Kohärentes Metropolis-Lichttransportmodell mit Mutationen mit mehreren Versuchen

Jan Niklas Grieb*
Universität Göttingen

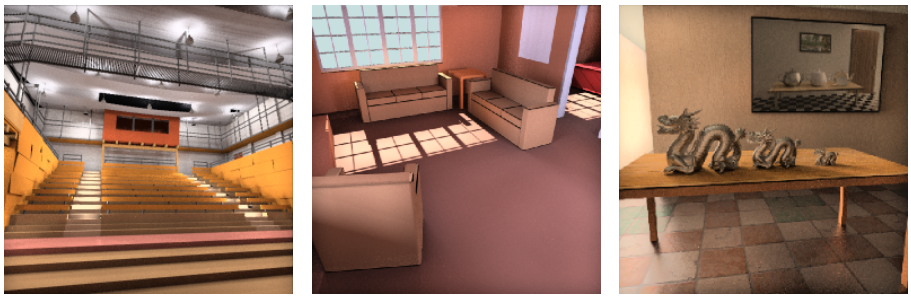


Abbildung 1: *Von Segovia et al. mit kohärentem* Metropolis Light Transport *jeweils in 10 Minuten gerenderte Bilder: Ein Theater, ein Zimmer mit Sitzecke und ein Raum mit Drachenskulptur und Spiegel.*

Zusammenfassung

Metropolis Light Transport (MLT) ist ein effizientes Verfahren zur Generierung von Computergrafiken mit realistischer, globaler Beleuchtung. Es wurde 1997 von Veach und Guibas vorgestellt und nutzt Monte-Carlo-Verfahren mit Metropolis-Hastings-Mutationen (MH). Es kann mit unterschiedlichsten Geometrien und Lichtverhältnissen fertig werden und ist äußerst robust.

Segovia, Iehl und Péroche haben nun ein Konzept vorgestellt, welches MLT in einer *kohärenten* Weise implementiert und sich daher für SIMD-Rechner-Architekturen eignet. Dadurch wird gleichzeitig die Leistung gesteigert und die Verträglichkeit des Algorithmus mit geometrischen Caches deutlich verbessert. Der neue Algorithmus besteht aus zwei Ebenen: in der ersten Ebene wird das bekannte MH-Verfahren zur Abdeckung der gesamten Szene verwendet. Im zweiten, untergeordneten Schritt wird jedes Sample kohärent mutiert und mit SIMD-Instruktionen vektoriell berechnet. Dabei wird auf den Multiple-Try-Algorithmus (MT) zurückgegriffen. Dieser wurde als Erweiterung von Metropolis-Hastings entwickelt, welche Effizienz-Hürden in vielen Dimensionen abbaut.

Durch die Stabilität und Leistungssteigerung rückt damit Echtzeit-MLT in Reichweite. Vorerst wird aber vor allem die Attraktivität von MLT für kommerzielle Renderer gesteigert.

Keywords: Monte-Carlo, Metropolis-Hastings, path tracing, coherent computation, metropolis light transport, multiple-try mutations

*e-mail: jan.grieb@googlemail.com

Inhaltsverzeichnis

1 Einführung

Vorweg will ich einige Bemerkungen zu realistischen Bildbeschreibungen in Computergrafiken machen, bevor hier die Grundkonzepte des Metropolis-Lichttransportmodells [Veach and Guibas 1997] vorgestellt werden. Im Hauptteil des Textes wird dann das von Segovia, Iehl und Péroche vorgestellte Konzept zur kohärenten Lichtpfadmutation beschrieben [Segovia et al. 2007].

1.1 Computergrafik

In der Computergrafik können die meisten verwendeten Algorithmen zur Bildsynthese in zwei große Klassen eingeteilt werden.

Rasterung/Rendering: Die Szene ist mit Polygonen modelliert, welche von der *Grafikpipeline* in eine 2D-Grafik umgesetzt werden. Die nötigen Geometrietransformationen, welche die Weltkoordinaten in die Koordinaten der Kameraebene transferiert und die eigentliche Rasterung der Polygone mit entsprechendem Shading wird in *Hardware* realisiert und ist schon lange in Echtzeit verfügbar.

Zur Optimierung werden *Clipping*- und *Culling*-Techniken angewendet, welche Objekte außerhalb des Sichtvolumens entfernen. Ein *Z-Buffer* wird zur Verdeckungsberechnung eingesetzt.

Strahlverfolgung: Ein konzeptionell völlig verschiedener Ansatz ist die Strahlverfolgung, welche seit den ersten Raycastern immer mehr an Beliebtheit gewinnt. Hierbei werden die Strahlen von der Kamera durch die Pixel der Bildebene in die Szene verfolgt (*Seh- oder Primärstrahlen*) und das erste Aufeinandertreffen von Strahl und Objekten der Szene bestimmt den Farbwert des Pixels. Erweitert um Schattenberechnungen, Sekundärstrahlen und komplizierte Beleuchtungsmodelle ist mittlerweile – um den Preis einer hohen Rechenintensivität – eine große Realtitätstreue von Strahlverfolgungssoftware (*Raytracern*) erreicht.

Mittlerweile ist auch mittels fortgeschrittener Konzepte das Echtzeitrendering möglich geworden, indem die GPU zur Parallelisierung verwendet wird.

1.1.1 Beleuchtungsmodelle

Auch bei den Beleuchtungsmodellen kann man grob in die einfache, schnelle Variante und die realistischere unterteilen.

Lokale Beleuchtung bezeichnet die einfachen Shadingalgorithmen, welche von Renderern eingesetzt werden, um die Helligkeiten der Pixel der gerasterten Polygone zu bestimmen. Die Farbe eines Pixels setzt sich demnach aus den Materialeigenschaften, der Textur, der Blickrichtung und den Lichtquellen zusammen.

Globale Beleuchtung benennt die Konzepte der Computer-Grafik, welche möglichst realistische Beleuchtungen von CG-Szenen erreichen will [Wikipedia 2009]. Dabei werden alle Ausbreitungsmöglichkeiten von Licht bestimmt, unter Berücksichtigung der geometrischen Gesetze und der Energieerhaltung.

1.1.2 Überblick über Simulationsmethoden

Allen Simulationsverfahren ist gemein, dass die Szene mit adäquaten geometrischen Objekten (Polygone, Kugeln, Splines, andere datentechnisch zu bearbeitende Geometrien) beschrieben sein muss. Wir wollen die Komplexität dieser Fragestellung der Modellierung hier außen vor lassen, da wir uns speziell mit der Bildgenerierung durch physikalisch motivierte Lichttransport-Verfahren beschäftigen wollen.

Raycasting. Das vor dem Aufkommen der Rendering-Gleichung (siehe Kap. 1.1.3) entstandene sogenannte *Raycasting* verfolgt die Lichtstrahlen vom Auge durch die Bildpunkte in die Szene. Dazu wird für jeden Bildpunkt eine Halbgerade vom Auge aus in die Szene „geschossen". Durch Schnittpunktsbestimmungen wird das dem Auge am Nächsten liegende Zusammentreffen mit einer Objektoberfläche bestimmt. Aus dem Auftreffpunkt ergibt sich der Farbwerte des Pixel aus den Materialdaten (Textur) und dem Winkel zwischen Blickrichtung und Oberflächennormalen (*shading*). Es wird also eine (unrealisitische) homogene Beleuchtung der Szene angenommen.

Dies ist ein sehr simples und limitiertes Verfahren. Erweiterungen, welche realistischere Lichtquellen und Schatten implementieren wollen, arbeiten z.B. mit Schattenstrahlen zu den Lichtquellen, um Schattenwürfe zu simulieren und die Helligkeit zu berechnen (siehe Abb. 2).

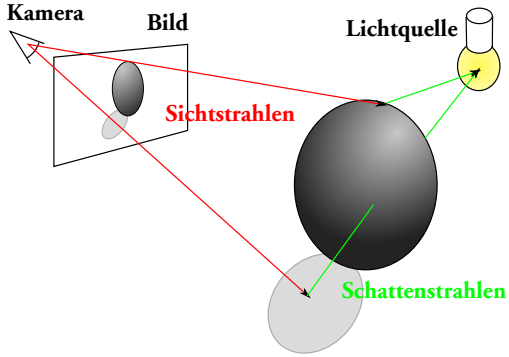


Abbildung 2: *Illustration des Raytracings mit Kamerastrahlen und Schattenstrahlen.*

Whitted-Raytracing. Für spiegelnde und durchlässige Materialien ist *rekursives Raytracing* (auch *Whitted-Raytracing*) nötig, welches Sekundärstrahlen aus den Fresnelschen Formeln (Brechungsgesetz) berechnet und diese weiterverfolgt [Whitted 1980].

Beim *Distribution Ray Tracing* [Cook et al. 1984] werden mehrere, *stochastisch* verteilte Strahlverfolgungen pro Pixel ausgelöst, welche sich über die Zeit, den Winkel und die Flächen verteilen sollen. Zudem können mehrere Sekundärstrahlen für diffuse Oberflächen ausgesendet werden. Diese Aufsplittung ist aber ineffizient, da hier sehr viel Rechenzeit für die exponentiell anwachsenden Anzahl von Sekundärstrahlen verloren geht, obwohl der Beitrag zum Pixel der Strahlen hoher Ordnung sehr gering ist. Bei *diffusem Raytracing* wird künstliche Unschärfe (*motion blur* für dynamische Szenen, Fokus- oder Blendenunschärfe) eingeführt, um weiche Schattenverläufe zu erhalten. Ineffizient, aber im Ergebnis besser ist Light Ray Tracing, bei dem die Strahlen an der Lichtquelle gestartet werden.

Pfadtracing. Pfadtracing will der Rendering-Gleichung [Kajiya 1986, Näheres siehe unten] gerecht werden, was mit Raytracing nach Whitted nicht möglich ist, da dort unendlich viele Sekundärstrahlen berechnet werden müssten. Daher wird im Pfadtracing, wie in dem später besprochenen Metropolis Light Transport, eine Monte-Carlo-Integration verwendet: es werden mehrere Strahlen pro Pixel gestartet, welche nach stochastischen Methoden weiter geleitet werden. Dies verbessert die Konvergenz. Der ausgesendete Lichtstrahl wird an spiegelnden und transparenten Oberflächen zufällig in eine der möglichen Richtungen weiter geschickt. Es bearbeitet folglich beliebige $L(D|S)^*E$-Pfade (zur Lichtpfadnotation siehe Kap. A.1). Im Gegensatz zum Distribution Ray Tracing findet kein Aufsplitten an den diffusen Oberflächen statt. Stattdessen werden sehr viele Strahlen für jeden Pixel gestartet.

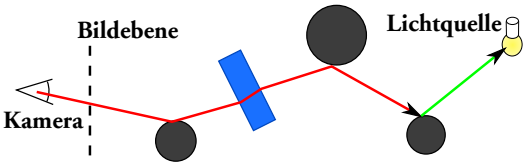


Abbildung 3: *Illustration des Pfadtracings mit Primär- und Sekundärstrahlen an diffusen und transparenten Objekten.*

Durch die Verwirklichung des globalen Beleuchtungsmodells wird auch die Synthese von realistischen Kaustiken möglich. Vorher wurde solche Muster (siehe Beispiel in Abb. 6), welche aus komplizierten Brechungen und Spiegelungen resultieren, nur durch spekulare Reflektionen oder kompliziert erscheinende Texturen simuliert.

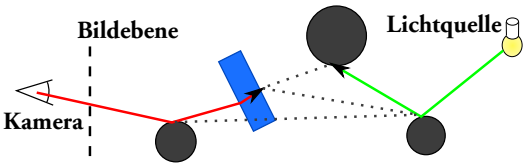
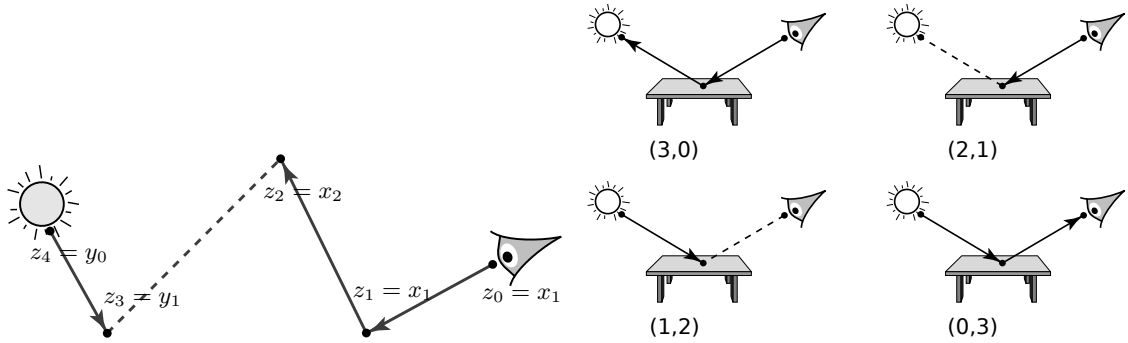


Abbildung 4: *Illustration des bidirektionalen Pfadtracings mit Kamera- und Lichtstrahlen. Es sind einige Kombinationsmöglichkeiten gezeigt..*

Bidirektionales Pfadtracing. Das *bidirektionale Pfadtracing* (BPT) verbessert die Effizienz, indem gleichzeitig und unabhängig Kamerastrahlen und Lichtquellenstrahlen konstruiert werden, welche dann kombiniert werden [Veach and Guibas 1994]. Dabei werden alle möglichen Teilpfadkombinationen in die Berechnung eingebunden (in Abb. 4 sind einige Kombinationsmöglichkeiten durch gepunktete Pfade angedeutet). Man charakterisiert die kombinierten Strahlen nach den Längen der Teilpfade als (s,t)-Wege, wobei s die Streckenabschnitte des Kamerapfads und t die

Streckenabschnitte des Lichtpfads angeben. Die Gesamtlänge ist dann $k = s + t - 1$. In Abb. 5(b) ist der Fall $k = 2$ illustriert.



(a) Birektionales Pfadtracing, hier an einem Beispiel aus einem Lichtpfad der Länge 1 und einem Kamerapfad der Länge 3.

(b) Alle (s, t)-Pfade der Gesamtlänge $k = 2$.

Abbildung 5: *Schematische Darstellung der Lichtpfadkonstruktion in bidirektionalen Pfadtracern – entnommen aus Veachs Doktorarbeit (1997), S. 299, Benennungskonvention angepasst.*

Multiple Importance Sampling. In seiner Doktorarbeit beschäftigt sich Veach mit der Frage, wie man die Monte-Carlo-Verfahren zur Lichttransportsimulation verbessern kann. Er geht auf sogenannte Multiple-Importance-Sampling-Modelle (MIS) ein, bei denen mehrere Importance-Sampling-Strategien gleichzeitig verwendet werden, um daraus ein optimales Endergebnis als gewichtete Summe zu ermitteln. Wird dies nicht gemacht, so ergibt sich, dass BPT sich sehr schlecht verhält.

Photon Mapping. Alternativ kann man das Pfadtracing mit *Photon Mapping* kombinieren, bei dem im Voraus die Beleuchtung der Szene durch von den Lichtquellen ausgesendete Lichtpartikel bestimmt wird. Diese werden am Auftreffpunkt in den Photon-Maps gespeichert und im eigentlichen Rendering wiederverwendet. Dieses Verfahren hat je nach Geometrie Vor- und Nachteile, da es nicht erwartungstreu, sondern nur konsistent ist (für diese Begriffe siehe Kap. A.3).

Alternatives Verfahren: Radiosity. Das Radiosity-Verfahren entfernt sich vom traditionellen Beobachter-zentrierten Rendering und berechnet die gesamte Beleuchtungsszene aus den Energieflusss-Gesetzen der Physik. Daher bietet es sich besonders für indirekt beschienene Szenen an. Jedoch können nur bestimmte Geometrien und ausschließlich ideal diffuse Oberflächen berechnet werden.

Abbildung 6: *Renderbild einer Glasskulptur mit Kaustiken. Quelle: Wikipedia.*

1.1.3 Rendering-Equation

1986 hat Kajiya die Rendering-Gleichung aufgestellt [Kajiya 1986], welche die physikalischen Gesetze berücksichtigend die Lichtausbreitung beschreibt. Diese ist die fundamentale Integralgleichung, welche von den Rendering-Algorithmen gelöst werden muss. Somit ist diese Gleichung das mathematische Fundament aller Methoden, welche globale Beleuchtung implementieren.

Eine Darstellung der Rendergleichung lautet[1]:

$$L(x' \to x'') = L_e(x' \to x'') + \int_{\mathcal{M}} L(x \to x') f_s(x \to x' \to x'') G(x \leftrightarrow x') \, \mathrm{d}A(x), \tag{1}$$

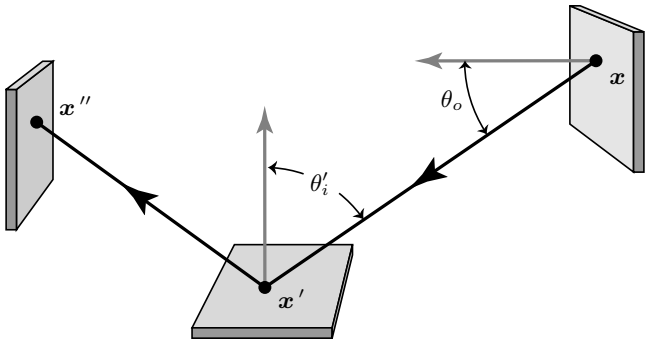
wobei L der ausgehende Licht-Energiefluss, L_e der emittierte Lichtfluss und G der geometrische Faktor,

$$G(x \leftrightarrow x') = V(x \leftrightarrow x') \frac{|\cos\theta_0 \cos\theta_i'|}{\|x - x'\|}, \tag{2}$$

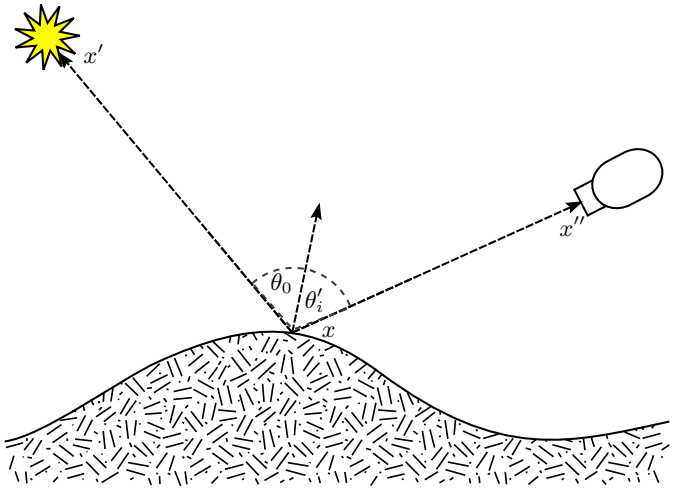
welcher im Wesentlichen Skalarprodukte zwischen Verbindungslinien und Normalenvektoren darstellt. Dieser wird mit dem visuellen Faktor $V(x \leftrightarrow x')$ gewertet, welcher 1 ist, falls eine direkte Sichtverbindung zwischen x und x' besteht, und ansonsten 0. Schließlich ist f_s die bidirektionale Reflektanzverteilungsfunktion (BRDF, siehe unten) des Materials.

Die Oberfläche am Punkt x hat eine Normale, welche mit $x \to x'$ den Winkel θ_0 einschließt. Die Normale an x' schließt mit dieser Richtung den Winkel θ_i' ein. Für eine Illustration der Geometrie sei auf Abb. 7(a) verwiesen. Abschließend muss gesagt werden, dass über die Vereinigung $\mathcal{M}$ aller Oberflächen der Szene mit dem Lebesque-Maß $\mathrm{d}A$ integriert wird.

[1] Hier wird die Notation aus [Veach 1997, vgl. ch. 8] verwendet, welche von [Segovia et al. 2007] übernommen wurde.



(a) entnommen aus Veachs Doktorarbeit (1997), S. 221



(b) basiert auf Wikipedia:BRDF (en)

Abbildung 7: *Schematische Darstellung der Geometrie zur Rendergleichung.*

Eine BRDF (*bidirectional reflectance distribution function*) $f_s(x \to x' \to x'')$ gibt an, wie sich Lichteinfluss E_i und -ausfluss L_r an einer Oberfläche unter den gegebenen Winkeln verhalten [Wikipedia 2010a]:

$$f_s(x \to x' \to x'') = f_s(x \to x', x \to x'') = \frac{\mathrm{d}L_r(x \to x'')}{\mathrm{d}E_i(x \to x')} = \frac{\mathrm{d}L_r(x \to x'')}{L_i(x \to x') \cos\theta_0 \, \mathrm{d}(x \to x')}, \tag{3}$$

wobei θ_0 der Winkel zwischen Einfallsvektor $x \to x'$ und der Oberflächen-Normalen ist (vgl. Abb. 7(b)). Für lichtdurchlässige Materialien gibt die *bidirectional transmittance distribution function* (BTDF) an, wie sich Lichtein- und -ausfluss verhalten. Zusammen ergeben diese beiden Funktionen die BSDF (*birectional scattering distribution function*).

Erweiterungen dieses Konzepts erlauben auch Streuung in der Oberfläche (*subsurface scattering*); dies führt aber für diese Arbeit zu weit.

Es wird eine Vielzahl von Algorithmen zur Lösung der Rendergleichung verwendet, darunter Radiosity und das Pfadtracing, welches Monte-Carlo-Verfahren verwendet. Eine neuere Methode ist das bidirektionalen Raytracing. Im Gegensatz zu den früheren Verfahren (Rasterizer, einfaches Raytracing) sind nun die Bedingungen von globaler Beleuchtung erfüllt, was bspw. die Simulation von Kaustiken wie in Abb. 6 möglich macht.

Im Hinblick darauf, dass die im Hauptteil vorgestellte Arbeit auf Metropolis-Hastings-Algorithmen aufbaut, sollen gleich Monte-Carlo-Verfahren und der *Metropolis Light Transport*-Algorithmus in Kap. 1.2 kurz eingeführt werden.

1.2 Monte-Carlo

Wir wollen gleich betrachten, inwiefern Monte-Carlo-Integrationsmethoden zur Lösung der Rendering-Gleichung beitragen. Doch zunächst einige einleitende Worte zur Monte-Carlo-Integration (kurz MC) an sich.

Es ist aus den Grundvorlesungen bekannt, dass man durch Ziehen von N Zufallszahlen $\{\boldsymbol{X}_i\}$ nach einer Wahrscheinlichkeitsdichtefunktion (*probability density function*, PDF; siehe Kap. A.2) aus dem Integrationsbereich M ein Integral numerisch lösen kann. Das Gebiet M habe ein Maß $\mathrm{d}\mu(x)$.

Genauer können wir das Integral

$$I = \int_M f(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) \tag{4}$$

dadurch lösen, indem wir als Annäherung

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\boldsymbol{X}_i)}{p(\boldsymbol{X}_i)} =: \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{Y}_i, \tag{5}$$

berechnen[2]. Denn dann gilt, dass wir im Mittel das richtige Ergebnis erwarten, d.h.

$$E\left[F_N\right] = I. \tag{6}$$

In solchen Fällen spricht man von einem *erwartungstreuen* Schätzer (siehe Kap. A.3).

Ein Maß für den *Fehler* δI, der dabei in Kauf genommen werden muss, ist die Standardabweichung. Mit $V\left[Y_i\right] = V\left[Y\right] = E\left[Y^2\right] - E\left[Y\right]^2$ (d.h. $Y = F_1$), sieht man, dass sich die Standardabweichung, also die Wurzel der mittleren quadratischen Abweichung, zu

$$\delta I = \sqrt{V\left[F_N\right]} = \sqrt{\frac{V\left[Y\right]}{N}} \propto N^{-\frac{1}{2}} \tag{7}$$

ergibt. Diese einfachste Variante der Monte-Carlo-Integration führt also schon auf beliebig genaue Ergebnisse, vorausgesetzt die Anzahl an Ziehungen ist groß genug. Das Besondere ist, dass dies auch für viele Dimensionen gilt, und diese Methode umgeht somit den sogenannten „Fluch der Dimensionalität“ (*curse of dimensionality*)[3]. Doch gibt es eine einfache und äußert zweckmäßige Verbesserung, das *Importance Sampling*.

Es ist nämlich offensichtlich günstig, genau dort viele Zufallszahlen zu ziehen, an denen sich die Funktion von 0 unterscheidet. Somit wäre statt einer beliebigen Verteilung der Zufallszahlen eine PDF der Form

$$p(\boldsymbol{x}_i) \propto f(\boldsymbol{x}_i) \tag{8}$$

wünschenswert.

Der Vorteil ist, dass dies die Varianz $V\left[\tilde{Y}\right]$ minimiert, so dass nach Gl. (eq:7) der Fehler der Monte-Carlo-Approximation verbessert wird. Doch wie finden wir Zufallszahlen einer entsprechende Verteilung $p(\boldsymbol{x})$, welche eventuell nicht für beliebige $\boldsymbol{x}$ berechenbar ist?

1.2.1 Metropolis-Hastings und Markov-Ketten

Wir wollen nun ergründen, wie der Metropolis-Algorithmus zur Generierung von Zufallszahlen gemäß einer PDF $p(\boldsymbol{x})$ funktioniert. Dieser benutzt Ketten von Realisierungen einer Zufallsvariablen

$$\ldots \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1} \to \boldsymbol{x}_{i+2} \to \ldots, \tag{9}$$

wobei die Übergangswahrscheinlichkeit $K(\boldsymbol{x}_i \to \boldsymbol{x}_{i+1})$ nur vom aktuellen Wert und der Sprungweite, und damit nicht von der Vorgeschichte, abhängt. Solche Zufallsprozesse werden Markov-Ketten genannt.

Detailliertes Gleichgewicht. Fordert man, dass ein Markov-Prozess in einem stationären Zustand verweilt, d.h. dass die Wahrscheinlichkeit $P(\boldsymbol{x}_i, i)$ im $i$ten Schritt den Zustand $\boldsymbol{x}_i$ anzutreffen die Bedingung

$$P(\boldsymbol{x}, i+1) = P(\boldsymbol{x}, i) =: P(\boldsymbol{x}) \tag{10}$$

erfüllt, dann ergibt sich die folgende *detaillierte Gleichgewichtsbedingung* (*detailed balancing*) an die Übergangswahrscheinlichkeiten

$$p(\boldsymbol{x}_i) K(\boldsymbol{x}_i \to \boldsymbol{x}_j) = p(\boldsymbol{x}_j) K(\boldsymbol{x}_j \to \boldsymbol{x}_i). \tag{11}$$

Ist dies nichttrivial[4] erfüllt, dann reproduziert die stationäre Markov-Kette die gesuchte Verteilung:

$$P(\boldsymbol{x}) \propto p(\boldsymbol{x}). \tag{12}$$

[2] Ein einfaches Beispiel ist $\mathrm{d}\mu(x) = \mathrm{d}^3 x$ und $p(\boldsymbol{x}) = 1/V$, nämlich gleichverteilte Zufallszahlen in einem 3D-Raum, welches ein Volumen V habe. Dann gilt: $F_N = \frac{V}{N} \sum_{i=1}^{N} f(\boldsymbol{X}_i)$.

[3] Hier ist die Konvergenz mit $\mathcal{O}\left(1/\sqrt{N}\right)$ dimensionsunabhängig! Hier ist die Konvergenz nach dem starken Gesetz der großen Zahlen sogar dann gegeben, wenn formell $E\left[f^2\right] = \infty$ gilt [Veach 1997].
Gewöhnliche Quadraturmethoden zur numerischen Berechnung von s-dimensionalen Integralen (Gauß, etc.) haben nach dem Theorem von Bakhvalov eine schlechtere Konvergenz von $\mathcal{O}\left(N^{-1/s}\right)$ [Veach 1997, S. 33].

[4] Die triviale Lösung ist der konstante Zustand für $T(i \to j) = \delta_{ij}$. Ebenso muss die Ergodizität des Markov-Prozesses sichergestellt sein: für jeweils zwei Zustände muss es eine Kette geben, deren Gesamtübergangswahrscheinlichkeit nicht verschwindet.

Metropolis-Übergangswahrscheinlichkeit. Es soll nun ein Algorithmus gefunden werden, der Markov-Ketten gemäß der genannten Gleichgewichtsbedingung liefert. Dazu wählt man sich bei jedem Schritt einen Kandidaten gemäß einer Vorschlagswahrscheinlichkeitsverteilung $T(\boldsymbol{x}_i, \boldsymbol{x}_j)$, welche symmetrisch (und unabhängig von der Zielverteilung) gewählt sein sollte. Nun akzeptiert man den Vorschlag gemäß der Akzeptanzwahrscheinlichkeit

$$W(\boldsymbol{x}_i \to \boldsymbol{x}_j) = \min\left(1, \frac{p(\boldsymbol{x}_j)}{p(\boldsymbol{x}_i)}\right), \tag{13}$$

in welche die gewünschte Zielverteilungsfunktion $p(\boldsymbol{x})$ einfließt. Dieses Schema, welches von Nicholas Metropolis erstmals vorgestellt und nach ihm benannt wurde, hat somit die folgende Übergangswahrscheinlichkeit:

$$K(\boldsymbol{x}_i \to \boldsymbol{x}_j) = T(\boldsymbol{x}_i \to \boldsymbol{x}_j) W(\boldsymbol{x}_i \to \boldsymbol{x}_j). \tag{14}$$

Dies erfüllt die detaillierte Gleichgewichtsbedingung, so dass man, bei beliebigem Startwert, nach einer gewissen Einschwingphase (*start-up bias*) eine stabile Markov-Kette erhält, welche $p(\boldsymbol{x})$ realisiert. Dieser Bias erklärt sich dadurch, dass, da $p(\boldsymbol{x})$ ja nicht zur Gänze berechenbar sein muss, auch kein guter Anfangswert $\boldsymbol{x}_0$ der Ketten gefunden werden kann.

Ein Beispiel-Algorithmus ist in Alg. 1 vorgestellt.

Algorithmus 1 Metropolis-Hastings-Algorithmus

1) Benenne einen Startwert $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_0$.
2) Schlage ein neues $\boldsymbol{x}^*$ gemäß $T(\boldsymbol{x}_i \to \boldsymbol{x}^*)$ ausgehend vom aktuellen $\boldsymbol{x}_i$ vor.
3) Berechne $w = p(\boldsymbol{x}^*)/p(\boldsymbol{x}_i)$.
   i) Für $rand(0,1) \leq w$ akzeptiere den Vorschlag, d.h. $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}^*$.
   ii) Ansonsten behalte den alten Wert: $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i$.
4) Füge das Sample $\boldsymbol{x}_{i+1}$ dem Histogramm zu.
5) Setze $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i+1}$ und beginne erneut bei 2).

Metropolis-Hastings-Verallgemeinerung. Genauer gesagt ist der beschriebene Prozess bereits eine Verallgemeinerung des von Nicholas Metropolis aufgestellten Verfahrens. Dieser arbeitete, entsprechend der ihm damals gegeben Anforderungen, nur mit speziellen Verteilungsfunktionen, nämlich Boltzmann-Verteilungen.

Man kann auch eine Akzeptanzwahrscheinlichkeit für nicht-symmetrische Sprungwahrscheinlichkeiten ($T(\boldsymbol{x}_i \to \boldsymbol{x}_j) \neq T(\boldsymbol{x}_j \to \boldsymbol{x}_i)$) definieren:

$$W(\boldsymbol{x}_i \to \boldsymbol{x}_j) = \min\left(1, \frac{p(\boldsymbol{x}_j) T(\boldsymbol{x}_j \to \boldsymbol{x}_i)}{p(\boldsymbol{x}_i) T(\boldsymbol{x}_i \to \boldsymbol{x}_j)}\right). \tag{15}$$

Dies findet jedoch seltener Verwendung, ist hier aber von Bedeutung, da die BSDFs, welche in der Rendergleichung (siehe Kap. 1.1.3) vorkommen, nicht-symmetrisch sind.

Generell sollten solche Approximationsverfahren keinen Bias aufweisen, was bedeutet, dass der mittlere approximierte Wert sich nicht von dem Originalwert unterscheidet. Die MC-Integration mit statistisch unabhängigen Samples ist ein solches Verfahren. Etwas schwächer sind konsistente Verfahren, bei denen diese Abweichung mit $N \to \infty$ verschwindet (vgl. Kap. A.3).

1.3 Metropolis Light Transport

Eine solche Metropolis-Methode wurde zur Verbesserung der Monte-Carlo-Integrationsmethoden in der Beleuchtungssimulation vorgeschlagen [Veach and Guibas 1997]. Bevor wir zur Besprechung dieser speziellen Methode kommen, wollen wir eine Einführung in die MC-Methoden in der Computergrafik geben.

1.3.1 Monte-Carlo-Integrationsmethoden

Veach hat in seiner Doktorarbeit die Grundlagen für den integralen Lichttransport gelegt [Veach 1997]. Da die Renderinggleichung (Gl. (eq:1)) keine Integration einer bekannten Größe, sondern eine Integralgleichung ist, muss zur Anwendung von Monte-Carlo-Verfahren erst ein modifizierter Formalismus gefunden werden, der hier vorgestellt werden soll.

Dazu wird die Messgleichung für einen hypothetischen Sensor, welcher von Licht der Intensität[5] $L(\boldsymbol{x} \to \boldsymbol{x}')$ beschienen wird und darauf mit Gewicht $W_e(\boldsymbol{x} \to \boldsymbol{x}')$ (Responsivität) reagiert, aufgestellt:

$$I = \int_{\mathcal{M}\times\mathcal{M}} W_e(\boldsymbol{x} \to \boldsymbol{x}') L(\boldsymbol{x} \to \boldsymbol{x}') G(\boldsymbol{x} \leftrightarrow \boldsymbol{x}) \,\mathrm{d}A(x) \,\mathrm{d}A(x'). \tag{16}$$

Hier wurde eine Äquivalenztransformation durchgeführt, um die Integration über die Winkel in eine Oberflächenintegration zu überführen, welche das Verfahren durchschaubarer und die Gleichungen übersichtlicher werden lässt.

[5] Wir beschränken uns auf frequenzunabhängige Intensitäten. Der Übergang zu frequenzabhängigen Intensitäten kann einfach vollzogen werden.

Pfadintegral-Formalismus. Diese Gleichung muss in ein simples Integral überführt werden, wobei sich Veach des Pfadintegral-Formalismus bedient. Dazu schreibt man

$$I = \int_{\Omega} f(\overline{\boldsymbol{x}}) \, \mathrm{d}\mu(\overline{\boldsymbol{x}}), \tag{17}$$

wobei $f(\overline{\boldsymbol{x}})$ die Messungsbeitragsfunktion des Pfades $\overline{\boldsymbol{x}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots)$ und μ das Maß der Menge der endlichen Transportwege Ω ist:

$$\Omega = \bigcup_{k=1}^{\infty} \Omega_k, \quad \mu(G) = \sum_{k=1}^{\infty} \mu_k(G \cap \Omega_k), \tag{18}$$

wobei Ω_k die Menge der Pfade mit Länge k ist, so dass

$$\mathrm{d}\mu_k(\overline{\boldsymbol{x}}) = \mathrm{d}A(\boldsymbol{x}_0) \ldots \mathrm{d}A(\boldsymbol{x}_k). \tag{19}$$

Da die Transportwege des Lichts in unendlichen Variationen auftreten, muss ein Pfadintegralausdruck gewonnen werden, um $f(\overline{\boldsymbol{x}})$ angeben zu können [Veach 1997, ch. 8].

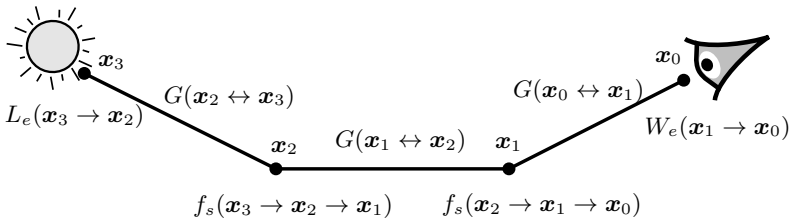


Abbildung 8: *Schematische Darstellung zur Berechnung von f, hier an einem Beispiel der Länge 3 ($k = 3$) – entnommen aus: Veachs Doktorarbeit 1997, S. 224, Benennungskonvention angepasst.*

Eine rekursive Expansion liefert[6]:

$$\begin{aligned} I = \sum_{k=1}^{\infty} \int_{\mathcal{M}^{k+1}} \Bigg[& L_e(\boldsymbol{x}_k \to \boldsymbol{x}_{k-1}) G(\boldsymbol{x}_k \leftrightarrow \boldsymbol{x}_{k-1}) W_e(\boldsymbol{x}_1 \to \boldsymbol{x}_0) \\ & \cdot \left(\prod_{i=1}^{k-1} f_s(\boldsymbol{x}_{i+1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i-1}) G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) \right) \mathrm{d}A(\boldsymbol{x}_0) \ldots \mathrm{d}A(\boldsymbol{x}_k) \Bigg]. \end{aligned} \tag{20}$$

Als weitere Veranschaulichung seien hier die ersten Entwicklungsterme gegeben:

$$\begin{aligned} I = & \int_{\mathcal{M}^2} L_e(\boldsymbol{x}_1 \to \boldsymbol{x}_0) G(\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_0) W_e(\boldsymbol{x}_1 \to \boldsymbol{x}_0) \, \mathrm{d}A(\boldsymbol{x}_0) \, \mathrm{d}A(\boldsymbol{x}_1) \\ & + \int_{\mathcal{M}^3} L_e(\boldsymbol{x}_2 \to \boldsymbol{x}_1) G(\boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_1) f_s(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0) G(\boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_1) W_e(\boldsymbol{x}_1 \to \boldsymbol{x}_0) \, \mathrm{d}A(\boldsymbol{x}_0) \, \mathrm{d}A(\boldsymbol{x}_1) \, \mathrm{d}A(\boldsymbol{x}_2) \\ & + \ldots \end{aligned} \tag{21}$$

Pfad-Wahrscheinlichkeit. Zur Anwendung von MC muss die Pfad-Wahrscheinlichkeit

$$p(\overline{\boldsymbol{x}}) = \frac{\mathrm{d}P}{\mathrm{d}\mu}(\overline{\boldsymbol{x}}) = \prod_{i=0}^{k} \frac{\mathrm{d}P}{\mathrm{d}A}(\boldsymbol{x}_i) \tag{22}$$

gefunden werden. Veach nutzt dazu lokale Pfad-Sampling-Algorithmen, um die Wahrscheinlichkeiten per Flächenelement für jeden Vertex $\boldsymbol{x}_i$ zu bestimmen. Dabei werden die Wahrscheinlichkeitsinformationen entweder *a-priori* als Verteilung über die Szenenoberfläche vorgegeben (Lichtquellen) oder aus dem am Vertex in eine zufällige Richtung abgestrahlten Sekundärstrahl bestimmt:

$$p(\boldsymbol{x}') = p(\omega_0) \frac{|\cos(\theta_i')|}{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}, \tag{23}$$

[6]Hier folgen wir der Konvention von [Segovia et al. 2007] und [Cline and Egbert 2005], so dass jeder Pfad $\overline{\boldsymbol{x}} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k) \in \Omega_k$ bei x_0 im Sensor beginnt und bei x_k an der Lichtquelle endet. Vorsicht: Im Gegensatz dazu arbeitet [Veach 1997] mit der Konvention, dass die Strahlen bei x_0 an der Lichtquelle beginnen und bei x_k im Sensor enden

wobei $\omega_0 = \widehat{\boldsymbol{x}' - \boldsymbol{x}}$ der Raumwinkel der Gerade $\boldsymbol{x} \to \boldsymbol{x}'$ ist. Oft arbeitet man auch mit dem *projizierten Raumwinkel* des Raumwinkels $D \subset S^2$:

$$\sigma^{\perp}_{\boldsymbol{x}}(D) = \int_D |\omega \cdot \boldsymbol{N}(\boldsymbol{x})| \, \mathrm{d}\sigma(\omega), \quad \text{infinitesimal}: \quad \mathrm{d}\sigma^{\perp}(\omega_0) = |\omega_0 \cdot \boldsymbol{N}(\boldsymbol{x})| \, \mathrm{d}\sigma(\omega_0), \tag{24}$$

wobei $\boldsymbol{N}(\boldsymbol{x})$ der Normalenvektor der Oberfläche bei $\boldsymbol{x}$ ist. Häufig arbeitet man mit dem Polarwinkel θ zwischen ω und $\boldsymbol{N}$, so dass $\omega \cdot \boldsymbol{N}(\boldsymbol{x}) = \cos\theta$. Dann kann die folgende Umrechnung zwischen den PDFs (vgl. [Veach 1997, ch. 8.2.2.2] verwendet werden:

$$p^{\perp}(\omega_0) = p(\omega_0) \frac{1}{\cos\theta_0}. \tag{25}$$

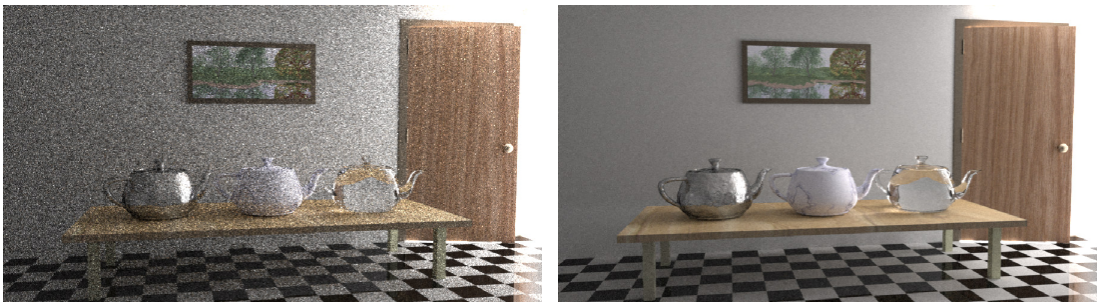
Alle Wahrscheinlichkeiten der generierten Vertizes zusammen ergeben aufmultipliziert die Wahrscheinlichkeit des Pfades.

Nach dieser Einführung in den Pfadintegral-Formalismus wollen wir uns speziell mit dem Metropolis-Lichttransportmodell (*Metropolis Light Transport*, MLT), welches Metropolis-Importance-Sampling (siehe Kap. 1.2.1) zur Effizienzsteigerung verwendet, befassen.

1.3.2 Metropolis-Algorithmus

Reines Pfadtracing sollte alle $L(D|S)^*E$-Pfade finden, wobei besonders die Spiegelungen S^* ein Problem sind, da hier eine stark richtungsabhängige Intensitätsverteilung vorliegt, und diese die Komplexität erhöhen. Der Metropolis-Algorithmus nutzt aus, dass Licht meist aus wenigen Richtungen, aber dafür sehr intensiv kommt. Dazu werden die einmal vom Pfadtracer erzeugte Pfade variiert, wobei biasing vermieden werden sollte.

Ein gutes Anwendungsbeispiel sind indirekte Beleuchtungsszenen. Es wurde schon besprochen, dass ein Raytracer einen mit gerichtetem Licht beleuchteten Raum nicht gut darstellen kann. Man stelle sich einen Raum vor, welcher durch ein im Rücken des Betrachters liegendes, großes Fenster von der Sonne beleuchtet wird. Für solche Szenen zahlt sich die Benutzung von Pfadtracern aus. Stellen wir uns nun aber vor, dass das Zimmer nur durch einen kleinen Türspalt beleuchtet wird, so werden die zufällig erzeugten Lichtwege eines Pfadtracer diesen Spalt nur selten treffen. Hier zahlt sich nun der Markovkettenansatz des MLT-Renderers aus, welcher einmal gefundene Pfade mit hoher Intensität (im Beispiel also einen Weg durch den Türspalt) nun leicht variiert und so in kürzerer Zeit mehr zum Bildinhalt beitragende Pfade findet als der gewöhnliche Pfadtracer.



(a) Birektionales Pfadtracing (b) Metropolis-Licht-Transport

Abbildung 9: *Ergebnisvergleich zwischen bidirektionalem Pfadtracing und MLT bei schlechten Beleuchtungsverhältnissen (durch einen Türspalt) und gleicher Rechenzeit – entnommen aus: Veachs Doktorarbeit (1997), S. 360.*

Vorgestellt wurde MLT von Veach & Guibas [Veach and Guibas 1997], [Veach 1997, ch. 11]. Als eine praktische Einführung ist [Cline and Egbert 2005] empfehlenswert. Mehr Hintergrund gibt es in [Pharr 2003].

Metropolis-Hastings-Mutationen. Bei der Integration von Gl. (eq:17) ist f nicht direkt berechenbar. Das ist aber auch nicht nötig. Ein MLT-Pfadtracer generiert Lichtpfade im hochdimensionalen Pfadraum, wertet diese aus, so dass der Beitrag zur Lichtintensität an der Kamera durch diesen Pfad bekannt ist, und mutiert den Pfad nach Metropolis-Hastings. Der Algorithmus ist in Alg. 2 gegeben.

Algorithmus 2 Metropolis-Light-Transport-Algorithmus

1) Benenne einen Startpfad $\overline{\boldsymbol{x}}_i \leftarrow \overline{\boldsymbol{x}}_0$.
2) Schlage ein neues $\overline{\boldsymbol{x}}^*$ gemäß $T(\overline{\boldsymbol{x}}_i \to \overline{\boldsymbol{x}}^*)$ ausgehend vom aktuellen $\overline{\boldsymbol{x}}_i$ vor.
3) Berechne $w = [p(\overline{\boldsymbol{x}}^*)T(\overline{\boldsymbol{x}}^* \to \overline{\boldsymbol{x}}_i)]/[p(\overline{\boldsymbol{x}}_i)T(\overline{\boldsymbol{x}}_i \to \overline{\boldsymbol{x}}^*)]$.
   i) Für $rand(0,1) \leq w$ akzeptiere den Vorschlag, d.h. $\overline{\boldsymbol{x}}_{i+1} \leftarrow \overline{\boldsymbol{x}}^*$.
   ii) Ansonsten behalte den alten Wert: $\overline{\boldsymbol{x}}_{i+1} \leftarrow \overline{\boldsymbol{x}}_i$.
4) Füge die Lichtintensität von $\overline{\boldsymbol{x}}_{i+1}$, bei Farben skaliert auf photometrische *Luminanz* ($|f| = 1$), dem Histogramm zu.
5) Setze $\overline{\boldsymbol{x}}_i \leftarrow \overline{\boldsymbol{x}}_{i+1}$ und beginne erneut bei 2) bis n MH-Mutationen durchgeführt sind.

Hierbei ist die Wahl von T von entscheidender Bedeutung, um gute Bilder zu erzeugen. Eine zu große Streuung führt zu vielen Rückweisungen. Man sollte auch beachten, dass bei einer zu kleine Streuung nur ein kleiner Teil des verfügbaren Raums abgedeckt wird. Dies hängt direkt

mit der Art, wie Lichtwege erzeugt werden, zusammen. Ausgehend vom Auge, werden die Lichtpfade in die Szene geworfen. An jeder Diffusivität oder Reflexion werden sie gemäß einer Wahrscheinlichkeitsdichtefunktion (PDF) weitergereicht. Dies bildet die *Kamerateilpfade* (lens subpaths). Trifft ein solcher Pfad eine Lichtquelle direkt, wird er impliziter Lichtweg (*implicit light path*) genannt. Ansonsten wird sein Endpunkt mit einem Punkt auf einer Lichtquelle verbunden: expliziter Lichtpfad (*explicit light path*).

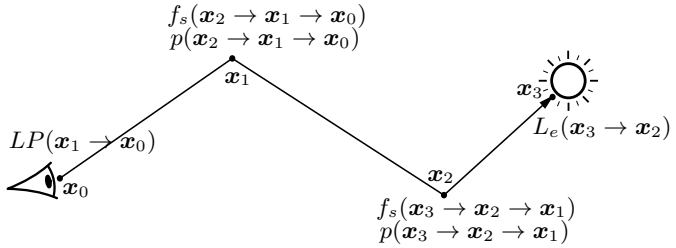
Intensität eines Lichtwegs. Damit MLT funktioniert, muss der Intensitätswert $LP(\overline{\boldsymbol{x}})$ eines Lichtpfads $\overline{\boldsymbol{x}} \in \Omega$ die Intensität am Sensor aus der Richtung der ersten Teilstücks samplen, d.h.

$$E[LP(\overline{\boldsymbol{x}})] = L(\boldsymbol{x}_1 \to \boldsymbol{x}_0). \tag{26}$$

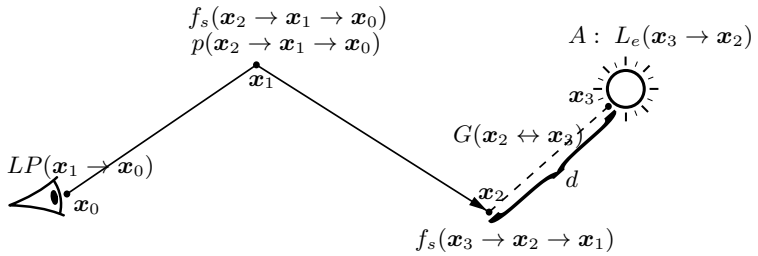
Dieser Wert wird für jeden Weg aus den BRDFs und den PDFs berechnet. Dies ist in Abb. 10 für Pfadtracer illustriert und in Gl. (eq:27) berechnet. Dabei ist P_{len} die Wahrscheinlichkeit, einen Lichtpfad der gegebenen Länge zu erzeugen, $P = AL_e$ die Ausgangsleistung der Lichtquelle, θ der Winkel zwischen $\boldsymbol{x}_3 \to \boldsymbol{x}_2$ und der Oberflächennormalen der Lichtquelle und P_{light} die Wahrscheinlichkeit, dass gerade diese Lichtquelle als Endpunkt gewählt wurde.

$$\begin{aligned}
\text{impliziter Lichtweg :} \quad & LP(\boldsymbol{x}_1 \to \boldsymbol{x}_0) = \frac{f_s(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0)}{p(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0)} \times \frac{f_s(\boldsymbol{x}_3 \to \boldsymbol{x}_2 \to \boldsymbol{x}_1)}{p(\boldsymbol{x}_3 \to \boldsymbol{x}_2 \to \boldsymbol{x}_1)} \times \frac{L_e(\boldsymbol{x}_3 \to \boldsymbol{x}_2)}{P_{\text{len}}} \\
\text{expliziter Lichtweg :} \quad & LP(\boldsymbol{x}_1 \to \boldsymbol{x}_0) = \frac{f_s(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0)}{p(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0)} \times \frac{G(\boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_3) f_s(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0) \cos\theta}{\pi d^2} \times \frac{AL_e(\boldsymbol{x}_3 \to \boldsymbol{x}_2)}{P_{\text{len}} P_{\text{light}}}
\end{aligned} \tag{27}$$

Für ideal diffuse oder ideal reflektierende Oberflächen kürzen sich BRDF und PDF, da die Wahrscheinlichkeit der Sekundärstrahlrichtung nach der BRDF gebildet werden sollte / muss. Für teilweise diffusive und reflektierende Oberflächen spielt noch die Wahrscheinlichkeit, ob der Strahl reflektiert wird oder nicht, eine Rolle.



(a) Impliziter Lichtpfad



(b) Expliziter Lichtpfad

Abbildung 10: *Illustration zur Lichtintensitätsberechnung entlang von Lichtwegen – entnommen aus: Cline & Egbert 2005, S. 8f.*

Mutationsstrategien. Für die Mutationsstrategien muss sichergestellt werden, dass die $T(\overline{\boldsymbol{x}}_i \to \overline{\boldsymbol{x}}_j)$ berechnet werden können, Ergodizität vorliegt und die Mutationen nicht zu häufig abgewiesen werden. Ergodizität bedeutet, dass es im Prinzip möglich sein muss, jeden anderen möglichen Lichtpfad durch Mutation mit endlicher Wahrscheinlichkeit zu verwirklichen.

Eine mögliche Mutation ist, einen Teilpfad herauszuschneiden und diesen von beiden Seiten neu zu samplen (*bidirectional mutations*). Ist keine Rekombination der neuen Teilpfade mögliche (Visibilität zwischen den Endpunkten), so wird die Mutation verworfen. Wenn es auch möglich ist, einen komplett neuen Pfad zu resamplen, dann ist $T(\overline{\boldsymbol{x}}_i \to \overline{\boldsymbol{x}}_j) > 0$ und die Ergodizität sichergestellt.

Andere Mutationen setzen an einzelnen Abschnitten oder Vertizes mit bestimmten Strategien an. Diese werden je nach Oberflächenmaterial ausgesucht. Einen Überblick über mögliche Strategien gibt es in [Cline and Egbert 2005]. Ausführlicheres in [Veach and Guibas 1997]. Eine Auswahl an Modifikationen sei hier kurz vorgestellt:

Lens perturbations: Hier wird der kameranahe Teil des Lichtpfads entfernt und der Bildpunkt leicht variiert. Nun wird der Lichtpfad wieder zusammengesampelt. Dies ist nur bei bestimmten Vertexkombinationen erlaubt.

Caustic perturbations: In anderen Fällen wird der entfernte Pfad dadurch neu bestimmt, dass der zuletzt entfernte Teilpfad in der Richtung gestört wird und dann der Pfad wieder zur Kamera verfolgt wird. Dies dient der Abdeckung von Kaustiken.

Multi-chain perturbation: Im Falle von komplizierten Mehrfachspiegelungen sind diese Modifikationen sinnvoll. Hier wird eine Linsenperturbation durch Spiegelungen weiterverfolgt und den kaustischen Störungen analoge Winkelvariationen an jeder diffusen Oberfläche durchgeführt.

Für eine gute Abdeckung des gesamten Bildbereichs (gegen stratification) sorgen schließlich sogenannte *Kamerateilpfadmutationen*. Kamerateilpfade besitzen die Form $(L|D)S^*E$. Diese werden entfernt und so ersetzt, dass durch jeden Bildpixel die gleiche Anzahl von Pfaden *vorgeschlagen* wird.

Normierung des Histogramms. Abschließend muss das erzeugte Pixelhistogramm normiert werden:

$$I_{\text{norm}} = sI, \quad s = \frac{f_{\text{avg}}}{h_{\text{avg}}}, \tag{28}$$

wobei f_{avg} die mittlere Intensität und h_{avg} die mittlere Anzahl von Samples pro Bin ist. Erstere Größe wird meist aus einigen vielen gesampelten Pfaden eines konvetionellen BPT mit MIS geschätzt. Danach wird MLT zur Rauschunterdrückung und Kaustikdarstellung verwendet.

Bidirektionales Pfadtracing. Bei der bidirektionalen Variante der Pfadverfolgung werden die Wege sowohl vom Auge aus (die oben besprochenen Kamerateilpfade, deren Vertizes mit $\boldsymbol{x}_0$ bis $\boldsymbol{x}_n$ bezeichnet werden) als auch von den Lichtquellen aus gesampelt (Lichtteilpfade, welche die Vertizes $\boldsymbol{y}_0$ bis $\boldsymbol{y}_m$ besitzen). Letztere, die Lichtunterwege, werden analog zu den Kamerateilpfaden generiert und berechnet. Die beiden Teilwege werden dann so verbunden, wie man eine Lichquelle an einen expliziten Linsensubweg gebunden hätte (siehe Abb. 11). Durch das Iterieren durch die einzelen Vertizes der Unterpfade entstehen verschiedene (s,t)-Pfade der Gesamtlänge $k = s + t - 1$ (mit Vertizes $\boldsymbol{z}_0$ bis $\boldsymbol{z}_k$), wobei $s \leq m+1, t \leq n+1$ und $k \leq m+n-1$:

$$\begin{aligned}
&\boldsymbol{z}_0 = \boldsymbol{x}_0, && \boldsymbol{z}_1 = \boldsymbol{x}_1, && \dots, && \boldsymbol{z}_{s-2} = \boldsymbol{x}_{s-2}, && \boldsymbol{z}_{s-1} = \boldsymbol{x}_{s-1},\\
&\boldsymbol{z}_s = \boldsymbol{y}_{t-1}, && \boldsymbol{z}_{s+1} = \boldsymbol{y}_{t-2}, && \dots, && \boldsymbol{z}_{s+t-2} = \boldsymbol{y}_1, && \boldsymbol{z}_{s+t-1} = \boldsymbol{y}_0.
\end{aligned} \tag{29}$$

Als Beispiele für diese Charakterisierungen sei gesagt, dass in Abb. 10(a) ein $(0,4)$-Pfad, in Abb. 10(b) ein $(1,3)$-Pfad und in Abb. 11 ein $(3,3)$-Pfad abgebildet ist.

$$\begin{aligned}
\text{bidirektionaler Lichtweg} : LP(\boldsymbol{x}_1 \to \boldsymbol{x}_0) = &\frac{f_s(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0)}{p(\boldsymbol{x}_2 \to \boldsymbol{x}_1 \to \boldsymbol{x}_0)} \times \frac{G(\boldsymbol{x}_2 \leftrightarrow \boldsymbol{y}_2) f_s(\boldsymbol{y}_2 \to \boldsymbol{x}_2 \to \boldsymbol{x}_1) f_s(\boldsymbol{x}_2 \to \boldsymbol{y}_2 \to \boldsymbol{y}_1)}{\pi d^2}\\
&\times \frac{f_s(\boldsymbol{y}_2 \to \boldsymbol{y}_1 \to \boldsymbol{y}_0)}{p(\boldsymbol{y}_2 \to \boldsymbol{y}_1 \to \boldsymbol{y}_0)} \times \frac{AL_e(\boldsymbol{y}_0 \to \boldsymbol{y}_1)}{P_{\text{len}} P_{\text{light}}}
\end{aligned} \tag{30}$$

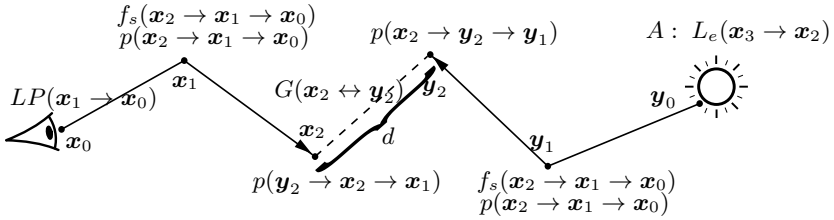


Abbildung 11: *Illustration zur Lichtintensitätsberechnung entlang von bidirektionalen Lichtwegen – entnommen aus: Cline & Egbert 2005, S. 15.*

Das Metropolis-Hastings-Verfahren, wie oben angegeben, ist einfach auf bidirektionale Mutationen übertragbar.

Ergebnis. Es zeigt sich, dass der MLT-Algorithmus aufgrund seiner Präferenz für intensive Strahlen deutlich bessere Ergebnisse bei schlechten Lichtverhältnissen erzielt. Dies ist sehr deutlich in einem Beispiel mit einem engen Türspalt zu sehen, durch den wenig Licht in einen Raum fällt (Abb. 9, entnommen aus [Veach 1997]). Hier werden genau diejenigen Kamerateilpfade mutiert, welche zu Pfaden gehören, die durch den Türspalt fallen.

Weiterführende Arbeiten. Metropolis-Light-Transport wurde in der CG-Community breit aufgegriffen und vielfältig erweitert. Weitere neuere Arbeiten zu MLT sind die Folgenden:

- Verbesserung der Mutations-Strategien [Kelemen et al. 2002] durch Verlagerung der Mutationen vom Pfadraum in einen unendlichdimensionalen Einheitswürfel, worin die Pfade repräsentiert werden. Dies verbessert für sogenannte kleine Mutationen die Varianz und erhöht die Akzeptanzraten. Zudem werden große Mutationen eingeführt, welche die Ergodizität sicherstellen und eine Art *random walk* im Pfadraum darstellen. Diese zweite Methode ähnelt BPT mit MIS. Somit werden also die Vorteile von MLT (in hellen Regionen) und BPT mit MIS (für dunkle Regionen) kombiniert.
- Zudem wurde Metropolis-Hastings in Energie-Verteilungs-Pfadtracern verwendet [Cline et al. 2005]. Dabei werden die Lichtpfade eines Pfadtracers (gewöhnliche Monte-Carlo-Integration) durch Metropolis-Hastings-Mutationen neu gesampelt und so deren Energie neu verteilt.
- Schließlich wurde *Replica Exchange* als Erweiterung von MH eingeführt [Kitaoka et al. 2009]. Diese alternative Monte-Carlo-Integrationsmethode erinnert an *simulated annealing* in Optimierungsprozessen.

2 Kohärenter Metropolis-Lichttransport

Durch die Einführung von MTMH in das Metropolis-Lichttranport-Modell haben Segovia, Iehl und Péroche eine Arbeit zur kohärenten Simulationsrechnung vorgestellt, welche MLT parallelisierbar macht [Segovia et al. 2007]. Sie nennen ihr modifiziertes Verfahren kohärenten Metropolis-Lichttransport (*coherent Metropolis Light Transport*, CMLT).

2.1 Was ist an MLT verbesserungswürdig?

Metropolis-Light-Transport ist bereits ein sehr elegantes und robustes Verfahren. Gegenüber anderen Simulationsmethoden zeichnet es sich durch numerische Stabilität und durch geringe Ansprüche an die Szenengeometrie aus. Es stellt eine allgemeine Lösung der Rendering-Gleichung dar und ist mit keinem Bias behaftet.

Dennoch ist die Methode verbesserungswürdig, da die bisherigen Implementierungen eine *sequentielle* Berechnung der Lichtpfadintensitäten vornehmen. Dies verhindert eine Parallelisierung des Prozesses. Weitere Nachteile von MLT sind größere Mutationsabweichungen, die auftreten, damit der Algorithmus biasfrei bleibt, welche aber Cache-Strategien (Clusterung) behindern, und die Flicker-Probleme, welche durch Rauschen in den Normierungsskalierungen auftreten.

Abhilfe für letzteres kann durch Glätten der Skalierung aufeinanderfolgender Frames geschehen. In den anderen Punkten kann die Verwendung einer Erweiterung des Metropolis-Hastings-Algorithmus für Abhilfe sorgen.

2.2 Multiple-Try-Muationen

Multiple-Try-Mutationen wurden als Modifizierung von MH für viele Dimensionen erdacht [Liu et al. 2000]. Hier im Beispiel mit multivariaten Normalverteilungen (d-dimensional) für die Schrittgröße wird diese mit $\sqrt{d}$ anwachsen (*curse of dimensionality*), was zu vielen Rückweisungen der Schritte führen wird. Ein Ausgleich durch Verringerung der Sprungweite führt zu einer zu geringen Abdeckung des Raums.

Daher haben Liu et al. den Multiple-Try-Metropolis-Hastings-Algorithmus (MTMH) vorgeschlagen. Für eine Vorschlagsfunktion $T(\boldsymbol{x} \to \boldsymbol{y})$, welche

$$T(\boldsymbol{x} \to \boldsymbol{y}) > 0 \Leftrightarrow T(\boldsymbol{y} \to \boldsymbol{x}) > 0 \tag{31}$$

erfüllen sollte, eine zu sampelnde Wahrscheinlichkeitsdichtefunktion $p(\boldsymbol{x})$ und eine zu wählende symmetrische, nicht-negative Funktion $\lambda(\boldsymbol{x}, \boldsymbol{y})$ definiert man:

$$w(\boldsymbol{x}, \boldsymbol{y}) := p(\boldsymbol{y}) T(\boldsymbol{y} \to \boldsymbol{x}) \lambda(\boldsymbol{x}, \boldsymbol{y}). \tag{32}$$

Algorithmus 3 Multiple-Try-Metropolis-Hastings-Algorithmus

1) Benenne einen Startwert $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_0$.
2) Schlage k neue $\{\boldsymbol{y}_l\}$ gemäß $T(\boldsymbol{x}_i \to \boldsymbol{y}_l)$ ausgehend vom aktuellen $\boldsymbol{x}_i$ vor.
3) Berechne $w(\boldsymbol{x}_i, \boldsymbol{y}_l)$ und wähle mit Wahrscheinlichkeiten gemäß dieser Gewichte ein $\boldsymbol{x}^* = \boldsymbol{y}_{l^*}$ aus.
4) Ziehe $k-1$ Widersacher $\{\boldsymbol{z}_l\}$ gemäß $T(\boldsymbol{x}^* \to \boldsymbol{z}_l)$ und setze $\boldsymbol{z}_k := \boldsymbol{x}_i$.
5) Berechne $\tilde{w} := [\sum_l w(\boldsymbol{x}_i, \boldsymbol{y}_l)] / [\sum_l w(\boldsymbol{x}^*, \boldsymbol{z}_l)]$.
   i) Für $rand(0,1) \leq \tilde{w}$ akzeptiere den Vorschlag, d.h. $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}^*$.
   ii) Ansonsten behalte den alten Wert: $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i$.
6) Füge das Sample $\boldsymbol{x}_{i+1}$ dem Histogramm zu.
7) Setze $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_{i+1}$ und beginne erneut bei 2).

Nun wird der Algorithmus in Alg. 3 durchgeführt. Die detaillierte Gleichgewichtsbedingung wird erfüllt, so dass eine Markov-Kette mit $p(\boldsymbol{x})$ als stationärer Verteilung erzeugt wird. Für $T(\boldsymbol{x} \to \boldsymbol{y})$ symmetrisch, ist

$$\lambda(\boldsymbol{x}, \boldsymbol{y}) = T(\boldsymbol{x} \to \boldsymbol{y})^{-1} \tag{33}$$

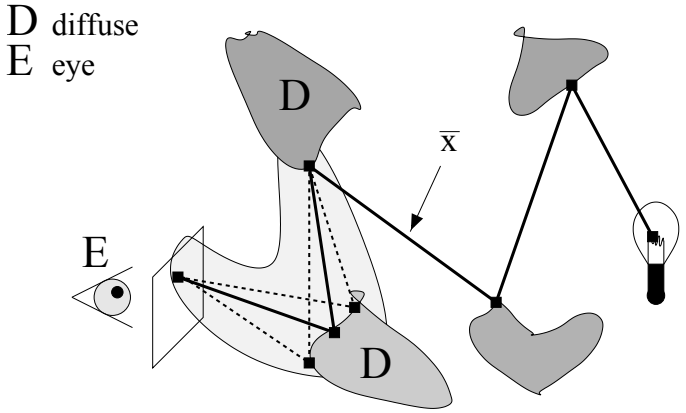
eine gültige Wahl, so dass einfach $w(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})$ [Wikipedia 2010b].

Dies verbessert die Akzeptanzraten, auch wenn die einzelnen Schritte groß sind. Damit vermindert MTMH das Dimensionalitätsproblem von MH in hochdimensionalen Räumen.

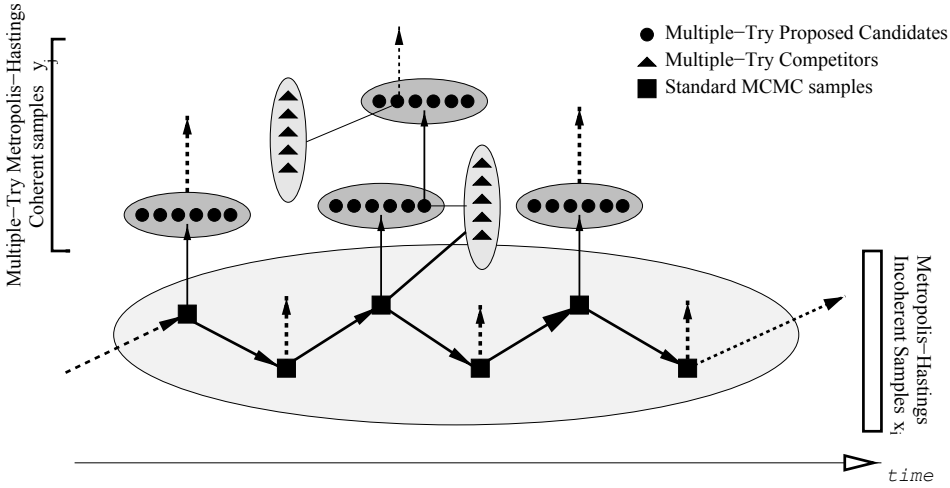
2.3 Der MTMH-Algorithmus

Die angesprochene sequentielle Abarbeitung in MLT soll durch die Einführung von MTMH (siehe Kap. 2.2) im Pfadtracer behoben werden: Es werden mehrere Samples auf einmal bearbeitet und kompetitiv verglichen. Diese Mutationsstrategie verbessert das Durchsuchen des Lichtpfadraums, indem ganze Familien von Samples verglichen werden.

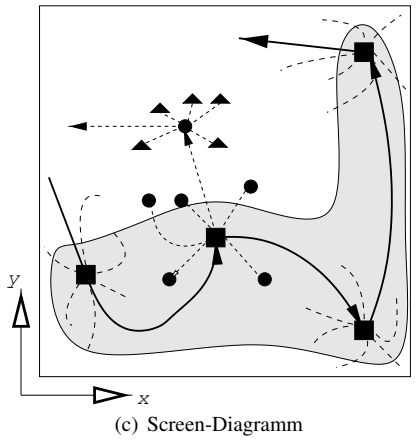
Die Autoren haben zur Darstellung die in Abb. 12 abgebildeten Illustrationen erstellt. Abb. 12(a) zeigt die Mutation eines Lichtweges in gleichzeitig prozessierte Alternativen. Wie die Vorschläge (Dreiecke) gegen die Widersacher (gefüllte Kreise) gegeneinander abgewogen werden, zeigt Abb. 12(b) im Zeitdiagramm und Abb. 12(c) in Projektion auf den Bildschirm.



(a) Multiple-Try-Mutationen



(b) Zeitdiagramm



(c) Screen-Diagramm

Abbildung 12: *Illustrationen zum MTMH-Verfahren – entnommen aus: Segovia et al. 2007.*

Leichte Veränderung durch Verschachtelung. Da eine Ersetzung von allen MH-Mutationen zu MTMH-Mutationen zu einer schlechteren Effizienz führt (es wird nur einer der k Kandidaten akzeptiert) und auch die Kohärenzeigenschaften in diesem Fall nicht optimal sind, wird stattdessen ein verschachtelter Algorithmus vorgeschlagen. Grob gesagt wird MLT in erster Ebene benutzt, um lichtintensive Pfade zu finden, und dann in einer zweiten Ebene MTMH, um komplizierte Strukturen, wie Kaustiken, aufzudecken.

Zuerst wird der Lichtwegraum durch Standard-MLT-Mutationen (bidirektional) erkundet. Dann wird für jeden MH-Pfad der zugehörige Linsenunterweg in kohärenter Weise durch MTMH-Störungen gesampelt. Hierbei werden aber im Gegensatz zum Multiple-Try-Verfahren im Original [Liu et al. 2000] alle Vorschläge nach den entsprechenden Gewichten und der Metropolis-Rate gewertet. Durch die Gewichtung bleibt die Eigenschaft erhalten, das detaillierte Gleichgewicht auch für diese Superposition zu erfüllen.

Dies nutzt aus, dass Kamerateilpfadräume nur durch Linsen- und kaustische Störungen erkundet werden können. Diese sind biasfrei. Ein Beispiel für Mutationen eines EDD-Unterpfades ist in Abb. 12(a) gegeben. Der gesamte Algorithmus ist in Alg. 4 gegeben. Hierbei sind T_{lc} die Übergangswahrscheinlichkeiten für die Linsen- und kaustischen Störungen der MTMH-Mutationen, so dass

$$w(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) := p(\overline{\boldsymbol{y}}) T_{\mathrm{lc}}(\overline{\boldsymbol{y}} \to \overline{\boldsymbol{x}}) \lambda(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \tag{34}$$

und $T_{\mathrm{bidir}}(\overline{\boldsymbol{x}} \to \overline{\boldsymbol{y}})$ die Übergangswahrscheinlichkeiten für die bidirektionalen Mutationen des Standard-MLT in erster Ebene.

Des Weiteren ist $p(\overline{\boldsymbol{x}})$ die Leuchtdichte, welche das Sample $\overline{\boldsymbol{x}}$ auf den Sensor wirft. Segovia et al. haben die Wahl

$$\lambda(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) := \frac{1}{T_{\mathrm{lc}}(\overline{\boldsymbol{y}} \to \overline{\boldsymbol{x}}) T_{\mathrm{lc}}(\overline{\boldsymbol{x}} \to \overline{\boldsymbol{y}})} \tag{35}$$

getroffen, aber im Prinzip keine große Abhängigkeit der Eigenschaften des Algorithmus von der Wahl von λ festgestellt.

Algorithmus 4 Kohärenter Metropolis-Light-Transport-Algorithmus

1) Benenne einen Startpfad $\overline{\boldsymbol{x}}_i \leftarrow \overline{\boldsymbol{x}}_0$ für die gewöhnlichen MH-Mutationen in erster Ebene.
   a) Benenne einen Startpfad $\overline{\boldsymbol{x}}_i^{(j)} \leftarrow \overline{\boldsymbol{x}}_i^{(0)} \leftarrow \overline{\boldsymbol{x}}_i$ für die MTMH-Mutationen der zweiten Ebene.
   b) Schlage k neue $\{\overline{\boldsymbol{y}}_l\}$ gemäß $T_{\mathrm{lc}}(\overline{\boldsymbol{x}}_i^{(j)} \to \overline{\boldsymbol{y}}_l)$ ausgehend vom aktuellen $\overline{\boldsymbol{x}}_i^{(j)}$ vor.
   c) Berechne $w(\overline{\boldsymbol{x}}_i^{(j)}, \overline{\boldsymbol{y}}_l)$ und wähle mit Wahrscheinlichkeiten gemäß dieser Gewichte ein $\overline{\boldsymbol{x}}^* = \overline{\boldsymbol{y}}_{l^*}$ aus.
   d) Ziehe $k-1$ Widersacher $\{\overline{\boldsymbol{z}}_l\}$ gemäß $T_{\mathrm{lc}}(\overline{\boldsymbol{x}}^* \to \overline{\boldsymbol{z}}_l)$ und setze $\overline{\boldsymbol{z}}_k := \overline{\boldsymbol{x}}_i^{(j)}$.
   e) Berechne $\tilde{w} := [\sum_l w(\overline{\boldsymbol{x}}_i^{(j)}, \overline{\boldsymbol{y}}_l)] / [\sum_l w(\overline{\boldsymbol{x}}^*, \overline{\boldsymbol{z}}_l)]$.
      i) Für $rand(0,1) \leq \tilde{w}$ akzeptiere den Vorschlag, d.h. $\overline{\boldsymbol{x}}_i^{(j+1)} \leftarrow \overline{\boldsymbol{x}}^*$.
      ii) Ansonsten behalte den alten Wert: $\overline{\boldsymbol{x}}_i^{(j+1)} \leftarrow \overline{\boldsymbol{x}}_i$.
   f) Füge die $2k$ Lichtintensitäten mit Gewichten

$$w \frac{f(\overline{\boldsymbol{y}}_l)}{F_y} \quad \text{und} \quad (1-w) \frac{f(\overline{\boldsymbol{z}}_l)}{F_z}, \quad \text{wobei} \quad F_y = \sum_{l=1}^{k} f(\overline{\boldsymbol{y}}_l), \quad F_z = \sum_{l=1}^{k} f(\overline{\boldsymbol{z}}_l), \tag{36}$$

      bei Farben skaliert auf photometrische *Luminanz* ($|f| = 1$), dem Histogramm zu.
   g) Setze $\overline{\boldsymbol{x}}_i^{(j)} \leftarrow \overline{\boldsymbol{x}}_i^{(j+1)}$ und beginne erneut bei b) bis m MTMH-Mutationen durchgeführt sind.
2) Schlage ein neues $\overline{\boldsymbol{x}}^*$ gemäß $T_{\mathrm{bidir}}(\overline{\boldsymbol{x}}_i \to \overline{\boldsymbol{x}}^*)$ ausgehend vom aktuellen $\overline{\boldsymbol{x}}_i$ vor.
3) Berechne $w = [p(\overline{\boldsymbol{x}}^*) T_{\mathrm{bidir}}(\overline{\boldsymbol{x}}_i \to \overline{\boldsymbol{x}}^*)] / [p(\overline{\boldsymbol{x}}_i) T_{\mathrm{bidir}}(\overline{\boldsymbol{x}}^* \to \overline{\boldsymbol{x}}_i)]$.
   i) Für $rand(0,1) \leq w$ akzeptiere den Vorschlag, d.h. $\overline{\boldsymbol{x}}_{i+1} \leftarrow \overline{\boldsymbol{x}}^*$.
   ii) Ansonsten behalte den alten Wert: $\overline{\boldsymbol{x}}_{i+1} \leftarrow \overline{\boldsymbol{x}}_i$.
4) Setze $\overline{\boldsymbol{x}}_i \leftarrow \overline{\boldsymbol{x}}_{i+1}$ und beginne erneut bei 2) bis n MH-Mutationen durchgeführt sind.

Die Standardabweichungen der Gaußschen Muationsübergangswahrscheinlichkeiten wurden gemäß den Erfahrungen von [Veach and Guibas 1997] gewählt, da sich diese auch hier bewähren. Ein kritischerer Parameter ist die Anzahl k von MTMH-Vorschlägen.

2.4 Implementation

Es folgt die Zusammenfassung der Implemtierungshinweise, welche in der Arbeit von Segovia, Iehl und Péroche angegeben sind.

2.4.1 Mutationsstrategien

Es werden unterschiedliche Mutationsstrategien für kaustische und nicht-kaustische Lichtwege besprochen. Je nach den Anforderungen werden Gauß-verteilte oder gleichverteilte Zufallsgrößen als Sprungparameter benutzt, wobei die Gauß-verteilten[7] bessere Resultate liefern und daher zu bevorzugen sind.

[7] Um genau zu sein: Es werden auf den Bereich Ω zugeschnittene Gaußverteilungen

$$q_{\mathrm{clamped},\mu,\sigma} = \frac{q_{\mu,\sigma}}{\int_\Omega q_{\mu,\sigma}(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}}, \quad q_{\mu,\sigma} = \frac{1}{2\sigma\pi} \exp\left(-\frac{(x-\mu)^2}{\sigma}\right) \tag{37}$$

benutzt, welche vorberechnet werden.

2.4.2 Kohärente Berechnungen

Für die parallele Berechnung der MTMH-Mutationen sind verschiedene Ansätze möglich: *SIMD*-Strahlbündel oder Strahlbündel-Frustums.

Kohärentes MLT mit SIMD-Strahlbündeln. Es werden zunächst uniform verteilte Mutationsschritte gebildet und geclustert. In Abb. 13 ist zu sehen, wie im Parameterraum Planquadrate gebildet werden, in welchen dann jeweils eine gleichförmige Zufallszahl gezogen wird. Benachbarte Zellen, wie links oben abgebildet, bilden Cluster. Dann werden diese in normalverteilte Schrittparameter umgewandelt (*Box-Muller*-Transformation). Die Mutationsschritte werden dann in Paketen, den oben gebildeten Clustern, gerechnet. Dies geschieht analog zu den kohärenten Raytracern in [Wald et al. 2001]. Die Kohärenz ist aufgrund der Nähe der mutierten Pfade (vgl. die *multi-chains*-Störungen in [Veach and Guibas 1997]) gegeben.

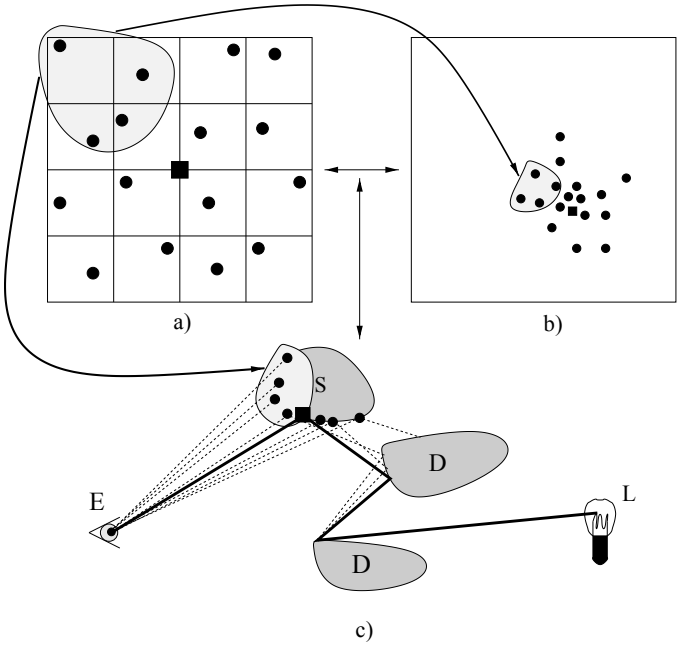


Abbildung 13: *Illustrationen zur kohärenten SIMD-Strahlbündelung – entnommen aus: Segovia et al. 2007.*

Kohärentes MLT mit Strahlbündeln-Frustums. Eine andere Möglichkeit sind Strahlbündel-Frustums wie in [Reshetov et al. 2005] vorgestellt. Dies ist besonders für EDD-Pfade geeignet. Nach Segovia et al. schlägt für komplexere Wege die zugrunde liegende Intervall-Arithmetik fehl. Haben aber Mutationen gemeinsame Ursprünge, so ist diese Methode jedoch sehr effizient.

2.5 Resultate

Die CMLT-Implementierung von Segovia et al. basiert auf der OpenRT-API [Dietrich et al. 2003]. Wie erklärt, besteht die Implementierung aus den beiden Teilen Standard-MLT und MTMH mit kohärenter SIMD-Strahlbündel-Berechnung (es findet noch keine Anwendung der Frustum-Technik statt).

2.5.1 Vergleich mit MLT und Parameterstudien

Die Erweiterungen, welche kohärentes MLT vom Standard-MLT abhebt, werden durch drei Parameter beschrieben:

1. Die Standardarbweichung σ der Gaußschen Normalverteilungen,
2. Die Anzahl m der MTMH-Schritte und
3. die Anzahl k der MTMH-Kandidaten und -Widersacher.

CMLT enthält MLT vollständig und geht in dieses für $m = 1$ und $k = 1$ über.

Segovia und seine Mitarbeiter haben die Abhängigkeit des Algorithmus von diesen zusätzlichen Parametern bestimmt und Folgendes gefunden:

- Insensitivität gegenüber der Länge der MTMH-Untersequenzen (Parameter m). Dies ist dadurch zu erklären, dass die MH-Samples bereits der Lichtintensität proportional sind. MTMH ändert nichts daran.

  Für die Verwendung von MTMH in einem gewöhnlichen Pfadtracer (wie z.B. den ER-Pfadtracern [Cline et al. 2005]) sollten längere MTMH-Sequenzen nötig sein, damit die Markov-Kette stabilisiert wird.

- Die Schrittgröße, welche durch σ parametrisiert wird, und die Anzahl an kompetitiven MTMH-Kandidaten hängen zusammen. Dies ist daran ersichtlich, dass eine große Schrittweite durch eine Mehrzahl an Kandidaten kompensiert werden kann. Somit bleibt die Akzeptanzrate stabil, wenn die Anzahl an Wettstreitern erhöht wird.

  Umgekehrt sorgen jedoch auch zu viele Kandidaten (bei kleinem σ) zu schlechten Resultaten: dies wird *stratification* genannt und entspricht dem *balls in bins*-Problem[8]. Die Auswirkungen sind in Abb. 14 zu sehen.

2.5.2 Leistungssteigerung

Segovia et al. berichten von einer Leistungssteigerung ihres Multithread-Raytracers mit SSE-SIMD-Ausnutzung um einen Faktor von 1.5 bis 2.3 (mit optimierten Parametern) gegenüber dem Standardsetup. Eine weitere Steigerung könnte das Benutzen von Strahlbündel-Frustums bringen.

2.5.3 Caching

Ein weiterer Ausbau der Engine von Segovia et al. beinhaltet einen Cache fester Größe für die Dreiecke der Geometrie. Im Vergleich mit gewöhnlichem MLT und, als Alternative, einem Radiosity-Renderer [Keller 1997] zeigt sich, dass CMLT wesentlich *Cache-freundlicher* ist als MLT und es diesbezüglich mit dem Radiosity-Verfahren aufnehmen kann. Dies liegt an der nahen arithmetischen Verwandtschaft dieser Verfahren: Tiling im IR vs. Jittering im CMLT.

(a) wenige Kandidaten (b) viele Kandidaten

Abbildung 14: *Stratification: Viele MTMH-Kandidaten und geringe Varianz – entnommen aus: Segovia et al. 2007.*

3 Ausblick

3.1 Beschränkungen

Neben den genannten Verbesserungen hat CMLT auch einige der Einschränkungen von MLT geerbt, die bestehen bleiben. Dies sind zum Einen die Flickerprobleme in Animationen, welche aber wie bereits erwähnt durch geglättete Skalierungen vermieden werden können.

CMLT kann hier aber auch einen anderen Weg gehen und für Zusammenhänge zwischen den Lichtpfaden von aufeinanderfolgenden Frames sorgen, indem die MH-Pfade zwischengespeichert werden und diese dann sequentiell angepasst werden. Die MTMH-Sequenzen können dann aus Quasi-Zufallszahlen gezogen werden, so dass sie reproduzierbar sind. Dies sollte die Korrelation zwischen den Frames drastisch erhöhen.

Es sei noch genannt, dass gerade die Abhängigkeit des (MT)MH-Algorithmus von guten (Pseudo)-Zufallszahlgeneratoren bestehen bleibt.

3.2 Anwendbarkeit in anderen Renderern

MTMH sollte auch statt MH im Energieverteilungs-Raytracer [Cline et al. 2005] zu besseren Ergebnissen führen. Aber auch die Verwendung in anderen Renderern ist möglich, da keine speziellen Anforderungen an die Geometrie oder die Materialien gestellt werden.

3.3 Verteilte Rechennetzwerke

Will man Computercluster benutzen, so steht die hohe Datenmenge, die zwischen separaten Prozessen hin- und hergeschoben werden muss, im Wege. Wie alle MC-Verfahren muss CMLT bei Mutationen und dem damit verbundenen Resampling auf alle Oberflächen der Geometrie zugreifen können und ferner müssen alle Prozese Schreibzugriff auf den ganzen Bildbereich besitzen. Eine Überlegung wäre der Rückgriff auf *Interleaved Sampling* [Keller and Heidrich 2001], welches ohne große Eingriffe in den Algorithmus auch hier zu verwirklichen wäre und die Berechnungen für bestimmte Pixeluntermengen verteilen würde.

3.4 Pfadtracing auf der GPU

Relativ neu ist die Implementierung des ersten Kelemen-MLT auf der GPU im Rahmen von Arauna[9].

3.5 Abschließende Betrachtungen

Segovia, Iehl und Péroche haben eine kohärente Erweiterung des Metropolis-Lichttransport-Modells präsentiert, welches Strahlbündel effizient parallelisiert berechnet. Dies kann einen bedeutenden Schritt in Richtung Echtzeit-Rendering für MLT-Systeme bedeuten.

Da auf MLT aufgebaut wird, ist dieser Algorithmus vergleichsweise robust, allgemein (da er die Rendering-Gleichung löst und somit globale Beleuchtung implementiert) und für jede Szene und alle Lichtverhältnisse tauglich ist.

[8] Werden n Bälle zufällig in n Körbe verteilt, so kann man nicht erwarten, dass in jedem Korb ein Ball liegt. Stattdessen sind viele Körbe leer und der vollste hat $\mathcal{O}(n \log n)$-Bälle [Veach 1997, S. 345].

[9] siehe `http://igad.nhtv.nl/~bikker/`

A Definitionen

A.1 Lichtpfadnotation

Im Folgenden werde ich, der mir vorliegenden Literatur folgend, die *Lichtpfadnotation* von Heckbert [Heckbert 1990] zur Charakterisierung von Lichtwegen benutzen.

Dies sind reguläre Ausdrücke in den folgenden Termen:

L Lichtquelle,
D diffuse Oberfläche,
S spiegelnde Oberfläche,
E Beobachter.

So beschreibt zum Beispiel $ES^*DS^+(D|L)$ einen kaustischen Lichtpfad von der Linse zu einer diffusen Oberfläche oder einer Lichtquelle. Diese wurde von Veach erweitert [Veach 1997, ch. 8.3.2], um auch (ausgedehnte) Lichtquellen und Kameras (mit finiten Aperturen) als Streuungen zu beschreiben.

A.2 Wahrscheinlichkeitsdichten und Erwartungswerte

Die *Wahrscheinlichkeitsdichte* (auch Wahrscheinlichkeitsdichtefunktion, *engl. probability density function*) $p(x)$ misst die infinitesimale Wahrscheinlichkeit einer eindimensionalen Zufallsvariablen X:

$$P(a < X < b) = \int_a^b p(x)\,\mathrm{d}x. \tag{38}$$

Man beachte, dass immer $p(x) \geq 0$. Meist wird eine PDF *normiert*, d.h. $\int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$. Die Stammfunktion der PDF wird *(kumulative) Verteilungsfunktion* genannt:

$$F(x) = \int_{-\infty}^{x} p(x')\,\mathrm{d}x'. \tag{39}$$

Die Erweiterung zu mehrdimensionalen PDFs $p(\boldsymbol{x})$ auf einem Gebiet M ist 'straightforward". Es sind beliebige Maße $\mathrm{d}\mu(\boldsymbol{x})$ möglich.

Erwartungswert und Varianz. Der *Erwartungswert* einer Zufallsvariablen $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X})$ bei Wahrscheinlichkeitsdichte $p(\boldsymbol{x})$ und Maß $\mathrm{d}\mu(\boldsymbol{x})$ im Gebiet M ist definiert als:

$$E\left[\boldsymbol{Y}\right] := \int_M \boldsymbol{f}(\boldsymbol{x}) p(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x}). \tag{40}$$

Die *Varianz* ergibt sich zu $V\left[\boldsymbol{Y}\right] := E\left[(\boldsymbol{Y} - E\left[\boldsymbol{Y}\right])^2\right]$. Rechenregeln für den Umgang mit Erwartungswerten und Varianzen, wie bspw. $E\left[\sum_i \boldsymbol{Y}_i\right] = \sum_i E\left[\boldsymbol{Y}_i\right]$ lassen sich jedem Statistikstandardwerk entnehmen.

A.3 Konsistenz und Erwartungstreue

Ein Schätzer F_N einer Größe I ist *konsistent*, wenn

$$\lim_{N\to\infty} \beta[F_N] := \lim_{N\to\infty} E[F_N - I] = 0 \quad \text{und} \quad \lim_{N\to\infty} V[F_N] = 0 \tag{41}$$

gilt, wobei F_N der geschätzte Wert aus N Samples ist [Veach 1997].

Eine äquivalente Definition ist

$$\lim_{N\to\infty} P[|F_N - I| > \epsilon] = 0. \tag{42}$$

Hierbei gibt $P[|F_N - I| > \epsilon]$ die Wahrscheinlichkeit an, dass der Fehler größer als ein festes ϵ ist.

Der Schätzer ist genau dann *erwartungstreu* (*engl. unbiased*), wenn

$$E[F_N - I] = 0. \tag{43}$$

Hier erwartet man also im Mittel einen richtigen Wert. Für solche Schätzer wird die Varianz des Schätzers als Fehlerangabe zu Rate gezogen.

A.4 Danksagung

Ich möchte mich herzlichst bei Prof. Dr. W. Kurth für die Betreuung des Computergrafik-Seminars bedanken. Ganz besonderer Dank geht vor allem an R. Hemmerling für seine freundliche Unterstützung bei der Recherche und Ausarbeitung dieses Vortrags und die vielen Hilfestellungen.

Abbildungsverzeichnis

Literatur

CLINE, D., AND EGBERT, P. 2005. A practical introduction to metropolis light transport. Tech. rep., Brigham Young University, May.

CLINE, D., TALBOT, J., AND EGBERT, P. 2005. Energy redistribution path tracing. *ACM Transactions on Graphics 24*, 3 (Aug.), 1186–1195.

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, vol. 18, 137–45.

DIETRICH, A., WALD, I., BENTHIN, C., AND SLUSALLEK, P. 2003. The OpenRT Application Programming Interface – Towards A Common API for Interactive Ray Tracing. In *Proceedings of the 2003 OpenSG Symposium*, Eurographics Association, Darmstadt, Germany, 23–31.

HECKBERT, P. S. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, 145–154.

KAJIYA, J. T. 1986. The rendering equation. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, 143–150.

KELEMEN, C., SZIRMAY-KALOS, L., ANTAL, G., AND CSONKA, F. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. *Computer Graphics Forum 21*, 3, 531–540.

KELLER, A., AND HEIDRICH, W. 2001. Interleaved sampling. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, 269–276.

KELLER, A. 1997. Instant radiosity. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 49–56.

KITAOKA, S., KITAMURA, Y., AND KISHINO, F. 2009. Replica exchange light transport. *Computer Graphics Forum 28*, 8 (Dec.), 2330–2342.

LIU, J. S., LIANG, F., AND WONG, W. H. 2000. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association 95*, 449 (March), 121+.

PHARR, M. 2003. Chapter 9: Metropolis sampling. In *SIGGRAPH 2003 Course Note #44: Monte Carlo Ray Tracing*.

RESHETOV, A., SOUPIKOV, A., AND HURLEY, J. 2005. Multi-level ray tracing algorithm. *ACM Transactions on Graphics 24*, 3 (Aug.), 1176–1185.

SEGOVIA, B., IEHL, J.-C., AND PÉROCHE, B. 2007. Coherent Metropolis Light Transport with Multiple-Try Mutations. Tech. Rep. RR-LIRIS-2007-015, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon, Apr.

VEACH, E., AND GUIBAS, L. 1994. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*, 147–162.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 65–76.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford.

WALD, I., SLUSALLEK, P., BENTHIN, C., AND WAGNER, M. 2001. Interactive rendering with coherent ray tracing. *Computer Graphics Forum 20*, 3, 153–164.

WHITTED, T. 1980. An improved illumination model for shaded display. *Communications of the ACM 23*, 6 (June), 343–349.

WIKIPEDIA, 2009. Globale Beleuchtung — Wikipedia, Die freie Enzyklopädie. [Online; Stand 23.11. 2010].

WIKIPEDIA, 2010a. Bidirectional reflectance distribution function — Wikipedia, The Free Encyclopedia. [Online; accessed 24.11.2010].

WIKIPEDIA, 2010b. Multiple-try Metropolis — Wikipedia, The Free Encyclopedia. [Online; accessed 8.1.2011].