Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

# Implementierung eines sichtbarkeitserhaltenden Tone-Mapping Operators

Florian Oetke
(Matrikel-Nr. 957795)

Projektdokumentation zur Vorlesung High Performance Computing

Betreuer: Prof. Dr. Christoph Lürig

Trier, 13.06.2018

# Kurzfassung

Diese Dokumentation beschreibt die Implementierung und Funktionsweise eines Histogramm-basierten Tone-Mapping Operators zur Abbildung von Bildern mit hohem Dynamikumfang auf einen niedrigeren – von handelsüblichen Monitoren darstellbaren – Dynamikumfangs. Hierbei wurde versucht den subjektiven Betrachtungseindruck der Szene bestmöglich wiederzugeben, indem die Sichtbarkeit der Objekte im Ausgangsbild, basierend auf einem Modell der menschlichen Kontrastwahrnehmung, im Ergebnis erhalten wird.

# Inhaltsverzeichnis

# 1

# Einführung

Mit aktuellen Realtime-Rendering-Verfahren gezeichnete Szenen enthalten, wie auch reale Umgebungen, eine große Anzahl stark unterschiedlicher Helligkeitswerte. Während die menschliche Wahrnehmung in der Lage ist einen Dynamikumfang von bis zu $1.000.000 : 1$ zu unterscheiden, bieten selbst aktuelle HDR Anzeigegeräte einen Dynamikumfang von lediglich $100.000 : 1$, was eine Dynamikkompression (Tone Mapping) für die Darstellung zwingend erforderlich macht. [RAG+11; WRP97]

Vor allem aufgrund ihrer geringen Laufzeitkosten kommen im Bereich des Realtime-Renderings zu diesem Zweck oft globale Tone-Mapping Operatoren zum Einsatz. Diese verarbeiten alle Pixel der Eingabe unabhängig voneinander. Hierdurch sind sie zwar von der Laufzeit her anderen Operatoren überlegen, neigen jedoch gerade bei Szenen mit einem hohen Dynamikumfang zum Verlust von Bilddetails (siehe Abb. 1.1). [WRP97]

Daher wurde für diese Arbeit der sichtbarkeitserhaltende Algorithmus von Ward Larson et al. [WRP97] implementiert. Dieser basiert auf einer modifizierten Histogrammäqualisation, wodurch unterschiedlich helle Bereiche des Bildes mit einem individuell zugeschnittenen Faktor multipliziert werden. Hierdurch wird der Kontrast des Anzeigegeräts optimal ausgenutzt und die Sichtbarkeit im Ausgangsbild so weit wie möglich erhalten (siehe Abb. 1.2). [WRP97]

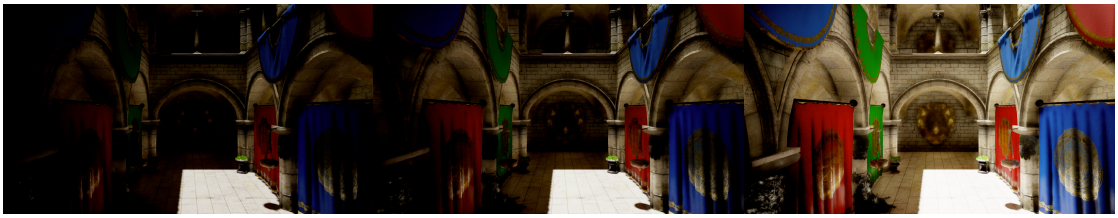


Abb. 1.1: Ein globaler Tone-Mapping Operator basierend auf einer filmischen Kurve mit unterschiedlichen Belichtungszeiten. Abhängig von der gewählten Belichtungszeit sind zwar Teile des Bildes gut abgebildet, aber der Rest über- oder unterbelichtet. Vgl. Abb. 1.2

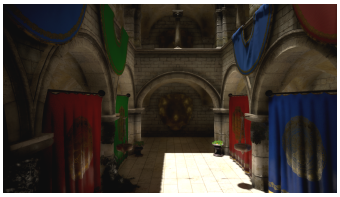


Abb. 1.2: Die selbe Szene wie in Abb. 1.1 mit dem in dieser Arbeit implementierten Tone-Mapping Operator.

# 2

# Tone-Mapping Algorithmus und Implementierung

In dieser Arbeit wurde der für das Modul *High-Performance Computing* relevante Teil des sichtbarkeitserhaltenden Tone-Mapping Operators von Ward Larson et al. [WRP97] implementiert.

Die für diese Arbeit nicht implementierten Teile des Verfahrens, simulieren Schwächen der menschlichen Wahrnehmung wie eingeschränkte Farbwahrnehmung und Auflösung in dunklen Szenen sowie Überstrahlungseffekte durch helle Oberflächen. [1] Auf die Implementierung wurde in dieser Arbeit verzichtet, da diese Effekte zum einen zur Funktionsweise und Bewertung des Tone-Mapping Verfahrens selbst nicht erforderlich sind und sie sich zum anderen trivial über die normale Grafik-Pipeline (Fragment-Shader und MIP-Mapping) implementieren lassen.

Das implementierte Verfahren basiert auf einer modifizierten Histogrammäqualisation und besteht aus den drei Einzelschritten: [WRP97]

- Generierung des Histogramms für das HDR Eingabebild
- Beschränkung des Histogramms basierend auf der menschlichen Kontrastempfindlichkeit
- Anwendung der Histogrammäqualisation auf das Eingabebild

Die Implementierung des Verfahrens erfolgte mittels Vulkan-Compute-Shadern. Hierdurch sollte sichergestellt werden, dass sich der Prozess möglichst optimal und ohne zusätzlichen Synchronisationsaufwand in den bestehenden Vulkan-Renderer integrieren lässt. Da der Funktionsumfang dieser allerdings im Wesentlichen dem von OpenGL Compute-Shadern entspricht, sind die Möglichkeiten zur Optimierung verglichen mit z.B. OpenCL leider stellenweise stark eingeschränkt. Dies betrifft vor allem das Fehlen von 8- und 16-Bit Integer Variablen, wodurch viele in der Literatur für OpenCL/CUDA vorgeschlagenen Optimierungen erschwert oder vollständig unpraktikabel werden. [VK59]

Die Interaktion mit dem Rest des Renderers erfolgt über eine 16-Bit 2D Eingabe-Textur, welche die aktuelle Szene enthält, sowie eine 16-Bit 1D Ausgabe-Textur, in welcher die Tone-Mapping-Faktoren der einzelnen Helligkeitsstufen kodiert sind. Die eigentliche Anwendung der berechneten Faktoren erfolgt später

[1] Diese Effekte werden in der Arbeit von Ward Larson et al. [WRP97] im Kapitel 5 *Human Visual Limitations* beschrieben.

basierend auf dieser Textur in einem nachgelagerten Fragment-Shader, welcher den gesamten Bildschirm umfasst.

Für die Übergabe der Texturen zwischen den Fragment- und Compute-Shadern, sowie zwischen den einzelnen Compute-Shadern selbst, ist darüber hinaus eine manuelle Synchronisation mittels Pipeline-Barriers erforderlich. Im Zuge dieser werden außerdem notwendige Layout-Transfer-Operationen der Texturen durchgeführt, da für Compute-Shader ein anderes *VkImageLayout* erforderlich ist[2] als für Fragment-Shader[3]. [VK59]

## 2.1 Generierung des Histogramms

Der erste Schritt des Verfahrens ist die Generierung eines Histogramms über den natürlichen Logarithmus der Luminanz des Eingabebildes, das später der Dynamikkompression unterzogen werden soll. Jede Klasse[4] in diesem Histogramm repräsentiert Bereiche im Bild mit ähnlicher Helligkeit, die bestmöglich in den begrenzten Dynamikumfang überführt werden sollen, als wäre das Auge optimal an diesen Helligkeitswert angepasst. [WRP97]

In der Realität erfolgt die Anpassung des Auges an eine bestimmte Umgebungshelligkeit primär innerhalb des zentralen 1° des Gesichtsfelds, d.h. der Punkt auf den das Auge aktuell fokussiert. Da es ohne zusätzliche Hardware nicht möglich ist festzustellen auf welchen Bereich des Bildes der Benutzer aktuell fokussiert, wird davon ausgegangen, dass jeder Punkt des Bildes ein potentieller Fokuspunkt ist. Hierzu wird in einem ersten Schritt basierend auf Gleichung 2.1 die Auflösung eines Foveal-Bildes bestimmt, bei dem jeder Pixel ca. 1° des Gesichtsfeldes entspricht und das anschließend als Basis für die Bestimmung des Histogramms dient. Da damit die ursprüngliche und die Zielauflösung bekannt ist, kann die hardwarebeschleunigte MIP-Map-Generierung der Grafikkarte verwendet werden um das Eingabebild effizient in eine annähernd korrekte Auflösung zu skalieren. [WRP97]

$$
\begin{aligned}
S &= \frac{2\tan(\frac{\Theta}{2})}{0.01745} \\
\text{mit } S &= \text{Breite/Höhe des Foveal-Bildes in Pixeln,} \\
\Theta &= \text{horizontales/vertikales Sichtfeld (FOV) und} \\
0.01745 &= \text{Radians in 1° des Gesichtsfeldes}
\end{aligned}
\tag{2.1}
$$

Das Histogramm besteht aus 256 Klassen um den gesamten von Menschen wahrnehmbaren Dynamikumfang bis zu $10^6 cd/m^2$ mit ausreichender Präzision zu erfassen. Da mit dem Logarithmus der Luminanz gearbeitet wird und $log(0) = -\infty$ muss außerdem eine minimale Luminanz festgelegt werden. Für diese Arbeit wurde hierfür die untere Grenze der menschlichen Wahrnehmung mit $10^{-4} cd/m^2$ gewählt. [WRP97]

[2] *VK_IMAGE_LAYOUT_GENERAL*
[3] *VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL*
[4] engl. Bin

Der nächste Schritt ist die eigentliche Berechnung des Histogramms. Dies wurde mittels eines Vulkan Compute-Shaders umgesetzt, der die RGB-Werte des skalierten Eingabebildes ausließt, den Logarithmus der Luminanz mittels $\log(c_{rgb} \bullet \langle 0.2126, 0.7152, 0.0722 \rangle^T)$ berechnet und abhängig von diesem Werte den Zähler der entsprechenden Histogramm-Klasse inkrementiert. Diese Operationen können für mehrere Pixel parallel durchgeführt werden, wobei die Inkrementierung selbstverständlich atomar erfolgen muss. Atomare Operationen auf globalem Speicher sind allerdings, verglichen mit geteiltem Speicher[5], mit höheren Kosten verbunden. Darüber hinaus kommt es durch viele atomare Zugriffe auf den selben Speicherbereich zu längeren Wartezeiten pro Aufruf (contention). Daher wird zuerst ein lokales Histogramm pro WorkGroup berechnet, welches erst am Ende atomar in das globale Histogramm eingerechnet wird.

Da die Histogrammberechnung auf einem zweidimensionalen Eingabebild operiert, ist die Anzahl der WorkGroups ebenfalls zweidimensional und basiert auf der Auflösung der Eingabe. Die Größe der WorkGroups stellt eine Abwägung zwischen der begrenzten maximalen Anzahl parallel ausgeführter WorkGroups und dem höheren Synchronisationsaufwands für die atomaren Zugriffe dar und wurde basierend auf den gemessenen Laufzeiten (siehe Abb. 3.2 in Kapitel 3) mit 32x32 gewählt, wobei jeder Shader-Aufruf 8x8 Pixel verarbeitet. Die Position der in jedem Aufruf verarbeiteten Pixel bestimmt sich dabei so, dass benachbarte Warps auch benachbarte Bereiche innerhalb der Textur auslesen[6].

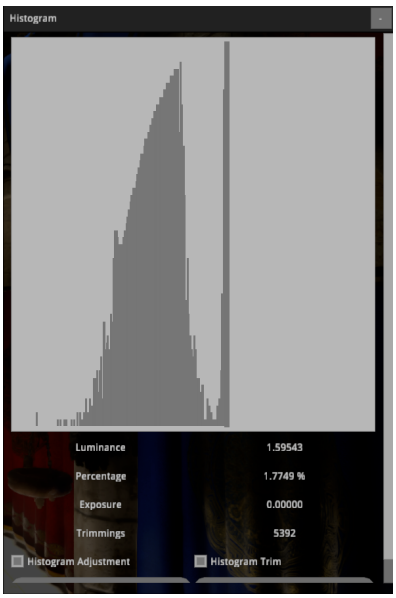



Abb. 2.1: Die Visualisierung des Berechneten Histogramms mit den wichtigsten Daten wie Luminanzwert und Häufigkeit der ausgewählten Klasse.

Zur Visualisierung des berechneten Histogramms kommen drei getrennte Speicherbereich zum Einsatz, die abwechselnd beschrieben werden. Daher ist zum Zugriff auf die jeweils ältesten Ergebnisse keine zusätzliche Synchronisation notwendig. Die Visualisierung selbst erfolgt schließlich in der auf der Nuklear-Bibliothek basierten Benutzeroberfläche (siehe Abb. 2.1).

[5] engl. shared memory. Ein Speicherbereich, der zwischen allen Aufrufen innerhalb einer Workgroup geteilt wird.

[6] engl. coalesced memory reads

## 2.2 Histogrammäqualisation

Die Histogrammäqualisation ist der Arbeitsschritt welcher die Helligkeitsbereiche des Bildes abhängig von ihrer Größe vergrößert oder verkleinert um eine bessere Ausnutzung des Dynamikumfangs zu erzielen. Das Ergebnis ist ein neuer Helligkeitswert pro Histogramm-Klasse, der sich aus Gleichung 2.2 ergibt. Zur Anwendung auf das Eingabebild wird pro Klasse aus der Ein- und Ausgabehelligkeit ein Faktor berechnet und in eine eindimensionale Textur gespeichert. Diese wird in einem eigenen Fragment-Shader mit der Eingabe multipliziert, indem mittels linearer Interpolation ein Faktor für den aktuellen Helligkeitswert ausgelesen wird. Auf diese Weise werden trotz der relativ geringen Anzahl an Klassen Banding-Artefakte verhindert. [WRP97]

$$
\begin{aligned}
B_{de} &= \log(L_{dmin}) + [\log(L_{dmax}) - \log(L_{dmin})] \cdot \frac{\sum_{b_i<B_w} f(b_i)}{\sum_{b_i} f(b_i)} \\
\text{mit } B_w &= \text{Helligkeit (Logarithmus der Luminanz) der Eingabe,} \\
B_{de} &= \text{Helligkeit (Logarithmus der Luminanz) in der Ausgabe,} \\
L_{dmin} &= \text{Minimale Luminanz des Anzeigegeräts (Schwarzlevel),} \\
L_{dmax} &= \text{Maximale Luminanz des Ausgabegeräts und} \\
f(b_i) &= \text{Häufigkeit der Histogramm-Klasse } b_i
\end{aligned}
\tag{2.2}
$$

Zur effizienten Lösung von Gleichung 2.2 ist eine Präfix-Summe über alle Histogramm-Klassen notwendig. Um diese parallel zu berechnen wurde der GPU Scan-Algorithmus aus Gpu Gems 3[Ngu07] implementiert. Abweichend von dem dort beschriebenen vorgehen schreibt der hier implementierte Algorithmus das Ergebnis allerdings nicht zurück in das Eingabe-Array (in-place), sondern in einen temporären geteilten Speicherbereich, da das ursprüngliche Histogramm auch noch in späteren Arbeitsschritten benötigt wird.

Das Verfahren besteht aus zwei Phasen, die jeweils am einfachsten über einen Binärbaum, wie in Abb. 2.2 dargestellt, visualisiert werden können. Jede dieser Phasen besteht aus $log_2(N)$ Iterationen mit einer bis $\frac{N}{2}$ Operationen pro Iteration, die als Breitensuche über den Baum interpretiert werden können. Die Parallelisierung erfolgt dabei trivial über parallele Verarbeitung der Knoten auf dem selben Level und Synchronisation zwischen den Iterationen/Leveln. [Ngu07]

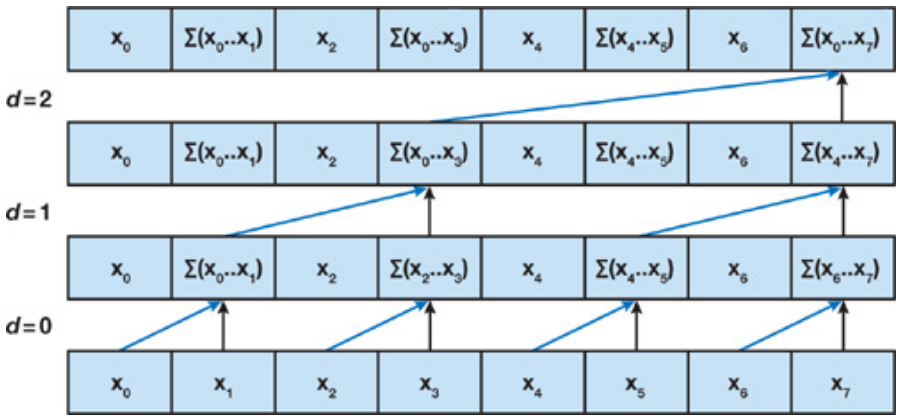
Die erste Phase (Up-Sweep, Abb. 2.2a) ist eine Iteration von den Blättern bis zur Wurzel, wobei jeweils die direkten Kinder addiert und im aktuellen Knoten gespeichert werden. Im Ergebnis dieses Schrittes stehen an jeder geraden Position im Array die Summe vom ersten bis zum aktuellen Knoten (inklusive). [Ngu07]

Die zweite Phase (Down-Sweep, Abb. 2.2b) fasst die Teilsummen im Ergebnis der ersten Phase zur finalen Präfix-Summe zusammen. Hierzu wird als erstes der Wert der Wurzel auf Null gesetzt und anschließend von der Wurzel bis zu den Blättern iteriert. Dabei wird bei jedem Knoten die Summe mit seinem linken Kind in den rechten Kindknoten geschrieben (blauer Pfeil) und der Wert des linken mit dem des aktuellen überschrieben (oranger Pfeil). [Ngu07]

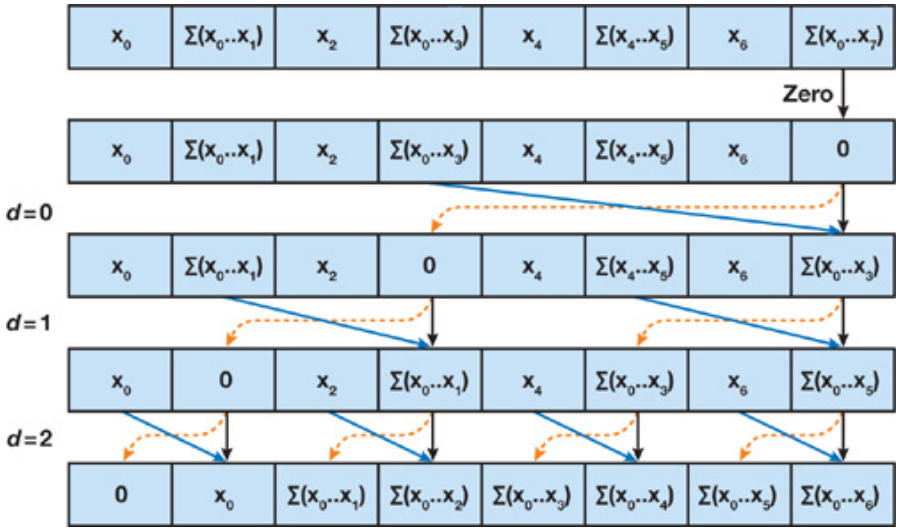
Das Ergebnis ist anschließend die Präfix-Summe der Eingabe wie in der letzten Zeile von Abb. 2.2b dargestellt. [Ngu07]

Für die Implementierung konnte der in Gpu Gems 3[Ngu07] bereitgestellte Code, mit geringfügigen Änderungen der verwendeten Typen und Funktionen, nahezu unverändert übernommen werden.

Die zusätzlichen in Gpu Gems 3[Ngu07] beschriebenen Verbesserungen, wie zur Vermeidung von Bank-Konflikten, wurden in dieser Arbeit aufgrund der geringen statischen Array-Größe von 256 und der durch die Compute-Shader bedingten Limitierungen (u.a. nur 32 Bit Integer) nicht umgesetzt.




(a) Initiale Up-Sweep-Phase

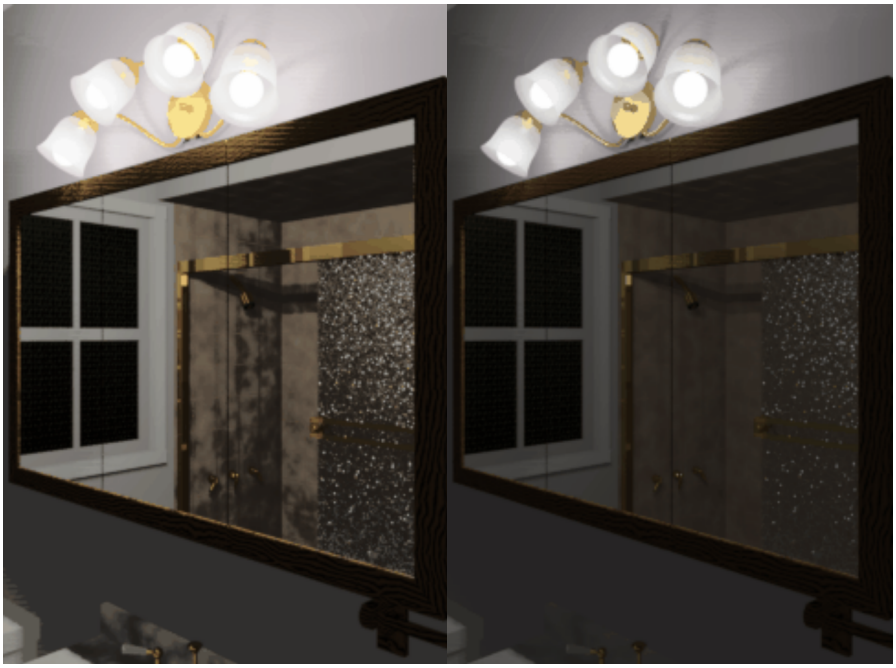



(b) Finale Down-Sweep-Phase

Abb. 2.2: Illustration der beiden Phase des Scan-Algorithmus. Das Ergebnis der ersten Phase (oberste Zeile in a) bildet die Eingabe der zweiten Phase (oberste Zeile in b). [Ngu07]

## 2.3 Beschränkung des Histogramms

Eine naive Histogrammäqualisation, wie im vorherigen Kapitel beschrieben, führt zwar die gewünschte Kontrastkompression durch, verursacht allerdings auch eine Kontrastspreizung in Bereichen mit vielen Messwerten. Hierdurch werden eigentlich kleine Helligkeitsbereichen über einen zu großen Teil des verfügbaren Dynamikumfangs ausgebreitet, was zum unerwünschten Eindruck eines überzeichneten Kontrastes führt (siehe Abb. 2.3). Die Lösung dieses Problems ist die Klassen des Histogramms – und damit auch ihre Ausbreitung im Dynamikumfang – vor der Äqualisation auf einen bestimmten Maximalwert zu beschränken. [WRP97]



(a) (b)

Abb. 2.3: Vergleich der naiven (a) und beschränkten Histogrammäqualisation (b). Die unerwünschte Kontrastspreizung ist bei den Fließen im Hintergrund deutlich zu erkennen. [WRP97]

Das Maximum der Klassen ergibt sich dabei aus der Gleichung 2.3 von Ward Larson et al. [WRP97]. Diese bestimmt das Maximum basierend auf der menschlichen Kontrastwahrnehmung, so dass ein gerade so wahrnehmbarer Helligkeitsunterschied[7] (JND) in der Eingabe gerade so einer JND in der Ausgabe entspricht. Hierzu dient die Funktion $\Delta L_t(L_a)$ in Gleichung 2.4, welche für eine bestimmte Luminanz die gerade noch von Menschen wahrnehmbare Änderung bestimmt. [WRP97]

[7] engl. Just noticable difference.

$$
\begin{aligned}
f(B_w) \leq \frac{\Delta L_t(L_d)}{\Delta L_t(L_w)} \cdot \frac{\sum_{b_i} f(b_i) \cdot \Delta b \cdot L_w}{[\log(L_{dmax}) - \log(L_{dmin})] L_d} \\
\text{mit } \Delta L_t &= \text{JND-Luminanz Funktion aus Gleichung 2.4,} \\
\Delta b &= \text{Breite einer Histogramm-Klasse,} \\
L_w &= \text{Luminanz der Eingabe } (= \mathrm{e}^{B_w}), \\
L_d &= \text{Luminanz der Ausgabe (siehe Gleichung 2.2) und} \\
L_{dmax}, L_{dmin} &= \text{Minimale/Maximale Luminanz des Ausgabegeräts}
\end{aligned}
\tag{2.3}
$$

$$
\Delta L_t(L_a) = \begin{cases}
10^{-2.86} & \text{für} \quad \log_{10}(L_a) \leq -3.94 \\
10^{(0.405 \cdot \log_{10}(L_a) + 1.6)^{2.18} - 2.86} & \text{für} -3.94 \leq \log_{10}(L_a) \leq -1.44 \\
10^{\log_{10}(L_a) - 0.395} & \text{für} -1.44 \leq \log_{10}(L_a) \leq -0.0184 \\
10^{(0.249 \cdot \log_{10}(L_a) + 0.65)^{2.7} - 0.72} & \text{für} -0.0184 \leq \log_{10}(L_a) \leq 1.9 \\
10^{\log_{10}(L_a) - 1.255} & \text{sonst}
\end{cases}
\tag{2.4}
$$

Die Limitierung der Histogrammklassen erfolgt analog zur Arbeit von Ward Larson et al. [WRP97] durch Reduzierung der Klassen, die oberhalb des Limits liegen, ohne eine Umverteilung der Werte auf andere Klassen durchzuführen. Da sich hierdurch allerdings u.a. auch die Summe aller Klassen reduziert, die in Gleichung 2.3 verwendet wird, kann sich nach der Limitierung das Limit erneut geändert haben. Daher erfolgt die Begrenzung des Histogramms Iterativ, bis eine Toleranzschwelle[8] unterschritten wurde. [WRP97]

Ein Sonderfall, der noch berücksichtigt werden muss, ist das während der Iterativen Lösung die verbliebene Summe aller Klassen kleiner als die Toleranzschwelle werden kann. Dieser Fall tritt ein wenn die Größe des Dynamikumfangs der Eingabe nicht größer als die des Anzeigegeräts ist, d.h. keine Dynamikkompression für die Darstellung notwendig ist. Daher wird in dieser Situation die Iteration abgebrochen und statt dessen ein linearer Operator verwendet, indem alle Histogrammklassen auf eine feste Konstante gesetzt werden bevor die Histogrammäqualisation durchgeführt wird. [WRP97]

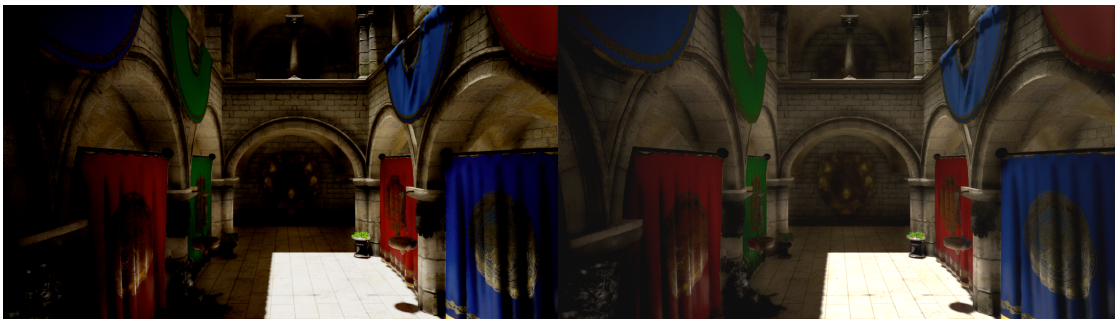
Für die Parallelisierung der Histogrammbeschränkung wurden zwei wichtige Ansätze verfolgt. Der erste ist, dass durch $L_d$ in Gleichung 2.3 eine Histogrammäqualisation pro Iteration durchgeführt werden muss und somit jedes Mal eine neue Präfix-Summe des Histogramms berechnet werden muss. Die Parallelisierung dieses Teilschrittes erfolgt somit mit dem in Kapitel 2.2 beschriebenen Algorithmus. Der zweite triviale Ansatzpunkt ist, dass pro Iteration für jede Klasse das Maximum bestimmt, ggf. die Klasse verkleinert und die Abgeschnittenen Werte aufsummiert werden müssen. Da durch die Parallelisierung der Präfix-Summe bereits eine WorkGroup-Größe von 128 verwendet werden muss, erfolgen diese Arbeitsschritte pro Iteration ebenfalls parallel, wobei nach jeder Iteration eine Synchronisation durchgeführt werden muss um das Abbruchkriterium zu prüfen.

[8] Summe der in dieser Iteration abgeschnittenen Werte ist ≤ 0,01 % der Ausgangsgröße.

# 3

# Auswertung

Der visuelle Eindruck der Ergebnisse ist bei Standbildern sehr positiv und überzeugt durch eine deutlich bessere Sichtbarkeit ohne die Notwendigkeit eine Belichtungszeit festzulegen (siehe Abb. 3.1).



(a) Globaler Operator (b) Lokaler Operator

Abb. 3.1: Das Ergebnis des implementierten Verfahrens im Vergleich mit einem traditionellen globalen Tone-Mapping Operator.

Das Laufzeitverhalten[1] des Verfahrens ist mit 0,25 ms ebenfalls sehr gut und sollte auch auf leistungsschwächeren Systemen keine merklichen negativen Einflüsse bedeuten. Die Laufzeiten der einzelnen Schritte beläuft sich dabei auf 0,13 ms für die Berechnung des Foveal-Bildes, 0,01 ms zur Bestimmung des Histogramms, 0,09 ms für die in Kapitel 2.3 beschriebene Einschränkung des Histogramms und 0,01 ms für die Berechnung der Textur mit den Faktoren.

Um den Erfolg der durchgeführten Optimierungen bewerten zu können, wurden außerdem Messungen der Laufzeit mit den nicht optimierten Varianten durchgeführt.

Der erste in dieser Form getestete Teilschritt ist die Beschränkung des Histogramms. Hier lag die Laufzeit des sequenziellen Algorithmus (WorkGroup Größe von 1 und naive Berechnung der Präfix-Summe) bei 0,22 ms. Durch die Optimierung und parallele Durchführung konnten diese Berechnung somit um einen Faktor von mehr als zwei verbessert werden.

[1] Alle Messungen in dieser Arbeit erfolgten mit einer NVIDIA GeForce GTX 1060.

Der andere betrachtete Teilschritt ist die Berechnung des Histogramms. Zur besseren Vergleichbarkeit, aufgrund der normalerweise sehr geringen Laufzeit, wurde die Berechnung in voller Auflösung von 1920x1080 durchgeführt. Eine sequentielle Durchführung auf der GPU ist mit ca. 3027,5 ms erwartungsgemäß nicht praktikabel. Einen besseren Vergleichswert liefert die nicht optimierte Berechnung ausschließlich mit atomaren Zugriffen auf globalen Speicher, welche eine Laufzeit von 24,2 ms aufwies. Verglichen mit der optimierten Laufzeit von 0,08 ms war diese Optimierung mit einem Faktor von über 300 somit ebenfalls sehr erfolgreich.

Die letzte wichtige Messung dient zur Bestimmung der in Kapitel 2.1 beschriebenen WorkGroup- und Batch-Größe der Histogrammberechnung. Diese wurde ebenfalls in voller Auflösung durchgeführt, sowie anschließend in normaler Auflösung verifiziert. Wie aus den Ergebnissen in Abb. 3.2 hervorgeht liegt die optimale Größe der WorkGroups bei 32x32 mit einer Batch-Größe von 8x8. Diese Werte wurden daher auch für die anderen oben genannten Messungen übernommen. Das stabile Umfeld lässt darüber hinaus darauf schließen, dass auch auf geringfügig abweichender Hardware mit einem ähnlichen Verhalten zu rechnen ist.

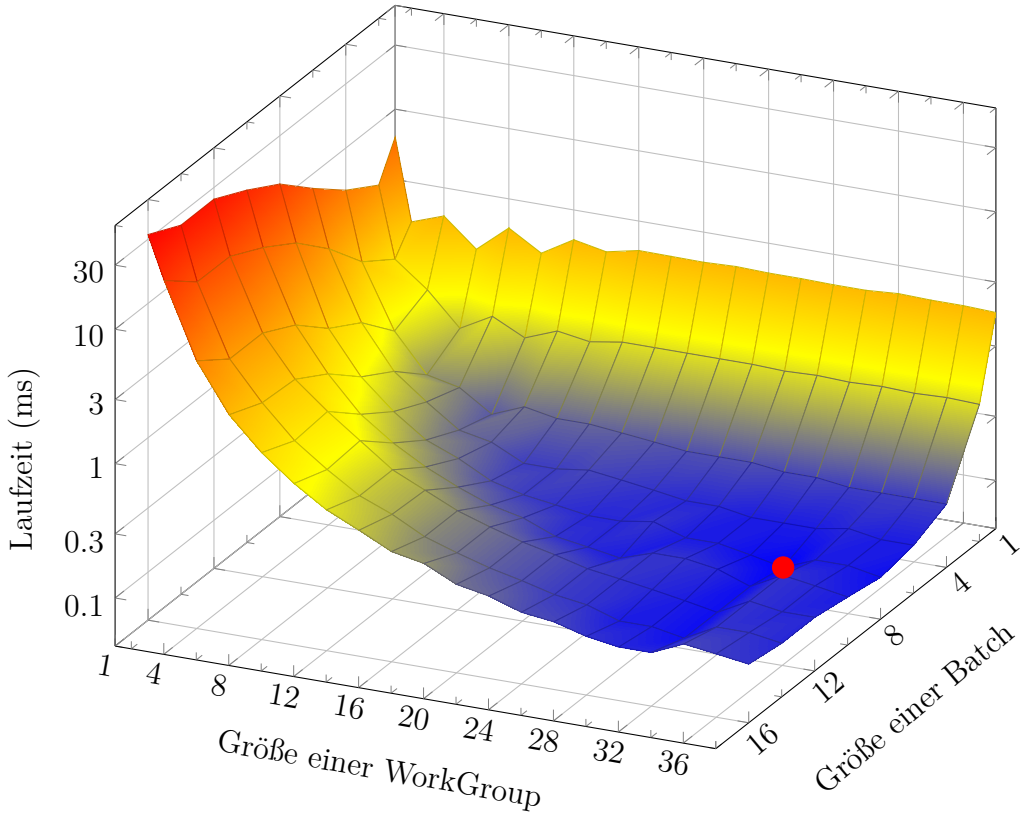



Abb. 3.2: Laufzeit-Visualisierung der Histogramm-Generierung bei unterschiedlichen WorkGroup- und Batch-Größen. Zur besseren Vergleichbarkeit wurde die Histogramm-Generierung bei voller Auflösung (1920x1080) durchgeführt. Die geringste Laufzeit mit 0.08 ms ergibt sich bei einer WorkGroup-Größe von 32 und einer Batch-Größe von 8 (rote Markierung in der Abbildung). Hierbei ist allerdings zu beachten, dass die umliegenden Werte in einem unter Berücksichtigung der Messtoleranz nahezu identischem Bereich liegen ($< 0.15$).

# 4

# Fazit

Das implementierte Verfahren liefert bei Standbildern sehr gute Ergebnisse bei geringen Laufzeitkosten, ist unter wechselnder Beleuchtung allerdings teilweise instabil. Durch eine Interpolation der berechneten Faktoren kann dies zwar größtenteils ausgeglichen werden, es gibt allerdings mit dem modifizierten Verfahren von Irawan, Ferwerda und Marschner [IFM05] auch eine bessere Alternative. Dieses löst nicht nur das genannte Problem der Instabilität, sondern ergänzt auch einen Adaptionseffekt (siehe Abb. 4.1), der in den meisten Anwendungen wünschenswert ist um große Helligkeitsänderungen zu kommunizieren. Das Verfahren ist allerdings komplexer und schlechter parallelisierbar als das in dieser Arbeit implementierte. Eine spätere Erweiterung auf das Verfahren von Irawan et al. [IFM05] ist jedoch sinnvoll möglich und wird ggf. im Anschluss an diese Arbeit erfolgen.

Abb. 4.1: Ein Beispiel des Adaptionseffekts während einer Kamerafahrt durch einen Tunnel beim Verfahren von Irawan, Ferwerda und Marschner [IFM05]. [IFM05]

## Literaturverzeichnis

[IFM05] Piti Irawan, James A. Ferwerda und Stephen R. Marschner. Perceptually based tone mapping of high dynamic range image streams. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, Seiten 231–242, Konstanz, Germany. Eurographics Association, 2005. ISBN: 3-905673-23-1. DOI: 10.2312/EGWR/EGSR05/231-242. URL: https://www.graphics.cornell.edu/pubs/2005/IFM05.pdf.

[Ngu07] Hubert Nguyen. Parallel prefix sum (scan) with cuda. In *Gpu Gems 3*, Kapitel 39. Addison-Wesley Professional, first Auflage, 2007. ISBN: 9780321545428. URL: https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch39.html.

[RAG+11] Ana Radonjić, Sarah R. Allred, Alan L. Gilchrist und David H. Brainard. The dynamic range of human lightness perception. *Current Biology*, 21(22):1931–1936, 2011. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2011.10.013. URL: http://www.sciencedirect.com/science/article/pii/S0960982211011377.

[VK59] The Khronos Vulkan Working Group. Vulkan® 1.0.59 - A Specification, 2017. URL: https://www.khronos.org/registry/vulkan/specs/1.0-extensions/pdf/vkspec.pdf (besucht am 18. 09. 2017).

[WRP97] Gregory Ward Larson, Holly Rushmeier und Christine Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. In *ACM SIGGRAPH 97 Visual Proceedings: The Art and Interdisciplinary Programs of SIGGRAPH '97*, SIGGRAPH '97, Seiten 155–, Los Angeles, California, USA. ACM, 1997. ISBN: 0-89791-921-1. DOI: 10.1145/259081.259242. URL: http://radsite.lbl.gov/radiance/papers/lbnl39882/tonemap.pdf.