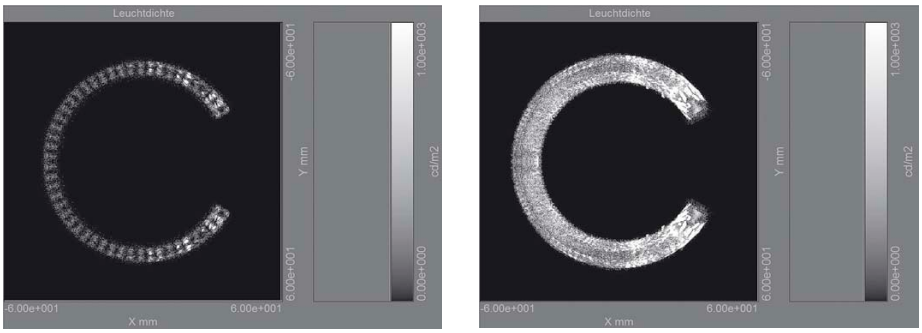

Bild 3: Leuchtdichtebild des Lichtleiters, aufgenommen mit einer virtuellen Leuchtdichtekamera, a) Ausgangsgeometrie, b) optimierte Geometrie

chung vom Idealzustand. Die Reihe wird bei einem relativ niedrigen Grad N ~5 – 6 abgebrochen, um die erwähnten feinskalige Inhomogenitäten weitgehend herauszufiltern.

Unserer Erfahrung nach kommt man mit diesem Trick, der sich sehr leicht in Python implementieren lässt (aber nur sehr schwer innerhalb eines Raytracers), mit deutlich weniger Strahlen und damit weniger Rechenzeit aus als bei den „normalen", vom Raytracer eingesetzten Verfahren. Zur Optimierung verwenden wir das so genannte "Simulated Annealing"; für diese Methode und für andere, eventuell genauso gute Optimierungsalgorithmen stehen fast fertige Python-Skripte als Freeware im Internet zur Verfügung. Damit wird der Raytracer, in unserem Falle ASAP, nur noch als "Server" für seine Kernaufgabe verwendet, alles Andere wird vom "Client" Python erledigt.

Sehr anschaulich ist das Ergebnis der Optimierung: **Bild 2a** zeigt die noch deutlich strukturierte Beleuchtungsstärkeverteilung der Startgeometrie, die sich außerdem zur Mitte des Lichtleiters hin stark abschwächt. Auch die Leuchtdichte (**Bild 3a**) weist die gleichen Muster auf. Nach der Optimierung ist die Beleuchtungsstärke nur noch von schwachen Strukturen gestört (Bild 2b), und es wird eine weitgehend homogene Leuchtdichte erreicht (Bild 3b).

Das vorgestellte Procedere lässt sich sehr gut auch für wesentlich komplexere Lichtleiter einsetzen und funktioniert prinzipiell mit allen gängigen Raytracern.

3.2 Beispiel Diffusor-Design

Eine typische Aufgabenstellung beim Design von Hinterleuchtungen (Backlights) und Lichtleiterstrukturen ist die Erzielung einer gleichmäßigen Leuchtdichte an der Auskoppelfläche. Dies lässt sich oft nur erreichen, indem man zusätzlich zu lichtlenkenden Elementen wie Reflektoren, Linsen und Fresnel-Strukturen noch Diffusoren einsetzt, die das Licht auch bezüglich der Richtungsverteilung mischen.

Neben Streufolien kommen hier zunehmend auch Volumenstreuer zum Einsatz, in denen kleine Partikel, die in ein Matrixmaterial eingelagert sind, die Lichtstreuung verursachen. Ihre Kenngrößen sind der diffuse und gesamte Transmissionskoeffizient – letzterer schließt auch das direkte, ungestreute Licht mit ein – sowie die Winkelverteilung des transmittierten Lichts.

Diese Größen sind abhängig von der Materialdicke, vom Brechungsindex sowohl der Matrix als auch der Teilchen sowie von deren Größenverteilung und Konzentration. Wenn wir von kugelförmigen Partikeln ausgehen, dann können die optischen Eigenschaften des Volumenstreuers im Rahmen der Mie-Theorie und Strahlungstransporttheorie berechnet werden („Vorwärtsrechnung"). Die meisten kommerziellen Raytracer sind hierzu in der Lage (siehe z.B. [4]), oder auch selbst-entwickelte Software, die zwar meist auf relativ einfache Geometrien beschränkt ist, aber maximale Flexibilität bei der Modellierung der Streuung bieten kann.

Es stellt sich nun die interessante Frage, ob man den Simulationsrichtung auch umkehren kann („Rückwärtsrechnung"): kann man bei vorgegebenen Kenngrößen des Diffusors auf eine passende Rezeptur für dessen Material zurückschließen? Dies ist nicht immer möglich, und die gefundene Rezeptur ist meist nicht eindeutig. Letzteres ist ein Vorteil, da es uns Freiheiten verschafft: wir suchen uns von den möglichen Designs die bezüglich Herstellbarkeit, Verfügbarkeit der Materialien und Kosten günstigsten heraus.

Um den Designspielraum zu erkunden, wurde in **Bild 4** der Einfluss verschiedener Größenverteilungen, Konzentrationen und Brechungsindizes der Partikel auf die Streuintensität untersucht, wobei für die Matrix PMMA und für die Mate-

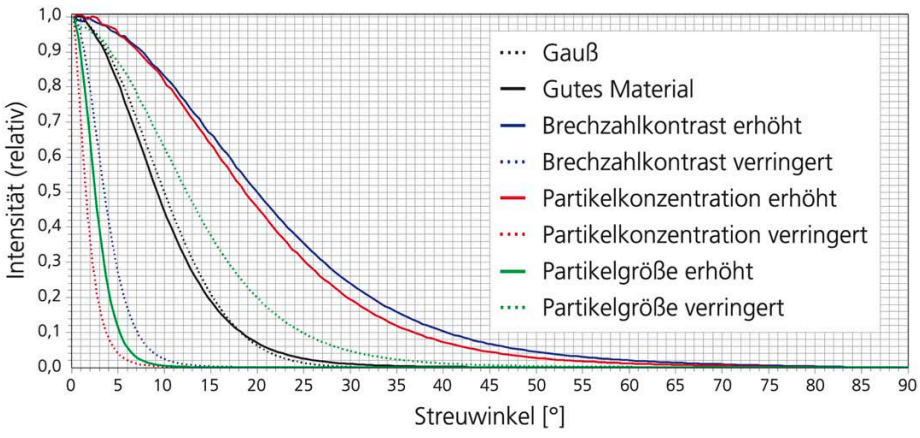

Bild 4: Streuintensität relativ zur optischen Eingangsleistung bei senkrechtem Lichteinfall und verschiedenen Diffusorrezepturen

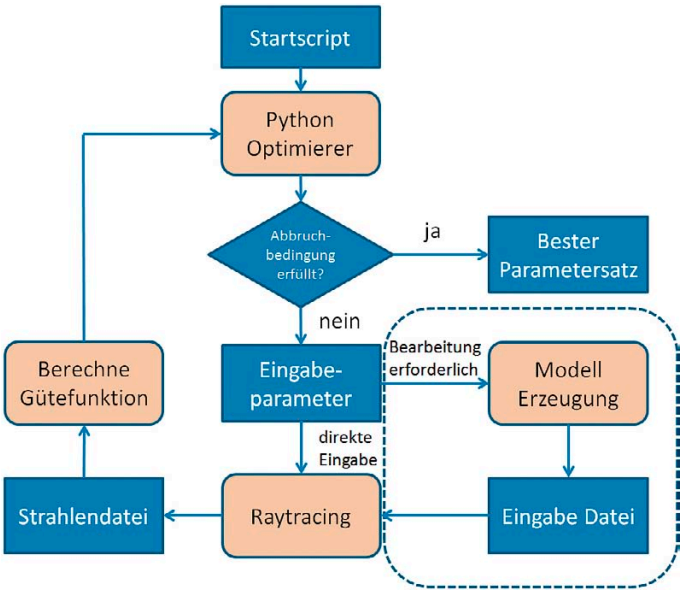

Bild 5: Flussdiagramm der Optimierung, Blöcke mit beigem Hintergrund sind austauschbar

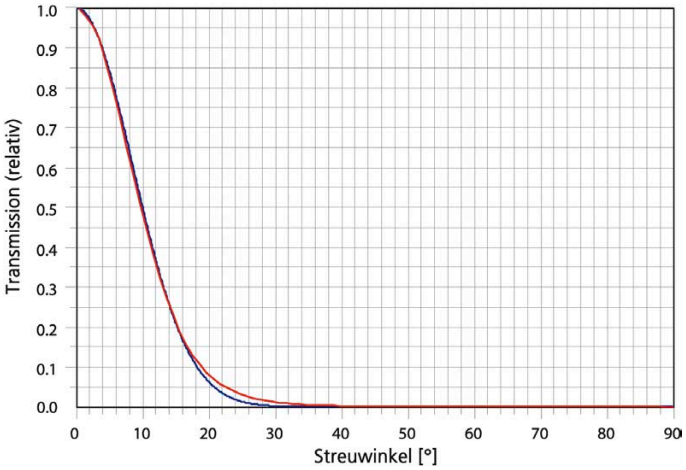

Bild 6: Gewünschte Lichtverteilung eines Diffusors (Dicke 2 mm, senkrechter Einfall) im Vergleich mit der Lichtverteilung eines optimierten realistischen Diffusormodells

rialdicke 4 mm angenommen wurden. Generell führen große Partikel zu einer stark vorwärts gerichteten Streuung, kleinere Teilchen und höhere Konzentrationen bewirken dagegen eine Verbreiterung der Streulichtverteilung. Einlagerungen mit hohem Brechzahlkontrast (z.B. TiO_2) führen zu deutlich erhöhter Streuung in große Winkel.

Die Modellierung des Diffusors haben wir auf verschiedene Weise realisiert:

1) Komplett als Modell in ASAP
2) Ein selbst entwickeltes Programm berechnet die so genannte Phasenfunktion der Streuung im Rahmen der Mie-Theorie. Ein ASAP-Modell liest diese Phasenfunktion über eine speziell erstellte DLL ein und führt dann die Simulation durch.
3) Komplette Modellierung mit einem eigenen Spezialprogramm.

Alle Modelle liefern die gleichen Ergebnisse; die maximale Flexibiltät bietet der zweite Ansatz, die maximale Geschwindigkeit der letzte. Die Berechnung der Phasenfunktion in eigener Software hat den weiteren Vorteil, dass das Ergebnis auch von anderen Raytracern (z.B. SPEOS) „verstanden" wird; so kann eine softwareunabhängige Bibliothek von z.B. Phasenfunktionen aufgebaut werden. Der Algorithmus zum Auffinden einer Rezeptur (Konzentration, Größenverteilung und Brechungsindex der Partikel) wurde in allen Fällen komplett in Python realisiert. Es sind lediglich kleine Änderungen in den Schnittstellen zum Raytracer erforderlich, um wahlweise einen der drei genannten Simulationsansätze umzusetzen.

Die Simulation verläuft wie in **Bild 5** gezeigt. Die Eingabeparameter werden zunächst an ein Programm übergeben, das die Phasenfunkton berechnet, die dann vom Raytracer eingelesen wird. Die berechnete Winkelverteilung wird mit der Sollverteilung verglichen und mit einer geeigneten Gütefunktion bewertet. Die Rezeptur wird daraufhin solange angepasst (neue Eingabeparametersätze), bis Soll- und Istwert innerhalb einer definierten Toleranz übereinstimmen.

Bild 6 zeigt als „Wunschverteilung" für die Transmission eine Gaußverteilung mit einer Breite von 25° (FWHM). sowie die beste Annäherung, die mit dem Simulationsmodell erzielt wurde. Die Transmission beträgt 90% und es wird kein ungestreutes Licht transmittiert. Das Ergebnis wurde durch relativ große Partikel (Durchmesser >20 µm) erzielt, deren Brechungsindex sich nur wenig von der Matrix unterscheidet. Allerdings sei auch erwähnt, dass für manche „Wunschverteilungen" (z.B. mit Doppelmaxima) keine Rezeptur gefunden werden kann. In diesem Falle sind Volumenstreuer als Diffusoren schlichtweg ungeeignet.

4 Fazit

Vorgestellt wurde die Strategie, bei der Optikentwicklung kommerzielle Raytracer als prinzipiell austauschbare „Rechenmaschinen" zu verwenden und deren Steuerung sowie alle höheren Aufgaben an spezialisierte Programme oder Skripte zu übergeben. Dies ist ein Erfolg versprechender Weg für viele Nicht-Standard-Aufgaben sowie anspruchsvolle Design- und Analyseprojekte. Tatsächlich besteht eine Marktlücke beim Angebot von „Sekundär-Software", mit der die Leistung und Flexibilität kommerzieller Raytracer erheblich gesteigert werden könnte.

Literaturhinweise:

[1] *www.python.org*
[2] *en.wikipedia.org/wiki/Legendre_polynomials*
[3] *en.wikipedia.org/wiki/Importance_sampling*
[4] B. Michel, P. Holcomb, *Simulation der Lichtstreuung in menschlicher Haut*, Photonik 3/2007, S. 60–63

Ansprechpartner:

Dr. Bernhard Michel
Geschäftsführer
Hembach Photonik GmbH
Finkenstr. 1-3
D-91126 Rednitzhembach
Tel. 09122/8899490
Fax 09122/8899499
eMail: bm@hembach-photonik.de
Internet: www.hembach-photonik.de

Monika Kroneberger
eMail: mk@hembach-photonik.de

Robert Hermann
eMail: rh@hembach-photonik.de

Optatec 2012: Stand 3.0/B51

www.photonik.de ▸Webcode **3003**