Aktuelle Techniken für das Entrauschen von Realtime-Raytracing

Matz Radloff
Student B.Sc. Informatik, Universität Hamburg, Deutschland
18.01.2021

Abstract

Raytracing und das darauf aufbauende Path Tracing sind die aktuell besten Methoden, um fotorealistische Bilder zu generieren. Für die Berechnung in Echtzeit sind mit heutiger Hardware Kompromisse erforderlich, die versuchen, möglichst wenig Abstriche in der visuellen Qualität bei gleicher Performance zu erreichen. Das Entrauschen von Bildern, die mit geringer Anzahl von Abtastpunkten mittels Path Tracing generiert werden, stellt eine vielversprechende Gruppe von Methoden dar, die dieses Papier erklärt und vergleicht.

1 Einleitung

Als Raytracing bzw. Path Tracing werden Techniken bezeichnet, welche die Rendergleichung lösen, die stark an das tatsächliche Verhalten von Licht in der Natur angelehnt ist. [1] Dadurch wird ein hohes Maß an visueller Genauigkeit erreicht und die Generierung von fotorealistischen Bildern ermöglicht. Einer der größten Nachteile dieser Methoden ist die hohe erforderliche Rechenkapazität, sodass die Berechnung in Echtzeit, also mehrerer Bilder pro Sekunde, mit aktuellen Computern nicht möglich ist. Dieses Papier betrachtet den Ansatz, Raytracing-Algorithmen mit sehr geringer Anzahl von Abtastpunkten zu berechnen und das entstehende Bild mithilfe verschiedener Entrauschtechniken visuell korrekt anzunähern.

2 Raytracing Hintergrund

Der Begriff “Raytracing” umfasst mehrere Techniken, die alle auf dem Prinzip beruhen, virtuelle Strahlen und deren Schnittpunkte mit anderen Objekten zu berechnen. Dies ähnelt dem Verhalten von natürlichem Licht, welches sich genauso linienförmig fortbewegt und dessen Interaktionen mit Materie bestimmen, wie viele Photonen und in welcher Art das menschliche Auge oder eine Kamera erreichen. Anders als in der Natur werden bei den meisten Methoden die Strahlen nicht von der Lichtquelle, sondern von der Kamera bzw. der Projektionsfläche emittiert, wodurch nur die Strahlen berechnet werden müssen, die zum generierten Bild beitragen.
Auf dieser Technik aufbauend gibt es viele verschiedene Erweiterungen, die z.B. die Berechnung von Schatten, Lichtbrechung, Lichtbeugung und Transmission ermöglichen.
Path Tracing bezeichnet die Methode, die alle diese Erweiterungen umfasst und jeden Strahl rekursiv entlang seines Pfades in der Szene berechnet.

Da es in den meisten Bildern pro Pixel viele verschiedene Objekte und Lichtquellen, und somit auch viele Strahlen gibt, die zu dessen endgültiger Farbe beitragen, muss der Path Tracing-Algorithmus an mehreren Abtastpunkten berechnet werden. Anhand dessen genauem Ort lässt sich schwer feststellen, welchen Pfad die Strahlen nehmen werden, wodurch eine zufällige Wahl in der Praxis gut funktioniert. Dies wird auch als stochastisches oder Monte-Carlo-Raytracing bezeichnet. [2]
Je weniger Abtastpunkte pro Pixel verwendet werden, desto weniger Informationen der beteiligten Lichtpfade können integriert werden, was sich als Rauschen im finalen Bild ausdrückt. Dieses Papier erläutert verschiedene Techniken, um das Rauschen zu verringern, ohne mehr Abtastpunkte verwenden zu müssen.

3 Methodenvergleich

Existierende Methoden lassen sich grob in folgende Bereiche einteilen:

Filter
Diese Techniken können effektiv sein, da sie sehr schnell zu berechnen sind. Allerdings leidet die Bildqualität proportional zu dem Faktor der Rauschentfernung. Vor allem kontraststarke Bereiche, schmale Kanten und hochfrequente Bildinformationen gehen verloren. Außerdem können Filter die Helligkeit und Farben des Eingabebilds stark verfälschen. Beispiele für Filtermethoden sind [3][4][5].
Vereinfacht beschrieben sind dies bekannte Weichzeichnungsmethoden, die mithilfe zusätzlicher Informationen aus der Rendering-Pipeline bessere Ergebnisse erzielen.

Sampling
In einer naiven Monte-Carlo-Implementierung von Path Tracing ist jeder Abtastpunkt gleichwertig. Einige dieser Punkte lassen sich allerdings sehr viel schneller berechnen als andere oder ändern sich nicht von einem Bild im Vergleich zu nachfolgenden. Dieses Verhalten kann ausgenutzt werden, um die vorhandene Rechenkapazität möglichst effizient zu verteilen und redundante Berechnungen zu vermeiden. [6][7]

Machine Learning
Unabhängig von Raytracing lässt sich Machine Learning mit viel Erfolg für das Entrauschen von Daten nutzen. Hierfür bieten sich Denoising Autoencoder an.
Außerdem gibt es Ansätze, die zusätzlich Raytracing- und Rendering-spezifische Informationen als Eingaben erhalten. Dazu gehören z.B. das zeitlich vorherige Bild, Informationen über die Wichtigkeit der Abtastpunkte, Normalen- und Bewegungsvektoren. [8][9]

Upscaling
Upscaling bezeichnet die von der Render-Methode unabhängige Technik ein gering auflösendes Bild in ein höher auflösendes zu überführen. Im Falle von Path Tracing kann die Zahl der Abtastpunkte höher gewählt werden, je niedriger die Auflösung ist, was zusammen mit Upscaling zu besserer Performance bei gleichzeitig besserer visueller Qualität führen kann. [10] [11]

Kombinationen
Die besten Ergebnisse lassen sich durch die Verbindung mehrerer der genannten Techniken

erreichen. Z.B. gibt es Upscaling-Methoden, die CNNs oder GANs verwenden, die eine gute Qualität versprechen. [11]
Eine andere Möglichkeit ist, die Parameter von Sampling- und Filter-Methoden dynamisch von einem Neuronalen Netzwerk bestimmen zu lassen, das wiederum Eingaben aus der Rendering-Pipeline erhält. In der Praxis ermöglichen vergleichbare Techniken bereits die Generierung von simplen 3D-Umgebungen durch Path Tracing in Echtzeit. [12] [13]

4 Ausgewählte Beispiele

SVGF (Sampling)

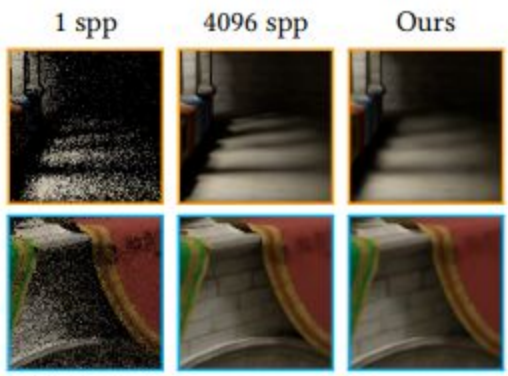

Abb. 1. Vergleich von 1-spp- und 4096-spp-Path-Tracing mit SVGF

Spatio-Temporal Variance Guided Filtering (SVGF) ist eine Technik, die aus einem Eingabebild, das einen Abtastpunkt pro Pixel verwendet, eine Ausgabe mit sehr wenig Rauschen generiert und zeitlich sehr stabil ist. [14]
Hierbei werden Informationen des aktuellen Bildes und vorheriger Bilder aus der Rendering-Pipeline verwendet (Bewegungsverktoren, Normalenvektoren, Farbe, Tiefe und Mesh-ID), um die Parameter mehrerer Wavelet-Filter zu bestimmen.
Integraler Bestandteil dieser Technik ist das temporale Filtern, das ähnlich funktioniert, wie temporal antialiasing (TAA). Dazu wird für jeden Pixel ein 2D-Bewegungsvektor benötigt, anhand dessen die Bewegung rückwärts in die Daten der vorherigen Bilder projiziert wird. Zusätzlich vergleicht SVGF dann die gefundenen Informationen über Tiefe, Normalenvektoren und Mesh-ID mit denen des aktuellen Frames. Stimmen diese ausreichend überein, werden alte und neue Farbinformationen mittels eines gleitenden Mittelwerts integriert. Wird kein übereinstimmender Wert gefunden, kann von einer Verdeckung ausgegangen werden und nur die aktuelle Information wird verwendet.
Desweiteren wird die Varianz der Leuchtkraft pro Pixel anhand der Farbe über mehrere Bilder bestimmt, um zu beurteilen, wie viel Rauschen dieser Punkt enthält. Dies ist zwar keine perfekte Methode, da Rauschen zwar die Varianz erhöht, letztere aber auch ohne Rauschen einen hohen Wert haben kann. Allerdings bietet es eine gute Basis, um die Parameter der folgenden Filter mitzubestimmen.
Für SVGF werden 5 Stufen eines a-trous wavelet Filters benutzt. Vereinfacht beschrieben kann dieser in Abhängigkeit von Farbvarianz, den temporal integrierten Farbinformationen und gefundenen Kanten aus der 3d-Pipeline unterschiedlich stark glätten. Das Ergebnis der ersten Filterstufe wird gespeichert und als Eingabe für die Berechnung der Varianz des nächsten Bildes verwendet.
Alle diese Techniken vereint produzieren ein sehr rauscharmes Bild, das auch, für Path Tracing mit wenigen Abtastpunkten typischerweise schwere Aspekte wie Reflektionen, Emissionen und schwach beleuchtete Bereiche, gut beherrscht. Die Autoren geben für die Berechnungszeit eines

Bildes in 720p Werte zwischen 4.1ms und 5.8ms an. Somit ist diese Methode für das Entrauschen von Path Tracing in Echtzeit gut geeignet.
Ein gutes Beispiel für die Verwendung von SVGF ist Minecraft RTX. [15]

Denoising Autoencoder (Machine Learning)

Diese Methode verwendet auch ein Eingabebild, das mit nur einem Abtastpunkt pro Pixel berechnet wird, das aber mithilfe eines Recurrent Convolutional Neural Networks (RCNN) entrauscht wird. [16]
Recurrent bedeutet, dass das Netzwerk Rückkopplungen zwischen verschiedenen Berechnungsschritten besitzt, um zeitliche Abhängigkeiten abzubilden. Convolutional bedeutet, dass die einzelnen Ebenen des Netzerks gewichtete Faltungsmatrizen (convolution kernel) verwenden, um Informationen zu integrieren und anschließend wieder zu extrahieren. Die letzten beiden Schritten werden auch als Encoder und Decoder bezeichnet und bilden zusammen die Basis eines Autoencoders.
Bei dieser Technik besteht der Vorteil, dass nicht nur die Farbinformationen als Eingabe des Netzwerks verwendet werden können, sondern zusätzliche Hilfsattribute aus der Rendering-Pipeline: Normalenvektoren (view-space, 2D), Tiefenwerte und Materialrauhigkeit. Es ergeben sich also 4 zusätzliche Zahlenwerte zu den 3 Farbwerten pro Pixel.
Mit geeigneter Wahl der Netzwerkparameter erreichen die Autoren ein sehr rauscharmes Bild. Allerdings benötigte diese Methode 54.9ms, um ein 720p zu berechnen. Dies liegt zwar zu hoch, um es in Echtzeitanwendungen zu benutzen, allerdings stellt es eine vielversprechende Basis für zukünftige Techniken dar.

5 Limitationen und Zusammenfassung

Path Tracing stellt die qualitativ bestmögliche Methode dar, fotorealistische Bilder zu generieren. Die vielen benötigten Rechenschritte und daraus resultierende lange Berechnungszeit machen es mit heutiger Hardware nur bei geringer Anzahl von Abtastpunkten möglich, Bilder in Echtzeit zu berechnen. Das resultierende, stark rauschbehaftete Bild, nachträglich zu entrauschen, stellt eine realistische Möglichkeit dar, qualitativ höherwertige Bilder zu erzeugen, als klassische Verfahren, die kein Raytracing verwenden.
Der größte limitierende Faktor beim Entrauschen ist, wie beim Path Tracing selbst, die benötigte Rechenzeit. Allerdings zeigen die verschiedenen Methoden, dass bei gut gewählten Kompromissen zwischen Bildqualität und Performance Ergebnisse erzielt werden können, die eine Verwendung von Path Tracing in Echtzeit erlauben. Die Verbreitung in echten Produkten zeigt allerdings, dass dies am besten funktioniert, wenn die abzubildende Welt weniger komplex ist, als in aktuellen Spielen und 3D-Anwendungen üblich. Sowohl Quake II RTX, Minecraft RTX und Teardown haben gemeinsam, dass ihre virtuellen Welten stark simplifiziert sind. Andere Anwendungen, die Formen von Raytracing verwenden beschränken sich aktuell meistens nur auf einzelne Aspekte, die eingebunden werden (Raytraced global illumination, Raytraced reflections, Raytraced shadows). [17][18]

6 Literatur

[1] James T. Kajiya: *The rendering equation*
https://doi.org/10.1145/15886.15902
August 1986
Letzter Zugriff: 15.01.2021

[2] Peter Shirley and Changyaw Wang and Kurt Zimmermann: *Monte Carlo Techniques for Direct Lighting Calculations*
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.6561
1996
Letzter Zugriff: 15.01.2021

[3] C. Tomasi und R. Manduchi: *Bilateral Filtering for Gray and Color Images*
https://dl.acm.org/doi/10.5555/938978.939190
Januar 1998
Letzter Zugriff: 16.01.2021

[4] Kaiming He, Jian Sun und Xiaoou Tang: *Guided Image Filtering*
http://kaiminghe.com/publications/pami12guidedfilter.pdf
October 2012
Letzter Zugriff: 16.01.2021

[5] Mara, Michael and McGuire, Morgan and Bitterli, Benedikt and Jarosz, Wojciech: *An Efficient Denoising Algorithm for Global Illumination*
https://cs.dartmouth.edu/~wjarosz/publications/mara17towards.pdf
Juli 2017
Letzter Zugriff: 16.01.2021

[6] Jonathan Korein und Norman Badler: *Temporal Anti-Aliasing in Computer Generated Animation*
https://dl.acm.org/doi/10.1145/800059.801168
1983
Letzter Zugriff: 16.01.2021

[7] Christoph Schied, Christoph Peters und Carsten Dachsbacher: *Gradient Estimation for Real-Time Adaptive Temporal Filtering*
https://cg.ivd.kit.edu/atf.php
2018
Letzter Zugriff: 16.01.2021

[8] Nima Khademi Kalantari, Steve Bako und Pradeep Sen: *A machine learning approach for filtering Monte Carlo noise*
https://dl.acm.org/doi/10.1145/2766977
Juli 2015
Letzter Zugriff: 16.01.2021

[9] Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney und Aaron Lefohn: *Neural Temporal Adaptive Sampling and Denoising*
https://research.nvidia.com/publication/2020-05_Neural-Temporal-Adaptive
Mai 2020
Letzter Zugriff: 16.01.2021

[10] Chao Dong, Chen Change Loy, Kaiming He und Xiaoou Tang: *Image Super-Resolution Using Deep Convolutional Networks*
https://arxiv.org/abs/1501.00092
Juli 2015
Letzter Zugriff: 16.01.2021

[11] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang and Wenzhe Shi: *Photo-Realistic Single Image*

Super-Resolution Using a Generative Adversarial Network
https://arxiv.org/abs/1609.04802
Mai 2017
Letzter Zugriff: 16.01.2021

[12] Quake II RTX
https://store.steampowered.com/app/1089130/Quake_II_RTX/
06. Juni 2019
Letzter Zugriff: 16.01.2021

[13] Teardown
https://www.teardowngame.com/
https://tuxedolabs.blogspot.com/
29 Oktober 2020
Letzter Zugriff: 16.01.2021

[14] Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn und Marco Salvi: *Spatiotemporal Variance-Guided Filtering: Real-Time*
Reconstruction for Path-Traced Global Illumination
https://cg.ivd.kit.edu/publications/2017/svgf/svgf_preprint.pdf
Juli 2017
Letzter Zugriff: 17.01.2021

[15] Minecraft RTX
https://www.minecraft.net/en-us/updates/ray-tracing/
16. April 2020
Letzter Zugriff: 17.01.2021

[16] Chakravarty R. Alla Chaitanya, Anton Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, Timo Aila: *Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder*
https://research.nvidia.com/sites/default/files/publications/dnn_denoise_author.pdf
Juli 2017
Letzter Zugriff: 17.01.2021

[17] Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai und Morgan McGuire: *Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields*
http://jcgt.org/published/0008/02/01/paper-lowres.pdf
Nov 2019
Letzer Zugriff: 17.01.2021

[18] Michal Olejnik und Pawel Kozlowski: *Raytraced Shadows in Call of Duty: Modern Warfare*
https://www.activision.com/cdn/research/Raytraced_Shadows_in_Call_of_Duty_Modern_Warfare.pdf
September 2020
Letzer Zugriff: 17.01.2021