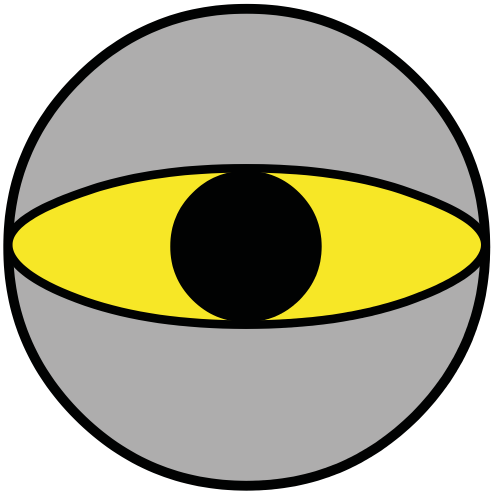
spy360.net



webcampanoramen bis 360° in hdr

Diplomarbeit

Lehrstuhl für Graphisch Interaktive Systeme
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften
Universität Tübingen

Prof. Dr. Andreas Schilling

Marc-Oliver Pahl

Tag der Abgabe: 2. Juni 2008

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Tübingen, den 2. Juni 2008

Kurzfassung:

Die Arbeit stellt eine ausführliche Einführung in die Themenfelder High Dynamic Range, Panoramastitching und Tonemapping dar.

Im Bereich High Dynamic Range werden alle Veröffentlichungen zur Kameraresponsekurvenbestimmung von *Debevec und Malik* [1997] bis 2008 vorgestellt.

Im Bereich Panoramastitching wird insbesondere auf Parallaxe und SIFT eingegangen.

Im Tonemappingteil werden Grundlagen zur fundierten qualitativen Beurteilung von Tonemappingoperatoren gegeben.

Im praktischen Teil der Arbeit wird die spy360-Software zur automatisierten Erstellung von HDR-Panoramen bis 360° unter Windows entwickelt sowie der dabei entstandene Belichtungsänderungsdetektionsalgorithmus (BÄD) vorgestellt.

Inhaltsverzeichnis

1. Einleitung

We now live in a Global Village.

Marshall McLuhan in The Gutenberg Galaxy, 1962.

1.1 Motivation der Arbeit

Durch die Verbreitung des Internet ist die Welt heute mehr denn je zum „globalen Dorf“ geworden, wie es der Medientheoretiker McLuhan schon 1962 in Bezug auf die Verbreitung des Fernsehens formulierte [*Mcluhan*, 1962]. Mit einem Klick können wir Informationen aus der ganzen Welt auf unseren Bildschirm holen.
Im Gegensatz zu McLuhans passivem Medium Fernsehen kann das Internet aktiv und damit vielfältiger genutzt werden: Wir können auswählen, was wir rezipieren möchten, Inhalte und Darstellung manipulieren, bidirektional (a-)synchron kommunizieren...

Ein wichtiger Aspekt des Zusammenrückens bleibt auch im Internetzeitalter das Bild. Es sagt bekanntlich mehr als tausend Worte und bietet in der Tat eine gute Möglichkeit, einen Eindruck von einem entfernten Ort, von Menschen oder anderen Dingen zu gewinnen.

Einen besonders guten Eindruck von einem Ort vermittelt ein kontrastreiches Panoramabild – also ein Bild mit realitätsnaher Dynamik und großem Betrachtungswinkel bis zu 360°.

1.2 Zielsetzung der Arbeit

In dieser Arbeit möchte ich mich mit der automatischen Erzeugung qualitativ hochwertiger (Panorama-)Bilder beschäftigen. Zum Einsatz kommen sollen dabei einfache Consumer-Webcams. Spezialequipment soll nicht verwendet werden. Die Kameras sollen nicht notwendigerweise zentral montiert sein.

Kurz: Das System soll so kostengünstig und unproblematisch in der Montage sein, dass es von „Jedermann“ eingesetzt werden kann.

Um dieses Ziel zu erreichen, bedarf es einigen Hintergrundwissens bezüglich möglicher einzusetzender Verfahren. Wie wir an der Gliederung (Abschnitt 1.4) sehen werden, müssen wir die Schritte High Dynamic Range-Bildgewinnung (HDR-Bildgewinnung), Stitching und Tonemapping betrachten.

HDRI (HDR-Imaging) ist eine Technologie, mit der wir in den nächsten Jahren zunehmend in Berührung kommen werden. Eingehende Recherche hat jedoch gezeigt, dass es keine geeignete Einführung in die Forschung im Bereich der HDR-Bildgewinnung –vor allem im für uns relevanten Bereich der Softwarealgorithmen– gibt. Im einzigen wissenschaftlichen Buch zum Thema „High Dynamic Range Imaging“ legen die Autoren [*Reinhard et al.*, 2005] den Fokus hauptsächlich auf das Tonemapping (siehe Kapitel 4) und Image-Based Lighting[1]. Wie wir in der Abgrenzung der Arbeit (Abschnitt 1.3) sehen werden, behandeln sie die Bildgewinnung nur sehr kurz.

Ziel dieser Arbeit ist es daher, einen fundierten wissenschaftlichen Überblick über die Gewinnung von HDR-Bilddaten zu geben. Aufgrund der Zielsetzung der Arbeit, mit kostengünstiger Standardhardware zu arbeiten, soll sich die Betrachtung vor allem mit der softwareseitigen Verarbeitung befassen.

In die beiden anderen genannten Themengebiete Stitching und Tonemapping soll der Leser vom wissenschaftlichen Standpunkt aus ebenfalls so weit eingeführt werden, dass er in der Lage ist, die später in der Software eingesetzten Verfahren zu verstehen, und wissenschaftliche Veröffentlichungen aus dem Bereich nachzuvollziehen.

Das praktische Ziel der Arbeit ist die Erstellung einer Hardware-/ Softwarelösung zum automatisierten Erstellen von 360°-HDR- und optimal belichteten LDR-Panoramen.

[1] Beim Image-Based Lighting verwendet man ein HDR-Bild als Ausgangsmaterial, um eine Szene realistisch zu beleuchten. Das Bild modelliert die Lichtquelle.

1.3 Abgrenzung der Arbeit

Eine, in beschriebener Weise geeignete, wissenschaftlich fundierte, Einführung in die Thematik existiert –zumindest im Bereich der HDR-Bildgewinnung– nicht.

Zum Thema HDR gibt es bis heute (2008) nur ein wissenschaftliches Buch: *Reinhard et al.* [2005]. Die Autoren gehen darin auf *einen* Algorithmus zur Gewinnung der Kameraresponsekurve ein (*Mitsunaga und Nayar* [1999]). Wie sich herausstellen wird, ist dieser Algorithmus geeignet.

Es gibt allerdings sowohl Weiterentwicklungen dieses Ansatzes als auch einige Verfahren auf anderer Basis. Diese sollen im ersten Teil ausführlich betrachtet werden. Die Arbeit vermittelt so einen umfassenden Überblick über die geschichtliche Entwicklung hin zu den aktuellen Forschungsergebnissen auf dem Gebiet der Responsekurvenbestimmung. Eine solche umfassende *Übersicht* **existiert noch nicht**. Die in einigen Veröffentlichungen beinhalteten, überblickartigen Bestandteile, behandeln stetes nur einen Ausschnitt der Forschung auf dem Gebiet. Sie betrachten die Algorithmen ebenfalls nicht in der hier dargebotenen Tiefe.

Das Thema Stitching wird in dieser Arbeit umfassender als in den vorhandenen, recherchierten, Veröffentlichungen abgehandelt werden.

Eine gute Einführung in das Tonemapping wird in dem bereits zitierten Buch von *Reinhard et al.* [2005] gegeben. Die Darstellung in der vorliegenden Arbeit ist geraffter, und versucht dem Leser einen schnellen, zugleich aber genügend tiefgehenden Überblick zu verschaffen.

Insgesamt liegt der Fokus in den theoretischen Kapiteln darauf, dem noch nicht vollständig in der Materie beheimateten interessierten Leser eine Einführung und einen Überblick über die jeweilige Thematik zu geben. Einen wichtigen Raum nehmen dabei auch relevante benachbarte Forschungsgebiete wie Mathematik, Physik (Optik), Wahrnehmungspsychologie, Biologie und Chemie ein. An diese knüpft die Informatik an mehreren Stellen der Arbeit an. Diese „Exkurse“ können auch für den erfahreneren Leser interessant sein und dienen der Motivation des informatischen Handelns.

Softwarelösungen zur automatischen Erstellung von (Panorama-)Bildern existieren bereits. Eine Lösung für die *automatisierte Aufnahme und Erstellung qualitativ sehr hochwertiger* (Panorama-)Bilder mit *einfacher Consumer-Hardware* existiert jedoch nicht.

Die vorhandenen Lösungen basieren zumeist auf Spezialhardware[2] und teuren Komponenten. Zum großen Teil benötigen sie außerdem einen besonderen Standort – beispielsweise mit Rundumsicht.

Existierende Softwarelösungen, die ebenfalls mit Standard-Consumer-Hardware arbeiten, verwenden in der Regel direkt die (oftmals schlechten) Kamerabilder. Dies liefert keine (qualitativ) optimalen Ergebnisse.

Neu in Bezug auf die zu entwickelnde Software ist sowohl der Ansatz, *optimal belichtete* Bildergebnisse mit der vorhandenen preisgünstigen Hardware zu erzielen, als auch eine –bis zu einem gewissen Grad– *dezentrale* Montage der Kameras zu erlauben – beispielsweise in den Fenstern eines Turmes.

[2] Beispielsweise Pilzspiegel über der Kamera, Fischauge, durch Motor rotierende Kamera

1.4 Gliederung der Arbeit

Die Arbeit gliedert sich folgende großen Abschnitte:

- High Dynamic Range Imaging (2)
- Stitching von Bildern (3)
- Tonemapping (4)
- spy360.net (5)
- Zusammenfassung und Fazit (6)
- Ausblick (7)

Die theoretischen Kapitel (2-4) folgen dem Aufbau:

- Motivation und Einführung
- Präsentation und Analyse bestehender Ansätze
- Zusammenfassung

Das zweite Kapitel beschäftigt sich mit der HDR-Gewinnung mithilfe von Hard- und Software. Es geht insbesondere auf die Algorithmen zur Berechnung der Kameraresponsekurve ein und gibt einen umfassenden Überblick über die bestehenden Forschungsarbeiten. Dabei wird auch auf Probleme der digitalen Bildgewinnung im Allgemeinen und im Speziellen für die vorgestellte LDR-/ HDR-Verarbeitung eingegangen.

Das dritte Kapitel gibt eine Einführung in die Problematik des Stitching, also des Zusammenfügens von Bildern. Dabei werden auch die Kalibrierung der intrinsischen Kameraparameter, die Überblendung der Einzelaufnahmen zu einem Gesamtpanorama und das Problem der Parallaxe behandelt. Außerdem werden die fünf Seidelschen Linsenaberrationen als mögliche Probleme für diesen Teil der Arbeit betrachtet.

In Kapitel vier geht es um das Tonemapping, also die Abbildung von HDR-Bilddaten auf ein LDR-Bild. Hier wird ebenfalls eine Einführung aus historischer und mathematischer Sicht gegeben. Außerdem wird in diesem Zusammenhang die visuelle Wahrnehmung des Menschen betrachtet.

In Kapitel fünf wird die im Rahmen dieser Arbeit entstandene Software vorgestellt.

Die Arbeit schließt in Kapitel sechs mit der Zusammenfassung und dem Fazit.

Kapitel sieben gibt einen Ausblick, wie die Arbeit weitergeführt werden könnte.

2. High Dynamic Range Imaging

> On ne voit bien qu´avec le coeur,
> l´essentiel est invisible pour les
> yeux.
>
> Antoine de Saint-Exupéry dans Le petit prince, 1943

Dieses Kapitel behandelt die Gewinnung von High Dynamic Range-Bildern (HDRI). Nach der Darstellung von grundlegenden Aspekten der digitalen Bildaufnahme und sich daraus ergebenden Problemen werden wir chronologisch die Veröffentlichngen zur Erzeugung von High Dynamic Range-Bildern mit LDR-Hardware betrachten. Dabei werden alle relevanten Veröffentlichungen behandelt[3] und anschließend bewertet. Abschließend werden zwei für die vorgestellten Verfahren spezifische Probleme und deren Lösungsmöglichkeiten betrachtet.

2.1 Grundbegrifflichkeit

Die Begriffe Kontrast und Dynamik mit verschiedenen Suffixen erscheinen in diversen Publikationen. Ebenso ist oft die Rede von Blenden. Wir wollen die Begriffe daher kurz definieren:

Als **Kontrast**(-umfang) bezeichnet man in der **Physik** den Quotienten der Helligkeiten[4] [*Hecht*, 2005; *Bergmann und Schaefer*, 2004b]:

$$K := \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$

Ein ideales System erreicht einen Kontrast von 1 ($I_{\min} = 0$). Bei einem Kontrast von 0 sind keine Helligkeitsunterschiede zu erkennen. In dieser Form wird der Begriff in der Computergrafik und Fotografie nur selten gebraucht.

[3] Eine Gewähr auf Vollständigkeit kann natürlich nicht gegeben werden. Es wurden dennoch sorgfältig alle bei IEEE, ACM und Citeseer auffindbaren Paper und deren Quellen recherchiert.

[4] Unter Helleigkeit verstehen wir in der Fotografie die Leuchtdichte

Im Sprachgebrauch von High Dynamic Range Imaging wird Kontrast oft synonym für das **Kontrastverhältnis** verwendet. Die Notation für das Kontrastverhältnis ist [*Brockhaus*, 2006a]:

$$\text{Kontrastverhältnis} := I_{\max} : I_{\min}$$

Das Ergebnis der Quotientenbildung wird als **Dynamik**(-umfang) bezeichnet und in Dezibel angegeben [*Nayar und Mitsunaga*, 2000]:

$$\text{Dynamik} := 20 \log \frac{I_{\max}}{I_{\min}} dB$$

Bei logarithmischen Größen entspricht die Division einer Differenzbildung ($\log \frac{I_1}{I_2} = \log I_1 - \log I_2$).

Der Kontrast wird in der Fotografie oft auch in Blenden angegeben. Die Einheit einer **Blende** bezeichnet dabei eine Verdoppelung der Helligkeit:

$$[1 Blende] \equiv 2 \cdot I$$

Die Begriffe Kontrast, Kontrastumfang, Kontrastverhältnis, Dynamik, Dynamikumfang und Dynamikverhältnis werden in der Computergrafik oftmals nicht sauber getrennt. Vielfach werden sie synonym gebraucht. Die jeweilig implizierte Bedeutung ergibt sich aus dem Kontext und der Einheit.

Die Ausgabe eines digitalen Helligkeitssensors (CCD/ CMOS) ist linear abhängig von der einfallenden Lichtmenge (vgl. Abschnitt 2.2.2). Daraus folgt, dass die Blenden gerade der Bitanzahl eines Sensors entsprechen: Die Hinzunahme eines Bit erlaubt es, doppelt so viele Helligkeitswerte aufzunehmen[5]: Ein CCD mit einer Dynamik von 8 Bit bildet einen Kontrast von 8 Blenden oder $2^8 = 256$ (48dB) ab.

Als **Low Dynamic Range** (**LDR**) bezeichnet man die Dynamik der klassischen Aufnahmemedien wie Foto, Film etc. Die Dynamik liegt ungefähr im Bereich $<$ 80dB.

Als **High Dynamic Range** (**HDR**) bezeichnet man eine Dynamik, die deutlich höher liegt. Ziel ist es, den (wahrnehmbaren) Kontrast der Realität zu erreichen. Die Dynamik liegt bei HDR im Bereich $\gg$ 80dB.

In Tabelle 2.1 werden wir die Dynamik verschiedener Objekte quantitativ einordnen.

Das Kontrastverhältnis ist eine *relative* Größe. Ein Verhältnis von 255:1 drückt aus, dass der hellste Punkt 255 mal heller ist als der dunkelste. Welchen *absoluten* Helligkeiten dies entspricht, ist nicht ausgesagt.

Die Anzahl der Bit eines Bildes steht folglich in keinem direkten Zusammenhang zum darstellbaren absoluten Helligkeitsbereich: Mit zwei Bit können wir beispielsweise einen Bereich von 10^4 Candela[6] (siehe auch Tabelle 2.2) abdecken, wenn wir festlegen: $0 \cong 1\text{cd/m}^2$ (Straßenbeleuchtung) und $1 \cong 10^4\text{cd/m}^2$ (natürliches Tageslicht) (vgl. Abbildungen 2.2 und 4.9). Die Anzahl der Bit gibt nur Auskunft darüber, wie viele unterschiedliche Werte gespeichert werden können (s.u.).

[5] Mit 7 Bit können wir 2^7 Werte darstellen, mit 8 Bit $2^8 = 2 \cdot 2^7$ Werte (Linksshift).

[6] „Die Candela ist die Lichtstärke in einer gegebenen Richtung einer Strahlungsquelle, die monochromatische Strahlung der Frequenz 540 * $10^1$2 Hertz aussendet und deren Strahlstärke I_e in dieser Richtung 1/683Wsr^{-1} beträgt.“, 1979 Bureau International des Poids et Mesures, Paris. [*Bergmann und Schaefer*, 2004b]

Bei einem Aufnahmechip stehen die Bit allerdings in direktem Verhältnis zu den einfallenden Photonen und damit zu der absoluten Helligkeit. Der Zusammenhang zwischen ausgegebener Spannung und Helligkeit kann in der Spezifikation des jeweiligen Chips nachgelesen werden (vgl. [*Eastman Kodak Company*, 2006]). Aus der Auflösung des Chips in Bit und diesem Wert kann der abgebildete absolute Dynamikbereich angegeben werden. Die Anzahl der unterscheidbaren abgebildeten Stufen entspricht $2^{\text{Anzahl der Bit}}$.

Die Abbildung der kontinuierlichen Werte auf diskrete Stufen bezeichnet man als **Quantisierung**. Alle Werte, die auf den gleichen diskreten Wert abgebildet sind, entsprechen einer Quantisierungsstufe. Sind diese Stufen gleich groß, so spricht man von *gleichförmiger Quantisierung*. An unser zuvor gewähltes Beispiel angelehnt wäre eine gleichförmige binäre Quantisierung des Intervalls $[0\ \text{cd}/m^2; 10^4\ \text{cd}/m^2]$ folgende Abbildung:

$$Q(l) = \begin{cases} 0, & 0\ \text{cd}/m^2 \leq l < 10^2\ \text{cd}/m^2 \\ 1, & 10^2\ \text{cd}/m^2 \leq l \leq 10^4 \text{cd}/m^2 \end{cases}$$

In Abbildung 2.8 sehen wir zwei Beispiele für ungleichförmige Quantisierung. Die untere Kurve ist fast gleichförmig quantisiert.

Da Computer nur diskrete Werte verarbeiten können, ist eine Quantisierung immer notwendig. Wir werden uns im Laufe der Arbeit an mehreren Stellen mit Quantisierung und den sich daraus ergebenden Effekten befassen (vergleiche Abschnitt 5.2.7.4).

2.2 Motivation und Grundlagen

Ein Foto stellt in vielerlei Hinsicht nicht das dar, was es abbildet. Ein Grund dafür ist der begrenzte Dynamikumfang.

In Grafik 2.1 sehen wir die charakteristische Kurve eines Schwarzweiß-Negativfilms.

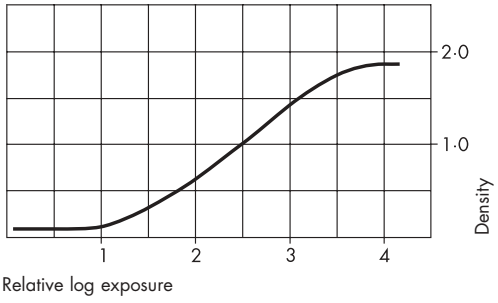


Abbildung 2.1: Die Hurter-Driffield-Kurve eines Schwarzweiß-Negativfilms. Je heller das einfallende Licht („Relative log Exposure“), desto dunkler wird das Negativ („Density“) – entsprechend heller wird das entwickelte Positiv. [*ILFORD Imaging UK*, 2004]

Diese, nach ihren „Erfindern“ auch als Hurter-Driffield- bekannte, Kurve gibt die Beziehung zwischen der auf das Negativ einfallenden Lichtmenge und der daraus resultierenden Dichte wieder. Die *Dichte* bezeichnet hierbei die Konzentration des lichtundurchlässigen Stoffes des Films (beispielsweise Silber).

Wir sehen, dass der Film im mittleren Bereich nahezu logarithmisch linear abbildet. Eine unterscheidbare Schwärzung des Negativs erfolgt zwischen 10^1 und 10^4. Der Dynamikumfang dieses Films ist damit < 1000 bzw. unter 60dB.

Der Dynamikumfang verschiedener Medien und Aufnahmegeräte sowie unserer visuellen Wahrnehmung ist in Tabelle 2.1 eingetragen.

Technik	$\frac{\{\textbf{Blenden}, Bit\}}{Kanal}$	**Dynamik** $_{2^{Blenden}}$	in **dB** $_{20 log Dynamik}$
Schwarzweißpapierpositiv	6	32	30
LDR-Bildformat (digital)	8	256	48
Auge in adaptiertem Zustand [*Ferwerda et al.*, 1996][a]	9	$\approx 10^3$	≈ 60
CCD [*Eastman Kodak Company*, 2006]	12	$5 * 10^3$	74
Schwarzweißfilm[b] (analog) [*ILFORD Imaging UK*, 2002]	<13	$< 10^4$	<80
HDR-CCD [*Kawahito*, 2007, 1865]	25	$< 10^8$	< 160
HDR-Bildformat (digital)	32	$4 * 10^9$	193
Auge insgesamt [*Ferwerda et al.*, 1996]	47	$< 10^{14}$	< 280

[a]Das Auge kann im Bereich $10^{-6} cd/m^2 - 10^8 cd/m^2$ sehen [*Ledda et al.*, 2004b] – allerdings nur ca. 9 Blenden (<1000) gleichzeitig. Um einen anderen Helligkeitsbereich zu sehen, muss es sich erst darauf adaptieren. Je nach Alter geschieht dies schneller oder langsamer.

[b]Ein Farbfilm hat einen geringeren Kontrastumfang als ein Schwarzweißfilm.

Tabelle 2.1: Vergleich des Kontrastumfangs

Die Dynamik, die unser Auge im adaptierten Zustand (siehe Kapitel 4.1.1) wahrnehmen kann, beträgt zwischen 10^3 und 10^4 ($\approx$ 80dB). In dieser Größenordnung liegen auch der eingetragene Schwarzweißfilm und das betrachtete Charge-Coupled-Device (CCD) [*Eastman Kodak Company*, 2006].

Durch Änderung der Adaption (siehe Kapitel 4.1.1) –also nicht gleichzeitig– sind wir allerdings in der Lage, eine Dynamik von bis zu 10^{14} (280dB) wahrzunehmen. Dies ist deutlich mehr, als der Schwarz-Weiß-Film oder der CCD-Chip aufzeichnen können [*Ledda et al.*, 2004b].

Eine Adaptionkann bis zu einem gewissen Grad sehr schnell erfolgen. Bei Szenen, die sowohl sehr dunkle als auch sehr helle Objekte beinhalten, wird dies deutlich: Schauen wir im Sommer aus einem Zimmer heraus ins Freie, so können wir *gleichzeitig* sowohl Gegenstände im Raum als auch draußen sehen. Das Auge adaptiert beständig zwischen Außen und Innen. (Näheres zur visuellen Wahrnehmung in Abschnitt 4.1.1.) Bei einer Fotoaufnahme der Szene werden entweder die äußeren Objekte überbelichtet oder die inneren unterbelichtet sein, also weiß oder schwarz erscheinen: Der Chip kann nicht den kompletten Kontrastumfang auf einmal abbilden[7].

Abbildung 2.2: Vom Menschen wahrgenommene (cd/m^2) Helligkeiten. [*Johnson*, 2004]

Es gibt mehrere Möglichkeiten, mit diesem Problem umzugehen: Beispielsweise kann man durch das gezielte Setzen von Licht den Kontrastumfang der Szene verringern[8], so dass der Sensor alles erfassen kann.

[7]Lieferte er beispielsweise eine Ausgabe von 24Bit RGB, so stünden pro Farbkanal 8 Bit und damit 8 Blenden Dynamik bzw. 256 Helligkeitsabstufungen zur Verfügung.

[8]Der Gesamtkontrast/ die Dynamik wird umso kleiner, je ähnlicher die vorhandenen Helligkeiten –im Beispiel die Lichtverhältnisse von Innen und Außen– werden.

Bei „normalen“ Aufnahmen wird es kaum möglich oder gewollt sein, das Motiv zu manipulieren. Hier muss vielmehr der *Aufnahmeprozess* verändert werden:

Einer der ersten, der sich dieses Problems annahm, war Anfang der 1960er Jahre Charles W. Wyckoff. Sein **Wyckoff-Film** besteht aus drei Filmstreifen [*Wyckoff*, 1962]. Diese reagieren jeweils auf das gleiche Lichtspektrum, sind aber unterschiedlich träge. Je träger ein Film ist, desto langsamer wirkt sich die Belichtung auf ihn aus[9]. Bei einer Belichtung entstehen so gleichzeitig drei Aufnahmen, die jeweils unterschiedlich auf das Licht reagiert haben: Wir erhalten einen dunklen, einen mittleren und einen hellen Ausschnitt der Dynamik.

Zur Darstellung färbte Wyckoff die drei Schwarzweißfilme unterschiedlich ein und legte sie wieder aufeinander. Das Resultat war ein dreifarbiger Film, mit einer stark, einer mittel und einer wenig belichteten Komponente. Auf dem Kompositum waren deutlich mehr Details –beispielsweise einer gefilmten Atombombenexplosion– zu sehen als auf einem einfachen Schwarzweißfilm.

Die Beschränktheit *eines* Aufnahmeprozesses wurde also durch Kombination mehrerer Aufnahmen teilweise umgangen.

Genau dieses Prinzip lässt sich auch auf die digitale Aufnahme anwenden: Es werden verschieden belichtete Aufnahmen einer Szene erstellt, die jeweils einen Ausschnitt des Dynamikbereichs abdecken. Anschließend werden diese so zusammengerechnet, dass ein Bild der Szene mit dem kompletten Kontrastumfang entsteht[10].

Die verschiedenen Aufnahmen können *gleichzeitig* oder *nacheinander* entstehen. Für die simultane Aufnahme wird modifizierte Hardware benötigt. Mit dieser Thematik werden wir uns in diesem Kapitel ausführlich befassen.

2.2.1 Radio-/ Photo-/ Colorimetrische Grundlagen

Wir betrachten sehr kurz einige radio[11]-, photo[12]- und colorimetrische[13] Grundlagen. Da die Details für die weitere Arbeit nicht relevant sind, sei zur ausführlicheren Information auf [*Wyszecki und Stiles*, 1967], Kapitel 4, [*Reinhard et al.*, 2005], Kapitel 2 und [*Encarnação et al.*, 1996], Kapitel 6 verwiesen.

Als sichtbares Licht nehmen wir elektromagnetische Strahlung im Wellenlängenbereich von 360nm (Violett) bis 830nm (Rot) wahr [*Encarnação et al.*, 1996]. Unsere Wahrnehmung ist hierbei für verschiedene Wellenlängen unterschiedlich, wie der Kurve 2.3 zu entnehmen ist.

Die **Radiometrie** befasst sich mit der Strahlung im *gesamten* elektromagnetischen Spektrum. Eine Übersicht über wichtige Kenngrößen und deren Berechnung gibt Tabelle 2.2. Die **Photometrie** beschäftigt sich nur mit der Strahlung im für uns sichtbaren Bereich. Die radio- und photometrischen Größen unterscheiden sich lediglich darin, welcher Wellenlängenbereich wie berücksichtigt wird: Bei der Photometrie wird

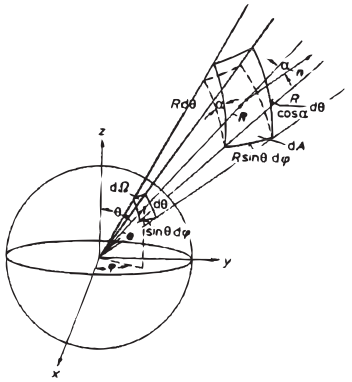
[9] Die Geschwindigkeit eines Flms wird in ISO angegeben. Je höher der ISO-Wert, desto empfindlicher (schneller) reagiert der Film auf Licht (siehe auch Abschnitt 4.1).

[10] Wyckoff hatte noch keine Möglichkeit seine Komponenten derart zusammenzufügen, dass sie ein homogenes Bild ergaben.

[11] Die Radiometrie misst elektromagnetische Strahlen.

[12] Die Photometrie misst sichtbares Licht. Sie beschäftigt sich neben den physikalischen Größen auch mit der Physiologie des menschlichen Auges [*Encarnação et al.*, 1996, 259]

[13] Die Colorimetrie misst Licht in seiner spektralen Verteilung.



Radiometrie (komplettes radiometrisches Spektrum)				**Photometrie** (Strahlung im sichtbaren Bereich)		
Deutsch	*Englisch*	*Einheit*	*Berechungsformel*	*Deutsch*	*Englisch*	*Einheit*
Strahlungsenergie	radiant energy	[J]	$Q = \int \Phi \mathrm{d}t$	Lichtmenge	quantity of light	[lm s]
Strahlungsleistung	radiant power	[W]	$\Phi = \frac{\partial Q_e}{\partial t} = \int_A S \mathrm{d}A = \int_A E \times H \mathrm{d}A$	Lichtstrom	luminous flux	[lm]
Strahlungsstärke	radiant intensity	[W sr^{-1}]	$I = \frac{\partial \Phi_e}{\partial \Omega_1}$	Lichtstärke	luminous intensity	[cd]
Spezifische Ausstrahlung	radiosity	[W m^{-2}]	$M = \frac{\partial \Phi_{e1}}{\partial A_1}$	Spezifische Lichtausstrahlung	luminousity	[lm m^{-2}]
Bestrahlungsstärke	irradiance	[W m^{-2}]	$E = \frac{\partial \Phi_{e2}}{\partial A_2} = I_e \frac{\cos \alpha_2}{R^2} = \int_{2\pi \text{ sr}} L_v \cos \Theta_1 \mathrm{d}\Omega_1$	Beleuchtungsstärke	illuminance	[lx = lmm^{-2}]
Strahldichte	radiance	[W sr^{-1} m^{-2}]	$L = \frac{\partial I_{e1}}{\partial A_1 \cos \Theta_1}$	Leuchtdichte	luminance	[cd m^{-2}]

Tabelle 2.2: Radio- und Photometrische Grundgrößen. Bei den photometrischen Größen werden die von der Wellenlänge des Lichts abhängigen Größen zusätzlich mit der Responsefunktion des Auges gewichtet (siehe Text). Die Einheiten sind: J=Joule, W=Watt, sr=Steradian (Raumwinkel), m=Meter, lm=Lumen (1lm = 1cdsr),cd=Candela, lx=Lux. (Korrigiert und erweitert nach [*Encarnação et al.*, 1996]; [*Bergmann und Schaefer*, 2004b]). Grafik: [*Encarnação et al.*, 1996]

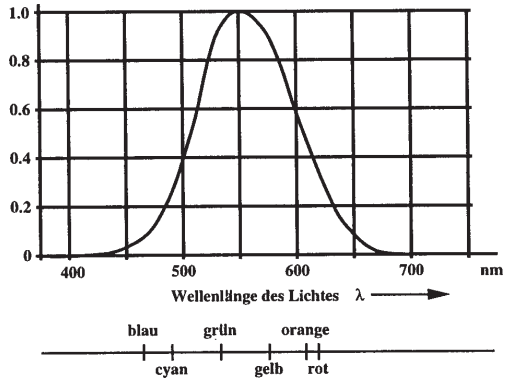


Abbildung 2.3: Die Helligkeitsempfindung des menschlichen Auges bei energiegleicher Strahlung verschiedener Wellenlänge. Wir sehen, dass wir grünes Licht als am intensivsten wahrnehmen. Bei Tag (helladaptiert, photopisches Sehen) sehen wir primär mit den farbempfindlichen Zapfen. Bei Nacht (dunkeladaptiert, skoptisches Sehen) erfolgt die Wahrnehmung hauptsächlich über die Stäbchen (siehe auch Abschnitt 4.1.1). Hier verschiebt die Kurve ihr Maximum auf ca. 510nm nach links. [*Encarnação et al.*, 1996, 276]

nur das für uns sichtbare Licht betrachtet. Dieses fließt nach Gewichtung mit der Empfindlichkeitskurve unseres Auges (siehe Abbildung 2.3) in die in Tabelle 2.2 gegebenen Gleichungen ein.

Die Umrechnung der wellenlängenabhängigen Parameter für die photometrischen Kenngrößen erfolgt nach der folgenden Berechnungsvorschrift [*Encarnação et al.*, 1996]:

$$X_v = K_m \int_{360}^{830} X_{e\lambda}(\lambda) V(\lambda) \mathrm{d}\lambda$$

$X_{v(\text{isible})}$ ist die photometrische Kenngröße. K_m ist ein Proportionalitätsfaktor[14]. $X_{e\lambda}$ ist die radiometrische Kenngröße. V ist die in Grafik 2.3 eingezeichnete Helligkeitsempfindungskurve des Auges.

So ergibt sich beispielsweise die wahrgenommene Helligkeit L_v aus dem gesamten emittierten elektromagnetischen Spektrum $L_e(\lambda)$. Objekte, die selbst keine Lichtquellen sind, können wir nur sehen, da sie Licht reflektieren. Die Art der Reflexion hängt hierbei von der Beschaffenheit[15], der Form, dem Winkel zu der entsprechenden Lichtquelle und ähnlichem ab. Grafik 2.4 verdeutlicht dies.

Die Verteilung des Lichts innerhalb einer Szene lässt sich durch eine Bidirectional Reflectance Distribution Function (BRDF) beschreiben. Mithilfe einer BRDF können wir für jeden Ort der Szene die Helligkeit berechnen. Die Funktion ist die Grundlage für computergenerierte (gerenderte) Szenen. Durch die Verschiedenartigkeit von Lichtquellen und anderen Objekten innerhalb der Szene und deren vielfältige Interaktion ist die Berechnung der Lichtverhältnisse aufwendig[16].

Beleuchtungstechnisch einfache Flächen sind so genannte lambertsche Flächen. Diese reflektieren das gesamte auf ihnen auftreffende Licht gleichmäßig (ideal diffus) in alle Richtungen [*Horn*, 1987, 212][*Encarnação et al.*, 1996, 287]. Solche Flächen eignen sich daher gut zur Kalibrierung, wie wir im Laufe des Kapitels sehen werden.

[14] K_m ist der Maximalwert des fotometrischen Strahlungsäquivalents. $K_m = 683\mathrm{lmW}^{-1}$. [*Bergmann und Schaefer*, 2004b]

[15] Material, Oberflächenstruktur u.ä.

[16] Dies spiegelt sich beispielsweise auch in der nötigen Grafikrechenleistung für heutige grafikintensive Anwendungen wie Computerspiele wider.

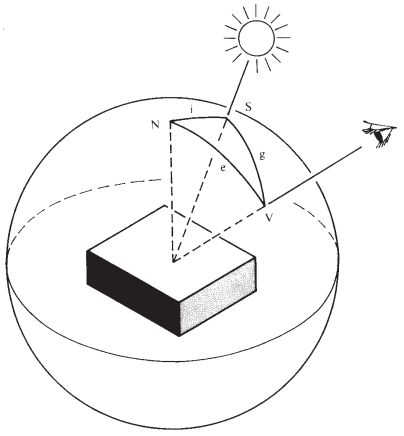


Abbildung 2.4: Bei Flächen, die selbst kein Licht emittieren, nehmen wir die reflektierten Lichtwellen wahr. Wie viel Licht von welchen Lichtquellen in welcher Form beim Betrachter ankommt, lässt sich mittels einer BRDF berechnen. [*Horn*, 1987, 11]

Mit [*Shafique und Shah*, 2004], [*Lin et al.*, 2004], [*Lin und Zhang*, 2005], [*Ng et al.*, 2007] werden wir in Abschnitt 2.3.3 Verfahren zur Responsekurvenbestimmung kennen lernen, die auf Beleuchtungseigenschaften innerhalb einer Szene aufbauen.

Da wir uns mit Farbbildern beschäftigen, ist die **Farbwahrnehmung** ebenfalls relevant für uns. Eines der gebräuchlichsten Farbmodelle ist das RGB-Modell[17]. Es beschreibt Farben als Tripel (R, G, B) aus den Grundbestandteilen Rot, Grün und Blau. Die genauen Wellenlängen für R, G und B sind nicht festgelegt. Daher müssen beispielsweise Monitore für eine „echte“ Farbwiedergabe kalibriert werden. Die Grundfarbreize werden von uns dabei wie in Grafik 2.5 veranschaulicht wahrgenommen.

Der Farbwert, den wir mithilfe der im Auge befindlichen Zapfen wahrnehmen (vgl. Abbildung 2.6; siehe Abschnitt 4.1.1) ergibt sich additiv aus den gegebenen drei Grundvalenzen (R, G, B)[18]:

$$Q_\lambda = \bar{r}(\lambda)R + \bar{g}(\lambda)G + \bar{b}(\lambda)B$$

Wie unsere Helligkeitswahrnehmung ist auch unsere Farbwahrnehmung nicht absolut sondern relativ. Abbildung 2.7 veranschaulicht dies.

Wenn wir die Farbe eines Objektes wahrgenommen haben und sich danach die Beleuchtung ändert, so nehmen wir das Objekt weiterhin in derselben Farbe wahr. Unsere Farbwahrnehmung weist also eine gewisse Konstanz und Invarianz gegenüber Beleuchtungsänderungen auf. Für elektronische Darstellungs- und Abbildungsprozesse stellt dies ein Problem dar, da sie nur absolute Werte darstellen bzw. messen können.

Die Farbe ergibt sich beim RGB-Modell über die Relation der Farbkanäle zueinander. Diese Relation wird über den so genannten **Weißabgleich** festgelegt. Bei diesem Vorgang wird bestimmt, welche Messwerte als weiß bzw. farblos gespeichert werden sollen. Daraus ergeben sich dann die Farbwerte aller gemessenen Kombinationen. Die einzelnen Farbmessungen werden beim Weißabgleich möglichst so in Relation gesetzt, wie sie uns erschienen, wenn wir die Szene direkt wahrnehmen. Stimmt der Weißabgleich nicht, so bekommt das Bild einen Farbstich, die Farben werden also „falsch“

[17] Die verschiedenen Modelle haben jeweils andere Grundkurven. Das Standardgrundmodell ist das CIE1931-2-degree XYZ Farbmodell [*Reinhard et al.*, 2005, 33].

[18] Die Grundvalenzen sind dabei nicht genau festgelegt. Insbesondere sind die drei Stimuli, auf die die Zapfen im Auge reagieren (siehe Abbildung 2.6) nicht direkt (getrennt) darstellbar (vgl. Abbildungen 2.5 und 2.6).

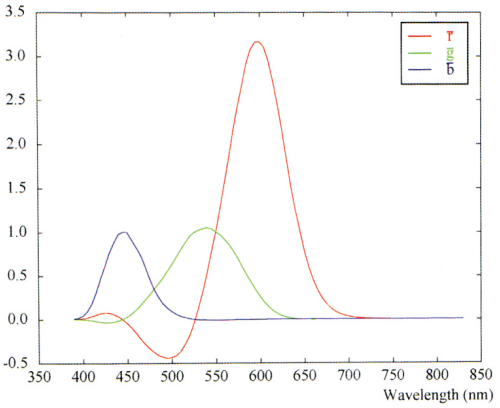


Abbildung 2.5: CIE1964 Standard Observer. Die Wahrnehmung der drei Primärvalenzen $\bar{r} = 645,2nm$, $\bar{g} = 525,3nm$ und $\bar{b} = 444,4nm$. Die y-Achse entspricht der Reizstärke. Bei dem zugrunde liegenden Experiment füllten die Lichtreize jeweils 10° des Sichtfeldes der Beobachter aus. [*Reinhard et al.*, 2005, 29]

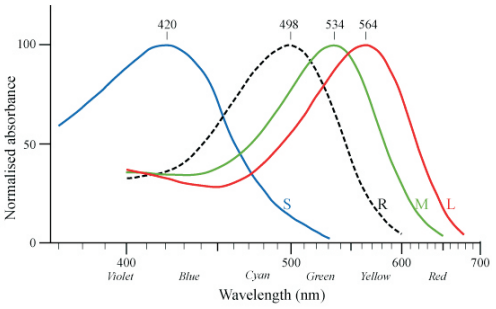


Abbildung 2.6: Die Abbildung zeigt die spektrale Empfindlichkeit der Stäbchen (R) und Zapfen im kurzwelligen (S) Blau-, im mittelwelligen (M) Grün- und im langwelligen (L) Rotbereich des Lichtes, jeweils relativ auf die maximale Empfindlichkeit der Sehzelle bezogen. Wir sehen, dass diese nicht den Primärvalenzen des RGB-Modells entsprechen. [de.wikipedia.org]

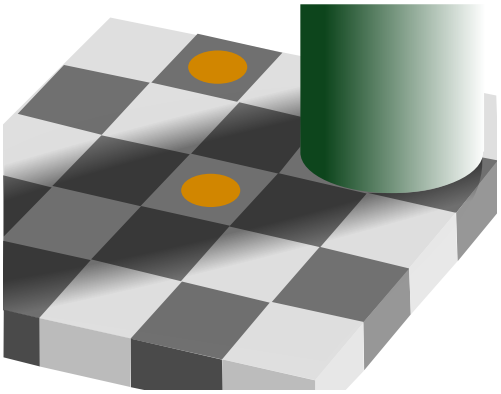


Abbildung 2.7: Der „orangefarbige“ und der „braune“ Kreis haben exakt dieselbe Farbe. Ebenso haben die sie umgebenden Felder jeweils dieselbe Farbe. Dennoch werden sie von uns nicht als identisch wahrgenommen. Es handelt sich hierbei um eine so genannte Mondriansche Illusion.

dargestellt. Achromatische (graue) Flächen erhalten beispielsweise einen Farbstich, werden also farbig. Dies liegt gerade an der nichtlinearen Responsekurve der Kamera: Die (Korrektur-)Verschiebung einzelner Kanäle ändert das Verhältnis der Kurven der Farbkanäle zueinander und damit die Farbwerte.

Wenn wir eine HDR-Aufnahme vorliegen haben, so handelt es sich um Werte, die unabhängig von der Kameraresponsekurve sind. In diesem Fall kann der Weißabgleich durchgeführt werden, indem eine als neutral (grau) darzustellende Fläche innerhalb des Bildes identifiziert wird und die Farbwerte derart linear zueinander verschoben werden, dass die Fläche als achromatisch gespeichert wird [*Reinhard et al.*, 2005, Kapitel 2].

Die **Gammakorrektur** hat ebenfalls Auswirkung auf die gespeicherten Bildwerte (siehe auch [*Encarnação et al.*, 1996, 29f], [*Reinhard et al.*, 2005, 73ff]). Die Gammakorrektur entspricht mathematisch einer Potenzierung des Messwertes mit γ: $f(x) = x^{\gamma}$. Sie führt zu einer nichtlinearen Anhebung des Signals (und damit der Helligkeit). Die obere blaue Kurve in Abbildung 2.8 zeigt die ungefähre Form der Kurve zu $\gamma = 2,2$.

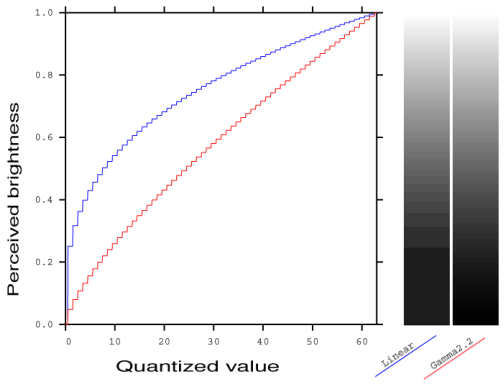


Abbildung 2.8: Unsere Helligkeitsempfindlichkeit entspricht ungefähr einer Gammakurve – in den Bereichen der größten Steigung nehmen wir Helligkeitsunterschiede am genauesten wahr. Daher erscheinen uns die Intervalle der nach erfolgter Gammakorrektur quantisierten absoluten Helligkeitswerte („Quantized value“) ungefähr gleichgroß (untere rote Kurve). Diejenigen bei direkter uniformer Quantisierung der linearen Sensorausgabe dagegen erscheinen uns ungleich (obere blaue Kurve). Zur Veranschaulichung wurde hier nur mit sechs Bit quantisiert, um größere Stufen zu erhalten. Der Effekt ist bei acht Bit derselbe. [*Reinhard et al.*, 2005, 74]

Die Gammakorrektur wurde zu Beginn der ersten Bildübertragungen für das Fernsehen[19] eingeführt, nachdem man festgestellt hatte, dass die Kathodenstrahlröhren der Fernsehgeräte genau invers der dargestellten oberen blauen Kurve auf Spannungen reagieren. Die Überlegung war damals, dass es deutlich mehr Fernsehgeräte als Aufnahmekameras geben werde und daher eine Signalanpassung innerhalb der Kameras günstiger sei [*Mann*, 2000]. Mann schreibt in [*Mann*, 2000], dass es ein glücklicher Zufall sei, dass die menschliche Helligkeitsempfindlichkeit ungefähr der Gammakurve entspricht.

Abbildung 2.8 verdeutlicht das Problem: Mit der Einführung der digitalen Signalverarbeitung mussten die Werte quantisiert werden. Hätte man eine dem menschlichen

[19] Die ersten regulären Fernsehübertragungen gab es 1936 zu den olympischen Spielen in München. Verbreitet hat sich das Medium aber erst nach dem 2. Weltkrieg. 1953 gab es beipielsweise 7000 Fernsehempfänger in Deutschland. [Vorlesung Nachrichtensprache im Wandel, Manfred Muckenhaupt, 2000]

Sehen nicht entsprechende Funktion gewählt, so wäre der Quantisierungsfehler bei der uniformen Quantisierung in Bezug auf die menschliche Wahrnehmung groß –und damit störend sichtbar– gewesen. Durch die Gammakorrektur wird die Auflösung in den Bereichen, in denen wir die feinsten Helligkeitsunterschiede wahrnehmen können erhöht. So verteilt sich der wahrgenommene Fehler mit Gammakorrektur gleichmäßig über die Quantisierungsintervalle, sodass wir ungefähr gleichgroße „Wahrnehmungsintervalle“ erhalten.

Da wir die absoluten Originalhelligkeiten zurückrechnen wollen, stellt die ungleiche –an der menschlichen Wahrnehmung orientierte– Quantisierung für deren Rückgewinnung ein Problem dar. Die obere blaue Kurve verdeutlicht dies: In einigen Bereichen ist die Auflösung ungenauer als in anderen, feiner quantisierten. [*Reinhard et al.*, 2005]

2.2.2 Bildaufnahme mithilfe eines CCD-/ CMOS-Sensors

Die digitale Aufnahme erfolgt heutzutage (2008) mit Charge-Coupled-Devices (CCD) oder CMOS[20]-(Active-Pixel-)Sensoren.

Die heute als **CCD** bekannten Schaltungen wurden 1968 bereits als „Bucket-Brigade Electronics“ [*Sangster und Teer*, 1969] an den Philips Eindhoven Labs erfunden (siehe Abbildung 2.9). 1969 tätigten Willard Boyle und George Smith unabhängig davon an

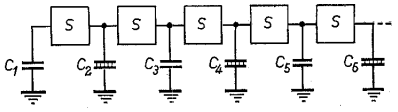


Abbildung 2.9: Die Bucket-Brigade, ein analoges Schieberegister. [*Sangster und Teer*, 1969]

den Bell Laboratorien ebenfalls die Erfindung [*Boyle und Smith*, 1993; *Boyle und Smith*, 1976]. In beiden Fällen wurde primär nach einem Speichermedium geforscht. Die Lichtempfindlichkeit stellte sich zufällig heraus, wenngleich in den Jahren zuvor schon an ähnlichen Schaltungen geforscht wurde, um Lichteinfall zu messen [*Fossum*, 1997].
In einem CCD sind einzelne Speicherzellen jeweils mit ihrem Nachbarn verbunden. Sie können ihre Ladungen –wie bei einer Eimerkette (Bucket-Brigade)– an ihren Nachbarn weiterreichen. Der Auslesevorgang erfolgt sequenziell, indem immer eine Zeile/ Spalte aus dem Speicherverbund „herausgeschoben“ wird. Die Anwendung als Fotosensor ergab sich zufällig aus der Lichtempfindlichkeit der einzelnen Speicherzellen [*Eastman Kodak Company*, 2001].

Die Lichtempfindlichkeit der einzelnen Pixel beruht auf dem inneren Fotoeffekt. Bei diesem werden mithilfe von Photonen –also Licht– Elektronen vom Valenzband in das Leitungsband gehoben. Es entstehen Elektronen-Loch-Paare. Mithilfe einer anliegenden positiven Spannung bleibt die durch das Licht hervorgerufene Ladungstrennung bis zum Auslesevorgang erhalten. *Die Ladung des Pixels ist proportional zum eingefallenen Licht*[21] [*Theuwissen*, 1995; *Bergmann und Schaefer*, 2004a, 2005].

CMOS-Sensoren unterscheiden sich von CCD-Sensoren durch den wahlfreien Zugriff auf die Speicherzellen und den geringeren Stromverbrauch. Zusätzlich lassen sich auf

[20] Complementary metal-oxide-semiconductor
[21] Die Ladung ist proportional zur Anzahl der eingefallenen Photonen.

CMOS-Sensoren Komponenten zur weiteren Signalverarbeitung (Verstärkung, Buffering, A/D-Wandlung) direkt in die Pixel integrieren. Man spricht dann von **Active-Pixel-Sensoren** (APS). Durch die höhere Integration können CMOS-Sensoren kostengünstiger als CCDs hergestellt werden. Der Preis dafür ist unter anderem ein höheres Sensorrauschen [*Tian et al.*, 2001]. Weiterführende Informationen zur Entwicklung von CCD zu CMOS und dessen Unterarten finden sich in [*Fossum*, 1997].

Viele Jahre wurde vor allem im Bereich der CCDs geforscht. CMOS-Sensoren waren aufgrund des deutlich höheren Rauschens nicht attraktiv. Erst seit 1990 wurde vermehrt in die Weiterentwicklung der CMOS-Sensoren investiert. Diese sind aufgrund ihrer Kosten und wegen ihrer höheren Miniaturisierungsfähigkeit interessant. *Fossum* [1997] schreiben, dass CMOS- und CCD-Kameras in Bezug auf Rauschen, Quantenausbeute und Kontrastumfang mittlerweile vergleichbar seien.

Für unsere weitere Verarbeitung der Helligkeitswerte ist deren Genauigkeit entscheidend. Für die in 2.3 vorgestellten Verfahren zur Gewinnung der Kameraresponsekurve ist relevant, dass die Helligkeitswerte Fehler enthalten können. Rauschquellen betrachten wir in 2.2.2.1 und 2.2.2.2.

Wie wir später sehen werden, stehen die von der Kamera gelieferten Werte durch die Nachverarbeitung On- oder Off-Chip (CMOS/ CCD) auch ohne Fehler heute in keinem linearen Verhältnis zur einfallenden Lichtmenge mehr (vgl. 2.2.5 und 2.3, Madden).

2.2.2.1 Aufnahmefehler des digitalen Films

Bei der elektronischen Datenverarbeitung kommt es fast immer zu Signalbeeinträchtigungen, die sich in Form von Rauschen im Signal bemerkbar machen. Auch der analoge Film kennt Signalverfälschungen. Diese äußern sich beispielsweise in grober Körnung.

Es gibt diverse Arten von Rauschen: Fixed Pattern Noise, Dunkelspannung, thermisches Rauschen, Blooming, Aufnahmerauschen (shot noise), Rücksetzrauschen (reset noise), Ladungstransferrauschen, Verstärkerrauschen, A/D-Wandlerrauschen, Quantisierungsfehler... [*Healey und Kondepudy*, 1994; *Liu und El Gamal*, 2003; *Theuwissen*, 1995]

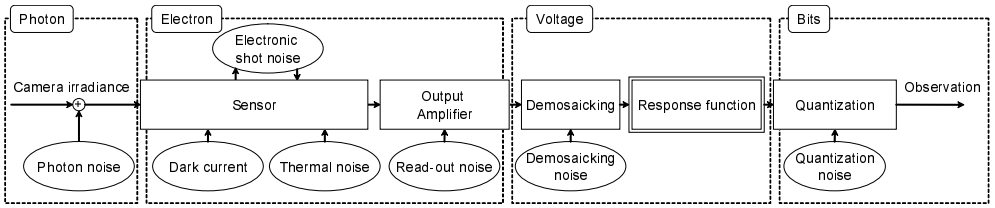


Abbildung 2.10: Die Grafik zeigt einige Rauschquellen innerhalb der Verarbeitungspipeline [*Matsushita und Lin*, 2007]. (Demosaicking bezeichnet hierbei den Prozess, der über das Bayer-Pattern –also die verschieden farbgefilterten Helligkeitspixel– interpoliert.)

Fixed Pattern Noise beschreibt eine örtlich fest auf dem Chip vorhandene Grundstörung. Diese entsteht dadurch, dass herstellungsbedingt nicht alle Pixel gleich sind. Eine konstante Helligkeit über dem Chip wird also nicht in allen Pixeln denselben Wert ergeben.

Die **Dunkelspannung** gibt den Wert an, den ein Pixel bei völliger Dunkelheit liefert. Sie hängt eng mit dem thermischen Rauschen zusammen:
Nicht nur Photonen können Elektronen aus dem Valenzband in das Leiterband heben sondern auch Wärme. Diese durch Wärme entstehende Störung nennt man **thermisches Rauschen**. Um es möglichst gering zu halten, kühlen beispielsweise Astronomen für ihre Langzeitaufnahmen die Sensoren auf -120° herunter.

Als **Blooming** wird ein Fehler bezeichnet, der auftreten kann, wenn besonders viele Photonen auf ein Pixel treffen. Es werden so viele Ladungen erzeugt, dass diese auf die umliegenden Pixel überfließen und somit als Ausblühungen (Blooming) im Bild erscheinen. Dieser Effekt tritt bei Webcams häufig auf. Die HDR-Erstellung entfernt das Blooming, da es nicht auf allen Aufnahmen vorhanden ist und die überbelichteten Pixel aussortiert werden. Für die Kalibrierung ist es wichtig, Pixel, bei denen Bloomingeffekte auftreten, auszuschließen. *Tsin et al.* [2001] schreiben, dass es sich in der Praxis bewährt habe, einen Bereich von drei Pixeln um die übersättigte Stelle auszuschließen.

Aufnahmerauschen (shot noise) entsteht während der Aufnahme. Es liegt in der Quanteneigenschaft des Lichts begründet, die eine genaue Aufenthaltsvorhersage unmöglich macht. Somit können bei gleichen Lichtverhältnissen zu unterschiedlichen Integrationszeitpunkten unterschiedliche Werte an einem Pixel gemessen werden.

Rücksetzrauschen (reset noise) entsteht, wenn die Pixel nach der Aufnahme nicht vollständig zu Null zurückgesetzt werden.

Ladungstransferrauschen tritt vor allem bei CCD-Chips auf und äußert sich beispielsweise im so genannten Smearing. Darunter versteht man die Fahnen, die helle Punkte im Bild nach sich ziehen. Sie entstehen dadurch, dass ein Teil der Ladung eines sehr hellen Pixels beim Weiterreichen zur nächsten CCD-Zelle in der alten überbleibt und sich anschließend zu nachfolgenden Ladungen addiert. Je nachdem, ob das CCD längs oder quer ausgelesen wird, können so vertikale oder horizontale Striche entstehen.

Das **Verstärkerrauschen** entsteht bei der Verstärkung der Pixelwerte.

Ein Teil des **A/D-Wandlerrauschen**s, das bei der Wandlung der analogen Werte in diskrete Werte entsteht, ist **Quantisierungsrauschen**. Quantisierungsrauschen kann zusätzlich bei der Kompression der Bilddaten, beispielsweise in das JPG-Format, entstehen. Bei diesem Vorgang entsteht allgemein **Komprimierungsrauschen**.

Die Aufzählung ist nicht erschöpfend, je nach individueller Signalverarbeitung können Rauschquellen hinzukommen oder wegfallen. Die Lage einiger Rauschquellen in der Signalverarbeitungskette ist aus Grafik 2.10 ersichtlich.

2.2.2.2 Aberrationen im Linsensystem – Teil 1

Abbildungsfehler entstehen nicht nur auf dem lichtempfindlichen Element der Kamera. Auch das vorgelagerte Linsensystem führt zu Verfälschungen der abgebildeten Szene, so genannten Aberrationen[22]. Wir wollen uns an dieser Stelle nur mit den für dieses Kapitel relevanten Fehlern beschäftigen. Weitere Aberrationen finden sich in der Fortsetzung dieses Kapitels in Abschnitt 3.2.1. An dieser Stelle sind für uns Helligkeits- und Farbfehler interessant.

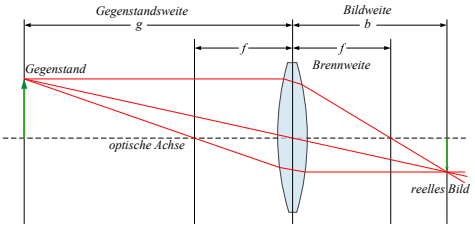


Abbildung 2.11: Eine Linse mit den entsprechenden Bezeichnungen.

Das erste Phänomen, das wir betrachten, ist der **natürliche Randlichtabfall** ($\cos^4$-Gesetz). Über eine einfache Linse aufgenommene Bilder werden zum Rand hin dunkler. Dieser Effekt ist besonders bei homogenen Flächen auffällig. Er ist vergleichbar mit der Sonneneinstrahlung auf die Erde im Winter: Obwohl die Strahlungsintensität der Sonne nicht merklich abnimmt, ist der Auftreffwinkel der Strahlen stumpfer und daher treffen bei ungefähr gleicher Leuchtstärke weniger Strahlen pro Fläche auf. So verhält es sich auch bei der Linse: Am Rand werden weniger Strahlen pro Fläche gebrochen und daher treffen weniger Photonen bei gleicher Helligkeit auf den Film. Die Abnahme der Bestrahlungsstärke erfolgt mit $\cos^4 \Theta$:

$$E = L\pi(\frac{l}{z})^2 \cos^4 \Theta$$

L ist die Leuchtdichte des Objektes, Θ der eingeschlossene Winkel zur optischen Achse, l ist der Radius der Linse, z der Abstand zwischen Linse und Film.

Asada et al. [1996] haben experimentell eine Schwächung der Leuchtdichte von kleiner 5% zwischen optischer Achse und Rand der Linse gemessen. *Aggarwal et al.* [2001] kommen auf bis zu 31% bei einer Lichtquelle mit 10° [23] zur optischen Achse.

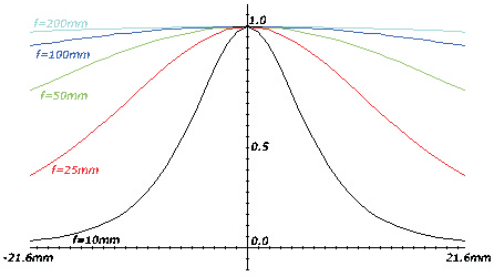


Abbildung 2.12: Darstellung des Helligkeitsabfalls über die Diagonale des Kleinbildformates (43.2mm). Die Kurven zeigen den Helligkeitsabfall für verschiedene Brennweiten gemäss des $\cos^4$-Gesetzes in Abhängigkeit der Entfernung zum Bildzentrum. [Deutsche Wikipedia]

Asada et al. [1996] schreiben, dass die $\cos^4 \Theta$-Abschattung durch entsprechendes Linsendesign kompensiert werden könne und damit vernachlässigbar sei. Dabei werden

[22] Von lateinisch ab-errare (fortirren; die Lichtstrahlen entfernen sich vom richtigen Weg).
[23] Elevation bezeichnet hier den Winkel zur optischen Achse.

die Linsen so konstruiert, dass je größer der Winkel Θ ist, desto mehr Licht von der vordersten Linse in den Strahlengang gebrochen wird. Das Verfahren findet heute bei vielen Linsen Anwendung. Dadurch können neue Probleme entstehen. Zum Beispiel können sich Lage und Form einer Lichtquelle je nach Eintrittspunkt in das Objektiv ändern [*Aggarwal et al.*, 2001].

Vignettierung führt ebenfalls zu einer Abschattung und tritt bei längerem Objektivgang auf, wenn die mit einer ersten Linse aufgefangenen Strahlen an einer zweiten Linse vorbeiprojiziert werden (auf die Objektivwand). Treffen also auf Linse A 10 Lichtstrahlen auf und davon werden 5 an Linse B vorbeigebrochen, so erreichen den Film schließlich nur 5 Strahlen und dieser zeichnet fälschlicherweise nur die halbe Leuchtdichte auf (vgl. Grafik 2.13).

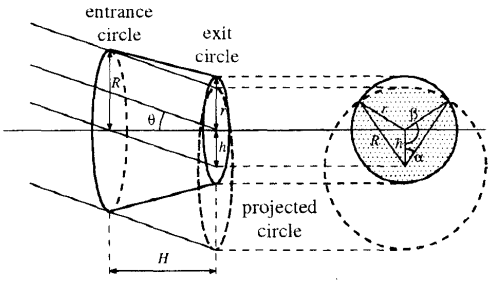


Abbildung 2.13: Das Bild veranschaulicht die Vignettierung: Die erste Linse projiziert nicht das gesamte auf sie auftreffende Licht auf die zweite Linse sondern an ihr vorbei. Dadurch fällt für die Randregionen weniger Licht auf Linse zwei – der Rand erscheint dunkler. [*Asada et al.*, 1996]

Vignettierung tritt bei kleiner Blendenzahl, also großer Blendenöffnung, auf. Da die Blende sich hinter der ersten Linse befindet, treffen bei geschlossener Blende alle Strahlen, die Linse A und die Blende passiert haben, auf Linse B auf – Linse A ist größer als der von der Blende nicht abgeschattete Bereich von Linse B [*Asada et al.*, 1996; *Aggarwal et al.*, 2001].

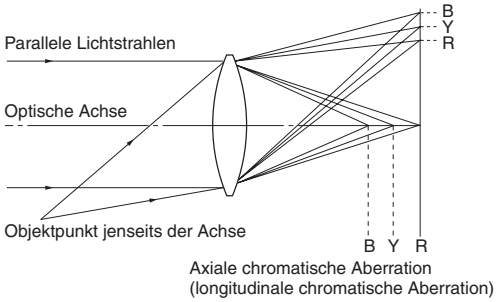


Abbildung 2.14: Die Grafik veranschaulicht die chromatische Aberration einer konvexen Linse. [*Canon Inc.*, 2006]

Neben der Helligkeit, kann auch die Farbe durch die Linse verändert werden. Diese, so genannte **chromatische Aberration**, tritt auf, weil die Lichtanteile verschiedener Wellenlängen jeweils anders gebrochen werden. Die Aberration betrifft die Brennweite und damit den Fokus auf der Bildebene sowie den Vergrößerungsfaktor. Es kann daher geschehen, dass die Farbkomponenten auf der Bildebene nicht zur Deckung kommen. Es entstehen beispielsweise farbige Schatten um Objekte. Wir kennen diesen Effekt von einem Prisma. [*Britannica*, 2008]

Ein weiterer für dieses Kapitel relevanter Effekt ist die Veränderung der Tiefenschärfe bei Änderung der Blende (siehe Grafik 2.15). Dieses Verhalten ist keine Aberration. Wie wir im Laufe des Kapitels sehen werden, brauchen wir zur Gewinnung eines HDR unter verschiedenen Helligkeiten aufgenommene LDR-Bilder. Die Regulierung der Helligkeit, die der Sensor misst, kann dabei über die Belichtungszeit oder die Blende geschehen. Eine Regulierung über die Blende bietet sich aus genanntem Grund nicht an. Die Bilder lassen sich anschließend nicht konsistent zusammenrechnen, da sie alle eine unterschiedliche Tiefenschärfe aufweisen.

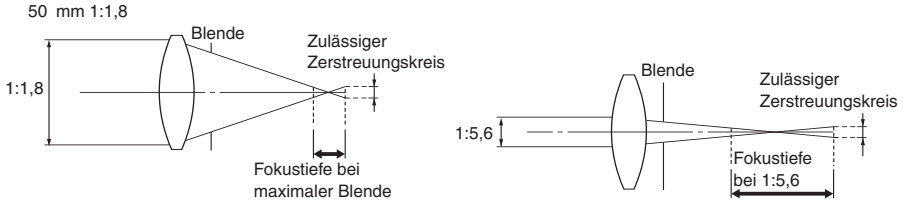


Abbildung 2.15: Die Grafik veranschaulicht die Veränderung der Tiefenschärfe durch die Veränderung der Blende. Eine so gewonnene Belichtungssequenz lässt sich nur schwer konsistent zusammenrechnen. [*Canon Inc.*, 2006]

2.2.3 HDRI mit veränderter Hardware

Wir haben einleitend die Dynamik definiert:

$$DR = 20 \log \frac{I_{max}}{I_{min}} [dB] \tag{2.1}$$

Sie ist definiert als der Quotient des nicht gesättigten Maximal- und Minimalwertes der Helligkeit. Der Maximalwert ist dabei der Wert, bei dem die Sensorelemente voll gesättigt sind, der Minimalwert entspricht in der Regel dem Grundrauschen des Chips [*Nayar und Mitsunaga*, 2000].

Ist der Dynamikumfang des Sensors geringer als die Szenendynamik, so übersättigt ein Teil der Sensorelemente. Dies führt dazu, dass die betroffenen Pixel maximale oder minimale Helligkeit als Messwert ausgeben. In den entsprechenden Bereichen des Bildes sind keinerlei Details mehr zu erkennen. Mit heutigen Sensoren ist eine Messung des gesamten Helligkeitsspektrums oft nicht auf einmal möglich (vgl. Tabelle 2.1).

Wie wir Tabelle 2.1 entnehmen können, erreichen spezielle (Forschungs-)sensoren im Jahr 2007 einen Kontrastumfang von bis zu 153dB [*Kawahito*, 2007]. Ein mögliches Verfahren dazu wird beispielsweise in [*Acosta-Serafini et al.*, 2004] präsentiert: Hier wird der Auslesevorgang des CCD-Elements derart modifiziert, dass die einzelnen Fotoelemente periodisch ausgelesen und nur bei Sättigung zurückgesetzt werden[24]. So erreicht man, dass für jedes Pixel des CCD ein sehr guter Wert kurz vor der Sättigung vorliegt. *Brajovic und Kanade* [1996] präsentieren ein CCD, das nicht die Lichtmenge in einer bestimmten Zeit sondern die Zeit bis zur Sättigung des Sensors aufzeichnet. In beiden Fällen ist die Belichtungszeit bis das fertige Ergebnis vorliegt die Zeitspanne, bis das letzte Pixel gesättigt ist. Dadurch sind diese Verfahren anfällig für Bewegungsunschärfe.

[24] Die Sensoren integrieren also permanent über die einfallenden Photonen. Sobald sie gesättigt sind, werden sie zurückgesetzt.

Nayar und Mitsunaga stellen in [*Nayar und Mitsunaga*, 2000] Hybridverfahren[25] vor, die keine Veränderung an der Arbeitsweise eines einzelnen Sensorelements erfordern: Eine Möglichkeit ist die Verwendung eines Prismas, das das Bild auf mehrere verschieden empfindliche Sensoren lenkt.
Ein anderes Verfahren ist die Kombination verschieden großer Detektoren auf einem Chip. Dadurch werden bei der Aufnahme einer homogen beleuchteten Fläche in manchen Pixeln mehr, in manchen weniger Photonen aufgefangen[26]. Wenn also ein größerer Sensor beim Auslesen gesättigt ist, ist es der benachbarte kleinere unter Umständen noch nicht. Durch Kombination niederdynamischer Sensoren kann so eine höherdynamische Messung erfolgen. Anstelle der verschieden großen Sensorelemente kann man auch deren Integrationszeit variieren[27].

Keine Änderung an den Fotozellen erfordert das von den Autoren vorgestellte Verfahren der „spatially varying pixel exposures“: Sie legen eine –nach einem alternierenden Muster abschattende– Folie auf die Sensoren[28]. So erhalten sie mit einer Aufnahme mehrere verschieden belichtete Teilbilder[29]. Ein großer Vorteil ist die feste Belichtungszeit. Nachteilig ist die verringerte Auflösung der Teilbilder[30]. *Nayar und Mitsunaga* [2000] präsentieren ein Interpolationsverfahren, um aus den geringer aufgelösten Teilbildern ein Bild mit voller Sensorauflösung und erhöhter Dynamik zu erhalten[31]. Bei den von ihnen zur Verfügung gestellten Testbildern erreichen sie deutlich bessere Ergebnisse bei gleicher (voller) Auflösung[32] als bei der Aufnahme der Szene mit einem Standard-CCD ohne das vorgestellte Filter-/ Interpolationsverfahren.

In [*Aggarwal und Ahuja*, 2001] wird ein Verfahren zur Gewinnung von HDR-Panoramabildern vorgestellt. Die Autoren arbeiten ebenfalls mit einem Standard-CCD und einer Maske. Zur Aufnahme des Panoramas rotieren sie eine kontinuierlich aufnehmende Kamera. Während der Aufnahme schatten sie die Linse mit einer ebenfalls rotierenden Maske unterschiedlich ab. Nach einer 360°-Umdrehung um die Kameraachse stehen so mehrere Bilder mit voller Auflösung und unterschiedlicher Belichtung zur Verfügung, die zu einem HDR-Panorama kombiniert werden können[33].

Die von den gerade vorgestellten Verfahren benötigte Kombination von verschieden belichteten Teilbildern zu einem HDR-Bild erfolgt in der Regel über einen gewichteten Mittelwert. Näheres hierzu sehen wir in den Abschnitten 2.2.5 und 2.3.

[25] Als hybrid seien hier Verfahren bezeichnet, die einen Standardaufnahmeprozess mit einer Nachverarbeitung kombinieren, um so HDR-Daten zu erhalten.

[26] Da die Sensoren unterschiedlich groß sind, treffen auf größere Sensoren in der gleichen Zeit mehr Photonen auf als auf kleinere.

[27] Dadurch wird das Verfahren den zuvor vorgestellten ähnlicher mit dem Unterschied allerdings, dass die Belichtungszeit ein festes Maximum nicht überschreitet.

[28] Die Autoren verwenden ein Muster, das jeweils vier benachbarte Pixel immer stärker abschattet ($e_0 < e_1 < e_2 < e_3$) und sich periodisch über den Chip wiederholt.

[29] Alle Pixel der gleichen Abschattung ergeben ein Teilbild. Bei einem Abschattungsmuster *abcabcabc*... erhalten wir beispielsweise drei Teilbilder A, B, C.

[30] Bei drei Abschattungsstufen ergeben sich beispielsweise drei nicht deckungsgleiche Teilbilder mit jeweils $\frac{1}{3}$ Pixeln.

[31] Übersättigte oder verrauschte Pixel können nicht verwendet werden.

[32] Durch die Interpolation wird das Bild auf die volle Auflösung hochgerechnet. Ein ähnlicher Ansatz wird mit dem Bayer-Pattern für die Farbaufzeichnung verwendet [*Bayer*, 1976].

[33] Diese Arbeit unterscheidet sich von unserer durch die Hardwaremodifikationen: Wir wollen weder eine drehbare Kamera, die Rundumsicht benötigt, noch eine abschattende Maske verwenden.

2.2.4 DRI mit Software

Einen Schritt in Richtung HDRI nur mit Softwareunterstützung stellt **Dynamic Range Increase (DRI)** dar. Damit bezeichnet man Verfahren, welche die jeweils gut belichteten Teile mehrerer LDR-Bilder direkt kombinieren. Das Ergebnis ist ein LDR, das mehr Dynamik abbildet, als es mit einer einzelnen Belichtung möglich gewesen wäre.

Bei DRI werden die über- oder unterbelichteten Teile einer Belichtung durch die entsprechenden Bereiche einer anderen Belichtung ersetzt. Ein automatisiertes Verfahren dazu wird zum Beispiel in [*Mann*, 2000] beschrieben. Zum nahtlosen Blenden der verschiedenen Bereiche kann man beispielsweise das Multi-Band-Blending verwenden (siehe Abschnitt 3.6).

Das Ergebnis kann dem eines tonegemappten HDRIs entsprechen. Wie wir in Kapitel 4 sehen werden, wird beim Tonemapping einiger Aufwand betrieben, um einen natürlichen Bildeindruck zu erreichen. Das hier skizzierte Vorgehen kombiniert lediglich gut belichtete Regionen verschiedener Aufnahmen. Dabei wird nicht berücksichtigt, welche Belichtung die einzelnen Teilregionen haben. Es kann dadurch leicht geschehen, dass das entstehende Bild sehr unnatürlich wirkt.

Mit DRI erreichen wir unser Ziel, das heißt eine möglichst optimal belichtete LDR-Aufnahme, in der Regel nicht. Außerdem erhalten wir keine HDR-Aufnahme der Szene. Daher beschäftigen wir uns als Nächstes mit „echtem“ HDRI durch Software und mappen das Ergebnis anschließend zu einem LDRI.

2.2.5 HDRI mit Software

Ohne Veränderungen an den Aufnahmechips kommt Software zur Erstellung von HDRIs aus[34]. Sie errechnet aus mehreren LDRIs ein HDR-Bild. Dazu werden mehrere LDR-Aufnahmen der Szene benötigt, die zusammen den kompletten Kontrastumfang abbilden. Ziel ist es, die in den einzelnen Bildern vorhandenen Werte auf die Originalhelligkeiten zurückzurechnen und diese als ein HDRI zu speichern, das den gesamten Dynamikumfang der abgebildeten Realität umfasst.

Die aus einem CCD gelesenen Werte sind *linear* zur einfallenden Lichtmenge [*Eastman Kodak Company*, 2006; *Theuwissen*, 1995, 7]: Ein doppelt so hoher Pixelwert bedeutet doppelten Lichteinfall und damit doppelte Helligkeit an der entsprechenden Stelle.
Die menschliche visuelle Wahrnehmung reagiert *nicht linear* auf Helligkeiten [*Ledda et al.*, 2004b, 153] sondern logarithmisch [*Poynton*, 1998]. Auch analoge Filme reagieren nicht linear auf die einfallende Lichtmenge [*ILFORD Imaging UK*, 2004, 4]. Dadurch sind sie in der Lage, eine größere Dynamik auf ihren begrenzten Kontrastumfang abzubilden. Sie ahmen damit unsere Wahrnehmung nach und erreichen so einen realistischeren Bildeindruck.

Für eine „natürlichere“ Wahrnehmung werden auch die Daten eines CCD nachverarbeitet (vgl. Anfang 2.3). Die von einer Kamera gelieferten Pixelwerte stehen daher zumeist in keinem linearen Zusammenhang zur gemessenen Helligkeit. Da wir für das

[34] Die vorgestellten Verfahren können mit modifizierter Hardware kombiniert werden. Das im Abschnitt 2.2.3 vorgestellte Verfahren mit dem Prisma könnte beispielsweise simultan mehrere Eingabebilder für einen der hier vorgestellten Algorithmen liefern.

HDRI jedoch die Originalhelligkeiten suchen, müssen wir die **Responsekurve**[35] des Aufnahmeprozesses ($f : Helligkeit \longmapsto Pixelwert$; siehe Abbildung 2.16) kennen, um aus den zur Verfügung gestellten Werten die Originalhelligkeiten zurückrechnen zu könnnen.

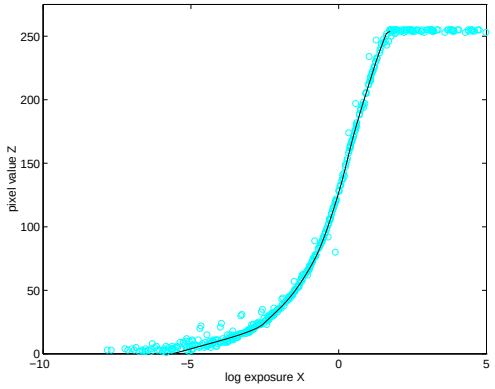


Abbildung 2.16: Eine mögliche rekonstruierte Responsekurve aus [*Debevec und Malik*, 1997].

Die Responsekurve ist kameraspezifisch und könnte vom Hersteller zur Verfügung gestellt werden. Da gerade sie einen wichtigen Teil einer Kamera ausmacht, veröffentlichen die Hersteller ihre Responsekurven meistens nicht. Der größte Teil des Kapitels wird sich daher mit der **Rückgewinnung der Responsekurve** aus verschiedenen Aufnahmen eines Sensors befassen.

Ist die Responsekurve bekannt und stehen genügend Aufnahmen zur Verfügung, um den kompletten Dynamikbereich abzudecken, können die Bilder zu einem HDRI kombiniert werden. Dazu werden die (relativen) Originalhelligkeiten der einzelnen Aufnahmen berechnet, gewichtet und kombiniert:

$$\widehat{I}_j = \frac{\sum_i \omega_{ij} M_{ij}}{\sum_i \omega_{ij}} \tag{2.2}$$

$\widehat{I}$ bezeichnet die Originalhelligkeit in Pixel j. Sie ergibt sich aus der mit der Funktion ω gewichteten Summe der in Bild i gemessenen Helligkeiten M_{ij}. Als Gewichtungsfunktion kann beispielsweise eine Hutfunktion oder ähnliches verwendet werden. Details finden sich im Abschnitt 2.3.3.

[35] Das Analogon bei analogen Filmmaterialien ist die Hurter-Driffield-Kurve, die den Zusammenhang zwischen einfallendem Licht und Filmdichte angibt.

2.2.6 Welchen Nutzen bringt HDRI?

Das heute (2008) wahrscheinlich am weitesten verbreitete Bildformat ist 24-Bit-JPEG [*Wallace*, 1991; *ITU*, 1993] (vergleiche Abschnitt 5.2.7.4). Es ermöglicht ein Kontrastverhältnis von 255:1.

CRT[36]-Monitore haben ein Kontrastverhältnis von 100:1 [*Reinhard et al.*, 2005, 6].

LCD[37]-Monitore erreichen einen Kontrast unter 1000:1 [*Seetzen et al.*, 2004].

Welchen Nutzen bringt unter derartigen Voraussetzungen die HDR-Bildaufnahme mit einem Kontrastverhältnis von bis zu 1 000 000 000 000 : 1?

Schenkt man aktuellen Prognosen Glauben, so werden sich HDR-Displays in den nächsten Jahren verbreiten[38]. Ein starkes Indiz dafür ist, dass Computerspiele und Hollywoodfilme schon heute zum Teil in HDR produziert werden [*Akyüz et al.*, 2007].

Die aktuelle Technik zur Darstellung von HDR auf einem Monitor wird in [*Seetzen et al.*, 2004] vorgestellt[39]. Die Autoren stellen ein LCD vor, dessen Hintergrundbeleuchtung nicht wie bei herkömmlichen LCDs aus einem möglichst homogenen Weiß besteht sondern aus vielen, einzeln ansteuerbaren, (weißen oder farbigen) LEDs. Diese bilden selbst ein Display niedriger Auflösung. Vor diesem groben Display befindet sich das Flüssigkristalldisplay (LCD). Durch geeignete Ansteuerung der beiden Schichten lassen sich im Jahr 2007 Kontraste von bis zu 200 000 : 1 [40] darstellen[41].

Um die Dynamik dieser Displays auszunutzen, werden HDRIs oder zumindest Bilder, die den erhöhten Kontrastbereich des Diplays nutzen, benötigt. Das Max-Planck-Institut für Biologische Kybernetik in Tübingen hat auf der Siggraph 2007 eine Studie zur *Wirkung von LDR-Bildern auf HDR-Displays* vorgestellt [*Akyüz et al.*, 2007]. Die Autoren kommen zu dem Schluss, dass Bilder in HDR für subjektiv angenehmer befunden werden als solche in LDR. Die Untersuchung zeigt weiterhin, dass es bei den heutigen Sehgewohnheiten subjektiv keinen merklichen Unterschied zwischen einem linear kontrastgespreizten (und optimal belichteten) LDRI und dem zugehörigen HDRI gibt[42]. Die Autoren führen dies darauf zurück, dass Menschen vor allem den höheren Kontrast als angenehm empfinden und weniger auf die fehlenden Details im Vergleich zu einem HDRI achten. Damit erscheint es zunächst unnötig, nur zur Anzeige HDR-Daten aufnehmen zu wollen[43]. Die Autoren führen allerdings weiter an, es sei wahrscheinlich, dass sich mit den erweiterten Möglichkeiten der Hardware auch unsere Sehgewohnheiten ändern, und wir dann sehr wohl die zusätzlichen Details der HDR-Bilder wahrnehmen werden. Wenn wir uns daran gewöhnt haben, werden wir HDR vermutlich nicht mehr missen wollen.

Für die vorliegende Arbeit besonders relevant ist das Ergebnis der Studie, dass es meist *keinen* bedeutenden Unterschied zwischen dem am besten belichteten LDRI und dem

[36] Cathode-Ray-Tube, Kathodenstrahlröhre.
[37] Liquid-Crystal-Display.
[38] [*Akyüz et al.*, 2007] oder auch C't 22/ 2007, Heise Zeitschriften Verlag, Seiten 220-223.
[39] Die Vorarbeit findet sich in [*Seetzen et al.*, 2003].
[40] [*Akyüz et al.*, 2007, 2]: BrightSide DR37-P.
[41] Weiterführende Information zu HDR-Hardware findet sich beispielsweise in [*Höfflinger*, 2007].
[42] Komplizierte inverse Mappings von einem LDRI auf ein HDRI, wie sie beispielsweise in [*Rempel et al.*, 2007] vorgestellt werden, sind damit für die Anzeige auf einem HDR-Display unnötig. Dies ist insofern relevant, als es noch lange Zeit LDR-Inhalte geben wird, die auf HDR-Bildschirmen dargestellt werden sollen und somit jedes LDR in den HDR transformiert werden muss.
[43] Für die Verwendung der Helligkeitsdaten in künstlichen Umgebungen oder die Einbettung in, von der Aufnahme abweichend belichtete, Szenen, die Objekterkennung und andere Anwendungen ist die HDR-Aufnahme in jedem Fall hilfreich.

tonegemappten HDRI gibt. Es erscheint also –zumindest für heutige Anzeigegeräte– unnötig, für die LDRI-Gewinnung über ein HDRI zu gehen. Dennoch werden wir in der vorliegenden Arbeit diesen Weg beschreiten. Für (semi-)professionelle Kameras mag die Aussage nämlich durchaus stimmen – nicht jedoch für Webcams. Die –im Vergleich zu Fotokameras sehr schlechten– (Consumer-)Webcamsensoren sind zumeist nicht in der Lage, ein optimal belichtetes LDRI zu erzeugen. Dies liegt an ihrer Größe[44] und der damit verbundenen schlechten Lichtausbeute, die auch zu erhöhtem Bildrauschen führt, sowie an der fehlenden Optik[45].

In dieser Arbeit soll gezeigt werden, dass LDR-Webcambilder stark vom Umweg über ein HDRI profitieren. Als gewünschter Nebeneffekt wird durch die Kombination mehrerer Aufnahmen das *Bildrauschen* verringert, da der Mittelwert über mehrere Bilder gebildet wird. Schließlich wird auch die in Kapitel 3 vorgenommene Registrierung der Bilder zueinander durch HDR-Bilddaten erleichtert, da im HDR sämtliche Featurepunkte vorhanden sind, die in angrenzenden LDRIs über- oder unterbelichtet sein könnten.

Sind die Bilder zueinander registriert, so können sie zusammengefügt werden, *ohne* dass eine *Helligkeitsanpassung* notwendig wird, da sie als HDR-Bilder alle den gleichen Helligkeitsraum abdecken. Dies ist ebenfalls ein Vorteil gegenüber der LDR-Verarbeitung.

Schließlich kann es gerade erwünscht sein, nicht natürlich wirkende Bilder zu erzeugen. Ein unnatürlicher Tonemapping-Operator (siehe Kapitel 4) kann bei Landschaftsaufnahmen zu faszinierenden künstlerischen Ergebnissen führen. Für eine derartig freie Verarbeitung der Originalhelligkeiten sind ebenfalls HDR-Bilddaten notwendig.

Durch die Verwendung der HDR-Software-Aufnahmetechnik erhalten wir sowohl *optimale LDRI*s als auch *HDR-Bilder* für die zukünftige Darstellung auf HDR-Bildschirmen.

[44] Ein Webcamsensor hat eine Größe von 5x4mm (Philips ToUCam Pro) [*Sony*, 2003]. Ein Digitalfotosensor hat die Größe 28x19mm (Canon EOS 1D Mark III) [*Canon Inc.*, 2007].

[45] Bei Consumerwebcams sind normalerweise weder Linsen noch Blenden vorhanden.

2.3 Analyse bestehender HDR-Algorithmen

Die ersten Versuche, die Dynamik eines Sensors mittels mehrerer Aufnahmen zu erhöhen[46], stammen aus der Zeit der ersten „verfügbaren“ Webcams, dem Jahr 1993[47]. Zum Anschluss an einen Rechner standen damals noch keine Kameras im heutigen Sinne zur Verfügung. **Madden** verwendete eine Apparatur aus einem CCD-Chip mit vorgelagerter Optik, die an einen Videograbber angeschlossen war. Er schlägt in seinem Artikel [*Madden*, 1993] vor, mehrere Aufnahmen einer Szene zu kombinieren, um ein detailreicheres Bild zu erhalten[48]. Für jedes Teilbild wird die Belichtungszeit verändert. Er beginnt dazu mit einem kleinen Wert, bei dem kein Pixel gesättigt ist, und erhöht die Belichtungszeit solange, bis alle Pixel einen Wert größer dem Minimum (beispielsweise 0) haben. Anschließend nimmt er aus jedem Bild diejenigen Bereiche, die gut im Arbeitsbereich des Sensors liegen[49], und kombiniert sie zur endgültigen Aufnahme. Er komprimiert die Dynamik dabei so, dass die Auflösung von 8 Bit nicht überschritten wird. Die entstehenden Quantisierungsintervalle müssen dabei nicht notwendigerweise gleich groß ausfallen.

Madden gleicht die unzureichende Dynamik des CCDs also durch mehrere verschieden belichtete Aufnahmen aus. Vor der Quantisierung bildet er die Teilbilder in ein gemeinsames Koordinatensystem ab. Bei seiner Aufnahmeapparatur (s.o.) stehen die aufgenommenen Pixelwerte in linearem Verhältnis zum jeweils aufgefangenen Licht. Durch den linearen Zusammenhang zwischen Originalhelligkeit und ausgegebenem Wert gestaltet sich die Rückrechnung in einen gemeinsamen Wertebereich bei bekannter Belichtungszeit recht einfach als Division: Die Responsekurve ist abschnittsweise linear.

Madden schlägt am Ende seines Artikels unter anderem vor, die Sensordaten direkt in der Kamera nachzuverarbeiten und die Dynamik zu komprimieren. Dadurch entfiele die in seinem Paper vorgestellte aufwendige Nachverarbeitung per Software (siehe 2.2.5). Heute werden die Sensordaten standardmäßig in der Kamera nachverarbeitet, um mehr Dynamik abzubilden und damit natürlicher wirkende Bilder zu erhalten. Heutige Bildsensoren sind deutlich leistungsfähiger als die von Madden eingesetzten. Sie haben eine Helligkeitsauflösung von weit über 8 Bit[50] [*Debevec und Malik*, 1997]. Damit ist es unnötig, verschiedene Aufnahmen zu machen. Das Ausgabeformat umfasst allerdings meist nur 8 Bit (z.B. JPEG). Also müssen die Bilder in der Kamera nicht-linear abgebildet werden, um einen natürlichen (analogen) Bildeindruck zu schaffen (vgl. Responsekurve in Bild 2.16). Dieses Ziel haben auch die in Kapitel 4 vorgestellten ToneMapping-Verfahren.

Mit zunehmend komplexerer Signalverarbeitung in den Kameras wird die Rückrechnung der Originalhelligkeiten komplexer. Für die Gewinnung von HDR-Daten ist die

[46] Das Wyckoff-Verfahren (siehe Abschnitt 4.1) wurde schon vor der Verfügbarkeit der Kamerasensoren auf eine digitale Verarbeitung adaptiert. *Miller und Hoffman* [1984] schlagen die Verwendung einer Sequenz mehrerer gescannter LDR-Aufnahmen für die Generierung einer Light Map für realistisches Rendering vor.

[47] Zur zeitlichen Orientierung: Im November 1993 ging die berühmte Kaffeewebcam in Stanford ans Internet: http://www.cl.cam.ac.uk/coffee/coffee.html.

[48] Maddens Ziel besteht dabei weniger in natürlicheren Aufnahmen als vielmehr in besseren Daten, um beispielsweise Kanten detektieren zu können.

[49] Im Arbeitsbereich eines Sensors liegen alle Pixel, die nicht unter- oder übersättigt sind. Insbesondere werden dies bei den meisten Sensoren die um den mittleren Augabewert (128) liegenden Pixel sein (bspw. Grauwerte $\in [50, 205]$).

[50] Eine professionelle Spiegelreflexkamera hat heute beispielsweise einen Sensor mit 14Bit Dynamikumfang [*Canon Inc.*, 2007].

hardwareseitige Datenmanipulation hinderlich. Maddens Anordnung lieferte eine abschnittsweise lineare Responsefunktion. In der Einleitung zu diesem Abschnitt haben wir die Responsekurve einer heutigen Kamera gesehen (Bild 2.16). Wie auch beim analogen Film ist diese nicht linear. Problematisch für die Rückrechnung auf Originalhelligkeiten ist hierbei weniger die Form als vielmehr die Tatsache, dass die Responsekurven der Kameras nicht zur Verfügung stehen, da die Hersteller sie ungern preisgeben [*Grossberg und Nayar*, 2003a].

In den nachfolgend vorgestellten Arbeiten geht es nicht mehr darum, einen dem analogen Film ähnlichen Abbildungsprozess zu schaffen. Es soll nicht möglichst viel Dynamik auf 8bit komprimiert werden – zur Zeit der Arbeiten wurde dies schon standardmäßig von den Kameras durchgeführt. Die Arbeiten befassen sich vielmehr damit, möglichst die Originalhelligkeiten zurückzurechnen und zu speichern. Sie machen damit in gewisser Weise den von Madden und Anderen[51] vorgeschlagenen Weg der Vorverarbeitung rückgängig. Gleichzeitig bedienen sie sich derselben Technik für einen Ausgleich der unzureichenden Sensorik: Mithilfe mehrerer verschieden belichteter Aufnahmen wird die Sensordynamik erweitert.

Steve Mann spricht in seinem Artikel [*Mann*, 2000] treffend von einem „Analysis–Synthesis"-Ansatz: Es werden verschieden belichtete Aufnahmen ein und derselben Szene gemacht. Dadurch werden verschiedene Bereiche des Helligkeitsspektrums analysiert. Anschließend werden die Aufnahmen zu einem Gesamtbild synthetisiert. Wenn die einzelnen Aufnahmen insgesamt das vorhandene Helligkeitsspektrum abdecken, erhalten wir ein diskretes Abbild der vollständigen Dynamik der Szene. Zur Synthese ist es notwendig, die einzelnen Aufnahmen zueinander in Bezug zu setzen und diese letztlich realen Helligkeiten zuzuordnen.

Der kritischste Punkt hierbei ist die Bestimmung der **Kameraresponsekurve** (vgl. Abb. 2.16), da diese in der Regel nicht zur Verfügung gestellt wird. Wie in der Einleitung geschrieben, ist sie kameraspezifisch und muss daher für jedes Modell nur einmal berechnet werden. Unter Laborbedingungen kann dieser Schritt mithilfe einer Kalibrierungstafel erfolgen (siehe 2.3.1). Die in Abschnitt 2.3.3 nachfolgend ausführlich vorgestellten Verfahren kommen ohne spezielle Kalibrierungsanordnungen aus und bestimmen die (relative) Response mithilfe von Standardaufnahmen. Sie sind daher nicht ausschließlich Experten vorbehalten (siehe Abschnitt 2.3.1) und benötigen keine spezielle Ausrüstung. Für eine absolute Kalibrierung und das Angleichen der Farbkanäle kann eine Kalibrierungstafel (vgl. [*Young-Chang und Reid*, 1996], [*Wyszecki und Stiles*, 1967, 393], [*Reinhard et al.*, 2005, 49]) allerdings weiterhin nützlich und notwendig sein. Ohne absoluten Bezugspunkt kann nur eine relative Kameraresponse berechnet werden.

2.3.1 Responsekurvenbestimmung mit Kalibrierungstafeln

Sind die Originalhelligkeiten jedes Pixels bekannt, so kann man diese gegen die im Bild resultierenden Helligkeitswerte auftragen. Wir erhalten so einzelne Punkte der Responsekurve.

Kennt man die Responsekurve und die Belichtung, so kann man für jeden Bildwert die Originalhelligkeit zurückrechnen.

[51] Eine Auflistung von relevanten Papern bis zum Jahr 2000 findet sich in [*Nayar und Mitsunaga*, 2000, 1].

Die ermittelte Kurve ist spezifisch für die Kamera: Durch Variation der Belichtungszeit oder der Blende[52] wird sie sich lediglich auf der Achse der Originalhelligkeiten verschieben, ihre Form bleibt dabei gleich. Mithilfe der vordefinierten Kalibrierungswerte und der Kenntnis der Belichtungszeit sowie der Blende können wir die resultierenden Werte auf die Originalhelligkeiten zurückrechnen.

Ein Problem dieses Ansatzes ist es, die Originalhelligkeiten zu bestimmen. Eine Kalibrierungstafel[53] erreicht beispielsweise nur bei einer genau spezifizierten Beleuchtung ihre ausgewiesenen Helligkeiten. Eine derartige Lichtsituation ist zumeist nur unter Laborbedingungen zu erzielen.

Weiterhin können sich durch die Aufnahmefehler, die wir in 2.2.2.1 (Chip) und 2.2.2.2 (Linsensystem) betrachtet haben, Kalibrierungsfehler ergeben.

Eine Kalibrierung über eine Kalibrierungstafel ist für einen „Ottonormalanwender“ nur schwer durchzuführen.

2.3.2 Bestimmung einer relativen Responsekurve aus Standardaufnahmen

Wir befassen uns im Folgenden mit der Rückgewinnung der Kameraresponsekurve aus Standardaufnahmen.

Im Gegensatz zu einer absoluten Responsekurve, wie wir sie mithilfe der Kalibrierungstafeln erhalten können, ist bei einer relativen Reponsekurve nicht klar, welchem Originalhelligkeitenintervall die zurückgerechneten Werte entsprechen. Lediglich der Bezug der ermittelten Werte zueinander ist bekannt: Die Kurve ist relativ. Sobald ein Originalhelligkeitswert zur Kalibrierung vorliegt, wird aus der relativen Kurve eine absolute.

Der Großteil der zu betrachtenden Algorithmen beruht auf demselben Prinzip: Es werden Helligkeiten in verschieden belichteten Aufnahmen zueinander in Bezug gesetzt und dadurch eine relative Responsekurve gewonnen. Wenn die Belichtungsunterschiede bekannt sind, können Rückschlüsse über den vorhandenen Originalhelligkeitswert gezogen werden: „Bei dreifacher Belichtung wird der vom Sensor gelieferte Wert dreimal so hoch sein“. Siehe auch Schaubild 2.17.

Für diese Klasse von Algorithmen betrachten wir zunächst die grundlegende Vorgehensweise und analysieren das Problem mathematisch. Die Vorlage dazu bilden die Paper [*Grossberg und Nayar*, 2002] und zum Teil [*Mann*, 2000].

Anschließend gehen wir genauer auf die einzelnen Veröffentlichungen und die darin vorgestellten Verfahren ein. Wir werden uns dabei weitgehend am Erscheinungsdatum orientieren und chronologisch von den älteren zu den neueren Veröffentlichungen weitergehen. Dadurch lässt sich auch die Entwicklung der Forschung auf diesem speziellen Gebiet nachvollziehen. Wo dies sinnvoll erscheint, werden spätere Paper vorgezogen, um den Zusammenhang aufzuzeigen.

[52] Eine Variation der Blende geht immer mit einer Variation der Tiefenschärfe einher [*Robertson et al.*, 2003; *Zettl*, 2004] und verändert daher mitunter die Abbildung (vgl. Abschnitt 2.2.2.2). Außerdem kommt es eventuell zu Vignettierung (siehe 2.2.2.2).

[53] Ein Beispiel für eine MacBeth-Farbkalibrierungstafel findet sich in [*Reinhard et al.*, 2005, 49].

2.3.2.1 Mathematische Analyse des Standardansatzes

Sei f die Responsefunktion der Kamera, dann ergibt sich der resultierende Helligkeitswert M aus der Originalhelligkeit I:

$$M = f(I)$$

O.B.d.A. können wir den Definitionsbereich und den Wertebereich von f auf das Intervall $[0, 1]$ abbilden. Weiterhin legen wir o.B.d.A. fest, dass gelte $f(0) = 0$ und $f(1) = 1$.

Betrachten wir ein Bildpaar A, B mit den zugehörigen Belichtungen[54] e_A und e_B mit $e_A < e_B$. Sei I_A die Helligkeit an einem bestimmten Punkt in Bild A und I_B die Helligkeit im entsprechenden Punkt von Bild B. Dann gilt $I_A/e_A = I_B/e_B$. Sei $k = e_B/e_A$ das Verhältnis der beiden Belichtungen zueinander und damit $I_B = kI_A$. Weiterhin seien die Bildwerte $f(I_A) = M_A$ und $f(I_B) = M_B$. Unter der Annahme, dass f monoton ist, existiert die Umkehrabbildung[55] $g = f^{-1}$.

Aus $I_B = kI_A$ folgt:

$$g(M_B) = kg(M_A) \tag{2.3}$$

Die Beziehung zwischen dem idealen Helligkeitswert I und der Leuchtdichte L eines Punktes lässt sich beschreiben als [*Grossberg und Nayar*, 2002]:

$$I = LPe$$

P ist eine Konstante, die die Helligkeitsänderung durch die Linse beschreibt[56] (siehe Abschnitt 2.2.2.2), e ist die Belichtung. Die **Belichtung** e setzt sich zusammen aus dem Blendendurchmesser d und der Belichtungszeit t: $e = \left(\pi d^2\right) t$.

Daraus folgt für die Rückrechnung der Originalleuchtdichte:

$$L = \frac{1}{eP} g \circ f(I) \tag{2.4}$$

mit Szenenhelligkeit I[57], Belichtung e, Filmresponsefunktion f und ihrer Inversen g [*Grossberg und Nayar*, 2002; *Mann*, 2000].

Bei einigen Algorithmen ist die Kenntnis der genauen Belichtung e und damit der Belichtungsverhältnisse k nicht erforderlich. Diese wird dann im Zuge der Kalibrierung bestimmt.

Mit Hinzunahme eines jeden Punktpaares (M_A, M_B) wird der Lösungsraum für g weiter eingeschränkt. Dass es unmöglich ist, g zu bestimmen, ohne Anforderungen an die Kurve zu stellen, werden wir im Folgenden zeigen. Welche Anforderungen die bisherigen Paper im Einzelnen gewählt haben, sehen wir in Abschnitt 2.3.3.

Der Weg zur Gewinnung der Responsekurve führt über den Vergleich der Helligkeitspaare (M_A, M_B). Sei T die **Transferfunktion**, welche die Helligkeiten aus Bild A auf diejenigen in Bild B abbildet:

$$M_B = T(M_A) := f(kf^{-1}(M_A)) = g^{-1}(kg(M_A)) \tag{2.5}$$

[54] Der Begriff „Belichtung“ beschreibt die einfallende Lichtmenge. Er ist sowohl von der Belichtungszeit als auch von der Blende abhängig.

[55] Nur unter der Annahme, dass f *streng* monoton ist, existiert eine stetige Umkehrfunktion.

[56] $P = \cos^4 \alpha / c^2$, α ist der Winkel zur optischen Achse, c die Brennweite.

[57] Die Szenehelligkeit wird in dieser Arbeit mit I und E bezeichnet. E wird immer dann verwendet, wenn zum Ausdruck gebracht werden soll, dass es sich um die rückgewonnene Helligkeit, nicht jedoch um die „echte“ Helligkeit I handelt.

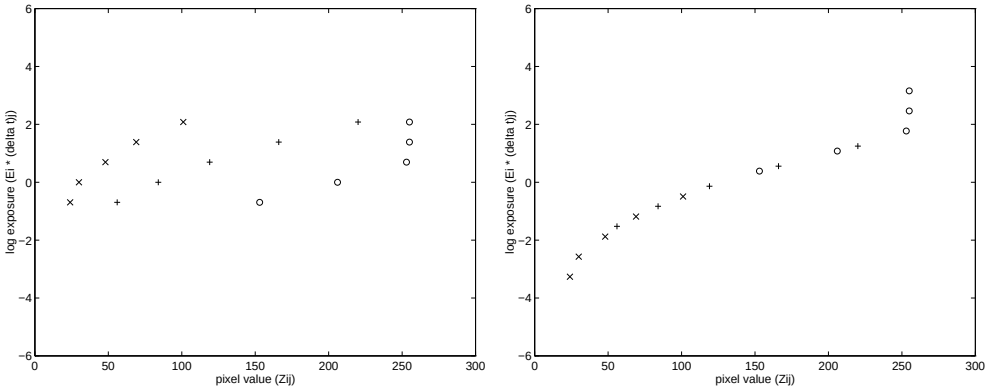


Abbildung 2.17: Das Schaubild zeigt die Werte dreier Pixel (x, +, ○) in jeweils fünf verschieden belichteten Bildern. Da die absolute Helligkeit unbekannt ist, sind die Referenzwerte im linken Diagramm zunächst alle gleich. Im rechten Bild wurde die Varianz in Bezug auf eine Kurve minimiert. Die vertikale Lage der gesamten Kurve ist immer noch unbekannt [*Debevec und Malik*, 1997].

Um lokale Fehler auszugleichen, ist es sinnvoll, jeweils *Gruppen* von Pixeln gleicher Helligkeit zu betrachten.

Globale, konstant auftretende Fehler, die die Verhältnisse der Bildhelligkeiten zueinander unverändert lassen, haben keinen Einfluss auf die Gewinnung der Responsekurve[58].

Mit der Transferfunktion T folgt aus 2.3:

$$g(T(M)) = kg(M) \tag{2.6}$$

Aus der Monotonie von g folgt in Gleichung 2.5, dass T monoton ist. Somit existiert die Umkehrabbildung T^{-1}. Wir definieren rekursiv:

$$\begin{aligned} T^0(M) &:= M \\ T^n(M) &:= T(T^{n-1}(M)) \\ T^{-n}(M) &:= T^{-1}(T^{-n+1}(M)) \end{aligned}$$

Theorem 2.1 (Eigenschaften der Helligkeitstransferfunktion) *Sei g eine monoton steigende, glatte Funktion mit glatter Inversen. Sei $g(0) = 0$, $g(1) = 1$, $k > 0$. Dann hat $T(M) := g^{-1}(kg(M))$ folgende Eigenschaften:*

1. *T ist monoton wachsend,*
2. *$T(0) = 0$,*
3. *$\lim_{n\to\infty} T^{-1}(M) = 0$,*
4. *Wenn $k > 1$ gilt $M \leq T(M)$.*

[58] Bei konstanter Blende zählen die Linsenaberrationen zu den konstanten Fehlern (siehe 2.2.2.2). Ändert sich jedoch die Blende zwischen den Bildern, so können die Linsen zu dynamischen Fehlern führen (Vignettierung etc.). Vor allem ab dem Sensor treten durch diverses Rauschen dynamische Fehler auf, die durch die Verwendung und Gewichtung innerhalb von Pixelpools verringert werden können.

Beweis 2.1 *zu Theorem 2.1*

1. *Da g glatt und monoton wachsend ist, gilt $g' \geq 0$. Aus Gleichung 2.6 folgt $T'(M) = kg'(M)/g'(T(M))$. Da $T'(M) \geq 0$ ist $T(M)$ monoton wachsend.*
2. $T(0) = g^{-1}(kg(0)) = g^{-1}(0) = 0$.
3. *Wir betrachten die absteigende Folge $M > T^{-1}(M) > T^{-2}(M) > \cdots$. Da $k > 1$ gilt $g(M) \leq kg(M) = g(T(M))$ (siehe 2.6). Die Folge ist nach unten durch 0 begrenzt. Also muss die Folge gegen einen Punkt M^* konvergieren. Für diesen Punkt gilt $T(M^*) = M^*$, also $g(M^*) = g(T(M^*)) = kg(M^*)$. Da $k > 1$ muss $g(M^*) = 0$ gelten und damit $M^* = 0$.*
4. *Da g monoton wachsend ist, folgt aus $M_1 \leq M_2$, $g(M_1) \leq g(M_2)$. Da $k > 1$, gilt $g(M) \leq kg(M) = g(T(M))$. Da g^{-1} ebenfalls monoton wächst, gilt weiter $M = g^{-1}(g(M)) \leq g^{-1}(g(T(M))) = T(M)$.*

Die Voraussetzung $k > 1$ bedeutet hierbei lediglich, dass die Bilder in aufsteigender Helligkeit sortiert sind.

Grossberg und Nayar [2002] fahren damit fort, zu zeigen, dass die **Lösung des Problems** ohne Weiteres **nicht eindeutig** ist. Aus Theorem 2.1 Punkt 4 folgt, dass die Transferfunktion bei $k > 1$ mit Hinzunahme eines jeden Bildes die Achse der Bildhelligkeiten M erweitert. Aus Gleichung 2.6 folgt, dass die Kurve durch Erweiterung des Definitionsbereichs mithilfe von T und des Wertebereichs mithilfe von k *selbstähnlich* wird. Zur Erweiterung werden die durch g abgebildeten Punkte aus dem Intervall $(T^{-1}(1), 1]$ auf der linken Seite mit solchen aus dem Intervall $(T^{-2}(1), T^{-1}(1)]$ auf der rechten Seite in Beziehung gesetzt.
Die Bedingungen an die Responsefunktion g führen nur dazu, dass g innerhalb des jeweiligen Intervalls stetig und monoton sein muss. Über die Kurvenform wird keine Aussage getroffen, da die Punkte innerhalb eines Intervalls (T^{-1}, T) nicht zueinander korreliert werden. Dies bezeichnen die Autoren als Fraktaleigenschaft von g.
Wir können also jede gültige Kurvenform $s(M)$ im Intervall $[T^{-1}(1), 1]$ auf eine gültige Lösung für das gesamte g erweitern.

Theorem 2.2 (Fraktaleigenschaft der Responsekurve) *Sei T eine Transferfunktion, die die Eigenschaften aus Theorem 2.1 erfüllt. Sei $s(M)$ eine stetige monotone Funktion auf dem Intervall $[T^{-1}(1), 1]$ mit $s(1) = 1$ und $s(T^{-1}(1)) = 1/k$. Dann kann s zu einer eindeutigen, stetigen und monotonen Funktion g auf $[0, 1]$ erweitert werden, sodass gilt $g(M) = s(M)$ für $M \in [T^{-1}(1), 1]$. Jede so gebildete Funktion g erfüllt die geforderten Eigenschaften $g(T(M)) = kg(M)$ für $M \in [0, T^{-1}(1)]$ mit $g(0) = 0$ und $g(1) = 1$.*

Beweis 2.2 *Wir betrachten eine absteigende Folge von Punkten*

$$1 > T^{-1}(1) > T^{-2}(1) > \cdots$$

Wegen $\lim_{n\to\infty} T^{-1}(M) = 0$ gibt es eine nicht negative Zahl $r(M)$, sodass

$$M \in (T^{-r(M)-1}(1), T^{-r(M)}(1)]$$

Wir definieren die Funktion

$$g(M) = \frac{1}{k^{r(M)}} s(T^{r(M)}(M)) für M > 0, g(0) = 0 \tag{2.7}$$

Mit $r(T(M)) = r(M) - 1$ *folgt*

$$\begin{aligned} g(T(M)) &= \frac{1}{k^{r(T(M))}} s(T^{r(T(M))}(T(M))) \\ &= \frac{1}{k^{r(M)-1}} s(T^{r(M)-1}(T(M))) \\ &= \frac{k}{k^{r(M)}} s(T^{r(M)}(M)) \\ &= kg(M) \end{aligned}$$

g erfüllt also die Bedingung $g(T(M)) = kg(M)$. Da s stetig und monoton ist, ist es auch $g(M)$ innerhalb der offenen Intervalle $(T^{-n-1}(1), T^{-n}(1))$. Wegen $s(1) = 1, s(T^{-1}(1)) = 1/k$ und der Monotonie von s folgt aus 2.7, dass auch g monoton ist. Da $s(M)$ an $s(1)$ und $s(T^{-1}(1))$ stetig ist, ist es auch g an $T^{-n}(1)$. Mit $\lim_{n\to\infty} g(T^{-n}(1)) = \lim_{n\to\infty} \frac{1}{k^n} = 0$ ist g an der Stelle 0 stetig.

Aus Theorem 2.2 folgt, dass es ohne weitere Einschränkungen der Form viele Lösungen für g gibt. Da die fraktale Mehrdeutigkeit zwischen den Messpunkten auftritt, liegt es nahe, möglichst viele Stützstellen für die Funktion zu messen. Ein k nahe 1 verkleinert die Intervalle $(T^{-1}(1), 1)$. Die Autoren schreiben, dass dies aufgrund von Rauschen nicht erfolgversprechend ist.

In [*Grossberg und Nayar*, 2003a] folgt eine Lösung für das Problem für den Fall, dass die Belichtungen nicht alle den gleichen Abstand k haben. Existierten zwei verschiedene Belichtungsverhältnisse k_1, k_2 mit den zugehörigen Transferfunktionen T_{k_1}, T_{k_2}. Dann gilt

$$T_{k_1}^m(T_{k_2}^{-n}) = g^{-1}(k_1^m/k_2^n g(1)) = g^{-1}(k_1^m/k_2^n)$$

Für den Fall, dass $\log_{k_1} k_2$ irrational ist, gilt k_1^m/k_2^n ist dicht in $(0, 1)$. Aus dem Theorem von Kronecker folgt, dass die Nachkommastellen der Vielfachen einer irrationalen Zahl im Intervall (0,1) dicht liegen [*Vinogradov*, 1990, 310]:

$$\{z - \lfloor nz \rfloor\}_{n=1}^{\infty} \text{ ist genau dann dicht im Intervall } (0, 1), \text{ wenn } z \in \mathbf{R} \setminus \mathbf{Q}$$

Daraus folgt, dass $\{m + nz \mid (m, n) \in \mathbf{Z}\}$ dicht in $\mathbf{R}$ liegt. Damit liegt es auch dicht in $(-\infty, 0)$. Exponenzieren wir mit einer positiven Zahl k_1 so folgt

$$k_1^m k_1^{-nz} \text{ liegt dicht in } (0, 1).$$

Setzen wir $z = log_{k_1} k_2$ so folgt $k_1^m k_1^{-nz} = k_1^m k_2^{-n}$. Damit folgt:

$$\frac{k_1^m}{k_2^n} \text{ liegt genau dann dicht im Intervall } (0, 1), \text{ wenn } log_{k_1} k_2 \in \mathbf{R} \setminus \mathbf{Q}$$

Wir haben also gesehen, dass die fraktale Mehrdeutigkeit für den Fall, dass wir zwei unterschiedliche Belichtungsreihen mit den eben beschriebenen Eigenschaften haben, auflösbar ist. Als ein gutes Verhältnis zwischen den Belichtungen schlagen die Autoren $\sqrt{2}$ vor. Diese Erkenntnis ist allerdings für reale Aufnahmegeräte nur theoretischer Natur, da die Belichtungszeiten Vielfache (Harmonische) voneinander sind und daher gerade kein irrationales Verhältnis zueinander haben.

Sind sowohl g als auch k unbekannt, erweitert sich der Lösungsraum erneut und zwar um die **Potenzen** der Lösungen. Grossberg und Nayar veranschaulichen dies folgendermaßen: Nehmen wir an, System A habe die inverse Kameraresponsefunktion

$g_A(M) = M^\gamma$. Für System B gelte $g_B(M) = M$. Das erste zur Kalibrierung verwendete Bild ist bei beiden Systemen gleich belichtet. Bei System A folgen die weiteren Bilder im Abstand 2^γ, bei System B im Abstand 2. A hat die Helligkeitstransferfunktion $T_A(M) = g_A^{-1}(2^\gamma g_A(M)) = (2^\gamma M^\gamma)^{-\gamma} = 2M = g_B^{-1}(2g_B(M)) = T_B(M)$. Obwohl A und B also verschiedene Bilder erzeugen, ist dies bei der Rückgewinnung aus den Bildern nicht mehr ersichtlich, da die Helligkeitstransferfunktion T identisch ist. Diese Erweiterung des Lösungsraumes um die Exponenten folgt auch direkt aus Gleichung 2.3 [*Lin et al.*, 2004]:

$$g^\gamma(M_B) = k^\gamma g^\gamma(M_A)$$

Theorem 2.3 (Uneindeutigkeit des Exponenten) *Seien g_1 und g_2 Inverse der Responsekurven und k_1, k_2 Belichtungen, sodass gilt*

$$\begin{aligned} g_1(T^{-1}(M)) &= k_1^{-1} g_1(M) \\ g_2(T^{-1}(M)) &= k_2^{-1} g_2(M) \end{aligned}$$

Wir definieren eine Funktion $\beta(I) := g_2(g_1^{-1}(I))$ [59]*. Dann existieren feste γ und K, sodass $k_1^\gamma = k_2$ und $\beta(I) = KI^\gamma$.*

Die Funktion β in Theorem 2.3 beschreibt den Zusammenhang zwischen den Funktionen, die zur selben inversen Responsefunktion g führen. Der Grund dafür ist die Gleichheit der Transferfunktion dieser Funktionen:

$$g^{-1}(kg(M)) = T(M) = g^{-\gamma}(k^\gamma g^\gamma(M))$$

Beweis 2.3 *β ist monoton, da g_1 und g_2 monoton sind. Es gilt $\beta(g_2(M)) = g_1(M)$.*

$$\begin{aligned} \beta(k_1^{-n} g_1(M)) &= \beta(g_1(T^{-n}(M)) \\ &= g_2(T^{-n}(M)) \\ &= k_2^{-n} g_2(M) \\ &= k_2^{-n} \beta(g_1(M)) \end{aligned}$$

Sei $c = g_1(M)$, $\gamma = \frac{ln(k_2)}{ln(k_1)}$ und $k_1^{-n} = a$. Mit $k_1^\gamma = k_2$ [60] *lässt sich die vorige Gleichung schreiben als*

$$\beta(ac) = a^\gamma \beta(c)$$

Hieraus folgt die Mehrdeutigkeit der Lösung.

Für eine Folge von Punkten

$$p \geq T^{-1}(p) \geq T^{-2}(p) \geq \cdots$$

[59] β bestimmt die Originalhelligkeit zu einem mit System1 aufgenommenen Punkt, die dieser gehabt hätte, wenn er mit System2 aufgenommen worden wäre:

$$f_1(I) = g_1^{-1}(I) = g_2^{-1}\beta(I) = f_2(\beta(I))$$

[60]

$$\begin{aligned} \gamma &= \frac{ln(k_2)}{ln(k_1)} \\ \gamma ln(k_1) &= ln(k_2) \\ k_1^\gamma &= k_2 \end{aligned}$$

gilt für g_1:

$$g_1(p) \geq \frac{1}{k_1} g_1(p) \geq \frac{1}{k_1^2} g_1(p) \geq \cdots$$

Für g_2 gilt:

$$g_2(p) \geq \tfrac{1}{k_2} g_2(p) \geq \tfrac{1}{k_2^2} g_2(p) \geq \cdots$$
$$g_2(p) \geq \tfrac{1}{k_1^\gamma} g_2(p) \geq \tfrac{1}{k_1^{2\gamma}} g_2(p) \geq \cdots$$
$$\beta(g_1(p)) \geq \tfrac{1}{k_1^\gamma} \beta(g_1(p)) \geq \tfrac{1}{k_1^{2\gamma}} \beta(g_1(p)) \geq \cdots$$
$$K \geq \tfrac{1}{k_1^\gamma} K \geq \tfrac{1}{k_1^{2\gamma}} K \geq \cdots$$

Mit $K = \beta(g_1(p))$ [61]. Es ist leicht zu sehen, dass die Transferfunktion zwischen den Folgengliedern sowohl für g_1 als auch für g_2 dieselbe ist. Da die Gleichungen bis auf einen Skalierungsfaktor eindeutig bestimmt sind, haben wir gezeigt, dass eine Mehrdeutigkeit nur in Bezug auf die Exponentialfunktion $\beta(M) = KM^\gamma$ besteht. Wenn gilt $\forall M \beta(M) = g_2(g_1^{-1}(M))$, so folgt aus $\beta(1) = 1$, dass $K = 1$ ist.

Die Mehrdeutigkeit des Exponenten tritt nur auf, wenn sowohl g als auch k unbekannt sind. Betrachten wir zwei Lösungsmöglichkeiten: *Mitsunaga und Nayar* [1999] legen fest, dass g ein Polynom ist. Dies schränkt die Möglichkeiten der Werte, die γ annehmen kann, stark ein. Weiterhin argumentieren sie, dass die ungefähren k bekannt sind und die Lösungen weit genug auseinander liegen, um iterativ zum richtigen Exponenten zu konvergieren. *Tsin et al.* [2001] stellen als Anforderungen an g nur die Glattheit und die Monotonie. Zusätzliche Einschränkungen finden sich bei dem statistischen Ansatz in der Fehlerfunktion, die minimiert wird.

Wir haben gesehen, dass die Bestimmung einer eindeutigen (relativen[62]) inversen Responsekurve g nicht möglich ist, wenn wir als Einschränkungen lediglich Monotonie und Stetigkeit der Funktion fordern. Weiterhin haben wir gesehen, dass der Lösungsraum sich vergrößert, wenn wir die genauen Belichtungen k nicht kennen.

Die nun folgenden Ansätze der verschiedenen Autoren innerhalb der eben vorgestellten Algorithmenklasse unterscheiden sich in ihren **Voraussetzungen**[63]. Augenscheinlich sind hierbei die Annahmen über die **Kurvenform**. Unterschiedlich sind auch die **Gewichtungsfunktion**, mit deren Hilfe die Punkte für die **Gewinnung** der Responsekurve bestimmt werden, wie auch diejenige Funktion, mit der die Einzelbilder dann später zum HDRI **kombiniert** werden. Die Abbildungsfehler (2.2.2.1, 2.2.2.2) legen es nahe, jeweils möglichst viele Aufnahmen zu verwenden, um den zusätzlichen Fehler zu minimieren.

Der Lösungsraum, in dem die Verfahren operieren, lässt sich beschreiben als [*Grossberg und Nayar*, 2003a]:

$$W_f := \{f \mid f(0) = 0, f(1) = 1, f' > 0\}$$

[61] Aus der Definition von β: $\beta(I) = g_2(g_1^{-1}(I))$ folgt durch Einsetzen $\beta(g_1(I)) = g_2(I)$.

[62] Dies ist kein Widerspruch zur Eindeutigkeit. Ohne absolute Kalibrierung gegen bekannte Helligkeitswerte können wir nur eine relative Kurve erhalten. Relativ ist hierbei auch die Skalierung, da sie ebenfalls nicht eindeutig bestimmbar ist.

[63] Einmal wird die genaue Kenntnis der Belichtungszeit vorausgesetzt, ein anderes Mal müssen die Belichtungen nur sortiert sein etc.

2.3.3 Veröffentlichungen zur Responsekurvenbestimmung 1995-2007

Mann und Picard sind 1995 die Ersten, die ein Vorgehen nach eben besprochenem Muster vorschlagen. In vorausgegangenen Veröffentlichungen hatte sich Mann damit beschäftigt, die räumliche Auflösung von Bildern durch Kombination mehrerer Aufnahmen zu vergrößern. In [*Mann und Picard*, 1995] geht es um die Erweiterung des abgebildeten Helligkeitsbereiches mit dieser Technik.

Die Autoren fordern die Monotonie und den Durchgang der Responsekurve durch den Punkt $(0, 0)$. Sie machen weitreichende Einschränkungen an die Kurvenform, indem sie eine Gammakurve als Responsefunktion wählen:

$$f(q) = \alpha + \beta q^{\gamma} \tag{2.8}$$

α modelliert in der Gleichung das Grundrauschen und wird in einem initialen Kalibrierungsschritt[64] ermittelt und fortan abgezogen.

Sie setzen voraus, dass das Verhältnis der Belichtungen zueinander bekannt ist[65]. Ihre Gewichtungsfunktion zur Kombination der Aufnahmen (siehe Gleichung 2.2) ist die Ableitung der entsprechend der Belichtung verschobenen Responsekurve.

Wie im vorigen Abschnitt beschrieben, betrachten die Autoren Pixelpaare aus verschiedenen Belichtungen. Sie beginnen mit einem dunklen Pixel in Bild A. Dieses Pixel identifizieren sie in Bild B. Es gilt $f_{B_{x,y}}(I) = f_{A_{x,y}}(kI)$: Der an $B_{x,y}$ gemessene Wert ist durch k-fache Belichtung entstanden und korrespondiert daher mit einem entsprechenden Wert $f_{A_{x,y}}(kI)$ aus Bild A. Ein Pixel dieses Wertes in Bild A wird als Ausgangspunkt für die rekursive Fortsetzung des Verfahrens verwendet. Am Ende stehen mit $f(I), f(kI), f(k^2I), \ldots, f(k^nI)$ Werte für die Belichtungen $1, k, k^2, \ldots, k^n$ zur Verfügung. An diese kann die gammaförmige Responsekurve mithilfe linearer Regression angenähert werden.

In [*Mann*, 1999] erläutert Mann den Ansatz genauer. Die obige Helligkeitenreihe besteht aus den Komponenten $R_n = f(k^nI)$. Sei p ein so genannter projektiver Wyckoff-Operator[66], sodass gilt $\widehat{R}_1 = p_{12}(R_2)$. Dann gilt für die anderen Punkte der Reihe folgende Korrespondenz:

$$\widehat{R}_1 = p_{12} \circ p_{23} \circ p_{34} \circ \cdots \circ p_{n-1,n}(R_n) \tag{2.9}$$

Damit legt er den Grundstein für seinen „comparametric approach“ in [*Mann*, 2000], den wir weiter unten betrachten werden.

Debevec und Malik kritisieren am vorgestellten Verfahren die willkürliche Kurvenform. Weiterhin bemängeln sie das nicht berücksichtigte Bildrauschen sowie die Nichtausschöpfung der vorhandenen Messdaten, da immer nur Paare aus benachbarten Belichtungen unter den oben beschriebenen Kriterien ausgewählt werden. Sie motivieren ihr Verfahren in [*Debevec und Malik*, 1997] mit folgender physikalischer Grundlage:

Die Belichtung X in $[\frac{J}{m^2}]$ sei definiert als $X = E\Delta t$, wobei E die Bestrahlungsstärke (einfallende Lichtmenge) ist. Sei f die Kameraresponsefunktion, die die Verarbeitungsschritte hin zum gespeicherten Pixelwert M beschreibt. Wenn f monoton ist, existiert

[64] Die Autoren schlagen zur Kalibrierung die Aufnahme mit geschlossener Objektivabdeckung vor.
[65] $I_B = kI_A$
[66] Vergleiche Definition der Transferfunktion $T(M)$ in Gleichung 2.6

die Umkehrabbildung und es gilt $E = f^{-1}(M)$. Weiterhin gilt $E \sim I$. E sei also proportional zur real existierenden Lichtmenge. Der Proportionalitätsfaktor kann dabei beispielsweise an verschiedenen Stellen einer Linse unterschiedlich sein:

$$E = I(\frac{cos^4\Phi}{h^2})(\frac{\Pi d^2}{4}\Delta t) = Ike$$

h ist hierbei die Brennweite, d der Blendendurchmesser und Φ der zwischen dem Hauptlichtstrahl und der optischen Achse eingeschlossene Winkel. e bezeichnet den für die aktuelle Belichtung charakteristischen Faktor (siehe auch 2.2.2.2) [*Mitsunaga und Nayar*, 1999].

Der Messwert des einfallenden Lichts ist abhängig von dessen Wellenlänge:

$$\int X(\lambda)R(\lambda)d\lambda,$$

mit $R(\lambda)$ als spektrale Empfindlichkeit des Sensors auf Wellenlänge λ. Daher können auch die Responsekurven für die drei Farbkanäle jeweils anders aussehen.

Wie auch beim vorigen Verfahren setzen die Autoren voraus, dass die Belichtungszeiten bekannt sind sowie die Beleuchtung und der Bildinhalt während der Aufnahme konstant bleiben. Als Eingabe in das lineare Gleichungssystem wird lediglich die Glattheit von f verwendet.

Nehmen wir i als Index über die Pixel und j als Index über die verschieden belichteten Bilder, so folgt:

$$\begin{aligned} M_{ij} &= f(E_i\Delta t_j) \\ f^{-1}(M_{ij}) &= E_i\Delta t \\ ln(f^{-1}(M_{ij})) &= ln(E_i) + ln(\Delta t) \\ g(M_{ij}) &= ln(E_i) + ln(\Delta t) \end{aligned}$$

(Zu beachten: Im Gegensatz zu der im allgemeinen Teil 2.3.2 vorgestellten inversen Responsefunktion bezeichnen die Autoren hier den Logarithmus dieser Funktion mit g.)

Als quadratisches Minimierungsproblem geschrieben ergibt sich:

$$o = \sum_{i=1}^{N}\sum_{j=1}^{P}[g(M_{ij}) - lnE_i - ln\Delta t_j]^2 + \sigma \sum_{m=M_{min}+1}^{M_{max}-1} g''(m)^2 \tag{2.10}$$

Der hintere Summand dient hierbei zur Modellierung der Glattheit der Funktion[67]. Unbekannt sind die Funktion g sowie die Koeffizienten E_i. Wichtig bei diesem Verfahren ist die genaue Kenntnis der Δt.

Eine spezielle Kurvenform wird bis auf die beschriebenen Annahmen nicht vorausgesetzt.

Es werden lediglich die diskreten Werte für die 255 Helligkeitsstufen eines Bildes ermittelt, jedoch keine kontinuierliche Responsekurve.

Die Autoren nehmen an, dass der Fehler am Rande des Wertebereichs am größten ist. Daher gewichten sie die Messwerte mit einer Hutfunktion zwischen den Extrema, um die Abbildungsfehler am Rand des Sensorarbeitsbereichs schwächer einfließen zu lassen:

$$w(m) = \begin{cases} m - M_{min}, & m \leq \frac{1}{2}(M_{min} + M_{max}) \\ M_{max} - m, & m > \frac{1}{2}(M_{min} + M_{max}) \end{cases} \tag{2.11}$$

[67] σ wird abhängig vom erwarteten Rauschen gewählt.

w wird sowohl zur Gewinnung der Responsekurve als auch zur Gewichtung im Zuge der Kombination der einzelnen Aufnahmen verwendet (siehe Gleichung 2.2).

Setzen wir N="Anzahl der Pixel" und P="Anzahl der Einzelbilder", so ist der Berechnungsaufwand des Verfahrens $N \times P + M_{max} - M_{min}$. Die Autoren bemerken, dass es für die Gewinnung eines HDRI ausreicht, N und P so zu wählen, dass $N(P-1) > (M_{max} - M_{min})$ gilt. Die N Pixel sollten über $[M_{min}, M_{max}]$ sowie lokal über das Bild gleichverteilt sein.

Die Autoren setzen sich kurz mit der Problematik von Farbbildern auseinander: Wie [*Mann und Picard*, 1995] und wie durch die einleitende Erklärung auch als verfahrensinhärent bekannt, berechnet der Algorithmus nur eine relative Responsekurve ($L \sim E$). Um daraus eine Absolute zu erhalten, kann man die absolute Helligkeit mit einem Belichtungsmesser messen oder eine Kalibrierungstafel fotografieren (vergleiche Abschnitt 2.3.1). Besonders bei Farbbildern ist dies eventuell notwendig, denn die Responsekurven werden für jeden Farbkanal separat berechnet und können so gegeneinander verschoben sein[68]. Debevek und Malik normieren die Kurven der einzelnen Grundfarben auf einen Punkt in der Mitte des Dynamikbereichs $g(M_{mid}) = 0$. Spricht der CCD-Sensor auf einen Farbkanal stärker oder schwächer an, so ist dessen Responsekurve nach oben oder unten verschoben. Dadurch bekommt das Bild einen Farbstich, der durch eine einmalige Bestimmung des chipspezifischen Kanalverhältnisses beseitigt werden kann.

Für die Erstellung des HDR-Bildes schlagen die Autoren vor, *alle* Bilder unter Verwendung der Gewichtungsfunktion (Gleichung 2.11) zu verwenden:

$$lnE_i = \frac{\sum_{j=1}^{P} \omega(M_{ij})(g(M_{ij}) - ln\Delta t)}{\sum_{j=1}^{P} \omega(M_{ij})}$$

Dadurch wird das Verfahren stabiler und fehlertoleranter als ein Verfahren, das nur ausgewählte Bilder verwendet.

Robertson, Borman und Stevenson stellen in [*Robertson et al.*, 1999] und etwas ausführlicher in [*Robertson et al.*, 2003] ein ähnliches Verfahren vor. Sie erwähnen explizit mögliches Rauschen als Problem, lassen dies aber bei ihrem Ansatz mit Verweis auf die wünschenswerte Allgemeinheit der Lösung weg. Im späteren Paper schreiben sie, dass eine sich ändernde Gainspannung, also die Verstärkung des Signals seitens der Kamera, ebenfalls für die Rückgewinnung der Responsefunktion berücksichtigt werden muss. Auch darauf gehen sie allerdings nicht näher ein.

Als Gewichtungsfunktion für die Gewinnung der Responsekurve sowie die Kombination der Aufnahmen wählen sie im ersten Paper eine Gaußkurve mit dem Zentrum bei Pixelwert 128 [69]. Im zweiten Paper wählen sie ähnlich [*Mann und Picard*, 1995] zur Rekonstruktion die Ableitung der Responsekurve im aktuellen Iterationsschritt. Diese nähern sie durch einen kubischen Spline an. Um den Rand des Wertebereichs auszuschließen, setzen sie dort die Ableitungen auf Null.

Zur Rückgewinnung der Originalhelligkeiten E_j [70] minimieren sie die Gleichung

$$O(M) = \sum_{i,j} w_{i,j}(R_{y_{i,j}} - e_i E_j)^2$$

Der gewichtete Fehler zwischen rückgewonnener Originalhelligkeit des j-ten Pixels im i-ten Bild $R_{y_{i,j}}$ und der mit der Belichtung multiplizierten Originalhelligkeit $e_i E_j$ wird also minimiert. Somit haben länger belichtete Werte höheres Gewicht.

$$\widehat{E}_j = \frac{\sum_i w_{j,j} e_i R_{y_{i,j}}}{\sum_i w_{j,j} e_i^2} \tag{2.12}$$

[68] Graue Flächen haben dann beispielsweise einen Farbstich.
[69] Das Zentrum liegt also in der Mitte des Wertebereichs.
[70] Der Index j bezeichne hier das j-te Pixel.

Für die Gewinnung der Responsekurve schlagen sie ein iteratives Verfahren vor. Sie wählen als initiale inverse Responsefunktion $\widehat{R}$ eine lineare Funktion. Diese verfeinern sie mittels

$$\widehat{R}_M = \frac{1}{|P_M|} \sum_{(i,j)\in P_M} e_i E_j$$

mit $P_M = \{(i,j) : y_{i,j} = M\}$. Die E_j bestimmen sie mittels Gleichung 2.12. Sie iterieren über die Kurvenform bis sie eine gewisse Glattheit erreicht. Eine explizite Annahme an die Kurvenform machen sie nicht.

Mitsunaga und Nayar bemängeln in ihrem Paper [*Mitsunaga und Nayar*, 1999], dass es für das beschriebene Verfahren [*Debevec und Malik*, 1997] notwendig ist, die genaue Belichtungszeit zu kennen. Außerdem müssen die Bilder weiterhin rauscharm sein.

Sie stellen ein Verfahren vor, das ohne präzise Kenntnis der Belichtungszeit auskommt und bildfehlertoleranter ist.

Ausgehend von bekannten Responsekurven legen sie zugrunde, dass sich die inverse Responsekurve durch ein Polynom beschreiben ließe:

$$E = g(M) = \sum_{k=0}^{K} c_k M^k$$

Unbekannt sind hier sowohl der Grad K als auch die Koeffizienten c_k.

Sei $R_{j,j+1} = \frac{e_j}{e_{j+1}}$ das Verhältnis zweier Belichtungen zueinander. Dann gilt

$$R_{j,j+1} = \frac{e_j}{e_{j+1}} = \frac{L_i k_i e_j}{L_i k_i e_{j+1}} = \frac{g(M_{i,j})}{g(M_{i,j+1})}$$

Da wir f nicht kennen, ergeben sich durch die Potenzen unendlich viele Lösungen: $R^u_{j,j+1} = \frac{g(M_{i,j})}{g(M_{i,j+1})}^u$. Wir haben dies in Theorem 2.3 gezeigt. Die Autoren versuchen diese Mehrdeutigkeit zu umgehen, indem sie festlegen, dass f ein Polynom bleiben muss. Daher sind viele Exponenten u unzulässig. Weiterhin kann über den Abstand der Lösungen zueinander argumentiert werden, der es bei einer ungefähren Abschätzung der Belichtungszeiten wahrscheinlich macht, dass der Algorithmus zur richtigen Lösung konvergiert.

Wenn wir die Bilder so ordnen, dass $e_q < e_{q+1}$ und damit $0 < R_{q,q+1} < 1$, gilt:

$$\frac{\sum_{k=0}^{K} c_k M^k_{i,j}}{\sum_{k=0}^{K} c_k M^k_{i,j+1}} = R_{j,j+1}$$

Die Minimierung sieht damit folgendermaßen aus:

$$\xi = \sum_{i=1}^{N-1} \sum_{j=1}^{P} \left[\sum_{k=0}^{K} c_k M^k_{ij} - R_{j,j+1} \sum_{k=0}^{K} c_k M^k_{i,j+1} \right]^2 \tag{2.13}$$

Mitsunaga und Nayar skalieren die resultierenden Helligkeitswerte M ins Intervall [0..1] und definieren $g(1) = E_{max}$. Damit verringert sich die Dimension des linearen Gleichungssystems um 1, da

$$c_K = E_{max} - \sum_{k=0}^{K-1} c_k$$

Die Lösung ergibt sich durch Finden der Nullstellen der partiellen Ableitungen:

$$\frac{\delta \xi}{\delta c_k} = 0$$

Setzen wir

$$d_{i,j,k} = M^k_{ij} - R_{j,j+1} M^k_{i,j+1} \tag{2.14}$$

und damit

$$\xi = \sum_{i=1}^{N-1} \sum_{j=1}^{P} \left[\sum_{k=0}^{K} c_k d_{i,j,k} \right]^2$$

so ist folgendes lineares Gleichungssystem zu lösen:

$$\begin{bmatrix} \sum_{i=1}^{N-1} \sum_{j=1}^{P} d_{i,j,0}(d_{i,j,0} - d_{i,j,K}) & \cdots & \sum_{i=1}^{N-1} \sum_{j=1}^{P} d_{i,j,0}(d_{i,j,K-1} - d_{i,j,K}) \\ \vdots & \vdots & \vdots \\ \sum_{i=1}^{N-1} \sum_{j=1}^{P} d_{i,j,K-1}(d_{i,j,0} - d_{i,j,K}) & \cdots & \sum_{i=1}^{N-1} \sum_{j=1}^{P} d_{i,j,K-1}(d_{i,j,K-1} - d_{i,j,K}) \end{bmatrix} * \begin{pmatrix} c_0 \\ \vdots \\ c_{K-1} \end{pmatrix} = \begin{bmatrix} -\sum_{i=1}^{N-1} \sum_{j=1}^{P} d_{i,j,0} d_{i,j,K} \\ \vdots \\ -\sum_{i=1}^{N-1} \sum_{j=1}^{P} d_{i,j,K-1} d_{i,j,K} \end{bmatrix} \quad (2.15)$$

Zur Stabilisierung des Verfahrens schlagen [*Reinhard et al.*, 2005, 141] vor, 2.14 und 2.15 derart abzuändern, dass nicht nur benachbarte Paare innerhalb der Belichtungsreihe sondern alle Paare beim Lösen berücksichtigt werden. Die neue Definition für d lautet dann:

$$d_{i,j,j',k} = M_{ij}^k - R_{j,j'} M_{i,j'}^k$$

Dementsprechend muss die Doppelsumme in 2.13 um die Summe $\sum_{j' \neq j}$ ergänzt werden. Entsprechend in 2.15. Durch diese Optimierung sind zahlreiche $R_{j,j'}^{(r+1)}$ zusätzlich zu bestimmen.

Im Gegensatz zu [*Debevec und Malik*, 1997] müssen hier die Belichtungszeiten nicht exakt bekannt sein. Voraussetzung ist lediglich eine Sortierung der verschiedenen Belichtungen.

Mitsunaga und Nayar schlagen vor, iterativ die $R_{j,j+1}$ zu verfeinern. Als Ausgangswert können dabei die ungefähren Angaben des Aufnahmegerätes benutzt werden:

$$R_{j,j+1}^{(r+1)} = \sum_{i=1}^{N} \frac{\sum_{k=0}^{K} c_k^{(r)} M_{ij}^k}{\sum_{k=0}^{K} c_k^{(r)} M_{i,j+1}^k}$$

Als Konvergenzkriterium wählen die Autoren:

$$\forall M : \left| g^{(r)}(M) - g^{(r-1)}(M) \right| < \epsilon$$

Es bleibt der Grad der Funktion festzulegen. Hier schlagen die Autoren vor, eine obere Grenze für K zu wählen (z.B. $K = 10$) und dann denjenigen Exponenten zu nehmen, für den der Fehler ξ am kleinsten wird. [*Reinhard et al.*, 2005, 141] ergänzen, dass für alle Farbkanäle derselbe Grad gewählt werden sollte. Außerdem schlagen sie vor, die Fehlerfunktion als Kombination der drei Farbkanäle zu formulieren.

Die Festlegung eines maximalen Grades zusammen mit dem Konvergenzkriterium ist insbesondere relevant, da die Berechnung für alle Exponenten k durchgeführt wird. Durch die in [*Reinhard et al.*, 2005, 141] vorgeschlagene Optimierung, alle Verhältnisse $R_{jj'}$ zu betrachten, vervielfacht sich der Berechnungsaufwand. Eine Begrenzung der Laufzeit des Verfahrens ist daher unter Umständen angebracht. Hier ist anzumerken, dass die Kalibrierung ein *einmaliger* Prozess ist und daher normalerweise nicht performanzkritisch.

Eine Gewichtungsfunktion für die Bestimmung der Responsekurve gibt es bei diesem Verfahren nicht, da alle gültigen Pixel verwendet werden. Für die Kombination der Aufnahmen schlagen die Autoren die Verwendung des Signal-Rauschabstandes vor:

$$SNR = E \frac{dM}{dE} \frac{1}{\sigma_K(M)} = \frac{g(M)}{\sigma_K(M) g'(M)}$$

$\sigma_K(M)$ ist die Standardabweichung des Rauschens der Messung. Unter der Annahme, dass dieses unabhängig vom Pixelwert M ist, kann die Gewichtungsfunktion geschrieben werden als:

$$w(Z) = \frac{g(M)}{g'(M)}$$

Wie die anderen Algorithmen bestimmt auch dieser die inverse Responsekurve nur für einen Farbkanal. Um die erhaltenen Responsekurven für Rot, Grün und Blau ins Verhältnis zu setzen, verwenden die Autoren das, in den aufgenommenen Daten vorhandene, Verhältnis: $M_r : M_g : M_b \hat{=} I_r : I_g : I_b$. Die Verhältnisse erhalten sie wiederum durch Minimierung des quadratischen Fehlers über alle Pixelwerte.

Debevec und Malik motivierten ihr Vorgehen physikalisch. **Tsin, Ramesh und Kanade** nehmen als Ausgangsmodell die **statistische Modellierung der Transferfunktion**, also der Funktion, die die verschieden belichteten Aufnahmen ineinander überführt (siehe auch Gleichung 2.6). Sie machen in [*Tsin et al.*, 2001] jeweils eine **statistisch**e Vorhersage für das nächste Bild innerhalb der Kalibrierungsserie, schätzen an diesem dann den Fehler ihres Modells ab und verbessern es gegebenenfalls. In ihrem Paper bestimmen sie die Koeffizienten für die Taylorapproximierte ihres Modells iterativ. Neu ist –neben dem Verfahren– die Berücksichtigung von Gammakorrektur und Weißabgleich. Beide beeinflussen die Responsekurve. Die bisherigen Verfahren setzten voraus, dass beide Parameter bei Kalibrierung und Aufnahme konstant gehalten werden und daher nicht bestimmungsrelevant sind.

Hauptkritikpunkt der Autoren an den zuvor vorgestellten Verfahren ist wieder die Form der Responsekurve. Diese sei in den seltensten Fällen durch ein (niedergradiges) Polynom zu modellieren. Sie berechnen den Fehler der vorgestellten Verfahren [*Debevec und Malik*, 1997] und [*Mitsunaga und Nayar*, 1999]. Diesen geben sie als <0,5% an. Der Fehler ihres Modells liegt darunter (Faktor >5). An [*Mitsunaga und Nayar*, 1999] bemängeln sie, dass mittelstarkes Rauschen und geringe Fehlertoleranz für den Exponenten u oft zur trivialen Lösung führten, also $g^u(z) = a^u E^u$ und $u = 0$.

Die Autoren modellieren Aufnahmerauschen (shot noise) N_s, thermisches Rauschen N_{c1}, Weißabgleich a (beinhaltet auch die Belichtungszeit t) und b sowie Quantisierungsfehler, Verstärkerrauschen und D/A- A/D-Wandlerrauschen als N_{c2}.
Der Pixelwert $\widehat{z}$ ergibt sich damit wie folgt:

$$\widehat{z} = f(aE + N_s + N_{c1} + b) + N_{c2}$$

Die Responsefunktion $g = f^{-1}$ folgt mithilfe einer Taylor-Approximierung (und hat keine fest vorgegebene Form):

$$g(\widehat{z}) \approx aE + N_s + N_{c1} + b + g'(\widehat{z})N_{c2}$$

Ohne Rauschen gilt $g(z) = aE+b$. Das Zero Mean Rauschen ist $N(a, \widehat{z}) = N_s+N_{c1}+g'(\widehat{z})N_{c2}$. Die Varianz des Rauschens ergibt sich damit zu

$$\sigma^2(a, \widehat{z}) = aE + N_s + N_{c1} + b + g'(\widehat{z})N_{c2} \tag{2.16}$$

Die Autoren bilden nun Pools von Pixeln mit ähnlichen Helligkeitswerten $\widehat{z}$: $P(k, \widehat{z}, \epsilon) = \{x : |I_k(x) - \widehat{z}| < \epsilon\}$. Die Responsefunktion g und die Weißabgleichswerte a, b werden iterativ bestimmt unter Hinzunahme je einer weiteren Belichtungsstufe. Die Autoren wählen während des Kalibrierungsvorgangs die Gewichte $\widehat{w}_i$ fest:

$$\widehat{E}^{(j)}(x, k) = \frac{\widehat{g}^{(j)}(\widehat{z}_k(x)) - \widehat{b}_k^{(j)}}{\widehat{a}_k^{(j)}}, k = 1, 2, \ldots, N$$

$$\widehat{E}^{(j)}(x) = \sum_{i=1}^{N} \widehat{w}_i^{(j)} \frac{\widehat{g}^{(j)}(\widehat{z}_i(x)) - \widehat{b}_i^{(j)}}{\widehat{a}_i^{(j)}}$$

Es werden jeweils die übersättigten Pixel aussortiert sowie innerhalb der Pools nur die mittleren Werte verwendet. Mit jedem hinzu genommenen Bild kann mithilfe der Standardabweichung der Fehler und damit die Güte der Abschätzung bestimmt werden. Als Anforderungen an die Kurve stellen die Autoren Glattheit und Monotonie. Sie beginnen mit der ersten Winkelhalbierenden als Ausgangsresponsekurve, die sie dann iterativ verfeinern.

Der mittlere Fehler $\widehat{e}$ ergibt sich aus:

$$\widehat{e}_k^{(j)}(x, \widehat{z}_k) = \widehat{a}_k^{(j)} \widehat{E}^{(j)}(x) + \widehat{b}_k^{(j)} - \widehat{g}^{(j)}(\widehat{z}_k)$$

$$\widehat{e}_k^{(j)}(\widehat{z}_k) = \Phi(\widehat{e}_k^{(j)}(x, \widehat{z}_k), x \in P(k, \widehat{z}, \epsilon))$$

$$\frac{1}{\widehat{\omega}_k^{(j)}} = \widehat{\sigma}_k^{(j)}(\widehat{z}_k) = c \cdot median_{x \in P(k,\widehat{z},\epsilon)} \left| \widehat{e}_k(x, \widehat{z}_k) - \widehat{e}_k^{(j)}(\widehat{z}_k) \right|$$

Φ beschreibt hierbei den mittleren Schätzwert des Pools.

Die Lösung ergibt sich durch Minimieren der aus 2.16 folgende Gleichungen

$$\sigma_k^2(\widehat{z}_k) = \sum_{i \neq k} w_i^2 \sigma^2(a_i, z_i) + (w_k - 1)^2 \sigma^2(a_k, \widehat{z}_k)$$

Als Gewichtungsfunktion für die Kombination der Bilder verwenden die Autoren

$$w_k = \frac{a_k / \sigma(a_k, \widehat{z}_k)}{\sum_i a_i / \sigma(a_i, \widehat{z}_i)}$$

Die Autoren wollen lediglich die Responsefunktion g und die Weißabgleichkoeffizienten a und b bestimmen. Für die Kalibrierung der globalen Beeinträchtigungen verweisen sie auf [*Healey und Kondepudy*, 1994] (Rauschen) und [*Asada et al.*, 1996] (Abschattung). Wie [*Debevec und Malik*, 1997] bestimmen sie lediglich die diskreten Responsewerte für die 256 Helligkeitsstufen und keine Kurve. Ihr Verfahren liefert bessere Ergebnisse als die zuvor beschriebenen Verfahren, wenn der Weißabgleich bei der Kalibrierung nicht konstant bleibt. Das Verfahren beruht darauf, Vorhersagen für das nächste zur Kalibrierung hinzugenommene Bild zu treffen und dabei den Fehler zu bestimmen. Es kann somit gut zur Änderungsdetektion innerhalb einer Szene verwendet werden, da sich Szenenänderungen unabhängig von Änderungen der Beleuchtung und ähnlichem bestimmen lassen[71].

Ein ähnlicher statistischer Ansatz wird in [*Liu und El Gamal*, 2003] für CMOS Sensoren verfolgt. Der Vorteil der verwendeten CMOS-Sensoren ist, dass sie ausgelesen werden und gleichzeitig noch weiter integrieren können (non-destructive). So können die Autoren in $n * \tau$ n verschieden belichtete Bilder innerhalb einer Aufnahme gewinnen.

Mann stellt in [*Mann*, 2000] ebenfalls ein iteratives Verfahren vor, das ohne explizite Modellierung der Responsekurve auskommt. Da die Responsekurve ohne absolute Kalibrierung nicht eindeutig bestimmt werden kann –Skalierung und horizontale Lage sind unbekannt–, reicht es mitunter, die Bilder in einen gemeinsamen (relativen) Raum zu transformieren. Er spricht hierbei von „comparametric equations“. Konkret kann dazu ein Ausgangsbild als Referenz festgelegt und alle anderen Bilder zu dessen Helligkeitswerten in Relation gesetzt werden.

[71] Die Autoren zeigen das Beispiel einer Parkplatzüberwachung, bei dem sie trotz veränderter Tageszeit und veränderter Lichtverhältnisse recht zuverlässig Änderungen detektieren können.

Mann formuliert die Anforderungen, die zu einer vollständigen Abdeckung des Dynamikbereichs führen:

$$\forall(x_0, y_0) \in (x, y), \exists k_i q(x_0, y_0) \begin{cases} 1) k_i q(x_0, y_0) \gg n_{q_i} \\ 2) c_i(q(x_0, y_0)) \gg c_i((1/k_i) f^{-1}(n_{f_i}) \end{cases} \tag{2.17}$$

Die erste Gleichung beschreibt hierbei, dass es mindestens ein Pixel in der Belichtungsreihe gibt, bei dem sich das Signal vom Sensorrauschen (n_{q_i}) abhebt. q beschreibt hierbei die Helligkeit im entsprechenden Pixel. Die zweite Gleichung beschreibt, dass es mindestens ein Pixel gibt, das sich vom Kamerarauschen (n_{f_i}) abhebt. c_i ist hierbei die Gewichtungsfunktion. Mit ihr umfasst dieses Kriterium auch übersättigte Pixel.

In so genannten „**Komparagramme**n“ wird die Helligkeit in einem Bild gegen die korrespondierenden in einem anderen Bild der Belichtungsreihe aufgetragen. Mit dieser Information können die Bilder kombiniert werden, ohne die Responsekurve oder ihre Inverse zu bestimmen. Das Ergebnis ist ein Bild, das alle Helligkeitswerte ausgerichtet an einem Referenzpunkt enthält. Vorteil ist, dass die Lösung eindeutig ist. In [*Mann und Mann*, 2001] wird das Verfahren vertieft. Hier gehen die Autoren ebenfalls auf Fehler ein.

Das genauere Prinzip der Komparagramme ist aus Grafik 2.18 ersichtlich.

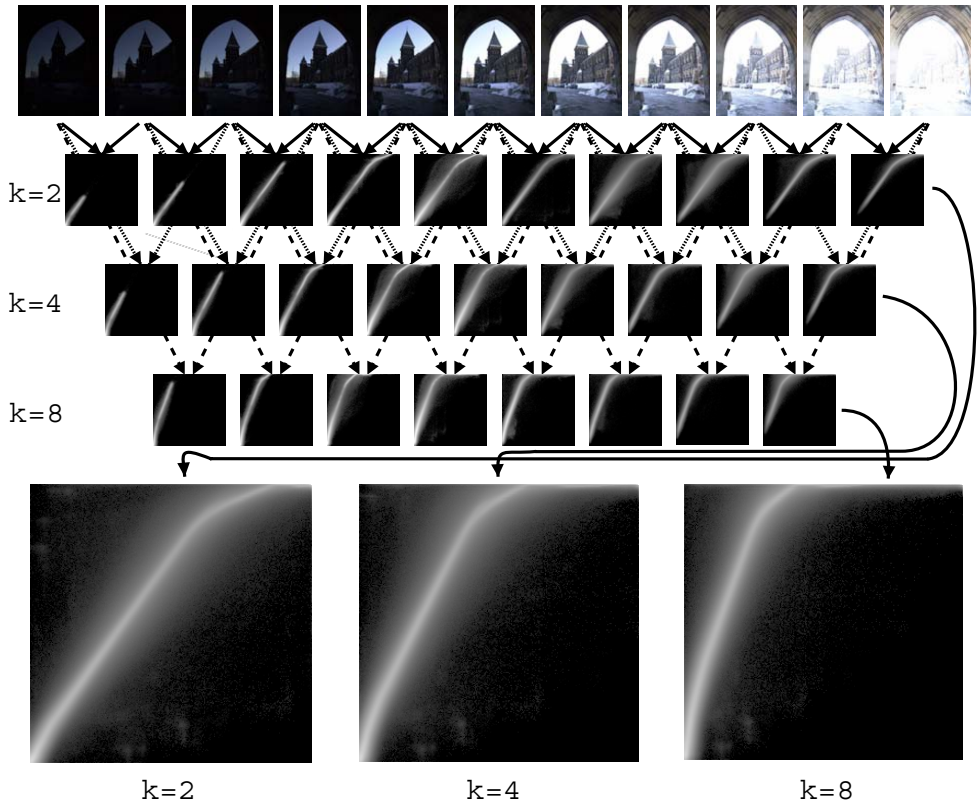


Abbildung 2.18: Das Schaubild aus [*Mann und Mann*, 2001] zeigt die Komparagramme, wie sie bei einer Belichtungsänderung von 1 (k=2), 2 (k=4) und 3 (k=8) zwischen den Bildern entstehen. (Eine Belichtungsstufe führt in der Regel zu einer Verdoppelung der Belichtung.) Die Achsen des Komparagramms sind jeweils die Helligkeitswerte in einem und dem nächsten Bild. Die Intensität gibt an, wie viele Pixel die Helligkeit M_1 in Bild 1 und B_2 in Bild 2 haben. Die unterste Zeile zeigt die zusammengesetzten Komparagramme. Dass die Komparagramme bei unterschiedlicher Schrittweite nicht zur selben Kurve führen, wird sofort einsichtig, wenn man sich ihren Inhalt noch einmal vor Augen führt: Es werden Helligkeiten gegeneinander aufgetragen. Damit ist klar, dass bei weiter auseinander liegenden Belichtungen der Anstieg der Kurve stärker sein muss, da das zweite Bild im Bildpaar viel heller ist als eines mit geringererem Belichtungsabstand.

Barros und Candocia erweitern den komparametrischen Ansatz dadurch, dass sie in [*Barros und Candocia*, 2002] das Komparagramm durch eine abschnittsweise lineare Funktion annähern. In [*Candocia*, 2005] gibt Candocia einen Überblick über den Ansatz:

Gegeben seien zwei $N_1 \times N_2$-Bilder, A und B, mit den Kameraresponsefunktionen f_1 und f_2. Sei $k = \frac{k_A}{k_B}$ und die Pixelwerte auf das halboffene Intervall $[0, 1)$ normalisiert. Dann lässt sich die Komparagrammkurve für die meisten Aufnahmesysteme beschreiben durch

$$f_2(f_1, k) = \left[\frac{k^a \sqrt[c]{f_2}}{\sqrt[c]{f_2}(k^a - 1) + 1} \right]^c$$

Die Werte von f_1, f_2 und k sind hierbei $\in \mathbf{R}$. a und c sind feste Kameraparameter. Diese Kurve wird nun abschnittsweise linear modelliert. In den verschiedenen Papern [*Candocia*, 2003, 2005; *Candocia und Mandarino*, 2005] ändert sich im Wesentlichen die Parametrisierung der linearen Approximation.

Durch die Approximierung sind die Autoren in der Lage, mit deutlich weniger Datenpunkten eine aussagekräftige Kurve zu erhalten. Die Wahl einer abschnittsweise linearen Funktion motivieren Candocia und Mandarino in [*Candocia und Mandarino*, 2005]: Durch geeignete Wahl der Stützstellen lässt sich jede Kurve annähern. Zur deutlichen Abgrenzung sei angemerkt, dass hier die Komparagrammkurve linear angenähert wird und nicht die Responsekurve wie bei Mann und Picard. Die Linearität vereinfacht die Berechnungen innerhalb des Modells sehr stark. Die Modellierung des Komparagramms mittels einer abschnittsweise linearen Funktion erlaubt es, das Problem der Responsekurvenabschätzung direkt als lineares Optimierungsproblem ohne weitere Einschränkungen zu formulieren. In dem referenzierten Paper stellen die Autoren ein Verfahren zur Gewinnung der Responsekurve aus den linear angenäherten Komparagrammen vor.

Ein Problem beim direkten Vergleich von Pixeln ist die notwendige Deckungsgleichheit der Bilder (siehe auch Abschnitt 2.4.1). Die Bilder müssen also sowohl zueinander registriert sein, als auch dasselbe zeigen. Dies ist nicht bei allen Verfahren erforderlich. Grossberg und Nayar schlagen in diversen Papern [*Grossberg und Nayar*, 2002, 2003a,b, 2004] vor, anstelle der Ortsinformationen direkt das **Histogramm**[72] zu verwenden. Vorteil ist, dass das Verfahren toleranter gegenüber Ortsänderungen innerhalb des Bildes wird. Bewegen sich also Objekte, ohne ihre Beleuchtung zu ändern, so verändert dies das Histogramm nicht. In [*Grossberg und Nayar*, 2003a] beschreiben sie ein Verfahren, bei dem sie von kumulativen Histogrammen ausgehen[73]. Mithilfe der Histogrammanalyse stellen die Autoren Komparagramme auf und erhalten so die Transferfunktion zwischen den einzelnen Bildpaaren. Für die Gewinnung der Responsekurve verwenden sie eine leicht modifizierte Version von [*Mitsunaga und Nayar*, 1999], bei der sie die Wertepaare aus dem Histogramm generieren. Ihre Gewichtungsfunktion ist abhängig vom Histogrammwert der entsprechenden Helligkeit, also von der Anzahl der Pixel mit dem entsprechenden Wert. Sie zeigen, dass das Verfahren eine gute Abschätzung der Responsekurve erlaubt und zugleich unabhäng von bestimmten Veränderungen der Szene ist[74].

Die Autoren zeigen noch eine weitere interessante Eigenschaft der Responsekurve: Die Steigung an der Stelle 0 gibt den ungefähren Belichtungswert an. Dies folgt unmittelbar aus Gleichung 2.6:

$$k = \frac{T'(M)g'(T(M))}{g'(M)}$$

[72] Eine Definition für ein Histogramm findet sich in Abschnitt 3.4.1

[73] Ein Eintrag eines kumulativen Histogramms besteht aus der Aufsummierung aller Einträge kleinerer Ordnung im normalen Histogramm. Die Werte eines kumulativen Histogramms sind also monoton steigend. Vorteil des kumulativen Histogramms ist, dass die Beziehung zweier Histogramme zueinander meist eindeutiger ist (Verschiebung auf der x-Achse). Weiterhin bereiten Selbstähnlichkeiten innerhalb des Histogramms keine Probleme.

[74] Solange Korrespondenzen im Histogramm erkennbar bleiben, funktioniert das Verfahren. Eine Verschiebung der Kamera um ein Objekt wird beispielsweise bis zu einem gewissen Grad möglich.

Unter der Annahme, dass $g'(0) \neq 0$ folgt $k = T(0)$. Die Autoren bemerken allerdings, dass es gerade um 0 zu starkem Rauschen kommt und die Abschätzung daher sehr ungenau ausfallen kann.

Wie einleitend geschrieben und aus den Verfahren ersichtlich, ist es schwierig, einen Mittelweg zwischen zu restriktiven Modellen und zu vielen Freiheitsgraden zu finden.

Die Modelle mit einer vorgegebenen Kurvenform begründen dies damit, dass sich die meisten Responsekurven mit ihrem Modell beschreiben ließen. Diesen Gedanken weiterführend, wählen **Grossberg und Nayar** in [*Grossberg und Nayar*, 2003a] einen neuen Ansatz: Sie bestimmen die Responsekurve anhand schon bekannter Responsekurven. Dazu verwenden sie ein Modell mit 201 ihnen bekannten Responsekurven als **Basiskurven, mit deren Hilfe sie jede neue Kurve parametrisieren**.

W_f beschreibt o.B.d.A. den Lösungsraum der Responsekurven

$$W_f := \{f \mid f(0) = 0, f(1) = 1, f' > 0\}$$

In Vektorschreibweise gilt für die P diskreten Helligkeitswerte M:

$$(M_1, \ldots, M_P) = (f(E_1), \ldots, f(E_P))$$

mit Normalisierung von Definitions- und Wertebereich auf $[0, 1]$ folgt

$$(0, E_2, \ldots, E_{P-1}, 1) \xrightarrow{f} (0, M_2, \ldots, M_{P-1}, 1)$$

Die Autoren fahren mit einer etwas schwächeren Formulierung fort, bei der lediglich $M_P = E_P = 1$ gilt:

$$(E_1, E_2, \ldots, E_{P-1}, 1) \xrightarrow{\tilde{f}} (M_1, M_2, \ldots, M_{P-1}, 1)$$

Geometrisch liegen diese Vektoren in der Hyperebene $W_1 := \{(M_1, \ldots, M_P) \mid M_P = 1\}$. Für zwei Responsekurven $\tilde{f}$, $\tilde{f}_0$ gilt $\tilde{f} - \tilde{f}_0 = h$ mit $h \in W_0 := \{(M_1, \ldots, M_P) \mid M_P = 0\}$. Alle möglichen Responsekurven lassen sich also schreiben als

$$\tilde{f} = \tilde{f}_0 + h$$

beziehungsweise

$$W_1 = \tilde{f}_0 + W_0$$

Eine Approximation M-ter Ordnung ergibt sich daher aus

$$\tilde{f}_0(E) + \sum_{n=1}^{M} c_n h_n(E) \tag{2.18}$$

Die c_n sind hierbei die Paramter des Modells. Die h_n beschreiben die verwendeten Basen, die sich aus den 201 Kurven mittels Karhunen-Loève-Transformation (Hauptkomponentenanalyse) ergeben (siehe auch Schaubild 2.19). Als f_0 wählen sie die über alle Responsekurven gemittelte Kurve.

Die Autoren stellen fest, dass sie mit drei Parametern in der Regel eine gute Näherung erreichten.

Die Monotonieeigenschaft von $\tilde{f}$ spiegelt sich in der Konvexität des betrachteten Teiles von W_1 im Modell wider[75] [*Grossberg und Nayar*, 2004]. Bei der Lösungsfindung muss sie erneut berücksichtigt werden, da sich innerhalb des konvexen Hyperebenenteils auch nicht monotone Funktionen bilden lassen.

[75] Der Lösungsraum besteht anschaulich aus dem Schnitt eines (konvexen) Kegels mit der Hyperebene W_1 und ist somit ebenfalls konvex. Für Details siehe [*Grossberg und Nayar*, 2004].

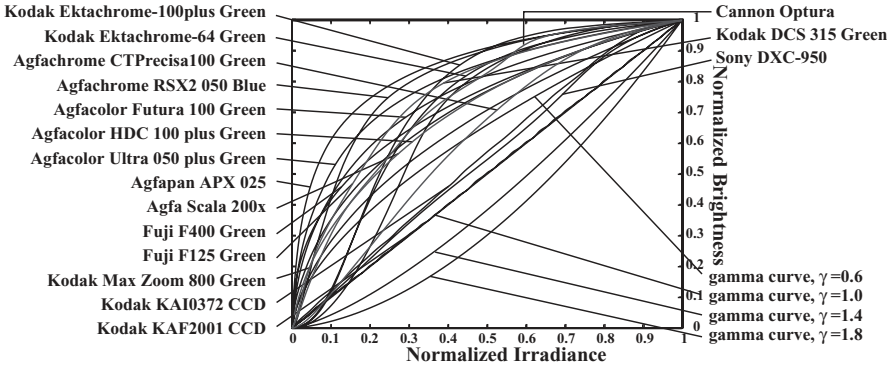


Abbildung 2.19: Eine Auswahl der in [*Grossberg und Nayar*, 2003a] untersuchten Kurven aus denen die Basis für das Modell gewonnen wurde.

In [*Grossberg und Nayar*, 2004] stellen die Autoren ihr Modell auf *logarithmische* Basis. Für diese gilt $\overline{f}_{\log}(1) = 0$, $\lim_{E \to 0} \overline{f}_{\log}(E) = -\infty$. Gleichung 2.18 wird dann im logarithmischen Raum zu

$$\prod_{n=1}^{M} \left(e^{h_{\log,n}(E)}\right)^{c_n}$$

Setzen wir $h_{\log,1}(E) = E$, so erhalten wir bei Parametrisierung mit genau einem Parameter gerade die Gammakurven.

Als Vorteile der Verwendung des logarithmischen Raumes nennen die Autoren Kameradaten, auf die Gammakorrektur angewandt wurde. Im logarithmischen Raum erschöpft sich diese Operation in der Multiplikation mit einer Konstanten.

Nachteilig ist allerdings, dass die Hauptkomponentenanalyse keine so optimalen Basen liefert wie im nicht logarithmischen Fall. Außerdem wirken sich Fehler im Bereich niedriger Bildwerte stärker aus als im linearen Modell.

Die Stärke der Modellierung der Responsekurve mithilfe von Basisfunktionen liegt gerade darin, schon bei wenigen Messpunkten eine gute Responsekurve f zu erhalten[76]. Es wurde also ein Mittelweg zwischen Restriktion und Freiheit gefunden, der vielversprechend ist.

Kim und Pollefeys verwenden das soeben gewonnene Modell in ihrer Arbeit [*Kim und Pollefeys*, 2004]. Sie stellen ein Verfahren vor, um aus nur drei Bildern mit automatischer Belichtung eine Responsekurve zu gewinnen. Automatische Belichtung bedeutet hier, dass die Belichtung und der Weißabgleich unbekannt sind. Außerdem führen sie einen Vorverarbeitungsschritt ein, der aus der Bildsequenz korrespondierende Punkte extrahiert. Dadurch ist es möglich, dass sich die Kamera während der Aufnahme bewegt. Im Wesentlichen setzen sie die drei Bilder paarweise in Beziehung und minimieren den quadratischen Fehler zum Modell.

Pal, Szeliski, Uyttendaele und Jojic berücksichtigen in [*Pal et al.*, 2004] ebenfalls, dass die Responsekurve bei automatischer Belichtung von mehreren Parametern zugleich abhängen kann. Dies ist der Fall, wenn sich der Weißabgleich oder die ISO-Empfindlichkeit[77] innerhalb der Kalibrierungssequenz ändern. Die Autoren modellieren den Aufnahmeprozess nicht statistisch wie in [*Tsin et al.*, 2001] sondern **wahrscheinlichkeitstheoretisch** nach Bayes. Das Modell dient dabei als Generator für die Responsekurve. Die Modellknoten sind die Pixelwerte, in die die Eingabeparameter $f(x)$ und I_x einfließen. Die Autoren modellieren

[76] Bei einem Beispiel der Autoren mit 6 Datenpunkten erreicht ein monotones polynomielles Annähern der Kurve in der Wurzel des mittleren quadratischen Fehlers einen Wert von 0,57, während das beschriebene Verfahren bei 0,11 liegt.

[77] Zur Bedeutung des ISO-Wertes siehe Abschnitt 4.1.

die Glattheit der Funktion innerhalb ihres Modells. Sie verfeinern die Kurve iterativ. Die insgesamt zu maximierende Wahrscheinlicheit ergibt sich wie folgt:

$$\log P\left(\{x_{k,i}\}\right) = \int_{\{r_i\}} \int_{\{f_k\}} \int_{\{v_k\}} p\left(\{x_{k,i}\}, \{r_i\}, \{f_k\}, \{v_k\}\right) \tag{2.19}$$

Ohne genauer auf die Notation einzugehen, sehen wir, dass die Wahrscheinlichkeit von den Pixelwerten $\{x_{k,i}\}$, den zugehörigen Originalhelligkeiten $\{r_i\}$, der Responsefunktion $\{f_k\}$ und der Varianz des Sensorrauschens $\{v_k\}$ abhängt.

Da das Verfahren von unabhängigen Pixeln ausgeht und diese vergleicht, ist eine genaue Registrierung der Bilder zueinander erforderlich (vergleiche Abschnitt 2.4.1). Bei zwei Beispielsequenzen braucht der Algorithmus sechs und 25 Schritte bis zur Konvergenz.

Manders, Aimone und Mann stellen in [*Manders et al.*, 2004] einen neuen Ansatz zur tafellosen Kalibrierung vor. Während die bisherigen Verfahren von gleich bleibender Beleuchtung und wechselnder Belichtung ausgegangen sind, wählen sie den umgekehrten Ansatz: Feste Belichtung und wechselnde Beleuchtung. Das zugrunde liegende Verfahren wurde bereits 1993 in [*Bichsel und Ohnesorge*, 1993] vorgestellt.

Mathematisch nutzen die vorigen Verfahren die Homogenitätseigenschaft des Lichts [*Wyszecki und Stiles*, 1967, 376] bzw. des Sensorchips mit $h = f^{-1} \circ f$:

$$h(ax) = ah(x) \text{ für skalares } a$$

Hier wird nun die Additivität verwendet:

$$h(x + y) = h(x) + h(y) \tag{2.20}$$

Ihre Versuchsanordnung besteht aus zwei Lichtquellen. Es werden drei Aufnahmen gemacht. Eine mit eingeschalteter Lichtquelle A und Beleuchtung E_A, eine mit eingeschalteter Lichtquelle B (E_B) und eine mit beiden Lichtquellen aktiviert (E_C). Die Belichtung ist jeweils dieselbe. Die Annahme ist nun:

$$f^{-1}(f(E_A)) + f^{-1}(f(E_B)) = E_C = f^{-1}(f(E_A + E_B)) + \xi_Q$$

ξ_Q bezeichnet hierbei das Quantisierungsrauschen. Es gilt also folgenden Gleichung zu minimieren:

$$\epsilon = \sum_{j \in J} \left(f^{-1}(f\left(E_A(j)\right)) + f^{-1}(f\left(E_B(j)\right)) - f^{-1}(f\left(E_C(j)\right))\right)^2$$

Das Verfahren geht, wie die meisten vorgestellten Verfahren, von einem sich im Mittel zu Null summierenden Rauschen aus. Die Autoren erstellen in Anlehnung an das Komparagramm (s.o.) ein so genanntes Superposigramm, in dem sie zu gegebenen $f(a)$, $f(b)$ jeweils $f(c)$ auftragen. Die so entstehende Fläche glätten sie vor Rekonstruktion der Response. Die Verfasser schreiben, dass es bei drei Bildern möglich ist, eine Responsekurve zu gewinnen, das Verfahren aber erst bei mehreren Beleuchtungstripeln verlässlich sei.

Unabhängig vom vorigen Paper verfolgen **Shafique und Shah** in [*Shafique und Shah*, 2004] ebenfalls den Ansatz, eine Kalibrierung anhand wechselnder Beleuchtung derselben Szene vorzunehmen. Anders als [*Manders et al.*, 2004] setzen sie keine Annahmen über die Beleuchtung voraus[78]. Die Beleuchtung muss lediglich in den verschiedenen Aufnahmen unterschiedlich sein. Auch die Belichtungszeit muss nicht konstant gehalten werden. Möglich wird dies dadurch, dass sie die Responsekurve als Gammakurve modellieren.

Das Paper fußt auf einem Paper von Narasimhan, Visvanatan und Nayar mit dem Titel „A Class of Photometric Invariants: Separating Material from Shape and Illumination“ [*Narasimhan et al.*, 2003]. Darin versuchen die Autoren, wie auch andere vor ihnen, Invarianten

[78] Es müssen also nicht exakt die drei Messungen mit Lichtquellen A, B und $A + B$ durchgeführt werden.

innerhalb mehrerer Aufnahmen zu finden. Eine solche Invariante muss zum einen „invariant“ sein, sich zum anderen aber auch von anderen Parametern deutlich genug abheben. Sie stellen allgemein die folgende Gleichung für die Beleuchtung auf:

$$L = \sum_{j=1}^{n} K_j(\text{material}) G_j(\text{shape, lighting, viewpoint}) \tag{2.21}$$

Sie zeigen weiter, dass sich sowohl die G- als auch die K-Eigenschaften als Invarianten auffassen lassen und das Modell weiterhin allgemeine Gültigkeit unabhängig vom verwendeten Beleuchtungsmodell (BRDF[79]) hat.

Die Leuchtdichte eines Objektes setzt sich aus Materialkonstanten K_i und Beleuchtungs-/Geometrieeigenschaften G_i zusammen:

$$L = \sum_{a=1}^{r} K_a G_a$$

Die Bestrahlungsstärke folgt, wie wir einleitend schon gesehen haben:

$$E = L \frac{\Pi}{4} (\frac{l}{h})^2 \cos^4 \alpha$$

Die gemessene Helligkeit ergibt sich via $M = f(Et) = f(X)$. Die Modellierung mittels einer Gammakurve lässt sich ausdrücken als $f(X) = \beta X^\gamma + \alpha$. α beschreibt hierbei die Dunkelspannung und kann bei entsprechender initialer Kalibrierung über ein Schwarzbild weggelassen werden. Der Definitions- und Wertebereich wird von den Autoren auf $[0, 1]$ normalisiert, $f(0) = 0$ und $f(1) = 1$. Damit folgt

$$M = f(Et) = (Et)^\gamma$$

Die Inverse ergibt sich somit zu

$$E = g(M) = \frac{1}{t} f^{-1}(M) = \frac{1}{t} M^{\frac{1}{\gamma}}$$

Soweit ist die Theorie gleich dem ersten Paper von Mann und Picard.

Seien nun n Bilder der gleichen Szene unter unterschiedlicher Beleuchtung $I_1, I_2, \ldots, I_n$ gegeben. Bezeichne $p(x, y)$ einen Bildpunkt. Dann ist die Beleuchtung an Punkt p für den Farbkanal $c \in \{r, g, b\}$ des Bildes $I_i, 1 \leq i \leq n$, gegeben durch

$$E_{ic} = \left(\sum_{a=1}^{r} K_a^{(c)} G_a^{(i)} \right) \frac{\Pi}{4} \left(\frac{l_i}{h_i} \right)^2 \cos^4 \alpha_i = F_i \left(\sum_{a=1}^{r} K_a^{(c)} G_a^{(i)} \right)$$

mit $F_i = \frac{\Pi}{4} \left(\frac{l_i}{h_i} \right)^2 \cos^4 \alpha_i$.

Die $G_a^{(i)}$ als Beleuchtungsterme sind für alle Farbkanäle innerhalb eines Bildes konstant, variieren aber von Bild zu Bild. Die Materialkoeffizienten $K_a^{(c)}$ sind über alle Bilder konstant, ändern sich aber pro Farbkanal.

Gemäß [*Narasimhan et al.*, 2003] ergibt sich nun für Bildpaare $i \neq j$ die Invariante

$$\frac{E_{ir} E_{jg} - E_{jr} E_{ig}}{E_{ir} E_{jb} - E_{jr} E_{ib}} = \frac{K_1^{(r)} K_2^{(g)} - K_1^{(g)} K_2^{(r)}}{K_1^{(r)} K_2^{(b)} - K_1^{(b)} K_2^{(r)}}$$

E_{ir} bezeichnet hierbei den Wert des Farbkanals „r“ in Bild i. Wie wir sehen, ist die rechte Seite unabhängig vom gewählten Bildpaar (i, j). Durch zyklisches Vertauschen der r, g, b können zwei weitere Gleichungen der obigen Art aufgestellt werden.

[79] Bidirectional Reflectance Distribution Function, bspw. Lambertian, Torrence-Sparrow, Oren-Nayar, etc.

Durch Einsetzen ergibt sich für die Modellparameter $\mathbf{z} = (\gamma_r, \gamma_g, \gamma_b)$

$$V_{i,j}(\mathbf{z}) = \frac{E_{ir}E_{jg} - E_{jr}E_{ig}}{E_{ir}E_{jb} - E_{jr}E_{ib}} = \frac{M_{ir}^{1/\gamma_r}M_{jg}^{1/\gamma_g} - M_{jr}^{1/\gamma_r}M_{ig}^{1/\gamma_g}}{M_{ir}^{1/\gamma_r}M_{jb}^{1/\gamma_b} - M_{jr}^{1/\gamma_r}M_{ib}^{1/\gamma_b}}$$

Im Verfahren werden nun für alle Bildpaare (i, j) stabile $V_{i,j}(\mathbf{z})$ gesucht. Dazu werden die gewonnenen Histogramme für die $V(\mathbf{z})$, $H(\mathbf{z})$ verglichen und der Fehler minimiert.

Aimone und Mann bemerken in [*Aimone und Mann*, 2007], dass die Beschränkung der Responsekurve auf eine Gammaform sehr restriktiv ist. Dieses Problem haben wir schon mehrfach bei den vorgestellten Verfahren gesehen (siehe auch Abbildung 2.19). Sie erweitern ihr weiter oben vorgestelltes Verfahren [*Manders et al.*, 2004] leicht und beschreiben das Rekonstruktionsschema genauer. Diesmal setzen sie lediglich Semi-Monotonie voraus. Die genaue Belichtung muss nicht bekannt sein, die beiden Lichtquellen müssen jedoch wieder kontrolliert ein- und ausgeschaltet werden (siehe obige Beschreibung von *Manders et al.* [2004]). Zur Stabilisierung des Verfahrens stellen sie eine Methode vor, aus Paaren von Bildern Daten zu interpolieren. Die Autoren schreiben, dass es bei Webcams und ähnlichen einfachen Kameras oft nicht möglich sei, die Belichtung zu kontrollieren. In diesem Fall funktioniert das präsentierte Verfahren dennoch, da es keine genaue Kenntnis der Belichtungszeit erfordert und verschieden belichtete Bilder oft gewonnen werden können, indem man während des Adaptionsprozesses der Kamera an eine neue Beleuchtung[80] verschiedene Aufnahmen macht und diese zur Kalibrierung verwendet.

Eine weitere Serie von Papern verfolgt den Ansatz, die **Responsekurve anhand eines einzigen Bildes** unbekannter Belichtung zu gewinnen. Farid zeigt in seiner hier bereits betrachteten Arbeit [*Farid*, 2001], dass es möglich ist, die Gammakorrektur eines Bildes abzuschätzen und so herauszurechnen. Da die Responsekurve allerdings nicht immer eine Gammakurve ist, wie wir gesehen haben, reicht dieses Verfahren zur Abschätzung der Responsekurve nicht aus.

Lin, Gu, Yamazaki und Shum stellen daher in [*Lin et al.*, 2004] ein Verfahren vor, aus einem einzigen Bild eine Responsekurve nach dem weiter oben vorgestellten Modell [*Grossberg und Nayar*, 2003a] zu bestimmen. Dazu betrachten die Autoren Kanten innerhalb des Bildes. Die räumliche Auflösung eines Sensors ist begrenzt. Bei einer zu den Sensorelementen schräg stehenden Kante fällt die integrierte Helligkeit pro Farbkanal linear zwischen der einen und der anderen angrenzenden Fläche ab (siehe Grafik 2.20). Durch die nicht lineare Kameraresponse wird der lineare Abfall jedoch nicht linear abgebildet. Die Responsefunktion für diesen Bereich kann gewonnen werden, indem die nicht linear zusammenhängenden Pixel der Ecken in linearen Zusammenhang gebracht werden. Siehe Abbildung 2.20. Grundlegend für das Verfahren ist die Verschiebung der Punkte im dreidimensionalen Farbraum. Für den eindimensionalen Raum der Grauwertbilder muss das Verfahren abgeändert werden. Diesen Fall betrachten wir nachfolgend.

Die Autoren suchen die Kanten mittels des Canny-Operators. Anschließend werden die Kanten auf ausreichende Helligkeitsunterschiede untersucht sowie in nicht überlappende Fenster der Größe 15x15 aufgeteilt. Es können verfahrensgemäß nur diejenigen Bereiche der Responsekurve gewonnen werden, für die innerhalb der betrachteten Ecken Werte vorliegen. Die Autoren verwenden die gewonnenen Erkenntnisse als Basis für das von uns weiter oben betrachtete Modell [*Grossberg und Nayar*, 2003a], bei dem die inverse Responsekurve mittels vorher festgelegter Basisfunktionen beschrieben wird. Liegen mehrere mit einer Kamera erstellte Aufnahmen vor, bei denen die Kanten insgesamt größere Helligkeitsbereiche abdecken, so wird das Verfahren genauer.

Lin und Zhang modifizieren das soeben vorgestellte Verfahren in [*Lin und Zhang*, 2005] für die Verwendung auf Grauwertbildern. Da hier keine Verschiebung der Farbwerte im dreidimensionalen Raum erfolgt, greifen die Autoren stattdessen auf lokale Histogramme zurück.

[80] Beispielsweise könnte in einem dunklen Raum das Licht angeschaltet werden.

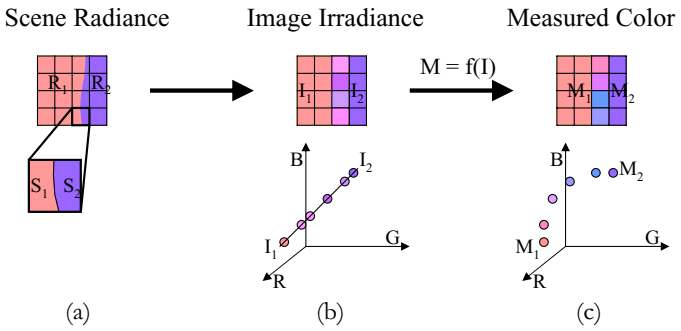


Abbildung 2.20: Das Gitter stellt die räumliche Auflösung des Sensors dar. Aus der Lage der Kante in Bild (a) folgt, dass diese von unten nach oben linear von R_1 nach R_2 verläuft. Bei einer direkten Abbildung des linearen Farbverlaufs würde das Bild (b) entsprechen. Durch die nicht lineare Kameraresponse erhalten wir jedoch Bild (c). Daher kann die Kameraresponsefunktion an solchen Kanten bestimmt werden, indem eine Abbildung der Messwerte auf eine Gerade gesucht wird. [*Lin et al.*, 2004]

Siehe Grafik 2.21. An verschiedenen Kanten können so verschiedene Stücke der Response-

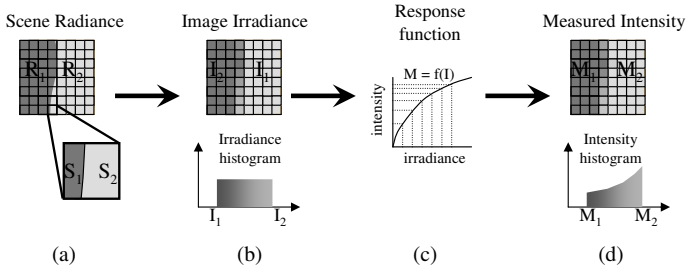


Abbildung 2.21: Das Gitter stellt die räumliche Auflösung des Sensors dar. Aus der Lage der Kante in Bild (a) folgt, dass diese von unten nach oben linear von I_1 nach I_2 verläuft. Bei einer direkten Abbildung des linearen Helligkeitsverlaufs würde das Histogramm (b) entsprechen. Durch die nicht lineare Kameraresponse (c) erhalten wir jedoch Histogramm (d). Daher kann die Kameraresponsefunktion an solchen Kanten bestimmt werden, indem eine Abbildung des nicht einheitlichen Histogrammes auf das gleichverteilte gesucht wird. [*Lin und Zhang*, 2005]

kurve gewonnen werden. Die einzelnen Abschnitte werden von den Autoren gemäß der zur Rekonstruktion verwendeten Messwerte gewichtet. Die Rekonstruktion erfolgt, wie zuvor für [*Lin et al.*, 2004] beschrieben.

Die Autoren schließen ihr Paper mit dem Hinweis, dass die Methode auch für Farbilder eingesetzt werden kann, indem jeder Kanal einzeln als Grauwertbild betrachtet wird.

Ng, Chang und Tsui präsentieren in [*Ng et al.*, 2007] ebenfalls ein Verfahren zur Kalibrierung mittels eines einzigen Bildes. Zu den soeben vorgestellten Verfahren, die auf gemessenen Farb- bzw. Helligkeitsänderungen an Kanten arbeiten, schreiben sie, dass es schwierig sei, geeignete Kanten zu finden bzw. die gemachten Annahmen über die Einheitlichkeit der Abstrahlung von der Kante nicht immer zutreffend seien. Daher versuchen sie geeignete (ebene) Kalibrierungsregionen innerhalb des Bildes zu identifizieren, indem sie geometrische Invarianten (siehe auch Beschreibung zu [*Shafique und Shah*, 2004] weiter oben bzw. [*Narasimhan et al.*, 2003]) finden. Zur Rekonstruktion verwenden sie die erste und zweite Ableitung der Kameraresponsefunktion $R = f(x)$:

$$DR(x,y) = (R_x R_y) = f'(r)\,(r_x r_y)\,, r = g(R)$$

$f'(r)$ hängt nur von der Responsefunktion ab, r_x und r_y von der Helligkeitsverteilung innerhalb des Bildes.

Stellt man r mittels der Taylorapproximierten dar, so beschreibt der erste Term gerade die Ebenen. Die zweite Richtungsableitung ergibt sich daher zu $r_{xx} = r_{xy} = r_{yy} = 0$. Damit gilt:

$$D^2R(x,y) = \begin{pmatrix} R_{xx} & R_{xy} \\ R_{yx} & R_{yy} \end{pmatrix} = f''(r) \begin{pmatrix} r_x^2 & r_x r_y \\ r_y r_x & r_y^2 \end{pmatrix}$$

Die erste geometrische Invariante G_1 ergibt sich aus dem Verhältnis:

$$\frac{R_{xx}}{R_x^2} = \frac{R_{yy}}{R_y^2} = \frac{R_{xy}}{R_x R_y} = \frac{f''(f^{-1}(R))}{(f'(f^{-1}(R)))^2} = G_1(R)$$

Das Verfahren arbeitet gerade auf solchen Ebenen (Locally Planar Irradiance Points) innerhalb des Bildes. Die Rekonstruktion erfolgt, ähnlich wie im vorigen Paper, über die Korrelation der in der Realität linearen Änderung und deren Transformierten innerhalb des Bildes. Die Interpolation erfolgt ebenfalls wieder über das Basiskurvenmodell aus 201 echten Responsekurven von Grossberg und Nayar [*Grossberg und Nayar*, 2003a]. Für Kurvenformen, die von der ersten Winkelhalbierenden (linear) abweichen, liefert das Verfahren in den Experimenten der Autoren für die verwendeten fünf Kameras gute Ergebnisse.

Matsushita und Lin präsentieren in [*Matsushita und Lin*, 2007] einen weiteren neuen Ansatz: Sie bestimmen die Repsonsekurve über die **Verteilung des Bildrauschens** (siehe auch Abschnitt 2.2.2.1 zu möglichen Rauschquellen). Die Autoren führen an, dass alle betrachteten Rauschquellen poissonverteilt bzw. zero-mean sind und mit einer (symmetrischen) Gaußkurve angenähert werden können. Das Rauschen wird aus der Veränderung der Pixel über mehrere Aufnahmen mit identischer Szene und Kameraeinstellung gewonnen. (Die Autoren verwenden Videodaten.)

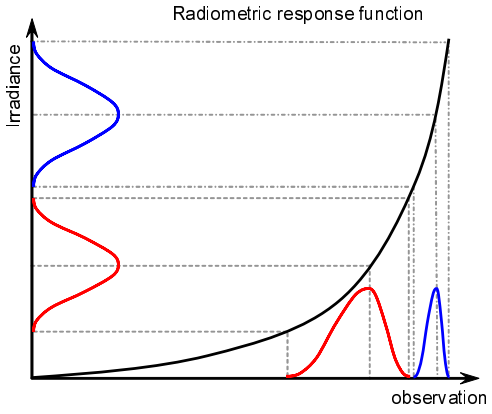


Abbildung 2.22: Die Ordinate zeigt die symmetrische Rauschverteilung des Sensors. Die Abszisse zeigt die durch die Nachverarbeitung veränderte Verteilung. Durch die eingezeichnete Korrelation ergibt sich eine Responsekurve [*Matsushita und Lin*, 2007].

Für die Gewinnung der inversen Responsekurve verwenden die Autoren ebenfalls das Modell von Grossberg und Nayar um die Kurvenform einzuschränken. Die Autoren schlagen abschließend vor, ihr Verfahren mit Standardkalibrierungsverfahren zu kombinieren, um die Daten noch effektiver auszunutzen.

2.3.4 Diskussion und Zusammenfassung der Verfahren

Auf den vorangegangenen Seiten haben wir einen sehr großen Teil[81] der zu dem Thema „Gewinnung der Kameraresponsekurve“ veröffentlichten Arbeiten betrachtet.

Die ersten (und meisten der vorgestellten Arbeiten) beruhen auf der einleitend vorgestellten Gleichung 2.3:

$$g(M_B) = kg(M_A)$$

Sie setzen verschieden belichtete Bilder zueinander in Beziehung und bestimmen aus dem Verhältnis zur jeweils gleichen Originalhelligkeit die Kameraresponsefunktion.

Die Verfahren unterscheiden sich dabei darin, welche *Parameter* als Veränderliche angesehen werden. Wichtigster Vertreter ist hierbei die Belichtungszeit. Manche der Verfahren erlauben auch veränderlichen Weißabgleich, veränderliche ISO-Einstellung[82] oder gar vollständig automatische Bildkontrolle, die dann zusätzlich die Gammakorrektur und die Gainspannung regelt. Auch die Registrierung der Bilder zueinander[83], also deren Deckungsgleichheit, wird nicht von allen Verfahren vorausgesetzt. Sie ist nicht notwendig, wenn wir auf Histogrammen und Pixelpools arbeiten oder explizite Featurepunkte in den verschiedenen Aufnahmen identifizieren.

Wir haben außerdem eine Evolution in der angenommenen Kurvenform gesehen. Wie in Abschnitt 2.3.2 gezeigt, sind Einschränkungen an die Kurvenform unvermeidbar, um zu einem Ergebnis zu kommen, da das Problem sonst unterbestimmt ist. Die Freiheitsgrade an die Kurve reichen in den betrachteten Verfahren von einer Gamma-Kurve bis hin zu Modellen, die keine Einschränkungen an die Kurvenform legen. Erstere Form ist zu restriktiv und für viele Modelle nicht zutreffend. Letztere (Frei-)Form ist sehr anfällig gegenüber Fehlern wie Rauschen. Grossberg und Nayar haben in [*Grossberg und Nayar*, 2003a] eine Parametrisierung für Responsekurven zur Verfügung gestellt, die aufgrund der Verwendung in einigen nachfolgenden Papern für heutige Kameras gute Ergebnisse zu liefern scheint. Durch die Verwendung von 201 echten Responsekurven zur Erstellung der Basen des erzeugten Raumes, ist das Modell flexibler als ein polynomielles Modell und kommt gleichzeitig oft mit deutlich weniger Parametern aus.

Wir haben in diesem Teil ebenfalls ein statistisches und ein wahrscheinlichkeitstheoretisches Verfahren betrachtet. Ersteres ist interessant, da die Idee dahinter ist, den Aufnahmeprozess nachzumodellieren. Bei den vorausgehenden Verfahren wurde weniger Wert auf den Prozess der Bildgewinnung als mehr auf die Verknüpfung von Ein- und Ausgabe gelegt.

Mit den Arbeiten von Mann und Candocia haben wir Verfahren betrachtet, die Bilder zueinander zu registrieren, ohne explizit die Responsekurve auszurechnen. Vorteil dieses Vorgehens ist die Verminderung der Unbekannten.

Für unser Ziel, die Bilder mehrerer fest montierter Kameras mit wenig Überlappung zusammen zu bauen, brauchen wir allerdings eine absolute Kalibrierung. Wie wir gesehen haben, geht dies nur mithilfe von Kalibrierungstafeln, da Bezugspunkte zur Originalhelligkeit sonst fehlen. Die gewonnenen Kurven sind also in ihrer Lage und auch Skalierung nicht eindeutig bestimmbar. Als „absolut“ genügt uns hier, dass für alle Kameras die gleiche Responsefunktion zugrunde gelegt wird und diese qualitativ stimmige Ergebnisse liefert (vergleiche Abschnitt 4.1.5). Sollten für unsere Aufnahme eines Panoramas verschiedene Kameramodelle zum Einsatz kommen, erfordert dies eine Kalibrierung der Kurven zueinander, damit die Bilder sich beim Zusammenfügen in ihren Helligkeiten entsprechen (vergleiche Abschnitt 3.6).

[81] Es wurden alle referenzierten Arbeiten, sowie die über ACM, IEEE und Citeseer auffindbaren Arbeiten zum Thema berücksichtigt. Die Betrachtung sollte also fast vollständig sein.
[82] Siehe Abschnitt 4.1.
[83] Vergleiche Abschnitt 2.4.1.

Einen neuen Ansatz stellte die Verwendung von unterschiedlicher Beleuchtung anstelle unterschiedlicher Belichtung dar. Dieser Ansatz beruht auf der Additivität des Lichts, die wir in Gleichung 2.20 betrachtet haben:

$$h(x + y) = h(x) + h(y)$$

Vorteil ist, dass wir die genaue Belichtung nicht kennen müssen sondern lediglich zwei Lichtquellen kontrolliert an- und ausschalten bzw. beim zweiten Verfahren unterschiedlich beleuchtete Aufnahmen zur Verfügung stellen. Dieses Verfahren macht sich die Möglichkeit der Trennung von Geometrie- und Beleuchtungseigenschaften zu Nutze.

Auf ähnlicher Grundlage bauen auch die Kalibrierungsverfahren aus einer einzigen Aufnahme auf. Sie suchen nach einheitlich gefärbten Objekten innerhalb des Bildes und verwenden deren unterschiedliche Beleuchtung. Auf diese Weise erhalten sie ähnliche Daten wie aus einer Mehrfachbelichtung. Die oft nicht ausreichende Datenbasis versuchen die Verfahren durch das Kurvenmodell von Grossberg und Nayar auszugleichen.

Das neuste Verfahren, das wir betrachtet haben, beschreibt einen indirekten Ansatz: Es verwendet die unsymmetrische Verteilung des Bildrauschens zur Detektion der Responsekurve.

2.4 Beseitigung von HDR-Bildfehlern

In Abschnitt 2.2.5 haben wir gesehen, wie sich LDR-Aufnahmen zu HDRIs kombinieren lassen. Durch die Kombination mehrerer Aufnahmen können spezifische Probleme –und daraus resultierend Bildfehler– entstehen. In diesem Teil wollen wir die wichtigsten betrachten.

2.4.1 Ausrichtung der Bilder zueinander

Die pixelgenaue Ausrichtung der LDR-Bilder zueinander ist für eine Rekonstruktion der HDR-Werte notwendig. Wie in Abschnitt 2.2.5 beschrieben, sind die HDR-Pixelwerte eine gewichtete Kombination der LDR-Pixel M_j:

$$\widehat{I}_j = \frac{\sum_i \omega_{ij} M_{ij}}{\sum_i \omega_{ij}}$$

Sind die Bilder nicht pixelgenau zueinander registriert, zeigt ein Pixel in Bild A einen anderen Bildausschnitt als das gleiche Pixel in Bild B. Somit werden falsche Helligkeitswerte kombiniert, da gilt: $M_{A,j} \not\sim M_{B,j}$. Daher ist eine Ausrichtung der LDRIs für die Kombination zu einem scharfen HDR-Bild notwendig.

Wie wir gesehen haben, beruhen auch einige der Verfahren zur Bestimmung der Responsekurve auf Pixelkorrelationen (siehe Abschnitt 2.3.3). Diese Verfahren benötigen ebenfalls aneinander ausgerichtete LDR-Basisbilder.

Es sind zahlreiche Algorithmen bekannt, Bilder deckungsgleich anzuordnen. In Kapitel 3 gehen wir genauer auf einige davon ein. Viele funktionieren mit unseren Bildsequenzen nicht. Das Problem sind die Helligkeitsunterschiede und die dadurch möglicherweise verschiedenen Featurepunkte (siehe hierzu auch Abschnitt 3.4) in heller oder dunkler belichteten Teilen der Aufnahme. Eine Ausrichtung daran wird somit schwierig. Kantenbasierte Verfahren scheiden damit beispielsweise zumeist aus, da sie die Kanten nicht gleichmäßig auf allen Bildern der Belichtungsreihe detektieren können. *Reinhard et al.* [2005] schreiben von zwei geeigneten Verfahren: [*Kang et al.*, 2003] und [*Ward*, 2003].

Kang et al. [2003] beschäftigen sich mit HDR-Video, das sie aus alternierenden hellen und dunklen LDR-Aufnahmen gewinnen. Sie schätzen die Bewegung innerhalb benachbarter Bilder mithilfe des Lucas-Kanade-Verfahrens[84] [*Lucas und Kanade*, 1981] ab. Anschließend warpen[85] sie diese zueinander. Wir betrachten den Ansatz in Abschnitt 3.3.2.6 genauer. Das Verfahren ist relativ rechenaufwendig[86]. Da das Lucas-Kanade-Verfahren auf Intensitäten beruht und diese sich zwischen den verschiedenen Belichtungen ändern, müssen die Ausgangsbilder zuerst in einen gemeinsamen Raum transformiert werden. Das Verfahren eignet sich daher nicht zur Gewinnung der Responsekurve, da diese gerade für die Transformation benötigt wird. Verfahren, die auf dem optischen Fluss –also der Verschiebung einzelner Helligkeitsintensitäten zwischen Bildern– basieren und komplexe Korrelationen zwischen den Aufnahmen suchen, sind für die einfachen Bildverschiebungen (s.u.), mit denen wir es hier in der Regel zu tun haben, zu aufwendig.

Tomaszewska und Mantiuk [2007] präsentieren ein ähnliches Verfahren. Sie verwenden einen modifizierten Scale-Invariant-Feature-Transform-Algorithmus (siehe Abschnitt 3.4.1), um Verschiebungen innerhalb der Sequenz zu bestimmen. Das SIFT-Verfahren wurde dabei modifiziert, um mit den verschiedenen Belichtungen arbeiten zu können. Im Vergleich zu dem

[84] Es wird der optische Fluss, also die räumliche Verschiebung von Pixelintensitäten bestimmt.

[85] Als Warping bezeichnet man die digitale Manipulation eines Bildes, insbesondere auch den Vorgang, Bildinformationen aus verschiedenen Perspektiven in ein gemeinsames Koordinatensystem zu transformieren. Bspw. [*Trucco und Verri*, 1998; *Chen und Williams*, 1993; *McMillan und Bishop*, 1995]

[86] Die Nachbearbeitung eines Frames benötigt auf einem Pentium4–2GHz 10 Sekunden. 8 Sekunden für die Registrierung und die Erstellung der HDR-Radiancemap und 2 Sekunden für das Tonemapping.

anschließend vorgestellten Verfahren ist es zu aufwendig für den an dieser Stelle verfolgten Zweck[87].

Das zweite in [*Reinhard et al.*, 2005] vorgeschlagene Verfahren ist wesentlich einfacher. Es beschränkt sich auf horizontale und vertikale Ausrichtung. Die Autoren schreiben, dass dies in der Regel ausreicht. Es ist die **Mean Threshhold Bitmap Alignment**-Methode [*Ward*, 2003]. Diese operiert auf Grauwertbildern[88]. Zuerst wird die mittlere Helligkeit aus dem Grauwerthistogramm bestimmt. Als Nächstes werden alle Pixel größerer Helligkeit auf 1 gesetzt, alle anderen auf 0. So erhalten wir eine binäre Bitmap, die Median Threshold Bitmap (MTB)[89]. Eine solche MTB ist für zwei verschieden belichtete Aufnahmen in Abbildung 2.23 dargestellt. Ein XOR über mehrere MTBs gibt die Ausrichtungsfehler aus. Die Ausrichtung

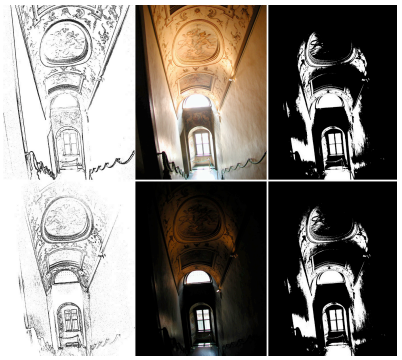


Abbildung 2.23: In der Mitte sehen wir zwei verschieden belichtete Aufnahmen eines Treppenhauses. Links sehen wir die detektierten Kanten. Wie wir sehen, verändern sie sich stark zwischen den verschiedenen Belichtungen. Rechts sehen wir die jeweilige Mean Threshhold Bitmap. Diese ist unter Belichtungsänderung stabil. [*Ward*, 2003]

kann nun entlang eines Gradienten erfolgen. Dadurch kann ein lokales Minimum gefunden werden. Der Autor schlägt die Verwendung von MTB-Bildpyramiden vor, um das globale Minimum zu finden. Die Ausrichtung beginnt bei der geringsten Auflösungsstufe auf eine Genauigkeit von +/- 1 Pixel. Bei der nächsten Auflösungsstufe entspricht dieses zwei Pixeln. Es wird wieder auf ein Pixel genau ausgerichtet. In der nächsten Stufe entspricht dies wieder zwei Pixeln... So verfeinern wir das Ergebnis bis zur vollen Auflösungsstufe. Ward schreibt, dass der Aufwand in den meisten Fällen mit denen des Gradientenverfahrens vergleichbar sei. Probleme ergeben sich, wenn Rauschen an Pixelwerten nahe der Trennschwelle auftritt, da diese Pixel einmal 1 und einmal 0 sein können. Der Autor schlägt hier vor, einen bestimmten Bereich um die Trennschwelle auszumaskieren.

2.4.2 Geister (ghosts)

Wie wir einleitend zum Abschnitt Bildregistrierung schon betrachtet haben, ergeben sich Fehler, wenn gleiche Pixel innerhalb der LDR-Sequenz unterschiedliche Bildinhalte zeigen. Dies ist nicht nur bei fehlregistrierten Bildern der Fall sondern auch, wenn sich Objekte innerhalb der Bildes bewegen. Es entstehen so genannte Geister (ghosts) (siehe Abbildung 2.24).

Da die entsprechenden Objekte nun nur in Teilen der LDR-Aufnehmensequenz vorhanden sind, erscheinen sie, entsprechend gewichtet, als mehr oder weniger durchsichtige Schatten.

[87] Die Autoren schreiben, dass es ca. 450 Sekunden dauert, drei Bilder auszurichten.

[88] Das Grauwertbild lässt sich in guter Näherung durch die Ganzzahloperation $Y = (54R + 183G + 19B)/256$ bestimmen. ($/256 \simeq >> 8$). Der Autor schreibt, dass es keinen signifikanten Unterschied zwischen der Bestimmung der Grauwertes und dem Verwenden des Grünsignals gab und daher die rechnerisch günstigere Näherung Y=G verwendet werden kann.

[89] Konstruktionsbedingt ist dieses Verfahren auf alle Stufen der LDR-Sequenz anwendbar. Bei fast saturierten Bildern kann eine Verschiebung der Grenze nach unten bzw. oben sinnvoll werden.

Abbildung 2.24: Da die Personen bei jeder LDR-Aufnahme an einer anderen Stelle im Bild laufen, entstehen Geisterbilder [*Khan et al.*, 2006].

Es gibt mehrere Möglichkeiten, dem entgegenzuwirken, wenngleich sich nur wenige Veröffentlichungen zur Problematik der Geisterbeseitigung finden[90].

Uyttendaele et al. [2001] beschäftigen sich mit dem Problem für Panoramamosaike. Für diesen Fall reicht es aus, die betroffenen Bereiche (Region Of Difference) zu identifizieren und konsistent aus einem Bild zu verwenden. Es entsteht in der Regel kein Detailverlust bei dieser LDR-Bildverarbeitung.

Das in [*Reinhard et al.*, 2005, 147ff] beschriebene Verfahren geht ähnlich vor. Die Autoren identifizieren die betroffenen Stellen und ersetzen sie durch Werte einer einzigen Belichtungsstufe. Da das HDR gerade durch Kombination mehrerer Aufnahmen an derselben Stelle zustande kommt, geht zumeist in diesen Bereichen ein Teil der HDR-Information verloren. Dafür ist das Bild konsistent. Bei geeigneter Wahl der Ersetzung ist der Verlust gering. Problematisch ist die Wahl der zu ersetzenden Gebiete. Im ungünstigen Fall würden beispielsweise nur Teile eines Objektes ersetzt und dessen Darstellung dadurch inhomogen. Dies versucht man zu verhindern. Die Autoren wenden dazu mehrere Verfahren an, um zusammenhängende Gebiete mit einer gewissen Mindestgröße zu erhalten. Einzelne ersetzte Pixel werden dadurch ebenfalls vermieden. Dennoch kann es zu „Fehlern“ wie abgeschnittenen Beinen kommen.

Jacobs et al. [2005] erweitern das Verfahren zur Detektion der betroffenen Regionen. Es werden zwei Arten von Bewegung identifiziert: High Contrast Movement (HCM) und Low Contrast Movement (LCM). Letztere waren für das Ursprungsverfahren schwer zu finden. Die gefundenen Gebiete werden jeweils mit Daten aus einem einzigen LDR gefüllt.

Khan et al. [2006] gehen einleitend kurz auf Image-Warping-Verfahren der im letzten Abschnitt und später in Abschnitt 3.3.2.6 genauer beschriebenen Art (*Kang et al.* [2003]) ein, da diese bei erfolgreichem Warping ebenfalls Geisterbilder entfernen. Wie oben schon gesehen, funktioniert das Warping nur für geringe Änderungen. Durch das Bild laufende Personen beispielsweise kann es nicht entfernen.

In ihrem Paper stellen *Khan et al.* [2006] ein Verfahren vor, das sich direkt in den HDR-Generierungsprozess integrieren lässt. Der Extraschritt, betroffene Regionen ausfindig zu machen, entfällt also. In ihrem Algorithmus klassifizieren sie die Pixel kontinuierlich in Vordergrund und Hintergrund. Ihre Annahme ist, dass die Mehrzahl der LDR-Sequenzpixel „Hintergrundpixel“ sind und über die Sequenz das Gleiche zeigen. Außerdem bestimmen sie ein Maß für die Güte der Belichtung des aktuellen Pixels. Durch die Klassifikation wird es möglich, gezielt nur die „gestörten“ Pixel auszusortieren. Je nach Anzahl dieser wird eine deutlich höhere Dynamik erreicht, da nicht nur eine Belichtungsstufe für den Geisterbereich verwendet wird sondern alle, die konsistent sind, Berücksichtigung finden.

[90] Diese Aussage ist Ergebnis einer Suche bei ACM, IEEE, Citeseer, Google-Scholar Anfang des Jahres 2008.

2.5 Zusammenfassung

In diesem Kapitel haben wir uns mit der HDR-Bildaufnahme befasst.

Zuerst haben wir den Dynamikumfang verschiedener Szenen und Aufnahmeformen betrachtet. Vertiefend sind wir auf die radio-/ photo- und colorimetrischen Grundlagen sowie auf unsere visuelle Wahrnehmung eingegangen. Anschließend haben wir uns dem digitalen Aufnahmeprozess mithilfe von CCD- und CMOS-Sensoren gewidmet. Ein wichtiger Aspekt dabei waren mögliche Fehlerquellen. Zum Abschluss der Einleitung sind wir in der Aufnahmepipeline einen Schritt zurück gegangen und haben diejenigen Helligkeitsaberrationen im Linsensystem betrachtet, die für dieses Kapitel relevant sind. Auf weitere Linsenaberrationen werden wir in Abschnitt 3.2.1 eingehen.

Bezüglich der HDR-Aufnahme haben wir uns kurz mit Hardwareverfahren beschäftigt und sind dann über die Hybridverfahren zu den reinen Softwarelösungen gelangt, die ohne Veränderungen an den Aufnahmesensoren auskommen.

Der Hauptteil des Kapitels umfasste eine ausgiebige Betrachtung der Forschungsergebnisse zum Schlüssel der HDR-Gewinnung aus einer LDR-Belichtungsreihe: der Responsekurvenbestimmung mithilfe von Standardaufnahmen[91]. Wir konnten dabei sowohl die historische Entwicklung als auch die mathematischen und physikalischen Grundlagen nachvollziehen. Insbesondere haben wir die wesentlichen algorithmischen Merkmale aller Verfahren gesehen und für die größte Gruppe der auf Pixelkorrelation basierenden Verfahren sogar mathematisch einige Eigenschaften bewiesen. In der chronologischen Betrachtung haben wir gesehen, dass es nach einigen Jahren Forschung in dieselbe Richtung auch zu neuen vielversprechenden Ansätzen kam.

Den Abschluss des Kapitels bildete die Betrachtung eines effizienten Algorithmus zur Registrierung einzelner Aufnahmen der Belichtungsreihe zueinander, sowie die Vorstellung verschiedener Ansätze zum Entfernen von Geisterbildern. Beides sind Probleme, die sich aus der vorgestellten Verarbeitung einer LDR-Belichtungsreihe zu einem HDR ergeben.

Dass der vorgeschlagene Weg, qualitativ relativ schlechte Webcambilder als Ausgangsmaterial für die HDR-Gewinnung zu verwenden, erfolgversprechend ist, zeigt Abbildung 2.26. Wir sehen das Ergebnis eines ersten Tests. Die Parameter sind lediglich grob justiert worden und dennoch ist das Ergebnis bereits vielversprechend.

[91] Einleitend haben wir gesehen, wie sich mithilfe dieser Responsefunktion die LDR-Bilder zu einem HDRI kombinieren lassen.

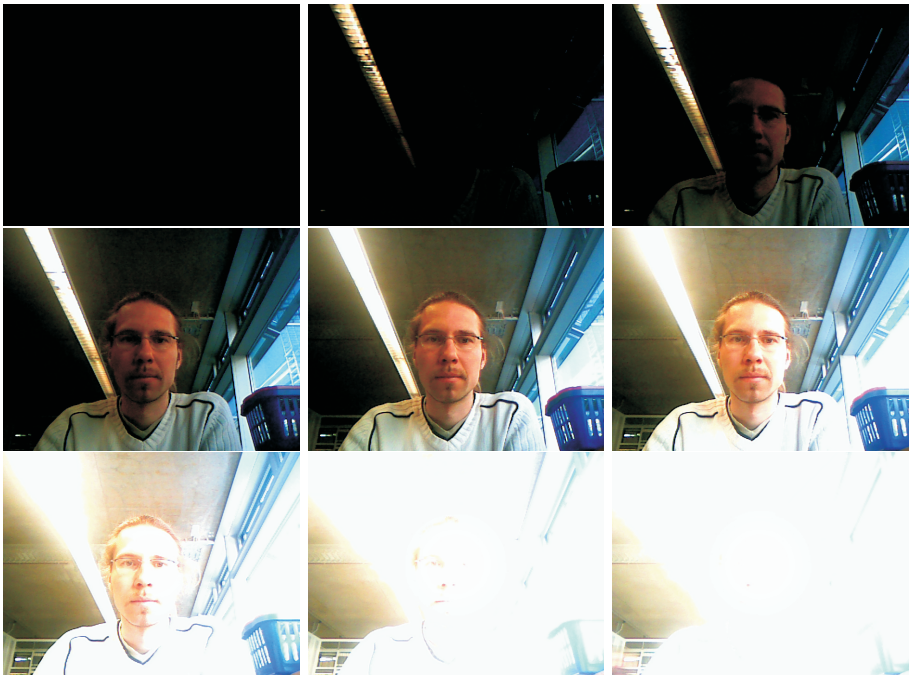


Abbildung 2.25: Quell-LDR-Sequenz einer Webcam.

Abbildung 2.26: Tonegemapptes HDR (Mapping-Operator: Mantiuk, contrast mapping=0.3, saturation factor=0). Wie wir sehen, zeigt das Bild erheblich mehr Details als die einzelnen Aufnahmen oben: Sowohl das Regal links im Hintergrund als auch die Details außerhalb des Fensters sind erkennbar. Die Lamellen in der Lampe sind ebenfalls nicht überbelichtet.

3. Stitching von Bildern

> Jetzt wächst zusammen, was zusammen gehört!
>
> Willy Brandt am Abend des 10. November 1989 vor dem Schöneberger Rathaus

Im letzten Kapitel haben wir uns damit beschäftigt, die Helligkeitsdynamik der Aufnahme zu erhöhen. In diesem Kapitel geht es darum, den abgebildeten Ausschnitt der Realität räumlich zu erweitern. Wir gehen auf Methoden zum Zusammenfügen von Bildern zu einem (360°-)Panorama ein, das so genannte Stitching. Das Hauptaugenmerk liegt dabei auf der Behandlung von Parallaxe –also Abbildungsinkonsistenzen, die dadurch entstehen, dass nicht alle Teilbilder vom selben Punkt aufgenommen werden– und Verfahren zur Featuredetektion. Darunter verstehen wir die Detektion von markanten Punkten. Die von den Algorithmen erkannten Details können anschließend zur Registrierung verschiedener Bilder zueinander genutzt werden.

3.1 Motivation

Unser Ziel ist es, ein HDR-Panorama zu erstellen.

Das Wort Panorama setzt sich aus den griechischen Wörtern $\pi\hat{\alpha}\varsigma$ (pas = ganz, alles) und $o\text{'}\rho\acute{\alpha}\omega$ (horao = sehen, hórama = das Geschaute) zusammen. Es bezeichnet einen Rundblick. In der Malerei wurde der Begriff Ende des 18. Jahrhunderts von Johann Adam Breysig und Robert Barker eingeführt. Barker malte Rundumblicke, die dann beispielsweise vom Inneren eines Zylinders betrachtet werden konnten. In der Fotografie bezeichnet der Begriff heute eine sehr weitwinklige Aufnahme bis hin zur 360°-Rundumsicht. [*Brockhaus*, 2006b]

Wie wir schon in der Einleitung in Abschnitt 1.3 gehört haben, gibt es verschiedene Methoden ein Panorama zu erstellen: Pilzspiegel, die einen großen Raumwinkel widerspiegeln, extreme Weitwinkellinsen (Fischauge) und eine rotierende Kamera seien hier erneut aufgeführt.

Aus verschiedenen Gründen haben wir uns für den Ansatz entschieden, mehrere Kameras zu verwenden:

- Der Aufstellungsort muss *keine Rundumsicht* von einem Punkt aus bieten. Die Geräte können beispielsweise in die Fenster eines Turmes montiert werden.
- Wir benötigen *keine Mechanik*, wie sie beispielsweise zum Drehen einer Kamera notwendig wäre.
- Gegenüber den Ansätzen, die die Optik der Kamera modifizieren, um den Blickwinkel zu erweitern, sind wir in der Lage eine *höhere Auflösung* zu erzielen, da wir in jeder Teilaufnahme fast die gesamte Kameraauflösung verwenden können.
- Wir können dadurch (kostengünstigere) *Hardware mit geringerer Auflösung* verwenden als es beispielsweise bei einem Pilzspiegel notwendig wäre, um ausreichend Details aus dem verzerrten Bild zu erhalten.

Durch das gewählte Verfahren zur Aufnahme müssen wir die Einzelbilder zu einem Gesamtpanorama zusammensetzen. Diesen Vorgang bezeichnet man als **Stitching**. Die dazu notwendigen Schritte, die wir in diesem Kapitel betrachten, umfassen:

- Vorbereitung der einzelnen Aufnahmen, um beispielsweise die Linsenverzerrung herauszurechnen
- Registrierung der Aufnahmen zueinander
- Zusammenfügen der einzelnen Aufnahmen zu einem Gesamtbild

3.2 Geometrische Entzerrung

3.2.1 Aberrationen im Linsensystem – Teil 2

In Abschnitt 2.2.2.2 haben wir gesehen, welche Abbildungsfehler in Bezug auf Helligkeit (natürlicher Randlichtabfall/ Vignettierung) und Farbe (chromatische Aberrationen) durch die Verwendung einer Linse entstehen können. Hier betrachten wir Schärfe- und Geometriefehler, da diese insbesondere Einfluss auf die in diesem Kapitel betrachteten Verfahren haben[92]: Sie erschweren das Identifizieren von Bildkorrespondenzen und das eigentliche Zusammenfügen der Bilder.

Die hier betrachteten Abbildungsfehler wurden schon 1850 von Ludwig Seidel veröffentlicht[93]. Die folgende Summe beschreibt die fünf klassischen Seidel-Aberrationen [*Britannica*, 2008]:

$$OPD = S_1 \left(x^2 + y^2\right)^2 + S_2 y \left(x^2 + y^2\right) h + S_3 \left(x^2 + 3y^2\right) h^2 + S_4 \left(x^2 + y^2\right) h^2 + S_5 y h^3 \quad (3.1)$$

OPD steht hierbei für „Optical Path Difference“, also die Abweichung eines Strahls von der Bahn, die er bei einer perfekten Linse nehmen würde. Die $S_1, \ldots, S_5$ sind Konstanten[94] der jeweiligen Aberration. x, y sind die Koordinaten in Bezug auf den 0-Punkt im optischen Zentrum der Linse. Die y-Achse ist hierbei die Meridionalebene (vgl. Abb. 3.3). h ist der Abstand des Punktes auf der Bildebene zur optischen Achse.

Die fünf Summanden beschreiben die folgenden fünf Abbildungsfehler (in der angegebenen Reihenfolge) mathematisch [*Britannica*, 2008; *Bergmann und Schaefer*, 2004b; *Hecht*, 2005]:

- Schärfefehler
  - **Sphärische Aberration** tritt auf, wenn mehrere von einem einzigen Objekt ausgehende Lichtstrahlen von der Linse nicht im gleichen Punkt auf der Abbildungsebene gebündelt werden.

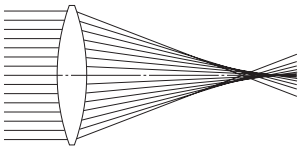


Abbildung 3.1: Erste Seidel-Aberration (S_1): Die sphärische Aberration. Die Lichtstrahlen eines Szenenpunktes werden nicht in einem Bildpunkt gebündelt. [*Canon Inc.*, 2006]

  - **Koma** (von lat. coma, der Schweif) tritt auf, wenn ein Objekt mehrfach auf die Abbildungsebene geworfen wird. Meist haben die verschiedenen Abbildungen des Objektes unterschiedliche Größe. Das Objekt scheint einen Schweif nach sich zu ziehen.

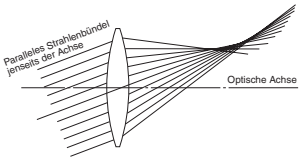


Abbildung 3.2: Zweite Seidel-Aberration (S_2): Koma. Die diagonalen Lichtstrahlen sind nicht auf einen Punkt der Bildebene fokussiert. Es entsteht ein Schweif. [*Canon Inc.*, 2006]

[92] Erklärende Applets zu den hier genannten Aberrationen finden sich hier: http://www.olympusmicro.com/primer/anatomy/aberrationhome.html (Besucht am 10.3.2008).

[93] Bereits vor Seidel untersuchte der ungarische Mathematiker Joseph Petzval die Eigenschaften von Linsen. Seine Arbeiten wurden jedoch vor Veröffentlichung zerstört [*Born und Wolf*, 2000].

[94] Die genauen Formeln der Koeffizienten und ihre Berechnung finden sich beispielsweise in [*Britannica*, 2008].

– Eine Linse lässt sich in zwei Ebenen aufteilen: Die Meridionalebene und die Sagittalebene. Bei Lichtstrahlen von Objekten, die sich nicht in der optischen Achse befinden, definiert sich die Meridionalebene durch die Gerade zwischen Objekt und Linsenmittelpunkt und die optische Achse. Sie enthält also die optische Achse. Die Sagittalebene ist dazu senkrecht und beinhaltet ebenfalls den Objekt- und den Linsenmittelpunkt. Als **Astigmatismus** bezeichnet man es nun, wenn sich die Brennpunkte der beiden Ebenen nicht decken. Der Brennpunkt der Meridionalebene (enthält optische Achse) liegt dabei vor demjenigen der Sagittalebene.

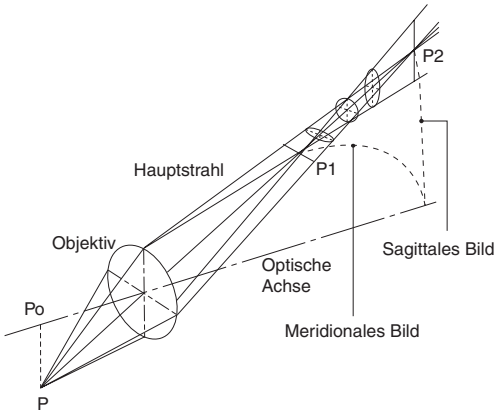


Abbildung 3.3: Dritte Seidel-Aberration (S_3): Der Astigmatismus. Das Bild des Meridionalschnittes (mit optischer Achse) und des dazu senkrechten Sagitalschnittes decken sich nicht. Die gestrichelten Linien zeigen an, wie meridionales und sagittales Bild immer näher zusammenrücken, je näher der Punkt der optischen Achse kommt. [*Canon Inc.*, 2006]

- Geometriefehler

  – Die von der Linse gebrochenen Strahlen einer linsenparallelen Ebene werden eventuell nicht in einer Ebene sondern in einer gewölbten Fläche gebündelt. Dieses Phänomen heißt **Bildfeldwölbung** oder **Petzval-Krümmung**. Die Ursache ist in Grafik 3.4 veranschaulicht: Je weiter die Punkte der Szene von der optischen Achse entfernt sind, desto weiter in Richtung Linse liegen ihre Brennpunkte, da ihre Strahlen entsprechend der Grafik gebrochen werden. Wenn eine Linse eine starke Petzval-Krümmung aufweist, ist das Bild nur an den Schnittpunkten von Bildebene und der gekrümmten Kurve/ Fläche scharf. Da das projizierte Bild auf der gekrümmten Fläche scharf abgebildet wird, liegt kein Schärfefehler vor. Dadurch, dass der Film oder der Chip nicht in dieser Projektionsfläche liegen, entsteht dennoch Unschärfe.

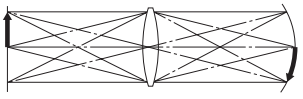


Abbildung 3.4: Die Grafik veranschaulicht die Petzval-Krümmung der Fläche, auf der die Brennpunkte liegen (S_4). Objekte, die links auf einer Geraden liegen, werden rechts auf eine Kurve abgebildet. [*Canon Inc.*, 2006]

  – **Verzeichnung** bezeichnet eine Änderung des Abbildungsmaßstabes in Abhängigkeit vom Abstand des Bildpunktes zur optischen Achse. Der Vergrößerungsfaktor der Linse kann dabei zum Rand hin kleiner bzw. größer werden. Die Verzeichnung wird auch als optische Verzerrung bezeichnet. Sie ist radialsymmetrisch.

    Verzeichnung führt dazu, dass alle Geraden, die nicht durch die optische Achse verlaufen, gewölbt werden. Es gibt kissen- und tonnenförmige Verzeichnung.

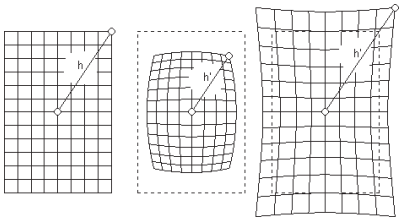


Abbildung 3.5: Original, kissenförmige Verzeichnung, tonnenförmige Verzeichnung. Die Verzeichnung hängt jeweils von der Höhe über der optischen Achse ab. Wir sehen an den Linien, dass die Verzeichnung umso stärker ist, je weiter wir vom Zentrum weg sind. Wie wir in Summe 3.1 sehen (S_5), ist der Zusammenhang h_0^3. [Grafik: http://www.vanwalree.com/optics/distortion.html]

Die Schärfefehler stellen für die Featurepunktdetektion ein Problem dar, da eventuell Features auf unscharfen Bildern nicht identifizierbar sind (vergleiche Abschnitt 3.4). Bei „normalen“ Objektiven sind die Schärfefehler so gering, dass wir sie für unsere Anwendung vernachlässigen können. Die abgedruckten Beispiele sind extreme Darstellungen der jeweiligen Aberration zur Veranschaulichung.

Interessanter und gravierender sind für uns die geometrischen Abbildungsfehler. Die Bildwölbung fällt bei entsprechend gebauten Linsen ebenfalls gering aus. Die Verzeichnung stellt allerdings ein Problem dar: Selbst wenn eine Registrierung der Bilder gelingt (siehe Abschnitt 3.4), sind die Übergänge zwischen den einzelnen Teilbildern unseres Panoramas an der geometrischen Verzerrung zu den Übergängen (Wellen innerhalb der Bilder) deutlich erkennbar. Dies wollen wir nicht. Wir müssen die Verzeichnung also korrigieren.

3.2.2 Korrektur der Verzeichnung

Die optische Verzerrung eines Objektivs ist umso größer, je größer sein abgebildeter Raumwinkel ist. Zur Veranschaulichung betrachten wir Grafik 3.6.

Abbildung 3.6: Auf der linken Seite sehen wir den schematischen Aufbau eines 16mm-Fischaugenobjektivs. Der Strahlgang von stark am Abbildungsrand liegenden Lichtstrahlen ist eingezeichnet. Er ergibt sich aus Form und Brechungseigenschaft der jeweiligen Linse innerhalb des Objektivs. In der Mitte sehen wir das simulierte Ergebnis. Die tonnenförmige Verzeichnung ist vor allem an der Decke sehr gut zu erkennen. Das Bild auf der rechten Seite ist eine reale Aufnahme mit deutlich sichtbarer tonnenförmiger Verzeichnung. Links/ Mitte: [*Kolb et al.*, 1995], Rechts: [*Swaminathan und Nayar*, 2000]

Ein Ansatz zum Beschreiben und Korrigieren der radialsymmetrischen Verzeichnung ist das folgende Modell. Es hängt nur vom Parameter κ ab [*Farid und Popescu*, 2001]:

$$f_u(x,y) \overset{\text{distortion}}{\longrightarrow} f_d(\widetilde{x},\widetilde{y}) \tag{3.2}$$

$$\widetilde{x} = x(1+\kappa r^2) \quad \text{und} \quad \widetilde{y} = y(1+\kappa r^2) \tag{3.3}$$

Es gilt $r^2 = x^2 + y^2$. κ gibt die Stärke der Verzeichnung an.

Mit diesem einparametrigen mathematischen Modell lassen sich für unsere Zwecke bereits gute Ergebnisse erzielen.

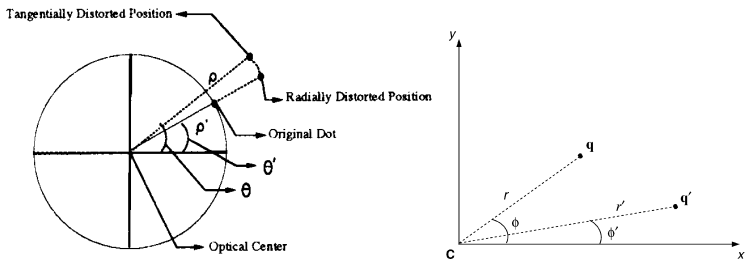


Abbildung 3.7: Die Parameter der Verzeichnung. Links: [*Shah und Aggarwal*, 1994], Rechts: [*Swaminathan und Nayar*, 2000]

In Abbildung 3.7 links sind die Modellparameter eingezeichnet.

Swaminathan und Nayar [2000] beschreiben die Verzeichnung mit folgendem Modell höherer Ordnung:

$$\Delta r(q) = \sum_{i=1}^{\infty} C_{2i+1} r^{2i+1} \text{ mit } r = \sqrt{\bar{x}^2 + \bar{y}^2}, \tan(\Phi) = \frac{\bar{y}}{\bar{x}}$$

Die Parameter sind aus Grafik 3.7 rechts ersichtlich: q' ist der Ort des Punktes bei einer idealen Abbildung, q der aus dem Abbildungsfehler resultierende Punkt („Tangentially distorted point“) in Polarkoordinaten (r, Θ), Δr entspricht $|\rho - \rho'|$ (linke Abbildung) bzw. $|r - r'|$ (rechte Abbildung). Die C_{2i+1} sind die Parameter.

Die Autoren entwickeln das Polynom lediglich bis zum Grad 5:

$$\Delta r(q) \approx C_3 r^3 + C_5 r^5$$

Sie schreiben, dass dies eine ausreichende Näherung sei.

Zusätzlich modellieren sie eine Verschiebung des optischen Zentrums. Die bisher vorgestellten Formeln setzen voraus, dass das optische Zentrum in der Mitte der Linse liegt. Bei nicht ideal symmetrischen Linsen muss diese Annahme nicht stimmen. In diesem Fall ist die folgende Korrektur notwendig:

$$\begin{aligned}
\Delta T_x(q) &= \left[P_1 r^2 \left(1 + 2\cos^2(\Phi)\right) + 2P_2 r^2 \sin(\Phi)\cos(\Phi)\right] \cdot \left[1 + \sum_{i=1}^{\infty} P_{i+2} r^{2i}\right] \\
\Delta T_y(q) &= \left[P_2 r^2 \left(1 + 2\sin^2(\Phi)\right) + 2P_1 r^2 \sin(\Phi)\cos(\Phi)\right] \cdot \left[1 + \sum_{i=1}^{\infty} P_{i+2} r^{2i}\right]
\end{aligned}$$

Vereinfacht ergibt sich hier:

$$\begin{aligned}
\Delta T_x(q) &\approx \left[P_1 r^2 \left(1 + 2\cos^2(\Phi)\right) + 2P_2 r^2 \sin(\Phi) cos(\Phi)\right] \\
\Delta T_y(q) &\approx \left[P_2 r^2 \left(1 + 2\sin^2(\Phi)\right) + 2P_1 r^2 \sin(\Phi) cos(\Phi)\right]
\end{aligned}$$

Die gesamte Korrektur der Verzeichnung ergibt sich damit aus der folgenden Formel:

$$\begin{aligned}
\Delta x(q) &\approx \cos(\Phi)\left(\Delta r(q)\right) + \Delta T_x(q) \\
\Delta y(q) &\approx \sin(\Phi)\left(\Delta r(q)\right) + \Delta T_y(q)
\end{aligned}$$

Die zu bestimmenden charakteristischen Parameter der Linse sind:

$$S = \{C_3, C_5, P_1, P_2, x_p, y_p\}$$

(x_p, y_p) bezeichnet hierbei das optische Zentrum.

Für die Kalibrierung der unbekannten Parameter gibt es mehrere Möglichkeiten. Eine Übersicht über Kalibrierungsverfahren findet sich in [*Devernay und Faugeras*, 2001] und [*Swaminathan und Nayar*, 2000].

Swaminathan und Nayar [2000] und *Devernay und Faugeras* [2001] extrahieren Punkte, die in der Realität auf einer Geraden liegen. Im Bild werden diese durch die Verzeichnung auf eine Kurve abgebildet. Sie bestimmen die Parameter S nun durch Fehlerminimierung so, dass die Punkte im Abbild ebenfalls auf einer Geraden liegen. Ihr Verfahren nutzt die Kenntnis von Geometrieeigenschaften (Gerade) der abgebildeten Gegenstände zur Kalibrierung.

Shah und Aggarwal [1994] verwenden eine Kalibrierungstafel mit einem Gitter aus Punkten. Sie nutzen ebenfalls das Wissen aus, dass jeweils mehrere Gitterpunkte auf einer Geraden liegen und minimieren mittels Lagrange-Iteration den Fehler bezüglich aller Punkte.

Verfahren, die besondere Kalibrierungsanordnungen verwenden, bestimmen oft sowohl die inneren (Brennweite, Längenverhältnis, Verzeichnung etc.) als auch die äußeren (Rotation, Translation) Kameraparameter zugleich. *Devernay und Faugeras* [2001] schreiben, dass es dabei zu großen Fehlern bei den inneren Kameraparametern kommen kann.

Farid und Popescu [2001] arbeiten ohne spezielle Kalibrierungsgegenstände. Sie untersuchen das Frequenzspektrum des verzeichneten Bildes[95]. Durch eine Bispektralanalyse stellen sie fest, dass eine Verzeichnung Korrelationen höherer Ordnung im Signal erzeugt. Die Stärke dieser Korrelationen ist proportional zur Verzeichnungsintensität und ermöglicht daher die Bestimmung des Parameters κ in Modell 3.3. Vorteil an ihrem Verfahren ist, dass es keine besonderen Voraussetzungen an den Bildinhalt stellt und daher prinzipiell auf alle Bilder nachträglich anwendbar ist.

Die bisher vorgestellten Verfahren sind iterativ. Ihre Laufzeit ist also nicht im Voraus bestimmbar[96]. *Fitzgibbon* [2001] stellt einen linearen Algorithmus vor. Voraussetzung für sein Verfahren sind mehrere Bilder des gleichen Motivs von unterschiedlichen Standpunkten. In den jeweiligen Bildern werden Featurepunkte detektiert (vgl. Abschnitt 3.4), anhand derer die gesuchten Parameter bestimmt werden.

Sind die Parameter zum Entfernen der Verzeichnung bekannt, so müssen die Pixel entsprechend verschoben werden. Da die Verschiebung zumeist nicht im Ganzzahlbereich liegen wird, müssen die Farbwerte der neuen Pixel interpoliert werden. Dies kann mittels bekannter Verfahren wie (bi-)cubischer, (bi-/ tri-)linearer etc. Interpolation erfolgen.

Wir haben nun einige Ansätze zur Korrektur der optischen Verzerrung betrachtet. Im nächsten Abschnitt können wir daher davon ausgehen, dass wir Quellbilder ohne optische Verzerrung vorliegen haben.

[95] Vergleiche auch das Paper zur „Blind inverse gamma correction“ [*Farid*, 2001] in Abschnitt 2.3.3.

[96] Iterative Algorithmen terminieren oftmals über eine Abbruchbedingung. Eine solche Abbruchbedingung ist beispielsweise, dass der Fehler unter eine ϵ-Schranke fällt.

3.3 Registrieren und Transformieren der Einzelbilder zu einem Panorama

In diesem Kapitel ist es unser Ziel, aus mehreren Aufnahmen ein Panorama zu erstellen. Um die Einzelbilder kombinieren zu können, müssen wir sie an den Überlappungsstellen zur Deckung bringen. Dazu müssen wir bestimmen, welche Pixel in den zusammenzufügenden Bildern homolog sind, also die gleichen Bildinhalte zeigen – wir müssen die *Korrespondenzen* finden.

Wir betrachten zuerst eine ideale Aufnahmesituation und erweitern die Betrachtung schrittweise, bis wir bei einem für uns realistischen Aufbau ankommen.

3.3.1 Der Idealfall

Abbildung 3.8: Die Abbildung zeigt die Aufnahme eines Panoramas mithilfe einer um das optische Zentrum des Objektivs rotierten Kamera. [reallyrightstuff.com]

Idealerweise werden die einzelnen Aufnahmen für unser Panorama erstellt, indem die Kamera über demjenigen Punkt gedreht wird, an dem sich der Mittelpunkt der Linse befindet. So werden alle Weltpunkte durch eine Kameradrehung zwar verschoben, perspektivisch aber gleich abgebildet – die Objekte verändern also nur ihre Lage auf dem Abbild. Da heutige Objektive aus mehreren Linsen bestehen, ist das optische Zentrum ein bestimmter Punkt innerhalb des Objektivs. In Abbildung 3.9 ist dieser beispielhaft eingezeichnet.

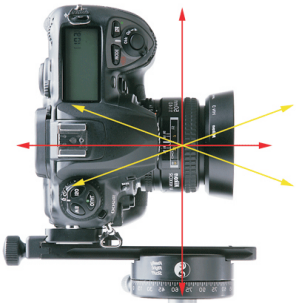


Abbildung 3.9: Eine solche Vorrichtung kann benutzt werden, um die Kamera genau am optischen Zentrum der Linse zu drehen. Dadurch entstehen keine Parallaxeeffekte in der fertigen Aufnahme (vergleiche Abschnitt 3.3.2.5). [reallyrightstuff.com]

Für den Idealfall, dass die Drehachse durch das Zentrum der Linse geht und die Lage der Kamera sich während der Aufnahme nicht in der Höhe verändert –das optische Zentrum des Objektivs also im selben Punkt verweilt–, ist die Aufgabe der Korrespondenzenfindung einfach: Durch die Drehung der Kamera werden die Bildpunkte lediglich verschoben. Wir müssen die Aufnahmen daher nur so horizontal zueinander verschieben, dass die Objekte sich in den verschiedenen Aufnahmen decken. Korrespondierende Bildpunkte haben in allen Aufnahmen

dieselbe y-Koordinate. (Die Ermittlung des Fehlerminimums bei der Kreuzkorrelation (siehe Abschnitt 3.3.2.3) einer Zeile kann beispielsweise den Verschiebungsparameter bestimmen.) Sind die Drehwinkel zwischen den Aufnahmen gleich, so ist der ermittelte Parameter sogar für jedes Bildpaar gleich. Sind die Bilder zueinander registriert, also passend aufeinander gelegt, so können wir die Bildpunkte perspektivisch in eine gemeinsame Bildfläche abbilden. Das Zusammenfügen von Einzelbildern bezeichnet man als **Stitching**:

In Abbildung 3.10 ist dies für die Zylinderprojektion aufgezeigt. Diese Projektionsart eignet sich besonders für 360°-Panoramen, da sie geschlossen ist. Eine ebenfalls häufig eingesetzte

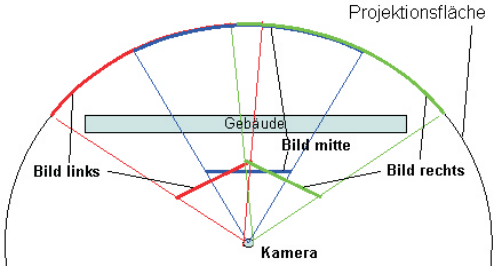


Abbildung 3.10: Eine der am häufigsten für Panoramabilder verwendeten Projektionen ist die *Zylinderprojektion*. Gespeichert wird das abgerollte Kreissegment, das hier mit „Projektionsfläche" bezeichnet ist. Wie abgebildet werden die zueinander registrierten Bildflächen auf den Zylinder projiziert. [panoclub.de]

Projektion ist die rechtwinklige Position. Während Geraden bei der Zylinderprojektion auf Kurven abgebildet werden, bleiben sie bei der rechtwinkligen Projektion gerade.

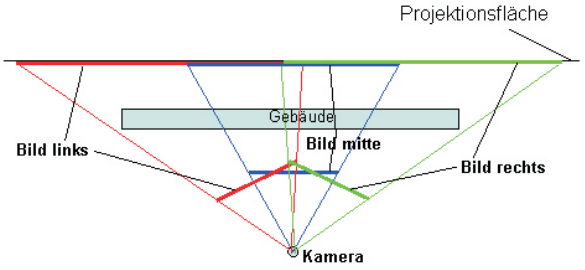


Abbildung 3.11: Bei der *rechtwinkligen Projektion* werden die Bilder wie aufgezeigt projiziert. Ein Schließen des Panoramas ist damit nicht möglich. Wie wir sehen, werden die Verzerrungen der Quellbilder zum Rand der Projektionsfläche hin deutlich größer als bei der Zylinderprojektion. [panoclub.de]

Leider widerspricht der skizzierte ideale Aufbau unseren Anforderungen: Es ist schwierig, mehrere Kameras so zu montieren, dass ihre optischen Zentren im selben Punkt sind[97]. Darüber hinaus haben wir sogar eine dezentrale Montage der Kameras gefordert. Die Annahme eines gemeinsamen Linsenzentrums wird daher in jedem Fall nicht zutreffend sein.

Durch die Betrachtung des idealisierten Falles haben wir uns die grundlegenden Verarbeitungsschritte vor Augen geführt:

- Korrespondenzenfindung
- (perspektivische) Transformation der Einzelaufnahmen zu einem Gesamtbild (Stitching)
- gegebenenfalls Überblendung der Teilaufnahmen zueinander (siehe Abschnitt 3.6)

Für den, im Folgenden entwickelten, Realfall ist das grundsätzliche Vorgehen gleich; lediglich die Wege zum Erreichen des Ziels sind etwas weiter...

[97] Dazu müssten sie ein gemeinsames Objektiv verwenden.

3.3.2 Die Realität

In unserer Anordnung liegt das optische Zentrum der verschiedenen Kameraobjektive nicht im selben Punkt. Soweit nicht anders beschrieben, gehen wir im Folgenden von zwei Kameras aus. Dieser Fall lässt sich leicht auf n Kameras erweitern. Wir verallgemeinern unseren Idealfall nun schrittweise dahingehend, dass die zweite Kamera sich beliebig im Raum befinden kann.

Wir setzen zunächst voraus, dass das abgebildete Objekt invariant gegenüber einer Perspektivenänderung ist.

Um ein Verständnis für die, durch die Erweiterung entstandene, Problematik zu bekommen, ist in Grafik 3.12 eine äquivalente Voraussetzung visualisiert: Das Objekt ist nicht invariant unter Perspektivenänderung, dafür bleibt die Aufnahmeposition fest.

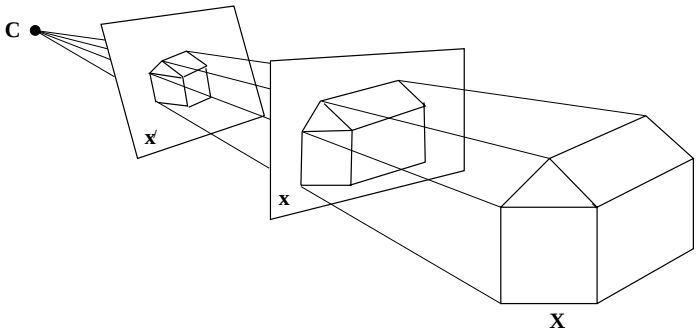


Abbildung 3.12: In der Grafik ist erkennbar, wie sich die Geometrie des Gebäudes verändert, wenn die Bildfläche (Kamera) entsprechend gedreht wird. Die von uns zunächst geforderte perspektivische Invarianz wird dadurch erreicht, dass lediglich die Bildebene verschoben und gedreht wird, das Projektionszentrum aber gleich bleibt. [Folien zur Vorlesung „Multiple View Geometry“ von Richard Hartley und Andrew Zisserman im Juni 1999]

Abbildung 3.12 zeigt, wie stark sich das Abbild bereits durch Drehung und Verschiebung der Projektionsfläche verändert. Die beiden Abbilder auf den Bildflächen x und x' lassen sich mithilfe der Operationen Verschiebung, Skalierung, Scherung und Drehung ineinander überführen. Unter Skalierung versteht man eine gleichmäßige Größenänderung bezüglich der Achsen durch Multiplikation der jeweiligen Koordinate mit einem konstanten Faktor. Als Scherung oder auch Transvektion bezeichnet man in der Elementargeometrie die Überführung einer zweidimensionalen geometrischen Figur, z. B. eines Rechtecks oder Dreiecks in eine andere Figur unter Beibehaltung der Höhe[98]. Durch die Operation wird zum Beispiel ein Rechteck zu einem Parallelogramm. Bei der Scherung bleibt der Flächeninhalt erhalten [*Schupp*, c 1977]. Dies sind jeweils Operationen im zweidimensionalen Bildraum(!).

In der Computergrafik möchte man gerne eine *einheitliche* und vor allem *leicht verknüpfbare* Beschreibung der genannten Operatoren haben. Eine Verschiebung ist eine additive Operation. Skalierung, Scherung und Drehung sind multiplikative Operationen. Per se sind diese Operationen nicht verknüpfbar.

Aus diesem Grund wurden die **homogenen Koordinaten** eingeführt.

[98] $(x, y) \mapsto (x + my, y)$

3.3.2.1 Homogene Koordinaten

Für die Darstellung in homogenen Koordinaten wird die Dimension eines Bildpunktes um Eins erhöht. Wie wir sehen werden, ist es dadurch möglich, den additiven Operator (Verschiebung) als Multiplikation zu schreiben.

Für den zweidimensionalen Fall ergibt sich die folgende Transformation in den –in diesem Fall dreidimensionalen– **projektiven Raum**:

$$(x, y)^T \longmapsto (x, y, 1)$$

Eine Addition von t zu jeder Komponenten (*Translation/ Verschiebung*) lässt sich nun multiplikativ schreiben als:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \begin{bmatrix} 1 & 0 & t \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} x+t \\ y+t \\ 1 \end{pmatrix}$$

Eine *Skalierung* mit dem Faktor $\frac{1}{s}$ ergibt sich durch:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = \begin{pmatrix} x \\ y \\ s \end{pmatrix} = \begin{pmatrix} \frac{x}{s} \\ \frac{y}{s} \\ 1 \end{pmatrix}$$

Da wir den zweidimensionalen Bildraum in den dreidimensionalen *projektiven Raum* eingebettet haben, sind folgende *Äquivalenzklassen* entstanden:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \left\{ \begin{pmatrix} \omega x \\ \omega y \\ \omega \end{pmatrix} \mid \omega \in \mathbf{R} \backslash 0 \right\}$$

(Die Punkte für die gilt $\omega = 0$ bilden eine Hyperebene und können als unendlich ferne Punkte aufgefasst werden.)

Eine *Scherung* parallel zur x-Achse ergibt sich durch:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \begin{bmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} x+my \\ y \\ 1 \end{pmatrix}$$

Eine *Drehung* um α im mathematisch positiven Sinn (gegen den Uhrzeigersinn) um den Nullpunkt ergibt sich durch Multiplikation mit der folgenden Matrix:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix}$$

Der Vorteil der Notation der benötigten Operatoren als 3×3-Matrizen ist die Möglichkeit, mehrere Operationen (durch Multiplikation der Einzelmatrizen) in einer Matrix zusammenzufassen.

Weiterführende Information zur **projektiven Geometrie** findet sich in [*Encarnação et al.*, 1996; *Trucco und Verri*, 1998].

Die von uns gewünschte Transformation eines Bildpunktes von Bild A in Bild B lässt sich mit der neu eingeführten Notation in homogenen Koordinaten folgendermaßen schreiben [*Szeliski und Shum*, 1997]:

$$\begin{bmatrix} x_B \\ y_B \\ \omega_B \end{bmatrix} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ 1 \end{bmatrix} \tag{3.4}$$

Diese Matrix ist die so genannte **Fundamentalmatrix** der **Epipolargeometrie**.

3.3.2.2 Epipolargeometrie

Die Epipolargeometrie beschreibt die Beziehung zwischen Objekten in der Realität und ihren Abbildern von verschiedenen Aufnahmepunkten aus. Die Transformation eines Bildpunktes in einem Bild in das Koordinatensystem des anderen Bildes führt über die Fundamentalmatrix 3.4. Sie ist damit ein geeignetes Werkzeug für unsere geforderte Erweiterung des Idealfalles um die beliebige Position der Kamera im Raum, da sie die Überführung der verschiedenen Kamerabildkoordinatensysteme in einander und damit in ein gemeinsames System ermöglicht. Durch unsere Forderung der perspektiveninvarianten Objekte ist das mithilfe der Fundamentalmatrix transformierte Abbild geometrisch korrekt. Wie wir beim Betrachten der Parallaxe sehen werden, gilt dies normalerweise nicht.

Wir wollen hier einen kurzen Einblick in die Epipolargeometrie geben. Eine ausführliche Einführung findet sich beispielsweise in [*Faugeras*, 1993, 165ff].

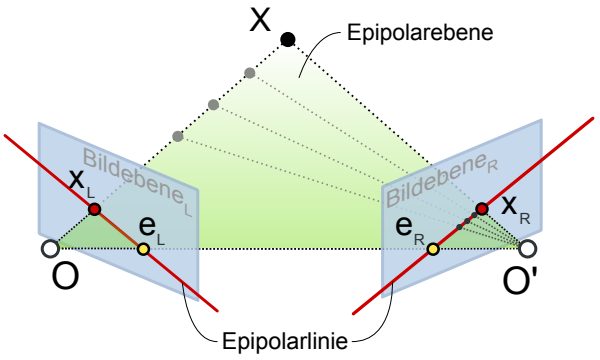


Abbildung 3.13: Die Grafik veranschaulicht die Grundlage der Epipolargeometrie: Alle Weltpunkte aus der Epipolarebene werden auf die beiden Schnittgeraden mit den Bildebenen (Epipolarlinien) abgebildet. O, O' sind die Projektionszentren der jeweiligen Kamera. X ist ein Weltpunkt, x_L, x_R sind dessen Abbilder. e_L, e_R sind die Schnittpunkte der Geraden durch die beiden Projektionszentren mit der jeweiligen Bildebene. Sie sind die so genannten Epipole. Diese Punkte ändern sich nicht, solange die Kameras ihre Position zueinander nicht ändern. Die Epipole sind in jeder Epipolargeraden enthalten (Büschelpunkt des Geradenbüschels).

Grafik 3.13 veranschaulicht die Grundlage der Epipolargeometrie: Die Vektoren von einem Szenenpunkt X zu den projektiven Ursprüngen O, O' der beiden Kameras spannen eine Ebene auf. Alle Szenenpunkte innerhalb der Epipolarebene werden auf die entsprechenden Schnittgeraden (Epipolarlinien) mit den Bildflächen abgebildet [*Pollefeys et al.*, 1999].

Diese Erkenntnis hilft uns sowohl dabei, die Transformation der Bildpunkte von der einen Bildebene in die Koordinaten der anderen zu beschreiben, als auch, korrespondierende Bildpunkte zu identifizieren.

Die Transformation der Bildpunkte auf der linken Ebene in die Koordinaten der rechten Ebene erfolgt mittels der Fundamentalmatrix 3.4. Die Koeffizienten dieser Matrix lassen sich bestimmen, wenn wir mindestens sieben Punktkorrespondenzen zwischen den Ebenen identifiziert haben.

3.3.2.3 Korrespondenzfindung zwischen (benachbarten) Bildern

Mann und Picard [1997] teilen die Verfahren zur Ermittlung von Korrespondenzen in zwei Oberklassen: Solche, die den Fehler „allgemein“ betrachten und solche, die mithilfe von so genannten „Features“ arbeiten. Features bezeichnen hierbei markante Objekte, wie Punkte, Kanten oder ähnliches [*Brown und Lowe*, 2003].

Die historisch ältesten Verfahren gehören zur ersten Klasse. Sie berechnen die **Kreuzkorrelation** der Bilder bei allen möglichen Bildüberlagerungen. Aus der Menge aller möglichen

geometrischen Bildkombinationen wird die Transformation mit dem geringsten Fehler identifiziert. Es ist leicht zu sehen, dass dieser Ansatz sehr berechnungsintensiv ist. Außerdem toleriert er keine lokalen Helligkeitsänderungen zwischen den Bildern. Dafür ist er sehr genau, da alle Bilddaten verwendet werden [*Capel und Zisserman*, 1998].

Eine Beschleunigung stellt der „*hill-climbing*"-Ansatz dar: Ausgehend von einer anfänglichen Schätzung wird das Bild in diejenigen Richtungen verschoben, in die das Fehlermaß[99] sinkt, bis ein Minimum erreicht ist. *Lucas und Kanade* [1981] und *Capel und Zisserman* [1998] schreiben, dass es dabei leicht geschehen kann, dass nur ein lokales Minimum, nicht aber das globale erreicht wird.

Die reine Fehlerberechnung lässt sich durch den „*sequential similarity detection algorithm*" (SSDA) beschleunigen. Die Annahme des Algorithmus ist es, dass hohe Genauigkeit nur in der Nähe des Maximum erforderlich ist. Es werden die Fehler abschnittsweiser kleiner Verschiebungen berechnet. Unter der Voraussetzung, dass ein kumulatives Fehlermaß verwendet wird, kann man diese (Teil-)Fehler aufsummieren und erhält so zwar einen falschen, aber dennoch ungefähren Gesamtfehler einer kombinierten Verschiebung. Es kann bei diesem Verfahren nicht garantiert werden, dass ein Maximum gefunden wird. Für breite Maxima funktioniert das Verfahren gut. [*Lewis*, 1995; *Barnea und Silverman*, 1972]

Als weiteren Ansatz nennen *Lucas und Kanade* [1981] die „*coarse-fine*"-Suchstrategie. Dabei wird zunächst in einer groben Auflösungsstufe der geringste Fehler gesucht. Dieses Ergebnis wird als Ausgangspunkt zur Verfeinerung in einer besseren Auflösungsstufe verwendet. In der grobsten Auflösungsstufe wurde dabei das globale Maximum identifiziert. Mit jedem Skalierungsschritt wird die Näherung verfeinert.

Lucas und Kanade [1981] schreiben, dass sich verschiedene Strategien kombinieren lassen und machen genau dies in ihrer Veröffentlichung zur Bestimmung des **Optischen Flusses** (optical flow). Das Verfahren wurde bereits im Zusammenhang mit der Bildregistrierung in Abschnitt 2.4.2 angesprochen. Als optischen Fluss bezeichnet man die Verschiebung von (Pixel-)Intensitäten beim Übergang von einem Bild zum Nächsten, in unserem Fall also die Verschiebung der Pixel mit gleichem Inhalt zwischen benachbarten Aufnahmen[100] [*Horn und Schunck*, 1981]. *Lucas und Kanade* [1981] berechnen diesen Verschiebungsvektor zwischen zwei Aufnahmen iterativ über eine „coarse-fine"-Suche. *Horn und Schunck* [1981] stellen unabhängig davon ebenfalls ein iteratives Verfahren mit dem gleichen Ziel vor.

Die **featurebasierten Verfahren** operieren zur Korrespondenzfindung nicht auf dem gesamten Bild. Vielmehr werden Features extrahiert. Sind geeignete Features extrahiert, wird versucht, diese in anderen Aufnahmen zu identifizieren. Anhand dieser Punkte können dann die Matrixkoeffizienten in Gleichung 3.4 bestimmt werden. In Abschnitt 3.4.1 betrachten wir ein featurebasiertes Verfahren genauer. An dieser Stelle genügt es zu wissen, dass wir Punktkorrespondenzen finden können.

Die Suche der Korrespondenzen vereinfacht sich bei bekannten Epipolarebenen dadurch, dass die Punkte der Ebene jeweils auf der entsprechenden **Epipolarlinie** liegen müssen. Es muss also nicht das gesamte Bild, sondern nur die korrespondierende Epipolarlinie durchsucht werden. Für den bisher betrachteten Fall, dass die Objekte invariant unter der Perspektive sind, ist dies uninteressant, da direkt aus der Transformation mittels Fundamentalmatrix folgt, wo der korrespondierende Bildpunkt liegt.

[99] Als Fehlermaß können beispielsweise die (jeweils diskrete) L_1-Norm: $\sum |d|$ oder L_2-Norm: $\sqrt{\sum d^2}$ (euklidische Norm) über die Differenzen der einzelnen überlagerten Pixel ($d = p_A - p_B$) dienen.

[100] Bei einer Bildsequenz besteht der optische Fluss aus Richtung und Geschwindigkeit. (Vergleiche auch MPEG-Video-Motion Estimation.) Bei einem konstanten Kameraschwenk wird der Optische Fluss beipielsweise aus konstanten Vektoren entgegen der Schwenkrichtung bestehen.

3.3.2.4 Perspektivenänderung

In der Realität sind die meisten Objekte nicht invariant unter der Perspektive. Wir erweitern unsere Betrachtung um diese Eigenschaft.

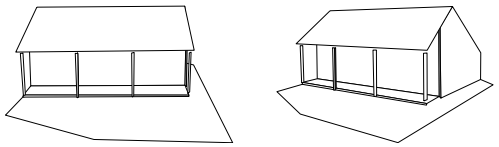
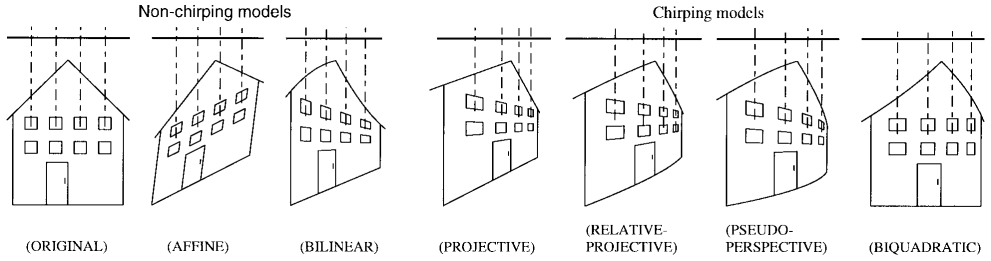


Abbildung 3.14: Die Grafik zeigt die Geometrieänderung eines Objektes bei Abbildung von verschiedenen Augpunkten aus. [Folien zur Vorlesung „Multiple View Geometry“ von Richard Hartley und Andrew Zisserman im Juni 1999]

Grafik 3.14 veranschaulicht mögliche Geometrieänderungen, wie sie durch das Verändern des Augpunktes auftreten. Die dargestellte Änderung ist im Beispiel deutlich stärker als die in unserem konkreten Fall zu erwartende Perspektivenverschiebung.

Eine Zusammenfassung von Geometrieänderungen sowie die zugehörigen mathematischen Transformationen finden wir in Abbildung 3.15.



Model	Coordinate transformation from $\mathbf{x}$ to $\mathbf{x}'$	Parameters
Translation	$\mathbf{x}' = \mathbf{x} + \mathbf{b}$	$\mathbf{b} \in \mathbb{R}^2$
Affine	$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$	$\mathbf{A} \in \mathbb{R}^{2\times 2}, \mathbf{b} \in \mathbb{R}^2$
Bilinear	$x' = q_{x'xy}xy + q_{x'x}x + q_{x'y}y + q_{x'}$ $y' = q_{y'xy}xy + q_{y'x}x + q_{y'y}y + q_{y'}$	$bfq_* \in \mathbb{R}$
Projective	$\mathbf{x}' = \frac{\mathbf{A}\mathbf{x}+\mathbf{b}}{\mathbf{c}^T\mathbf{x}+1}$	$\mathbf{A} \in \mathbb{R}^{2\times 2}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^2$
Relative-projective	$\mathbf{x}' = \frac{\mathbf{A}\mathbf{x}+\mathbf{b}}{\mathbf{c}^T\mathbf{x}+1} + \mathbf{x}$	$\mathbf{A} \in \mathbb{R}^{2\times 2}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^2$
Pseudo-perspective	$x' = q_{x'x}x + q_{x'y}y + q_{x'} + q_\alpha x^2 + q_\beta xy$ $y' = q_{y'x}x + q_{y'y}y + q_{y'} + q_\alpha xy + q_\beta y^2$	$\mathbf{q}_* \in \mathbb{R}$
Biquadratic	$x' = q_{x'x^2}x^2 + q_{x'xy}xy + q_{x'y^2}y^2 + q_{x'x}x + q_{x'y}y + q_{x'}$ $y' = q_{y'x^2}x^2 + q_{y'xy}xy + q_{y'y^2}y^2 + q_{y'x}x + q_{y'y}y + q_{y'}$	$bfq_* \in \mathbb{R}$

Abbildung 3.15: Die Grafiken veranschaulichen die in der Tabelle aufgelisteten Transformationen. Diese Transformationen können Objekte in zwei zueinander zu registrierenden Bildern ineinander überführen. Als „CHIRPING“ (change in spatial frequency with position) bezeichnen die Autoren auf das Beispiel bezogen, dass abhängig von der Position mehr oder weniger Fenster im gleichen Intervall abgebildet werden, obwohl sie in der Realität gleichverteilt sind. [*Mann und Picard*, 1997]

Einige der dargestellten Geometrieänderungen wurden von uns bereits im Rahmen der Verzeichnungskorrektur (siehe Abschnitt 3.2.2) behandelt und treten daher an dieser Stelle nicht auf.

Die skizzierten Geometrieänderungen können nicht mithilfe der Fundamentalmatrix 3.4 ineinander überführt werden. Dazu sind kompliziertere Operationen notwendig, wie sie *Mann und Picard* [1997] in der Tabelle in Abbildung 3.15 notiert haben.

3.3.2.5 Parallaxe

Zum Beschreiben der Blickpunktänderung führen wir zunächst den Begriff der **Parallaxe** ein. Als Parallaxe bezeichnet man den Winkel zwischen den Verbindungsgeraden zu einem Punkt von zwei verschiedenen Orten aus [*Brockhaus*, 2006c]. In Grafik 3.13 ist die Parallaxe der Kameras zueinander der Winkel:

$$\angle(\overline{OX}, \overline{O'X})$$

Dadurch, dass wir verschiedene Kamerapositionen haben, erhalten wir Bilder von unterschiedlichen Blickpunkten. Von den leicht entfernten Aufnahmepunkten können wir sowohl verschiedene Dinge, als auch die Dinge verschieden sehen. Abbildung 3.16 verdeutlicht dies.

Abbildung 3.16: Zwischen den beiden Aufnahmen wurde das optische Zentrum der Kamera verschoben. Dadurch tritt vor allem in der Nähe Parallaxe auf. An der relativen Position der Lampe zum Geländer und den Gebäuden im Hintergrund wird dies deutlich. [astro.livjm.ac.uk]

Wie an den Aufnahmen zu sehen ist, zeigen die Bilder besonders im Nahbereich unterschiedliche Objektkonstellationen: Die relative Position der Objekte zueinander ändert sich. Dadurch ergeben sich sowohl Probleme bei der Korrespondenzfindung, als auch beim Zusammenfügen der Bilder.

Die Ursache des Problems wird aus Abbildung 3.17 deutlich. Die Grafik auf der linken Seite

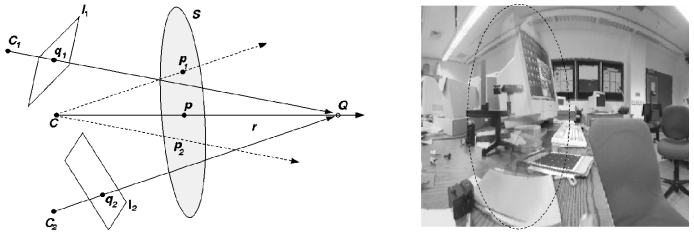


Abbildung 3.17: Die Kameras C_1 und C_2 bilden den Szenenpunkt Q auf $q1$ bzw. q_2 ab. Bei der Projektion auf die gemeinsame Ebene S (z.B. Panoramastitching) entstehen die Punkte P_1 und P_2. P wäre die perspektivisch korrekte Projektion von Q auf S. Rechts wird der Effekt bildlich gezeigt. [*Swaminathan und Nayar*, 2000]

zeigt zwei Kameras mit unterschiedlichem Blickpunkt. Beide Kameras bilden das Objekt Q ab. Zum Stitchen der Aufnahmen werden die Punkte der Bildebenen I_1 und I_2 in das gemeinsame Koordinatensystem der virtuellen Ebene S abgebildet. P wäre die Abbildung von Q, wenn wir die Szene vom virtuellen Blickpunkt C betrachteten. Ohne Parallaxe würden die Punkte P_1 und P_2 mit dem Punkt P zusammenfallen.

Die rechte Seite der Abbildung zeigt ein mögliches Stitchingergebnis. Im Bereich des Monitorrumpfes sehen wir die durch die Parallaxe entstehende Inkonstistenz der Bildinhalte der beiden Kameras im überlappenden Bildteil.

Aus der linken Abbildung ist ersichtlich, warum die Parallaxe bei kameranahen Objekten besonders stark ausfällt: Je weiter das Objekt Q von den Kameras entfernt ist, desto kleiner wird der Winkel $\angle(\overline{C_1Q}, \overline{C_2Q})$ und damit die Parallaxe zwischen P und P_i: Die projizierten Punkte P_1 und P_2 rücken näher an ihre eigentliche Positon P.

Wir haben gesehen, dass Parallaxe für weit entfernte Objekte eine geringere Rolle spielt als für **nahe Objekte**.

Durch die Parallaxe und die Transformation der verschiedenen Quellbilder in eine Zielprojektion ergibt sich ein weiteres Problem: die **Verdeckung**.

In Abbildung 3.18 sehen wir zwei Blickpunkte A und B, von denen aus Bildinformation vorliegt. M ist unser virtueller Blickpunkt, von dem aus wir das Zielbild darstellen wollen. In der Abbildung ist der jeweils von den Objekten abgeschattete Bereich dargestellt. Relevant

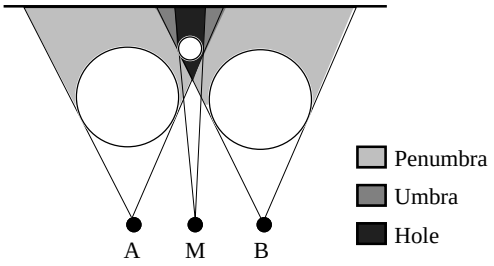


Abbildung 3.18: Die Abbildung veranschaulicht ein Problem bei der Transformation zweier Bilder zu einem dritten –virtuellen– Standpunkt M: Es kann Regionen der realen Szene geben, die auf keiner Aufnahme abgebildet wurden („Umbra“). An der Stelle des Loches („Hole“) liegt keine Bildinformation vor. Der kleine Kreis wird in der Aufnahme vom virtuellen Standpunkt M nicht zu sehen sein. [*Chen und Williams*, 1993]

für uns ist der halbdunkle Bereich „Umbra“. Dieser Bereich wird weder von Kamera A noch von Kamera B abgebildet. Als Folge liegt keinerlei Information über die Existenz des kleinen runden Kreises vor. Dieser kann somit niemals auf der berechneten Abbildung vom virtuellen Blickpunkt M aus erscheinen.

Parallaxe kann sowohl zu *inkonsistenter Bildinformation* –in Bezug auf das Zusammenfügen der Bilder– als auch zu *fehlender Bildinformation* –in Bezug auf einen neuen virtuellen Standpunkt– führen. Daher müssen wir die Parallaxe für einen konsistenten Bildeindruck unseres, von mehreren Blickpunkten aufgenommenen, Panoramas korrigieren.

3.3.2.6 Parallaxekorrektur

Zur Korrektur der Parallaxe müssen wir zunächst zusammengehörige Pixel in den benachbarten Aufnahmen identifizieren. Verfahren dazu haben wir bereits im letzten Abschnitt kennen gelernt. Aufgrund der Parallaxe erwarten wir leichte Perspektivenänderungen zwischen unseren Teilaufnahmen. Daher sind solche Verfahren zur Korrespondenzfindung geeignet, die derartige ungleichmäßige Pixelverschiebungen von Bild zu Bild tolerieren. Dies trifft insbesondere auf die featurebasierten Ansätze zu, die wir in Abschnitt 3.4.1 betrachten werden.

Sind die Korrespondenzen erkannt, so müssen wir die Bilder geeignet ineinander überführen. Damit beschäftigen wir uns in diesem Abschnitt. Wir werden mit einfacher Rekonstruktion im Zweidimensionalen beginnen und dann zum (realitätsnäheren) Fall übergehen, der die dreidimensionale Geometrie berücksichtigt.

Die uns vorliegende Bildinformation der Kameras ist zweidimensional. Die Welt aus Sicht der Kamera wird von der so genannten **plenoptischen Funktion** beschrieben. Die plenoptische Funktion $p = P(\Theta, \Phi, \lambda, V_x, V_y, V_z, t)$ gibt an, was wir von einem Blickpunkt aus zu einer Zeit

t wahrnehmen können. Sie besteht in unserem Fall gerade aus den Pixelwerten. Der Vektor (V_x, V_y, V_z) ist unsere feste Kameraposition, t der Aufnahmezeitpunkt. Θ und Φ werden durch den Öffnungswinkel und die Auflösung der Kamera bestimmt[101]. λ ist der abgebildete Lichtwellenbereich. [*Adelson und Bergen*, 1991]

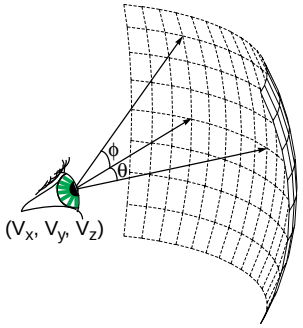


Abbildung 3.19: Die Abbildung skizziert die plenoptische Funktion $p = P(\Theta, \Phi, \lambda, V_x, V_y, V_z, t)$. [*McMillan und Bishop*, 1995]

Bereits diese Information reicht zusammen mit den ermittelten Korrespondenzen aus, um die Parallaxe für einen konsistenten Bildeindruck zu korrigieren:

Für ein (geometrisch nicht notwendigerweise korrektes) konsistentes Zusammenfügen der Teilbilder kann man diese an den Überlappungsbereichen zusammenmorphen: Korrelierende Punkte in benachbarten Aufnahmen werden dazu im überlagerten Bild zusammengeschoben, bis sie sich decken. Die Zwischenbereiche werden (linear) interpoliert. *Beier und Neely* [1992] beschreiben dieses Verfahren[102].

Bei dem soeben skizzierten Verfahren wird lediglich zwischen den Aufnahmen „geometrisch interpoliert“. Wissen über die Lage der Blickpunkte zueinander und damit die Augpunktverschiebung zwischen den Aufnahmen wird nicht verwendet. Es ist leicht einzusehen, dass dadurch kein geometrisch korrektes Bild für unseren virtuellen (dritten) Blickpunkt entstehen muss, da die Interpolation nicht in geeigneter Weise erfolgt. Das Verfahren ist allerdings einfach und führt zu einem konsistenten Bild ohne erkennbare Unstetigkeiten.

Für eine geometrisch korrektere Interpolation der Aufnahmen müssen wir Wissen über die Lage der Objekte im Dreidimensionalen verwenden. Ein naheliegender Weg in diese Richtung ist es, die Aufnahmen gemäß der Aufnahmeposition auf geeignete Mannigfaltigkeiten[103] zu rendern und anschließend diese im Raum „stehenden“ Flächen zu samplen. Siehe Abbildung 3.20. Der Unterschied beispielsweise zur idealen Zylinderprojektion in Abbildung 3.10 besteht darin, dass die Flächen nicht so „schön“ im Raum angeordnet sind. Bei der Überlappung der

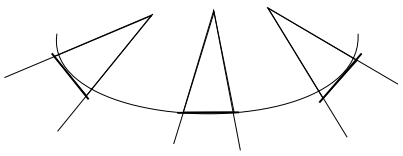


Abbildung 3.20: Die Abbildung skizziert die Projektion von Kamerabildern mit verschiedenen Blickpunkten auf eine geeignete Mannigfaltigkeit. Von dieser kann beispielsweise mittels Zylinderprojektion von einem virtuellen Standpunkt aus ein konsistentes Bild erzeugt werden. [*Peleg und Herman*, 1997]

Ebenen ist es wichtig, die Tiefeninformation zur Bestimmung der korrekten Verdeckung zu berücksichtigen.

[101] Sie sind diskret.
[102] Das in dem Paper als Anschauungsmaterial verwendete Musikvideo „Black or White“ von Michael Jackson macht von dieser Technik Gebrauch.
[103] Bei Kameras, die planar abbilden, sind dies Ebenen.

Peleg und Herman [1997] und *Wood et al.* [1997] beschreiben ein solches Vorgehen für Videodaten, die sie auf Mannigfaltigkeiten rendern. *Swaninathan et al.* [2003] berechnen zu gegebenen Aufnahmen solche von einem virtuellen Standpunkt aus. *Eisemann et al.* [2008] projizieren die Bilddaten als Textur auf geeignete geometrische Objekte und erhalten so ein konsistentes Bild.

Der vorgestellte Ansatz bietet sich für die Hardwarebeschleunigung an (siehe dazu auch Kapitel 7).

Nachdem wir nun bereits die Bildflächen in einen dreidimensionalen virtuellen Raum eingebettet haben, können wir auch die abgebildeten Objekte selbst dorthin projizieren.

Wie wir bereits gesehen haben, ist die Verschiebung von Bildpunkten bei Perspektivenänderung abhängig von ihrer Entfernung zum Betrachter: Weiter entfernte Punkte werden weniger verschoben als nahe. Für eine korrekte Rekonstruktion der Perspektive ist daher die Tiefeninformation notwendig. Mithilfe der in Abschnitt 3.3.2.2 betrachteten Epipolargeometrie können wir die Tiefeninformation bei bekannter Lage der Kameras zueinander mitunter aus den Aufnahmen zurückrechnen.

Sind die Tiefenwerte bekannt, so können wir die Pixelwerte in einem virtuellen dreidimensionalen Raum anordnen und –wie perspektivisch gewünscht– erneut samplen. Die Perspektivenänderung auf Basis der beiden zweidimensionalen Quellaufnahmen und der Tiefeninformation bezeichnet *McMillan Jr* [1997] als **Warping**. Je nach gewähltem neuen Blickpunkt ist die vorhandene Bildinformation lückenhaft, wie wir in Abbildung 3.21 sehen (vergleiche auch Abbildung 3.18).

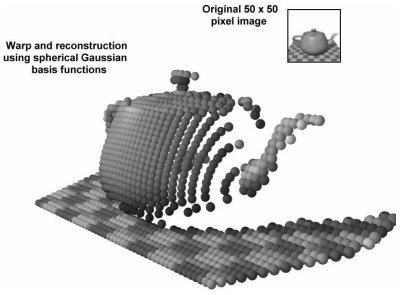


Abbildung 3.21: Die Abbildung zeigt oben das Quellbild. Aus diesem wurde zusammen mit der Tiefeninformation das linke dreidimensionale Modell erstellt und gedreht. Jeder Bildpunkt wurde dabei als Kugel modelliert. Wir sehen das Problem der fehlenden Information. [*McMillan Jr*, 1997]

McMillan und Bishop [1995] verwenden diese Technik, um aus zwei Zylinderpanoramen die plenoptische Funktion eines dritten (virtuellen) Standpunktes zu berechnen. In unserer Anordnung erhalten wir von jeder Kamera nur ein Ausschnittsbild und kein komplettes Zylinderpanorama. Wir betrachten dennoch kurz die Ausführungen von *McMillan und Bishop* [1995], da sie ein mögliches Vorgehen aufzeigen und veranschaulichen, mit welchen Bildfehlern bei einem, lediglich auf den beiden Quellbildern basierenden, Warping zu rechnen ist:

Die Autoren bestimmen korrelierende Bildpunkte und minimieren anschließend den Fehler der möglichen Warpings unter der getroffenen Annahme der Lage der Augpunkte zueinander. Mögliche Fehler, die zum Teil aufgrund der diskreten[104] Fehlerminimierung auftreten, sind in Abbildung 3.22 dargestellt.

Haben wir die Pixel korrekt im dreidimensionalen Raum eingebettet, so erhalten wir beim erneuten Sampling, von unserem gewünschten (virtuellen) Augpunkt aus, perspektivisch korrekte Repräsentationen der Objekte.

[104] Es wird nur über die korrelierten Punkte minimiert und dazwischen interpoliert.

Abbildung 3.22: Die Abbildung zeigt Bildfehler, wie sie bei einem Stitching nach beschriebenem Schema entstehen können. Auf der linken Seite sehen wir an der Hausecke Fehler, die mutmaßlich aus der „fehlerhaften“ Interpolation zwischen korrellierten Punkten stammen. Die rechten Fehler zeigen vermutlich eine zu geringe Auflösung beziehungsweise fehlende Information der Quellbilder, die für das Zielbild interpoliert werden musste. [*McMillan und Bishop*, 1995]

Auch ohne die auftretenden Lücken muss der Bildeindruck nicht dem realen Bildeindruck vom neuen Blickpunkt aus entsprechen. Dies liegt daran, dass beispielsweise die Beleuchtung vom neuen Augpunkt aus anders erscheinen kann. Hier sind vor allem falsche Schatten auffällig.

Ein möglicher Ansatz ist es, so genannte Geometrieproxies für Objekte und Lichtquellen zu modellieren, die Pixel als Textur zu verwenden und anschließend die Szene neu zu rendern. Aus den Pixeldaten werden also 3D-Modelle generiert, die mit der Bildinformation texturiert werden. Dies ist auch für das „einfache Warping“ teilweise nützlich. Zusätzlich werden nun auch die Lichtquellen modelliert. Hier entstehen erneut Probleme. Beispielsweise muss versucht werden, die Pixelinformation von der Beleuchtungsinformation zu trennen, da die Beleuchtung erst beim erneuten Rendern einfließt (siehe dazu Abschnitt 4.1.4). Auch die Bestimmung der Geometrieproxies ist schwierig. Dafür wird es allerdings möglich, die Schatten und Glanzpunkte auch für die neue Perspektive korrekt darzustellen.

Für unsere Anwendung ist dieser Ansatz zu aufwendig und aufgrund der Komplexität realer Szenen auch nur schwer durchführbar. Für nähere Information zu dem Ansatz über die Geometrieproxies sei beispielsweise auf [*Chen und Williams*, 1993], [*Carranza et al.*, 2003], [*Vedula et al.*, 2005] und [*Eisemann et al.*, 2008] verwiesen.

Abbildung 3.23: Im Vordergrund der gestitchten Aufnahme sind deutlich Parallaxefehler zu erkennen. Bei in der Ferne liegenden Punkten fällt die Parallaxe dagegen nicht auf. [Bruno@panotools.info]

Wie wir gesehen haben, wird der Parallaxefehler umso geringer, je weiter ein Objekt von der Kamera entfernt ist. Wenn wir lediglich weit entfernte Objekte abbilden wollen, können wir zumeist auf eine komplizierte Parallaxekorrektur verzichten. Würden wir in Abbildung 3.23 die nahen Bildteile abschneiden, so fiele die Parallaxe nicht auf.

Da unsere Kameras fest montiert sind, reicht es aus, die perspektivische Korrektur zum weitgehenden Beheben der Parallaxe einmal zu definieren. Da wir zweidimensionale Quell- und Zielbilder haben, können wir anschließend den aus der Korrektur resultierenden optischen Fluss von den Quellbildern zum Zielbild bestimmen und fortan direkt die zweidimensionale Transformation anwenden. Eine Rekalibrierung der Registrierungsparameter von Zeit zu Zeit sollte ausreichen.

3.4 Featurebasierte Korrespondenzfindung

In Abschnitt 3.3.2.3 haben wir allgemeine Verfahren zur Korrespondenzfindung zwischen Bildern betrachtet. In diesem Teil wollen wir auf die featurebasierten Verfahren am Beispiel von Shift Invariant Feature Transform (SIFT) eingehen.

Die featurebasierten Verfahren ermitteln die Korrespondenzen zwischen besonderen Bildmerkmalen – den so genannten Features. Sie unterscheiden sich dadurch von den „allgemeinen“ Verfahren, die darauf basieren, ganze Bildteile übereinander zu schieben und den Fehler zu minimieren (vergleiche Abschnitt 3.3.2.3).

In unserer Betrachtung des realen Aufnahmefalles in Abschnitt 3.3.2 haben wir gesehen, dass es durch die Parallaxe zu einer Geometrieänderung der abgebildeten Objekte zwischen den verschiedenen Augpunkten kommt. Dies ist für die allgemeinen Verfahren problematisch, da sich insbesondere näher am Betrachter befindliche Objekte stärker von einem Blickpunkt zum nächsten verschieben als fernere. Der Fehler der Überlagerung der unveränderten Quellbilder ist kein aussagekräftiges Fehlermaß mehr. Für die Korrektur der Parallaxe benötigen wir gerade die Beziehung einzelner Bildbereiche, um diese individuell geeignet verschieben zu können (siehe Abschnitt 3.3.2.6).

Für featurebasierte Ansätze stellt die Bildänderung durch Parallaxe kein Problem dar. *Lowe* [1999] schreibt zu SIFT, dass es robust in Bezug auf Skalierung, Translation, Drehung, affine und dreidimensionale Projektionen sowie Beleuchtungsänderungen bis zu einem bestimmten Grad sei. Um dies zu erreichen, ist eine geeignete *Auswahl von Features* ebenso wie deren *Charakterisierung* entscheidend.

Mann und Picard [1997] schreiben, dass für die *Auswahl* von geeigneten Punkten unter anderem Rauschen und Verdeckung problematisch sind. In verschiedenen Verfahren wurden Liniensegmente, Anordnungen von Kanten, Bildregionen und Ähnliches als Identifikationskriterien für mögliche Features gewählt [*Lowe*, 1999].

Die Parameter, mit denen die ausgewählten Features *charakterisiert* werden, sind wesentliches Alleinstellungsmerkmal des jeweiligen Verfahrens: *Schaffalitzky und Zisserman* [2002] gehen für die Auswahl ähnlich vor wie *Lowe* [1999] in seinem, von uns anschließend ausführlicher betrachteten, Verfahren. Sie charakterisieren ihre Features jedoch unterschiedlich.

Gewünschte Eigenschaften der extrahierten **charakteristischen Merkmale** eines Features sind die Einmaligkeit und die Robustheit gegenüber Änderungen.

Einmaligkeit bedeutet, dass die extrahierten Merkmale das Objekt eindeutig identifizieren und dabei deutlich von anderen Objekten abgrenzen. Je verschiedener die charakteristischen Merkmale der einzelnen Objekte sind, desto besser lassen sie sich aus dem Rauschen identifizieren[105].

Die **Robustheit** gegenüber Änderungen bezeichnet die erwünschte Unabhängigkeit der charakteristischen Eigenschaften von Änderungen wie Neigung, Drehung, aber auch Beleuchtung des Objektes.

Eine Übersicht über verschiedene Featurebeschreibungstechniken findet sich in [*Mikolajczyk und Schmid*, 2005].

Mit SIFT (**Shift Invariant Feature Transform**) [*Lowe*, 1999] existiert ein sehr robustes Verfahren, das heute vielfach eingesetzt wird[106]. Wir wollen daher im Folgenden auf das Verfahren eingehen. Eine Übersicht über die historischen Vorläufer, die bis in das Jahr 1981 zurückgehen, gibt *Lowe* [2004].

[105] Bei der Suche nach einem Feature stellen alle anderen Features Rauschen in Bezug auf die Identifikationskriterien dar. Sind die Beschreibungsvektoren mehrerer Features ähnlich, so ist deutlich mehr Information über jedes der betroffenen Objekte notwendig, als wenn die Unterschiede groß sind.

[106] Es gibt Weiterentwicklungen des Verfahrens. Eine solche ist beispielsweise „gradient location and orientation histogram“ GLOH [*Mikolajczyk und Schmid*, 2003]. Die Verbesserungen am Verfahren benötigen oft deutlich mehr Rechenzeit.

3.4.1 Shift Invariant Feature Transform

In Abschnitt 3.3.2.3 haben wir bereits gesehen, dass die featurebasierten Verfahren zur Korrespondenzfindung in drei Schritten operieren:

- Feature**extraktion**
- Feature**charakterisierung**
- Feature**identifikation**

Wir betrachten diese Schritte nun am Beispiel Shift Invariant Feature Transform (SIFT).

Die **Featureextraktion** erfolgt bei SIFT über einen mehrstufigen Filterprozess: In der ersten Stufe werden Maxima und Minima der Differenzen der gaußgefilterten (difference-of-Gaussian, DoG) skalierten Versionen des Bildes ermittelt[107]. Dazu wird eine Auflösungspyramide aus dem DoG-Bild aufgebaut[108].

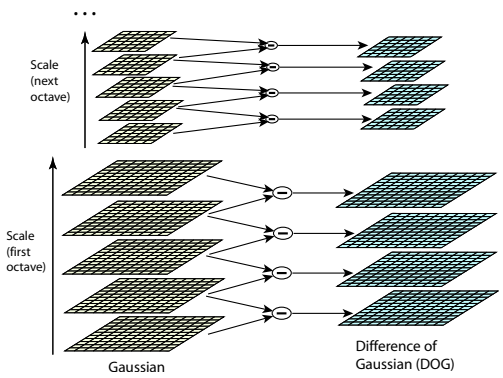


Abbildung 3.24: Auf der linken Seite sehen wir die mehrfach gaußgefilterten (in jeder Ebene jeweils einmal mehr) Versionen der jeweiligen Auflösungsstufe (Oktave genannt). Rechts sind die Differenzen eingezeichnet. Beim Übergang zur nächsten Oktave (oben) wird die Auflösung jeweils halbiert. [*Lowe*, 2004]

Es werden die Maxima und die Minima der verschiedenen DoG-Auflösungsstufen bestimmt. Diejenigen Extrema, die in mehreren Skalierungsstufen auftreten, sind Keypoints –potenzielle Features– und werden weiterverarbeitet: Für sie wird jeweils ein Beschreibungsvektor (s.u.) erstellt. Der Autor schreibt, dass in einem 512x512 Pixel Bild auf diese Weise an die 1000 SIFT-Keys entstehen. Im Laufe des Verfahrens werden schwache Keypoints, die keinen geeigneten Beschreibungsvektor liefern (Distinktivität, Robustheit), aussortiert. Am Ende bleiben deutlich weniger als 1000 Features übrig.

Zur **Charakterisierung** eines Features wird ein so genannter *Beschreibungsvektor* erstellt. Dazu werden die Richtungen der lokalen Kanten mit einem Richtungshistogramm abgeglichen. Ein **Histogramm** beschreibt die Häufigkeitsverteilung von Messwerten – in unserem Fall Helligkeitsintensitätswerte und deren Beziehung zueinander (siehe auch Abbildung 5.20). Ein geeignetes Mittel zum Beschreiben dieser Beziehung ist der **Gradient**. Der Gradient ist ein Vektor, der in die Richtung des stärksten Anstiegs der skalaren Helligkeitswerte zeigt und die Länge des Betrages der Helligkeitsänderung hat.

[107] Die DoG-Funktion ist eine Annäherung an den normalisierten Laplacian-of-Gaussian (LoG) $\sigma^2\Delta^2 G$. *Lindeberg* [1994] hat gezeigt, dass eine Skalierung mit σ^2 für Skaleninvarianz notwendig ist. Dass die Maxima und Minima sehr stabile Features darstellen, wird in den vergleichenden Studien [*Mikolajczyk und Schmid*, 2003] und [*Mikolajczyk und Schmid*, 2005] gezeigt. [*Lowe*, 2004]

[108] Die Auflösung wird zunächst durch bilineare Interpolation verdoppelt, um die höchsten Frequenzen bei der Differenzenbildung nicht zu verlieren. Dann wird zuerst zweimal mit $\sigma = \sqrt{2}$ gefiltert ($g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2}$). Anschließend wird mit dem Faktor 1.5 bilinear herunterskaliert. [*Lowe*, 1999]

Das SIFT-Gradientenhistogramm enthält 36 Einträge, die jeweils 10° Raumwinkel abdecken. Alle Vektoren, die jeweils innerhalb eines solchen 10° Winkelbereiches liegen, tragen zum entsprechenden Histogrammwert bei. In Grafik 3.25 sind zur Veranschaulichung acht (45°-)Äquivalenzklassen eingezeichnet. Die höchsten Histogrammeinträge werden weiterverarbeitet. Für genauere Information sei auf [*Lowe*, 2004] und die Grafiken 3.25 und 3.26 verwiesen.

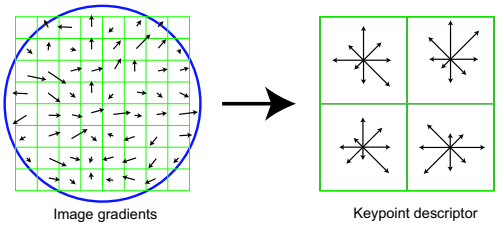


Abbildung 3.25: Zur Bestimmung des Beschreibungsvektors (Keypoint-Descriptor) werden zuerst die Gradientenstärke und -richtung in einem bestimmten Gebiet um den Keypoint berechnet (links). Diese werden dann mit einer Gaußfunktion gewichtet (angedeutet durch den Kreis). Anschließend werden Richtungshistogramme über 4x4-Regionen erstellt (rechts). Zur Veranschaulichung sind hier nur 8 Richtungen im Histogramm sowie ein 2x2-Beschreibungsvektor (Descriptor) aus einem 8x8-Array aufgezeigt. *Lowe* [2004] schreibt, dass ein 4x4-Histogrammarray mit einem jeweils 8-stufigen (wie eingezeichnet) Histogramm einen guten Beschreibungsvektor ergibt. Dieser hat 4x4x8=128 Einträge. [*Lowe*, 2004]

Zur Ermittlung des Beschreibungsvektors des Features werden die skalierten DoG-Stufen verwendet. Sie stellen eine unschärfere und abstraktere Version des Bildes dar. Zusätzlich werden Richtung und Stärke der Gradienten quantisiert [*Mikolajczyk und Schmid*, 2005]. Dadurch wird die Featurecharakterisierung unabhängig bezüglich geringer Geometrieänderungen und bezüglich einer affinen Intensitätsänderung der Gradienten [*Brown und Lowe*, 2003]. *Lowe* [1999] schreibt, dass dies dem Verhalten von signalverarbeitenden Zellen im cerebralen Kortex von Säugetieren ähnelt.

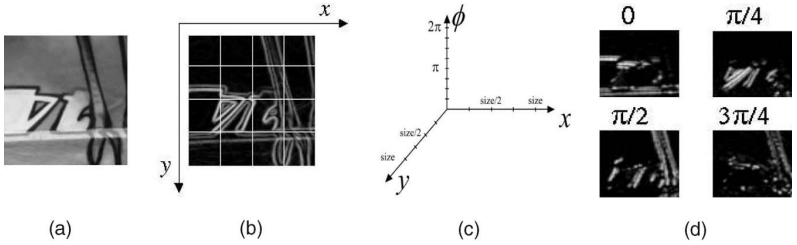


Abbildung 3.26: (a) Umgebung des Featurepunktes. (b) Gradientenbild mit Ortsgitter. (c) Die Dimensionen eines dreidimensionalen ortsbasierten Histogramms mit 8 Stufen (je $\frac{\pi}{4} \cong 45°$; vgl. Abb. 3.25). Ein Histogrammeintrag ist die jeweilige Anzahl der Pixel in der Schnittfläche mit der Φ-Achse. (d) Grafische Veranschaulichung von vier der acht Richtungsebenen des Histogramms. [*Mikolajczyk und Schmid*, 2005]

Um die Featurebeschreibung robuster zu machen, wird der berechnete Beschreibungsvektor normalisiert gespeichert. Durch die Normalisierung wird das Feature unabhängig von linearer Kontraständerung. Eine Helligkeitsänderung (additiv) wirkt sich auf den Gradienten nicht aus. Nichtlineare Beleuchtungsänderungen (z.B. Sättigung oder Umorientierung des Features bezüglich der Lichtquelle) haben vor allem Einfluss auf die Gradientenstärke. Daher werden große Gradienten vor der Normalisierung auf einen Maximalwert gekürzt. Die Richtung wird durch die nichtlineare Beleuchtungsänderung in der Regel nicht so stark verändert.

Zur **Identifizierung** von Objekten werden die Beschreibungsvektoren der für das Objekt charakteristischen Features verglichen (siehe Abbildung 3.28). Dazu wird die in [*Beis und Lowe*, 1997] vorgestellte Best-Bin-First-Methode auf einem Suchbaum verwendet. Eine Über-

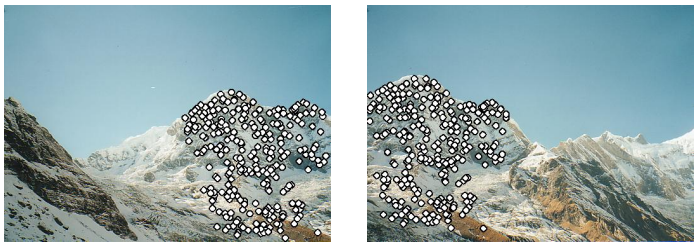


Abbildung 3.27: Korrespondierende SIFT-Features in zwei Bildern. [*Brown und Lowe*, 2003]

einstimmung von drei Features mit niedrigem Residuum[109] indiziert das Vorhandensein des entsprechenden Objektes. Ebene Objekte können sich dabei bis zu 60° um eine zur Bildebene orthogonale Achse drehen, um noch erkannt zu werden. Eine Drehung um eine andere Achse ist um bis zu 20° möglich. Ebenso kann die Beleuchtung im Vergleich zu anderen (vor allem früheren) Verfahren zu einem recht hohen Grad variieren. Die Effizienz des Verfahrens wird in Grafik 3.28 veranschaulicht. [*Lowe*, 1999]

Bei Best-Bin-First wird eine Nearest-Neighbor-Suche[110] verwendet (siehe auch *Beis und Lowe* [1997]). Dazu werden mögliche Kandidaten zunächst mittels Hough-Transformation [*Hough*, 1962] auf einer Hashtable ermittelt. Die weitere Eingrenzung der möglichen Matchings erfolgt mittels Minimierung des quadratischen Fehlers. [*Lowe*, 1999, 2001; *Brown und Lowe*, 2003; *Lowe*, 2004]

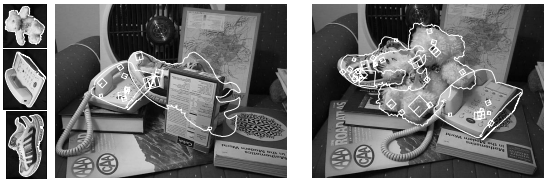


Abbildung 3.28: Die Grafik zeigt die Effektivität des SIFT-Ansatzes. Links sind die Objekte, aus denen die SIFT-Keys erzeugt wurden. Die beiden großen Bilder zeigen, dass die Objekte trotz teilweiser Verdeckung zuverlässig erkannt werden (drei Übereinstimmungen reichen). Die Quadrate bezeichnen hierbei die charakteristischen Merkmale, die im SIFT-Key gespeichert wurden. [*Lowe*, 1999]

Wir haben die drei Kernprobleme zur Korrespondenzfindung zwischen Bildern mithilfe von Features (**Featurematching**) identifiziert und betrachtet:

- Was sind geeignete mögliche Features?
- Wie charakterisiert man diese am besten (Invarianz/ Distinktivität)?
- Wie identifiziert man die Features am geschicktesten?

Mit SIFT haben wir eine Lösungsmöglichkeit kennen gelernt. Zuvor gab es zwar ähnliche Ansätze (z.B. [*Zhang et al.*, 1995], [*Schmid und Mohr*, 1997]), Lowe schreibt jedoch, dass diese nur auf einer räumlichen Skalierungsstufe operiert, und daher keine skalierungsinvarianten Schlüssel extrahiert und erkannt haben. Eine Evaluation verschiedener Verfahren findet sich beispielsweise in [*Mikolajczyk und Schmid*, 2003] und [*Mikolajczyk und Schmid*, 2005].

Auch wenn SIFT weniger anfällig gegenüber Beleuchtungsänderungen ist, so kann es dennoch Probleme geben, wenn Features von einer Kamera zur nächsten aufgrund von Über- oder Unterbelichtung „verschwinden“. Durch die Verwendung von HDR-Daten wird dieses Problem bei uns nicht auftreten. Auch für die Registrierung erweist sich unser Ansatz als Vorteil.

[109] Als Residuum bezeichnet man in der Statistik den Unterschied zwischen dem beobachteten und dem vorhergesagten Wert.

[110] Als Nähekriterium wird die Euklidische Distanz zum entsprechenden Beschreibungsvektor bestimmt.

3.5 Stitching mithilfe von Featurematching

Mit SIFT steht uns ein robustes Verfahren zum Ermitteln von Featurekorrespondenzen zur Verfügung. Damit haben wir alle Bestandteile, die wir zum **Stitching**, also dem konsistenten Zusammenfügen mehrerer Aufnahmen, benötigen.

Wir wollen eine mögliche Verarbeitungskette nach *Brown und Lowe* [2003, 2007] betrachten:

Als Ergebnis der Verarbeitung in Kapitel 2 liegen uns die HDR-Aufnahmen der verschiedenen Kameras vor. Weiterhin kennen wir die Anordnung der Kameras im Raum. Da die Bestimmung dieses Parameters jedoch aufwendig ist und zudem nicht unbedingt benötigt wird, werden wir ihn nicht berücksichtigen und stattdessen die Beziehungen der Einzelbilder zueinander anhand der Featurekorrespondenzen durch Fehlerminimierung bestimmen.

Im ersten Schritt extrahieren wir mithilfe von SIFT geeignete Features in allen Aufnahmen. Anschließend versuchen wir diese Features gegeneinander zu matchen. Dazu ist es nicht notwendig, dass die Bilder in einer bestimmten Reihenfolge vorliegen. Anhand der Korrespondenzen versuchen wir nun, eine geometrische Transformation zwischen den Bildern zu finden, die den Fehler minimiert. Ausreißende Features, die nicht in die Geometrie passen, können wir dabei beispielsweise mittels des Random Sample Consensus-Algorithmus (RANSAC, *Fischler und Bolles* [1981]) ausschließen [*Schaffalitzky und Zisserman*, 2002].

Wenn wir durch Fehlerminimierung für jede Nahtstelle eine geeignete Transformation gefunden haben, können wir das Bild zusammenfügen und in einen gemeinsamen Bildraum überführen. In unserem Fall ist die geeignete Parametrisierung die Zylinderprojektion, die wir in Abschnitt 3.3.1 betrachtet haben. Wichtig ist bei einem 360°-Panorama, dass die beiden Enden beim Stitching verbunden werden müssen. Ein mögliches Ergebnis der bisherigen Verarbeitungsschritte ist in Abbildung 3.29 gezeigt.

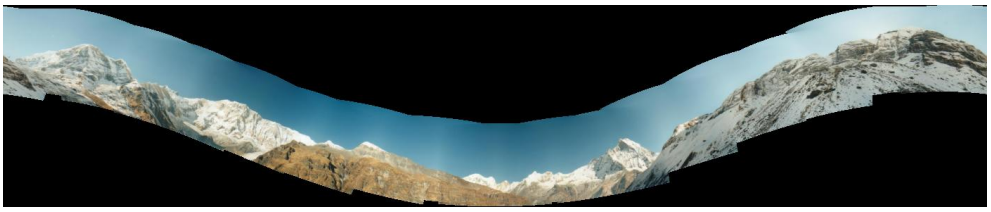


Abbildung 3.29: Ohne globale Optimierung der Horizontlinie erhalten wir zumeist ein derart wellenförmiges Panorama. Unter der Annahme, dass der Fotograf die Kamera waagrecht zum Horizont gehalten hat, kann man das Panorama derart (global) optimieren, dass die y-Achsen der einzelnen Bilder nach oben zeigen. [*Brown und Lowe*, 2007]

Unter der Annahme, dass die dargestellten Berge ungefähr die gleiche Höhe haben, lägen ihre Spitzen auf einer Epipolarlinie des Bildes (vergleiche Abbildung 3.13).

Die geschwungene Form der Epipolarlinie resultiert aus der Zylinderprojektion, die im Gegensatz zur rechtwinkligen Projektion Geraden auf Kurven abbildet: In Abbildung 3.30 sehen wir beispielhaft eingezeichnet zwei Epipole einer Projektion. Wir gehen bei der Darstellung davon aus, dass wir zwei Quellbilder haben. Das eine zeigt die 180° Blickwinkel vor dem Betrachter, das andere den Auschnitt hinter dem Betrachter. Da das Zylinderpanorama geschlossen ist, schneidet die Verbindungsgerade zwischen den beiden Aufnahmezentren den Zylinder zweimal und wir erhalten zwei Epipole.

Abbildung 3.30: Die Abbildung zeigt die Auswirkung der Zylinderprojektion auf beispielhaft eingezeichnete Epipolarlinien. Die „Zentren“ sind die beiden Epipole. [*McMillan und Bishop*, 1995]

Abbildung 3.31 zeigt die Epipolarlinien für den Fall, dass die Epipole nicht auf gleicher Höhe liegen und erklärt damit die Form des Panoramas in Bild 3.29.

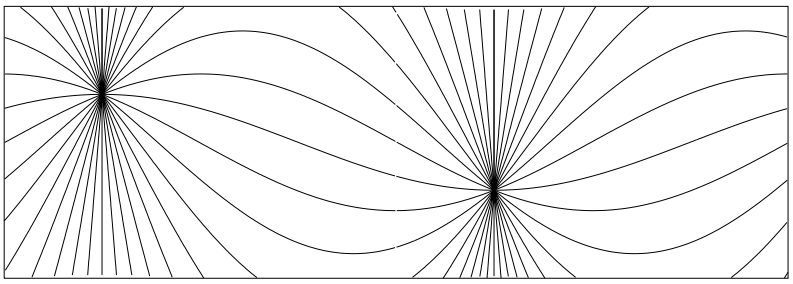


Abbildung 3.31: Die Grafik veranschaulicht das Zustandekommen der Form anhand von Epipolarlinien. [*McMillan und Bishop*, 1995]

Die zusätzliche Einschränkung, dass der Up-Vektor eines jeden Quellbildes nach der Optimierung möglichst nach oben zeigen soll, die Bilder also nicht gedreht werden sollen, führt zu einem ebenen Horizont, wie wir es in Abbildung 3.32 sehen. Voraussetzung für das Funktionieren des Verfahrens ist, dass die Kameras bei der Aufnahme ebenfalls einen waagrechten Horizont abgebildet haben, was zumeist der Fall ist. [*Brown und Lowe*, 2007]

Abbildung 3.32: Mithilfe des globalen Optimierungsconstraints, dass der Up-Vektor auch nach dem Stitching nach oben zeigen muss, erhalten wir dieses Stitchingergebnis. [*Brown und Lowe*, 2007]

Aus den bisherigen Verarbeitungsschritten erhalten wir bei LDR-Quellbildern möglicherweise ein Panorama wie in Abbildung 3.33. Wie wir sehen, sind die Bilder korrekt ineinander transformiert. Da die Belichtung der Aufnahmen divergiert, sind die Grenzen der Einzelbilder deutlich zu erkennen.

Abbildung 3.33: Das Bild zeigt 80 mittels SIFT-Features zueinander registrierte Bilder. Da LDR-Bilder verwendet wurden, sind die Kanten der Einzelbilder deutlich zu erkennen. Durch geeignetes Überblenden lässt sich ein nahtloser Übergang herstellen. [*Brown und Lowe*, 2003]

Da wir mit HDR-Quellbildern arbeiten, ist das dargestellte Fehlen einer geeigneten Überblendung der LDR-Quellbilder für uns idealerweise unnötig: Alle Pixelintensitäten sind gegen einen absoluten Wert kalibriert und somit gibt es keine Belichtungsunterschiede zwischen den HDR-Aufnahmen.

Wir gehen nachfolgend kurz auf ein mögliches Überblendverfahren, das so genannte Multi-Band-Blending, ein. Diese Technik ist im Zusammenhang mit der Geisterentfernung (deghosting), die wir in Abschnitt 2.4.2 bereits angesprochen haben, und im Zusammenhang mit dem Tonemapping, das wir in Kapitel 4 behandeln, von Interesse.

3.6 Multi-Band-Blending

Die Notwendigkeit zur Überblendung bei LDR-Aufnahmen wird aus Abbildung 3.33 und 3.34 deutlich: Die Übergänge zwischen den einzelnen Aufnahmen sollen unsichtbar werden.

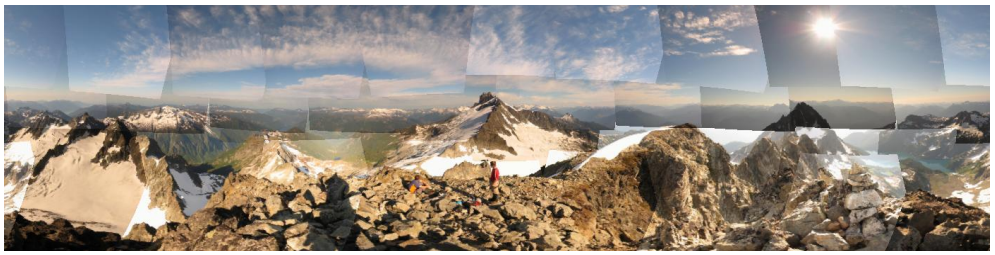


Abbildung 3.34: Die Abbildung zeigt mithilfe von SIFT zueinander registrierte Aufnahmen. [*Brown und Lowe*, 2007]

Brown und Lowe [2007] führen als Vorverarbeitung einen „Gain compensation"-Schritt aus. In diesem Schritt passen sie die Helligkeiten der Bilder bereits aneinander an, indem sie den Fehler der überlappenden Bereiche minimieren. Abbildung 3.35 zeigt das Ergebnis.

Abbildung 3.35: Das Bild zeigt die Komposition der Einzelbilder nach der „Gain compensation". An der Kachel, die die Sonne enthält, ist beispielsweise zu erkennen, dass das Bild aufgehellt wurde. [*Brown und Lowe*, 2007]

Als eigentlichen Überblendalgorithmus verwenden sie anschließend das „Multi-Band-Blending". Beim Multi-Band-Blending werden niedrige/ tiefe Frequenzen über einen großen Bereich, hohe Frequenzen (Details) über einen kleinen Ortsbereich überblendet. Somit bleiben die Details erhalten, während die Grundhelligkeiten und Farben aneinander angeglichen werden. Die sichtbaren Übergänge verschwinden.

Die Theorie hinter dem Verfahren kann ausführlicher in der Veröffentlichung von *Burt und Adelson* [1983] nachgelesen werden. Zur Zerlegung der Quellbilder in verschiedene Frequenzbänder kommt die, uns bereits aus Abbildung 3.24 bekannte, Laplacepyramide zum Einsatz.

Das untere Teilbild von Abbildung 3.36 zeigt das Ergebnis.

Abbildung 3.36: Die Abbildung zeigt das Panorama nach dem „Multiband-Blending". Es sind keine Einzelbildränder mehr zu erkennen. [*Brown und Lowe*, 2007]

3.7 Zusammenfassung

Inhalt dieses Kapitels war das Stitching von mehreren Kameraaufnahmen zu einem Panorama.

Hinführend sind wir auf die für diese Aufgabe relevanten Linsenaberrationen eingegangen (Schärfefehler/ Geometriefehler). Dieser Teil führte unsere Übersicht aus Abschnitt 2.2.2.2 fort. Die Verzeichnung und ihre Korrektur haben wir als Hauptproblem für das Stitching identifiziert und genauer betrachtet.

Den Hauptteil des Kapitels bildeten die Registrierung und Transformation von Einzelbildern zu einem Panorama.

Wir haben dabei mit dem Idealfall begonnen: Alle Aufnahmen werden von einer über dem optischen Zentrum der Linse montierten Kamera aufgenommen.

Ausgehend von diesem Fall haben wir gesehen, dass sich die Verarbeitung in die Phasen Korrespondenzfindung → Stitching → Überblendung aufgliedern lässt. Als Zielprojektion des Stitchings haben wir hier die Zylinderprojektion und die rechtwinklige Projektion betrachtet.

Um vom Idealfall zur Realität zu gelangen, haben wir die Betrachtung um mehrere Kamerastandpunkte bei perspektiveninvarianten Objekten erweitert und dabei die homogenen Koordinaten sowie die Epipolargeometrie eingeführt.

Wir sind auf die für diesen Fall relevanten Algorithmen zur allgemeinen Korrespondenzfindung durch Kreuzkorrelation eingegangen und bei der Betrachtung diverser Optimierungen bei den Verfahren angekommen, die auf der Ermittlung des optischen Flusses basieren.

Durch Einführung nicht perspektiveninvarianter Objekte sind wir schließlich zu einer realistischen Betrachtung gelangt. Dabei sind wir auf ein größeres Problem bei dezentraler Montage der Kameras aufmerksam geworden: Parallaxe. Wir haben gesehen, dass Parallaxe zu inkonsistenter Bildinformation in benachbarten Aufnahmen sowie zu fehlender Bildinformation führen kann.

Da Parallaxe zu fehlerhaften Stitchings führt, haben wir uns mit Korrekturmöglichkeiten befasst. Wir haben dabei die plenoptische Funktion und das Warping kennen gelernt. Außerdem haben wir bereits Ergebnisse eines nur auf den Quellbildern basierenden Algorithmus und daraus resultierende mögliche Fehler gesehen.

Da die featurelosen Algorithmen zur Korrespondenzfindung Probleme mit Perspektivenänderungen haben, sind wir ausführlich der auf einen featurebasierten Algorithmus eingegangen. Am Beispiel von Shift Invariant Feature Transform (SIFT) haben wir wichtige Verarbeitungsschritte identifiziert: Featureextraktion, Featurecharakterisierung und Featureidentifikation. Wir haben gesehen, nach welchen Kriterien man Features extrahieren kann, welche Features dafür geeignet sind und wie sie effizient identifiziert werden können. Auch hier haben wir festgestellt, dass die HDR-Bildinformation ein Vorteil sein kann, da keine Features „wegbelichtet“ werden und korrespondierende Elemente sich daher auf allen Quellbildern befinden.

Mit den notwendigen Grundlagen ausgestattet haben wir das Stitching mithilfe von SIFT betrachtet. Wir haben gesehen, dass wir für ein 360°-Panorama mit waagrechtem Horizont die Bedingung einbauen müssen, dass das Panorama geschlossen wird (Transformation zwischen letztem und erstem Bild) und, dass der Up-Vektor der einzelnen Bilder möglichst auch im gestitchten Panorama nach oben zeigen soll.

Abschließend sind wir auf das Multi-Band-Blending als geeignete Technik zum nahtlosen Überblenden von Einzelbildern bei gleichzeitiger Detailerhaltung eingegangen. Aufgrund des gemeinsamen kalibrierten Wertebereichs existiert dieses Problem für die HDR-Verarbeitung nicht.

Der Algorithmus ist allerdings im Zusammenhang mit der Geisterentfernung (deghosting) in Abschnitt 2.4.2 und dem Tonemapping (Kapitel 4) interessant.

4. Tonemapping

> Chew, if only you could see what
> I've seen with your eyes!
>
> Blade Runner, 1982

Inhalt dieses Kapitels ist das Tonemapping. Das Hauptaugenmerk liegt auf den wahrnehmungspsychologischen und algorithmischen Grundlagen sowie der qualitativen Beurteilung der Verfahren.

Die Analyse der einzelnen Algorithmen ist im Gegensatz zur Analyse in Kapitel 2 stark ergebnisorientiert. Wir werden nicht umfassend auf die individuellen mathematischen Details eingehen. Stattdessen werden wir Bildergebnisse beurteilen, da die tonegemappten HDR-Daten im Gegensatz zu den HDR-Bildern druckbar sind. Dies bietet sich besonders an, da das Zielobjekt des Mappingoperators der Mensch ist. Unsere subjektive Beurteilung des Ergebnisses eines Operators stellt daher ein gutes Maß für dessen Tauglichkeit dar.

Auch mit der Verbreitung von HDR-Diplays (vergleiche hierzu Abschnitt 2.2.6) wird das Tonemapping relevant bleiben, da wir als Menschen nicht in der Lage sind, einen Kontrastumfang von mehr als 10000:1 auf einem Display zugleich wahrzunehmen. Die Kompression der Dynamik wird dann vielleicht zwar nicht mehr von der Displaytechnologie beschränkt (so wie wir heute zum Beispiel auf 255:1 komprimieren müssen) wohl aber vom Rezipienten vor dem Medium.

Wie wir sehen werden, ist es mit Tonemappingalgorithmen möglich Bildergebnisse zu erzielen, die über die menschliche Wahrnehmung hinaus gehen: Wir können beispielsweise einen höheren Kontrast sichtbar machen, als es unsere Wahrnehmung beim Betrachten der Szene in der Realität vermag.

Die in diesem Kapitel vorgestellte Technik bietet daher ein auch künstlerisch nutzbares Mittel, unsere Wahrnehmung direkt zu lenken und zu erweitern.

4.1 Motivation und Grundlagen

Wie wir zu Beginn der Arbeit gesehen haben, ist die Diskrepanz im Dynamikumfang zwischen HDR-Daten und LDR-Anzeigegeräten groß (siehe Tabelle 2.1).

In Kapitel 2 haben wir uns damit beschäftigt, den Kontrastumfang in der realen Welt aufzunehmen. In diesem Teil geht es darum, höherdynamische Daten auf einem LDR-Anzeigegerät darzustellen. Die dazu vorgenommene Abbildung wird als **Tonemapping** bezeichnet.

Der Vorgang des Tonemapping ist vergleichbar mit der Arbeit eines Malers: Durch gezielte Verminderung der Dynamik und die damit verbundene Abstraktion von der Wirklichkeit bildet er etwas ab, so dass genau der Ausschnitt der Realität gezeigt wird, den er vermitteln möchte. Genau dies wollen wir mit dem Tonemapping ebenfalls erreichen – nur ohne den Maler bzw. weitgehend ohne menschlichen Eingriff[111]. Die Herausforderung besteht darin, dass der Mensch –in unserem Fall der Maler– eine sehr komplexe Reduzierung der Dynamik vornimmt und wir müssen versuchen, diese im Rechner nachzubilden. Eine mit der Malerei verwandte Kunstrichtung, die zur Abbildung ein technisches Gerät verwendet und daher unserem Fall noch näher kommt, ist die Fotografie.

Das Problem, vor dem wir stehen, stellt sich seit der Erfindung der Fotografie Anfang des 19. Jahrhunderts[112]: Das Aufnahme- oder Wiedergabemedium bietet einen zu geringen Dynamikumfang für die festzuhaltende Szene. Wir müssen daher den abzubildenden Inhalt geeignet vorverarbeiten, so dass er auf das Medium „passt“.

Anders als bei den Anfängen der Fotografie sind wir mit den in Kapitel 2 vorgestellten Verfahren heute in der Lage, fast den gesamten Kontrastumfang aufzuzeichnen. Das Aufnahmemedium stellt also in Hinsicht auf die Helligkeitsdynamik keine gravierende Einschränkung mehr dar. Ein geeignetes Wiedergabemedium fehlt im Jahr 2008 allerdings noch und wird auch in absehbarer Zeit nicht verfügbar sein, da die Darstellung des kompletten Dynamikumfangs einer sehr hochdynamischen Szene auf dem begrenzten Raum eines Bildschirms –selbst wenn dies technisch möglich wäre– von einem Menschen nicht wahrgenommen werden kann. Gründe dafür lernen wir im Rahmen der wahrnehmungspsychologischen Grundlagen in Abschnitt 4.1.1 kennen.

Mit einer HDR-Aufnahme haben wir viele Messwerte, mithilfe derer wir versuchen können, das nachzubilden, was erfahrene Abbilder wie Maler oder Fotografen aufgrund ihrer Erfahrung und ihrer menschlichen Verarbeitungsmöglichkeiten zu leisten vermögen. In Analogie zur klassischen Fotografie haben wir mit einem HDR-Bild sehr viele sehr gut aufgelöste Helligkeitsmessungen. Der klassische Fotograf musste diese Messwerte einzeln mit einem Belichtungsmesser ausmessen[113].

Unsere Ausgangslage ist messtechnisch besser als diejenige des Fotografen und auch die Verarbeitungsgeschwindigkeit für die große Zahl von Messergebnissen ist bei einem Rechner höher als bei einem Menschen. In den Tonemappingalgorithmen müssen wir demnach „nur“ noch die menschliche Wahrnehmung und die Erfahrung der Abbildenden modellieren. Dazu werden wir uns in Abschnitt 4.1.1 mit der menschlichen Wahrnehmung befassen. Um einen Eindruck davon zu bekommen, was mit Erfahrung gemeint sein könnte, betrachten wir die Arbeit eines Fotografiepioniers, der seine Erfahrung aufgeschrieben hat. Sein Verfahren wurde direkt in einen Tonemappingoperator umgesetzt, wie wir später sehen werden[114]:

Fotografen haben im Laufe der Zeit Methoden entwickelt, wie sie durch geschickte Wahl ihrer Parameter –der Blende und der Belichtungszeit– möglichst viel Kontrast aus der Realität auf den viel kontrastärmeren Film und das noch kontrastärmere Positiv abbilden können

[111] Wie wir sehen werden, können wir bei den meisten Operatoren das Ergebnis durch Veränderung von Parametern beeinflussen.

[112] Die erste Fotografie wurde 1826 durch Joseph Nicéphore Nièpce angefertigt. 1837 benutzte Louis Jacques Mandé Daguerre ein besseres Verfahren, das auf der Entwicklung der Fotos mit Hilfe von Quecksilber-Dämpfen und anschließender Fixierung in einer heißen Kochsalzlösung oder einer normal temperierten Natriumthiosulfatlösung beruhte. Bereits 1835 erfand der Engländer William Fox Talbot das Negativ-Positiv-Verfahren. [Wikipedia]

[113] Moderne Kameras haben eingebaute Belichtungsmesser, berücksichtigen aber normalerweise ebenfalls nur endlich viele Messpunkte („11-Punkt-Messfeld“ etc.).

[114] [*Reinhard et al.*, 2002b]

(vergleiche Tabelle 2.1). Der amerikanische Fotopionier Ansel Adams beschreibt ein solches Verfahren in seinem 1948 begonnenen Werk „Basic Photo“ [*Adams*, 1970]. Wir gehen kurz auf seine **Zonentechnik** ein, da sie ein grundlegendes Vorgehen bei der analogen Aufnahme und damit auch ein mögliches Vorgehen für unsere „digitale Entwicklung“ skizziert.

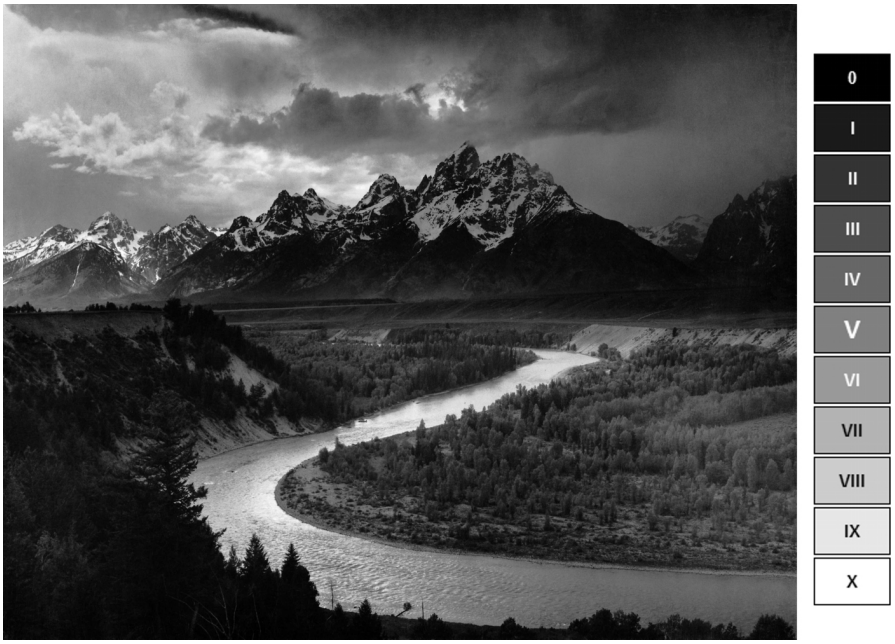


Abbildung 4.1: Im Bild können wir verschiedene Belichtungszonen identifizieren. Beispielsweise ist der Wald vorne eine Zone, der Fluss, das rechte umflossene Landstück... Im Foto gibt es keine größeren Flächen ohne Zeichnung. Rechts sehen wir zum Vergleich die Helligkeiten, die den einzelnen Zonen zugeordnet sind. [Bildquelle: Ansel Adams The Tetons and the Snake River (1942) Grand Teton National Park, Wyoming. National Archives and Records Administration, Records of the National Park Service]

Die aufzunehmende Szene wird mit einem Belichtungsmesser vermessen. Adams unterteilt das Bild in Bereiche ähnlicher Belichtung, so genannte Zonen (vergleiche Abbildung 4.1). Um das Verfahren geeignet anzuwenden, muss sich der Fotograf die Helligkeitswerte der verschiedenen Zonen –insbesondere der hellen– notieren. Die Negativbelichtung (Blende, Belichtungszeit) wird so gewählt, dass der dunkelste Bereich noch Zeichnung hat, denn diese lässt sich später nicht mehr gewinnen.

Sehr verkürzt dargestellt, werden die im Negativfilm enthaltenen Silberhalogenide bei der Belichtung durch das eingestrahlte Licht in elementares Silber reduziert. Die Geschwindigkeit eines Films in *ISO* gibt dabei an, wie schnell dies geschieht. Die entstandenen Kristallkeime bewirken, dass bei der Entwicklung die umliegenden Bereiche schneller zu metallischem Silber reduziert werden als die Bereiche mit unbelichteten Silberionen. Belichten wir nun zu kurz, so entstehen an den Stellen mit geringer Lichteinstrahlung keine Silberkristalle und wir haben folglich keinerlei Information gespeichert[115]. Eine moderate Überbelichtung können wir dagegen durch eine gezielte Verkürzung der Entwicklung bis zu einem gewissen Grad ausgleichen: Da in einer bestimmten Zeit eine bestimmte Menge an Silberionen in Silber verwandelt wird, sind die dunklen Bildstellen des Positivs auch bei verkürzter Entwicklungszeit korrekt entwickelt. Bei einem Negativfilm bedeutet dies, dass die wenigen Silberionen, die von Licht-Photonen getroffen wurden, auch nach der verkürzten Entwicklungszeit bereits vollständig zu Silber reduziert wurden. (Die dunkelsten Bereiche des Positivs sind die lichtdurchlässigsten des Negativs, also diejenigen mit den wenigsten Silberkörnern auf dem Filmstreifen.) Die hellen Bereiche des Positivs sind dagegen durch die zu kurze Entwicklung

[115] Das durch die Belichtung entstandene Muster auf dem Negativfilm nennt man latentes Bild.

noch nicht vollständig lichtundurchlässig entwickelt und zeigen somit noch Details – sie haben noch Zeichnung.

Umgekehrt kann man bei dynamikarmen Szenen auch den Kontrast erhöhen, indem man länger entwickelt. Dadurch entsteht mehr Silber und die hellen Bereiche erscheinen noch heller, der Kontrast also größer, da noch weniger Licht durch den Filmstreifen auf das Positiv fällt. Dies ist natürlich nur dann sinnvoll, wenn die über die gesamte Szene gemessene Dynamik kleiner als diejenige des Films ist.

Bei der Entwicklung des Negativs auf ein Positiv ergibt sich das Problem der Dynamikreduzierung erneut, wie wir in Tabelle 2.1 gesehen haben. Hier arbeitet man mit dem so genannten „Dodging and Burning“. Beim **Dodging** werden die zu dunklen Bildteile bei der Positivbelichtung nach einem Teil der Belichtungszeit abgedeckt, als ob sie auf dem Negativ nicht so lichtdurchlässig abgebildet worden wären. Sie bleiben dadurch auf dem entwickelten Positiv heller. **Burning** bezeichnet den umgekehrten Effekt. Zu helle Bereiche werden hier nachbelichtet, um sie abzudunkeln.

Wie wir gesehen haben, gibt es im analogen Bereich Techniken, die Dynamik durch gezielten Einsatz von Wissen zu erhöhen. Entscheidender Faktor in diesem Verfahren ist der Mensch und dessen Erfahrung, da er die Techniken entsprechend seines Ergebniswunsches einsetzen muss. Beim Tonemapping will man dies mit dem Computer so nachahmen, dass möglichst wenige Eingriffe durch den Benutzer notwendig sind. Mit verschiedenen Ansätzen dazu und den dahinter stehenden Grundlagen wollen wir uns in diesem Abschnitt beschäftigen.

In den meisten Fällen kann das Tonemapping nicht den gesamten Dynamikbereich wiedergeben[116]. Die Kunst des Tonemappingoperators besteht darin, einen geeigneten Ausschnitt der Dynamik zu wählen und diese geeignet zu komprimieren. Was dabei als geeignet anzusehen ist, hängt vom Verwendungszweck ab: Der naive Ansatz, die gesamte Szenendynamik auf den sichtbaren Bereich des Mediums zu komprimieren, führt meist zu einem schlechten Bildeindruck, da der Kontrast zwangsläufig sehr gering sein wird[117] (vgl. Abbildung 4.2). Das Darstellen eines geeignet komprimierten Kontrastbereiches, der größer ist, als das, was wir als Menschen wahrnehmen können, kann dagegen sehr interessant und ästhetisch wirken. Siehe Abbildungen 4.3 und 4.4.

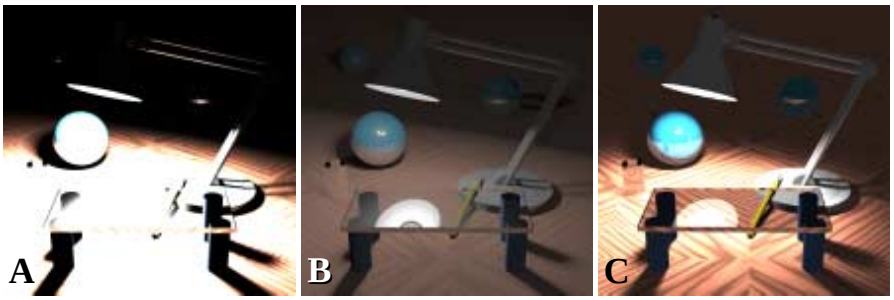


Abbildung 4.2: In Bild A wurde ein Ausschnitt der Dynamik eines HDR visualisiert. Die Größe des Ausschnitts ist von der Wiedergabefähigkeit des Mediums bestimmt. Alle Information, die nicht druckbar ist, wurde verworfen. In Bild B wurde die gesamte Dynamik auf den darstellbaren Bereich komprimiert. Wir sehen deutlich, dass der Kontrast des Bildes sehr gering ist: Das Bild wirkt flau. In Bild C wurde ein Tonemappingoperator angewandt. [*Tumblin et al.*, 1999]

[116] Dies gelingt nur, wenn die Szenendynamik weniger Stufen aufweist als das Ausgabemedium darstellen kann.

[117] Eine Kompression zieht das Histogramm zusammen und verringert somit die Abstände zwischen den Helligkeitswerten und damit den Kontrast.

Abbildung 4.3: „A really cool looking sunset“, condog87. Dieses Bild ist ein gutes Beispiel für unnatürliches Tonemapping: Vor Ort würden wir diese Szene nicht so wahrnehmen (können). [flickr.com]

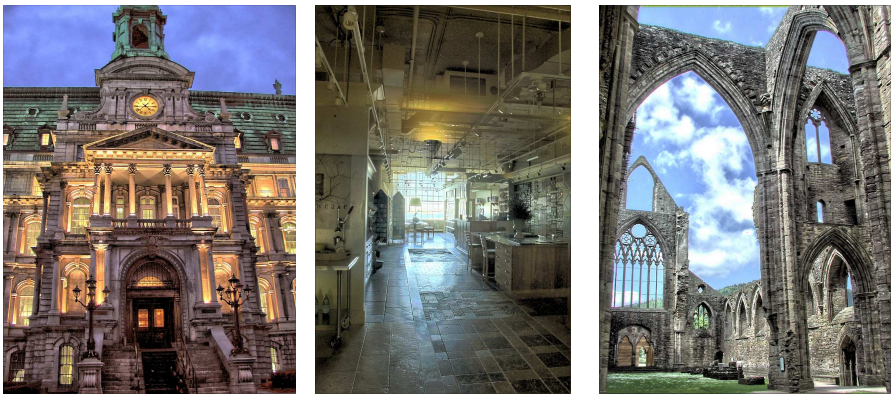


Abbildung 4.4: Beispielbilder eines lokalen Tonemapping-Operators (mehr dazu in diesem Kapitel) von *Chen et al.* [2005]. Der Algorithmus segmentiert das Bild zunächst in Regionen, die wir ebenfalls zu einem gewissen Maß getrennt wahrnehmen, und mappt diese anschließend lokal. [*Chen et al.*, 2005]

4.1.1 Die visuelle Wahrnehmung des Menschen

Die meisten Tonemappingmodelle orientieren sich an unserer visuellen Wahrnehmung. Daher wollen wir diese, in Anlehnung an [*Reinhard et al.*, 2005], kurz betrachten. Wir haben damit in Abschnitt 2.2.1 schon begonnen, sind dort allerdings nur auf die für die Aufnahme relevanten[118] Eigenschaften eingegangen.

Die vorgestellten Erkenntnisse bilden die wahrnehmungspsychologische Grundlage für die anschließend betrachteten Tonemappingoperatoren. Bezüglich der verwendeten Einheiten und Größen sei erneut auf Tabelle 2.2 verwiesen.

Als weitere fachliche Quellen neben *Reinhard et al.* [2005] dienen aus der Physik *Bergmann und Schaefer* [2004b] und aus der Medizin *Grehn* [2006] und *Hansen* [2007]:

Unsere visuelle Wahrnehmung erfolgt mit dem Auge. Wie wir zu Anfang der Arbeit (Abschnitt 2.2.1) schon gesehen haben, nehmen wir nur einen bestimmten Ausschnitt des elektromagnetischen Spektrums wahr. Der Bereich des für uns sichtbaren Lichts liegt zwischen 360nm (Violett) und 830nm (Rot).

Licht fällt über die Pupille ins Auge ein. Nachdem es die Hornhaut und die Linse passiert hat, trifft es auf die Netzhaut. Dort befinden sich zwei Arten von lichtempfindlichen Zellen: Die Stäbchen und die Zapfen[119].

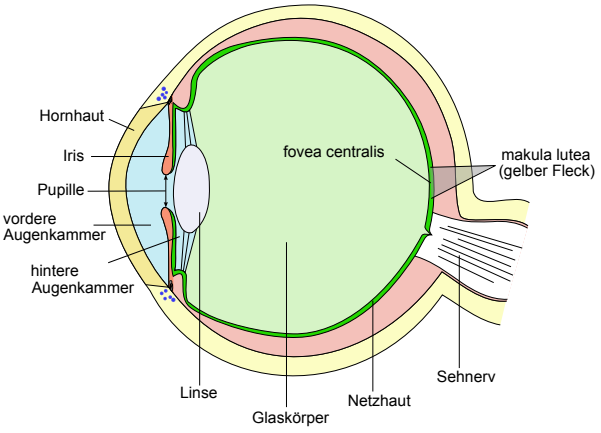


Abbildung 4.5: Der Aufbau des Auges.

Die **Stäbchen** sind die lichtempfindlicheren Zelltypen. Sie können nur Helligkeitsunterschiede wahrnehmen. Nehmen wir hauptsächlich mit den Stäbchen wahr, so sprechen wir von skoptischem Sehen, dem Nachtsehen (10^{-6} cd/ $m^2 - 10$ cd/ m^2).

Den zentralen Bereich der Netzhaut bildet die fovea centralis (siehe Abbildung 4.5). Hier befindet sich der Brennpunkt der Linse und damit der Bereich der schärfsten Abbildung. Im Bereich der fovea centralis befinden sich keine Stäbchen. Deshalb sehen wir bei Dunkelheit (mit den Stäbchen) nicht so scharf wie bei Helligkeit – wir sehen peripher.

Die Signalisierung der Lichtreize zum Gehirn erfolgt über so genannte Ganglienzellen. Diese verarbeiten die Information vor. An eine Ganglienzelle sind mehrere Stäbchen angeschlossen. Dies macht die Lokalisation eines von den angeschlossenen Stäbchen ausgehenden Helligkeitsreizes ungenau.

Mit zuehmender Lichtintensität sind die Stäbchen immer weniger in der Lage, Helligkeitsunterschiede wahrzunehmen. Sie beginnen chemisch zu übersättigen. In der Übergangsphase, wenn beide Sehzellen aktiv sind, spricht man vom mesoptischen Sehen (10^{-2} cd/ m^2 –

[118] Für die Aufnahme war uns lediglich wichtig, keine wahrnehmbare Information bei der Aufzeichnung zu vergessen.
[119] Pro Auge besitzen wir ca. 7 Mio. Zapfen und 120 Mio. Stäbchen [*Grehn*, 2006, 225].

10 cd/ m^2). Das Tagsehen heißt photoptisches Sehen (10^{-2} cd/ $m^2 - 10^8$ cd/ m^2). Es erfolgt primär mit den **Zapfen**.

In der fovea centralis befinden sich nur Zapfen, die jeweils mit einer einzigen Ganglienzelle verbunden sind[120]. Das damit mögliche scharfe Sehen nennt man daher foveales Sehen.

Die Zapfen sind weniger lichtempfindlich. Es gibt Zapfen für den kurzen (blau), mittleren (grün) und langen (rot) Wellenbereich (vgl. Abbildung 2.6). Mit ihrer Hilfe sind wir in der Lage, Farben wahrzunehmen. [*Grehn*, 2006, 226ff] [*Devlin et al.*, 2002]

Die elektrischen Reize in den Sehzellen entstehen durch einen photochemischen Prozess. Dabei wird der Zustand des in der Zelle befindlichen Stoffes Rhodopsin mithilfe von Licht geändert – das Rhodopsin bleicht aus. Die entstehende Spannung wird noch im Auge vorverarbeitet und dann an das Gehirn weitergeleitet (siehe Grafik 4.6).

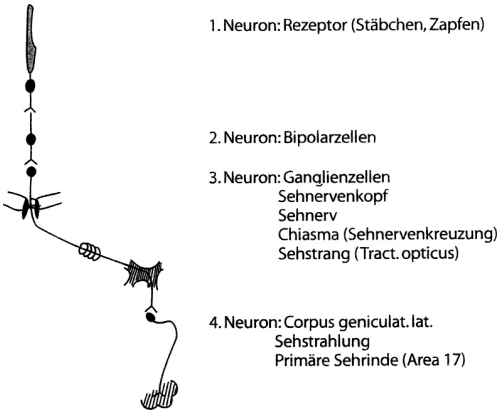


Abbildung 4.6: Die Signalverarbeitung des Helligkeitsreizes vom Rezeptor bis zum Gehirn. [*Hansen*, 2007, 45]

Wie unsere anderen Sinne reagiert unsere visuelle Wahrnehmung *relativ* auf Reize: Die Beleuchtung eines Autos fällt uns bei Tag kaum auf – der Unterschied zum Umgebungslicht ist gering. Bei Nacht dagegen sticht sie aus dem Umgebungslicht hervor, obwohl sich die absolute Helligkeit der Lampe nicht ändert.

Der Vorgang der Anpassung an die jeweilige Umgebungshelligkeit nennt sich **Adaption**. Grafik 4.7 zeigt die Helladaption. Grafik 4.8 veranschaulicht die Dunkeladaption. Neben den unterschiedlichen finalen Anpassungen der Personen verschiedenen Alters sind die Anpassungszeiten interessant:

Die **Helladaption**, also die Adaption von einer dunklen Umgebung an eine helle, geht innerhalb von wenigen Sekunden von statten (vergleiche auch Grafiken 4.7 und 4.10).

Die Regeneration des Rhodopsins für die **Dunkeladaption** nimmt mehr Zeit in Anspruch. Sie ist nach ungefähr 5 Minuten weitgehend abgeschlossen. Eine völlige Anpassung dauert ca. 45 Minuten (vergleiche auch Grafik 4.8).

Wir kennen den Effekt, wenn wir im Sommer aus der gleißenden Sonne in einen dunklen Raum kommen: Es dauert deutlich länger, bis wir wieder etwas sehen als beim Herausgehen ins Freie. Die für das Hellsehen zuständigen Zapfen passen sich insgesamt schneller an neue Lichtverhältnisse an als die Stäbchen. [*Hansen*, 2007, 42f]

Abhängig von der adaptierten Gesichtsfeldleuchtdichte werden wir geblendet. Abbildung 4.9 zeigt die dazu benötigte Helligkeit auf. Die **Blendung** wird bei einigen Tonemappingoperatoren berücksichtigt.

[120] Man spricht hierbei auch von einer Konvergenz von 1:1.

Abbildung 4.7: Simulierte Helladaption im Alter von 20 und 70 Jahren. [*Irawan et al.*, 2005]

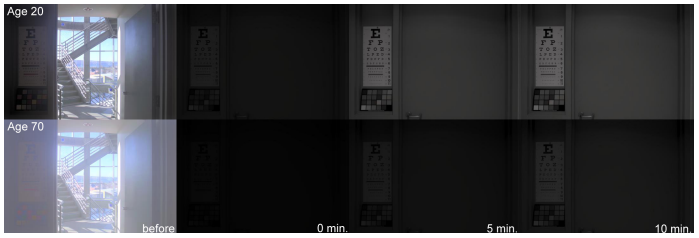


Abbildung 4.8: Simulierte Dunkeladaption im Alter von 20 und 70 Jahren. [*Irawan et al.*, 2005]

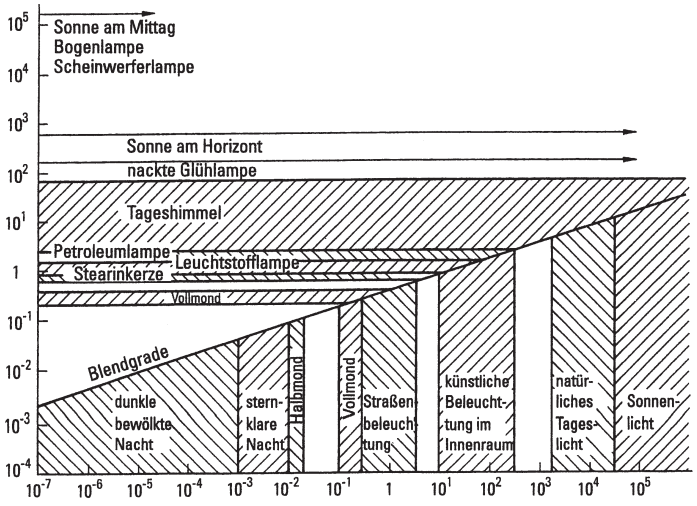


Abbildung 4.9: Die Abbildug zeigt auf der Abszisse die adaptierte Gesichtsfeldleuchtdichte in lx/ m^2. Diese ist abhängig vom Umgebungslicht. Auf der Ordinate sind die wahrgenommenen Helligkeiten eines Objektes angezeichnet. Oberhalb der Blendgeraden tritt jeweils Blendung auf. [*Bergmann und Schaefer*, 2004b]

Die Regulierung unserer Helligkeitswahrnehmung erfolgt hauptsächlich mithilfe der Stäbchen und Zapfen. Unsere Pupille ist lediglich in der Lage, die einfallende Lichtmenge um einen Faktor <16 zu vermindern. Bei einer Dynamik von bis zu 10^{10} : 1 trägt sie nur wenig zur Helligkeitsregulierung bei. [*Reinhard et al.*, 2005]

Die Wahrnehmungspsychologen *Stevens und Stevens* [1963] haben die Adaption des Auges experimentell untersucht. Ihre Arbeit und diejenige von *Blackwell* [1981] (siehe unten) bilden die Grundlage einiger Tonemappingoperatoren.

Stevens und Stevens [1963] ließen ein Auge ihrer Probanden auf Dunkelheit adaptieren, das andere auf helles Sehen. Anschließend präsentierten sie den Augen Reize und ließen die Testteilnehmer diese bewerten. Die entstandenen Helligkeitsempfindungskurven sind in Grafik 4.10 rechts wiedergegeben. Das linke Diagramm zeigt, wie sich die Wahrnehmung während der Adaptierung verändert und auf welchen Wert sie sich stabilisiert.

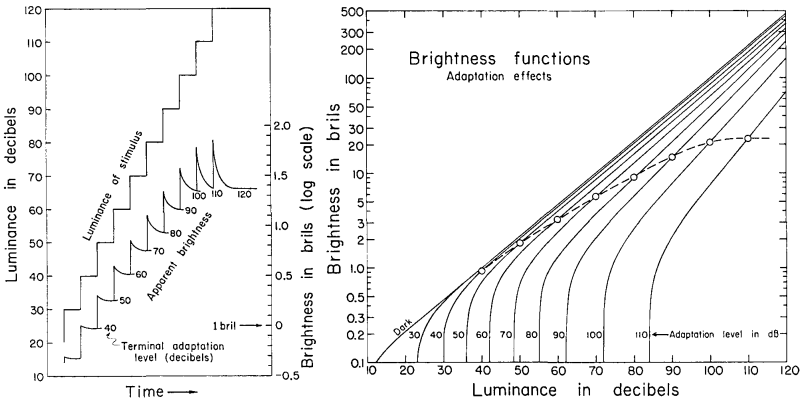


Abbildung 4.10: Die Grafiken zeigen die Helligkeitswahrnehmung in Abhängigkeit der Adaption des Auges. **Links** sehen wir eine Veranschaulichung des Versuchsablaufes und damit der Adaption (ohne korrekte Zeiten). Die Treppenfunktion zeigt die präsentierten Reizintensitäten. Darunter befindet sich die jeweils wahrgenommene Intensität. Man sieht, wie die wahrgenommene Helligkeit während der Adaption sinkt. Die finalen Adaptionszustände bilden eine logarithmische Kurve. Im **rechten** Log-log-Diagramm wird dies an der gestrichelten Linie ebenfalls deutlich. Die einzelnen Kurven veranschaulichen hier die wahrgenommenen Helligkeiten unter verschiedenen Adaptionsstufen. Die gestrichelte Linie zeigt an, welcher Helligkeitswert nach vollständiger Adaptierung wahrgenommen wird. [*Stevens und Stevens*, 1963]

Unsere Reaktion auf Lichtreize erfolgt relativ zum Adaptionszustand. Grafik 4.12 zeigt, um wieviel ein Reiz relativ stärker als das Umgebungslicht sein muss, um wahrgenommen zu werden.

Wie wir sehen, ist das Verhältnis von Umgebungshelligkeit zu gerade noch wahrnehmbarer Helligkeitsdifferenz über weite Teile konstant (Gerade). Dies besagt das Webersche Gesetz von 1834:

$$\frac{\Delta I_b}{I_b} = \frac{\Delta L}{L} \approx 1\%$$

Je heller es ist, desto größer muss ein Helligkeitsunterschied sein, damit wir ihn wahrnehmen können (siehe auch Grafik 4.11). Den gerade noch wahrnehmbaren Unterschied bezeichnet man als „Just Noticable Difference“, abgekürzt JND. Die gerade noch wahrnehmbare Helligkeitsdifferenz wird in so genannten „Threshold versus Intensity“-Diagrammen (TVI), wie wir eines in Grafik 4.12 sehen, abgetragen. Die experimentelle Basis für JND und TVI wurde von *Blackwell* [1981] gelegt. [*Devlin et al.*, 2002]

Projizieren wir den Bereich der Kurven in Abbildung 4.11, in dem wir etwas wahrnehmen, auf die Abszisse, so erhalten wir den gleichzeitig wahrnehmbaren Dynamikumfang. Wie wir

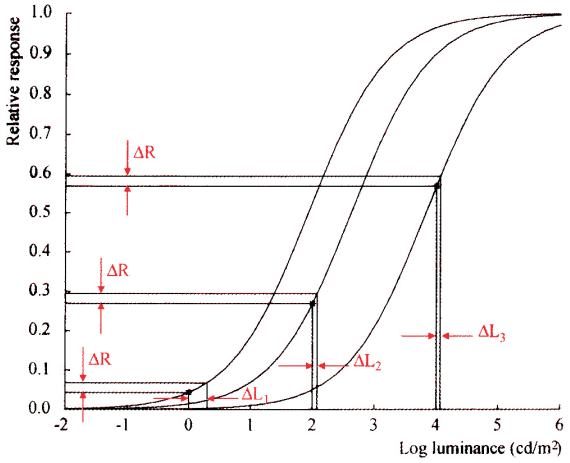


Abbildung 4.11: Die Abbildung zeigt drei Photorezeptorresponsekurven bei drei verschiedenen Hintergrundbeleuchtungen L_1, L_2, L_3 – jeweils markiert durch den Stern auf der jeweiligen Kurve. Wie wir sehen, liegen die Hintergrundbeleuchtungswerte jeweils um einen Faktor 100 (2 log-Einheiten) auseinander. Die Sterne repräsentieren jeweils das Umgebungslicht und dessen Reizintensität. Wenn wir sie verbinden, erhalten wir die gestrichelte Linie in der rechten Abbildung von Grafik 4.10. Die Kurven stellen die Responsekurve des Auges nach erfolgreicher Adaptierung dar. Die ΔL_i veranschaulichen, welcher Helligkeitsunterschied jeweils dem gleichen Reiz entspricht. Der notwendige Helligkeitsunterschied auf der logarithmischen Abszisse wird immer größer, je heller unsere Wahrnehmung adaptiert ist (siehe auch Abbildung 4.12). [*Reinhard et al.*, 2005, 203]

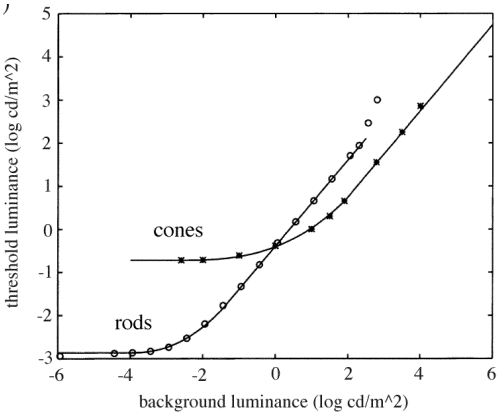


Abbildung 4.12: Die Grafik trägt die kleinsten wahrnehmbaren Helligkeitsdifferenzen ΔL gegen die Gesamthelligkeit der Szene auf. Sie veranschaulicht die Unterschiedsempfindlichkeit des Menschen. Die Grafik ist ein Threshold versus Intensity (TVI)-Diagramm. (rod =Stäbchen, cone = Zapfen). [*Pattanaik et al.*, 1998]

sehen, beträgt dieser ungefär 3 logarithmische Einheiten, also ein Verhältnis von 1000:1. Die Wahrnehmungsschwelle –also die Lage der Kurven– kann von 10^{-6} bis über 10^{2} adaptiert werden [*Hansen*, 2007, 43].

Zusätzlich zur Helligkeitsadaption findet auch eine **chromatische Adaption** statt. Diese führt dazu, dass wir Objekte gleicher Farbe auch bei unterschiedlicher Beleuchtung gleichfarbig wahrnehmen. *Durand und Dorsey* [2000] schreiben dazu, dass die menschliche Wahrnehmung fähig ist, die Beleuchtungsfarbe aus der wahrgenommenen Farbe herauszufiltern. Die wahrgenommene Farbe entspricht so nicht der absoluten Farbe.

Werden Szenen der Realität abgebildet, so muss dies berücksichtigt werden, da in der Aufnahme der Kontext, aufgrund dessen die entsprechend chromatische Adaption stattfand, fehlt. Der entsprechende digitale Vorgang ist der Weißabgleich. Dabei wird festgelegt, welche Spektralkombination auf der Aufnahme als Weiß dargestellt werden soll. Bei falsch eingestelltem Weißabgleich bekommt die Aufnahme einen Farbstich, der nachträglich nicht mehr zu korrigieren ist, da die Farbwerte entsprechend (falsch) quantisiert wurden.

Das Auge führt zu jeder Zeit komprimierende Abbildungen aus – wir „tonemappen“ die Realität beständig. Das Signal-/ Rauschverhältnis, also die Anzahl der diskreten wahrnehmbaren Signalstufen, der Nervenverbindung zwischen Auge und Gehirn beträgt nämlich nur 32:1 [*Reinhard et al.*, 2005].

Wie wir in diesem Abschnitt gesehen haben, funktioniert die Kompression vor allem durch Adaptierung: Wir stellen uns auf ein bestimmtes Helligkeitsniveau ein und nehmen dann relativ zu diesem Niveau wahr.

4.1.2 Ein generelles Tonemapping-Modell

Ziel des Tonemappings ist häufig, eine Szene möglichst so zu visualisieren, wie ein Betrachter sie in der Realität sehen würde. Dies folgt aus dem Modell, das **Tumblin und Rushmeier** in der ersten[121] Veröffentlichung zum Tonemapping [*Tumblin und Rushmeier*, 1993] aufstellen:

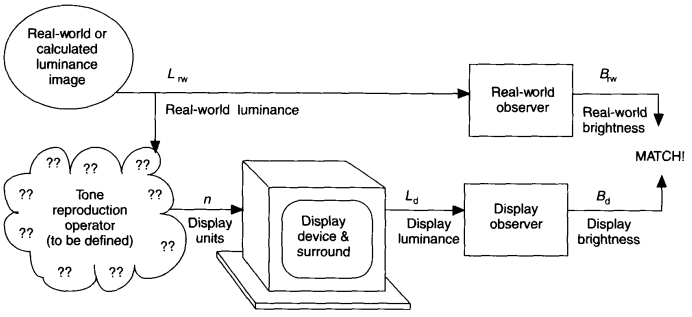


Abbildung 4.13: Die Grafik beschreibt die Aufgabe, die Tumblin und Rushmeier für das Tonemapping sehen. [*Tumblin und Rushmeier*, 1993]

Ein wichtiger und zumeist unbekannter Faktor dabei ist die Umgebung des Ausgabegerätes („Display device & surround“). Dieser Kontext, der beispielsweise die Wand, das Büro und

[121] Eine Operation, die das tut, was hier als Tonemapping bezeichnet wird, findet sich schon in *Oppenheim et al.* [1968]. *Miller und Hoffman* [1984] beschäftigen sich wie *Tumblin und Rushmeier* [1993] mit gerenderten Szenen. Sie verwenden bereits mehrere verschieden belichtete (analoge) Aufnahmen, um eine Light Map der Realität aufzunehmen (vgl. Kapitel 2). Für das Tonemapping relevant ist ihre Erkenntnis, dass es für einen realistischen Eindruck eines mit solch einer Light Map gerenderten Bildes notwendig ist, einen deutlich größeren Teil des Lichtspektrums bei der Berechung zu berücksichtigen, als für die Ausgabe benötigt wird.

Ähnliches sein kann, beeinflusst unsere Wahrnehmung maßgeblich: Ein eher blasses Bild hat vor einer grellen Wand eine andere Wirkung als vor einer grauen. Auch der Bilderrahmen ist entscheidend (siehe auch Abbildung 2.7). Wie wir im vorangegangenen Kapitel gesehen haben, adaptieren wir je nach Umgebung anders. Experimente zur qualitativen Beurteilung von Tonemappings finden daher oft in abgedunkelten Räumen statt, in denen es kein, den Sinneseindruck beeinflussendes, Beiwerk gibt (z.B. [*Ledda et al.*, 2005]). Mithilfe so genannter Color Appearance Models (CAM) kann die Umgebung bei der Ausgabe berücksichtigt werden (siehe Abschnitt 4.2) [*Akyüz und Reinhard*, 2006].

Ausgehend von Abildung 4.13 lassen sich verschiedene allgemeine Modelle zum Vorgehen beim (realistischen) Tonemapping aufstellen. In Grafik 4.14 betrachten wir dasjenige von *Pattanaik et al.* [2000]. Im ersten Schritt des Modells („Adaption Model“) werden aus den

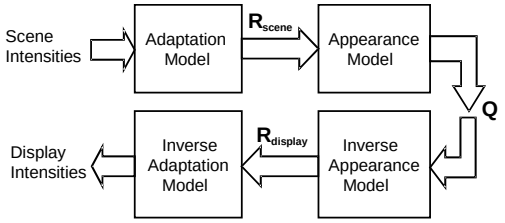


Abbildung 4.14: Generisches Tonemappingmodell. Der obere Teil modelliert die menschliche Wahrnehmung. Der untere Teil bereitet das Ergebnis für die bildschirmvermittelte Präsentation auf. [*Pattanaik et al.*, 2000]

Szenenintensitäten („Scene Intensities“) die vom Auge wahrgenommenen Reize R_{scene} abgeleitet. Das „Appearance Model“ bildet die menschliche Reizverarbeitung nach und beschreibt mit Q das, was wir aufgrund des angewandten Modells wahrnehmen werden. Da wir diese Information nicht direkt in das Gehirn des Betrachters schicken können, benötigen wir den unteren Zweig. Das „Inverse Appearance Model“ passt Q an das Ausgabegerät an und liefert eine an das bestimmte Anzeigegerät angepasste Ausgabe R_{display}. Im „Inverse Adaption Model“ könnte eine Anpassung an die Rezeptionsbedingungen am Darstellungsort erfolgen. [*Pattanaik et al.*, 1998]

Reinhard et al. [2005] bezeichnen die erste Stufe als „Forward adaptation model“, da hier aus den dargebotenen Intensitäten die für uns relevanten extrahiert werden. Den zweiten Schritt bezeichnen sie als „Tone Mapping“ und die beiden unteren Schritte fassen sie zum so genannten „Inverse adaptation model“ zusammen, bei dem die allgemeinen tonegemappten Daten Q für das konkrete Anzeigegerät aufbereitet werden.

Zweck der Präsentation der Modelle an dieser Stelle ist es, uns die grundlegenden Verarbeitungsschritte beim Tonemapping vor Augen zu führen. Wenn ein Algorithmus dieser Modulstruktur insbesondere im unteren Zweig ([*Pattanaik et al.*, 1998]) folgend aufgebaut ist, kann er leicht an neue Gegebenheiten, wie beispielsweise HDR-Displays, angepasst werden.

4.1.3 Mathematische Modelle für die Wahrnehmung

Ziel des realistischen Tonemappings ist es, unsere Wahrnehmung nachzubilden und uns einen möglichst realistischen Bildeindruck zu liefern (vergleiche Abbildung 4.13). Zu diesem Zweck wurden mehrere Modelle auf Basis der in Abschnitt 4.1.1 betrachteten Erkenntnisse entwickelt.

Devlin et al. [2002] wie auch andere Autoren bemerken, dass das menschliche Sehen noch nicht vollständig erforscht ist. DEs in ein mathematisches Modell zu fassen, wird dadurch umso schwerer.

Die vorhandenen Modelle bilden nur Teilschritte unserer Wahrnehmung ab: Das Modell von *Tumblin und Rushmeier* [1993] befasst sich vor allem mit einer realistischen Modellierung des Helligkeitseindrucks. *Ward* [1994] legt besonderen Wert darauf, dass der wahrnehmbare Kontrast und damit die Details erhalten bleiben. *Ferwerda et al.* [1996] modellieren davon ausgehend die zeitliche Adaption des Sehens...

Die gesamte Komplexität unserer visuellen Wahrnehmung erfasst kein Modell. [*Devlin et al.*, 2002]

Zur Veranschaulichung wollen wir kurz auf eine mögliche Tonemappinggleichung eingehen: Wie wir in Abschnitt 4.1.1 gesehen haben, erfolgt unsere Helligkeitswahrnehmung über eine chemische Reaktion in den Stäbchen und Zapfen.

Die Korrelation zwischen einfallendem Licht und dadurch erzeugtem Reiz lässt sich mithilfe der Michaelis-Menten-Gleichung von 1913 beschreiben. Michaelis-Menten haben festgestellt, dass in Enzymen die Reaktion einen Sättigungseffekt erfahren kann (vgl. gestrichelte Linie in Grafik 4.10 rechts).

Auf die visuelle Wahrnehmung übertragen beschreibt die Gleichung den Zusammenhang zwischen der einfallenden Lichtintensität und dem daraus entstehenden Lichtreiz:

$$\frac{R}{R_{max}} = \frac{I^n}{I^n + \sigma_b^n} \tag{4.1}$$

R ist die Reizstärke, I die Lichtintensität. σ ist der Wert, bei dem halbe Sättigung, also eine Response von $\frac{R_{max}}{2}$ erwirkt wird. n gibt die Empfindlichkeit an und liegt normalerweise in $[0,7; 1]$. Der angestellte Index b symbolisiert, dass der Wert vom Umgebungslicht, auf das das Auge gerade adaptiert ist, abhängt. (Die rechte Seite der Gleichung ist auch als Naka-Rushton-Gleichung bekannt [*Naka und Rushton*, 1967].)

Die vorgestellte Gleichung ist ein Modell für einen Teilaspekt unserer Wahrnehmung. Sie eignet sich bereits in dieser Form zum automatischen Tonemapping [*Reinhard et al.*, 2005, 198].

Der Wertebereich liegt im Intervall $[0; 1]$. Dies ermöglicht eine leichte Skalierung auf den konkreten Ausgabebereich eines Ausgabegerätes (siehe auch Abbildung 4.14). Die Gleichung hat einen logarithmisch-linearen Verlauf über nahezu 4 log-Einheiten im log-Linear-Diagramm 4.11. Der Hintergrundbeleuchtungswert liegt innerhalb dieses log-linearen Bereichs. Dies ermöglicht eine einfache und einheitliche Implementierung.

In [*Reinhard et al.*, 2005, 208ff] werden einige weitere Modelle aufgezählt. Die meisten Tonemappingmodelle führen zu einer mathematisch zu 4.1 sehr ähnlich gehaltenen Formel mit leicht veränderten Koeffizienten. Ohne weitere Hintergründe zum Modell ist die Formel aussagenschwach. Daher wollen wir die weiteren Modelle nur kurz erwähnen. Die Funktionsweise ist für den Leser, der sich in den jeweiligen Publikationen näher informieren möchte, in der Klammer angegeben. Die Modelle sind aufgrund eines grundlegenden Unterscheidungskriteriums gruppiert: Die erste Gruppe behandelt das Auge als eine geschlossene Einheit. Die zweite Gruppe modelliert explizit die beiden Sinneszellarten Stäbchen und Zapfen.

Schlick [1994] (rationale Funktion), *Tumblin et al.* [1999] (S-Kurve) und {*Reinhard* [2002], *Reinhard et al.* [2002b]} betrachten das Auge als Einheit innerhalb ihres Modells. *Pattanaik et al.* [1998] (Gain), *Pattanaik et al.* [2000] & *Reinhard und Devlin* [2005b] (Adaptionsmodell der Fotorezeptoren) behandeln die Stäbchen und Zapfen getrennt.

Anstelle der genauen Formel betrachten wir die Abbildungsfunktion einiger Formeln in Grafik 4.15. Die log-log-Diagramme zeigen auf, welche Szenenhelligkeiten auf welche Displayhelligkeiten abgebildet werden. Der dickere Punkt markiert dabei die simulierte Adaptionshelligkeit des Auges. Aufgrund dieses Wertes wird die entsprechende Abbildungsgerade ausgewählt. In Abschnitt 4.2 werden wir die Bildergebnisse einiger Operatoren sehen. Ein

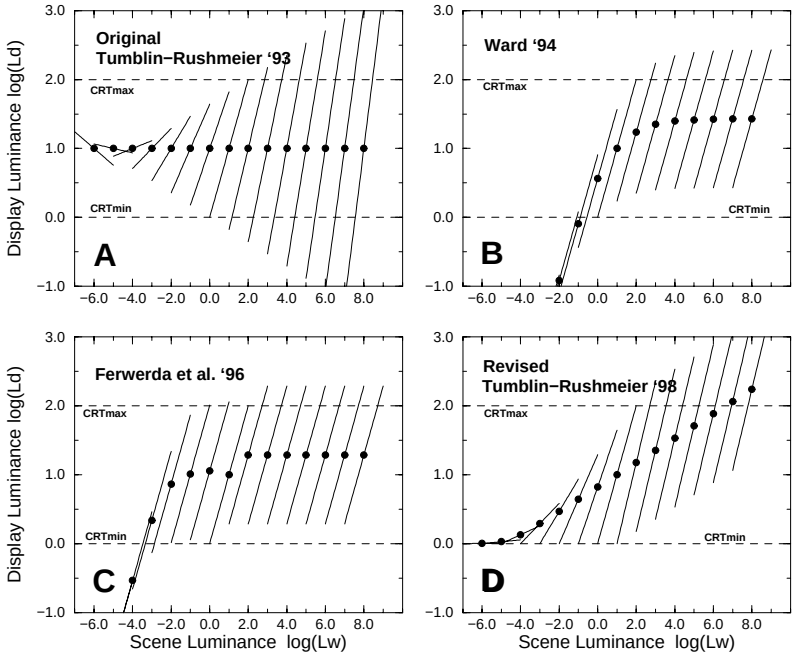


Abbildung 4.15: Ein Vergleich verschiedener Tonemappingoperatoren. Die Geraden geben an, wie bei der jeweiligen angenommenen Adaptionsstufe unseres Auges –durch den Punkt symbolisiert– von gemessener HDR-Szenenhelligkeit („Scene Luminance“) auf eine Bildschirmhelligkeit („Display Luminance“) gemappt wird (Gerade). [*Tumblin et al.*, 1999]

Vergleich von Tonemappingoperatoren bis 2002 findet sich in [*Devlin et al.*, 2002]. Eine Übersichtstabelle aus der Publikation ist in der Tabelle in Abbildung 4.18 wiedergegeben. Eine ausführlichere Beschreibung, die auch mathematisch auf die Algorithmen eingeht, findet sich in [*Reinhard et al.*, 2005].

Der bisher vorgestellte Ansatz geht von der Theorie über die menschliche Wahrnehmung aus und modelliert darauf basierend einen Tonemappingoperator („bottom-up“).

Ebenso ist es möglich von einem gewünschten (Bild-)Ergebnis ausgehend („top-down“) zu modellieren oder das Modell direkt gemäß dem erzielten Ergebnis anzupassen. Durch dieses Vorgehen können wahrnehmungserweiternde, „unrealistische“ Tonemappingoperatoren entstehen. Sie sind in der Lage, uns die Realität anders zu zeigen, als wir sie wahrnehmen können (siehe Abbildung 4.3).

4.1.4 Klassifikation von Tonemappingoperatoren

Es gibt orts-, frequenz- und gradientenbasierte Tonemappingoperatoren – entsprechend dem Raum, auf dem sie arbeiten.

Die **ortsraumbasierten** Tonemappingoperatoren lassen sich aufgrund ihrer Funktionsweise in zwei große Klassen unterteilen: Globale und lokale Operatoren. [*Reinhard et al.*, 2005] [*Ledda et al.*, 2005] [*Devlin et al.*, 2002]

- Als **globale Operatoren** (single-scale, spatially uniform) bezeichnet man solche, die jedes Pixel unabhängig von den anderen betrachten. Globale Operatoren wenden auf alle Pixel dieselbe Funktion (Kurve) an. Als Veränderliche dient nur die Pixelhelligkeit des gerade zu mappenden Pixels. Als weiteren Parameter verwenden sie beispielsweise die Grundhelligkeit des gesamten Bildes.

  Globale Operatoren sind sehr schnell und können zum Teil in Echtzeit angewandt werden. Nachteil ist, dass sie bei Bildern mit großem Kontrast keine so realistischen und detailreichen Ergebnisse liefern wie lokale Operatoren. Grund dafür ist unter Anderem die verwendete globale Grundhelligkeit. Unsere Wahrnehmung bestimmt die Helligkeit, auf die wir adaptieren nicht derart global sondern anhand der lokalen Umgebung. Daher nehmen wir die echte Szene anders wahr als sie uns der globale Operator präsentiert.

  Zu den globalen Operatoren zählen auch solche, die nur auf dem Histogramm arbeiten.
- Die **lokalen Operatoren** (multi-scale, spatially varying) mappen ein Pixel anhand seiner Umgebung: In die Abbildung eines Pixels fließen die Werte der Pixelumgebung ein. Ein Pixel mit gleichem Absolutwert kann so je nach Umgebung auf verschiedene Displayhelligkeiten abgebildet werden. Vorteil ist eine bessere Detailschärfe, da lokale Kontraste erhalten bleiben. Nachteil ist der höhere Berechnungsaufwand. Außerdem kann es beim lokalen Vorgehen zu Artefakten kommen, wie wir in Abschnitt 4.1.5 sehen werden.

Die zweite Gruppe von Tonemappingoperatoren arbeitet im (Orts-)**Frequenzraum** und im **Gradientenraum** [*Reinhard et al.*, 2005, 325ff]. Diese Repräsentation der Bildinformation erlaubt andere Manipulationen:

- Mithilfe der **frequenzraumbasierten** Operatoren[122] ist es möglich, verschiedene Frequenzteile zu trennen und gesondert zu behandeln. Schon in [*Oppenheim et al.*, 1968] wird vorgeschlagen, die hohen Frequenzanteile weniger stark zu komprimieren als die niederfrequenten. Dadurch bleiben Details erhalten.

  Mithilfe entsprechender Filterung kann eine Trennung von ambientem (Grundhelligkeit) und spekularem (Glanzlichter) Lichtanteil eines Objektes versucht werden. Ersterer wird von uns schwächer wahrgenommen und kann daher stärker in seiner Dynamik komprimiert werden. Eine saubere Trennung in die beiden Komponenten gelingt zumeist nur bei computergenerierten Bildern, bei denen die Komponenten ohnehin getrennt berechnet werden.

  Reinhard et al. [2005] schreiben, *Horn* [1974] folgend, dass die Helligkeit L_v eines Bildpunktes sich unter bestimmten Umständen als Produkt des selbst emittierten Lichtes E_v und des reflektierten Lichtes r auffassen lässt:

  $$L_v = E_v r \tag{4.2}$$

  Durch Logarithmieren der Gleichung lassen sich die Multiplikatoren in zwei additive Bestandteile zerlegen, die dann getrennt voneinander betrachtet werden können. Man spricht dabei von **homomorphem Filtern**:

  $$log(L_v) = log(E_v) + log(r) \tag{4.3}$$

[122] Eine Transformation in den Frequenzraum und zurück in den Ortsraum ist die Fourieranalyse.

Ein anderes Verfahren zur Zerlegung in zwei Frequenzbereiche wird von *Durand und Dorsey* [2000] vorgeschlagen. Beim so genannten **bilateralen Filtern** wird das Bild mit einem Filterkern gefaltet, der das Bild glättet, also Helligkeitsunterschiede zu einem gewissen Grad einebnet, gleichzeitig aber die Kanten bewahrt. Die beiden entstehenden Schichten werden hier „base layer“ und „detail layer“ genannt. Sie entsprechen nicht unbedingt den zuvor betrachteten Schichten aus ambientem und spekularem Licht. Näheres in Abschnitt 4.2.

- Der **Gradient** gibt in unserem Fall Richtung und Stärke (Betrag) der größten Helligkeitsänderung an. Die Aufgabe, die Helligkeitsdynamik eines Bildes zu reduzieren, lässt sich durch entsprechendes Verändern des Gradientenbildes erreichen. Vorteil gegenüber der direkten Operation auf den Helligkeitswerten ist, dass wir eine Repräsentation der Helligkeitsunterschiede vorliegen haben. Wie wir in Abschnitt 4.1.1 gesehen haben, sind es gerade diese Unterschiede, auf die unsere Wahrnehmung besonders reagiert.

  Damit auch auf dem tonegemappten Bild feine Details erhalten bleiben, ist es sinnvoll, große Gradienten im Betrag ungleichmäßig stärker zu verringern als kleine.

  Das Gradientenbild entsteht gerade durch Auswertung der Richtungsableitungen. Nach erfolgter Veränderung dieses Bildes wird das Gradientenfeld wieder zu einem Helligkeitsintensitätenbild aufintegriert. Dies erfolgt durch Lösen einer Poissongleichung. Dabei entstehen Ungenauigkeiten. Näheres findet sich in *Horn* [1974].

4.1.5 Allgemeine algorithmische Grundlagen

Alle Tonemappingoperatoren verändern die Verteilung der Dynamik im Bild. Die meisten reduzieren dabei die Dynamik. Daher sind ihnen viele Voraussetzungen und Verarbeitungsschritte gemein. Einige davon wollen wir betrachten, bevor wir genauer auf einzelne Algorithmen eingehen. Wir erlangen damit Grundwissen zur Beurteilung von konkreten Tonemappingalgorithmen.

Für ein adäquates Tonemapping ist es entscheidend, dass die **Helligkeitswerte absolut kalibriert** sind. Ohne eine solche Kalibrierung ist es für die Tonemappingoperatoren nicht entscheidbar, wie wir die Szene wahrnehmen würden, da sonst unterschiedliche absolute Szenenhelligkeiten auf gleiche relative Bildhelligkeiten abgebildet werden. Am Anfang von Abschnitt 4.2 betrachten wir diesen Punkt näher. Wie wir in Kapitel 2.3 gesehen haben ist diese Voraussetzung bei unserer Gewinnung von HDR-Daten mithilfe der LDR-Aufnahmen nicht notwendigerweise gegeben, da wir in der Regel keinen absoluten Bezugspunkt haben.

Auch ohne Kalibrierung gegen einen absoluten Helligkeitswert erhalten wir durch unser Verfahren zur HDR-Gewinnung relativ zueinander korrekte Helligkeitswerte (vergleiche Abschnitt 2.3.3). Eine nachträgliche Kalibrierung kann daher mithilfe eines *Skalierungsfaktors* erfolgen. Dieser kann qualitativ über die Generierung entsprechender Tonemappings und deren Bewertung durch einen Betrachter bestimmt werden. Auch eine automatische Kalibrierung anhand des Histogramms ist in vielen Fällen möglich: Es zeigt sich, dass in Szenen mit wenig Licht der größte Teil der Pixel am linken Rand des Histogrammes versammelt ist. Entsprechend sind bei vornehmlich hellen Szenen die meisten Pixel am rechten Rand zu finden. Anhand dieser Information lässt sich eine absolute Helligkeit schätzen. [*Reinhard et al.*, 2005]

Wie wir gesehen haben, liegt das Hauptaugenmerk beim Tonemapping oft auf der Reduktion der Helligkeitsdynamik. Bei vielen Tonemappingoperatoren wird dazu ein normalisiertes **Grauwertbild** (L_W) erzeugt, da dieses die Helligkeitsverteilung innerhalb des Bildes widerspiegelt. Anschließend werden dessen Helligkeiten tonegemappt, also der Kontrastumfang des Grauwertbildes reduziert.

Zur Reproduktion der **Farbe** werden die daraus resultierenden Faktoren (L_D) anschließend mit den originalen Farbwerten des entsprechenden Pixels multipliziert, um die darzustellenden Farbwerte $[R_D, G_D, B_D]$ zu erhalten:

$$\begin{bmatrix} R_D \\ G_D \\ B_D \end{bmatrix} = \frac{L_D}{L_W^s} \begin{bmatrix} R_W^s \\ G_W^s \\ B_W^s \end{bmatrix}$$

Der Index W bezeichnet hier Weltfarbwerte, D die Displayfarbwerte. Mithilfe des Skalierungsfaktors s kann die Sättigung kontrolliert werden. [*Reinhard et al.*, 2005]

Für alle Algorithmen ist es entscheidend, die **Grundhelligkeit**, auf die unser Auge adaptieren würde, zu bestimmen. Global lässt sie sich beispielsweise über das arithmetische[123] oder geometrische[124] Mittel berechnen. Wie wir wissen, adaptiert die visuelle Wahrnehmung nicht global sondern lokal. Die mit dem global errechneten Wert erzielten Tonemappings stimmen daher nicht unbedingt mit unserem „realen“ Eindruck der Szene überein.

Insbesondere bei sehr kontrastreichen Bildern kann das globale Mittel zu falschen Ergebnissen führen. Dies ist beispielsweise der Fall, wenn es sehr viele sehr helle oder sehr viele sehr dunkle Anteile im Bild gibt. Das vorgestellte Maß für Grundhelligkeit ist in diesem Fall deutlich heller bzw. dunkler als die menschliche Wahrnehmung.

Es bietet sich daher an, Bildregionen und deren lokale Helligkeiten zu bestimmen. Dieses Vorgehen orientiert sich an der menschlichen Wahrnehmung: Wir verarbeiten Muster in verschiedenen Ortsfrequenzbändern auf verschiedenen Auflösungsstufen (Tiefpassfilter) des gerade betrachteten Bildes [*Reinhard et al.*, 2005][*Peli*, 1990].

Die Operatoren, die dieses Helligkeitsmaß verwenden, sind gerade die lokalen Operatoren.

Die lokale Umgebungshelligkeit eines Pixels kann beispielsweise durch einen Gaußfilter oder einen Boxfilter über eine feste Umgebung bestimmt werden. Probleme treten bei diesem Ansatz an kontrastreichen Kanten auf. Hier können so genannte **Halo**s entstehen – helle oder dunkle Schatten um Kanten mit hohem Kontrast (siehe Abbildung 4.16). Dies liegt daran,

Abbildung 4.16: Auf dem Bild sind vor allem um die Stele und die Bäume deutlich Halos zu erkennen. (Das Bild zeigt das Capital Building in Washington DC.) [sunyata@flickr.com]

dass die Pixel mit entgegengesetzter Helligkeit aus der Nachbarschaft in die lokale Grundhelligkeitsberechnung einfließen und das Mapping daher zu dunkel oder zu hell ausfallen lassen. Grafik 4.17 zeigt das Entstehen eines solchen Halo bei Filterung im Ortsfrequenzbereich.

Eine geeignete Auswahl der „lokalen“ Regionen kann Halos verhindern. *Yee und Pattanaik* [2003] stellen ein Verfahren dazu vor: Sie segmentieren das Bild auf Basis der Pixelhelligkeiten und vereinigen die Segmente anschließend so lange, bis sie geeignete Regionen im

[123] $\frac{1}{n}\sum_{i=1}^{n} I_i$
[124] $\sqrt[n]{\prod_{i=1}^{n} I_i}$

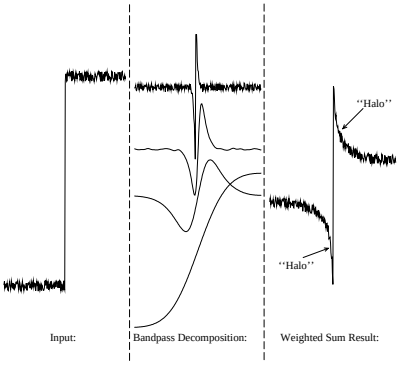


Abbildung 4.17: Die Veranschaulichung einer Entstehungsmöglichkeit eines Halo im Frequenzspektrum: Wir sehen links das Bildsignal. In der Mitte die Dekomposition zur Filterung von unten nach oben in aufsteigenden Frequenzen. Um keine Details zu verlieren, ist es sinnvoll, beim Tonemapping vor allem die tieffrequenten Signalbestandteile zu komprimieren. Bei diesem Beispiel aus [*Tumblin und Turk*, 1999] wird die Kante „unnatürlich" verstärkt. *Tumblin und Turk* [1999] präsentieren in ihrem Paper daher eine andere Dekorrelationsfiltermethode, die sie LCIS nennen und die dieses Problem berücksichtigt. Vergleiche dazu Grafik 4.21.

Graphen haben. Diese Wahl der Regionen vermeidet Artefakte beim anschließenden lokalen Tonemapping.

Eine *geeignete Wahl der Größe der Region*, auf die der Gauß- oder Boxfilter angewandt wird, kann ebenfalls Abhilfe schaffen. *Ashikhmin* [2002] und *Reinhard et al.* [2002b] präsentieren Algorithmen zur Auswahl. Sie vergrößern die Regionen so lange, bis die Abweichung vom bisherigen Mittelwert eine Schwelle überschreitet.

Durand und Dorsey [2000], *Pattanaik und Yee* [2002] und *Choudhury und Tumblin* [2005] nehmen eine Verfeinerung der Region vor, indem sie innerhalb des eben bestimmten Radius nur bestimmte –in der Helligkeit ähnliche– Pixel betrachten.

Tumblin und Turk [1999] zerlegen das Bild in geeignete Frequenzbänder, so dass eine anschließende Filterung der niedrigfrequenten Bestandteile (Grundhelligkeit) nicht zu den in Grafik 4.17 ersichtlichen Artefakten führt und gleichzeitig die Details im tonegemappten Bild erhalten bleiben.

4.2 Analyse bestehender Tonemapping-Algorithmen

Tumblin und Rushmeier [1993] haben Tonemapping in der Computergraphik eingeführt [*Reinhard et al.*, 2005]. Die HDR-Daten, die sie auf den Bildschirm tonemappen, sind keine Aufnahmen –wie wir in Abschnitt 2.3 gesehen haben, war dies erst die Anfangszeit der Computerkameras– sondern computergeneriert: Es sind gerenderte Szenen.

Tumblin und Rushmeier schreiben, dass es zuvor gängige Praxis war, eine Szene zu berechnen und das Ergebnis danach auf den kompletten Abbildungsbereich des Ausgabemediums zu skalieren.

Das dadurch entstehende Problem verdeutlichen sie in einem Beispiel: Ein von einem Glühwürmchen beleuchteter Raum wird auf dem Monitor genau gleich dargestellt wie derselbe Raum unter Beleuchtung eines Suchscheinwerfers, da beide auf denselben Bereich skaliert werden.

Unsere, durchaus von dieser Darstellung differierende, Wahrnehmung der in der Realität sehr unterschiedlich beleuchteten Szenen kommt auf dem Monitor nicht zum Ausdruck. Folglich sehen sie die Aufgabe des Tonemappings darin, diesen Missstand zu beheben. Die grafische Darstellung der Zieles haben wir bereits in Abbildung 4.13 betrachtet.

Die Autoren modellieren die menschliche Wahrnehmung sowie die Charakteristika eines Kathodenstrahlmonitors. Die unbekannten Faktoren, wie beispielsweise die absolute Monitorhelligkeit, nehmen sie dabei als konstant an. Fundament ihres Operators sind die Ergebnisse von *Stevens und Stevens* [1963] zur menschlichen Wahrnehmung. Die entsprechenden Erkenntnisse haben wir bereits in Abbildung 4.10 gesehen. Mithilfe ihres Operators ist es von nun an möglich, Unterschiede in der absoluten Helligkeit einer computergenerierten Szene darzustellen.

Neben einem realistischen Helligkeitseindruck können Tonemappingoperatoren auch andere Ziele haben:

Ein solches Ziel kann es beispielsweise sein, die Details zu bewahren. Dazu muss versucht werden, den Kontrast möglichst im unterscheidbaren Bereich zu halten. Wir haben dies bereits im letzten Abschnitt kurz angesprochen.

Ein weiteres Ziel kann es sein, die Farben der Helligkeitsadaptionsstufe des Auges entsprechend anzuzeigen.

Viele weitere Ziele sind denkbar.

Eine chronologische Übersicht über die in der Folgezeit nach *Tumblin und Rushmeier* [1993] entwickelten Verfahren findet sich in Tabelle 4.18. Wir wollen, [*Devlin et al.*, 2002] folgend[125], ebenfalls einige Stationen des Tonemappings kurz betrachten. Teilweise finden sich die grafischen Veranschaulichungen des Mapping in Abbildung 4.15.

Auf die konkrete Mathematik hinter den Operatoren verzichten wir an dieser Stelle. Durch die Betrachtungen in Abschnitt 4.1.5 und die nun folgenden Anmerkungen sollte der Leser dennoch ein Gefühl für mögliche und notwendige Operationen beim Tonemapping bekommen.

[125] In *Devlin* [2002] finden sich zum Teil ausführlichere Beschreibungen zu den Algorithmen.

Algorithm	Spatially Uniform	Spatially Varying	Time dependent	Strengths and Weaknesses
Tumblin and Rushmeier 199[3]	√			Preserves brightness. Does not preserve visibility or account for adaptation. Grey scale only.
Chiu et al 1993		√		Preserves local contrast. Ad hoc. Computationally demanding. Halo artifacts.
Ward 1994	√			Preserves contrast - crucial for predictive lighting analysis. Clipping of very high and very low values. Does not consider complexities of typical workplace viewing.
Schlick 1994		√		Speed and simplification of uniform and varying operators. Ad hoc - no perceptual accuracy.
Spencer et al. 1995		√		Greater dynamic range through glare effects.
Ferwerda et al. 1996	√		√	Accounts for changes in threshold visibility, colour appearance, visual acuity and sensitivity over time. Psychophysical model. Useful for immersive displays.
Ward Larson et al. 1997	√			Histogram equalisation. Preserves local contrast visibility. Uses models for glare, colour sensitivity and visual acuity to increase perceptual realism.
Jobson et al. 1997		√		Multi-scale retinex model - perceptually valid. Problems with monochrome scenes and maximum contrasts outside 24-bit RGB range.
Pattanaik et al. 1998		√		Multi-scale psychophysical representation of pattern, luminance and colour processing resulting in increased perceptual fidelity. Does not incorporate temporal aspects.
Tumblin et al. 1999	√			Layering method for static, synthetic images. Foveal method for interactive scenes.
Tumblin and Turk 1999		√		LCIS method preserves subtle detail and avoids halo artifacts.
Pattanaik et al. 2000	√		√	Psychophysical operator with time-dependent adaptation and appearance models. Does not support local adaptation.
Scheel et al. 2000	√			Interactive method using texturing hardware. Does not account for adaptation over time.
Durand and Dorsey 200[0]	√		√	Interactive method with time-dependent adaptation and simulation of visual acuity and chromatic adaptation.
Ashikhmin 2002		√		Preserves image details and absolute brightness information. Uses simple functional perceptual model. Simple to implement with moderate computational efficiency.
Fattal et al. 2002		√		Computationally efficient and simple operator. Does not attempt psychophysical accuracy.
Reinhard et al. 2002		√		Method based on photographic technique. Suits a wide variety of images, but limited to photographic dynamic range rather than the wider computer graphics dynamic range.

Abbildung 4.18: Eine Aufstellung verschiedener Tonemappingoperatoren bis zum Jahr 2002 aus [*Devlin et al.*, 2002].

4.2.1 Tonemapping-Algorithmen

Ward [1994] entwickelt einen Operator auf Basis der Experimente von *Blackwell* [1981]. Die zugrunde liegenden Threshold-versus-Intensity-Diagramme (TVI), die wir in Abbildung 4.12 gesehen haben, beschreiben, welche Helligkeitsunterschiede für uns gerade noch wahrnehmbar sind. Wards Operator versucht dementsprechend den Kontrast innerhalb der Szene im wahrnehmbaren Bereich zu belassen. Damit bleiben die Details erkennbar.

Larson et al. [1997] wollen ebenfalls die Sichtbarkeit des Kontrasts und damit der Details erhalten. Anders als Ward arbeiten sie mit einem Histogramm. Für das Histogramm bestimmen sie die logarithmischen mittleren Helligkeiten über 1° Raumwinkel. Um dem Eindruck eines Beobachters der realen Szene besser zu entsprechen, simulieren sie zusätzlich Blendeffekte. Die Bedingungen, wann wir Blendung wahrnehmen, haben wir in Grafik 4.9 betrachtet. Blendeffekte erscheinen uns beispielsweise als Kreise oder Sterne um eine besonders helle Lichtquelle herum. Ein entsprechender Algorithmus findet sich in [*Spencer et al.*, 1995]. Weiterhin berücksichtigen sie die Sehschärfe und unser Farbempfinden bei der jeweiligen Helligkeit (vgl. Abschnitt 4.1.1).

In *Tumblin et al.* [1999] findet sich die Weiterentwicklung des ursprünglichen Tumblin-Rushmeier-Verfahrens. Die Autoren trennen zwischen ambientem und spekularem Lichtanteil eines Objektes (vergleiche Abschnitt 4.1.4). Da wir stärker auf Glanzlichter als auf ambientes Leuchten reagieren, lässt sich der ambiente Teil ohne störenden Effekt bis zu einem gewissen Grad komprimieren. Die Kompression beruht dabei oft auf der Arbeit von *Schlick* [1994].

Eine derartige Trennung in die beiden Beleuchtungsanteile ist nur bei synthetischen Daten problemlos möglich. Bei aufgenommenen Daten kann man versuchen, die Trennung mithilfe entsprechender Filter zu erreichen. Das Verfahren eignet sich also weniger für unsere HDR-Daten. Es sei dennoch hier erwähnt, da es uns zeigt, welche Daten sich eher zur Datenreduktion anbieten, um einen guten Bildeindruck zu erhalten.

In einem zweiten Ansatz präsentieren *Tumblin et al.* [1999] ein interaktives Verfahren, dass das foveale Sehen nachzubilden versucht. Dazu muss der Betrachter den Mauscursor an die Stelle des Bildes bewegen, die er gerade betrachtet. Der Algorithmus passt das Mapping dann interaktiv an, so dass in diesem Bereich möglichst viele Details erhalten bleiben.

Die bisher betrachteten Operatoren sind *globale* Operatoren. Wie wir bei der Klassifikation (Abschnitt 4.1.4) gesehen haben, verwenden sie einmal bestimmte Werte für die Umgebungshelligkeit für alle Pixel. Als Vorteil der globalen Verfahren sei hier erneut der geringe Berechnungsaufwand und damit ihre Echtzeitfähigkeit genannt.

Die Helligkeitsadaption unseres visuellen Systems ist komplexer. Wir bestimmen die Helligkeiten sowohl örtlich lokal –also für verschiedene Bildbereiche unterschiedlich– als auch über verschiedene Auflösungsstufen des Bildes [*Pattanaik et al.*, 1998]. Ein lokales Vorgehen trägt dem eher Rechnung. Die Studien, die wir in Abschnitt 4.2.4 betrachten, zeigen ebenfalls, dass wir das lokale Mapping als realistischer empfinden.

Im Folgenden betrachten wir *lokales* Tonemapping.

Die Auswahl der Region für die Bestimmung der lokalen Helligkeit und die Wahl der Region, für die diese Helligkeit als Adaptionsstufe verwendet wird, spielen eine entscheidende Rolle. Welche Auswirkungen die unterschiedliche Größe und Gewichtung einer Region für das Ergebnis haben können, wird in Grafik 4.19 veranschaulicht. Wie wir sehen, entstehen je nach Parameterwahl sehr verschiedene Bildeindrücke mit unterschiedlicher Erkennbarkeit der Details. Bei einer ungeschickten Wahl der Regionen können unnatürliche Bildeffekte, wie beispielsweise Halos (siehe Abschnitt 4.1.5), entstehen.

Jobson et al. [1997] führen lokales Tonemapping auf Basis eines so genannten Color Appearance Models (CAM) durch. Ein CAM beschreibt die menschliche Farbwahrnehmung. Dazu werden zuerst die chromatische Adaption eines Beobachters und anschließend davon

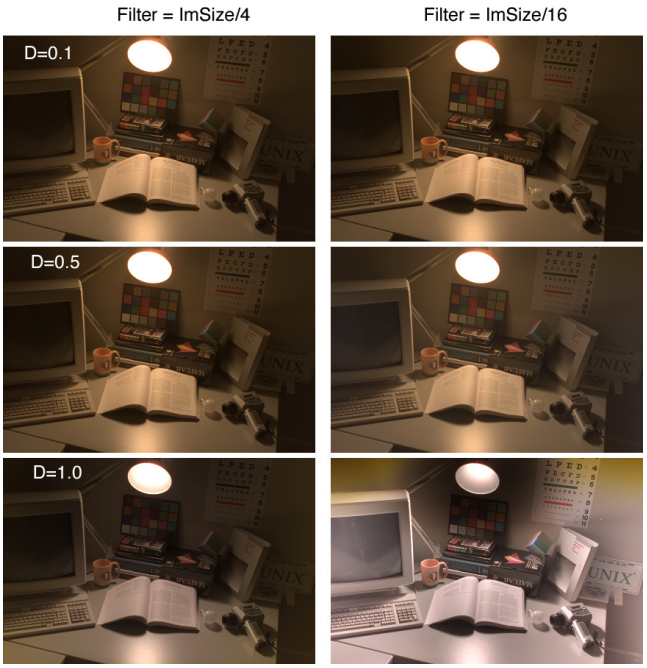


Abbildung 4.19: Qualitativer Vergleich der Auswirkung verschiedener Parameter auf den Algorithmus [*Johnson und Fairchild*, 2003]. Die beiden Spalten wurden mit unterschiedlicher Tiefpassfiltergröße berechnet. Wir sehen, dass links auf ein größeres Gebiet adaptiert wurde. In den Zeilen wurde der Grad der Berücksichtigung des lokalen Adaptionslevels variiert. Rechts unten sehen wir deutlich, dass die Helligkeiten lokal angepasst wurden. Das Gesamtbild wirkt unnatürlich, zeigt uns allerdings mehr Details als das entsprechende Bild links daneben, bei dem die Adaption auf einen größeren Bereich erfolgt ist.

unabhängig die relativen und absoluten Farbeigenschaften[126] bestimmt. Vorteil an der Verwendung eines CAM ist, dass die Bilder bei der Ausgabe an die Wiedergabeumgebung des Betrachters angepasst werden können. Dies ist im Sinne unserer in Abschnitt 4.1.2 skizzierten Verarbeitungspipeline mit einer finalen Anpassung an die lokalen Betrachtungsgegebenheiten beim Rezipienten wünschenswert. So kann unabhängig von der Betrachtungsumgebung der gleiche Bildeindruck generiert werden [*Akyüz und Reinhard*, 2006]. *Jobson et al.* [1997] setzen das so genannte Retinex-Modell von *Land und McCann* [1971] ein. *Devlin et al.* [2002] schreiben, dass das Modell Probleme bei Bildern mit einer Dynamik von über 24 Bit habe. Ebenfalls problematisch seien stark gesättigte Aufnahmen.

Pattanaik et al. [1998] arbeiten mithilfe des in Grafik 4.20 wiedergegebenen Filterschemas. Sie teilen die Verarbeitung, wie in unseren eingangs vorgestellten Modellen vorgeschlagen, in ein „visual model“ und ein „display model“. Die Helligkeit, auf die lokal adaptiert wird, bestimmen sie mithilfe verschieden gefilterter (und insbesondere skalierter) Versionen des Bildes. Das Ablaufdiagramm ihres Algorithmus ist in Grafik 4.20 wiedergegeben. Das Diagramm gibt einen Eindruck möglicher Verarbeitungsschritte beim Mapping.

Tumblin und Turk [1999] stellen mit dem Low Curvature Image Simplifier (LCIS) ein frequenzraumbasiertes Verfahren vor. Sie teilen das Bild in zwei Schichten: Die eine beinhaltet die Grobstruktur, die andere die Details. Die Besonderheit ihres Filters ist, dass es gerade nicht zu den normalerweise auftretenden Halo-Artefakten führt. Dies wird im Vergleich der Grafiken 4.17 und 4.21 deutlich.

Ashikhmin [2002] verwendet ein dreistufiges Modell. In einem ersten Schritt wird ein lokaler Adaptionswert für jedes Pixel berechnet. Im zweiten Schritt wird die Dynamik reduziert. Dies geschieht mithilfe der TVI-Funktion des Auges (siehe Abschnitt 4.1.1 und Grafik 4.12).

[126] Hierzu zählen Sättigung, Buntheit, Farbton etc.

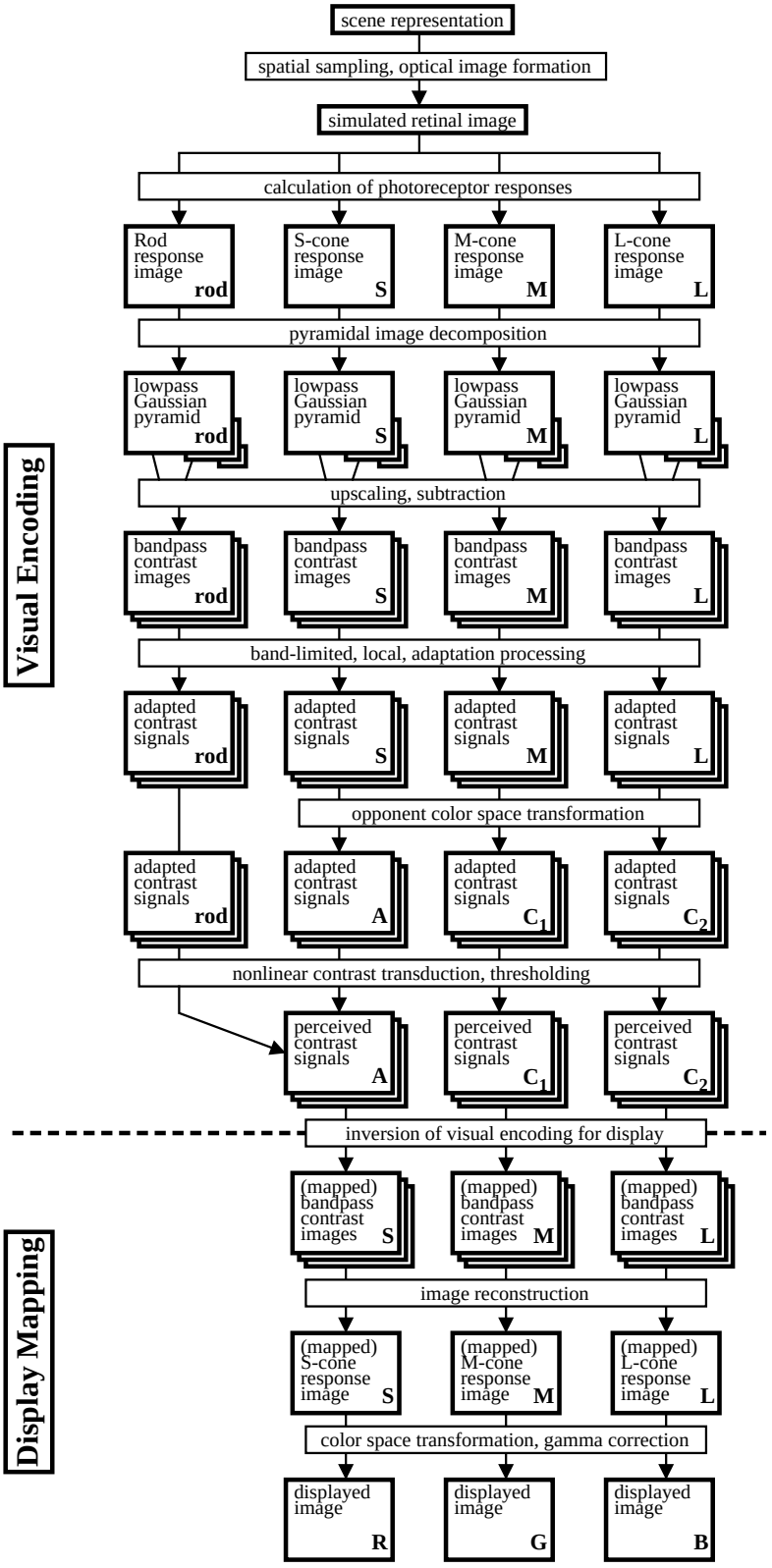


Abbildung 4.20: Das Ablaufdiagramm des Tonemappingalgorithmus von *Pattanaik et al.* [1998]. Es zeigt mögliche Verarbeitungsschritte beim Tonemapping und ist in leicht modifizierter Form auch für andere Algorithmen gültig.

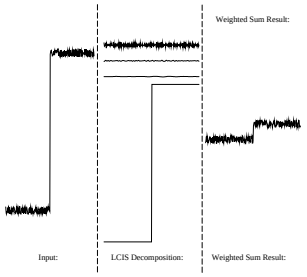


Abbildung 4.21: Die Dekorrelation mithilfe des LCIS vermeidet in diesem Beispiel die Bildung so genannter HALOs. Vergleiche lineare Filterung in Grafik 4.17. [*Tumblin und Turk*, 1999]

Im dritten Schritt werden eventuell im zweiten Schritt verloren gegangene Details wieder hinzugefügt.[*Reinhard et al.*, 2005]

Durand und Dorsey [2002] verwenden einen Filterkern, der das Bild tiefpassfiltert aber gleichzeitig die Kanten bewahrt. Dies erreichen sie durch einen gewichteten Gaußfilter. Die Gewichtung des Gaußfilterkerns wird verschoben, sobald der Dichteunterschied innerhalb der Maske zu groß wird. Siehe Grafik 4.22. Es entstehen ein „base layer“ und ein „detail layer“,

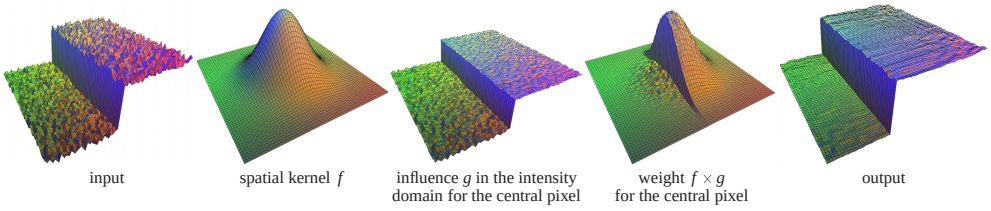


Abbildung 4.22: Das Schaubild veranschaulicht die Anpassung des Filterkerns an einer Kante. Diese bilaterale Filterung ist die Grundlage von [*Durand und Dorsey*, 2002].

von denen der erste stärker komprimiert wird. Das Verfahren nennt sich bilaterale Filterung. [*Reinhard et al.*, 2005]

Choudhury und Tumblin [2005] fügen dem bilateralen Filter eine weitere Dimension hinzu und filtern trilateral. Sie wollen damit Probleme des bilateralen Filters beheben. Ein solches Problem ist das konstruktionsbedingte schwache Einebnen des Gradienten an scharfen Kanten[127]. Daraus resultiert eine schlechte Kompressionseigenschaft für kanten- und kurvenreiche Bildregionen. Zur Verbesserung bestimmen *Choudhury und Tumblin* [2005] mit einem ersten bilateralen Filter die geglätteten Gradienten in x- und y-Richtung – das Bild wird also tiefpassgefiltert. Aus dem Ergebnis berechnen sie die Trennstellen für die eigentliche bilaterale Filterung. Die Autoren schreiben und veranschaulichen, dass ihr Algorithmus die Daten effektiver glättet als die bilaterale Filterung. Siehe Grafik 4.23.

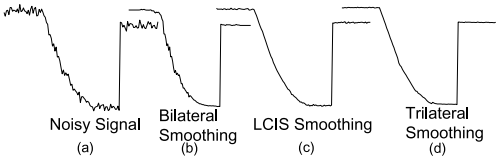


Abbildung 4.23: Wir sehen den glättenden Effekt von bilateraler Filterung, LCIS und trilateraler Filterung. [*Choudhury und Tumblin*, 2005]

[127] Die jeweiligen Flächen werden eingeebnet, nicht jedoch die Kante und damit ebensowenig der Gradient zwischen den Flächen.

Fattal et al. [2002] arbeiten mit dem Gradientenbild. Sie schwächen große Gradienten und damit große Helligkeitsunterschiede im Originalbild ab. Anschließend integrieren sie das Gradientenbild wieder zu den Helligkeitswerten (Details siehe [*Horn*, 1974]). Wir haben solch ein Verfahren bereits bei der Klassifizierung der Tonemappingoperatoren in Abschnitt 4.1.4 angesprochen.

Da nur an großen Gradienten komprimiert wird, bleiben feine Helligkeitsabstufungen und damit Details sichtbar. Zur Integration wird ein Poisson-Verfahren verwendet. Dies führt zu Ungenauigkeiten und ist berechnungsintensiv.

Reinhard et al. [2002b] orientieren sich am Zonensystem der Fotografie, das wir einleitend in 4.1 schon betrachtet haben. Sie wenden das besprochene Verfahren des „dodging and burning“ an. Damit adaptieren sie eine erprobte analoge Technik.

Ferwerda et al. [1996] stellen ein Modell vor, in dem sie die relative Empfindlichkeit (TVI), die Schärfeanpassung, die zeitliche Adaption und die chromatische Anpassung des Auges berücksichtigen (siehe auch Abschnitt 4.1.1). [*Drago et al.*, 2002]

Devlin et al. [2002] fahren in ihrem Vergleich mit Verfahren fort, die die zeitliche Adaption des Auges nachbilden. Da wir es mit Standbildern zu tun haben, sind diese Verfahren für uns nicht so relevant. Wir werden sie daher hier nicht betrachten.

4.2.2 Tonemapping-Ergebnisse

Wir haben nun einen Eindruck davon, auf welche Art die Operatoren eine Komprimierung der Dynamik erreichen. Damit sind wir in der Lage, tonegemappte Bilder mit geschulterem Auge zu betrachten und nicht nur anhand unserer Intuition zu beurteilen.

Wir wissen beispielsweise, dass Kanten Problemgebiete darstellen können: Unter Anderem können dort Halos entstehen.

Weiterhin wissen wir, dass Details oder der Farbigkeitseindruck durch das Mapping verloren gehen können.

Da das Ziel des Tonemapping der Mensch bzw. die Wahrnehmung des Menschen ist, betrachten wir abschließend einige Bildergebnisse. Diese veranschaulichen erneut einige Unterschiede der verschiedenen Algorithmen. Siehe Grafiken 4.24 und 4.25.

Abbildung 4.24: Die Abbildung zeigt den visuellen Unterschied zwischen vier verschiedenen Tonemappingoperatoren. Von links nach rechts: *Durand und Dorsey* [2002], *Funt et al.* [2002], *Reinhard et al.* [2002b], *Larson et al.* [1997]. [*Devlin et al.*, 2002]

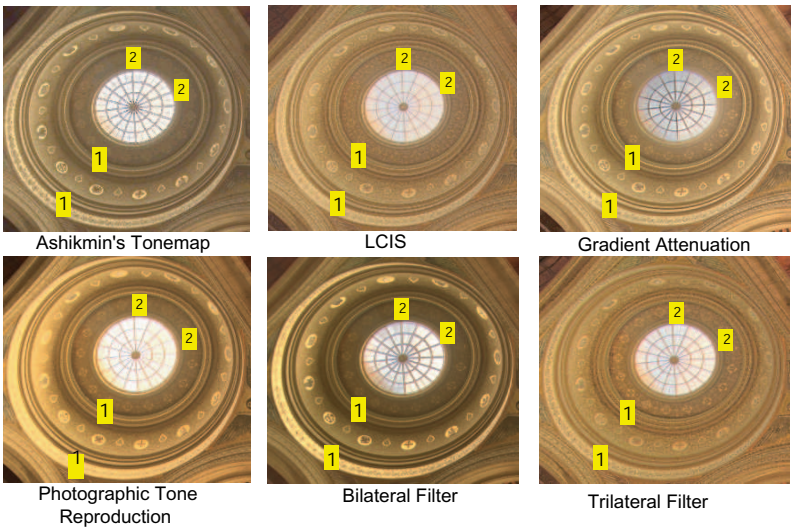


Abbildung 4.25: Ein visueller Vergleich von Tonemappingoperatoren. Der Vergleich soll veranschaulichen, welche Details bei welchen Verfahren erhalten bleiben. Die schmalen Ringe innerhalb der Region (1) bleiben nur bei der „Gradient Attenuation“ und dem „Trilateral Filter“ (rechts) erhalten. Ähnlich sieht es bei den goldenen Ornamenten im inneren Deckenfries und den ringförmigen Umfassungen des Fensters aus (2). [*Choudhury und Tumblin*, 2005]

4.2.3 Laufzeit der Algorithmen

Reinhard et al. [2005] haben die Laufzeit einiger Algorithmen verglichen. Dies ist für unsere Anwendung insofern relevant, als wir relativ große Datenmengen (horizontale Auflösung über 360°) haben, die bei jedem Bild tonegemappt werden müssen. Die Laufzeit des Tonemapping bestimmt wesentlich die Laufzeit unseres Verfahrens. Die Laufzeiten einiger Algorithmen sind in Grafik 4.26 wiedergegeben.

Operator	Time (in seconds)
GLOBAL OPERATORS	
Miller's operator	≈ 15.0
Tumblin–Rushmeier's operator	3.2
Ward's scale factor	0.96
Ferwerda's operator	1.0*
Ferschin's exponential mapping	3.0
Logarithmic mapping	3.4
Drago's logarithmic mapping	2.8
Reinhard's global photographic operator	3.7
Reinhard and Devlin's photoreceptor model	9.7
Ward's histogram adjustment	3.4
Schlick's uniform rational quantization	3.4
LOCAL OPERATORS	
Chiu's spatially variant operator	10.0
Rahman and Jobson's multiscale retinex	120.0
Johnson and Fairchild's iCAM	66.0
Ashikhmin's operator	120.0
Reinhard's local photographic operator	80.0
GRADIENT DOMAIN OPERATORS	
Horn's lightness computation	45.0
Fattal's gradient domain compression	45.5
FREQUENCY-BASED OPERATORS	
Oppenheim's operator	12.4
Durand's bilateral filtering	23.5

Abbildung 4.26: Ein Laufzeitvergleich verschiedener Algorithmen. Für uns sind nur die relativen Zeiten interessant. Die Messung fand mit einem 1600x1200 Bild auf einem 800MHz Apple iBook G3 mit 512 MB RAM statt. [*Reinhard et al.*, 2005, 358]

Wie aufgrund der Beschreibung im vorliegenden Kapitel erwartet, haben die globalen Operatoren im Mittel die kürzeste Laufzeit und die lokalen die längste, da sie die meisten Berechnungen benötigen.

Bei den Gradientenoperatoren ist die Integration berechnungstechnisch am teuersten. Sie bestimmt wesentlich die Laufzeit von Tonemappingoperatoren, die dieses Verfahren verwenden. Sie ist allerdings billiger als der Aufbau von Auflösungspyramiden, die wir bei einigen der vorgestellten Verfahren berechnen müssen.

Frequenzbasierte Verfahren benötigen zum Wechsel der Bildrepräsentation in den Frequenzraum die (Fast-)Fourier-Transformation ([F]FT). Die Ausführungszeit solcher Algorithmen wird von dieser Transformation dominiert. Eine Möglichkeit, die diskrete Fouriertransformation stark zu beschleunigen ist die schnelle Fouriertransformation (Fast Fourier Transformation , FFT). Durch die Aufteilung der Berechnungsschritte bei der schnellen Fouriertransformation (FFT) ist es besonders effektiv, Bilder mit einer Höhe und Breite zu berechnen, die jeweils Potenzen von 2 sind. Entsprechend sind primzahlige Dimensionen am langsamsten, da sie sich nicht zerlegen lassen.

Wie wir gesehen haben, kann auch mittels eines Filters in den Frequenzraum transformiert werden. Bei kleinen Filtern ist zwar der Berechnungsaufwand geringer als die Durchführung einer Fouriertransformation, dafür sind sie deutlich ungenauer. Bei großen Filtern schwindet der Berechnungsvorteil zunehmend. [*Reinhard et al.*, 2005]

4.2.4 Diskussion der Verfahren

Für die Algorithmen zur Aufnahme der High-Dynamic-Range-Daten hatten wir ein objektives Vergleichskriterium: Sie sind mit einem Belichtungsmesser nachprüfbar[128].

Diese Möglichkeit haben wir hier nicht. Ein wichtiger Grund für fehlende objektive Beurteilungskriterien ist, dass kein vollständiges Modell für die visuelle menschliche Wahrnehmung vorliegt, mit dem wir die erzeugten Daten abgleichen können.

Selbst mit solch einem Verfahren wäre eine umfassende vergleichende Beurteilung der Tonemappingalgorithmen nicht möglich – ihre Ziele sind dafür zu verschieden. *Reinhard et al.* [2005] schreiben, dass es nicht den Operator für alle Zwecke gibt sondern vielmehr für verschiedene Anwendungen der richtige Operator ausgewählt werden sollte.

Wie bereits einleitend besprochen, greifen wir auf den Menschen und seine subjektive Wahrnehmung zurück. Wir haben in diesem Kapitel mehrfach gesehen, dass dies sinnvoll ist, da der Mensch gerade das Ziel der Operatoren ist. Das Modell von Tumblin-Rushmeier stellt dies dar (vergleiche Grafik 4.13). Die subjektive Beurteilung des Bildeindrucks durch den Menschen ist maßgeblich.

Es wurden mehrere Studien zum Vergleich der Bildergebnisse der einzelnen Verfahren durchgeführt. Mögliche Versuchsanordnungen für solch ein Experiment finden sich, neben den unten erwähnten Quellen, in *Ledda et al.* [2004a][129] und *Johnson* [2004]. Letzteres ist eine von der *Commission Internationale de l´Eclairage* (CIE) einberufene Kommision, die sich mit der Schaffung einer solchen Testumgebung befasst.

Der praktische Versuchsablauf ist meist so gestaltet, dass die Probanden paarweise die, mit verschiedenen Operatoren gemappten, Bilder gegeneinander bewerten.

Drago et al. [2003a] haben Kontrast, Detailreichtum und natürlichen Bildeindruck als Beurteilungskriterien gewählt. Sie schreiben, dass Detailreichtum und Natürlichkeit sich als gut charakterisierende Merkmale herausgestellt haben. Sie kommen zu dem in Grafik 4.27 dargestellten Ergebnis. Die Autoren schreiben, dass das fotografische Tonemapping von *Reinhard et al.* [2002b] dem Idealpunkt („ideal point“ in der Grafik) am nächsten kommt [*Drago et al.*, 2002].

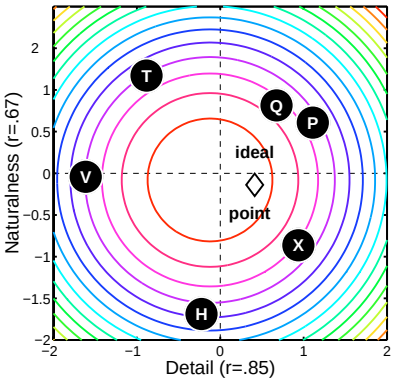


Abbildung 4.27: Das Ergebnis der Studie von *Drago et al.* [2003a]. Die Algorithmen sind: H: Histogram Adjustment [*Larson et al.*, 1997], P: Photographic Tonereproduction [*Reinhard et al.*, 2002b], Q: Uniform Scaling Quantisation [*Schlick*, 1994], T: Revised Tumblin-Rushmeier [*Tumblin et al.*, 1999], V: Visual Adjustment [*Ferwerda et al.*, 1996], X: Retinex [*Funt et al.*, 2002]. (Die Zuordnung wurde nach [*Drago et al.*, 2002] vorgenommen.) [*Drago et al.*, 2003a]

[128] Bestimmte Messpunkte können beispielsweise mit einem Luminanzmeter ausgemessen und mit den HDR-Daten abgeglichen werden.

[129] Die Autoren schlagen unter Anderem einen Vergleich des HDR-Bildes und der tonegemappten LDR-Daten auf einem HDR-Display vor.

Ledda et al. [2005] kommen zu dem Ergebnis, dass iCam [*Fairchild und Johnson*, 2002] bei Farbbildern in Bezug auf Ähnlichkeit und Detailreichtum sowohl in hellen als auch in dunklen Regionen die besten Ergebnisse liefert.

Kuang et al. [2007] haben Tonemappingoperatoren in Bezug auf Genauigkeit und subjektive Präferenz getestet. Sie kommen zu dem Ergebnis, dass beide Kriterien ungefähr zu den gleichen Bewertungen führen und daher austauschbar sind. Sie schreiben, dass die lokalen Operatoren den globalen im subjektiven Eindruck überlegen sind. Der bilaterale Filter erreicht die besten Ergebnisse, während die auf dem Retinex-Modell basierenden Verfahren am schlechtesten abschneiden.

Aus Grafik 4.27 ergibt sich, dass das Retinex-basierte Verfahren in [*Drago et al.*, 2003a] recht gut abschneidet.

Wie wir sehen, liefern die drei betrachteten Studien unterschiedliche Ergebnisse. Dies liegt unter anderem an der unterschiedlichen Methodik der Versuche und der Auswahl und Anpassung der verwendeten Algorithmen.

Zugleich spiegelt es den zuvor zitierten Sachverhalt wider, dass es **für verschiedene Zwecke verschiedene Operatoren** gibt. Den besten Tonemappingoperator für alle Zwecke gibt es nicht.

Bereits für einen Zweck können die subjektiven Präferenzen des jeweiligen Betrachters und damit die subjektive Auswahl des am besten geeigneten Tonemappingalgorithmus unterschiedlich sein.

Aus diesem Grund haben wir in diesem Kapitel verschiedene Ansätze und Beurteilungskriterien betrachtet und sind nun in der Lage, geeignet auszuwählen.

4.3 Zusammenfassung

In diesem Kapitel haben wir uns mit dem Tonemapping befasst. Den Einstieg bildete die Betrachtung eines analogen Verfahrens zur Kontrastreduzierung aus der Fotografie. Wir haben daran gesehen, dass das Problem der Kontrastreduzierung bei der Abbildung nicht computerspezifisch ist.

Um eine Grundlage für die Arbeitsweise und vor allem die Ziele von Tonemappingoperatoren zu legen, haben wir uns mit der menschlichen Wahrnehmung beschäftigt. Wir haben gesehen, dass wir unsere Wahrnehmung vor allem durch Adaption auf eine Grundhelligkeit verändern. Weiterhin haben wir uns mit der Kontrastwahrnehmung und der chromatischen Adaption befasst.

Den Einstieg in das eigentliche Tonemapping bildeten zwei allgemeine Modelle, denen wir eine sinnvolle Untergliederung in Arbeitsschritte und damit Modularisierungsmöglichkeiten für Tonemappingalgorithmen entnehmen konnten.

Nach dem abstrakten Modell sind wir auf die konkrete mathematische Umsetzung eines globalen (realistischen) Mappingschrittes eingegangen. Dabei haben wir ein Gefühl für die allgemeine Modellierung einer Mappingfunktion bekommen.

Die mathematische Betrachtung weiterführend, haben wir die Algorithmen aufgrund ihres Arbeitsraumes in Klassen eingeteilt. Unsere Oberklassen waren hierbei globale und lokale ortsraumbasierte Operatoren, solche im Frequenzraum und diejenigen, die auf dem Gradientenfeld operieren.

Unser Fokus galt nun konkreten Algorithmen. Hier haben wir zuerst allgemeinere Verarbeitungsschritte betrachtet, wie sie in vielen Operatoren Verwendung finden. Wir haben Grundlagen wie die Verarbeitung von Farbe und die Bestimmung der Grundhelligkeit behandelt.

Wir haben festgestellt, dass die globalen Verfahren oft zu einem unnatürlichen Bildeindruck führen, da unsere Wahrnehmung nicht über das gesamte Sichtfeld funktioniert. Bei den anschließend betrachteten lokalen Verfahren haben wir gesehen, dass die Wahl einer geeigneten Region für das Ermitteln der Helligkeit entscheidend ist, damit es nicht zu Problemen wie Halos kommt.

Anschließend haben wir deskriptiv einige der existierenden Tonemappingoperatoren kennengelernt. Obwohl wir nicht so stark wie in Kapitel 2 auf die Mathematik der Algorithmen eingegangen sind, haben wir dennoch einen Überblick über die verwendeten Techniken bekommen. Mit diesem Wissen waren wir in der Lage, die qualitativen Beispiele in Abschnitt 4.2.2 zu beurteilen.

Wir haben die Laufzeit verschiedener vorgestellter Verfahren betrachtet, da diese maßgeblich zur Gesamtlaufzeit unseres HDR-Panoramagenerierungsverfahrens beiträgt.

Als Abschluss des Kapitels haben wir Kriterien zur Beurteilung der Mappingoperatoren in Bezug auf die Güte des Bildergebnisses kennen gelernt. Da diese Beurteilung subjektiv erfolgt, wurden verschiedene Studien betrachtet. Wir haben gesehen, dass es nicht *den* besten Tonemappingoperator gibt sondern vielmehr, in Abhängigkeit des gewünschten Ergebnisses, ein geeigneter Operator gewählt werden muss.

Aufgrund der Ausführungen innerhalb dieses Kapitels sollte der Leser in der Lage sein, sowohl Veröffentlichungen zum Tonemapping nachzuvollziehen, als auch ein qualifiziertes Urteil über einen Tonemappingoperator zu fällen.

–leere Seite–

5. spy360

> What is a magician but a practising theorist?
>
> Obi-Wan Kenobi, Return of the Jedi

In den vorangehenden Kapiteln haben wir uns ausführlich mit der Theorie vertraut gemacht, die für unser Ziel, die Erstellung eines 360°-HDR-Panoramas bzw. optimalen 360°-LDR-Panoramas, relevant ist:

Inhalt von Kapitel 2 war die Gewinnung von HDR-Bilddaten mithilfe von LDR-Kameras. In Kapitel 3 sind wir auf das Stitching von Bildern eingegangen. Im vorigen Kapitel lag unser Augenmerk auf dem Tonemapping (Kapitel 4).

In diesem Kapitel wollen wir eine Software entwickeln, die unser Ziel praktisch erreicht.

An manchen Stellen dieses Kapitels wird Wissen über den Aufbau von Betriebssystemen und den Ablauf von Programmen vorausgesetzt. Soweit dies vom Umfang her vertretbar ist, wird das benötigte Wissen an der entsprechenden Stelle vermittelt. Für zusätzliche Erläuterungen und eine ausführlichere Abhandlung sei auf [*Tanenbaum*, 2001] und [*Glatz*, 2006] verwiesen. Viele mögliche Unklarheiten lassen sich auch durch eine Recherche im Internet beseitigen.

5.1 Ursprüngliches Ziel: Neuerfindung des Rades

Das ursprüngliche Ziel der Arbeit war es, eine komplett neue Software zu erschaffen, die alle Aufgaben von der Bildaufnahme mit den Webcams bis hin zum fertig gestitchten (tonegemappten) Bild übernimmt. Die Arbeit hätte sich damit in eine –unter Informatikern sehr beliebte– Tradition gereiht, das Rad immer wieder mit Begeisterung neu zu erfinden.

Das Ziel wurde nicht realisiert.

Software zur Erstellung von HDR-Bildern gibt es bereits.
Software zum Stitchen von (HDR-)Bildern zu einem Panorama gibt es bereits.
Software zum Tonemappen eines HDR-Bildes gibt es bereits.

Lediglich die Komponente zur Aufnahme von LDR-Belichtungsreihen mit Webcams und eine Steuerungseinheit, die die vorhandene Software integriert und die notwendigen Abläufe automatisiert, fehlen. Diese Komponenten werden wir in diesem Kapitel entwickeln.

Anschließend gehen wir kurz auf die Fremdsoftware ein, die wir für die oben genannten Teilprobleme einsetzen.

5.2 Design und Grundlagen

In diesem Abschnitt betrachten wir die grundlegenden Softwarearchitekturentscheidungen des Projektes. Wie im vorigen Abschnitt motiviert, befassen wir uns dabei hauptsächlich mit der Bildgewinnung.

Zum Verständnis einiger Entscheidungen ist, zum Teil tiefgehendes, Vorwissen über die Interna der Windows- und dort insbesondere der COM-Programmierung erforderlich.

Dieses Wissen wird, wo dies möglich erscheint, gegeben. An einigen Stellen muss jedoch auf weiterführende Literatur verwiesen werden, da die Erklärungen zu umfangreich für diese Arbeit sind. Die gegebenen Erläuterungen sollten dem Leser, der sich bereits etwas mit der Thematik auseinander gesetzt hat, genügend Anhaltspunkte geben, um der Darlegung zu folgen.

In diesem Kapitel geben die Fußnoten oft nähere Erläuterungen. Diese zusätzlichen Erläuterungen wurden in die Fußnoten verlagert, um den Lesefluss für den erfahrenen Leser nicht zu stören.

5.2.1 Gerätetreiber

Die Kommunikation zwischen der Anwendungssoftware und der Hardware erfolgt in der Regel über das Betriebssystem und den, meist bei der Hardware mitgelieferten, Gerätetreiber.

Ein Gerätetreiber stellt eine logische Schnittstelle zwischen Hardware und Software dar. Abbildung 5.1 veranschaulicht dies.

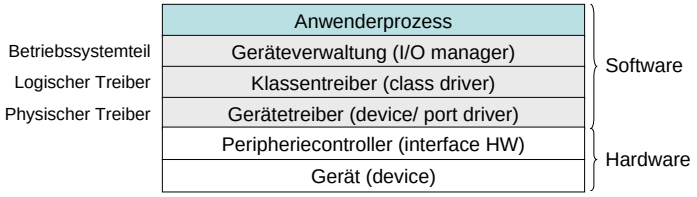


Abbildung 5.1: Die Abbildung zeigt die Treiberhierarchie innerhalb eines Computers auf. Nach [*Glatz*, 2006, 328].

Um mögliche Bestandteile eines Treibers kennen zu lernen, betrachten wir in Abbildung 5.2 einen Windows-Treiber. Das Bild zeigt den internen Aufbau eines Windows Driver Model (WDM)-Treibers. In der Grafik sind die Beziehungen zwischen den (potenziellen) Bestandteilen eines Windows-Treibers eingezeichnet.

Je nachdem, welche Interaktion mit dem Betriebssystem und der Hardware notwendig ist, müssen Teile eines Treibers im Kernelmode ablaufen. In der Grafik sind daher Treiberteile im Kernel-Mode und solche im User-Mode eingezeichnet. Für weitere Information zur Bedeutung von Kernel- und User-Mode sowie die Grundlagen von Betriebssystemen sei der Leser an dieser Stelle auf [*Tanenbaum*, 2001; *Glatz*, 2006] verwiesen.

Ein Treiber stellt insbesondere eine Schnittstelle zwischen Hardware und Betriebssystem dar, auf dessen Funktionalität dann der Anwenderprozess zugreift (vgl. Abb. 5.1). Dadurch ist bereits eine Abstraktionsschicht und damit eine einheitliche Zugriffsmöglichkeit auf eine Geräteklasse[130] innerhalb des Betriebssystemkontexts gegeben[131]. Eine weitere (umfassendere) abstrahierende Softwareschnittstelle betrachten wir mit DirectShow in Abschnitt 5.2.3.

[130] Eine solche Geräteklasse sind beispielsweise Webcams. Alle Webcamtreiber stellen beipielsweise die Funktion zur Verfügung, Bilder zu liefern. Der Zugriff auf ein solches Bild oder einen Stream erfolgt für alle Kameras gleich. Die konkrete Anpassung an das spezifische Gerät übernimmt der Treiber.

[131] Dies erfordert bereits, dass alle Gerätetreiber einer Geräteklasse genormte Schnittstellen anbieten.

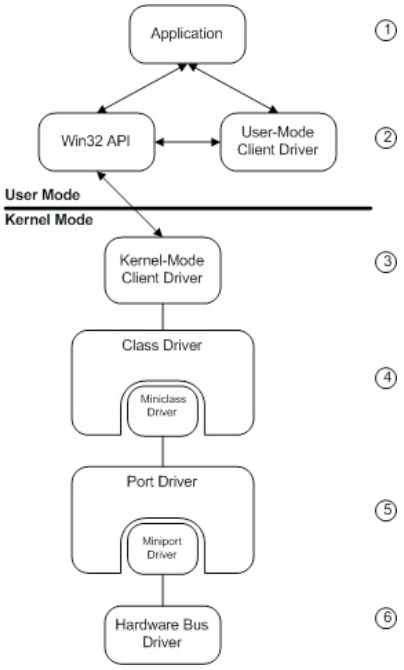


Abbildung 5.2: Das Schaubild veranschaulicht die verschiedenen Komponenten, die ein Windows-Treiber gemäß dem Windows Driver Model (WDM) enthalten kann. Die meisten Treiber beinhalten nicht alle Stufen. Wir sehen, dass Teile des Treibers im Usermode, andere Teile im Kernelmode ablaufen können. [*Microsoft Corporation*, 2008, Windows Driver Kit: Getting Started with Windows Drivers]

Unter Linux werden fast alle Geräte als Datei innderhalb des Dateisystems (beispielsweise „/dev/usb0“) angesprochen. Unter diesem System ist die Abstraktion daher noch ausgeprägter.

Aus der engen Verzahnung mit dem Betriebssystem ergibt sich, dass Treiber für ein bestimmtes Betriebssystem erstellt werden: Ein Windowstreiber funktioniert unter Linux nicht. Er folgt in der Regel auch nicht dem Aufbau aus Grafik 5.2.

Ohne Gerätetreiber kann ein Gerät nur mit größerem Aufwand in Betrieb genommen werden – es müssen Rohdaten an die Hardwareschnittstelle geschickt und von ihr gelesen werden[132].

Da wir eine universelle, nicht speziell an einen Kameratyp angepasste[133], Software entwickeln wollen, ist die Verfügbarkeit von Gerätetreibern für uns wichtig.

[132] Zusätzlich sind die Schnittstellen oft nicht dokumentiert und es werden proprietäre (undokumentierte) („geheime“) Datenformate verwendet. Dies stellt insbesondere bei der Treiberentwicklung für Linux ein Problem dar, da die Entwickler dieser Treiber oft nicht mit dem Hardwarehersteller verbunden sind und daher keinen Zugriff auf interne Informationen haben. Die für die Linuxgerätetreiberentwicklung verwendete Information beruht oftmals auf Reverse-Engineering der Windows-Treiber.

[133] Vor allem für digitale Fotokameras gibt es so genannte Software Developper Kits (SDK). Diese bieten spezielle Schnittstellen zu kompatiblen Kameras des jeweiligen Herstellers. Die Software „AHDRIA“ von *O'Malley* [2006] baut beispielsweise auf dem SDK von Canon auf. AHDRIA erfüllt für digitale Canon-Fotokameras eine ähnliche Aufgabe wie unsere Software.

5.2.2 Welches Betriebssystem?

Eine grundsätzliche Entscheidung vor dem Programmieren einer Software ist die **Wahl des Betriebssystems**.

Wie wir im letzten Abschnitt gesehen haben, ist die Verfügbarkeit von **Gerätetreibern** wichtig. In die engere Wahl kommen daher nur Linux und Windows als Betriebssystem[134].

Die von uns gewählte Fremdsoftware (siehe Abschnitt 5.5) steht für beide Systeme zur Verfügung.

Wir betrachten zunächst einige Argumente, die für das jeweilige Betriebssystem sprechen.

Für **Linux** als Betriebssystem sprechen:

- Die freie Verfügbarkeit.
- Die zu erwartende höhere Verarbeitungsgeschwindigkeit.
- Die zu erwartende höhere Stabilität des Systems.
- Der häufige Einsatz von Linuxsystemen als Webserver.

Der letzte Punkt ist relevant, da wir die gewonnenen Bilder via Internet zur Verfügung stellen möchten.

Für **Windows** sprechen:

- Die breitere Installationsbasis, die ca. 90% aller PC-Systeme umfasst[135].
- Die daraus resultierende größere Gerätetreiberbasis.
- Die –für den „nicht informatischen Anwender“– vermeintlich einfachere und gewohnte Bedienung.

Aus den genannten Punkten folgt, dass wir mehr Benutzer unter Windows ereichen können. Vor allem unsere Zielgruppe des nicht informatischen Anwenders wird vermutlich Windows als Betriebssystem einsetzen.

Der entscheidende Punkt gegen Linux ist jedoch die **fehlende Treiberbasis**[136] und die **fehlende einheitliche Abstraktionsschicht**[137] über den Geräten.

Mit der **DirectShow**-Schnittstelle bietet Windows ein einheitliches Interface zu den von uns benötigten Multimediafunktionen. Abbildung 5.3 illustriert dies.

Wie wir in Grafik 5.3 sehen, abstrahiert DirectShow zusätzlich zum Gerätetreiber über die Hardware.

Der entscheidende Vorteil dabei ist, dass wir eine **Vielzahl unterschiedlicher Hardware** über die **einheitliche** DirectShow-**Schnittstelle** ansprechen können: Unsere Anwendung kann dadurch ohne größere Modifikationen[138] beispielsweise auch TV-Karten, DV-Geräte oder andere Bildeingabegeräte verwenden.

---

[134] Mac-OS-X zählen wir bei unserem Vergleich zu Linux dazu, weil es auf BSD aufbaut und ähnliche Treiberprobleme hat. Da es jedoch ein kommerzielles Betriebssystem ist, werden von Herstellerseite eher Treiber angeboten als für Linux. Bei Linux ist die Entwicklergemeinschaft dafür größer und so entstehen Treiber.

[135] Quelle: Forrester.

[136] Es gibt für viele Geräte mittlerweile auch Treiber für Linux. Diese sind jedoch zumeist nicht offiziell vom Hersteller unterstützt.

[137] Das Video4Linux-Projekt stellt eine ähnliche Schicht für Linux bereit, bietet allerdings keine so breite Hardwareunterstützung. (Vergleiche auch Kapitel 6.)

[138] Eventuell müssen die Geräte für die Benutzung noch speziell gesteuert (Kanaleinstellung, Band starten etc.) werden. Die Schnittstellen dazu werden wiederum von DirectShow angeboten.

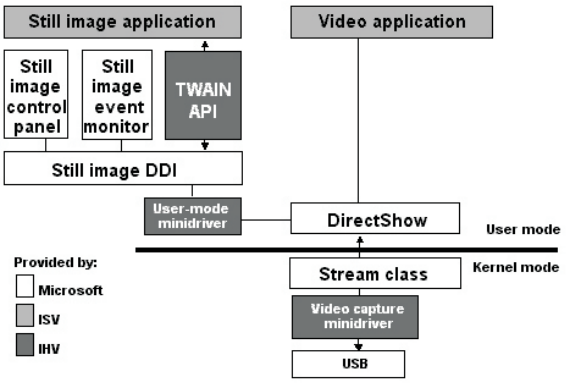


Abbildung 5.3: Die Grafik zeigt die Lage und Bedeutung der DirectShow-Schnittstelle innerhalb einer Anwendung auf. (IHV = Imaging Hardware Vendor, ISV = Imaging Software Vendor) [*Microsoft Corporation*, 2008, Driver Development Kit: USB Webcam]

5.2.3 DirectShow

Zum weiteren Verständnis betrachten wir kurz Grundlagen von DirectShow und COM:

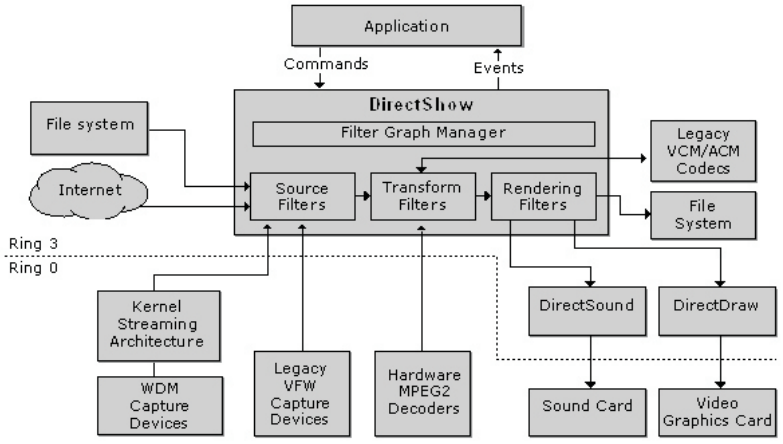


Abbildung 5.4: Die Abbildung zeigt Lage und Funktion von DirectShow als Abstraktionsschicht zwischen Anwendung und Hardware etc. (Ring 3 ≅ User-, Ring 0 ≅ Kernelmode). [*Microsoft Corporation*, 2008, DirectShow System Overview]

Der Ursprung der DirectShow-Bibliothek wurde 1996 als *ActiveMovie* zum Abspielen von MPEG1, AVI und Quicktime-Videos entwickelt. Seitdem wurden weitere Funktionen hinzugefügt, unter anderem die Unterstützung für Webcams, DVD, digitales Fernsehen...

Die Bibliothek bündelt die Multimediafähigkeiten von Windows.

1998 wurde die Bibliothek in das DirectX-SDK[139] integriert. 2005 wurde DirectShow aus dem DirectX-Paket in das Platform-SDK verschoben. Das Platform-SDK ist die Schnittstellenbibliothek für das Betriebssystem Windows und wird daher direkt oder indirekt für alle Anwendungen, die unter diesem Betriebssystem laufen sollen, benötigt. Die Verlagerung aus dem –eher zur Spiele- oder Grafikanwendungsentwicklung verwendeten– DirectX in das Platform-SDK spiegelt wieder, dass DirectShow Standardfunktionalität eines Multimediabetriebssystems enthält[140].

[139] Software Developper Kit

[140] Interessant ist, dass auch im aktuellen Platform-SDK 6.0a noch DirectX-Headerfiles eingebunden sind. Um kompilieren zu können, ohne DirectX zu installieren, muss man die entsprechenden Zeilen von Hand auskommentieren.

5.2.3.1 Datenverarbeitung in DirectShow: FilterGraphen

Die Datenverarbeitung innerhalb von DirectShow erfolg über so genannte **Filtergraphen**.

Ein **Graph** ist in der Informatik ein Tupel (V, E) aus Knoten V (vertice) und diese verbindenden Kanten E (edge). Wenn die Kanten eine Richtung haben („Pfeil“), so bezeichnet man den Graphen als gerichteten Graphen. [*Diestel*, 2006]

Die Filtergraphen in DirectShow sind gerichtete Graphen. Die Knoten werden in DirectShow als **Filter** bezeichnet. Die Schnittstellen der einzelnen Knoten heißen PINs[141]. Ein PIN stellt eine unidirektionale Schnittstelle dar: Er kann eine Datenquelle oder -senke sein. Ein PIN ist genau mit einem anderen PIN verbunden.

Ein FilterGraph bildet den Datenfluss zwischen Filtern ab.

Ein Filtergraph für unsere Anwendung ist in Abbildung 5.5 dargestellt. Wie wir sehen, wird eine Reihe von Komponenten benötigt, um ein Vorschaubild zu erhalten. Je nach Kameratyp reichen auch die Datenquelle „CapDev“ auf der linken Seite und die Datensenke „Video Renderer“ auf der rechten Seite aus.

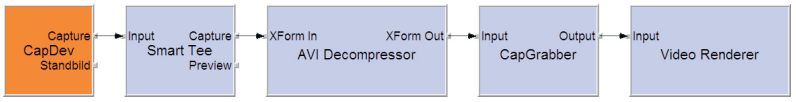


Abbildung 5.5: Die Abbildung zeigt den Filtergraphen der Philipps ToUCam Pro PCV 740K während das Vorschaubild angezeigt wird.

Die eingezeichnete Komponente „Smart Tee“ splittet das, aus der Kamera kommende, Videosignal in ein Vorschau- (Preview) und ein Aufnahmesignal (Capture). Der „AVI Decompressor“ wandelt das Farbformat vom kameraseitigen „I420“ in das vom „Video Renderer“ erwartete „ RGB24-Format (vgl. Abschnitt 2.2.1). Der „CapGrabber“ ist die Komponente, die es uns erlaubt, Standbilder aus dem Videodatenstrom zu entnehmen. Der „Video Renderer“ schließlich bringt die Daten auf den Bildschirm. Wenn keine Bildschirmanzeige benötigt wird, kann als Datensenke der „NullRenderer“ verwendet werden. Dies spart Resourcen, da die Daten nicht für die Bildschirmausgabe aufbereitet[142] werden müssen.

Programmiertechnisch besteht DirectShow heute zum großen Teil aus Objekten, die nach dem Component Object Model entwickelt wurden (COM-Objekte). In Abschnitt 5.2.5 befassen wir uns mit COM. Jedes Objekt unseres Filtergraphen ist als COM-Objekt realisiert.

Dadurch wird ein hoher Abstraktionsgrad erreicht und die Komponenten lassen sich vielfältig kombinieren, da sie aufgrund der COM-Schnittstelle kompatibel sind.

Die Programmierung von COM ist mitunter aufwendig (siehe Abschnitt 5.2.5.1). Für so genannte „Clientanwendungen“, also solche Anwendungen, die nur Funktionalität in Anspruch nehmen, selbst aber keine bereitstellen, kann die Verwendung von COM-Programmierparadigmen auf ein Minimum beschränkt werden. Es gibt sogar spezielle Bibliotheken, die die COM-Details vor dem Programmierer komplett verbergen.

Für die Entwicklung *wiederverwendbarer* DirectShow-Komponenten ist die Programmierung von COM unumgänglich.

Bevor wir uns in Abschnitt 5.2.5 mit COM befassen, gehen wir auf das Konzept der Bibliothek ein. Diese werden wir in unserem Programm benutzen. Die Entwicklung von COM ist zum Teil auch motiviert durch das Verhalten von dynamischen Bibliotheken, die wir nun betrachten.

[141] pin, englisch: Steckkontakt; Da die PINs die Kantenendpunkte sind könnte man diese als Knoten bezeichnen. Da jedoch letztendlich die Filter miteinander verbunden werden, bezeichnen wir diese als Knoten.

[142] Die Skalierung auf die Fenstergröße ist beispielsweise solch ein Verarbeitungsschritt.

5.2.4 Statische und Dynamische Biliotheken

Mehrere Komponenten unserer Anwendung sind als dynamische Bibliothek realisiert.

Als Bibliothek (library) bezeichnet man „eine Sammlung von Unterprogrammen, Routinen oder Prozeduren, die sich in einer Datei befindet“ [*Brockhaus*, 2003]. Diese Sammlung von Funktionen kann von Programmen benutzt werden. Der entstehende Vorteil ist, dass vielfach benutzte Funktionalität nur einmal in solch einer Bibliothek abgelegt werden muss und dann von verschiedenen Programmen auf diese Programmteile zurückgegriffen werden kann. Der Zugriff erfolgt dazu über eine definierte Schnittstelle[143].

Es gibt statische und dynamische Bibliotheken. Die unterschiedliche Benennung bezieht sich darauf, wann die in der Bibliothek enthaltenen Funktionen für das aufrufende Programm ansprechbar gemacht werden[144]. Bei statischen Bibliotheken[145] geschieht dies zur Kompilierzeit[146]. Bei dynamischen Bibliotheken, die auch als Laufzeitbibliotheken bezeichnet werden, geschieht dies zur Laufzeit.

Um ansprechbar zu sein, muss die Adresse einer Bibliotheksfunktion bekannt sein (siehe auch Abbildung 5.6). Eine **statische Bibliothek** wird beim Kompilieren zum Programm dazugelinkt, also in die ausführbare Datei integriert. Die Adressen der Funktionen stehen damit ab diesem Zeitpunkt fest[147]. Nachteil der statischen Bibliothek ist, dass sie in die ausführbare Datei eingelinkt wird und diese damit vergößert. Die komplette Bibliothek befindet sich also in der ausführbaren Datei. Dies ist gleichzeitig ein Vorteil, da so sichergestellt ist, dass die Bibliothek auf dem Zielsystem vorhanden ist und vor allem in der richtigen Version vorliegt.

Eine **dynamische Bibliothek** wird erst bei Benutzung in den Speicher geladen. Ihre Funktionsadressen stehen daher erst zur Laufzeit fest. Der Vorteil der dynamischen Bibliothek ist, dass sie nicht in den Programmcode des sie benutzenden Programms eingebaut wird. Die Programme bleiben dadurch kleiner und eine Bibliothek, die von mehreren Programmen benutzt wird, muss nur einmal auf dem Rechner vorhanden sein. Zusätzlich wird es möglich, eine Bibliothek nachträglich auszutauschen, ohne die sie verwendenden Programme ändern zu müssen.

Dies gilt allerdings nur, solange die neue Bibliotheksversion kompatibel[148] zur alten bleibt. Die Aufrechterhaltung dieser Kompatibilität und die damit zusammenhängende Versionsverwaltung können zu Problemen führen bis hin zur Inkompatibilität einer neuen DLL. Dies war ein Grund für die Entwicklung des Component Object Model (COM), das wir in Abschnitt 5.2.5 betrachten werden.

Unter Windows wird eine dynamische Bibliothek als Dynamic Link Library (**DLL**) bezeichnet [*Petzold*, 2000].

[143] Dies kann beispielsweise bei der Programmiersprache C eine Header-Datei sein.

[144] Bei einem von Neumann-Rechner befindet sich das auszuführende Programm im Speicher des Computers. Der so genannte Instruktionszeiger (instruction pointer)/ Programmzähler (instruction counter) zeigt auf die Stelle des Programms, die als Nächstes ausgeführt werden soll. Bei einem Funktionsaufruf wird der Zeiger auf die Speicheradresse der aufgerufenen Funktion gesetzt. Bei statisch gelinkten Bibliotheken ist diese Adresse fest, da sich der Programmcode während der Ausführung nicht ändert. Da das Programm beim Starten in den Speicher geladen wird, ist klar, an welcher Adresse der Funktionsaufruf steht. Bei dynamischen Bibliotheken muss die Adresse in Abhängigkeit von der Lage der Bibliothek im Speicher bestimmt werden. [*Tanenbaum*, 2001; *Glatz*, 2006; *Brockhaus*, 2003]

[145] Eine statische Bibliothek hat üblicherweise die Endung *.lib* unter Windows.

[146] Als Kompilieren bezeichnet man den Vorgang, den Programmcode in eine auf einem Prozessor ausführbare Form zu verwandeln.

[147] Die Adressen sind relativ zur Startadresse des Programms im Hauptspeicher bekannt. Diese kann sich bei jedem Programmstart ändern, da das Programm an eine andere Stelle im Speicher geladen werden kann.

[148] Genauer bedeutet dies, dass die Exportschnittstelle gleich bleiben muss. Diese kann sich aus verschiedenen Gründen ändern.

Eine Auslagerung von Programmteilen in eine Bibliothek führt zu einer starken (Ab-)**Kapselung** dieser Teile: Die Kommunikation kann nur über die Bibliotheksschnittstelle erfolgen. Dies sei hier deshalb erwähnt, weil es trotz bester Vorsätze immer wieder dazu kommt, dass die Abstraktion zwischen verschiedenen Modulen verletzt wird, indem über Modulgrenzen hinweg in andere Module zugegriffen wird. Bei der Bibliothekserstellung wird das Brechen der Abstraktion erschwert. Sie kann daher als Erziehungsmaßnahme bezüglich des Abstraktionsprinzips aufgefasst werden.

Neben diesem programmierlogischen Aspekt führt die Abspaltung eines Programmteils in eine Bibliothek praktisch zu dem Vorteil, dass die Kamerakomponente nach Fertigstellung nicht mehr (zeitaufwendig) mitkompiliert werden muss.

Durch die Verwendung der *dynamischen* Bibliothek können (bei Beibehaltung der Exportschnittstelle) das Programm oder die Bibliothek später durch eine neue Version ersetzt werden, ohne die jeweils andere Komponente verändern zu müssen.

5.2.5 Distributed Component Object Model

Wie die anderen DirectX-Komponenten[149] basiert DirectShow teilweise auf Microsofts Distributed Component Object Model (**DCOM**). Für die Programmierung von DirectShow ist die Kenntnis der Grundlagen von COM nützlich.

COM wurde Anfang der 90er von Microsoft entwickelt. Seine Vorgänger waren Dynamic Data Exchange (DDE) und Object Linking and Embedding (OLE) in erster Version. Beide dienen der Kommunikation von Anwendungen untereinander. Insbesondere wurden die Verfahren zum Datenaustausch verwendet – beispielsweise über die Zwischenablage.

COM stellt heute die Grundlage für den Datenaustausch zwischen Anwendungen dar, seine Funktionalität geht jedoch darüber hinaus.

Thompson et al. [1997] umreißen einige Ziele von DCOM:

- **Interoperabilität** zwischen Modulen über System-/ Plattform- und Programmiersprachengrenzen hinaus.
- **Versionierung** in Bezug auf Kompatibilität zwischen alten und neuen Releases einer Bibliothek[150] und in Bezug auf den Namensraum[151].
- Handhabbarkeit der Komplexität und Größe verteilt arbeitender Systeme[152]

Als Container für die Programmodule können die in Abschnitt 5.2.4 bereits besprochenen Dynamic Link Libraries (DLL) verwendet werden. Durch die dynamische Bindung zur Laufzeit eines Programms können die COM-Objekte ihre Funktionalität so mehreren Programmen zugleich zur Verfügung stellen. Idealerweise werden die Module dadurch sowohl wiederverwendbar als auch austauschbar (s.o.).

[149] DirectShow war bis 2005 Bestandteil von DirectX (siehe Abschnitt 5.2.3).

[150] Wie wir bereits in Abschnitt 5.2.4 angesprochen haben, können sich Kompatibilitätsprobleme ergeben, wenn eine Bibliothek durch eine neue Version ersetzt wird. Insbesondere ergeben sich Probleme, wenn sich die Funktionalität einer Methode bei einer neuen Version ändert. Daher kann man bei COM-Objekten die Methodenversion abfragen und eine entsprechende Version der Methode aufrufen. Es wurde weiterhin darauf geachtet, dass auch das Problem der Vererbung konsistent gelöst wird. Bei Änderungen im Mutterobjekt kann es bei anderen Mechanismen geschehen, dass erbende Kinder durch die Änderung nicht mehr funktionieren. Bei COM wird dies verhindert.

[151] Werden beispielsweise mehrere DLLs in einem Programm eingebunden und verwenden diese dieselben Methoden oder Variablenbezeichner, so kann dies zu Problemen führen.

[152] Verteiltes Arbeiten muss hier nicht die räumliche Verteilung beinhalten sondern kann sich auch lediglich auf mehrere zusammen arbeitende Komponenten innerhalb eines Hosts beziehen.

Das aufrufende Programm muss zur Benutzung der bereitgestellten Funktionalität davon in Kenntnis gesetzt werden, wo sich im Speicher die Funktionen befinden, um diese aufrufen zu können (siehe auch Abschnitt 5.2.4). Siehe Abbildung 5.6.

Die Trennung zwischen Schnittstelle und Implementierung wurde durch die Einführung des COM-Konzeptes gegenüber dem traditionellen DLL-Schnittstellenmodell stark verbessert (siehe obenstehende Aufzählung).

Zwei Hauptprobleme waren: Der ehemalige DLL-Mechanismus funktioniert oft nur mit einer Programmiersprache. Neue Releases können mit dem „alten“ Mechanismus leicht inkompatibel zu älteren werden.

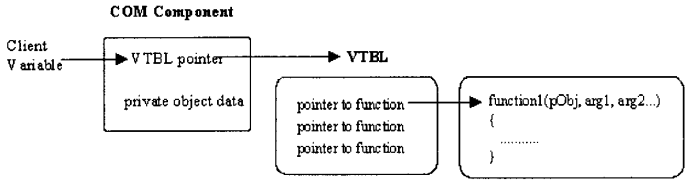


Abbildung 5.6: Die Grafik zeigt, wie die Adressierung von Funktionen innerhalb von COM (und DLLs) funktioniert. Die gemeinsam genutzte Bibliothek stellt die Adressen ihrer Funktionen zur Verfügung. Darüber können die aufrufenden Programme zugreifen. Die lokalen Variablen der Bibliothek werden hierbei für jeden Aufrufer in einer eigenen Instanz zur Verfügung gestellt (links). Zusätzlich sind auch gemeinsame Variablen zwischen Anwendungen möglich. [*Thompson et al.*, 1997] [*Microsoft Corporation*, 2008, COM General Technical Articles: The Component Object Model: A Technical Overview]

DCOM stellt die Erweiterung von COM über die Grenze eines Systems hinaus dar[153]. Mit DCOM können Objekte auf einem entfernten Rechner aufgerufen werden. Siehe Abbildung 5.7.

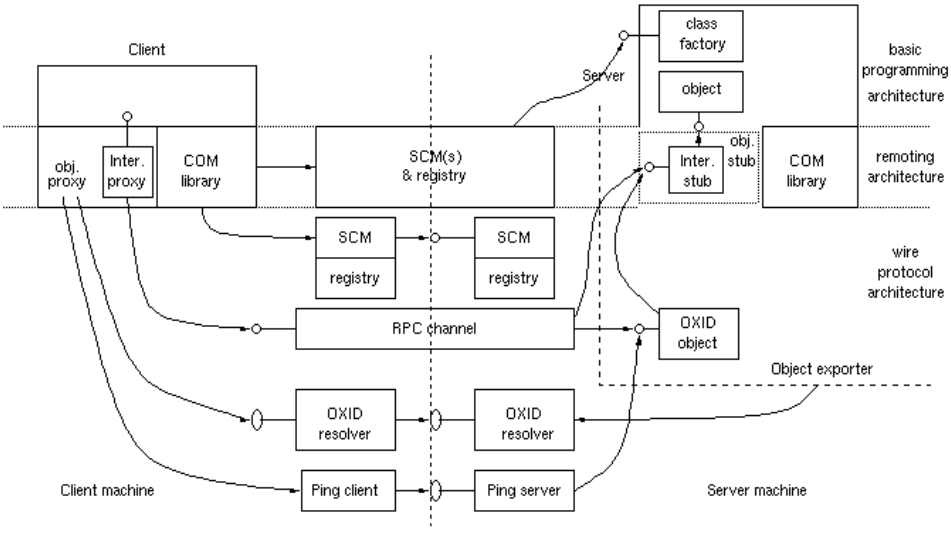


Abbildung 5.7: DCOM-Architekturübersicht. Dieses Diagramm erschließt sich nicht aus dem Text. Eine Einführung in die dargestellten Konzepte findet sich beispielsweise in [*Tanenbaum und Steen*, 2002]. [*Chung et al.*, 1997]

[153]DCOM war Microsofts Antwort auf CORBA (Common Object Request Broker Architecture) im Rennen um das Protokoll für Anwendungen, die über das Internet verteilt sind und miteinander kommunizieren wollen [http://en.wikipedia.org/w/index.php?title=Distributed_Component_Object_Model&oldid=181619020]. Ein Vergleich von CORBA und DCOM findet sich in [*Thompson et al.*, 1997] und [*Chung et al.*, 1997].

5.2.5.1 Kurze Einführung in die Programmierung des COM

Die Bezeichnung „Einführung“ ist hier nicht ganz wörtlich zu verstehen. Für die hier gegebene Einführung ist einiges Vorwissen erforderlich, das hier nicht gegeben werden kann. Es sei beispielsweise auf [*Eddon und Eddon*, 1998] und [*Box*, 1999] verwiesen. Der Abschnitt soll einen sehr gerafften Überblick über die Struktur von COM und dessen Programmierung geben. Dies soll dabei helfen, die Softwarearchitektur und die Implementierung von spy360 zu verstehen.

Die Informationen stammen größtenteils aus [*Microsoft Corporation*, 2008, COM: The Component Object Model].

Objekte innerhalb des Component Object Models haben einen eindeutigen Bezeichner, den so genannten Globally Unique Identifier (**GUID**).

Eine Indirektionsstufe über dem eindeutig bezeichneten Objekt ist der so genannte **Moniker**: Neben dem GUID kann das Objekt einen Typ haben. Dieser kann über den Moniker abgefragt werden. Ein Beispiel ist die CLSID_VideoInputDeviceCategory, zu der auch Webcams gehören. Geräte werden innerhalb einer Applikation üblicherweise über die Klasse und nicht der GUID identifiziert (in Listing 5.1 sehen wir oben einen GUID: „uuid“). Dadurch wird es möglich, verschiedene Geräte gleichen Typs (bspw. Webcams) einheitlich anzusprechen.

Ein Moniker dient als Schnittstelle zum eigentlichen COM-Objekt. Bei verteilten COM-Objekten stellen Moniker den lokalen Repräsentanten des aufgerufenen Objektes dar.

Über COM bereitgestellte Objekte können nicht instanziert werden. Dies ist nach der historischen Einführung klar: Die Komponenten stellen unabhängig von einer Clientanwendung Dienste zur Verfügung und existieren daher auch außerhalb des Aufrufkontextes. Ein Moniker stellt eine Instanz eines Objektes dar, liefert die Funktionalität aber erst nach Verbinden mit dem eigentlichen Objekt aus.

COM ist hierarchisch aufgebaut. Alle Komponenten müssen das „Urinterface“ **IUnknown** implementieren. Seine Schnittstelle sehen wir in Listing 5.1.

```
1  // Standardschnittstelle aller COM-Komponenten
2  [
3    object,
4    uuid(00000000-0000-0000-C000-000000000046)
5  ]
6  interface IUnknown {
7       [restricted]
8       HRESULT _stdcall QueryInterface([in] GUID* rrid,
9                                       [out] void** ppvObj);
10      [restricted]
11      unsigned long _stdcall AddRef();
12      [restricted]
13      unsigned long _stdcall Release();
14 }
```

Listing 5.1: Das IUnknown-Interface. Es muss von jedem COM-Objekt implementiert werden.

Die Funktionalität einer Komponente kann mittels der **QueryInterface**-Methode abgefragt werden. Jede Schnittstelle liefert einen eindeutigen Interface Identifier (**IID**) zurück. Dieser kann fest gewählt werden oder dynamisch generiert sein, muss aber für die Instanz des Objektes eindeutig sein. Das heißt, das Interface muss immer denselben Wert (GUID) zurückliefern, so lange die Instanz existiert (s.u.).

Die beiden Methoden **AddRef** und **Release** sind für die automatische Speicherverwaltung[154] notwendig. Jedes COM-Objekt beinhaltet einen Zähler, der die Anzahl der Referen-

[154]Programme belegen bei ihrer Ausführung Speicher. Speicher ist eine endliche Resource. Daher ist es wünschenswert, nur solche Objekte im Speicher zu haben, die auch benutzt werden. Bei der

zen auf das Objekt angibt[155]. Mit *AddRef* wird dieser Zähler um eins erhöht – mit *Release* entsprechend um eins dekrementiert. Sobald der Zähler Null wird, sollten keine Referenzen mehr auf das Objekt existieren und das Objekt löscht sich. Dies kann bei der Programmierung zu Problemen führen, wenn Objekte plötzlich verschwinden, weil der Zähler falsch benutzt wurde. In C++ gibt es daher Klassen, die die automatische Verwaltung der Zähler übernehmen („CComPtr< IBaseFilter >“).

Der Rückgabewert von COM-Funktionsaufrufen ist standardisiert. Alle COM-Objekt-Funktionen liefern einen 32-Bit **HRESULT**-Wert[156].

Um COM verwenden zu können, muss die COM-Bibliothek zur Verwendung im aktuellen Programm initialisiert werden. Dies erfolgt über den Befehl **CoInitializeEx**.

Bei dieser Initialisierung werden grundsätzliche Eigenschaften der COM-Umgebung festgelegt: Es kann zwischen zwei verschiedenen **Threading**-Modellen gewählt werden.

Beim so genannten **Apartment-threading** werden Funktionsaufrufe nacheinander über die Messagequeue abgearbeitet. Spezielle Synchronisationsmechanismen sind daher in der Regel nicht erforderlich.

Beim **Multi-threading/ Free-threading** werden die Aufrufe parallel abgearbeitet. Dadurch werden Synchronisationsmechanismen über kritische Abschnitte, Semaphore und Mutexe notwendig, um die einzelnen Threads bei Bedarf zu synchronisieren[157].

Mit **CoCreateInstance** werden Instanzen der COM-Objekte für die Benutzung mit der aufrufenden Anwendung aktiviert. Im Falle eines nicht laufenden COM-Objekt-Services wird dieser gestartet.

CoUninitialize beendet DCOM für die Anwendung und gibt alle noch von der Anwendung gehaltenen DCOM-Resourcen frei.

Näheres zu (D)COM findet sich beispielsweise in [*Eddon und Eddon*, 1998] oder [*Box*, 1999]. Einen kurzen Überblick über die Entwicklung und Architektur gibt [*Thompson et al.*, 1997]. Ausreichende Kenntnisse, um die Schnittstellen programmieren zu können, liefert das Microsoft Developper Network [*Microsoft Corporation*, 2008].

5.2.6 Nachrichten unter Windows

In unserer Anwendung werden an verschiedenen Stellen Nachrichten versandt.

Nachrichten spielen in Windows eine große Rolle. Die Kommunikation der meisten Objekte funktioniert über Nachrichten. Die Initialisierung eines Fensters, die Kommunikation der Komponenten des Fensters und ähnliches laufen nachrichtenbasiert ab.

Es gibt synchrone und asynchrone Nachrichten. Die synchronen Nachrichten werden direkt vom Empfänger behandelt. Solche Nachrichten initiiert man mithilfe einer *send*-Methode.

Die asynchronen Nachrichten werden in eine so genannte Message-Queue eingeordnet und nacheinander abgearbeitet. Eine asynchrone Nachricht initiiert man mit einer *post*-Methode.

nicht automatisierten Speicherverwaltung muss sich der Programmierer um die Freigabe von nicht benutztem Speicher kümmern. Dies ist fehleranfällig. Daher verwenden die meisten Systeme heute automatische Speicherverwaltung. Diese überprüft beispielsweise in regelmäßigen Abständen, ob Objekte im Speicher existieren, die nicht mehr verwendet werden und löscht diese im so genannten „garbage collection“-Schritt.

[155] Vereinfachend gesagt, entspricht die Zahl der Anzahl der Funktionen, die das COM-Objekt gerade benutzen.

[156] Auf 16-Bit-Systemen wird der SCODE zurückgeliefert, aus dem wiederum der HRESULT-Wert generiert wird.

[157] Details zu den theoretischen Konzepten von Multithreading und den zugehörigen Synchronisationsmechanismen finden sich in [*Tanenbaum*, 2001].

In Abschnitt 5.4.4 werden wir Nachrichten zwischen verschiedenen Threads austauschen. Diese Nachrichten werden wir asynchron verschicken, da wir gerade nicht wollen, dass unsere versendende Funktion blockiert, bis die Nachricht abgearbeitet ist.

```
1 BOOL PostMessage(
2   HWND hWnd,
3   UINT Msg,
4   WPARAM wParam,
5   LPARAM lParam
6 );
```

Listing 5.2: Der PostMessage-Funktionskopf.

In Listing 5.2 sehen wir den Funktionskopf der PostMessage-Methode. Wir sehen, dass eine Nachricht aus drei Bestandteilen besteht: Msg, wParam, lParam. „Msg“ identifiziert die Nachricht. Als Identifikation dient eine vorzeichenlose Ganzzahl. Innerhalb des Windows-SDK sind für die meisten Nachrichtentypen Konstanten vordefiniert. Wenn das Fenster einer Anwendung geschlossen werden soll, wird beispielsweise das Signal *WM_CLOSE* versendet, das der hexadezimalen Ganzzahl 0x0010 entspricht. Für benutzerdefinierte Signale, wie unsere beiden Nachrichten für bereitstehende Samples und bereitstehende LDR-Belichtungsreihen (siehe Abschnitt 5.3.2), müssen ebenfalls (gemeinsame) Nummern verwendet werden. Wie

Range	Description
0 through WM_USER -1	Messages reserved for use by the system.
WM_USER through 0x7FFF	Integer messages for use by private window classes.
WM_APP through 0xBFFF	Messages available for use by applications.
0xC000 through 0xFFFF	String messages for use by applications.
Greater than 0xFFFF	Reserved by the system for future use.

Tabelle 5.1: Identifizierer und ihre vorgesehene Benutzung unter Windows.

wir Tabelle 5.1 entnehmen können, ist der Bereich von WM_APP bis 0xBFFF hexadezimal dafür vorgesehen. Bei 32-Bit Windows-XP mit SDK 6.0a liegen die Bereiche bei 0x0400 für WM_USER und 0x8000 für WM_APP.

Die anwendungsspezifischen Nummern sind programmweit eindeutig. Da wir über die Grenze unserer DLL an das aufrufende Programm Nachrichten schicken möchten, müssen wir dafür sorgen, dass bei bei Anwendung und Bibliothek jeweils dieselben Nummern verwendet werden, damit beide Programme die Nachricht korrekt zuordnen können. Die *RegisterWindowMessage*-Methode erstellt anhand eines Identifikationsstrings derartige Nummern, die für beide Programme identisch sind.

Die beiden weiteren Parameter einer Nachricht sind der wParam und der lParam. Aus der Bezeichnung folgt, dass der zweite Parameter vom Typ Word (16 Bit) und der dritte Parameter vom Typ Long (32 Bit) ist. Dies galt für 16-Bit-Betriebssysteme wie Windows 3. Auf 32-Bit-Betriebssystemen sind beide Werte von der Länge 32 Bit.

Diese Parameter sind die Nutzdaten einer Nachricht. In ihnen kann Information verpackt werden. Wir werden den wParam zur Übertragung der Nummer der Kamera verwenden.

Für weitere Informationen zur Nachrichtenbehandlung unter Windows sei beispielsweise auf das Standardwerk zur Windowsprogrammierung, [*Petzold*, 2000], verwiesen.

5.2.7 Bildformate

Die im Rahmen dieser Arbeit entwickelte Anwendung erstellt die LDR-Bilderserien. Zur Weiterverarbeitung ruft sie direkt externe Programme auf. Um die Bilddaten an die externe Anwendung weiterzureichen, werden die LDR-Bilder auf der Festplatte zwischengespeichert.

Wir wollen daher an dieser Stelle ein paar Bildformate betrachten.

5.2.7.1 Bitmap

Unter Windows ist das Standard-Dateiformat eine Bitmap[158].

Die korrekt als „Device Independat Bitmap“ (DIB) bezeichnete Struktur besteht im Wesentlichen aus einem Bitfeld, das 8-Bit-Helligkeitswerte für jede Primärfarbe des Farbtripels (RGB; siehe Abschnitt 2.2.1) für jedes Pixel beinhaltet.

Da fast alle Bilddarstellungen unter Windows Bitmaps sind, wird das Speichern in diesem Format vom Betriebssystem unterstützt. Für andere Bildformate benötigt man zusätzliche Bibliotheken[159].

Für unseren Zweck ist das Bitmap-Dateiformat nur bedingt geeignet, da es das Speichern von Zusatzinformationen wie Belichtungszeit und ähnliches in der Bilddatei nicht vorsieht.

5.2.7.2 Tag Image File Format

Ein Format, das das Speichern von nahezu beliebiger Metainformation zu Bilddaten erlaubt, ist das „Tag Image File Format“ (TIFF) [*Aldus Corporation*, 1992].

TIFF wurde 1986 von der Aldus Corporation zusammen mit diversen Scannerherstellern entwickelt, um die Daten, die von diesen Geräten anfallen, speichern zu können. Nachdem Aldus von Adobe Systems übernommen wurde, gehört das Format heute zu Adobe.

Das TIF-Format stellt einen Container für verschiedene Bildformate dar. Es lassen sich beispielsweise JPEG- oder Vektorgrafiken in einer TIFF-Datei speichern.

TIFF ist heutzutage das systemübergreifende Standardformat für verlustlos oder gar nicht komprimierte Bildaten. Daher werden wir es zum Austausch unserer LDR-Bilder mit den Fremdanwendungen einsetzen.

Eine TIFF-Datei enthält so genannte „Tags“, um Zusatzinformation zu speichern, wie sie zum Beispiel für die Handhabung der verschiedenen nutzbaren Formate innerhalb des TIFF-Containers notwendig ist.

Ein Tag in TIFF besteht aus einem Tagbezeichner, einem Typbezeichner für den, dem Tag zugeordneten, Wert und dem eigentlichen Wert. Die einzelnen Tags werden zu so genannten „Image File Directories“ zusammengefasst. In Abbildung 5.8 sehen wir dies.

Neben den „normalen“ Daten wie Bildhöhe und Breite, finden sich unter den Tags auch für unsere Anwendung interessante wie dasjenige für die GrayResponseCurve, die die optische Dichte für jeden Grauwert angibt.

Das TIF-Format ist um beliebige Tags erweiterbar. Allerdings müssen die Anwendungen die Tags auslesen und kennen, um die Zusatzinformation nutzen zu können. Diese Voraussetzung ist zumeist nicht gegeben.

[158] Das Bildformat stammt ursprünglich aus der Entwicklung des Gemeinschaftsbetriebssystems OS/2 von IBM und Microsoft. [*Petzold*, 2000]

[159] Unter .net sind weiter Grafikformate standardmäßig unterstützt.

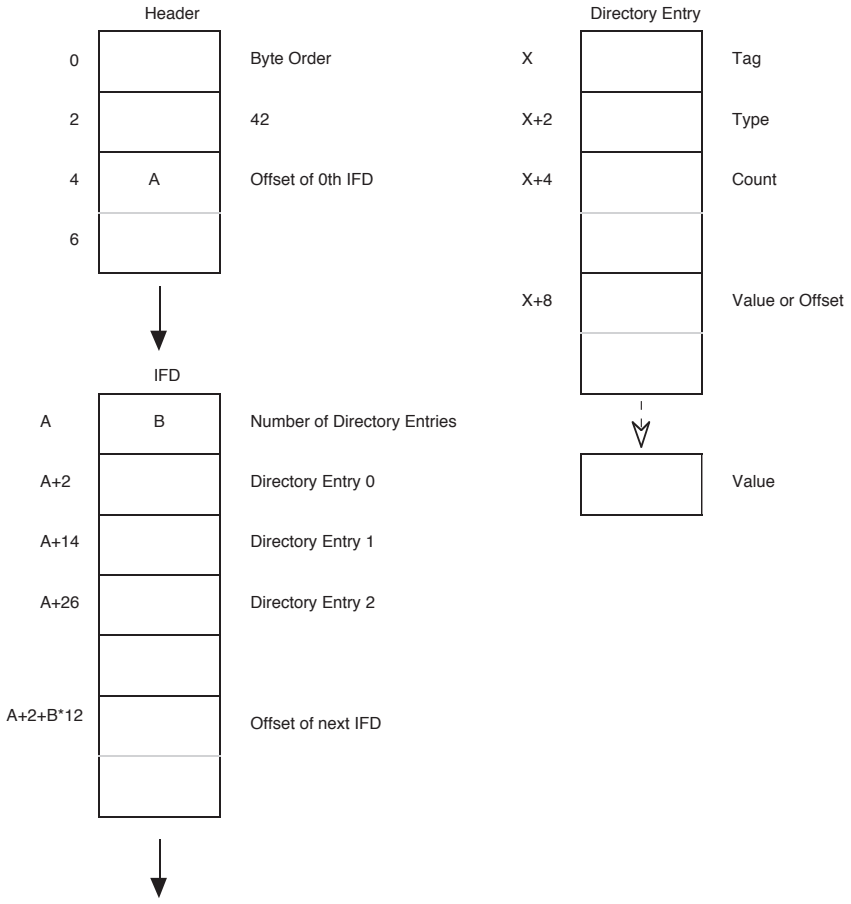


Abbildung 5.8: Die Abbildung zeigt die Struktur, in der die TIFF-Tags abgelegt werden. Mehrere Tags werden in einem Image File Directory (IFD) aggregiert. [*Aldus Corporation*, 1992]

Im Zuge der Verbreitung von Digitalkameras wurde zum Zweck der Speicherung von Zusatzinformationen das EXIF-Format entwickelt, das wir in Abschnitt 5.2.7.3 betrachten. Die EXIF-Tags werden von den meisten Programmen „verstanden“.

Das TIF-Format bietet die Möglichkeit, mit einem Tag auf ein EXIF-Image-File-Directory zu verweisen. Viele Anwendungen unterstützen dieses Verweistag.

5.2.7.3 EXchangeable Image File Format

Wie wir in Kapitel 2 ausführlich betrachtet haben, ist es für viele der vorgestellten Algorithmen notwendig, die Belichtungszeit und die Blende zu kennen.

Wie wir in Abschnitt 5.4.2 sehen werden, ist es schwierig, diese Werte zu ermitteln. Wenn wir die Werte jedoch ermittelt haben, möchten wir sie an die Fremdanwendung weitergeben.

Mit dem „EXchangeable Image File Format“ (EXIFF) [*JEITA*, 2002] der Japan Electronics and Information Technology Industries Association [*JEITA*, 2002] wurde 1992 ein Quasi-Standard verabschiedet, der ein Format spezifiziert, wie Metainformation[160] in einem Bild abgelegt werden kann. Die EXIFF-Information wird heute in nahezu jeder Digitalkamera verwendet um Informationen wie Belichtungszeit, Blende, Fokus, Objektentfernung etc. zu speichern.

In Abbildung 5.9 ist der Aufbau von EXIFF gezeigt. Wir sehen, dass aus unserem TIFF-Header auf den Anfang der EXIFF-Struktur verwiesen wird. Der Aufbau der EXIFF-Information ist sehr ähnlich zu demjenigen der TIFF-Tags (vergleiche Abbildung 5.8). Wie oben bereits geschrieben, besteht der Unterschied darin, dass die EXIF-Tags im Gegensatz zu erweiterten TIFF-Tags von vielen Grafikanwendungen interpretiert werden, da nahezu alle Digitalkamerabilder Information in diesem Format beinhalten.

Der Grund, warum von den Digitalkameraherstellern nicht TIFF verwendet wurde, ist vermutlich derjenige, dass die Hersteller nicht im quasi proprietären Format von Adobe speichern wollten und daher direkt als JPEG speichern. Da JPEG allerdings die Tag-Funktionalität von TIFF fehlte, wurde sie mit EXIFF hinzugefügt.

Unsere Fremdanwendung liest EXIFF-Tags aus. Wir werden daher TIFF mit EXIFF-Erweiterung zum Datenaustausch verwenden.

5.2.7.4 JPEG

Die meisten heute im Internet verbreiteten Bilddaten liegen im JPEG-Format [*Wallace*, 1991; *ITU*, 1993] vor. Die Abkürzung JPEG steht für „Joint Photographic Experts Group“. Dieses Gremium entwickelte die so benannte, 1992 vorgestellte, Norm ISO/IEC 10918-1, die auch als CCITT Recommendation T.81 bekannt ist.

Auch wir werden das JPEG-Format verwenden, um unsere tonegemappten LDR-Bilder über das Web zu publizieren.

Das TIF-Format ermöglicht *verlustlose* Kompression. Dies bedeutet, dass die Bildinformation zum Speichern verkleinert wird und dabei ein solcher Algorithmus angewandt wird, der es erlaubt, die Daten so wieder zu entpacken, dass sie den Originaldaten entsprechen.

[160] Mit Metainformation ist in diesem Fall nicht direkt notwendige Information gemeint. Dies ist beispielsweise die angesprochene Belichtung und Blendeneinstellung. Es können aber auch GPS-Koordinaten oder ähnliches gespeichert werden. Die für die Darstellung des Bildes relevanten Informationen finden sich schon in den Tags des TIFF-Bildes.

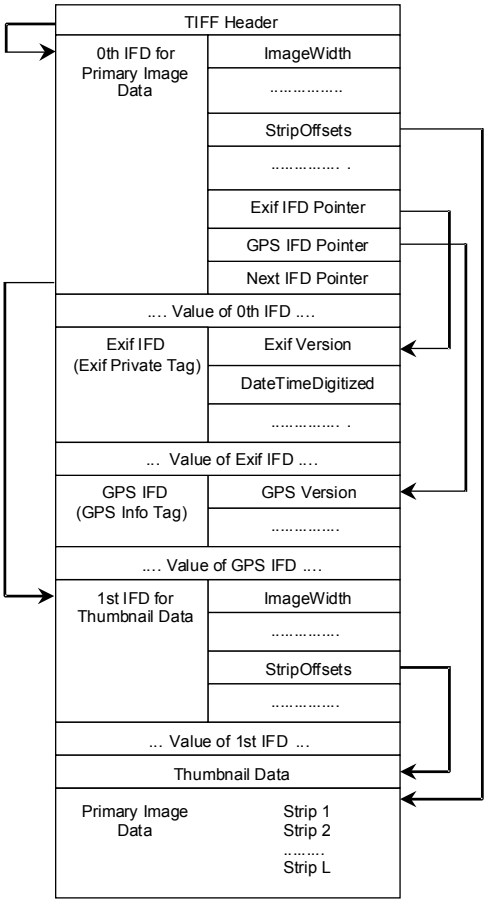


Abbildung 5.9: Das Schaubild veranschaulicht die Lage der EXIFF-Information innerhalb einer TIFF-Bilddatei. [*JEITA*, 2002]

JPEG dagegen realisiert eine *verlustbehaftete* Kompression[161]. Zur Anzeige müssen die Bilddaten dekomprimiert werden. Ein dekomprimiertes JPEG entspricht fast nie dem Originalbild. Dieser Nachteil wird gerne in Kauf genommen, da ein JPEG um ein Vielfaches kleiner (ca. 40:1) als ein TIFF-Bild ist.

Das Blockdiagramm in Abbildung 5.10 zeigt den Ablauf bei der verlustbehafteten JPEG-Kompression. Wir betrachten nicht die genauen Abläufe sondern lediglich grob das Vorgehen:

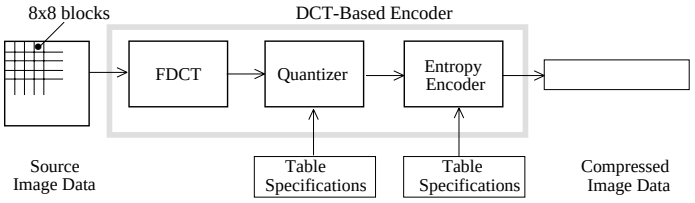


Abbildung 5.10: Die Abbildung zeigt die Hauptverarbeitungsschritte zur Bildkompression nach dem JPEG-Standard. [*Wallace*, 1991]

Zuerst[162] wird das Bild in 8x8 Pixel große Blöcke zerlegt. Diese Größe wurde gewählt, weil sie ein gutes Verhältnis von Rechenaufwand zu Qualität erlaubt.

Durch diese Einteilung wird die nun folgende diskrete Cosinustransformation (DCT) stark beschleunigt[163]. Diese Transformation wandelt das Signal, ähnlich der in Kapitel 4 vorgestellten Fourier-Transformation, in den Frequenzraum. Wie Tonemapping-Kapitel bereits ausgeführt, ist der Sinn hier ebenfalls, nicht so stark wahrgenommene Signalteile bei der Quantisierung eher mit Fehlern zu belegen als stark wahrgenommene.

Der Fehler entsteht bei der **Quantisierung**. In diesem Schritt werden die wertkontinuierlichen Signale auf diskrete Werte abgebildet. Mehrere kontinuierliche Werte werden so auf einen diskreten abgebildet.

Üblicherweise wählt man bestimmte disjunkte Intervalle und legt für jedes Intervall einen Repräsentanten fest. Alle Werte, die in dieses Intervall fallen, werden fortan von diesem einen Wert vertreten. Dadurch reduziert sich das Datenvolumen stark.

Da dem Empfänger der quantisierten Werte allerdings nicht bekannt ist, welcher Wert innerhalb des Intervalls auf den gespeicherten Wert abgebildet wurde, entsteht dort ein Fehler.

Bei **gleichförmig**er Quantisierung sind alle Quantisierungsintervalle gleich groß. Der Fehler ist damit kleiner als die halbe Intervallbreite.

Wie wir mehrfach in der Arbeit gesehen haben (vergleiche zum Beispiel Abschnitt 2.2.1 und Abbildung 2.8), nehmen Menschen Dinge nicht absolut sondern relativ wahr. Daher bietet es sich oft an, logarithmische Skalen zu verwenden, da räumlich gleiche Schritte innerhalb solch einer Skala ungefähr gleichen wahrgenommenen Schritten entsprechen, obwohl die absoluten Unterschiede bei größeren Werten viel größer sind (Abbildung 4.11 ist ein gutes Beispiel für diese Aussage).

Oft verwendet man daher **ungleichförmig**e Quantisierung (vergleiche Abbildung 2.8). Bei dieser Arte der Quantisierung sind die Quantisierungsintervalle nicht gleich groß.

Will man Daten so quantisieren, dass der wahrgenommene Fehler für jedes Intervall gleich groß ist, so muss man diejenigen Intervalle, die in Bereiche mit kleinen Absolutwerten des

[161] JPEG erlaubt auch verlustlose Kompression mithilfe eines Prediktors. [*Wallace*, 1991]

[162] Zunächst wird meist der Farbraum nach YUV transformiert, da dadurch ein Subsampling der UV-Komponenten (Farbe) möglich wird (4:2:2, 4:1:1). Das Helligkeitssignal (Y) wird vollständig übertragen.

[163] Insbesondere kann die FDCT, die Fast-Discrete-Cosinus-Transformation angewandt werden, da wir eine gerade Anzahl an Werten pro Richtung und Kanal haben.

Eingangssignals fallen, kleiner machen und diejenigen, die in Bereiche mit großen Absolutwerten fallen, vergrößern.

Die Quantisierung ist dadurch für schwache Signale fein, für starke Signale grob. Der Absolute Fehler wird dadurch in den großen Intervallen viel größer als in den kleinen. Relativ zur Signalstärke bleibt er allerdings gleich.

Bei JPEG wird ungleichförmige Quantisierung angewandt.

Abschließend werden die Daten noch umsortiert und nach ihrer Entropie, also ihrem Informationsgehalt –beispielsweise der Zeichenhäufigkeit– kodiert.

5.2.7.5 HDR-Bildformate

Die Fremdanwendung generiert aus den LDR-Eingabedaten HDR-Ausgabedaten.

Wie wir aus Kapitel 2 wissen, lassen sich HDR-Daten nicht verlustlos in einem Format speichern, das lediglich 8Bit Helligkeitsstufen vorsieht (z.B. JPEG). Eine Diskussion, wie viel Information wir speichern müssen, findet sich beispielsweise in [*Ward*, 2001].

In Abschnitt 5.2.7.2 haben wir das TIF-Format kennengelernt. Als vielseitiges Containerformat bietet das Format ebenfalls Unterstützung zum Speichern von HDR-Bildern. Auch in diesem Umfeld stellt es das Standardformat für verlustloses Speichern dar.

Die Farbwerte werden pro Farbkanal als 32-Bit[164] IEEE-754-Gleitkommazahlen gespeichert.

Dadurch ergeben sich große Dateien. Die Gleitkommazahlen lassen sich nur schwach komprimieren, da die Bit nahezu gleichverteilt sind.

Weitere verbreitete Formate sind das HDR-Format (.hdr) und das EXR-Format (.exr). Beide Formate enkodieren die HDR-Helligkeitswerte mit weniger Bit als das 96Bit-TIFF (32Bit pro Farbkanal).

Wir betrachten kurz das HDR-Format, da unsere Fremdanwendung dieses benutzt:

Das HDR-Format wurde für ein Renderingsystem[165] entwickelt. Es speichert Lauflängenkodierte (Run Length Encoded, RLE)[166] 4 Byte lange RGBE- bzw. XYZE-Pixelwerte. RGB und XYZ sind hierbei unterschiedliche Farbmodelle, die wir bereits in Abschnitt 2.2.1 kennen gelernt haben.

Wir betrachten die Enkodierung real gemessener HDR-Farbwerte:

$$(R_W, G_W, B_W) \mapsto (R, G, B, E)$$

$$\begin{aligned} E &= \lceil \log_2 (\max(R_W, G_W, B_W)) + 128 \rceil \\ R &= \left\lfloor \frac{256 R_W}{2^{E-128}} \right\rfloor \\ G &= \left\lfloor \frac{256 G_W}{2^{E-128}} \right\rfloor \\ B &= \left\lfloor \frac{256 B_W}{2^{E-128}} \right\rfloor \end{aligned}$$

Die Dekodierung kehrt den dargestellten Vorgang um.

[164] Von Adobe wurden ebenfalls Abwandlungen mit nur 24 oder 16Bit festgelegt. [*Adobe Systems Incorporated*, 2005]

[165] Radiance von *Larson und Shakespeare* [1998].

[166] Lauflängenkodierung bedeutet, dass bei sich wiederholenden Zeichen innerhalb der Daten nur das Zeichen und die Länge der Sequenz gespeichert werden. Schema: 0000000000000000 ↦ 0(16).

Neben den reinen HDR-Bildformaten gibt es auch solche, die sowohl ein anzeigefähiges LDR als auch Zusatzinformation für die Darstellung des LDR beinhalten:

Ward und Simmons [2004] teilen die Bildinformation in einen tonegemappten LDR-Teil und die Differenzinformation dieses Bildes zum HDRI. Beides speichern sie in einem standardkonformaen JPEG, das damit von jedem JPEG-fähigen Programm geöffnet werden kann.

Zum Speichern des Differenzbildes verwenden sie einen so genannten „JPEG-Marker“. Ein solcher Marker hat eine maximale Größe von 64KiB[167]. Die Differenzinformation kodieren die Autoren in diese Größe – unabhängig von der Dimension des Quellbildes. Dies erreichen sie, indem sie die Information skalieren und ebenfalls JPEG-komprimieren.

Der Vorgang der JPEG-Komprimierung ist verlustbehaftet, wie wir in Abschnitt 5.2.7.4 gesehen haben. Da die Autoren sowohl die Bilddaten als auch die Differenzinformation verlustbehaftet kodieren, addiert sich der Fehler bei naivem Vorgehen.

Die Autoren schlagen daher vor, zuerst ein Teilbild (tonegemapptes HDR oder Differenzbild) entsprechend zu kodieren und anschließend die –nun bekannten– entstandenen Fehler im noch nicht komprimierten Teilbild auszugleichen, so dass bei Kombination des Originalteilbildes mit dem komprimierten anderen Teilbild das Originalbild entsteht. Anschließend wird der korrigierte, noch nicht komprimierte Teil komprimiert.

So erhält man beim Dekomprimieren ein Bildergebnis, das nur den Fehler einer (statt zwei) JPEG-Kompressionen enthält.

Komprimiert man zuerst das Differenzbild und korrigiert anschließend das LDR, so erhält man eine bessere Rekonstruktion des HDRI. Dies ist naheliegend, da die Auflösung des LDR-Teils viel höher ist. Dafür ist allerdings das LDR schlechter. Das linke Bild in Abbildung 5.11 verdeutlicht dies. Komprimiert man zuerst das LDR, so ist dieses besser, das HDRI allerdings schlechter (siehe Abbildung 5.11 rechts).

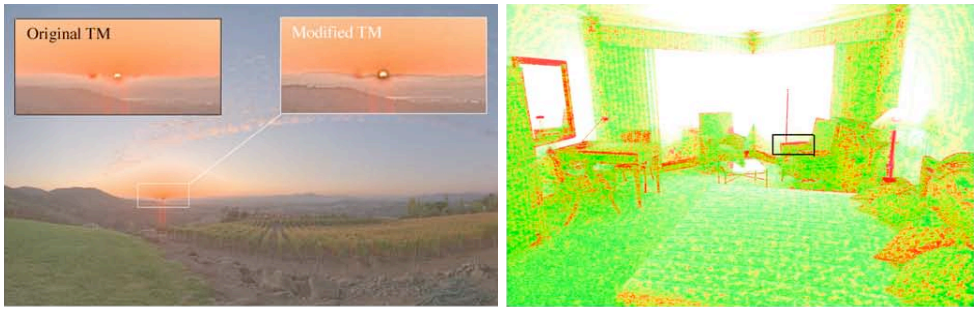


Abbildung 5.11: Die Abbildung zeigt links den Fehler, der entsteht, wenn man zuerst das Differenzbild JPEG komprimiert und anschließend das geänderte HDR tonemappt (rechts oben). Das LDR ist in diesem Fall fehlerbehafteter. Auf der rechten Seite ist eine Metrik visualisiert, die angibt, welche Fehler zu mehr als 75% Wahrscheinlichkeit anders als das Original wahrgenommen werden (Rot stellt den stärksten Fehler dar). Nicht das LDR hat in diesem Fall zusätzliche Fehler sondern das HDRI. [*Ward und Simmons*, 2004]

Hassan und Carletta [2007] entwickeln ein ähnliches Verfahren für JPEG2000. Die Besonderheit an ihrem Verfahren ist die Echtzeitfähigkeit durch Implementierung auf einem Field Programmable Gate Array (FPGA)[168].

[167] KiB=KibiByte=2^{10} Byte.
[168] Ein FPGA ist ein vollkommen konfigurierbarer (programmierbarer) integrierter Schaltkreis.

In [*Reinhard et al.*, 2005, 89f] werden noch weitere Bildformate sowie deren Vor- und Nachteile behandelt. Als Beurteilungskriterien werden genannt:

- Abdeckung des sichtbaren Farbspektrums
- Abdecken des Helligkeitsspektrums
- Nicht sichtbare Quantisierungsstufen
- Geringer Speicherverbrauch der Datei
- Rechenzeitschonendes (De-)Kodieren

Der Artikel [*Ward*, 2005] eines der Autoren von [*Reinhard et al.*, 2005] gibt ebenfalls einen guten Überblick über die Verfahren und die Hintergründe der aufgelisteten Kriterien. Aus dieser Veröffentlichung stammt der Vergleich in Abbildung 5.12.

Encoding	Covers Gamut	Bits / pixel	Dynamic Range	Quant. Step
sRGB	No	24	1.6 (1.0:0.025)	Variable
Pixar Log	No	33	3.8 (25.0:0.004)	0.4%
RGBE XYZE	No Yes	32	76 (10^{38}:10^{-38})	1%
LogLuv 24	Yes	24	4.8 (15.9:0.00025)	1.1%
LogLuv 32	Yes	32	38 (10^{19}:10^{-20})	0.3%
EXR	Yes	48	10.7 (65000:0.0000012)	0.1%
scRGB	Yes	48	3.5 (7.5:0.0023)	Variable
scRGB-nl scYCC-nl	Yes Yes	36	3.2 (6.2:0.0039)	Variable

Abbildung 5.12: In der Tabelle sehen wir die verschiedenen Verfahren (Encoding), ob sie das komplette Farbspektrum abdecken (Gamut), wie viele Bit/ Pixel verwendet werden, welchen Wertebereich (Dynamic Range) die Verfahren abdecken [je in ()] und wie groß die Quantisierungsintervalle bei den einzelnen Verfahren sind. [*Ward*, 2005]

Wie wir sehen, deckt das von unserer Fremdsoftware erzeugte RGBE-HDR-Format nicht das komplette sichtbare Farbspektrum ab.

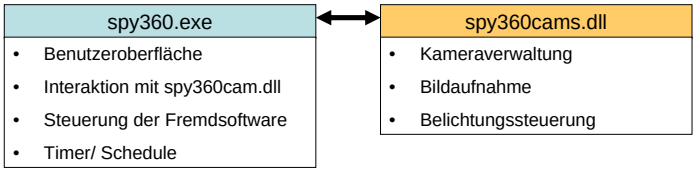
5.3 Modellierung

Wie wir bereits einleitend gesehen haben, muss die Software folgendes leisten („Lastenheft“):

- Aufnahme einer Belichtungsreihe mit den angeschlossenen Webcams
- Steuerung der Software zum Erstellen des HDRI aus der LDRI-Reihe
- Steuerung der Software zum Stitching
- Steuerung der Software zum Tonemappen
- Zusatzfunktionalität wie Speichern, Aufräumen und Veröffentlichen

Der Ablauf sollte dabei vollkommen automatisiert erfolgen können.

Zur Umsetzung dieser Anforderungen („Pflichtenheft“) werden wir die Anwendung in zwei Teile trennen: Den Steuerungsteil mit der graphischen Benutzeroberfläche (GUI) und den Teil, der mit der Kamerahardware (genauer dem Kameratreiber bzw. DirectShow) interagiert und die Aufnahmen macht:



Zur vollständigen Trennung zwischen Steuerungsanwendung und Kamerateil werden wir letzteren als externe Dynamic Link Library (DLL, Laufzeitbibliothek) realisieren (siehe Abschnitt 5.2.4).

Weiterhin wollen wir unsere Anwendung nebenläufig (multithreaded) programmieren, da wir verschiedene Eingabegeräte ansteuern wollen und dadurch ein Geschwindigkeitsvorteil aufgrund der Asynchronität der verschiedenen Programmteile zu erwarten ist.

Durch diese Entscheidung ergeben sich zusätzliche Probleme der Synchronisierung. Über die Nachrichten in Windows steht uns eine Standardmöglichkeit dazu zur Verfügung (vergleiche Abschnitt 5.2.6).

Das zur Aufnahme verwendete DirectShow kann so initialisiert werden, dass es sich nebenläufig verhält. In Abschnitt 5.2.5 haben wir dies bereits betrachtet. Wir verwenden das Multi-threading-/ Free-threading-Modell.

5.3.1 spy360cams.dll

Wir entwerfen zuerst die Kamerasteuerungsbibliothek.

Wir beginnen mit der Außenschnittstelle der Bibliothek, da sie die „Trennlinie“ zwischen der Funktionalität der Anwendung und der Bibliothek spezifiziert. Diese ist in Listing 5.3 abgedruckt.

Die „get“-Methoden liefern jeweils Werte, die „set“-Methoden setzen diese entsprechend. Die „is“-Methoden liefern boolsche Ergebnisse zurück.

Zur Veranschaulichung der Funktionalität wird diese anhand der Funktionsausgaben im Folgenden illustriert.

Über das abgedruckte Interface erfolgt die gesamte Kommunikation mit der Kamerafunktionalität.

```
class DLLEXPORT Cspy360cams{
public:
  // Devices
  size_t  getDeviceAmount();
  CString getDevicesFriendlyName(size_t devNo);

  // Device's PINs (Connectors like Video1, Video2 etc.)
  size_t  getPinAmount(size_t devNo);
  size_t  getRunningPin(size_t devNo);
  CString getPinFriendlyName(size_t devNo, size_t PinNo);

  // PIN's MediaTypes (like 320x640, RGB24)
  size_t  getMtAmount(size_t devNo, size_t PinNo);
  CString getRunningMtFriendlyName(size_t devNo, size_t PinNo);
  CString getMtFriendlyName(size_t devNo,size_t PinNo, size_t MtNo);

  // Preview Window
  void    setPrvWnd(size_t devNo, HWND* wnd);
  void    setPreviewState(size_t devNo, bool setON);
  bool    isPreviewON(size_t devNo);

  // Exposure
  bool    isExpControllable(size_t devNo);
  void    setMultExpShotState(size_t devNo, bool setON);
  bool    isMultExpShotON(size_t devNo);



  // Get the Shots!
  void    goShooting(size_t devNo);

  // Shooting-Thread is calling back...
  /*
   * Since we use a MultiThreaded Environment, we decide to communi-
   * cate via Messages. Windows allows Thread- and Windowmessages.
   * The Messages arrive at the Main-Thread/ -Window.
   * So we have to pass them to the Library.
   */
  void    sampleWaiting(size_t devNo);



  // Un- and Reload the Devices
  void    reInitialize();

  // Clean up the GUI related stuff, the Program wants to exit!
  void    performShutdown();



  // Property Page
  void    showFilterPropertyPage(size_t devNo);
};
```

Listing 5.3: DLLExport.h

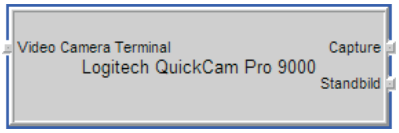


Abbildung 5.13: Die Kamerakomponente eines Filtergraphen.

Wie wir in Abschnitt 5.2.3.1 gesehen haben arbeiten wir mit Geräten innerhalb eines Filtergraphen. Dadurch ergeben sich einige Bezeichnungen. In Abbildung 5.13 sehen wir zur Erläuterung erneut eine Kamerakomponente des Graphen.

Die oberste Gruppe (Zeile 3ff) gibt die Anzahl der verfügbaren Geräte sowie deren „FriendlyName“ zurück. Als FriendlyName bezeichnet man bei der Windows-Programmierung zumeist eine für Menschen aufbereitete Form des Bezeichners. Dieser String wird zumeist vom Hersteller bereitgestellt und besteht oft aus dem Herstellernamen und dem Modelltyp des Gerätes. In Abbildung 5.13 ist der FriendlyName der Bezeichner in der Mitte.

Wie wir in Abschnitt 5.2.3.1 gehört haben, erfolt die Weitergabe der Bilddaten über die Ausgänge des Knotens, die so genannten PINs. Die verfügbaren PINs, der gerade aktive PIN und der FriendlyName (in Abbildung 5.13 ist der FriendlyName der Bezeichner des jeweiligen Ausgangs) werden vom zweiten Funktionsblock (Zeilen 7ff) an die dienstnehmende Anwendung bereitgestellt.

Der dritte Abschnitt (Zeilen 12ff) liefert Information über die von dem jeweiligen PIN bereitgestellten Medientypen. Für die abgebildete Kamera ist das Ergebnis beispielsweise[169]:

```
1  Capture is capable of:
2    [RGB24] 320x240
3    [RGB24] 160x120
4    [RGB24] 176x144
5    [RGB24] 352x288
6    [RGB24] 640x480
7    [RGB24] 800x600
8    [RGB24] 960x720
9    [RGB24] 1600x1200
10   [I420] 320x240
11   [I420] 160x120
12   [I420] 176x144
13   [I420] 352x288
14   [I420] 640x480
15   [I420] 800x600
16   [I420] 960x720
17   [I420] 1600x1200
```

Listing 5.4: Debugausgabe der Anwendung zu den am Capture-PIN verfügbaren Medientypen.

Der „Preview Window“-Abschnitt (Zeilen 17ff) wird für die Vorschau benötigt. Da unsere Bibliothek selbst keine Fenster besitzt, müssen wir uns von der aufrufenden Anwendung ein Fenster geben lassen oder selbst eines erzeugen.

Der Bereich „Exposure“ (Zeilen 22ff) ist für die dienstnehmende Anwendung relevant, da er ihr mitteilt, ob die Bibliothek mit dem jeweiligen Gerät in der Lage ist, die Belichtung zu steuern und damit Aufnahmen bei verschiedenen Belichtungsstufen zu machen.

Die „goShooting“-Methode veranlasst unsere Bibliothek, eine Aufnahmesession zu starten.

[169] Die Medientypen werden in dieser Reihenfolge zurückgeliefert. Der erste Eintrag entspricht zumeist dem Standardmedientyp, der benutzt wird, wenn kein anderer Typ angegeben wird.

„sampleWaiting“ ist eine Funktion, die aufgerufen wird, sobald bei unserer Aplikation die Window-Message (siehe Abschnitt 5.2.6) eintrifft, dass Sampledaten bereitstehen. Näheres hierzu in Abschnitt 5.4.

Der „reInitialize“-Aufruf sorgt dafür, dass alle Geräte in den Ausgangszustand zurückgesetzt werden und insbesondere neue Geräte angezeigt werden.

Der „performShutdown“-Befehl teilt unserer Bibliothek mit, dass die Anwendung und insbesondere deren Fenster zerstört werden. Dies ist notwendig, da die Bibliothek eventuell noch Resourcen der Anwendung hält, die vor dem normalen Destruktoraufruf gelöst werden müssen.

„showFilterPropertyPage“ schließlich zeigt die Eigenschaftenseite des Gerätes an (vergleiche Abbildung 5.22).

Obwohl es durchaus möglich ist, die Schnittstelle in dieser Form vor dem Entwicklungsstart zu spezifizieren, sei an dieser Stelle erwähnt, dass einige der notwendigen Kommunikationspunkte sich erst bei der Programmierung ergeben haben[170].

Eine derart detaillierte Planung im Voraus setzt viel Wissen über die internen Abläufe des Windows-Application-Programming-Interface (WinAPI)[171] und von DirectShow voraus. In Abschnitt 5.2.3 haben wir einen einführend Einblick in die Thematik erhalten.

Die genauere Spezifikation der Klassenstruktur erfordert ebenfalls tiefgehendere Kenntnis von DirectShow. Sie ist in Abbildung 5.14 wiedergegeben. Die Exportschnittstelle (vgl. Listing 5.3) befindet sich auf der linken Seite.

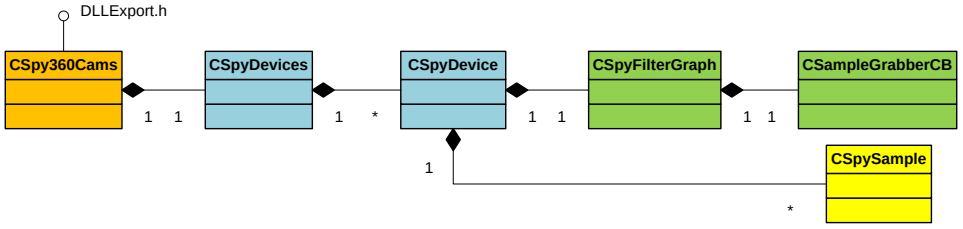


Abbildung 5.14: Die Abbildung zeigt das Klassendiagramm der Bibliothek *spy360cams.dll*.

Anhand der Klassenstruktur sehen wir, wie sich die Funktionalität innerhalb der Bibliothek feingranularer als in Abbildund 5.3 dargestellt verteilt:

Die DLL-Hauptklasse heißt *CSpy360Cams*. Sie übernimmt die Kommunikation der Bibliothek mit der Außenwelt und beinhaltet die Hauptsteuerungslogik.

Die Klasse enthält eine Instanz der Klasse *CSpyDevices*. In dieser Klasse sind die einzelnen Eingabegeräte (in der Regel die Webcams) aggregiert.

Die softwareseitige Repräsentation eines Eingabegerätes ist die Klasse *CSpyDevice*.

Um Bilder von dem Gerät zu erhalten, benötigen wir einen so genannten *FilterGraphen* (vergleiche Abschnitt 5.2.3.1). Dieser wird in der Klasse *CSpyFilterGraph* gekapselt.

Eine Aufnahme wird in *CSpySample* gespeichert.

[170] Beispielsweise ist nicht unbedingt zu erwarten, dass es nicht möglich ist, eine Nachricht an den Bibliotheksthread zu senden. Wie wir im Sequenzdiagramm in Abbildung 5.15 sehen, muss diese Nachricht vom Hauptprogramm an unsere DLL zurückgereicht werden. Dies erfolgt mittels der „sampleWaiting“-Funktion.

[171] Die Windows-API stellt unter Windows die Schnittstelle zum Betriebssystem dar.

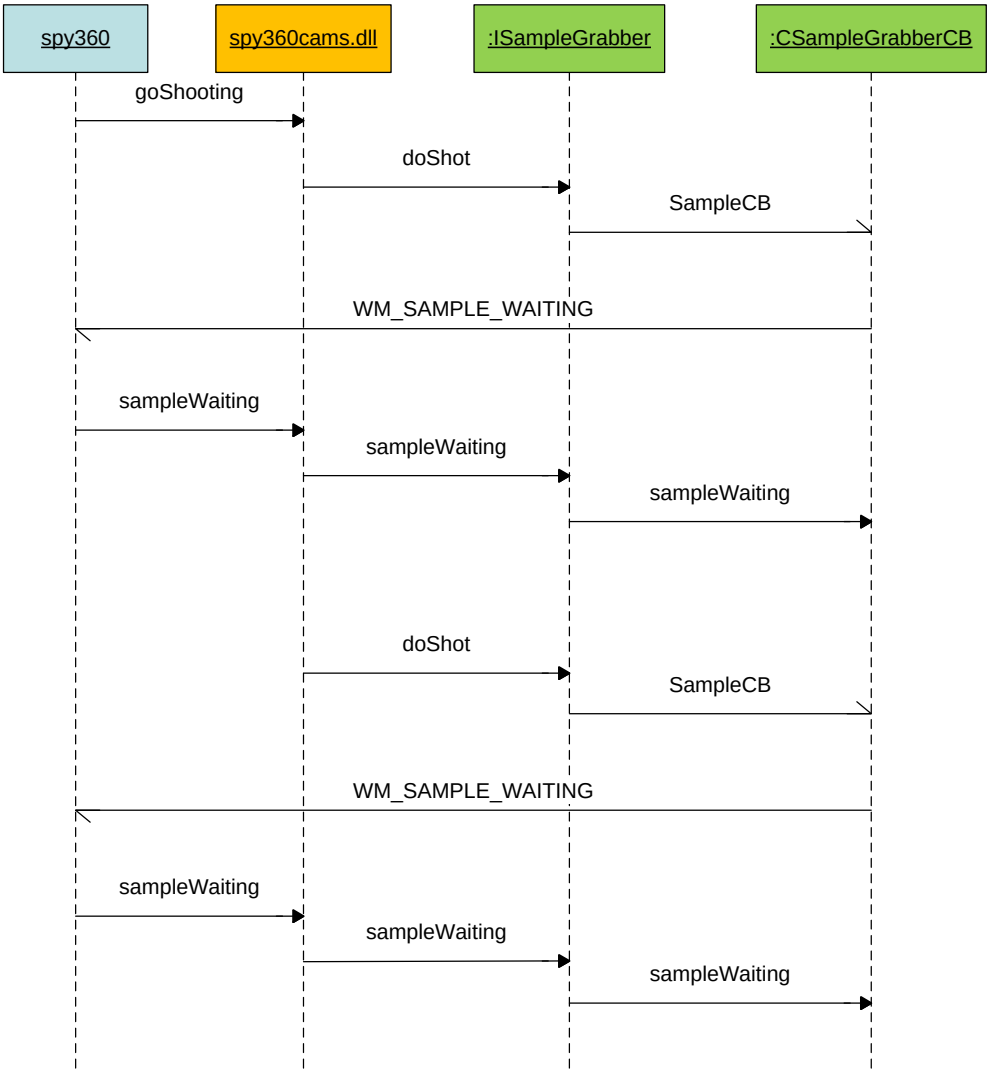


Abbildung 5.15: Die Abbildung zeigt das Sequenzdiagramm einer Aufnahmereihe aus zwei Aufnahmen. Gegebenenfalls werden noch mehrere Aufnahmen von der DLL getriggert. Wir sehen, dass die Nachricht der SampleGrabber-Callbacks an die Hauptapplikation geschickt wird und von dort wieder zurückgereicht werden muss, da wir uns in einer asynchronen Multithreadingumgebung befinden.

5.3.2 spy360/ -cams – gemeinsame Nachrichten

Durch die Kapselung kann die Bibliothek keine Funktionen der Anwendung aufrufen. Es steht auch kein gemeinsamer Speicherbereich zur Verfügung[172]. Wir werden daher den in Abschnitt 5.2.6 vorgestellten Windows-Standardmechanismus zur Komunikation und Synchronisation einsetzen: Nachrichten.

Nachrichten unter Windows sind durch ihre Nummer identifiziert. Wir müssen daher sicherstellen, dass beide Threads dieselben Nummern für die entsprechenden Window-Messages verwenden und diese beim Einbinden der Bibiothek entsprechend registrieren.

Wir benötigen zwei Nachrichten zur Synchronisation der Programmabläufe: Eine erste, die von der Bibliothek generiert wird, sobald von einer Kamera die komplette Belichtungsreihe als Temporärdateien vorliegt und eine zweite, die vom SampleGrabberCB versandt wird, sobald ein Sample in den Puffer aufgenommen wurde.

Um die Anwendung zu informieren, dass von einer Kamera alle temporären LDR-Bilder vorliegen, werden wir eine Nachricht in die MessageQueue der Anwendung einhängen. Diese Nachricht kommt hauptsächlich zum Einsatz, da dies ein eleganter Kommunikationsweg ist.

Die zweite Nachricht dagegen hat den Zweck, die Nebenläufigkeit aufrecht zu erhalten: Wir wollen eine *asynchrone* Benachrichtigung, wenn ein Aufnahme-Thread ein Bild aufgenommen hat, damit die nächsten Verarbeitungsschritte eingeleitet werden können. Dazu soll der Thread ebenfalls eine Nachricht schicken, deren Empfänger der Hauptthread unserer Bibiothek ist.

Leider wird sich herausstellen, dass dies nicht ohne Weiteres funktioniert. Unsere Bibliothek wird nämlich im selben Thread laufen, wie unsere Hauptanwendung[173] (im Gegensatz zu einem echten COM-Objekt, das unabhängig von der Anwendung läuft). Dadurch steht uns allerdings erneut nur die MessageQueue der Anwendung als Nachrichtenempfänger zur Verfügung. Die Anwendung muss unsere Bibliothek daher über den Eingang solcher Nachrichten informieren:

In Abbildung 5.15 sehen wir das Sequenzdiagramm einer Aufnahmereihe. Die Pfeile mit der halben Spitze signalisieren asynchrone Aufrufe. Diese werden über Nachrichten realisiert und sind notwendig, da der SampleGrabber in einem anderen Thread läuft als die Bibliothek und die Anwendung[174]. Die Threads sind insbesondere nicht synchronisiert, wehalb der in Abschnitt 5.2.6 vorgestellte post-Mechanismus zum Einsatz kommt.

Zum Transport der Nummer der jeweils betroffenen Kamera verwenden wir den wParam (vergleiche Abschnitt 5.2.6). In den SDL-Diagrammen (s.u.) ist dies jeweils der „camNo“-Parameter.

Die erste Nachricht muss innerhalb von spy360 dazu führen, dass die Daten weiter verarbeitet werden: Das HDR-Panorama und dessen tonegemappte Version müssen erstellt werden. Anschließend müssen die notwendigen Schritte zur Veröffentlichung der beiden Bildversionen ausgeführt werden.

Die zweite Nachricht muss von der Anwendung an die Bibliothek signalisiert werden, damit diese die nächste Belichtung einleiten kann.

Die folgenden SDL-Diagramme in den Abbildungen 5.16, 5.18 und 5.17 spezifizieren den Ablauf bei Eintreffen der entsprechenden Benachrichtigungen genauer.

Unsere beiden Nachrichten sind dort als „DONE“ und „SAMPLE_WAITING“ bezeichnet.

Die Diagramme visualisieren jeweils die für die Aufnahme wichtigen Teilaspekte des SampleGrabber-Callbacks, der spy360cams-Bibliothek und der spy360-Anwendung. Da die Diagramme nur Teilaspekte abbilden, haben sie unter Anderem kein Ende.

[172] Dieser könnte zur Kommunikation und Synchronisation eingesetzt werden.
[173] Die Aufnahme selbst erfolgt in einem anderen Thread.
[174] Bibliothek und Anwendung laufen im selben Thread.

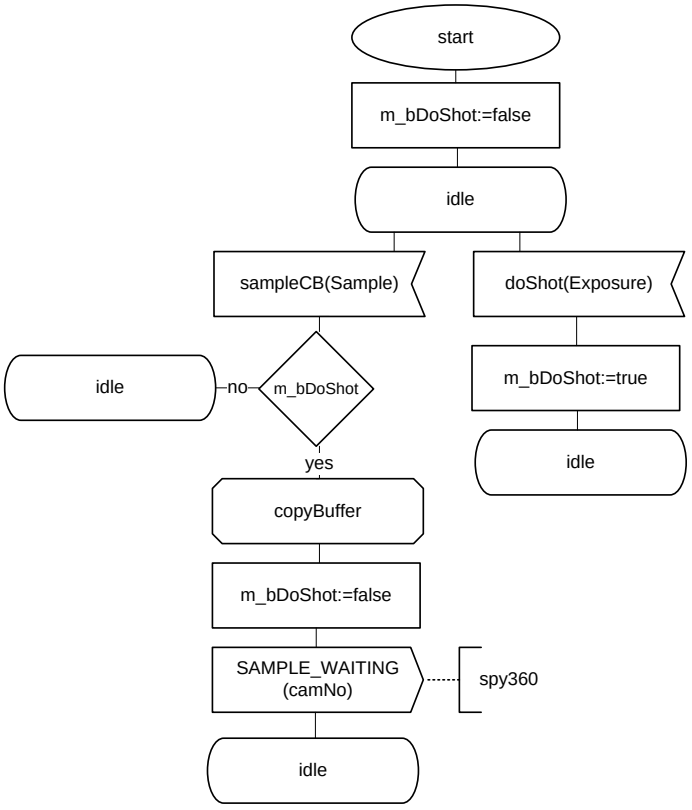


Abbildung 5.16: Das SDL-Diagramm spezifiziert einen Teil der Funktionalität des SampleGrabber-CallBack.

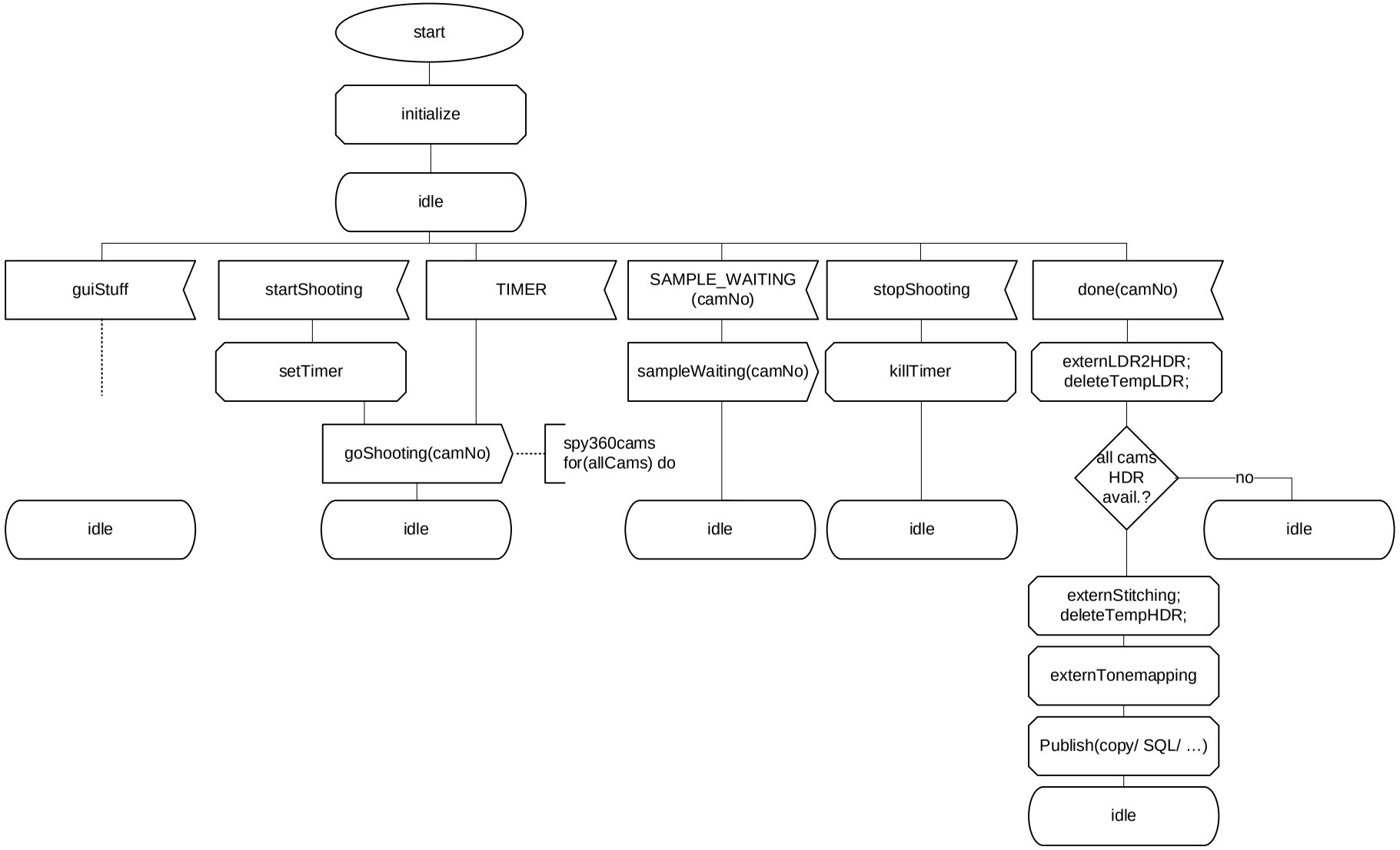


Abbildung 5.17: Das SDL-Diagramm spezifiziert einen Teil der Funktionalität der spy360-Anwendung.

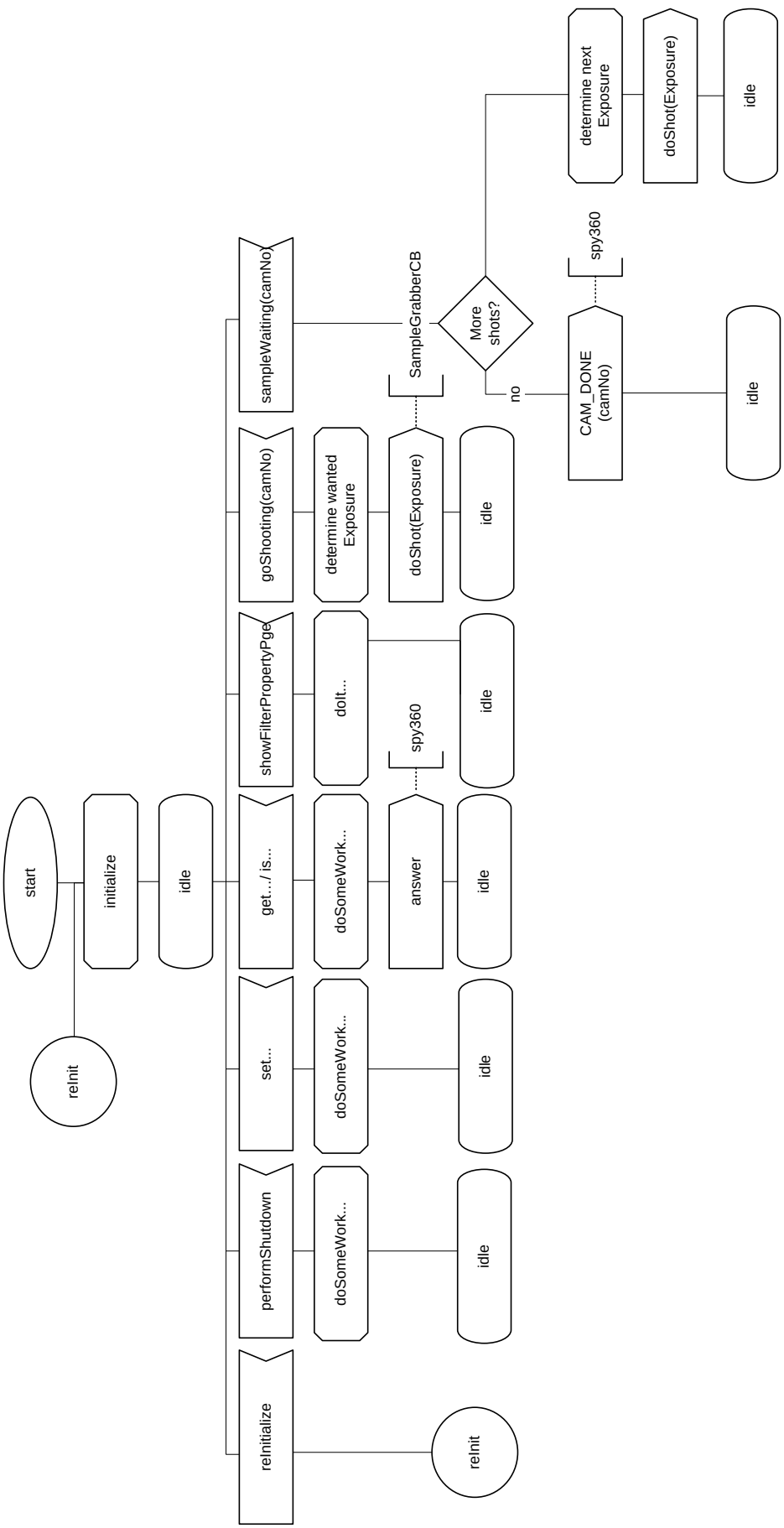


Abbildung 5.18: Das SDL-Diagramm spezifiziert einen Teil der Funktionalität der spy360cams.dll.

5.3.3 spy360

Über die Bibliotheksfunktionalität haben wir nun bereits große Teile der Funktionalität der Anwendung spezifiziert.

Wir werden dies in diesem Teil noch etwas präzisieren. Darüber hinaus kann die Anwendung beliebige Funktionalität beinhalten. Insbesondere zur Nachbearbeitung der Daten, sobald das fertige Panorama zur Verfügung steht. Auf diese Funktionalität gehen wir bei der Spezifikation an dieser Stelle nicht ein. Zum Teil findet sich solche Funktionalität im Ausblick in Kapitel 7.

Als zusätzliche Mechanismen werden wir Timer und Möglichkeiten zum Aufruf der externen Programme benötigen:

Als Timer (vergleiche Abbildung 5.17) wird der entsprechende Windows-Mechanismus zum Einsatz kommen. Der Windows-Timer ruft, sobald er abgelaufen ist, eine bestimmte Nachricht auf, die wir in unserer Anwendung behandeln können. Für hochpräzise Messungen ist dieser Mechanismus ungeeignet. Für unsere Anwendung reicht die Präzision aus.

Die Kommunikation mit der Fremdsoftware soll über je einen Shellaufruf erfolgen. Damit können wir die Verarbeitung auf einfache Weise durch Hinzufügen weiterer Kommandos erweitern.

Zur genaueren Spezifikation eines Aufnahmeablaufs fassen wir diesen in Worte. Als Veranschaulichung dienen insbesondere das SDL-Diagramm in Abbildung 5.17 und das Sequenzdiagramm in Abbildung 5.15:

Für die Aufnahme muss die Anwendung den Prozess anstoßen, indem sie die „goShooting“-Funktion der spy360cams-Bibliothek aufruft.

Anschließend muss sie die eintreffenden „SAMPLE_WAITING“-Nachrichten als Funktionsaufruf an die Bibliothek weiterleiten (vergleiche Abbildung 5.15). Die Bibliothek wird der Anwendung mitteilen, sobald alle Aufnahmen für eine Kamera gemacht sind („DONE“).

Erfolgt diese Mitteilung so muss die Anwendung das externe Programm zum Mapping aufrufen, das aus den LDR-Temporärdateien ein HDR berechnet. Anschließend muss die Anwendung die Temporärdateien löschen.

Sind für alle Kameras HDR-Dateien vorhanden, muss die Anwendung das externe Programm zum Berechnen des HDR-Panoramas aufrufen und anschließend die temporären HDR-Dateien löschen.

Dann muss die Anwendung das HDR-Panorama tonemappen.

An dieser Stelle stehen ein HDR-Panorama sowie, je nach Anzahl der gewünschen verschiedenen Tonemappingoperatoren[175], n gemappte LDR-Panoramen.

Am Ende müssen diese Dateien an ihren Zielort kopiert werden und eventuelle andere finale Operationen wie beispielsweise das Publizieren der Aufnahme in eine Datenbank ausgeführt werden.

Durch den Timer angestoßen startet der Prozess von Neuem.

Wie wir den SDL-Spezifikationen entnehmen können, wird keine Nachricht von der Bibliothek versandt, dass *alle* Kameras ihre LDR-Belichtungsreihe gespeichert haben. Stattdessen wird dies für jede Kamera *einzeln* mitgeteilt.

Dies hat performancetechnische Gründe: Da wir asynchron arbeiten, werden auch die Kameras zu unterschiedlichen Zeiten fertig werden. Sobald eine Kamera eine komplette Belichtungsreihe gespeichert hat, kann diese jedoch in ein HDR verwandelt werden.

[175] Beispielsweise kan man einen realistischen und einen bewusst unrealistischen einsetzen um zusätzlich ein künstlerisches Ergebnis zu haben. (Vergleiche Kapitel 4.)

Im anderen Fall einer einzigen Nachricht, die mitteilt, dass alle Kamera-LDR-Belichtungsreihen vorliegen, würde dieser Prozess für alle Kameras zugleich angestoßen und damit langsamer, da die komplette Rechenzeit sich auf alle $LDR \rightarrow HDR$-Verarbeitungen gleichzeitig verteilen würde.

Bei der hier spezifizierten Realisierung können wir davon ausgehen, dass die Prozesse sich nicht die gesamte Zeit überlagern, da sie zu unterschiedlichen Startzeitpunkten gestartet wurden. Die Verarbeitung wird somit schneller erfolgen. Insbesondere gilt dies auch für Multiprozessorsysteme, da auch dort die Rechenkerne dann zu unterschiedlichen Zeitpunkten wieder freigegeben werden können.

5.3.4 Belichtungseinstellung

Der entscheidende Punkt für unsere Anwendung ist die Kontrolle der Belichtung. Dabei gibt es diverse Probleme, wie wir in Abschnitt 5.4.2 sehen werden. Unabhängig von den dort genannten technischen Schwierigkeiten entsteht das nachfolgend beschriebene Problem, für dessen Lösung wir an dieser Stelle einen Algorithmus entwickeln.

In Abbildung 5.19 sehen wir drei nacheinander aufgenommene Belichtungsreihen. Die Belichtungen sind jeweils Automatik, $\frac{1}{500}$, $\frac{1}{250}$, $\frac{1}{125}$, $\frac{1}{63}$, $\frac{1}{31}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{5}$ und $\frac{1}{5}$.

Die beiden linken Messreihen sind entstanden, indem die Kamera jeweils auf eine neue Belichtung gesetzt und sofort anschließend ein Sample entnommen wurde. Bei einer Fotokamera wäre dies das normale Vorgehen, da die Blende genau bei der Aufnahme richtig eingestellt sein wird. Da wir allerdings mit Webcams arbeiten, die permanent einen Videostrom liefern, müssen wir warten, bis die Belichtung entsprechend eingestellt ist[176].

Die eingestellte Belichtung einer Webcam kann über eine entsprechende DirectShow-Methode ausgelesen werden. Leider misst eine normale Webcam nicht den Wert, den sie zurückgibt sondern gibt scheinbar direkt den aktuell eingestellten Wert zurück. Sobald wir den gewünschten Belichtungswert einstellen, wird dieser als aktuelle Belichtung zurückgeliefert, unabhängig davon, ob sich die Kamera schon darauf eingestellt hat.

Die linken beiden Aufnahmereihen in Abbildung 5.19 zeigen genau dies: Die Kamera hat den Einstellungsvorgang noch nicht abgeschlossen und wir entnehmen unser Sample.

Die auf dem Bild jeweils eingezeichneten waagrechten Balken stellen in Rot die aktuelle Helligkeit, in Blau den Mittelwert über die letzten Helligkeiten und in Grün die letzte Helligkeit dar (s.u.).

Sind der rote und der grüne Balken nicht auf einer Höhe, so bedeutet dies, dass das aktuelle Bild das erste mit dieser Belichtung ist. Stimmt der blaue Balken nicht mit den anderen überein, so bedeutet dies, dass das drittletzte Sample weder dem letzten noch dem aktuellen entsprach. Aus diesem Umstand lässt sich schließen, dass sich die Kamera gerade nicht in einem stabilen Zustand befindet.

Wir müssen also ein Verfahren entwickeln, mit dem es möglich ist, festzustellen, ob sich die Kamera auf eine gewünschte Belichtungsänderung eingestellt hat.

5.3.4.1 BÄD – ein Belichtungsänderungsdetektor

In Kapitel 2 haben wir Algorithmen zur Responsekurvenbestimmung betrachtet. Einige dieser Algorithmen arbeiten auf Histogrammen. Wie wir bereits in Abschnitt 3.4.1 besprochen haben beschreibt ein Histogramm die Häufigkeit von Messwerten. In Abbildung 5.20 sehen wir ein Histogramm.

Die x-Achse stellt in aufsteigender Form die Helligkeitswerte dar, die y-Achse die Häufung der Vorkommnisse des jeweiligen Helligkeitswertes. Wenn sich nun die Belichtung ändert, so wird sich das Histogramm verschieben. Wenn das gesamte Bild heller wird, wandern die „Berge“ nach rechts und umgekehrt.

In einer ersten Version der Software wurde das Histogramm zur Entscheidung herangezogen, ob sich die Kamera auf eine Belichtung eingestellt hat. Um nicht jedes Mal alle Werte berücksichtigen zu müssen, wurde ein Hashwert, also ein für das Histogramm aus unserer Sicht charakteristischer skalarer Wert, bestimmt.

In Anlehnung an die Mean Threshhold Bitmap Alignment-Technik aus Abschnitt 2.4.1 wurde als Hashwert die Mitte des kumulativen Histogramms gewählt. Dies bedeutet, dass genau

[176] Webcams haben in der Regel keine Blende. Die Belichtung wird zumeist über eine Regelung der Integrationszeit der CCD- oder CMOS-Sensorelemente erreicht. (Vergleiche Abschnitt 2.2.2.)

Abbildung 5.19: Links: Zwei Belichtungsreihen, die entstehen, wenn man die Kamera anweist, die nächste Belichtung einzustellen und sofort die nächste Aufnahme zu machen, sobald ein Sample gespeichert ist. Rechts: Diese (erwünschte) Reihe entsteht, wenn der BÄD eingeschaltet ist. Bei den obersten Bildern ist jeweils die Belichtungsautomatik aktiviert. Nach unten folgen die Belichtungsstufen $\frac{1}{500}$, $\frac{1}{250}$, $\frac{1}{125}$, $\frac{1}{63}$, $\frac{1}{31}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{5}$, $\frac{1}{5}$.

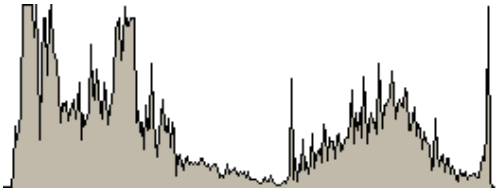


Abbildung 5.20: Die Abbildung zeigt ein Histogramm zu Abbildung 5.21.

der Intensitätswert als charakteristisch ausgegeben wurde, an dem links genauso viele Pixel mit geringerer Intensität liegen wie rechts solche mit höherer Intensität.

Dieses Vorgehen hat denselben Vorteil, den es auch beim Alignment hat: Die Mitte des kumulativen Histogramms ist, bei sich nicht ändernder Belichtung, sehr stabil. Geringe Helligkeitsunterschiede, wie sie zum Beispiel durch Aufnahmestörungen (vergleiche Abschnitt 2.2.2.1) entstehen, fallen nicht störend ins Gewicht.

Entscheidender Nachteil ist jedoch die aufwendige Berechnung: Um ein kumulatives Histogramm zu berechnen, muss jedes Pixel einmal betrachtet werden. Die Laufzeit wird also bestimmt durch das Produkt Höhe $\times$ Breite. Bei einer Auflösung von 1600x1200 Pixeln, wie sie unsere Kameras liefern, ist dies recht viel.

Daher wurde ein anderer Belichtungsänderungsdetektor entwickelt, der das Ziel hat, schnell und zuverlässig zu arbeiten. Der neue Algorithmus orientiert sich an der Technik moderner Spiegelreflexkameras. Diese messen die Belichtung anhand von verschiedenen Messpunkten innerhalb des Sichtfeldes der Kamera.

Da wir digitale Helligkeitswerte vorliegen haben, können wir beliebig viele Messpunkte zu Rate ziehen. Bei der konkreten Implementierung wurden 64, gleich über das Bild verteilte, Messpunkte gewählt. In Abbildung 5.21 sind diese als grüne Punkte eingezeichnet.

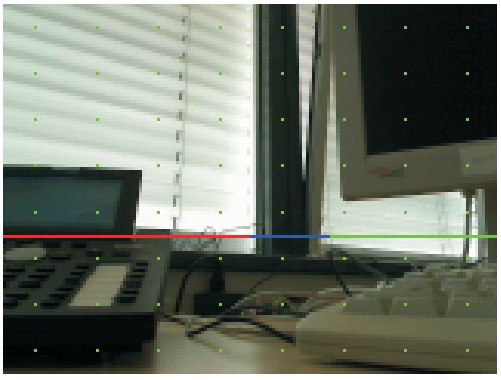


Abbildung 5.21: Die Abbildung zeigt die Messpunkte sowie die normierte Summe über die Messpunkte (rot), den Mittelwert über die letzten drei Messungen (blau) und den letzten Messwert (grün).

Die Helligkeitswerte sind gerade die Grauwerte der einzelnen Pixel. Aus dem RGB-Farbtripel errechnet sich der Grauwert durch folgende Ganzzahloperation (vergleiche Abschnitt 2.4.1):

$$Y = \frac{54R + 183G + 19B}{256}$$

Da der Grünwert den mit Abstand größten Beitrag zum Ergebnis leistet, werden wir nur diesen Wert betrachten. Die dargestellte Berechnung fällt damit weg.

Als Hashwert nehmen wir die Summe über die Grünwerte der 64 Messpunkte. Unser Verfahren muss somit nur auf 64 Werten rechnen. Dies ermöglicht eine sehr schnelle Ausführung. Dies ist insofern wichtig, da der Hashwert in der Umstellphase der Belichtung für jedes ankommende Sample bestimmt werden muss, um einen frühen nächsten Aufnahmezeitpunkt zu

finden. Da wir Belichtungsreihen aufnehmen, müssen wir für jedes HDR mehrfach belichten und bei jeder Belichtung darauf warten, dass der Algorithmus ausgibt, dass das Bild nun stabil ist.

Der Hashwert wird absichtlich nicht normiert, um sowohl die Ausführungszeit[177] als auch die Genauigkeit beizubehalten[178]. Durch Verwendung der Summe liegt der Hashwert bei einem 24Bit-LDRI im Intervall $[0; 64 \cdot 2^8 = 2^{14} = 16320]$. Der Datentyp (unsigned) short (2 Byte) reicht zum Speichern des Hashs aus.

Durch die Reduzierung der Messpunkte im Vergleich zum vorher skizzierten histogrammbasierten Verfahren bekommen wir eventuell ein höheres Rauschen für unseren Hashwert. Zur Stabilisierung wurde daher im Verfahren noch eine Mittelwertbetrachtung über die Zeit eingebaut: Es wird der Mittelwert über den letzten und vorletzten Hashwert als Bezugsgröße verwendet.

Zusätzlich wird für eine Aufnahme noch gefordert, dass der Hashwert „*stableCount*“ viele eintreffende Samples lang als stabil angenommen wurde. Stabil ist der Wert, wenn er vom gerade eingeführten Mittelwert um weniger als *stableDelta* abweicht.

Das Verfahren ergibt sich in Pseudocode:

```
1  // stableDelta and stableCount have to be initialized for each
       shot.
2
3  // get information for decision
4  currentHash = getHash(Image);
5  midVal2 = previousHash + prePreviousHash >> 1; // ">> 1" is "/2"
6
7  // store old values
8  previousHash = currentHash;
9  prePreviousHash = previousHash;
10
11 currentDelta = abs(currentHash - midVal2);
12 isCurrentlyStable = (currentDelta <= stableDelta);
13
14
15
16
17
18 if (isCurrentlyStable) stableCount--;
19 if (stableCount < 1) *deliverSample*;
20 else *waitForNextSample*;
```

Listing 5.5: Pseudocode des Belichtungsänderungsdetektors BÄD.

Wir wollen möglichst schnell die verschiedenen Aufnahmen machen. Für eine geeignete Wahl der Werte ist im Falle des *stableCount* zu berücksichtigen, mit welcher Bildwiederholrate wir Samples bekommen. Diese liegt üblicherweise bei 25-30 Bildern pro Sekunde. Ein $stableCount = 30$ bedeutet also, dass die Belichtung ungefähr eine Sekunde stabil gewesen sein muss, bis eine Aufnahme gemacht wird. Die Hälfte reicht zumeist auch aus, da diese Zeit nur die Zeitspanne ist, bevor die Umstellung beginnt und danach. Während der Umstellung wird oftmals $currentDelta > stableDelta$ gelten und der *stableCount* daher nicht heruntergezählt.

Für die Wahl des *stableDelta* ist wichtig, zu berücksichtigen, dass wir 64 Messpunkte haben. Wenn alle diese Messpunkte sich um Eins in dieselbe Richtung ändern[179], entspricht dies

[177] Mit Ganzzahlen rechnen Rechner schneller als mit Gleitkommazahlen.
[178] Eine Normierung auf 255 Werte beispielsweise würde die Genauigkeit stark verringern.
[179] Wenn die Hälfte um Eins größer wird und die andere Hälfte um Eins kleiner, ändert sich am Hashwert nichts.

einem *currentDelta* von 64. Durch die Verwendung des Mittelwertes ist das Verfahren sehr stabil. Ein Wert kleiner als 64 kann benutzt werden. Dies hängt allerdings von der abgebildeten Szene ab. Wenn sich beispielsweise ein Blinklicht an der Stelle eines Messpunktes befindet, wird sich dessen Helligkeit periodisch stark ändern.

Bei jeder Belichtungsänderung müssen wir mindestens *stableCount* warten, bis die nächste Aufnahme gemacht werden kann. Wie oben beschrieben benötigen wir den Wert vor allem, um nicht bei einer besonders langsamen Umschaltung (bei wenig Licht erfolgt die Umschaltung langsamer als bei viel Licht) fälschlicher Weise eine Aufnahme zu machen, bevor der Umschaltvorgang begonnen hat.

Wir können den *stableCount* deutlich verkleinern, wenn wir dem Algorithmus mitteilen, dass wir eine Umschaltung vorgenommen haben und daher eine Hashwertänderung erwarten:

```
1       // stableDelta and stableCount have to be initialized for each
            shot.
2
3       // get information for decision
4       currentHash = getHash(Image);
5       midVal2 = previousHash + prePreviousHash >> 1; // ">> 1" is
            "/2"
6
7       // store old values
8       previousHash = currentHash;
9       prePreviousHash = previousHash;
10
11      currentDelta = abs(currentHash - midVal2);
12      isCurrentlyStable = (currentDelta <= stableDelta);
13
14  // new change criterion
15  if (isWaiting4change) isWaiting4change = isCurrentlyStable;
16  if (isWaiting4change) *waitForNextSample*;
17
18      if (isCurrentlyStable) stableCount--;
19      if (stableCount < 1) *deliverSample*;
20      else *waitForNextSample*;
```

Listing 5.6: Pseudocodeerweiterung des Belichtungsänderungsdetektors BÄD.

Die boolsche Variable *isWaiting4change* wird in diesem Fall gesetzt. Der Algorithmus selbst kann sie nur zurücksetzen.

Er tut dies, wenn der instabile Fall eintritt, also die Helligkeit sich an den Messpunkten signifikant ändert.

Das Verfahren funktioniert über die Koppelung an das *stableDelta*. Allerdings sind die Helligkeitsänderungen durch eine Belichtungsänderung abhängig von der Gesamtbeleuchtung (vergleiche Abbildung 5.19). Eine Anpassung des *stableDelta* oder die Einführung einer neuen Schwelle in Abhängigkeit der Gesamthelligkeit könnten das Verfahren daher noch zusätzlich verbessern.

Wie wir an der rechten Belichtungsreihe in Abbildung 5.19 sehen können, funktioniert das Verfahren sehr gut.

In der Abbildung sind zum Teil fast gleiche Abbildungen untereinander. Dies liegt vermutlich an der fehlerhaften Umschaltung der Kamera, auf die wir in Abschnitt 5.4.2 kurz eingehen werden.

5.4 Details der Implementierung

In diesem Abschnitt wird auf wenige ausgewählte Details, die sich in der Implementierungsphase ergeben haben und die interessant erscheinen, eingegangen.

5.4.1 Gerätetreiberprobleme

Ein sehr wichtiger Entscheidungsgrund für Windows als Wirtsbetriebssystem für unsere Anwendung waren in Abschnitt 5.2.2 die Verfügbarkeit von Gerätetreibern und die Abstraktionsschicht DirectShow.

Damit DirectShow zur Gerätesteuerung genutzt werden kann, müssen die darunter liegenden Gerätetreiber entsprechende genormte Schnittstellen anbieten. Dies ist in Abbildung 5.4 im linken unteren Bereich eingezeichnet.

Konkret muss der Treiberhersteller das Property-Set „PROPSETID_VIDCAP_CAMERACONTROL“ anbieten, welches Teil des T.RDC Standards der International Telecommunication Unit (ITU) ist. Erfüllt der Treiber den Standard, steht automatisch das von unserer Anwendung benötigte DirectShow-Interface *IAMCameraControl* zur Verfügung, mit dem sich die Belichtung steuern lässt.

Eine gute Übersicht über die softwareseitigen Ansteuermöglichkeiten eines Gerätes bietet sein Eigenschaftendialog. Zumeist sieht der Eigenschaftendialog ähnlich aus wie die in Abbildung 5.22 gezeigten.

Der Grund für die Ähnlichkeit der Dialoge ist, dass diese Art Eigenschaftendialog automatisch via COM erzeugt werden kann: Zur Darstellung wird die *OleCreatePropertyFrame*-Funktion aufgerufen. Diese fragt direkt die öffentlichen Schnittstellen des Treiber-COM-Objektes ab und offenbart so alle Schnittstellen. ((D)COM haben wir in Abschnitt 5.2.5 betrachtet.)

In Abbildung 5.22 sehen wir eine Seite des Eigenschaftendialogs zweier Webcams. Die Dialoge sind recht verschiedenartig aufgebaut. Dies lässt bereits auf unterschiedliche Anordnung der Softwareschnittstellen schließen.

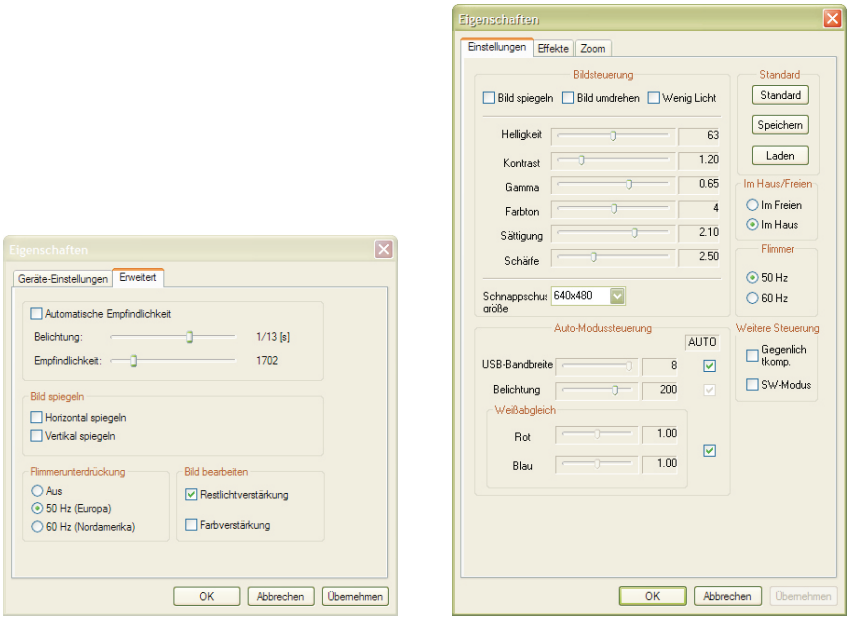


Abbildung 5.22: Die Eigenschaftendialogseite mit der Belichtungseinstellung einer Philips ToUCam Pro PCV 740K (links) und einer Labtec WebCam Pro (rechts).

In der Tat ist es so, dass nicht beide Kameras das standardkonforme Interface anbieten: Obwohl wir über den Eigenschaftendialog die Belichtung beider Kameras steuern können, ist die linke Kamera über DirectShow nicht ansprechbar.

Ausgiebige Recherche im Internet[180] hat ergeben, dass es keine Standardmechanismen zu geben scheint, Werte eines Eigenschaftendialogs zu ändern. Insbesondere sind Eigenschaftendialoge der in Abbildung 5.22 dargestellten Art laut Microsoft nur zu Debugzwecken und keinesfalls zur Anzeige für den Benutzer gedacht. Dies mag den Umstand der fehlenden standardisierten Steuerbarkeit erklären.

Eine Möglichkeit, die Werte des Dialogs zu setzen, besteht darin, sich die Adresse des Steuerelements zu ermitteln, den Dialog anschließend zu verstecken und über Fensternachrichten fernzusteuern. Für die wenigsten Nutzer wird es möglich sein, die Fensteradresse des Belichtungsschiebereglers zu ermitteln[181]. Eine automatische Ermittlung der Adresse ist nicht möglich, da die Eigenschaften gerade nicht standardkonform benannt sind. Wenn sie dies wären, stünden sie in DirectShow zur Verfügung.

Der Weg über die Fernsteuerung des Eigenschaftendialogs scheint im Fall solcher nicht standardkonformen Treiber der einzige etwas abstrakte Belichtungssteuerungsmechanismus zu sein.

Alternativ bleibt der Weg, direkt den Treiber oder gar die Hardware anzusprechen. Dieser Weg ist zwar möglich, erfordert allerdings für fast jede Hardware neuen Programmcode und widerspricht damit unserem Ziel, eine universell einsetzbare Software zu entwickeln.

Ein Mittelweg wäre es, um die Treiber solcher Kameras einen Proxytreiber zu programmieren. Dieser müsste zwischen den standardkonformen Aufrufen von außen in die proprietären Aufrufe nach innen zum Treiber übersetzen. Übersetzung ist hierbei wörtlich zu verstehen, da es oftmals an der „falschen“ Bennenung der Eigenschaften scheitert.

Wir mussten so leider feststellen, dass einige Webcamtreiber eben gerade nicht die erwünschte Abstraktion erlauben und damit für unsere Software nicht benutzbar sind.

Dieses Problem führte dazu, dass die von uns ursprünglich vorgesehene Hardware von Philipps nicht genutzt werden kann und wir stattdessen auf Logitech-Kameras umsteigen mussten.

5.4.2 Probleme bei der Einstellung der Belichtungszeit

Das im vorigen Abschnitt geschilderte Problem setzt sich leider fort. Wenn man verschiedene Kameras betrachtet, die die DirectShow-Schnittstelle anbieten, also soweit den ITU-Standard erfüllen, so stellt man fest, dass die Werte, die diese Kameras zum Setzen und Auslesen der Belichtung verwenden, leider genauso wenig einem Standard folgen, wie die Schnittstellen im vorherigen Absatz.

Betrachten wir die Belichtungszeiten, ihre Einstellgrößen und die Rückgabewerte der entsprechenden DirectShow-Funktionen in Tabelle 5.2.

Wie wir sehen können, ist das Resultat selbst bei einem einzigen Hersteller sehr unterschiedlich.

Aus der Dokumentation [*Microsoft Corporation*, 2008] ergibt sich, dass die Werte, die eingestellt werden, die Bedeutung 2^{Wert} haben sollen. Bei der rechten Kamera stimmt dies ungefähr.

[180] Die Recherche beinhaltet mehrere Anfragen im Quickcam-Entwicklerforum, an dem Mitarbeiter aus der Treiberabteilung von Logitech beteiligt sind.

[181] Zur Ermittlung der Adresse eines Windows-Elements werden spezielle Tools aus der Windows-Programmierwelt benötigt, deren Bedienung zumeist Programmierern vorenthalten ist.

Logitech Quickcam 9000 Pro (Treiber 11.5)			**Logitech Quickcam Pro**		
set	get	exposure	set	get	exposure
−1	−1	$\frac{1}{500}$	−1	−1	$\frac{1}{3}$
−2	−2	$\frac{1}{250}$	−2	−2	$\frac{1}{4}$
−3	−3	$\frac{1}{125}$	−3	−3	$\frac{1}{8}$
−4	−4	$\frac{1}{63}$	−4	−4	$\frac{1}{16}$
−5	−6	$\frac{1}{31}$	−5	−5	$\frac{1}{32}$
−6	−7	$\frac{1}{16}$	−6	−6	$\frac{1}{66}$
−7	−8	$\frac{1}{8}$	−7	−7	$\frac{1}{142}$
−8	−9	$\frac{1}{5}$	−8	−8	$\frac{1}{333}$
−9	−10	$\frac{1}{5}$	−9	−9	$\frac{1}{1000}$
−10	−11	$\frac{1}{5}$			
−11	−12	$\frac{1}{5}$			
−12	−13	$\frac{1}{5}$			
−13	−14	$\frac{1}{5}$			
−14	−8	$\frac{1}{5}$			

Tabelle 5.2: Die Tabelle zeigt, welche Werte eingestellt werden („get“, „exposure“), wenn man der Kamera den Befehl gibt, den in der „set“-Spalte eingetragenen Wert einzustellen.

Der Treiber auf der linken Seite scheint noch nicht zuende programmiert zu sein[182]. Dies könnte auch der Grund sein, warum ab einer Einstellung von -5 jeweils sofort ein Wert eins darunter als Rückgabe geliefert wird und die Belichtung $\frac{1}{1000}$ sich nicht ansteuern lässt[183].

Es gibt leider keine Möglichkeit, festzustellen, welche Belichtung eingestellt ist. Die hier eingetragenen Werte stammen aus dem Eigenschaftendialog der jeweiligen Kamera (vergleiche Abbildung 5.22) und wurden von Hand abgelesen. Dies ist ein gravierendes Problem, da wir für die HDR-Berechnung die Belichtungszeit und die Blende benötigen.

Als Workaround müssen diese nun von Hand in einer Textdatei eingetragen werden, die der jeweiligen Kamera zugeordnet wird. Aus dieser Datei werden dann die Zuordnungen „Rückgabewert $\mapsto$ Belichtungszeit“ gemäß Tabelle 5.2 für die EXIFF-Information vorgenommen (vergleiche Abschnitt 5.2.7.3).

Bei Tabelle 5.2 fällt weiterhin auf, dass beide Kameramodelle eine unterschiedliche Anzahl an Einstellstufen bereitstellen. Aus dem Eigenschaftendialog lässt sich ablesen, dass die Quickcam 9000 Pro ein Einstellungsintervall von $[\frac{1}{5}; \frac{1}{1000}]$, die Quickcam Pro eines von $[\frac{1}{3}; \frac{1}{1000}]$ abdeckt.

Unter der Annahme, dass die Logitech-Entwickler das Treiberproblem lösen und mit Absicht 14 Einstellstufen anbieten, hieße dies, dass die Kamera mit dem größeren Belichtungsumfang weniger Stufen anbietet. Daraus folgt die unschöne Eigenart, dass DirectShow ein recht grobgranulares Einstellraster vorsieht.

Die Kameras selbst sind (über den Eigenschaftendialog) jedoch in der Lage, die Belichtungszeit viel differenzierter einzustellen.

5.4.3 Die anderen Parameter

Wie an den Eigenschaftenseiten in Abbildung 5.22 zu erkennen ist, gibt es eine ganze Reihe an Parametern, die wir an der Webcam einstellen können. Wie wir in Kapitel 2 gesehen haben, sind diese für die Aufnahme der LDRIs zur späteren HDR-Berechnung relevant.

Tests haben ergeben, dass auch ohne die Kontrolle aller Parameter gute Bildergebnisse erzielt werden können.

[182] Es ist die erste Treiberrevision, die die Belichtungseinstellung erlaubt. Die Unterstützung wurde nach Anfragen im QuickCam-Developper-Forum eingebaut.

[183] Dies ist ein schwerwiegendes Problem, weil diese kürzeste Belichtungsstufe benötigt wird, um beispielsweise Wolken an einem hellen Himmel fotografisch einzufangen.

Die Kontrolle eines Parameters ist allerdings wichtig. Dieser Parameter ist die **Gainspannung**.

Mithilfe der Gainspannung wird das Signal des Aufnahmechips derart angehoben, dass es verarbeitet werden kann. Liegt die Spannung bei gleich bleibender Belichtung zu niedrig, ist unser Bild zu dunkel. Liegt die Spannung zu hoch, so ist das Bild eventuell zu hell. In jedem Fall entsteht allerdings störendes Rauschen. Die Gainspannung muss also bei unserer Anwendung gesteuert werden.

Zu diesem Zweck haben fast alle Kameras eingebaute Regelungsmechanismen. Diese lassen sich bei unseren getesteten Modellen allerdings scheinbar nicht einschalten, wenn die automatische Belichtungssteuerung ausgeschaltet ist.

Um nicht selbst Software zur Regelung der Gainspannung schreiben zu müssen, was auf dem Abstraktionslevel, auf dem sich unsere Anwendung befindet ungewöhnlich wäre, benutzen wir trotz der beschriebenen Probleme die automatische Gainspannungssteuerung (automatic gain control, AGC): Vor der ersten Aufnahme der Belichtungsreihe stellen wir die Belichtung auf Automatik und warten mittels des in Abschnitt 5.3.4.1 entwickelten Algorithmus, bis sie sich eingestellt hat. Die Gainspannung steht nun auf einem optimalen Level für die automatische Belichtung. Wenn wir jetzt ledigleich die Belichtungssteuerung ausschalten, bleibt die Gainspannung auf dem eingestellten Wert.

Auf diese Weise liefert uns die Hardware trotz der beschriebenen Schwierigkeiten das gewünschte Ergebnis.

5.4.4 Der Samplegrabber-Callback

Der Samplegrabber-Callback ist für den nicht COM-erfahrenen Programmierer wahrscheinlich die interessanteste Komponente. Daher betrachten wir sie ein wenig näher.

Sobald ein Mediensample vorliegt, ruft die Samplegrabber-Komponente (vergleiche Abbildung 5.5) eine zuvor registrierte Funktion, einen so genannten „Callback“, auf.

Der Callback muss als COM-Komponente realisiert sein. Es reicht allerdings, lediglich die Grundfunktionen zu implementieren bzw. deren Verhalten (vergleiche Abschnitt 5.2.5.1). In Listing 5.7 sind wichtige Komponenten der Schnittstelle abgedruckt.

```
1  class CSampleGrabberCB:public ISampleGrabberCB
2  {
3  public:
4    CSampleGrabberCB(ISampleGrabber* myGrabber);
5    ~CSampleGrabberCB(void);
6
7    // COM-Functionality for ISampleGrabberCB-COM-Object
8    STDMETHODIMP QueryInterface(REFIID riid, void ** ppv);
9    STDMETHODIMP_(ULONG) AddRef();
10   STDMETHODIMP_(ULONG) Release();
11   STDMETHODIMP SampleCB(double dblSampleTime, IMediaSample* pSample)
         ;
12   STDMETHODIMP BufferCB(double dblSampleTime, BYTE* pBuffer, long
         lBufferSize);
13 private:
14   bool m_bDoShot; // true -> 1 Sample => false
15 };
```

Listing 5.7: Ein Auszug aus CSampleGrabberCB.h.

Es stehen zwei Funktionen zur Implementierung bereit, die *SampleCB*- und die *BufferCB*-Funktion.

Der SampleCB ermöglicht direkten Zugiff auf das eigentliche Mediensample (und weitere Daten). Die BufferCB-Funktion kopiert hingegen das Sample und stellt einen Zeiger auf die Kopie bereit.

Da wir die Daten für die Nutzung ohnehin kopieren, verwenden wir die SampleCB-Funktion. Dadurch wird Verarbeitungszeit gespart, da nicht –wie bei BufferCB– jedes Sample des Streams kopiert wird sondern nur diejenigen, die wir verarbeiten.

Außerdem haben wir zu Debugzwecken die Möglichkeit, Bildinformation zu verändern, die dann auch automatisch angezeigt wird, da wir die Originaldaten verändert haben. In Abbildung 5.20 wurde beispielsweise davon Gebrauch gemacht.

Unsere Callbackfunktion besitzt eine boolsche Variable (m_bDoShot), die anzeigt, ob das nächste Sample weiterverarbeitet werden soll. Ist diese auf false gesetzt, so kehrt der Callback zurück, ohne den Buffer zu kopieren.

Wenn wir ein Bild aufnehmen wollen, setzen wir diese Variable auf true. Beim nächsten Aufruf des Callbacks wird dann der Buffer kopiert und der Callback setzt die Variable automatisch wieder auf false, damit unsere Daten nicht überschrieben werden.

Weiterhin sendet er die „WM_SAMPLE_WAITING“-Nachricht mit der aktuellen Kameranummer im *wParam* ab. Die weitere Verarbeitung erfolgt wie in Abschnitt 5.3 beschrieben.

Da wir unsere Anwendung multithreaded entwickelt haben, kann der Callback durch die asynchrone nachrichtengesteuerte Weiterverarbeitung der Daten schnell zurückkehren. Der Stream wird dadurch nicht lange aufgehalten[184].

Ein weiterer –historischer– Grund für die asynchrone Verarbeitung ist derjenige, dass das Win16-Lock bei 16-Bit-Anwendungen vom Videograbber gehalten werden kann und damit weitere 16-Bit-Aufrufe blockieren könnten. Rufen wir im Callback auf einem solchen System andere Funktionen auf, die versuchen, das Lock zu aquirieren, so würde eine Deadlocksituation entstehen und das System blockieren.

Auf heutigen 32- oder 64-Bitsystemen besteht diese Einschränkung bezüglich der aufrufbaren Funktionen nicht.

[184] Da wir den Stream nicht weiterverarbeiten, wäre es für unsere Anwendung egal, ob die Bildrate sinkt.

5.5 Fremdsoftware

Wir werden an dieser Stelle nur sehr kurz auf die Fremdsoftware zur weiteren Verarbeitung der LDR-Belichtungsreihen eingehen.

Die Tools sind im Rahmen dieser Arbeit nicht ausgiebig getestet worden, haben sich aber vielfach in der Praxis bewährt.

Für die HDR-Gewinnung gibt es zahlreiche Implementierungen im Internet. Zum Teil haben auch die Forscher selbst ihre in Kapitel2 vorgestellten Algorithmen veröffentlicht. Wir haben für unseren Zweck die PFSTools gewählt.

Zum Panoramastitching stellen die Panoramatools eine gute frei verfügbare Software dar. Deshalb werden wir diese verwenden.

Als SIFT-Implementierung zum Finden von Punktkorrespondenzen verwenden wir Autopano-SIFT.

5.5.1 PFS-Tools

Die PFS-Tools zur Verarbeitung von „Portable Floating-point Streams“ sind eine, am Max-Planck-Institut Saarbrücken um das Jahr 2004 entwickelte, Sammlung von Programmen zum Verarbeiten von HDR-Daten. Die Tools liegen in einer Kommandozeilenversion vor, die unter Linux nativ und unter Windows mithilfe von Cygwin gestartet werden kann. Da der Sourcecode vorliegt, lassen sie sich in eine Bibliothek umwandeln und dann direkt von einem Programm aus unter Windows aufrufen (Vergleiche Abschnitt 5.2.4).

Eine ausführliche Motivation für die PFS-Tools findet sich in [*Mantiuk et al.*, 2007].

In der Programmsammlung sind verschiedene, der von uns in Abschnitt 2.3.3 betrachteten, Algorithmen implementiert:

- *pfshdrcalibrate* verwendet eine Implementierung des Algorithmus [*Robertson et al.*, 2003]. Die Wahl fiel auf diesen Algorithmus, da die Autoren ursprünglich HDR-Sensoren kalibrieren wollten[185] und deren Responsekurve nicht glatt sein muss und sich daher auch nicht durch die in Abschnitt 2.3.3 vorgestellten Modelle annähern lässt [*Krawczyk et al.*, 2005].
  Das Verfahren von *Robertson et al.* [2003] eignet sich, da es keine Bedingungen (Glattheit) an die Kurvenform stellt[186].
  Außerdem benötigt es zusätzlich zu den gespeicherten Kalibrierungsaufnahmen keinen Speicherplatz, was bei den HDR-Daten, die deutlich umfangreicher als die LDR-Daten sind, von Bedeutung ist.
- *hdrgen* basiert auf *Debevec und Malik* [1997]. Diesen Algorithmus haben wir in Abschnitt 2.3.3 aufführlich besprochen.

Mit den Tools ist es möglich, die Kameraresponsekurve der Webcams zu bestimmen. Diese kann dann für die Verarbeitung weiterer Bilder verwendet werden und muss nicht jedes Mal erneut bestimmt werden, genau so, wie wir es vorgesehen haben.

[185] Die Autoren kommen zu dem Schluss, dass die HDR-Kameras noch nicht in der Lage sind, konstant so gut kalibrierte Daten zu liefern, wie eine professionelle LDR-Kamera durch mehrere Aufnahmen.
[186] Die kalibrierten HDR-Kameras liefern unstetige Responsekurven.

Die PSF-Tools werden von uns nicht nur zur HDR-Berechnung sondern auch für das Tonemapping eingesetzt.

Sie beinhalten Implementierungen zu den folgenden Tonemappingoperatoren, von denen wir einige schon aus Kapitel 4 kennen:

- *Drago et al.* [2003b]
- *Pattanaik et al.* [2000]
- *Reinhard und Devlin* [2005a]
- *Reinhard et al.* [2002a]
- *Durand und Dorsey* [2002]
- *Ashikhmin* [2002]
- *Fattal et al.* [2002]
- *Mantiuk et al.* [2006]

Die PFS-Tools sind auf der Website http://www.mpi-inf.mpg.de/resources/pfstools/ zu finden.

5.5.2 Panoramatools

Die Panoramatools sind eine Sammlung von Tools, die zur Erstellung von Panoramen benötigt werden.

Die ursprüngliche Version wurde 1998 von Helmut Dersch entwickelt. Mittlerweile werden die Tools von einer großen Entwicklergemeinschaft weiterentwickelt und dabei unter anderem auch HDR-fähig gemacht.

Die Panoramatools bieten alle von uns in Kapitel 3 betrachteten Möglichkeiten und gehen sogar noch darüber hinaus.

Die Panoramatools sind auf der Website http://www.panotools.org zu finden.

5.5.3 Autopano-SIFT

Die Panoramatools bieten selbst keinen automatischen Mechanismus, um Bildkorrespondenzen zu bestimmen. Für unsere Anwendung ist dies nicht kritisch, da wir das Stitching ebenso wie die Kameraresponse nur einmal bestimmen müssen und diese Aufgabe daher auch von Hand erledigen können.

Eine gute Möglichkeit, das Korrespondenzproblem automatisch zu lösen, haben wir mit SIFT in Abschnitt 3.4.1 betrachtet. Eine, mit den Panoramatools kompatible, Implementierung des Algorithmus ist das Programm Autopano-SIFT.

Wir können das Programm dazu nutzen, ein initiales Stitching vorzuschlagen und dieses gegebenenfalls anschließend von Hand nachjustieren. Dadurch ist unsere gesamte Anwendung sehr schnell konfigurierbar und einsatzfähig.

Autopano-SIFT ist auf der Website http://user.cs.tu-berlin.de/~nowozin/autopano-sift/ zu finden.

5.6 Zusammenfassung

Nach den drei großen theoretischen Kapiteln haben wir uns in diesem Teil mit der praktischen Entwicklung der Software beschäftigt, die HDR Panoramen bis hin zu 360° sowie verschieden gemappte LDR-Panoramen erstellen kann.

Da die Funktionalität für alle Verarbeitungsschritte außer der Aufnahme bereits vorhanden ist, haben wir uns hauptsächlich mit der Aufnahmeproblematik befasst.

Wir haben einleitend die Wahl von Windows als Plattform motiviert und uns im weiteren Verlauf des Kapitels mit dieser Plattform auseinandergesetzt.

Wie in den anderen Kapiteln haben wir verschiedene Grundlagen betrachtet. Dabei sind wir auf diverse Konzepte wie Nachrichten, Bibliotheken, (D)COM und Windows-Driver-Model eingegangen.

Außerdem haben wir uns mit DirectShow, FilterGraphen und verschiedenen Bildformaten befasst.

Mit diesem Rüstzeug ausgestattet, haben wir die eigentliche Anwendung mithilfe von UML- und SDL-Diagrammen spezifiziert.

Wir haben dabei einen Algorithmus zur Belichtungsänderungsdetektion entwickelt, der es uns ermöglicht, festzustellen, wann eine Kamera sich mit großer Wahrscheinlichkeit auf eine neue Belichtung eingestellt hat. Diese zentrale Funktionalität musste erst noch entwickelt werden, da die Kameras keine Möglichkeit vorsehen, einer Anwendung mitzuteilen, dass eine neue Belichtung eingestellt ist.

Bezüglich der konkreten Implementierung haben wir Probleme im Zusammenhang mit den Treibern und der Belichtungssteuerung sowie die mit COM kommunizierenden Samplegrabber-Callbackfunktion betrachtet.

Zuletzt sind wir kurz auf die Fremdsoftware zur weiteren Verarbeitung der Daten eingegangen.

–leere Seite–

6. Zusammenfassung und Fazit

> Denn alles Streben entspringt aus Mangel, aus Unzufriedenheit mit seinem Zustand, ist also Leiden, solange es nicht befriedigt ist; keine Befriedigung aber ist dauernd, vielmehr ist sie nur der Anfangspunkt eines neuen Strebens.
>
> Arthur Schopenhauer, Die Welt als Wille und Vorstellung

Auf den vorangegangenen Seiten haben wir unzählige Aspekte bei der Erstellung eines 360°-HDR-Panoramas und dessen tonegemappten Pendant betrachtet.

Wir haben dabei den Spagat zwischen wissenschaftlichem Anspruch und verständlicher Darstellung gewagt. Ein Ziel der Arbeit war es schließlich, dem Leser ein Grundverständnis für die dargelegten Sachverhalte und Verfahren mitzugeben. Gleichzeitig sollte vor allem im Kapitel zum High Dynamic Range Imaging ein fundierter Überblick über die Algorithmen vermittelt werden, was an vielen Stellen die höhere Mathematik unabdingbar machte.

6.1 Zusammenfassung

Wir haben unsere Reise in der Welt begonnen und sie über die Linse und den Aufnahmechip in den Rechner fortgesetzt. Dort haben wir uns sehr ausführlich mit verschiedenen Techniken zur Überwindung der Beschränkungen, die sich durch den Low Dynamic Range-Aufnahmechip ergeben, befasst. In diesem Kapitel lag der Fokus sowohl auf den Details der einzelnen Algorithmen als auch darauf, einen Überblick und ein Grundverständnis für die Materie zu schaffen. In diesem Teil haben wir uns am ausführlichsten mit der hinter den Verfahren stehenden Mathematik befasst und einige Sachverhalte mathematisch bewiesen.

Der weitere Weg der Bildinformation führte uns zum Stitching. Wir haben uns Schritt für Schritt an die uns erwartenden Bedingungen beim Zusammenfügen der Einzelbilder angenähert und dabei festgestellt, dass unsere gewünschten Voraussetzungen einige Probleme mit sich bringen. In diesem Teil lag unser Augenmerk weniger auf konkreten Algorithmen als

mehr darauf, die sich ergebenden Schwierigkeiten zu erkennen und zu lösen. Auch in diesem Kapitel haben wir zahlreiche Konzepte der Computergrafik kennen gelernt.

An dieser Stelle waren wir aus theoretischer Sicht schon fast am Ziel angelangt: Ein ausreichendes Maß an Theorie zur Erstellung von HDR-Panoramen war abgehandelt. Es blieb noch der Schritt zum Low Dynamic Range Image.

Bereits im hinführenden Teil zum Bewältigen dieser Aufgabe haben wir gesehen, dass das Tonemapping eine phantastische Möglichkeit bietet, Kunstwerke zu schaffen. Wir haben die Einleitung weiterhin dazu genutzt, uns eingehender mit unserer visuellen Wahrnehmung zu befassen und dabei wieder einmal festgestellt, dass Menschen sehr komplexe Signalverarbeiter sind.

Um das erlangte Wissen in praktischer Umsetzung zu erleben, haben wir uns im Vorangegangenen mit der Architektur und der Implementierung der spy360-Software auseinander gesetzt. Dabei haben wir einige Konzepte der Windows-Welt gestreift und einen Algorithmus entwickelt, der es uns erlaubt, Belichtungsänderungen geeigent zu detektieren.

6.2 Fazit

Am Ende der Arbeit lässt sich sagen, dass die eingangs genannten Ziele in den theoretischen Kapiteln zur vollsten Zufriedenheit des Autors erreicht wurden, der sich selbst beim Erstellen der Arbeit an sehr vielen Stellen stark weiterbilden konnte.

Fast alle dargestellten Inhalte konnten so dargelegt werden, dass sie auch mit wenig Fachwissen aus dem jeweiligen Themenkomplex nachvollzogen werden können.

Zusätzlich wurden, auch für den im Thema beheimateten Leser, Querbezüge zu anderen Wissenschaften gezogen. Diese Querbezüge haben gezeigt, wie sehr es sich lohnt, über den Tellerrand des eigenen Fachgebietes hinauszuschauen. Gerade die Informatik als Wissenschaft, die mit vielen anderen Disziplinen zusammenarbeitet, bietet sich dafür an.

Dass sich das Hinausschauen lohnt, hat sich an einigen Stellen konkret darin geäußert, dass dadurch Verfahren zum Lösen von Problemen entstanden sind. Als vielversprechende Quelle für Information stellten sich die Biologie und die Natur heraus.

Nicht ganz zufrieden ist der Autor mit der entstandenen Software. Weniger wegen der Software selbst als mehr aufgrund der Feststellung, dass die erhofften Vorteile, die sich durch die Abstraktionskonstrukte unter Windows ergeben sollten, nicht eingetreten sind.

Es hat sich vielmehr herausgestellt, dass im Falle einer (wahrscheinlich erfolgenden) Neuentwicklung Linux als Plattform vorzuziehen ist. Die beschriebenen Nachteile von Linux haben sich als ebenso existent unter Windows herausgestellt und vor allem der beabsichtigte Einsatz der Software als Serveranwendung legt Linux nahe.

Das Konzept, nicht die komplette Software „from scratch“ neu zu entwickeln, ist dagegen voll aufgegangen. Durch den Einsatz der externen Tools steht erprobte Software zur Verfügung, die ständig von einer großen Programmierergemeinschaft weiterentwickelt wird.

Die entstandene Anwendung ist trotz der Unwägbarkeiten nicht vollkommen unnütz, da sie für die Zielgruppe des „Ottonormalanwenders“ geeignet bleibt.

Neben der Portierung auf Linux fehlen in dieser Arbeit noch ein paar Dinge, die hoffentlich Einzug in eine erweiterte Version finden. Dazu gehören diverse Bildbeispiele und andere Fotos von Testaufbauten, die aufgrund meines Umzuges nach München leider nicht mehr in dieser Arbeit berücksichtigt werden konnten.

Nicht zuletzt fehlt auch die Dokumentation der Montage der Arbeit im Turm des Wilhelm-Schickard-Instituts, die hoffentlich bald erfolgen wird und von da an hoffentlich tolle Bilder von Tübingen in die ganze Welt verbreitet...

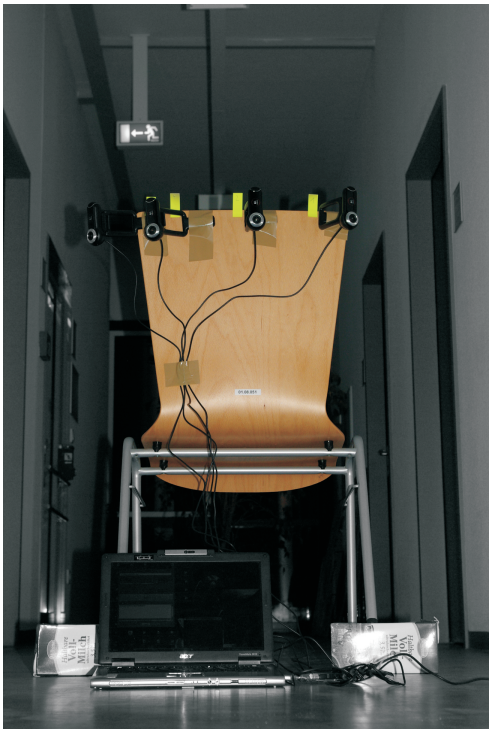


Abbildung 6.1: Das Bild zeigt den, an einem Stuhl befestigten, Testaufbau mit vier Kameras und einem Laptop. Die Kameras sind hochkant montiert, damit eine größere vertikale Auflösung (1600 Pixel) erreicht wird.

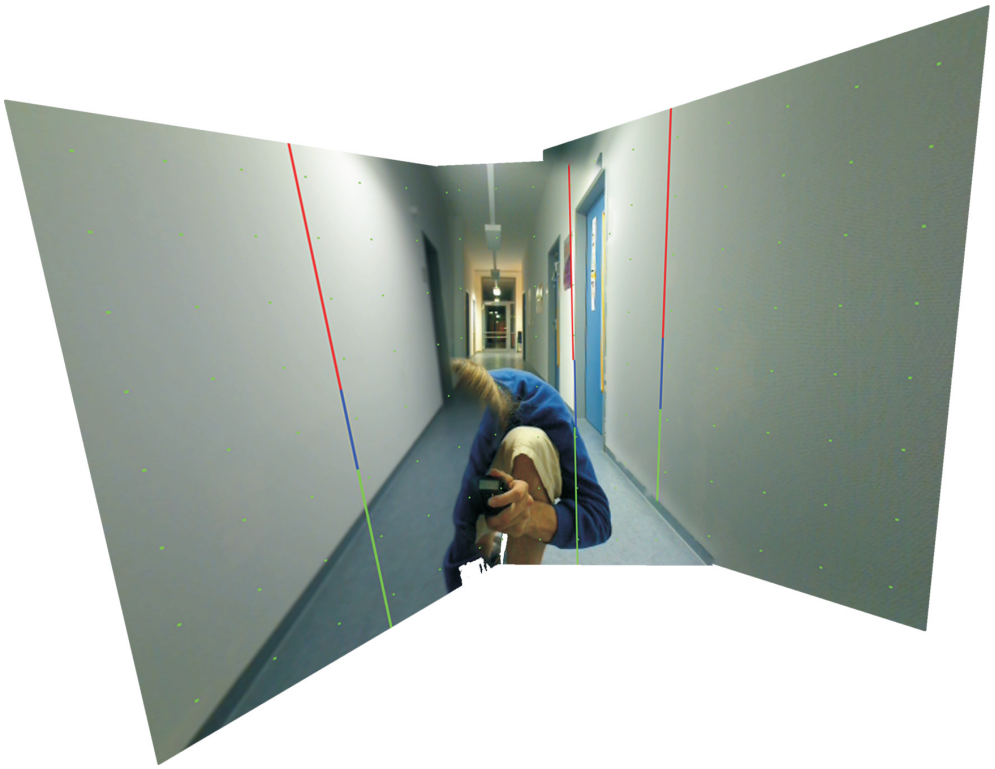


Abbildung 6.2: Die Flurszene wurde mit den Anordnungen auf der vorigen Seite aufgenommen. Wir sehen „Geister“ (vergleiche Abschnitt 2.4.2). Die bunten Striche sind der nicht abgeschaltete Debugoutput (vergleiche Abbildung 5.20).

Abbildung 6.3: Die Abbildung zeigt eine Außenszene an der TU-München. Leider stehen in der Tat noch keine Tagaufnahmen zur Verfügung.

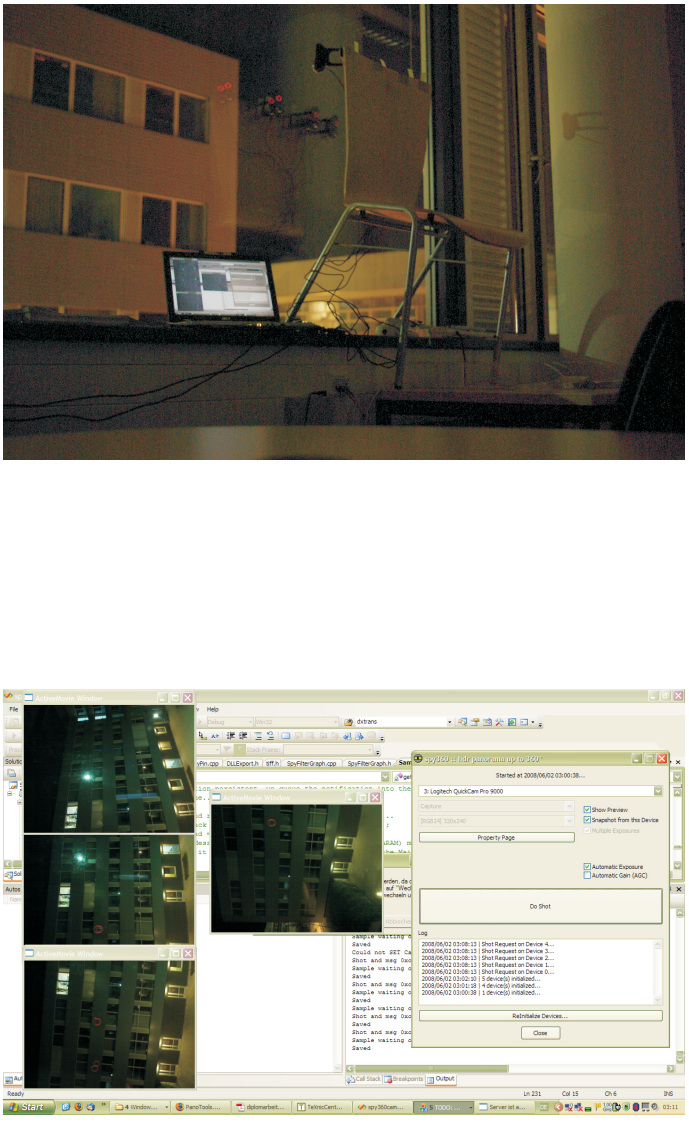


Abbildung 6.4: Das obere Bild zeigt die Anordnung bei der Aufnahme der Außenszene, das untere Bild ist ein Screenshot während der Aufnahme. Links sind die Livevorschaubilder zu sehen.

7. Ausblick

> Wenn Du ein Schiff bauen willst, so trommle nicht Männer zusammen, um Holz zu beschaffen, Werkzeuge vorzubereiten, die Arbeit einzuteilen und Aufgaben zu vergeben, sondern lehre die Männer die Sehnsucht nach dem endlosen weiten Meer!
>
> Antoine de Saint-Exupery, Der kleine Prinz

An die vorliegende Arbeit lässt sich auf vielerlei Weise anknüpfen. In diesem Kapitel werden einige Ideen hiezu vorgestellt.

7.1 HDRI

7.1.1 Vergleich verschiedener Algorithmen aus Kapitel 1

Wir haben in dieser Arbeit zwar die verschiedenen Algorithmen zur Gewinnung der Responsekurve vorgestellt. Ausführlich getestet und verglichen haben wir sie jedoch nicht.

Vor allem das am Ende präsentierte Verfahren, das über die Verteilung des Bildrauschens Rückschlüsse auf die Responsekurve zieht, könnte im Zusammenhang mit Webcams interessant sein. Durch die geringe Sensorgröße neigen Webcams eher zu Rauschen als Kameras mit größerem Sensor.

7.1.2 Ausprobieren von Short-Time-Stream-Sampling

Bei der Vorstellung der Hardwareverfahren in Abschnitt 2.2.3 haben wir solche kennen gelernt, die über die Modifikation der Belichtungszeiten der einzelnen Pixel funktionieren [*Acosta-Serafini et al.*, 2004]: Die Pixel werden dabei unterschiedlich lange belichtet, je nachdem, ob sie viel oder wenige Photonen aufnehmen.

Man könnte versuchen, solche Verfahren in Software zu simulieren, indem man eine sehr kurze Belichtungszeit einstellt und dann die Messwerte in Software aufaddiert. Der Ansatz

ist insbesondere für Webcams allerdings nicht unbedingt erfolgversprechend, da sich gerade bei kurzen Belichtungen Rauschen sehr stark bemerkbar macht (vergleiche Abschnitt 2.2.2.1) und wir dieses ebenfalls mit summieren würden.

7.2 Software

7.2.1 HDRI

7.2.1.1 Raw-Bayer-Mode

Wie wir in Kapitel 2 gesehen haben, ist es für die HDR-Gewinnung störend, dass die Pixelwerte des Sensorelements nicht linear abgebildet werden (vergleiche Abschnitt 2.3).

Bei einigen Kameras besteht die Möglichkeit, direkt die Daten des Chips auszulesen. Dieses Verfahren wird als „Raw-Bayer-Mode“ bezeichnet, da sich auf dem Sensor ein so genanntes Bayer-Pattern befindet. Dieses besteht aus Farbfiltern für Rot, Grün und Blau. Die Sensorelemente selbst nehmen keine Farben sondern nur Intensitäten auf.

Im Raw-Bayer-Mode erhalten wir die Rohdaten des Sensorchips, auf denen auch das, durch das Bayer-Pattern entstandene, Bild noch nicht zu einem RGB-Bild interpoliert wurde.

Zusätzlich steht der volle Dynamikumfang des Chips zur Verfügung, der oftmals größer als die nach Verarbeitung ausgegebenen 8 Bit ist.

Auf diese Weise stehen uns *lineare* Sensordaten zu Verfügung. Dies könnte für die HDR-Gewinnung interessant sein.

Insbesondere würde es die folgende Idee beschleunigen.

7.2.1.2 HDR-Video

Falls wir es schaffen, schnell genug verschiedene Belichtungen einzustellen, so können wir HDR-Video erzeugen.

7.2.1.3 Virtuelle HDR-Kamera

Ist die Framerate nebensächlich, so könnten wir bereits jetzt eine virtuelle Kamera als COM-Objekt realisieren (vergleihe Abschnitt 5.2.5), die sich wie eine reale Kamera unter Windows verhält, und daher von allen Anwendungen transparent eingesetzt werden kann.

Diese virtuelle HDR-Kamera könnte eine interessante Anwendung sein.

7.2.2 Stitching

7.2.2.1 Hardwarebasierter Ansatz

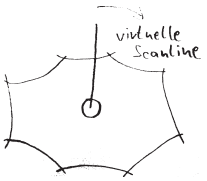


Abbildung 7.1: Skizze einer möglichen Realisierung mit Hardwareunterstützung.

Gerade in Bezug auf die Echtzeitfähigkeit ließe sich das Entfernen der Verzeichnung (siehe Abschnitt 3.2.2) und das Zusammenfügen zu einem Panorama in Hardware realisieren.

In der Skizze 7.1 ist dargestellt, dass die von den Kameras aufgenommenen Bilder als Textur auf, passend im Raum angeordnete, 3D-Flächen gelegt werden.

Formt man die Flächen entgegengesetzt der Verzeichnung, korrigiert man diese automatisch mit.

Alle rechenintensiven Mappingschritte beim Stitching können so von der Grafikhardware erledigt werden.

Insbesondere bei Verfügbarkeit von HDR-Kameras ist dies interessant, da dann auch der Schritt der Erzeugung des HDR entfällt und somit die Ausgabe der Kamera direkt in den Texturspeicher der Grafikkarte geschrieben werden kann. Wir erhalten so durch fast ausschließlichen Grafikhardwareeinsatz ein HDR-Panorama. Es ist zu erwarten, dass diese Verarbeitung deutlich schneller ablaufen wird als die Softwareverarbeitug auf der Haupt-CPU, da diese nicht für Grafikanwendungen optimiert ist.

Das endgültige Bild erhält man, z.B. indem man mit der eingezeichneten Scanline die Bildpunkte auf den Flächen samplet.

7.2.2.2 360°-LiveVideo

Ein ursprünglicher Ansatz von uns war die Bereitstellung von 360°-LiveVideo mittels der Kameras.

Die in dieser Arbeit vorgestellte Anwendung hat gezeigt, dass dies mit der verwendeten Hardware durchaus möglich ist. Insbesondere die zuvor vorgestellte Idee sollte dazu dienlich sein.

Auf die HDR-Fähigkeit muss dabei zu Gunsten einer höheren Bildwiederholrate momentan allerdings wohl verzichtet werden.

7.2.3 Allgemein

7.2.3.1 USB Video Class

Wie bereits in Kapitel 5 angesprochen, existieren mitlerweile Treiber nach dem USB Video Class-Standard (UVC). Der Standard existiert zwar schon seit 2005[187], die Treiber nach diesem Modell scheinen jedoch erst so langsam Verbreitung zu finden.

UVC-Treiber existieren auch für Linux. Mit dem Video4Linux2-Projekt steht ebenfalls ein abstrakter Zugriffsmechanismus bereit.

UVC bietet eine Schnittstelle, die Belichtungszeit in 1/ 10000 Sekunden-Schritten einzustellen.

[187] http://www.usb.org/developers/devclass_doc

7.2.3.2 Berechnung des HDR in spy360.dll

Die verwendete PSF-Tools-Bibliothek könnte dynamisch in die spy360cams-Bibliothek eingebunden werden. Dadurch müssten die Daten nicht auf der Festplatte zwischengespeichert werden. Der zu erwartende Geschwindigkeitsvorteil ist nicht allzu groß, da vor allem die Berechnungsschritte Zeit in Anspruch nehmen und nicht das Speichern.

7.2.3.3 Falloverbehandlung

Es könnte gewünscht sein, bei Ausfall einer Kamera weiterhin vollständige Panoramen zu liefern. Zu diesem Zweck könnte man alte Aufnahmen zwischenspeichern und im Falle eines Ausfalls automatisch passend einspielen.

7.2.3.4 Erhöhung der Aufnahmequalität

Es war gerade ein Ansatz dieser Arbeit, mit möglichst billiger Hardware gute Ergebnisse zu erzielen. Der Preisverfall auf dem Digitalkameramarkt sollte es in den nächsten Jahren möglich machen, zu einem vergleichbaren Preis „richtige“ Fotokameras einsetzen zu können. Diese bieten den Vorteil erhöhter Bildqualität.

7.2.3.5 Erstellen eines Viewers

Die hier vorgestellte Lösung bietet im Wesentlichen nur die Verarbeitung bis zum fertigen Bild. Dieses wird dann auf einem Webserver zur Verfügung gestellt.

Interessant sind aber auch die weiteren Schritte bis hin zur Anzeige beim Benutzer. Beipielsweise könnte eine perspektivische anpassbare Anzeige des zylindrischen Modells erfolgen. Zu diesem Zweck gibt es bereits fertige Software, die man evaluieren müsste.

Weiterhin könnte man Zusatzinformationen wie interessante Punkte (Point Of Interest, POI) oder Wetterdaten und ähnliches auf dem Bild einblenden.

7.2.3.6 Zeitraffervideogenerierung

Ein visuell sicher sehr ansprechendes Feature ist die automatische Generierung von Zeitrafferviedeos aus den Panoramen.

–leere Seite–

Literatur

Acosta-Serafini, P., I. Masaki, und C. Sodini, A 1/3“ vga linear wide dynamic range cmos image sensor implementing a predictive multiple sampling algorithm with overlapping integration intervals, *IEEE Journal of Solid-State Circuits*, *39*, 1487–1496, 2004.

Adams, A., *Basic Photo*, Morgan & Morgan, Hastings-on-Hudson, New York, 1970.

Adelson, E., und J. Bergen, The Plenoptic Function and the Elements of Early Vision, *Computational Models of Visual Processing*, 1991.

Adobe Systems Incorporated, Tiff technical note 3, *Tech. rep.*, Adobe Systems Incorporated, 2005.

Aggarwal, M., und N. Ahuja, High dynamic range panoramic imaging, *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, *1*, 2–9 vol.1, 2001.

Aggarwal, M., H. Hua, und N. Ahuja, On cosine-fourth and vignetting effects in real lenses, *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, *1*, 472–479 vol.1, 2001.

Aimone, C., und S. Mann, Camera response function recovery from auto-exposure cameras, *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, *4*, IV –233–IV –236, 2007.

Akyüz, A., und E. Reinhard, Color appearance in high-dynamic-range imaging, *Journal of Electronic Imaging*, *15*, 033,001, 2006.

Akyüz, A. O., R. Fleming, B. E. Riecke, E. Reinhard, und H. H. Bülthoff, Do hdr displays support ldr content?: a psychophysical evaluation, in *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, S. 38, ACM Press, New York, NY, USA, 2007.

Aldus Corporation, Tiff revision 6.0, *Tech. rep.*, Adobe Systems Incorporated, 1992.

Asada, N., A. Amano, und M. Baba, Photometric calibration of zoom lens systems, in *ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I*, S. 186, IEEE Computer Society, Washington, DC, USA, 1996.

Ashikhmin, M., A tone mapping algorithm for high contrast images, in *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pp. 145–156, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2002.

Barnea, D., und H. Silverman, A class of algorithms for fast digital image registration, *IEEE Trans. Comput*, *21*, 179–186, 1972.

Barros, A., und F. Candocia, Image registration in range using a constrained piecewise linear model, *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, *4*, IV–3345–IV–3348 vol.4, 2002.

Bayer, B., Color imaging array us patent 3971065, 1976.

Beier, T., und S. Neely, Feature-based image metamorphosis, *SIGGRAPH Comput. Graph.*, *26*, 35–42, 1992.

Beis, J. S., und D. G. Lowe, Shape indexing using approximate nearest-neighbour search in high-dimensional spaces, *cvpr*, *00*, 1000, 1997.

Bergmann, L., und C. Schaefer, *Lehrbuch der Experimentalphysik – Elektromagnetismus*, vol. 2, 9 ed., de Gruyter, 2004a.

Bergmann, L., und C. Schaefer, *Lehrbuch der Experimentalphysik – Optik: Wellen- und Teilchenoptik*, vol. 3, 10 ed., de Gruyter, 2004b.

Bergmann, L., und C. Schaefer, *Lehrbuch der Experimentalphysik – Festkörper*, vol. 6, 2 ed., de Gruyter, 2005.

Bichsel, M., und K. W. Ohnesorge, How to measure a camera"s response curve from scratch, *Tech. rep.*, University of Zurich, 1993.

Blackwell, H. R., An analytical model for describing the influence of lighting parameters upon visual performance, in *Technical Foundations*, vol. 1, Commission Internationale De L'Eclairage, 1981.

Born, M., und E. Wolf, *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, Cambridge University Press, 2000.

Box, D., *Essential COM*, 5. print. ed., Addison Wesley, 1999.

Boyle, W., und G. Smith, The inception of charge-coupled devices, *IEEE Transactions on Electron Devices*, *23*, 661–663, 1976.

Boyle, W. S., und G. E. Smith, *Charge coupled semiconductor devices (Bell System Technical Journal 1970)*, pp. 475–+, Selected Papers on Instrumentation in Astronomy, 1993.

Brajovic, V., und T. Kanade, A sorting image sensor : An example of massively parallel intensity-to-time processing for low-latency computational sensors, 1996.

Britannica, E., *Optics – Britannica Online Encyclopedia*, 2008.

Brockhaus, *Der Brockhaus Computer und Informationstechnologie*, Brockhaus, 2003.

Brockhaus, *Brockhaus Enzyklopädie in 30 Bänden*, 15 ed., F.A. Brockhaus GmbH, Leipzig, Bibliografisches Institut und F.A. Brockhaus AG, Mannheim, 2006a.

Brockhaus, *Brockhaus Enzyklopädie in 30 Bänden*, 20 ed., F.A. Brockhaus GmbH, Leipzig, Bibliografisches Institut und F.A. Brockhaus AG, Mannheim, 2006b.

Brockhaus, *Brockhaus Enzyklopädie in 30 Bänden*, 21 ed., F.A. Brockhaus GmbH, Leipzig, Bibliografisches Institut und F.A. Brockhaus AG, Mannheim, 2006c.

Brown, M., und D. Lowe, Automatic Panoramic Image Stitching using Invariant Features, *International Journal of Computer Vision*, *74*, 59–73, 2007.

Brown, M., und D. G. Lowe, Recognising panoramas, in *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, S. 1218, IEEE Computer Society, Washington, DC, USA, 2003.

Burt, P., und E. Adelson, A multiresolution spline with application to image mosaics, *ACM Transactions on Graphics*, *2*, 217–236, 1983.

Candocia, F., Jointly registering images in domain and range by piecewise linear comparametric analysis, *Image Processing, IEEE Transactions on*, *12*, 409–419, 2003.

Candocia, F., Analysis and enhancements to piecewise linear comparametric image registration, *Image Processing, IEEE Transactions on*, *14*, 181–188, 2005.

Candocia, F. M., und D. A. Mandarino, A semiparametric model for accurate camera response function modeling and exposure estimation from comparametric data, *Image Processing, IEEE Transactions on*, *14*, 1138–1150, 2005.

Canon Inc., Ef lens work iii, 2006.

Canon Inc., Eos 1d mark iii whitepaper, 2007.

Capel, D., und A. Zisserman, Automated mosaicing with super-resolution zoom, *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pp. 885–891, 1998.

Carranza, J., C. Theobalt, M. A. Magnor, und H.-P. Seidel, Free-viewpoint video of human actors, *ACM Trans. Graph.*, *22*, 569–577, 2003.

Chen, H.-T., T.-L. Liu, und C.-S. Fuh, Tone reproduction: A perspective from luminance-driven perceptual grouping, *Int. J. Comput. Vision*, *65*, 73–96, 2005.

Chen, S. E., und L. Williams, View interpolation for image synthesis, in *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 279–288, ACM, New York, NY, USA, 1993.

Choudhury, P., und J. Tumblin, The trilateral filter for high contrast images and meshes, in *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, S. 5, ACM, New York, NY, USA, 2005.

Chung, E., Y. Huang, S. Yajnik, D. Liang, J. C. Shih, C.-Y. Wang, und Y.-M. Wang, DCOM and CORBA side by side, step by step and layer by layer, 1997.

Debevec, P. E., und J. Malik, Recovering high dynamic range radiance maps from photographs, *Computer Graphics*, *31*, 369–378, 1997.

Devernay, F., und O. Faugeras, Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments, *Mach. Vision Appl.*, *13*, 14–24, 2001.

Devlin, K., A review of tone reproduction techniques, *Tech. Rep. CSTR-02-005*, Department of Computer Science, University of Bristol, 2002.

Devlin, K., A. Chalmers, A. Wilkie, und W. Purgathofer, Tone reproduction and physically based spectral rendering, 2002.

Diestel, R., *Graphentheorie*, 3., neu bearb. und erw. aufl. ed., Springer, 2006.

Drago, F., W. Martens, K. Myszkowski, und H.-P. Seidel, Perceptual evaluation of tone mapping operators with regard to similarity and preference, *Research Report MPI-I-2002-4-002*, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2002.

Drago, F., W. L. Martens, K. Myszkowski, und H.-P. Seidel, Perceptual evaluation of tone mapping operators, in *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*, pp. 1–1, ACM, New York, NY, USA, 2003a.

Drago, F., K. Myszkowski, T. Annen, und N. Chiba, Adaptive Logarithmic Mapping For Displaying High Contrast Scenes, *Computer Graphics Forum*, *22*, 419–426, 2003b.

Durand, F., und J. Dorsey, Interactive tone mapping, in *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pp. 219–230, Springer-Verlag, London, UK, 2000.

Durand, F., und J. Dorsey, Fast bilateral filtering for the display of high-dynamic-range images, in *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 257–266, ACM, New York, NY, USA, 2002.

Eastman Kodak Company, Charge-coupled device (ccd) image sensors, 2001.

Eastman Kodak Company, Kaf-18000 image sensor, 2006.

Eddon, G., und H. Eddon, *Inside distributed COM*, Microsoft Press, Redmond, WA, USA, 1998.

Eisemann, M., B. D. Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, und A. Sellent, Floating Textures, *Computer Graphics Forum (Proc. Eurographics EG'08)*, *27*, xx–xx, 2008.

Encarnação, J. L., W. Straßer, und R. Klein, *Graphische Datenverarbeitung*, Oldenbourg, 1996.

Fairchild, M., und G. Johnson, Meet icam: A next-generation color appearance model, 2002.

Farid, H., Blind inverse gamma correction, *Image Processing, IEEE Transactions on*, *10*, 1428–1433, 2001.

Farid, H., und A. Popescu, Blind removal of lens distortion, *Journal of the Optical Society of America A*, *18*, 2072–2078, 2001.

Fattal, R., D. Lischinski, und M. Werman, Gradient domain high dynamic range compression, 2002.

Faugeras, O., *Three-dimensional computer vision: a geometric viewpoint*, MIT Press, Cambridge, MA, USA, 1993.

Ferwerda, J. A., S. N. Pattanaik, P. Shirley, und D. P. Greenberg, A model of visual adaptation for realistic image synthesis, in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 249–258, ACM Press, New York, NY, USA, 1996.

Fischler, M. A., und R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM*, *24*, 381–395, 1981.

Fitzgibbon, A., Simultaneous linear estimation of multiple view geometry and lens distortion, *Computer Vision and Pattern Recognition, Proceedings of IEEE Conference on*, *1*, 125–132, 2001.

Fossum, E., Cmos image sensors: electronic camera-on-a-chip, *Electron Devices, IEEE Transactions on*, *44*, 1689–1698, 1997.

Funt, B. V., F. Ciurea, und J. J. McCann, Tuning retinex parameters, vol. 4662, pp. 358–366, SPIE, 2002.

Glatz, E., *Betriebssysteme*, 1. aufl. ed., dpunkt, 2006.

Grehn, F., *Augenheilkunde*, Springer, Heidelberg, 2006.

Grossberg, M., und S. Nayar, What can be known about the radiometric response from images?, *Lecture Notes in Computer Science, Proc. ECCV*, *2353*, 189–205, 2002.

Grossberg, M., und S. Nayar, What is the Space of Camera Response Functions?, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. II, pp. 602–609, 2003a.

Grossberg, M., und S. Nayar, Determining the Camera Response from Images: What is Knowable?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1455–1467, 2003b.

Grossberg, M., und S. Nayar, Modeling the Space of Camera Response Functions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 1272–1282, 2004.

Hansen, L. L., *Augenheilkunde systematisch*, UNI-MED-Verl., Bremen [u.a.], 2007.

Hassan, F., und J. Carletta, A high throughput encoder for high dynamic range images, *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, *6*, VI –213–VI –216, 2007.

Healey, G., und R. Kondepudy, Radiometric ccd camera calibration and noise estimation, *IEEE Trans. Pattern Anal. Mach. Intell.*, *16*, 267–276, 1994.

Hecht, E., *Optik*, 4., überarb. aufl. ed., Oldenbourg, 2005.

Horn, B., und B. Schunck, Determining Optical Flow, *Artificial Intelligence*, *17*, 185–203, 1981.

Horn, B. K. P., Determining lightness from an image, *j-CGIP*, *3*, 277–299, 1974.

Horn, B. K. P., *Robot vision*, 3. print. ed., MIT Pr., 1987.

Hough, P., Method and means for recognizing complex patterns US Patent 3069654, *Washington, DC: Patent and Trademark Office*, 1962.

Höfflinger, B., *High-Dynamic-Range (HDR) Vision: Microelectronics, Image Processing, Computer Graphics*, Springer, 2007.

ILFORD Imaging UK, Fact sheet delta 3200 professional, 2002.

ILFORD Imaging UK, Fact sheet fp4-plus, 2004.

Irawan, P., J. A. Ferwerda, und S. R. Marschner, Perceptually based tone mapping of high dynamic range image streams, 2005.

ITU, Iso/iec 10918-1 : 1993(e) ccit recommendation t.81, 1993.

Jacobs, K., G. Ward, und C. Loscos, Automatic hdri generation of dynamic environments, in *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, S. 43, ACM, New York, NY, USA, 2005.

JEITA, Exchangeable image file format for digital still cameras: Exif version 2.2, *Tech. rep.*, Japan Electronics and Information Technology Industries Association, 2002.

Jobson, D., Z. Rahman, und G. Woodell, A multiscale retinex for bridging the gap between color images and the human observation of scenes, *Image Processing, IEEE Transactions on*, *6*, 965–976, 1997.

Johnson, G. M., Cares and concerns of CIE TC8-08: spatial appearance modeling and HDR rendering, in *Image Quality and System Performance II. Edited by Rasmussen, Rene; Miyake, Yoichi. Proceedings of the SPIE, Volume 5668, pp. 148-156 (2004).*, edited by R. Rasmussen und Y. Miyake, vol. 5668 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pp. 148–156, 2004.

Johnson, G. M., und M. D. Fairchild, Rendering hdr images, *Eleventh Color Imaging Conference: Color Science and Engineering Systems, Technologies, Applications*, *11*, 36–41, 2003.

Kang, S. B., M. Uyttendaele, S. Winder, und R. Szeliski, High dynamic range video, *ACM Trans. Graph.*, *22*, 319–325, 2003.

Kawahito, S., CMOS Imaging Devices for New Markets of Vision Systems, *IEICE Trans Electron*, *E90-C*, 1858–1868, 2007.

Khan, E., A. Akyuz, und E. Reinhard, Ghost removal in high dynamic range images, *Image Processing, 2006 IEEE International Conference on*, pp. 2005–2008, 2006.

Kim, S. J., und M. Pollefeys, Radiometric alignment of image sequences, *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, *1*, I–645–I–651 Vol.1, 2004.

Kolb, C., D. Mitchell, und P. Hanrahan, A realistic camera model for computer graphics, in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 317–324, ACM, New York, NY, USA, 1995.

Krawczyk, G., M. Goesele, und H.-P. Seidel, Photometric calibration of high dynamic range cameras, *Research Report MPI-I-2005-4-005*, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2005.

Kuang, J., H. Yamaguchi, C. Liu, G. M. Johnson, und M. D. Fairchild, Evaluating hdr rendering algorithms, *ACM Trans. Appl. Percept.*, *4*, 9, 2007.

Land, E. H., und J. J. McCann, Lightness and retinex theory, *Journal of the Optical Society America*, *61*, 1, 1971.

Larson, G., und R. Shakespeare, *Rendering with radiance: the art and science of lighting visualization*, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1998.

Larson, G. W., H. Rushmeier, und C. Piatko, A visibility matching tone reproduction operator for high dynamic range scenes, *IEEE Transactions on Visualization and Computer Graphics*, *03*, 291–306, 1997.

Ledda, P., A. Chalmers, und H. Seetzen, A psychophysical validation of tone mapping operators using a high dynamic range display, in *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pp. 159–159, ACM, New York, NY, USA, 2004a.

Ledda, P., L. P. Santos, und A. Chalmers, A local model of eye adaptation for high dynamic range images, in *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pp. 151–160, ACM Press, New York, NY, USA, 2004b.

Ledda, P., A. Chalmers, T. Troscianko, und H. Seetzen, Evaluation of tone mapping operators using a high dynamic range display, *ACM Trans. Graph.*, *24*, 640–648, 2005.

Lewis, J., Fast normalized cross-correlation, in *Vision Interface*, pp. 120–123, Canadian Image Processing and Pattern Recognition Society, 1995.

Lin, S., und L. Zhang, Determining the radiometric response function from a single grayscale image, *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, *2*, 66–73 vol. 2, 2005.

Lin, S., J. Gu, S. Yamazaki, und H.-Y. Shum, Radiometric calibration from a single image, *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, *2*, II–938–II–945 Vol.2, 2004.

Lindeberg, T., Scale-space theory: a basic tool for analyzing structures at different scales, *Journal of Applied Statistics*, *21*, 225–270, 1994.

Liu, X., und A. El Gamal, Synthesis of high dynamic range motion blur free image from multiple captures, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on [see also Circuits and Systems I: Regular Papers, IEEE Transactions on]*, *50*, 530–539, 2003.

Lowe, D., Object recognition from local scale-invariant features, *International Conference on Computer Vision*, *2*, 1150–1157, 1999.

Lowe, D., Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, *60*, 91–110, 2004.

Lowe, D. G., Local feature view clustering for 3d object recognition, *cvpr*, *1*, 682, 2001.

Lucas, B., und T. Kanade, An iterative image registration technique with an application to stereo vision, in *Seventh International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Madden, B., Extended intensity range imaging, 1993.

Manders, C., C. Aimone, und S. Mann, Camera response function recovery from different illuminations of identical subject matter, *Image Processing, 2004. ICIP '04. 2004 International Conference on*, *5*, 2965–2968 Vol. 5, 2004.

Mann, S., Personal imaging and lookpainting as tools for personal documentary and investigative photojournalism, *Mob. Netw. Appl.*, *4*, 23–36, 1999.

Mann, S., Comparametric equations with practical applications in quantigraphic image processing, *Image Processing, IEEE Transactions on*, *9*, 1389–1406, 2000.

Mann, S., und R. Mann, Quantigraphic imaging: Estimating the camera response and exposures from differently exposed images, *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, *1*, I–842–I–849 vol.1, 2001.

Mann, S., und R. Picard, Being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures, 1995.

Mann, S., und R. W. Picard, Video orbits of the projective group a simple approach to featureless estimation of parameters, *Image Processing, IEEE Transactions on*, *6*, 1281–1295, 1997.

Mantiuk, R., K. Myszkowski, und H. Seidel, A perceptual framework for contrast processing of high dynamic range images, *ACM Transactions on Applied Perception (TAP)*, *3*, 286–308, 2006.

Mantiuk, R., G. Krawczyk, R. Mantiuk, und H. Seidel, High dynamic range imaging pipeline: Perception-motivated representation of visual content, *Human Vision and Electronic Imaging XII. Edited by Rogowitz, Bernice E.; Pappas, Thrasyvoulos N.; Daly, Scott J.. Proceedings of the SPIE,*, *6492*, 649,212, 2007.

Matsushita, Y., und S. Lin, Radiometric calibration from noise distributions, *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, 2007.

Mcluhan, H. M., *The Gutenberg Galaxy: The Making of Typographic Man*, University of Toronto Press, 1962.

McMillan, L., und G. Bishop, Plenoptic modeling: an image-based rendering system, in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 39–46, ACM, New York, NY, USA, 1995.

McMillan Jr, L., AN IMAGE-BASED APPROACH TO THREE-DIMENSIONAL COMPUTER GRAPHICS, Ph.D. thesis, University of North Carolina, 1997.

Microsoft Corporation, *Microsoft Developer Network Online Documentation*, http://msdn2.microsoft.com, 2008.

Mikolajczyk, K., und C. Schmid, A performance evaluation of local descriptors, *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, *2*, II–257–II–263 vol.2, 2003.

Mikolajczyk, K., und C. Schmid, A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 1615–1630, 2005.

Miller, G. S., und C. R. Hoffman, Illumination and reflection maps: Simulated objects in simulated and real environments, in *SIGGRAPH '84 Advanced Computer Graphics Animation seminar notes*, ACM, New York, NY, USA, 1984.

Mitsunaga, T., und S. Nayar, Radiometric Self Calibration, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 374–380, 1999.

Naka, K. I., und W. A. H. Rushton, The generation and spread of S-potentials in fish (Cyprinidae), *J Physiol*, *192*, 437–461, 1967.

Narasimhan, S., R. Visvanathan, und S. Nayar, A class of photometric invariants: separating material from shape and illumination, *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1387–1394 vol.2, 2003.

Nayar, S., und T. Mitsunaga, High dynamic range imaging: spatially varying pixel exposures, 2000.

Ng, T.-T., S.-F. Chang, und M.-P. Tsui, Using geometry invariants for camera response function estimation, *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, 2007.

O'Malley, S., A simple, effective system for automated capture of high dynamic range images, *Computer Vision Systems, 2006 ICVS '06. IEEE International Conference on*, pp. 15–15, 2006.

Oppenheim, A., R. Schafer, und T. S. Jr., Nonlinear filtering of multiplied and convolved signals, *Proceedings of the IEEE*, *56*, 1264–1291, 1968.

Pal, C., R. Szeliski, M. Uyttendaele, und N. Jojic, Probability models for high dynamic range imaging, *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, *2*, II–173–II–180 Vol.2, 2004.

Pattanaik, S., und H. Yee, Adaptive gain control for high dynamic range image display, in *SCCG '02: Proceedings of the 18th spring conference on Computer graphics*, pp. 83–87, ACM, New York, NY, USA, 2002.

Pattanaik, S. N., J. A. Ferwerda, M. D. Fairchild, und D. P. Greenberg, A multiscale model of adaptation and spatial vision for realistic image display, in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 287–298, ACM, New York, NY, USA, 1998.

Pattanaik, S. N., J. Tumblin, H. Yee, und D. P. Greenberg, Time-dependent visual adaptation for fast realistic image display, in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 47–54, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

Peleg, S., und J. Herman, Panoramic mosaics by manifold projection, *cvpr*, *00*, 338, 1997.

Peli, E., Contrast in complex images, *J. Opt. Soc. Am. A*, *7*, 2032, 1990.

Petzold, C., *Windows-Programmierung*, 5. auflage ed., Microsoft Press Deutschland, 2000.

Pollefeys, M., R. Koch, und L. V. Gool, A simple and efficient rectification method for general motion, *iccv*, *01*, 496, 1999.

Poynton, C. A., Rehabilitation of gamma, in *Proc. SPIE Vol. 3299, p. 232-249, Human Vision and Electronic Imaging III, Bernice E. Rogowitz; Thrasyvoulos N. Pappas; Eds.*, edited by B. E. Rogowitz und T. N. Pappas, vol. 3299 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pp. 232–249, 1998.

Reinhard, E., Parameter estimation for photographic tone reproduction, *J. Graph. Tools*, *7*, 45–52, 2002.

Reinhard, E., und K. Devlin, Dynamic Range Reduction Inspired by Photoreceptor Physiology, 2005a.

Reinhard, E., und K. Devlin, Dynamic range reduction inspired by photoreceptor physiology, *Visualization and Computer Graphics, IEEE Transactions on*, *11*, 13–24, 2005b.

Reinhard, E., M. Stark, P. Shirley, und J. Ferwerda, Photographic tone reproduction for digital images, *ACM Transactions on Graphics*, *21*, 267–276, 2002a.

Reinhard, E., M. Stark, P. Shirley, und J. Ferwerda, Photographic tone reproduction for digital images, *ACM Trans. Graph.*, *21*, 267–276, 2002b.

Reinhard, E., G. Ward, S. Pattanaik, und P. Debevec, *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*, Morgan Kaufmann, 2005.

Rempel, A. G., M. Trentacoste, H. Seetzen, H. D. Young, W. Heidrich, L. Whitehead, und G. Ward, Ldr2hdr: on-the-fly reverse tone mapping of legacy video and photographs, in *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, S. 39, ACM Press, New York, NY, USA, 2007.

Robertson, M., S. Borman, und R. Stevenson, Dynamic range improvement through multiple exposures, *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, *3*, 159–163 vol.3, 1999.

Robertson, M. A., S. Borman, und R. L. Stevenson, Estimation-theoretic approach to dynamic range enhancement using multiple exposures, *Journal of Electronic Imaging*, *12*, 219–228, 2003.

Sangster, F., und K. Teer, Bucket-brigade electronics: new possibilities for delay, time-axis conversion, and scanning, *IEEE Journal of Solid-State Circuits*, *4*, 131–136, 1969.

Schaffalitzky, F., und A. Zisserman, Multi-view matching for unordered image sets, or ¨how do i organize my holiday snaps?¿ in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pp. 414–431, Springer-Verlag, London, UK, 2002.

Schlick, C., Quantization techniques for the visualization of high dynamic range pictures, in *Photorealistic Rendering Techniques*, pp. 7–20, Springer-Verlag, New York, 1994.

Schmid, C., und R. Mohr, Local grayvalue invariants for image retrieval, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 530–535, 1997.

Schupp, H., *Elementargeometrie*, Schoeningh, c 1977.

Seetzen, H., L. A. Whitehead, und G. Ward, P. 54.2: A high dynamic range display using low and high resolution modulators, 2003.

Seetzen, H., W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, und A. Vorozcovs, High dynamic range display systems, 2004.

Shafique, K., und M. Shah, Estimation of the radiometric response functions of a color camera from differently illuminated images, *Image Processing, 2004. ICIP '04. 2004 International Conference on*, *4*, 2339–2342 Vol. 4, 2004.

Shah, S., und J. Aggarwal, A simple calibration procedure for fish-eye (high distortion) lens camera, *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3422–3427 vol.4, 1994.

Sony, Icx098bq, 2003.

Spencer, G., P. Shirley, K. Zimmerman, und D. P. Greenberg, Physically-based glare effects for digital images, in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 325–334, ACM, New York, NY, USA, 1995.

Stevens, J. C., und S. S. Stevens, Brightness function: Effects of adaptation, *J. Opt. Soc. Am.*, *53*, 375–385, 1963.

Swaminathan, R., und S. Nayar, Nonmetric calibration of wide-angle lenses and polycameras, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 1172–1178, 2000.

Swaninathan, R., M. Grossberg, und S. Nayar, A perspective on distortions, *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, *2*, 2003.

Szeliski, R., und H.-Y. Shum, Creating full view panoramic image mosaics and environment maps, in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 251–258, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997.

Tanenbaum, A. S., *Modern operating systems*, 2. ed. ed., Prentice Hall, 2001.

Tanenbaum, A. S., und M. v. Steen, *Distributed systems*, Prentice Hall, 2002.

Theuwissen, A. J. P., *Solid-state imaging with charge coupled devices*, Kluwer Academic Publ., 1995.

Thompson, D., C. Exton, L. Garrett, A. Sajeev, und D. Watkins, Distributed component object model (dcom), 1997.

Tian, H., B. Fowler, und A. Gamal, Analysis of temporal noise in cmos photodiode active pixel sensor, 2001.

Tomaszewska, A., und R. Mantiuk, Image registration for multi-exposure high dynamic range image acquisition, in *The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2007*, 2007.

Trucco, E., und A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

Tsin, Y., V. Ramesh, und T. Kanade, Statistical calibration of ccd imaging process, *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, *1*, 480–487 vol.1, 2001.

Tumblin, J., und H. Rushmeier, Tone reproduction for realistic images, *Computer Graphics and Applications, IEEE*, *13*, 42–48, 1993.

Tumblin, J., und G. Turk, Lcis: a boundary hierarchy for detail-preserving contrast reduction, in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 83–90, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

Tumblin, J., J. K. Hodgins, und B. K. Guenter, Two methods for display of high contrast images, *ACM Trans. Graph.*, *18*, 56–94, 1999.

Uyttendaele, M., A. Eden, und R. Skeliski, Eliminating ghosting and exposure artifacts in image mosaics, *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, *2*, II–509–II–516 vol.2, 2001.

Vedula, S., S. Baker, und T. Kanade, Image-based spatio-temporal modeling and view interpolation of dynamic events, *ACM Trans. Graph.*, *24*, 240–261, 2005.

Vinogradov, I. M. (Ed.), *Encyclopaedia of Mathematics*, vol. 5, Kluwer Academic Publishers, Doordrecht, 1990.

Wallace, G. K., The jpeg still picture compression standard, *Commun. ACM*, *34*, 30–44, 1991.

Ward, G., A contrast-based scalefactor for luminance display, pp. 415–421, 1994.

Ward, G., High dynamic range imaging, 2001.

Ward, G., Fast, robust image registration for compositing high dynamic range photographcs from hand-held exposures, *j-J-GRAPHICS-TOOLS*, *8*, 17–30, 2003.

Ward, G., High dynamic range image encodings, 2005.

Ward, G., und M. Simmons, Subband encoding of high dynamic range imagery, in *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pp. 83–90, ACM Press, New York, NY, USA, 2004.

Wood, D. N., A. Finkelstein, J. F. Hughes, C. E. Thayer, und D. H. Salesin, Multiperspective panoramas for cel animation, in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 243–250, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997.

Wyckoff, C. W., An experimental extended response film, 1962.

Wyszecki, G., und W. S. Stiles, *Color science*, Wiley, 1967.

Yee, Y. H., und S. N. Pattanaik, Segmentation and adaptive assimilation for detail-preserving display of high-dynamic range images, *19*, 457–466, 2003.

Young-Chang, C., und J. Reid, Rgb calibration for color image analysis in machine vision, *Image Processing, IEEE Transactions on*, *5*, 1414–1422, 1996.

Zettl, H., *Sight Sound Motion : Applied Media Aesthetics (with InfoTrac)*, Wadsworth Publishing, 2004.

Zhang, Z., R. Deriche, O. Faugeras, und Q. Luong, A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry, *Artificial Intelligence*, *78*, 87–119, 1995.

Index