Real-Time Algorithms for High Dynamic Range Video

Inauguraldissertation zur Erlangung
des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Dipl. Inf. Benjamin Guthier
aus Schwetzingen

Mannheim, 2012

Dekan:	Prof. Dr. Heinz Jürgen Müller, Universität Mannheim
Referent:	Prof. Dr. Wolfgang Effelsberg, Universität Mannheim
Korreferent:	Prof. Dr. Jürgen Hesser, Universität Heidelberg

Tag der mündlichen Prüfung: 27. Juni 2012

Danksagungen

An dieser Stelle möchte ich all denjenigen danken, die mich bei und während der Entstehung dieser Arbeit unterstützt haben. Als erstes sei mein Doktorvater Prof. Wolfgang "Effels" Effelsberg genannt, der mir die Möglichkeit zur Promotion an seinem Lehrstuhl gegeben hat. Dank seiner sehr herzlichen Art und seiner schier unerschöpflichen Geduld, war meine gesamte Promotion eine überaus positive Erfahrung – wenn auch nicht immer alles glatt lief (siehe Leiterplatten-Projekt...). Durch den Wechsel auf die Landesstelle später schaffte es Prof. Effelsberg, mein Interesse an der Lehre zu wecken, und beeinflusste damit maßgeblich meine Entscheidung, weiterhin an der Uni zu bleiben.
Außerdem möchte ich meinen lieben Kollegen danken, die in all der Zeit für ein ausgesprochen angenehmes Arbeitsklima gesorgt haben (danke Mino!). Die Stimmung am Lehrstuhl war immer sehr freundschaftlich und ermöglichte dadurch eine gute Zusammenarbeit an diversen Papern, von der am Ende alle profitieren konnten. Besondere Erwähnung verdient mein Kollege Stephan, der für mich vor allem am Anfang, aber auch später noch ein Mentor war, der mir immer wieder nützliche Tipps zum wissenschaftlichen Arbeiten geben konnte.
Ganz besonderer Dank gilt meinen Eltern Manfred und Annette. Ihre immer währende Unterstützung ermöglichte mir zunächst das Studium und dann später die Promotion. Auf meine Eltern konnte ich mich bisher mein gesamtes Leben lang bedingungslos verlassen. Ihrer Erziehung und Fürsorge schreibe ich einen großen Teil meines Erfolges bei der Promotion zu. Es tut gut zu wissen, dass sie jetzt sehr stolz auf mich sind!
Wenn auch nicht direkt an meiner Doktorarbeit beteiligt, möchte ich hier noch meine langjährigen Freunde erwähnen. Das sind vor allem mein bester Kumpel Rene, die Sveni und die Rini und der gesamte Treff. Danke, dass ihr es so lange mit mir ausgehalten habt! Ich hab euch alle lieb!

Abstract

A recurring problem in capturing video is the scene having a range of brightness values that exceeds the capabilities of the capturing device. An example would be a video camera in a bright outside area, directed at the entrance of a building. Because of the potentially big brightness difference, it may not be possible to capture details of the inside of the building and the outside simultaneously using just one shutter speed setting. This results in under- and overexposed pixels in the video footage. The approach we follow in this thesis to overcome this problem is temporal exposure bracketing, i.e., using a set of images captured in quick sequence at different shutter settings. Each image then captures one facet of the scene's brightness range. When fused together, a high dynamic range (HDR) video frame is created that reveals details in dark and bright regions simultaneously.

The process of creating a frame in an HDR video can be thought of as a pipeline where the output of each step is the input to the subsequent one. It begins by capturing a set of regular images using varying shutter speeds. Next, the images are aligned with respect to each other to compensate for camera and scene motion during capture. The aligned images are then merged together to create a single HDR frame containing accurate brightness values of the entire scene. As a last step, the HDR frame is tone mapped in order to be displayable on a regular screen with a lower dynamic range.

This thesis covers algorithms for these steps that allow the creation of HDR video in real-time. When creating videos instead of still images, the focus lies on high capturing and processing speed and on assuring temporal consistency between the video frames. In order to achieve this goal, we take advantage of the knowledge gained from the processing of previous frames in the video. This work addresses the following aspects in particular. The image size parameters for the set of base images are chosen such that only as little image data as possible is captured. We make use of the fact that it is not always necessary to capture full size images when only small portions of the scene require HDR. Avoiding redundancy in the image material is an obvious approach to reducing the overall time taken to generate a frame. With the aid of the previous frames, we calculate brightness statistics of the scene. The exposure values are chosen in a way, such that frequently occurring brightness values are well-exposed in at least one of the images in the sequence. The base images from which the HDR frame is created are captured in quick succession. The effects of intermediate camera motion are thus less intense than in the still image case, and a comparably simpler camera motion model can be used. At the same time, however, there is much less time available to estimate motion. For this reason, we use a fast heuristic that makes use of the motion information obtained in previous frames. It

is robust to the large brightness difference between the images of an exposure sequence. The range of luminance values of an HDR frame must be tone mapped to the displayable range of the output device. Most available tone mapping operators are designed for still images and scale the dynamic range of each frame independently. In situations where the scene's brightness statistics change quickly, these operators produce visible image flicker. We have developed an algorithm that detects such situations in an HDR video. Based on this detection, a temporal stability criterion for the tone mapping parameters then prevents image flicker.

All methods for capture, creation and display of HDR video introduced in this work have been fully implemented, tested and integrated into a running HDR video system. The algorithms were analyzed for parallelizability and, if applicable, adjusted and implemented on a high-performance graphics chip.

Zusammenfassung

Wenn eine Szene stärkere Helligkeitsunterschiede aufweist als eine Kamera aufzeichnen kann, führt dies oft zu Problemen bei der Aufnahme. Ein typisches Beispiel ist eine Videokamera, die im hellen Bereich vor einem Gebäude aufgestellt und auf dessen Eingang gerichtet wird. Aufgrund der potentiell großen Helligkeitsunterschiede kann es unmöglich sein, Details aus dem Inneren und dem Äußeren des Gebäudes gleichzeitig unter Verwendung von nur einer Belichtungszeit zu erfassen. Dies führt zu unter- und überbelichteten Pixeln im Videomaterial. Zur Lösung dieses Problems wird in der vorliegenden Arbeit das so genannte "Temporal Exposure Bracketing" verwendet. Dabei wird eine Serie von Bildern mit unterschiedlichen Belichtungszeiten in schneller Folge aufgenommen. Jedes einzelne Bild enthält dann einen Teil der gesamten Helligkeitsspanne der Szene. Durch Verschmelzen der Bildserie entsteht ein High-Dynamic-Range-Bild (HDR-Bild), in dem sowohl Details der dunklen als auch der hellen Bereiche gleichzeitig zu erkennen sind. Schnelles, wiederholtes Aufnehmen solcher HDR-Bilder führt dann zu einem HDR-Video.

Der Ablauf zur Erzeugung eines Einzelbildes in einem HDR-Video entspricht einer Pipeline, bei der die Ausgabe jedes Teilschritts die Eingabe des nachfolgenden darstellt. Die Pipeline beginnt mit der Aufnahme einer Serie von gewöhnlichen Bildern unter variierender Belichtungszeit. Als nächstes werden die Bilder aufeinander ausgerichtet, um zwischenzeitliche Bewegung der Kamera und der Szene auszugleichen. Die ausgerichteten Bilder werden dann verschmolzen, wodurch ein HDR-Einzelbild entsteht, das genaue Helligkeitswerte der gesamten Szene enthält. Zum Schluss wird ein Tone-Mapping-Operator auf das HDR-Bild angewendet, um es auf einem Bildschirm mit niedrigerer Dynamic Range anzeigen zu können.

In dieser Arbeit werden Algorithmen für die genannten Teilschritte vorgestellt, die es ermöglichen, ein HDR-Video in Echtzeit zu erzeugen. Im Gegensatz zur Erzeugung von Standbildern, liegt bei der Erzeugung von HDR-Videos der Schwerpunkt auf einer hohen Aufnahme- und Verarbeitungsgeschwindigkeit. Außerdem muss zeitliche Konsistenz zwischen den Videobildern gewährleistet werden. Um diese Ziele zu erreichen, wird das Wissen der vorangegangenen Videobilder genutzt. Die Arbeit widmet sich den im Folgenden beschriebenen Aspekten.

Die Parameter, die den aufzunehmenden Bildbereich spezifizieren, werden so gewählt, dass so wenig zusätzliches Bildmaterial wie möglich aufgenommen werden muss. Dabei wird von der Tatsache Gebrauch gemacht, dass nicht immer komplette Bilder benötigt werden, wenn nur ein kleiner Teil der Szene schlecht belichtet ist. Redundanz im Bildmaterial zu vermeiden ist ein naheliegender Ansatz zur Reduzierung der zur HDR-

Bilderzeugung benötigten Zeit. Auf Basis der vorherigen Videobilder wird die Helligkeitsverteilung der Szene analysiert. Die Belichtungszeiten für die nächste aufzunehmende Bildserie werden so gewählt, dass häufig auftretende Helligkeitswerte in mindestens einem Bild der Serie gut belichtet werden.
Die Belichtungsreihe, aus der das HDR-Videobild erstellt wird, wird in schneller Folge aufgenommen. Die durch zwischenzeitliche Kamerabewegung entstehende Verschiebung ist deshalb weniger stark ausgeprägt als dies bei HDR-Standbildern der Fall ist. Es genügt daher, ein vergleichsweise einfaches Bewegungsmodell für die Kamera zu verwenden. Gleichzeitig steht jedoch weniger Zeit für die Bewegungsschätzung zur Verfügung. Aus diesem Grund wird eine schnelle Heuristik verwendet, die sich der Bewegungsinformation aus vorangegangenen Videobildern bedient. Sie ist robust gegenüber den starken Helligkeitsunterschieden zwischen den Bildern einer Belichtungsreihe.
Die hohe Spanne an Helligkeitswerten, die in einem HDR-Bild enthalten sind, muss durch Tone Mapping auf den darstellbaren Bereich eines Ausgabegerätes abgebildet werden. Die meisten existierenden Tone-Mapping-Operatoren wurden für Standbilder konzipiert und nehmen diese Abbildung für jedes Videobild einzeln vor. In Situationen, in denen sich die Helligkeitsverhältnisse der Szene rapide ändern, kann durch Verwendung von Standbildoperatoren ein Bildflackern entstehen. Im Rahmen dieser Arbeit wurde ein Verfahren entwickelt, das solche Situationen in einem HDR-Video erkennen kann. Ein zeitliches Stabilitätskriterium wird verwendet, um Flackern im Video zu vermeiden.
Alle Verfahren zur Aufnahme, Erzeugung und Anzeige von HDR-Videos, die in dieser Arbeit vorgestellt werden, wurden vollständig implementiert, getestet und zu einem lauffähigen HDR-Video-System integriert. Die Algorithmen wurden hinsichtlich Parallelisierbarkeit untersucht und, falls zutreffend, angepasst und für eine leistungsfähige Grafikkarte implementiert.

Contents

List of Figures

List of Tables

Nomenclature

A/D	Analog-to-Digital
AOI	Automatic Optical Inspection
CAD	Computer-Aided Design
CC	Cross Correlation
CCD	Charge-Coupled Device
CFA	Color Filter Array
CIE	Commission Internationale de l'Eclairage
CPU	Central Processing Unit
CRT	Cathode Ray Tube
CUDA	Compute Unified Device Architecture
DoF	Degrees of Freedom
DR	Dynamic Range
GPU	Graphics Processing Unit
HDR	High Dynamic Range
IIDC	Instrumentation & Industrial Digital Camera
LCD	Liquid Crystal Display
LDR	Low Dynamic Range
LED	Light-Emitting Diode
MSE	Mean Squared Error
MTB	Mean Threshold Bitmap
NCC	Normalized Cross Correlation
ND	Neutral Density
PC	Personal Computer
PCA	Printed Circuit Assembly
RAM	Random Access Memory
RANSAC	Random Sample Consensus

RGB	Red Green Blue
ROI	Region of Interest
SIFT	Scale-Invariant Feature Transform
SNR	Signal-to-Noise Ratio
TM	Tone Mapping
VAOI	Video Automatic Optical Inspection

Chapter 1

Introduction

One of the challenges in photography is finding suitable exposure values for given scenes. Natural scenes often exhibit ranges of illumination values that exceed the capabilities of the capturing device. This range of illumination values is often referred to as the dynamic range (DR) of a scene or a device, respectively. One can imagine an entrance to a building as seen in Figure 1.1. Because of the limited dynamic range of current imaging sensors, it is not possible to properly expose the dark and the bright areas simultaneously in one picture. Choosing a large aperture and a long exposure time would allow to faithfully capture the inside of the building, but would lead to white saturated pixels in the outside parts. Conversely, a small aperture and a short exposure time would reveal details of the outside while the inside of the building would appear noisy and dark. A photographer is thus confronted with the task of carefully adjusting the shutter speed and aperture so that the focus of the scene is well-exposed. This is often done at the expense of the less relevant aspects of the given scene.

A term that is often used in this context is that of *luminance*. It is a photometric quantity that correlates well with the appearance of illumination to a human observer. It is given in the unit of candela per square meter cd/m^2. Due to the adaptability of the human eye, it is difficult to estimate luminance values subjectively. We thus give an overview over some typical values here: A night scene with illumination levels from only starlight to full moon exhibits luminance values ranging from 10^{-3} to 10^{-1} cd/m^2. Indoor lighting averages around 10^2 cd/m^2, varying by about one order of magnitude for home and office lighting. Daylight, ranging from an overcast to a clear sky, achieves luminance levels from 10^3 to 10^5 cd/m^2, with direct sunlight resulting in even higher values.

For comparison, the human visual system can adapt to ten orders of magnitude of luminance variation, which is sufficient for all natural lighting conditions. Without adapting, it can see five orders of magnitudes (or "log units") of luminance in a single instant. A typical digital imaging sensor, however, can only cover three orders of magnitude under a single exposure setting. This is actually a step backwards from traditional film

Figure 1.1: *The inside of the building is much darker than the outside. There is no shutter speed setting that exposes both correctly at the same time. A longer shutter overexposes the outside (left) while a short shutter underexposes the inside of the building (center). A solution to this problem is merging the two images into one HDR image (right). Note that due to the very poor dynamic range of print, a large portion of the HDR effect may be lost in this illustration.*

photography by one log unit. The dynamic range of current sensors may be sufficient for scenes with single lighting conditions, like daylight only, but fail in situations where multiple lighting conditions meet. Examples for such situations are numerous:

- A dark indoor scene with a window to the outside,
- monitoring the entrance to a building in a surveillance scenario,
- specular reflections of direct sunlight,
- the headlights of a car at night, ...

In these cases, over- and underexposure is inevitable. In the digital image, overexposure results in saturated pixels that are fully white. Contrast is completely lost. An underexposed pixel, on the other hand, is rarely entirely black. Instead, the low amount of light hitting the surface of the sensor cell creates a response with an intensity lower than the sensor's noise threshold. The response becomes indistinguishable from noise, and luminance information is lost as well. Therefore, the shutter speed and aperture setting of a camera must be carefully chosen so that the light emitted from the scene creates a sensor response within the usable range. This constitutes an imperfect mapping between real luminance and sensor response. In high dynamic range (HDR) imaging, the goal is to accurately measure physical luminance, that is, to get around the imperfect mapping, noise and saturation effects. An HDR image then becomes a digital representation of physical quantities in the scene. This of course requires an imaging sensor with a dynamic range at least as high as the scene's DR.
As discussed earlier, most current sensors are inadequate for capturing high dynamic range scenes in a single exposure. A possible solution to this problem is using multiple exposures, i.e., capturing images in a quick sequence and varying the shutter speed setting with each shot. The result is a sequence of images with varying brightness levels. Each exposure then contains details in a different luminance range. As long as a pixel is not too dark and not too bright in at least one of the images in the sequence, it conveys

information about the luminance of the corresponding point in the scene. This can be assured by capturing enough images to cover the full dynamic range. Merging them together results in a single HDR image, which faithfully reproduces dark and bright objects at the same time (see Figure 1.1, right image). This approach can be extended to video by reiterating the process of capturing an image sequence and merging it into an HDR video frame. Given that the rate at which the camera produces images is fast enough and the processing power is sufficient to merge images quickly, HDR videos can be created and displayed in real-time.

There are a number of issues to be considered when putting the above approach into practice. Most of them revolve around capturing and processing time. A typical rate to play back a video is 25 frames per second. So creating 25 HDR frames per second should be the goal of HDR video. Assuming that an HDR image can be created from four exposures, a camera would need to capture 100 exposures per second. In addition to this being a challenge for the image sensor and its data interface, there is only one hundredth of a second left for exposing each image. For this to be enough, a lens with a large aperture and possibly an increased gain is required. It is also likely that the camera or the objects in the scene move while acquiring the sequence of exposures. In order to merge them together, the intermediate camera and scene motion must be compensated. Otherwise there would be motion blur or ghosting artifacts. This motion compensation is a computationally costly step. Once the images are aligned, they can be merged together into an HDR frame, which is another costly operation. The dynamic range limitation of imaging sensors compared to the real world is also a problem for displays. Current LCD screens are insufficient for displaying the luminance range of real world scenes. The HDR image which represents real luminance values thus needs to be adapted to the display conditions. The process of compressing an HDR image to be displayable on a regular screen is called tone mapping. Depending on the desired output quality, this process can be computationally expensive, too.

Producing 25 HDR frames per second means that there are only 40 ms of processing time available for each frame. Capturing the low dynamic range exposures, aligning and merging them and then tone mapping the result for display thus needs to be performed within 40 ms. Considering this, it becomes apparent that there is a need for fast HDR algorithms to create HDR videos in real-time. Contrary to capturing HDR still images, temporal consistency becomes an issue here. To prevent artifacts like flickering, the parameter set that is used for capturing and processing a frame should not differ by too much from its predecessor. Parameters may only be changed smoothly. There are, however, also some benefits of capturing videos over still images. Most notably, one can take advantage of knowledge from previous frames. For example, the known brightness distribution of the previous frame can be analyzed to optimally estimate the sequence of shutter values to use for the next. Motion compensation is also more robust when there is information about the past trajectory of the camera.

This work focuses on creating high dynamic range video in real-time. It contains improvements to existing HDR approaches as well as novel algorithms to speed up the HDR capturing process. For this purpose, the advantage of prior knowledge from previous frames is exploited, and the newly arising problem of temporal consistency is tackled.

The thesis is structured as follows. Chapter 2 gives a technical introduction to HDR imaging, including existing approaches for capturing, motion compensation, HDR frame creation, and tone mapping. Chapter 3 then gives an overview of the high dynamic range video system presented in this thesis. The subsequent chapters describe our novel techniques for capturing LDR image sequences (4 and 5), compensating the intermediate camera motion (6), and reducing the visibility of flicker after tone mapping (7). Implementation-specific details and a performance evaluation of the HDR video system are presented in Chapter 8. HDR video can be applied in the context of automatic optical inspection of printed circuit boards. This is demonstrated in Chapter 9. Chapter 10 concludes the thesis and gives an outlook on future work.

Chapter 2

Fundamentals and Earlier Work

2.1 Light and Color

2.1.1 Measuring Light

There are two scientific domains that are concerned with measuring quantities of light. *Radiometry* measures electromagnetic waves in general, with visible light being a small part of the spectrum. *Photometry*, on the other hand, is specific to visible light and its intensity as perceived by humans. It is strongly centered around human vision. Both fields play an important role in HDR imaging. An HDR image contains real-world light values. They represent the scene as it really is. This is in contrast to regular images which store values that are specific to a particular capturing or target device. As an example, a pixel with a byte value of 150 appears to have different intensities depending on the type of display and contrast settings used, whereas a luminance value of 100 cd/m^2 is always the same. The main goal of HDR imaging is to accurately measure light in the scene by overcoming the inadequacies of the measuring sensor.

We start by discussing radiometric quantities and derive their photometric equivalents in the second half of this section. The quantities of both domains are summarized in Table 2.1. The most basic quantity to measure is *radiant energy*. It is the energy of an electromagnetic wave which is measured in joule (J). It is denoted by Q_e. The subscript e indicates that it is a radiometric quantity. Because electromagnetic waves propagate through space, one can measure the flow of radiant energy per time interval. This flow is called radiant flux or more commonly *radiant power* (P_e). Its unit is joule per second or watt (W). If one is interested in the amount of light incident on a surface, the amount of radiant energy per time unit and per unit of area may be measured. In other words, it is the total radiant flux hitting a surface, divided by its area. This is known as *irradiance* (E_e), and the unit is W/m^2.

Solid angles in space are measured in steradians (sr). They describe two-dimensional angles and are the extension of radians to three-dimensional space. One steradian is

Domain	Quantity	Symbol	Unit
Radiometry	Radiant energy	Q_e	J (joule)
	Radiant power	P_e	$J/s = W$ (watt)
	Irradiance	E_e	W/m^2
	Radiance	L_e	$W/(m^2 sr)$
Photometry	Luminous power	P_v	lm (lumen)
	Illuminance	E_v	$lm/m^2 = lx$ (lux)
	Luminous intensity	I_v	$lm/sr = cd$ (candela)
	Luminance	L_v	$lm/(m^2 sr) = cd/m^2$

Table 2.1: *Summary of all radiometric and photometric quantities relevant to this thesis.*

defined as the two-dimensional angle that covers a unit area on the surface of a sphere with a radius of 1. As an example, the sun as seen from the earth covers a solid angle of $6 \cdot 10^{-5}$ sr in the sky. A full sphere with radius 1 has an area of 4π, so 4π steradians represent the concept of "in all directions" in 3D just like 2π radians or 360° represent the same in 2D. Now, if we only consider the portion of light incident on a surface that arrives from a particular direction, we can divide the irradiance of the surface by the solid angle of the directions it comes from. The resulting radiometric quantity is called *radiance* (L_e) which is measured in $\frac{W}{m^2 sr}$. Radiance is important in the context of image formation. A pixel on an imaging sensor has a fixed surface area. The lens and the aperture of the camera limit the solid angle from which light can reach the sensor. The shutter speed setting determines how long radiant flux is integrated in the sensor cell. At the end of the integration phase, the accumulated radiant energy Q_e is converted into a voltage and read out. Putting it all together, it can be said that a sensor measures the radiance L_e in the scene, integrated over the surface area A of a pixel, the solid angle Ω of the aperture, and the exposure time t. This is expressed in the measurement equation [66]. In simplified terms, it states that a camera measures

$$Q_e = L_e \; A \; \Omega \; t. \tag{2.1}$$

The measured radiant energy is converted into a voltage, digitized and eventually stored as a pixel value.
In high dynamic range imaging, pixel values are converted back into real-world radiance. The exact values for the pixel surface area and the solid angle of the aperture are generally unknown, but assumed to be constant. Without prior calibration, the real-world values obtained by inverting the image formation process thus only represent radiance up to an unknown scale. This is sufficient for our purpose. Yet, it must be kept in mind that when "radiance" is mentioned in this thesis, it refers to a quantity that is merely proportional to radiometric radiance where the scaling factor is unknown. Furthermore, if the aperture is fixed, irradiance is proportional to radiance ($L_e\Omega = E_e$). Since we are only interested in radiance up to an unknown scale factor, these two quantities are used interchangeably unless their distinction is crucial in the context.
The quantities of radiometry presented so far have the subscript e to denote that they are

radiometric values. They may be measured for each wavelength of the electromagnetic spectrum individually. In that case, the additional subscript λ specifies the wavelength. For example, the total irradiance E_e on a surface may be calculated by summing up all wavelengths λ of light incident to a surface. Since irradiance for each wavelength $E_{e,\lambda}$ is a continuous function, this is expressed as an integral:

$$E_e = \int E_{e,\lambda} \, \mathrm{d}\lambda. \tag{2.2}$$

The human eye is only sensitive to a small range of wavelengths called visible light. The sensitivity also strongly varies with wavelength. It is blind to electromagnetic waves outside the range of approximately 400 to 700 nanometers, with a peak sensitivity at 555 nm (green light). This is expressed accurately by the luminosity function shown in Figure 2.1 which was standardized by the Commission Internationale de l'Eclairage (CIE). It represents the spectral sensitivity $V(\lambda)$ to light of a certain wavelength λ of a "standard observer" averaged over a large number of test subjects. Two surfaces having the same irradiance calculated over the entire spectrum thus do not necessarily *appear* to have the same brightness to a human observer. Due to the varying spectral sensitivity, a surface lit by blue light appears darker than one lit by green light of the same radiant power. In order to estimate the perceived brightness of a surface, the spectral irradiance $E_{e,\lambda}$ needs to be weighted with the sensitivity function $V(\lambda)$. This perceived brightness of a surface is called *illuminance* (E_v). The subscript v denotes that it is photometrically weighted. It is computed as

$$E_v = \int_{380}^{830} E_{e,\lambda} V(\lambda) \mathrm{d}\lambda. \tag{2.3}$$

In the same way, each radiometric quantity can be weighted to obtain a corresponding photometric value which expresses the same concept, but specific to human perception. One of the basic units in photometry is the lumen (lm) expressing *luminous power*. It corresponds to radiant power. Illuminance is derived from luminous power and has the unit $lm/m^2 = lx$ (lux). Illuminance is the quantity that most closely resembles the concept of subjective brightness. The photometric equivalent of radiance is *luminance*, which is luminous power per unit area arriving from unit solid angle. Its unit is $\frac{lm}{m^2 sr}$, which is equivalent to the more common unit candela per square meter cd/m^2 ($lm/sr = cd$).

It should be noted that imaging sensors have a sensitivity that varies with wavelength, too. Their sensitivity curve generally differs from the $V(\lambda)$ curve of the CIE human standard observer and is often supplied in the documentation. Hence, a sensor also measures spectrally weighted radiance, similar to the concept of luminance. However, this circumstance is mostly ignored in the literature on HDR imaging, and uniform sensitivity is assumed so that radiometric radiance can be measured.

A central concept used throughout this thesis is that of *dynamic range*. It is the ratio between the highest and the lowest value a signal can take on. Although it is defined for many types of signals like sound or voltage levels, we exclusively use the term in the context of light. As a ratio, the dynamic range is unitless. It is often expressed in

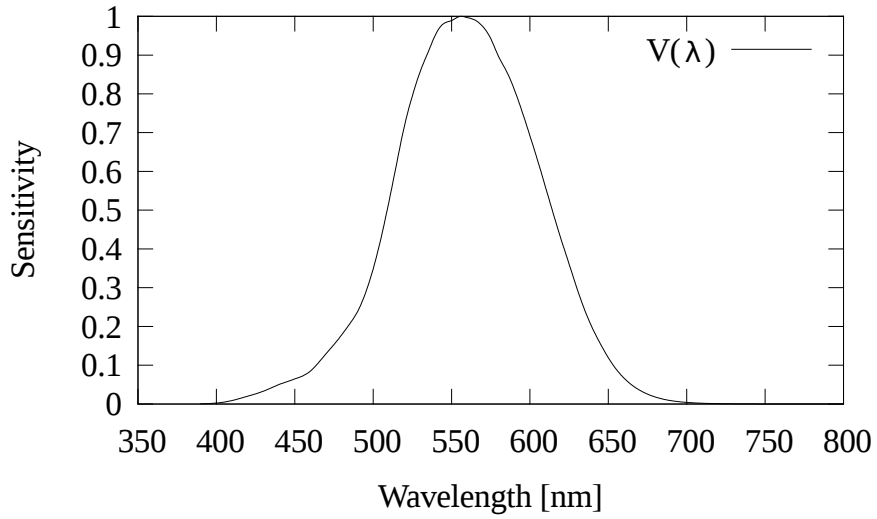


Figure 2.1: *Relative sensitivity $V(\lambda)$ of the human eye to light of a specific wavelength λ. This luminosity function is standardized by the CIE.*

decibel (dB), as orders of magnitude (i.e., powers of ten) or in stops (i.e., powers of two). When decibel are used to express the dynamic range of radiance values, the following convention is used:

$$DR_{dB} = 20 \log_{10} \left(\frac{L_{e,max}}{L_{e,min}} \right). \tag{2.4}$$

Dynamic range can be determined for the radiance values present in a scene, the capturable range of a camera, or a display output range. As an example, an LCD screen with a dynamic range of 1000:1 spans three orders of magnitude of radiance levels, which corresponds to 60 dB or roughly 10 stops. It is important to not mistake the contrast ratio of a television set given by the manufacturer for true dynamic range. Such values are often inflated by employing non-standardized measurement methods for marketing purposes.

Dynamic range is often erroneously represented as a number of bits. This representation confuses the ratio of radiance levels, i.e., the total range of physical values, with the number of quantization levels used for digitization of the range. Nevertheless, it is often desirable to store high dynamic range values with a higher number of bits. Using three 16 or 32-bit floating point values to store the red, green and blue component of an HDR pixel or storing 8 bits for the mantissa of each component and 8 bits for a common exponent are typical data formats. See the *Radiance Picture Format* introduced in [122] and the *OpenEXR* file format [14] for details.

2.1.2 Representing Color

What we perceive as color is actually the wavelength of light on a physical level. For instance, a wavelength of 460 nm is perceived as blue, 630 nm as red, and so on. In practice, a real stimulus rarely consists of light of only one wavelength, but is a mix of

many. A stimulus can be represented mathematically as a continuous function of radiant energy versus wavelength. The color appearance of a stimulus is then dominated by the wavelengths with the most energy.
The plot of the luminous function in Figure 2.1 only shows the sensitivity of the human eye to brightness. The human eye actually possesses three types of cone cells in the retina that are susceptible to light. Each type has its peak sensitivity in a different wavelength range. These ranges roughly correspond to the colors red, green and blue. The three sensitivity functions $r(\lambda)$, $g(\lambda)$, and $b(\lambda)$ are shown in Figure 2.2. Our brain can "guess" the spectral energy distribution Q_λ of a light stimulus based on the three scalar responses R, G, and B it receives from the three cone cell types. Mathematically, a cone cell's response to a given stimulus Q_λ is calculated as the inner product between the stimulus and the sensitivity function:

$$R = \int_{380}^{830} Q_\lambda r(\lambda) \mathrm{d}\lambda \tag{2.5}$$

$$G = \int_{380}^{830} Q_\lambda g(\lambda) \mathrm{d}\lambda \tag{2.6}$$

$$B = \int_{380}^{830} Q_\lambda b(\lambda) \mathrm{d}\lambda \tag{2.7}$$

The original stimulus can then be approximated from the response triplet (R, G, B), the so-called *tristimulus value* of Q by a linear combination of the sensitivity functions with the responses as coefficients:

$$Q_\lambda \approx r(\lambda)R + g(\lambda)G + b(\lambda)B \tag{2.8}$$

The human visual system estimates spectral power distributions from the responses of cone cells that are sensitive to red, green and blue light. This is the reason why RGB is the dominant color space to represent colors in devices that capture or reproduce images. Despite the RGB color space being widely in use, a number of other color spaces exist that are more suited for specific purposes. One notable example is the XYZ color space. It is defined according to three artificial sensitivity functions $x(\lambda)$, $y(\lambda)$, and $z(\lambda)$ that differ from those of the retina. A light stimulus is now matched by a linear combination with the coefficients (X, Y, Z). The sensitivity functions were defined in a way such that the functions and the resulting tristimulus have a number of beneficial properties. One of them is that the function $y(\lambda)$ is identical to the spectral sensitivity $V(\lambda)$ of the standard observer (see Figure 2.1). Calculating the Y coefficient for a stimulus is thus equivalent to photometric weighting (see Equation 2.3). This means that Y always represents the perceived brightness of the stimulus. Another useful attribute of the XYZ color space is the fact that a stimulus with unit radiant power over all wavelengths (perfect white) is mapped to a tristimulus of $(1, 1, 1)$.
Given the coefficients (R, G, B), the equivalent tristimulus in the XYZ color space can be calculated by multiplying it with a conversion matrix.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.9}$$

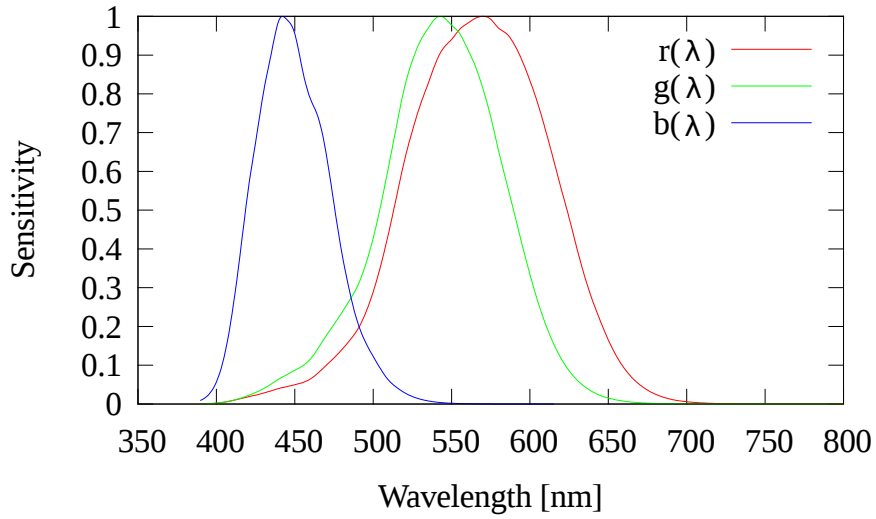


Figure 2.2: *Spectral sensitivities $r(\lambda)$, $g(\lambda)$, and $b(\lambda)$ of the three types of cone cells in the human eye. The peak sensitivities correspond to the colors red, green, and blue respectively.*

The coefficients of this matrix can be found in [59].
Most of the processing done in HDR imaging is only concerned with changing brightness levels while the color is kept the same. This gives rise to a color space that – unlike XYZ – explicitly separates color from brightness information. One such color space is the Yxy space, which is the one used in the HDR video system presented in this thesis. It consists of one luminance (Y) and two chrominance channels (x and y) that are independent of luminance. A color representation in Yxy can be derived directly from the triplet (X, Y, Z) of a stimulus. The Y component already represents perceived brightness, so it is the same in both color spaces. x and y can be calculated according to the following formula:

$$x = \frac{X}{X + Y + Z} \tag{2.10}$$

$$y = \frac{Y}{X + Y + Z} \tag{2.11}$$

Unlike cone cells in the human retina, the pixels of a charge-coupled device (CCD) sensor are color blind. They only have a single spectral response curve and thus only yield a single response coefficient corresponding to brightness. This makes them unable to distinguish between the wavelength distributions of two different light stimuli. A common workaround for this problem is to place a color filter array over the sensor surface. The filters only allow the transmission of light of a certain wavelength range for each pixel. The wavelengths are chosen to match the human eye, that is, red, green, and blue light. Each pixel underneath the filter then either measures the red, green, or blue component of the incoming light. This is conceptually identical to capturing three separate images, one for each color channel, at one third of the resolution and intensity.

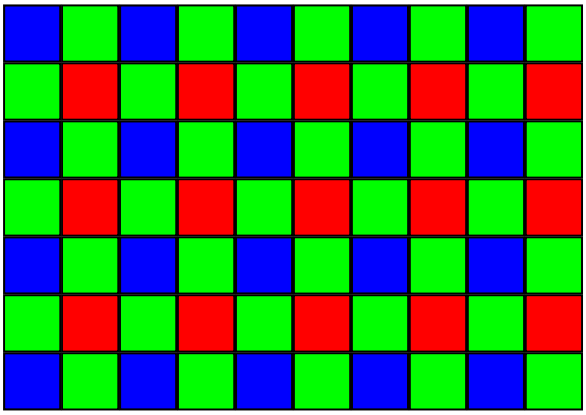


Figure 2.3: *A Bayer color filter array. Placing this pattern over a CCD sensor makes the pixels sensitive to only a certain wavelength range while everything else is filtered out. A pixel then only measures either red, green, or blue light. Full RGB values for each pixel are reconstructed by interpolation.*

Interpolation then allows the reconstruction of a full resolution RGB image. Figure 2.3 shows a specific color filter array that is widely used in practice. It is called *Bayer filter*. 50% of the pixels capture green, 25% red and 25% blue light. This imbalance mimics the human visual system's increased sensitivity to green light. As an example, the red $R_{x,y}$, green $G_{x,y}$, and blue $B_{x,y}$ components of a red pixel at position (x, y) can be obtained in the following way:

$$\begin{aligned} R_{x,y} &= R_{x,y} \\ G_{x,y} &= \frac{G_{x-1,y} + G_{x+1,y} + G_{x,y-1} + G_{x,y+1}}{4} \\ B_{x,y} &= \frac{B_{x-1,y-1} + B_{x+1,y-1} + B_{x+1,y+1} + B_{x-1,y+1}}{4} \end{aligned}$$

2.2 The HDR Pipeline

The process of creating a frame in an HDR video can be thought of as a pipeline consisting of the following steps: Setting camera parameters, capturing, color conversion, image registration, HDR stitching, tone mapping and display. The steps must be performed in order, and the output of each step is the input to the subsequent one. Statistical information gathered during the creation of one frame may then be used to set the camera parameters for the next to restart the pipeline. The HDR pipeline is depicted in Figure 2.4.

The *capturing* of low dynamic range (LDR) images constitutes the first step. In a situation where the dynamic range of the scene exceeds the one of the sensor, a single image cannot represent the scene with sufficient accuracy. It is thus required to obtain

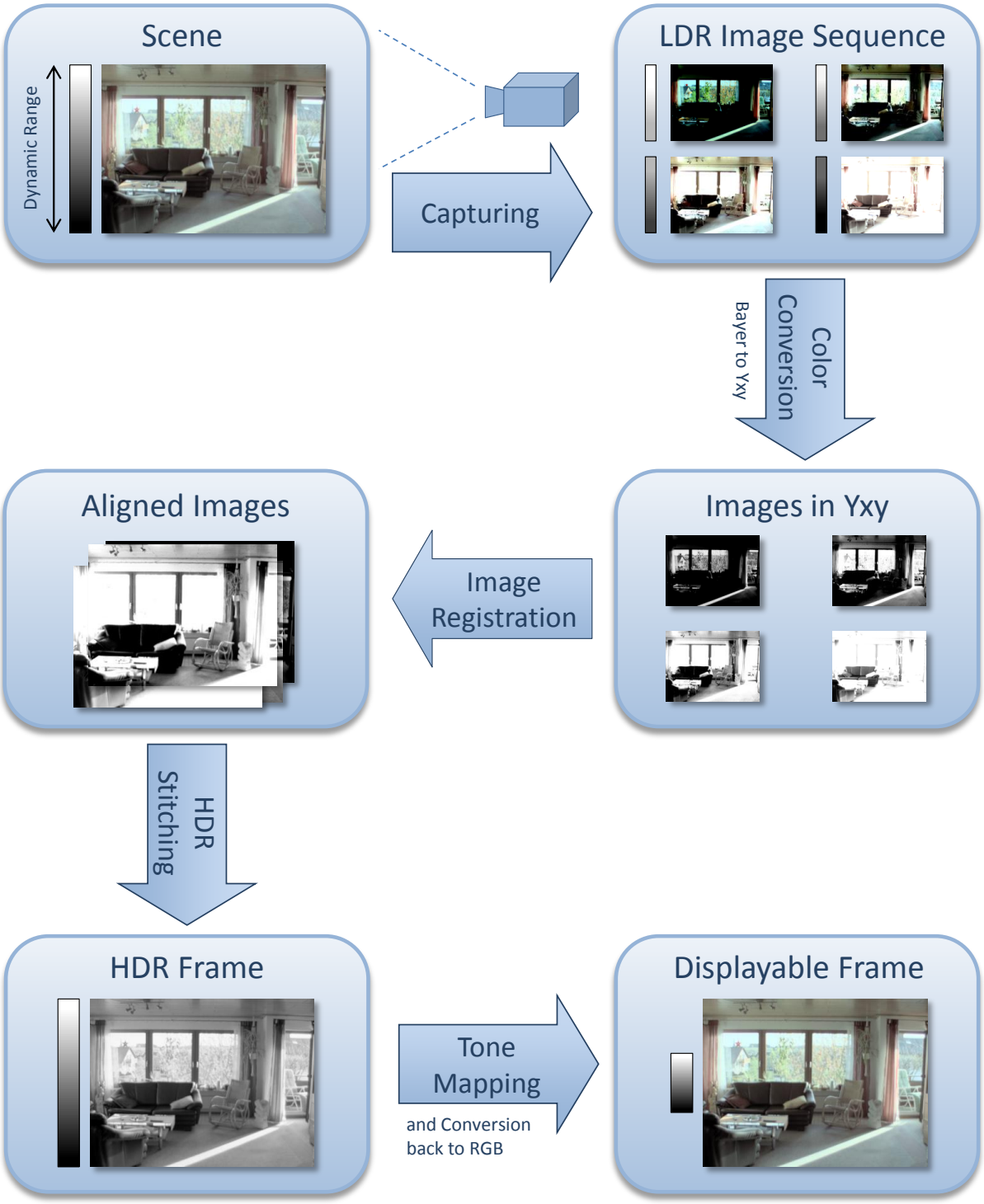


Figure 2.4: *The process of creating an HDR frame is a pipeline. It begins by capturing multiple images of the scene. Each spans a different dynamic range. They are then converted into the Yxy color space, registered and stitched into an HDR frame. The dynamic range of the HDR frame is compressed by tone mapping for display.*

multiple exposures of the same scene, each covering a different radiance range. In our work, we call this set of LDR exposures an *image sequence*. We exclusively use the term in this context, and it is not to be confused with a sequence of HDR frames, which we call *HDR video*. When creating an image sequence, the exposure values (e.g., shutter speed, gain or sometimes aperture) must be chosen carefully. The simplest approach is using a set of predefined values that vary by a constant factor. In photography, the factor is usually expressed as a number of stops, which is a power of two. A more sophisticated method would consider the minimum, maximum or average image brightness of the scene or even its entire histogram to determine the exposure settings. It is also possible to choose the parameters dynamically during image acquisition. A base image is captured using default parameters and then analyzed. The analysis triggers the capturing of more images if necessary which are analyzed in turn. This process ends when the full dynamic range of the scene is covered.

The output of the capturing process is a set of LDR images captured at different points in time using different exposure values. They are single channel color images obtained from a sensor equipped with a color filter array. For easier processing, they are converted into a different color format. The Yxy color space is suitable for this purpose, because it separates a pixel's brightness from the color information. Most of the subsequent steps only work on the brightness channel while leaving colors unchanged.

If the camera is moved or the scene is dynamic, the images of the sequence do not show exactly the same content. In order to establish pixel correspondence within the sequence, the intermediate motion needs to be estimated and compensated. This process is called *image registration*. There are two types of motion that can occur between two shots. The first type is camera motion like pans, tilts and zooms which affect the entire image. It can be globally estimated from corresponding points in the sequence. The second type is motion of individual objects visible in the scene like cars or people. Estimating it requires a more complex analysis of the scene and estimating a per-pixel flow. The real-time constraints in the HDR video scenario rather suggest using a global motion model that is fast to compute. The output of the image registration step is the original sequence annotated with motion parameters for each exposure.

For each point in the scene, a set of corresponding pixels can be found in the registered image sequence. Each pixel was taken under a different exposure setting, but all originate from the same radiance value in the scene. The pixel values are noisy measurements of the physical radiance under various conditions. As a rule of thumb, the brighter a pixel appears without being saturated, the more accurately it measures radiance. This is because a large portion of the image noise has a constant magnitude, which leads to a higher signal-to-noise ratio for higher pixel values. The real radiance can be reconstructed from a weighted average over all estimations, that is, all corresponding pixels. Doing this for every point in the scene yields a full map of physical radiance values in the scene. This radiance map is the resulting high dynamic range video frame.

In most cases, the radiance values contained in the HDR frame are not accurately displayable on a regular screen. This becomes apparent when considering an HDR image of a very bright spotlight. The comparably weak backlight of an LCD screen is most likely unable to create the same amount of brightness. What is more, linearly scaling

the radiance map to the output range of the display yields unsatisfying results when the dynamic range of the frame is much higher than that of the screen. It is thus desirable to create a customized mapping between scene radiance and display pixel values. This must be done in a way that keeps the visual appearance of the image on the screen similar to the appearance of the real scene. This process is called *tone mapping* and is the last step of the HDR pipeline before displaying the result.

2.3 Image Capture

The scenario assumed so far is that of capturing a video – or more specifically a single video frame – of a scene which has a higher dynamic range than the camera. No matter how the exposure parameters are set, there is always some under- or overexposure present in the acquired image material. This section introduces techniques to overcome the dynamic range limitation of an imaging sensor. Most of them revolve around taking several LDR images while varying the exposure parameters. They mainly differ in *how* the images are obtained. Each LDR image has different pixels well exposed. In the later processing, only the "good" pixels are kept and the others are discarded. As long as the exposure parameters are suitably chosen, at least one usable pixel is available for each point in the scene.
The exposure of an image can be controlled in a number of ways. Adjusting the *shutter speed* (also: "exposure time") of a camera controls the time period over which its image sensor is susceptible to incident light. The longer the shutter value is chosen, the more light can enter the sensor cells, and the brighter the image appears. Long shutter speeds can slow down the overall capturing process and increase the effect of motion blur. Setting a *gain value* adjusts the amplification of the A/D converter that digitizes the voltage in a sensor cell. Increasing gain makes the image brighter without increasing capture time or adding motion blur. However, it comes at the cost of simultaneously amplifying image noise. To reduce image noise at the expense of increased motion blur, it is often sensible to pursue a shutter before gain policy, that is to first increase the shutter value up to its maximum before increasing gain to obtain a brighter picture. Both shutter and gain are controlled electronically and can thus be adjusted per-frame. A more static way to control exposure is manually changing the *aperture* of the lens or using a lens with different maximum aperture. Alternatively, so-called neutral density (ND) filters can be placed in front of the lens to attenuate the incoming light. They are called "neutral density", because they ideally attenuate all wavelengths equally.

2.3.1 Temporal Exposure Bracketing

The most commonly used approach to capture LDR image sequences is temporal exposure bracketing. It is also the approach we follow in our work. Multiple pictures are taken in quick succession using a single camera. The exposure parameters for each image are controlled by a PC to which the camera is connected, e.g., over a FireWire bus. It is only feasible to adjust the dynamic parameters shutter speed and gain. They are transmitted

to the camera either as a list of parameters for an exposure sequence or individually. If an entire sequence of parameters is transmitted, the camera loops through them while continually capturing image sequences and sending them back to the PC. The advantage of this mode of operation is that images can be captured by the camera and processed by the PC asynchronously. On the other hand, triggering each shot separately gives more fine-grained control over the parameters at the cost of dependency between capturing and processing.

Taking multiple pictures at different points in time to create a single HDR frame has obvious disadvantages. The most prominent one is the increase of overall capture time. The frame rate of the HDR video is only a fraction of the camera's capturing rate. To overcome this disadvantage, a fast camera must be used. Another issue is scene or camera motion between the shots. Images of the sequence no longer show the same content, and motion compensation becomes necessary. The impact of motion artifacts like ghosting is directly related to the camera's frame rate. The less time the camera takes to capture the sequence, the less ghosting occurs. Despite these drawbacks, the low hardware requirements and costs, the easy setup and the high flexibility in choosing exposure parameters make it a viable approach.

Examples where temporal exposure bracketing is used as a sub-part of an HDR image or video creation technique are [17, 75, 76, 24, 82, 97, 60, 112, 88, 46].

2.3.2 Beam Splitting Devices

Another way of capturing an exposure sequence for an HDR frame is using multiple cameras and a beam splitter. The incoming light of the scene is split up and redirected towards the lenses of the cameras. Each camera sees the same image, but only at a fraction of the original intensity. They are all connected to the same control unit and synchronized, so that an entire sequence of images can be captured at once. The exposure of each camera is adjusted individually so that the resulting sequence can be used to create an HDR image. Here, it is possible to employ a combination of static and dynamic exposure control. ND filters and different apertures can be used to control the exposure offset between the cameras. Shutter or gain are then adjusted either individually for each camera or globally to adapt to the brightness level of the scene.

The advantage of beam splitting capturing devices is that all images are taken at exactly the same point in time under an identical perspective. This saves capturing time over temporal bracketing, and no image registration is required. However, a beam splitting apparatus is more expensive and more difficult to set up and calibrate than a single camera. It is also more restricted in the number of exposures that are captured and the choice of exposure values. Because the incoming light is divided among the imaging sensors, more light sensitivity is required for each sensor. This necessitates more costly lenses, more amplification or slower shutter speeds.

Publications that specifically use beam splitters to capture exposure sequences for HDR include [57, 3, 125, 115].

2.3.3 Finding Good Exposure Values

An open question that is common to all multiple exposure capturing techniques is how to set the exposure values for the given scene lighting and how many images to take. In this context, only shutter speed and gain are considered, because they offer the best adaptability.

At its core, HDR imaging is about reducing image noise, that is, to measure scene radiance as accurately as possible. The accuracy is bounded by the limited dynamic range of the imaging sensor. At the upper boundary, there is saturation of a sensor cell, i.e., an image pixel. Above a certain brightness level, more photons arriving at the cell no longer increase its charge, so that measuring the exact radiance becomes impossible. Near the lower end of the dynamic range, the charge induced by incident light is indistinguishable from noise in the circuitry. A large portion of the image noise (e.g., quantization noise, fixed pattern noise) is independent of the amount of light falling onto the pixel. As a result, radiance is best measured in the upper segment of the sensor cell's operating range just below saturation. Here, the sensor achieves the best ratio between the image signal and readout noise. So the goal is to control exposure in a way such that every radiance range occurring in the scene is measured with a high signal-to-noise ratio (SNR) in at least one of the images. Scene statistics like the minimum/maximum scene radiance or the histogram of scene radiance [49] may be used as a basis to determine exposure values.

As a general rule, averaging the signal from a higher number of images means less noise in the resulting radiance map. However, as seen above, not all images contribute equally well to the measurement of radiance (consider saturated or much too dark images). On the other hand, capturing fewer images means saving capture time, so a compromise has to be found. The problem is thus one of finding a set of exposure values that get the most out of a certain number of images. This has been addressed in several published works.

Barakat et al. [9] focus entirely on minimizing the number of exposures while covering the entire dynamic range of the scene. Minimum and maximum of the scene's irradiance range are taken into account, and the least possible overlap of exposures is chosen. They do not consider the SNR of the HDR result during the choice of exposure times, that is, each pixel is considered to contribute the same amount to the result regardless of its value. The algorithm is a fast heuristic suitable for real-time use.

In [19], the authors use a model of shot noise to determine the sequence of shutter values producing the highest SNR for a given number of exposures. The shutter speeds are obtained by solving a constrained optimization problem. For this purpose, a coarse approximation of the scene irradiance histogram is used. The authors always use the brightest pixel before saturation.

An approach to emulate an effective camera with a given response function and dynamic range was published in [42]. In an offline process, a static table of exposure times is created that spans the desired dynamic range. The static table prevents adaptation to changes in scene brightness distribution, for example when large reflective surfaces like cars appear.

A very recent method to determine noise-optimal exposure settings uses varying gain levels [53]. For a given maximum sum of exposure times, increasing gain also increases the SNR. The authors define SNR as a function over log radiance values. Only the extrema of the scene's brightness are considered. Computation of the exposure settings is too expensive to be suitable in a real-time scenario.
The authors of [54] developed a theoretical model for photons arriving at a sensor pixel by estimating the parameters of a Gamma distribution. From the model, exposure values are chosen that maximize a criterion for recoverability of the radiance map. The focus lies on the impact of saturated pixels on the HDR result.
In [58], an algorithm for estimating optimal exposure parameters from a single image is presented. The brightness of saturated pixels is estimated from the unsaturated surroundings. Using this estimation, the expected quality of the rendered HDR image for a given exposure time is calculated. The exposures leading to the lowest rendering error are chosen.

2.3.4 Direct HDR Capture

Instead of obtaining multiple exposures from a low dynamic range camera and combining them in software, it is also possible to increase the dynamic range of the sensor. Doing this increases the hardware cost, but allows direct capturing of HDR images. The dynamic range achieved by hardware solutions ranks somewhere between the four orders of magnitude of a high quality LDR sensor and the nearly arbitrarily high DR of multiple exposure approaches.
A CCD image sensor works in two phases: integration and read-out. During integration, photons are collected and stored as electrons in the sensor cells. The length of this phase is controlled by the shutter speed. In the read-out phase, the electronic charge is transported out of the sensor and converted into a voltage by a charge amplifier. The voltage is then quantized by an A/D converter and further processed.
The most effective approach to increasing a sensor's DR is similar to capturing multiple exposures, but on the sensor level [2, 72, 78, 126, 127]. The charge that is accumulated in a cell is read out multiple times during integration in a non-destructive way. Each reading then corresponds to a different exposure time. Alternatively, the charge can be read out and reset multiple times (destructive reading) while remembering the number of resets. This prevents saturation of the pixel. The readings obtained in this way are then combined in hardware into a single value per pixel, resulting in an image with increased dynamic range. As for temporal exposure bracketing in software, motion between the read-outs remains an issue.
Another possibility to increase a sensor's dynamic range is trading off spatial resolution for dynamic range. The authors of [86] place an array of ND filters over a high resolution CCD sensor, similar to the Bayer pattern described earlier. Each pixel in a 2×2 block is attenuated differently, leading to varying sensitivity to light. This is comparable to simultaneously capturing four differently exposed images at one fourth of the original resolution. Capturing simultaneously means that no image registration is required. However, as with all approaches that work with ND filters, a large portion

of the incoming light is blocked and lost for measuring. This approach is extended to filters with dynamically changeable density in [85].
A sensor design that differs from the multiple exposure techniques introduced so far was presented in [15] and [22]. Instead of measuring charge accumulated in a set amount of time, the sensor measures the time it takes for a cell to saturate. The time to saturation is roughly inversely proportional to the radiance. Since the full capacity of a sensor cell is used to measure this quantity, it is operated near the point of maximum SNR. The highest radiance that can be detected depends on the resolution of the internal clock. The lowest detectable radiance is limited by the maximum time to wait for saturation. This time span corresponds to a shutter speed where even the darkest pixel in an image would saturate.
Photocurrent induced by incoming light can be directly converted into a voltage instead of integrating charge in a sensor cell. This is done in [61]. Logarithmic compression during conversion gives the sensor a logarithmic response to light. This is an effect similar to how the human eye perceives brightness. It achieves a natural extension of the dynamic range. However, the compressed signal is very sensitive to noise introduced in the later processing steps, and expensive high-precision circuitry is required. The achieved SNR is actually worse than that of a regular sensor under low illumination.

2.4 Image Registration

The presented approaches for temporal exposure bracketing or multiple sensor readout sequentially capture image data with varying exposure parameters. For all these approaches, motion that takes place between the shots is a problem and must be compensated using image registration techniques. This section gives an overview of existing techniques in general and those specific to the creation of HDR images and video in particular.
Image registration – also called "image alignment" – is the process of establishing correspondences between the pixels of two or more images. The goal is for each pixel in a source image to find a corresponding pixel in the target image which represents the same point in the scene. In the following, this is described for two images at a time. Multiple images can be registered with respect to one chosen reference image or by aligning each image with its predecessor.
Images can be taken under different viewing angles (e.g., to create panoramic images), at different points in time (HDR from multiple exposures or long term medical studies), or by different imaging sensors (multimodal image fusion). Combining them into a single image without prior registration means merging images that do not show the same content. This results in artifacts like ghosting around the borders of moving objects or motion blur. In the context of HDR video, we are mostly concerned with merging images taken in quick sequence from a slightly changing viewpoint. The difference between these images that must be compensated is thus mainly due to motion that occurred in between the shots – as opposed to changing lighting conditions or reflectance properties. Note that the different brightness levels of the images are a intentional. They are not balanced

during image registration but utilized during the following step of HDR stitching.

There exist a number of specialized image registration techniques for a multitude of application scenarios. Each scenario brings different challenges: When creating a panoramic image, the offsets in the source images are usually very large, and the overlap between them is small. The source images are taken under arbitrary conditions. However, since the output is meant to be viewed by humans, registration errors are not critical. This changes when the source images capture parts of a printed circuit assembly (PCA) and the merged image of the entire board is used for automatic optical inspection (AOI) [47]. Here, a high degree of accuracy is required, but a large amount of prior information is available from the strictly controlled capturing conditions. In medical imaging, the images are often taken by devices with strongly varying characteristics, like x-ray computed tomography and magnetic resonance imaging. Such images may show the same content, but in an entirely different way. They may also be acquired over the course of several months. The challenges specific to the HDR video scenario are the large difference in image exposures making it difficult to detect correspondences, and achieving computability in real-time. On the positive side, knowledge obtained from previous frames in the video can be exploited, and when capturing at high frame-rates, motion has only little impact on the images.

Motion occurring while capturing image sequences can be classified into two categories: global camera motion and local object motion. Even though motion in practice is a mix of the two types, they can each be treated separately. When the camera is kept static, only motion of the objects in the scene remains. There may be walking people, moving cars, or trees moving in the wind. Object motion is generally articulated, deformable, and very difficult to model as it may come from a plenitude of sources. However, its influence is limited to a local image area. Camera motion like pans and zooms, on the other hand, have an impact on the entire image at once. This type of motion is easily modeled by a small number of camera parameters. Mathematically, camera motion can be expressed by a linear transformation between the coordinate systems of the two considered images. Transforming a 2D pixel position from one image to its corresponding position in the other image is done by multiplying the position vector by a transformation matrix. Depending on the type of motion assumed, the transformation matrix has a different number of degrees of freedom. Types of motion are: translation, rotation, scaling, affine transformation and perspective transformation. If a camera with a large focal length and a high frame rate is used, the camera motion between two frames can be approximated by simple transformations like pure translation.

It is common to distinguish between two classes of image registration. They are *feature-based* and *intensity-based* (or "dense") registration and are discussed individually in the following two sections. Methods in the former class first detect sets of notable points – so-called *feature points* – that can be found in both images. A global transformation is then estimated from the movement of the points between the images. Feature-based approaches are best suited for deriving camera motion. They allow for large amounts of motion, are fast to compute and robust to a certain amount of exposure change. Intensity-based techniques, on the other hand, work directly on pixel values. They compute a dense motion field with individual motion parameters for each block of pixels

or even each single pixel. Such motion is costly to compute, but achieves higher accuracy. It is well suited for local object motion. In practice, methods from both classes can be combined by first performing a feature-based global registration and then a pixel-based refinement.
When object motion is not completely eliminated before merging images together, a type of artifact called *ghosting* may appear in the final result. It is caused by a moving object that is visible in more than one of the images, but located in a different position in each of them. Without explicit ghost removal, the object then appears multiple times in the fused image. Ghost removal can either be used to reduce the visibility of misalignment after motion compensation or can replace object motion compensation completely. Ghost removal approaches are discussed in Section 2.4.3.

2.4.1 Feature-based Registration

Feature-based registration is image alignment based on a set of interest points detected in two images. It is performed in three separate steps: First, such points are detected individually for each image. Then, correspondences between the points of the two images are found. And lastly, motion parameters are estimated from the point correspondences. Techniques used for each of the steps are introduced in the following.

Detecting Feature Points

Good feature points for registration should be recognizable even under changed imaging conditions, and their position should be determinable accurately. The latter requirement is obviously not met by unstructured, uniform areas. Even lines or edges only allow for detecting motion perpendicular to their direction. Examples for good features are unevenly textured areas, corners or T intersections. The decision whether an image area is suited as a feature point is made locally on the pixel level.
One of the earliest automatic feature point detectors was published by Moravec in 1980 [84]. It is based on the assumption that interesting points exhibit strong intensity variation in all directions. This is the case for corners, but not for lines which do not vary along the edges. The detector considers pixel intensity differences in four directions in a local window around each pixel position. The directions are: horizontal, vertical and the two diagonals. Absolute intensity differences in these directions are summed up over the entire window, yielding four values. Since good feature points should show variation in all four directions, the pixel position is classified as a feature point if the *minimum* of the four values exceeds a threshold.
The approaches published by Harris and Stephens [52] and by Shi and Tomasi [103] are similar to each other in their working. Both define a virtual translational motion model. For each pixel in an image, they set up a linear equation system with the translational shift vector as free variable. The coefficients of the equation system contain values that are derived from the image gradient. They are identical to the Hessian matrix at that position. Solving the equations would yield an optimal translation vector. However, the two approaches are only concerned with determining how suitable a location is for

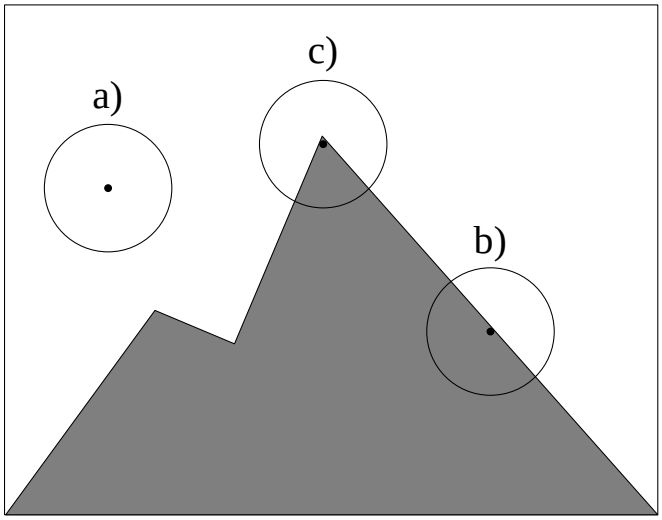


Figure 2.5: *The SUSAN corner detector inspects a circular area around a pixel. The lower the fraction of pixels in the circle with the same color as the center, the more likely it is to be a corner.*

deriving a motion vector from it. Good features to track are those locations that allow a stable solution of the linear equation system. They thus test how well-conditioned the Hessian matrix is. In [103], a feature point is a location where the minimum of the two eigenvalues of the matrix are both above a threshold. This is equivalent to saying that the gradients in the window around the location are high and have more than one preferred direction, i.e., the window contains a corner. [52] differs in that it uses a different thresholding mechanism for the two eigenvalues and a more efficient calculation. In [104], the SUSAN corner detector was published, which stands for "Smallest Univalue Segment Assimilating Nucleus". It considers sizes of areas of similar color. In order to decide if a pixel position constitutes a feature point, it inspects a local circular area around the pixel. It counts the number of pixels in the circle that have the same color as the center pixel. If this number is close to the full area of the circle, then the pixel must lie in a homogeneous image region and is thus not a good feature point (see Figure 2.5 a). For pixels lying on an edge, the neighboring pixels with the same color make up approximately half the circle's area (b). Only when the fraction of pixels with the same color as the center pixel is minimal, it is likely to be located in a highly textured region or at a corner and is thus chosen as a feature point (c).

Finding Correspondences

The result of the feature point detection step is two sets of feature positions, one for each of the two images to be registered. It is now necessary to find correspondences between the points in the two sets. A corresponding point pair consists of two feature points that represent the same point in the scene. The two images were taken under similar, but not identical conditions. If their exposure differs, a structured image area in one image may

become saturated in the other. Object motion between the shots may cause objects to be no longer visible, partially occluded or deformed. As a consequence, the images do not contain the exact same feature points, and matching becomes difficult.
The problem of finding correspondences can be subdivided into defining a suitable metric for comparing feature points and deciding which points to compare. A feature point can be characterized by using the intensity of the pixels in a local surrounding, or by values derived from it. Examples for derived values are gradient intensity and orientation, texture parameters, or values determined during feature detection [73, 34, 99, 114]. The list of values characteristic to a feature point obtained in this way is called a *feature descriptor* ([80] gives a good overview of existing descriptors). Good metrics for comparing the descriptors of two feature points are the mean squared difference and the cross correlation. Both metrics may be normalized in order to be robust to brightness variation.
In a simple case with only few detected features in each image, a brute force comparison between the feature point sets may be a feasible approach. The chosen metric is evaluated for the descriptors of every possible feature pair between the two sets. The best matching pair is added to the list of correspondences and removed from the point sets. This is repeated until one of the sets becomes empty or the score of the remaining matches falls below a threshold.
In practice, feature descriptors are elements of a high-dimensional vector space and a large number of features may be found. It is thus too costly to compute a matching score for all possible combinations. Instead, it is more efficient to use hashing [102], to organize the feature descriptors in data structures like k-dimensional trees [73, 98], or to use nearest neighbor search algorithms [87].

Estimating Motion Parameters

A feature point correspondence $(x_i, \hat{x}_i),\ i = 1, .., m$ describes the two pixel locations x_i and $\hat{x}_i$ which an interest point in the scene takes on in two different images. The difference in location is a result of motion occurring between the images. Since feature points are sparse in the image, feature-based registration techniques can only be used to estimate camera motion. So once a list of feature pairs has been established, a global transformation matrix A modeling the camera motion is estimated. It is the matrix that minimizes the mean squared distance between the points of the second image $\hat{x}_i$ and the transformed points of the first Ax_i. This is expressed by the following optimization problem:

$$\underset{A}{arg\,min} \sum_{i=1}^{m} (Ax_i - \hat{x}_i)^2 \tag{2.12}$$

Note that this notation implies homogeneous coordinates to represent x_i, and A is a 2×3 matrix. Minimization is done by taking the partial derivatives of the above term with respect to the coefficients of A and setting them to zero. This yields one equation for each degree of freedom (DoF) of the matrix. The minimum number of correspondences required thus depends on the DoF of A. Solving the linear equation system results in the optimal motion parameters between the two images.

The method described here assumes affine motion with up to six DoF. This includes translation, non-uniform scaling, rotation, and shear. If a full perspective transformation with eight degrees of freedom is assumed, the equation system derived from the optimization problem becomes non-linear. This is due to the fact that the scalar component of the homogeneous coordinates may now differ from 1, which necessitates perspective division. Such a non-linear equation system can be solved numerically with the Levenberg-Marquardt algorithm [70, 77]. It requires the computation of the Hessian matrix and partial derivatives with respect to the transformation parameters.
So far, we have assumed that the detected point pairs $(x_i, \hat{x}_i)$ are correct matches and represent a single feature point in the scene. This however is not always true in practice. Repetitive structures in images and deformable object motion can cause errors in the matching process, leading to false correspondences – so-called "outliers". They indicate false motion. If included during the estimation of the transformation matrix, outliers bias the result and lead to a registration mismatch. The negative effect of outliers can be alleviated by a probabilistic technique called *random sample consensus* (RANSAC) [32]. It randomly picks a small subset of the correspondences and uses it to estimate a transformation matrix $\tilde{A}$. Using $\tilde{A}$, the distance d between the feature points $\hat{x}_i$ in the second image and the transformed points $\tilde{A}x_i$ of the first can be calculated as $d(x_i, \hat{x}_i) = |\tilde{A}x_i - \hat{x}_i|$ for each other pair that was not used in the estimation of $\tilde{A}$. If d is smaller than a chosen threshold, the pair is considered to be an "inlier" that complies with the motion $\tilde{A}$. The number of inliers is stored, and a new transformation matrix is estimated from a different random subset of correspondences. This is repeated several times. In the end, the subset leading to the highest number of inliers and all compliant correspondences are used to calculate the final transformation matrix A. See Figure 2.6 for an illustration. The RANSAC approach leads to a correct estimate of motion with a certain probability, which strongly depends on the ratio between inliers and outliers in the correspondence set.
In addition to the individual methods described above, there exist complete systems for feature detection, efficient matching of high-dimensional feature descriptors, parameter estimation and outlier recognition. One prominent example is the scale-invariant feature transform (SIFT) introduced in [73]. It has been used specifically to register exposure sequences for HDR in [108].

2.4.2 Intensity-based Registration

The problem underlying intensity-based image registration is the same as in the feature-based case. Two pictures of the same scene were taken at different points in time and under slightly changed viewing conditions. The camera and scene motion that occurred in between must be estimated in order to establish pixel correspondences between the images. Rather than extracting feature points from the images, intensity-based methods compare pixel values directly. These values can be adapted to the different exposures of the images if necessary, they can be thresholded, or aggregated into various forms of histograms. Depending on the particular method that is used, a global image transformation as above or a dense motion field on a pixel level or pixel block level is estimated.

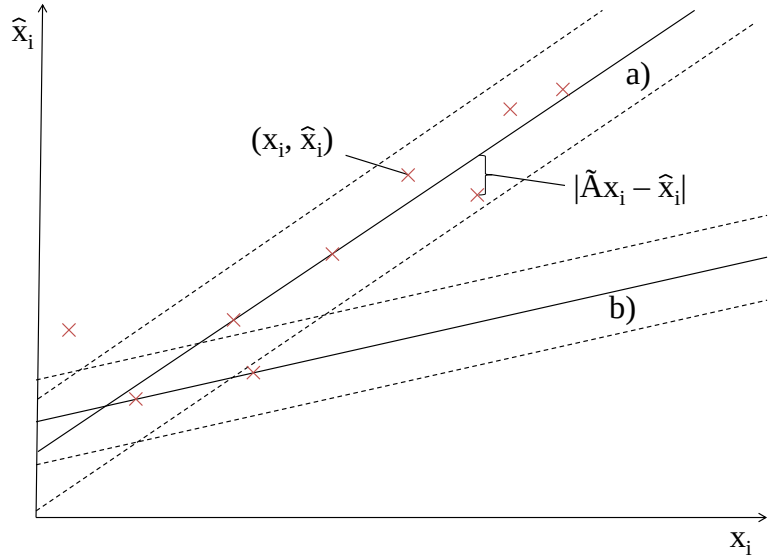


Figure 2.6: *Example for RANSAC with one-dimensional points. The transformation matrix A describes a straight line. a) and b) are two guesses $\tilde{A}$ based on two randomly chosen point pairs. More of the point pairs fall into the threshold distance of a). It is thus the better guess. All inliers to a) are then used to estimate a final matrix A.*

While a global transformation is fast to compute and suited for the real-time scenario considered in this thesis, it only allows for compensating camera motion. A dense motion field, on the other hand, is more costly to compute and only adequate for smaller amounts of motion, but achieves higher accuracy and provides a means of modeling object motion. Pixel block-based methods which are widely used in video compression make a compromise between the two by trading off estimation accuracy for computational efficiency.

Optical Flow

In general, the *optical flow* describes the movement of volume pixels along paths in three-dimensional space. In image processing, the optical flow is limited to the projection of the 3D motion to the 2D image plane. It is represented by a dense motion vector field, which typically assigns a translational motion vector to each pixel in the image. Mathematically, the intensity of a pixel in an image sequence is a function of position and time. The value of a pixel at position (x, y) at time t can thus be expressed by $I_t(x, y)$. If a pixel travels along the motion vector (u, v) between time t and $t+1$, it can be described by the following equation:

$$I_t(x, y) = I_{t+1}(x + u, y + v). \tag{2.13}$$

This equation means that radiance moves in a scene, but implicitly assumes that it remains constant from one frame to the next. The assumption is violated by the presence

of specular reflections whose intensity varies with the viewing angle and the angle of incident light – both are usually ignored in this context.
The intensity of the displaced pixel at time $t+1$ can be approximated from the original image by a first-order Taylor expansion ignoring the higher level terms

$$I_{t+1}(x+u, y+v) \approx I_t(x,y) + \nabla I_t(x,y) \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\partial I_t(x,y)}{\partial t}. \tag{2.14}$$

Here, $\nabla I_t(x,y)$ is the gradient vector of the image. Substituting equation 2.14 into 2.13 and subtracting $I_t(x,y)$ gives the *gradient constraint equation* [55]

$$\nabla I_t(x,y) \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\partial I_t(x,y)}{\partial t} = 0. \tag{2.15}$$

It relates the unknown motion vector (u,v) to the image gradient and the temporal derivative. The latter can be approximated by forward differences:

$$\frac{\partial I_t(x,y)}{\partial t} \approx I_{t+1}(x,y) - I_t(x,y) \tag{2.16}$$

The gradient constraint equation is one equation with two variables. It can therefore not be solved uniquely. In particular, the first term in the equation becomes zero whenever the motion vector is perpendicular to the image gradient, i.e., the motion is parallel to an edge in the image. Only motion along the gradient direction can be detected. This is known as the aperture problem which was first identified in [55]. It is alleviated by not only considering a single pixel position (x,y), but weighting pixels in a local window W centered around (x,y). Each pixel in the window provides one equation according to 2.15. Combining them leads to an overdetermined linear equation system in the variables u and v. It can be solved by least-squares optimization, that is, finding a solution (u,v) that minimizes the squared error averaged over all pixel positions (x',y') in the window W:

$$\underset{(u,v)}{arg\,min} \sum_{(x',y') \in W} g(x',y') \left(\nabla I_t(x',y') \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\partial I_t(x',y')}{\partial t} \right)^2 \tag{2.17}$$

g is a weighting function (e.g., Gaussian) that controls the support of the pixels in the window. The minimum can be found by computing the partial derivatives of the error with respect to u and v, setting them to zero and solving the resulting linear equation system. Repeating this for every pixel in the image produces a motion vector for every location with sufficient structure, i.e., where the equation system is well-conditioned.
Using the square in the above minimization problem makes it prone to outliers. In cases where a small number of pixels in W indicate greatly different parameters u and v, a bias is introduced into the estimation. There are many potential causes for outliers. Examples are:

- violation of the brightness constancy assumption by specular reflections or different imaging conditions,

- errors in the image gradient or temporal derivative due to noise,
- the presence of multiple simultaneous motions if W spans object boundaries or contains transparency or shadows.

To reduce the impact of outliers, two similar techniques were proposed at about the same time [7, 91]. They first use the least *median* of squares instead of the mean. This allows to find a motion vector (u, v) with which the majority of pixels in the window agree. All other pixels are discarded. Since the least median of squares only gives approximate motion parameters and behaves badly under Gaussian noise, it is only used to detect outliers. Accurate parameters are then obtained by solving the least-squares problem on the inliers only.
Black et al. [13] alleviate the outlier issue by reformulating the minimization problem in terms of more general error metrics ρ rather than using the square. The proposed error metrics are statistically more robust as they give limited weight to extreme values. The resulting optimization problem is no longer solvable by simply setting the partial derivatives to zero. Using ρ functions that are twice differentiable allows the use of iterative gradient descent to obtain good motion parameters.
Computing the flow at a pixel position from a local neighborhood has several drawbacks. The aperture problem makes the computation of good motion parameters unstable if the motion in the window is perpendicular to the gradient direction (see Equation 2.15). A similar problem arises when the local area is unstructured and the gradients are close to zero. Horn and Schunck [55] thus make the additional assumption that optical flow varies smoothly over the image. This allows the interpolation of flow in areas that are badly suited for direct computation. They came up with a global formulation of the optimization problem. Instead of determining a flow vector (u, v) for each pixel position individually, they optimize for the full motion vector field $[u(x, y),\, v(x, y)]^T$ of the entire image at once. u and v then become functions of position. To avoid clutter, we omit the position and time parameters. The introduced optimization problem can then be expressed as:

$$\underset{(u,v)}{arg\,min} \iint \left(\nabla I \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \frac{\partial I}{\partial t}\right)^2 + \lambda \left(\|\nabla u\|^2 + \|\nabla v\|^2\right) dxdy, \tag{2.18}$$

where $\|\cdot\|$ is the Euclidean vector norm. Here, the gradient constraint equation is minimized in the same way as above. Additionally, spatial variation of the vector field is minimized by the second term. The regularization parameter λ controls the smoothness. By discretizing the integral and all partial derivatives therein, the optimal vector field can be obtained by solving the associated Euler-Lagrange equations. This is described in [100, 16]. Global optimization generally leads to better results than local methods, but it is more costly to compute and also more sensitive to noise.
While the assumption of a simple translational motion model is valid for individual pixels, the motion of an entire region is better described by more complex models. Techniques that extend motion to affine flow or use non-parametric motion models can be found in [13, 91, 12, 123, 35]. A good overview of the computation of optical flow is given in [33] and [10].

Block-based and Global Registration

Pixel-wise optical flow is well-suited for accurately compensating object motion. It has applications in the field of HDR imaging when temporally bracketed exposure sequences are merged in an offline process. However, establishing and solving a linear equation system for each pixel is too costly in a real-time HDR video scenario, where multiple exposures need to be registered within one frame time. It is more efficient to determine motion parameters for blocks of pixels or only one parameter set for the entire image. The goal is then to find parameters that maximize similarity or minimize the difference between pixels of one image I_t and the transposed pixels of the other I_{t+1}. A generic algorithm to estimate good parameters is then:

1. Pick motion parameters
2. Compensate motion, i.e., transform one of the images with the chosen parameters
3. Calculate similarity between the images
4. Repeat from 1 until a stop criterion is met
5. Choose the parameters that lead to the highest similarity.

This algorithm necessitates the definition of a suitable metric for the difference or similarity. An intuitive definition is the mean squared error (also called sum of squared differences) between pixel patches.

$$MSE(u,v) = \sum_{(x,y)\in W} \left(I_{t+1}(x+u, y+v) - I_t(x,y)\right)^2 \tag{2.19}$$

It is a function of assumed motion parameters (here, we use translation for illustration). The window W over which the difference is calculated can be chosen according to the needs of the application. Choosing it as the full image allows to find the global motion parameters that lead to the smallest MSE when applied. As in the optical flow case, the square function can be replaced by any function ρ that is more statistically robust. Another widely used similarity function is the cross correlation.

$$CC(u,v) = \sum_{(x,y)\in W} I_t(x,y) I_{t+1}(x+u, y+v) \tag{2.20}$$

It uses the product of pixel values instead of the difference which grows with increasing similarity. Both metrics can be normalized to be independent of the average brightness of the two images. This is desirable when used for exposure sequences.
After defining a good similarity metric, one needs to decide upon a search strategy for the motion parameters (step 1 of the above algorithm). An exhaustive search over all possible parameters is only feasible with a very small number of parameters (e.g., rotation or translation) and when restricted to a small search range (e.g., -10 to 10 pixels). For example, exhaustive search of translational motion vectors over pixel blocks is used in video compression. Another common strategy is to perform a hierarchical search: First, two image pyramids consisting of multiple downscaled versions of the two images are

created. Parameters are then found by exhaustive search over a small search range on the smallest scale. These parameters are used to initialize a local search on the next higher pyramid level. In this way, the parameters are iteratively refined until the full-sized images are reached. This is faster than a full search on only the original image for the same effective search range.
There are ways to achieve sub-pixel registration accuracy through intensity-based image alignment. One is to fit a continuous intensity function to the pixels of the images. Best matching motion parameters can then be found on the continuous curve between the pixels. This is similar to upscaling images in the pyramid search for further refinement. It is also possible to evaluate the error metrics in a region around the discrete optimal parameters, in order to interpolate the values and to find an analytic minimum.

Registration under Brightness Variation

Some intensity-based image registration techniques also take variation of the brightness between the images into account. Effects like vignetting, automatic exposure control and color correction can cause small variations in brightness. In the case of vignetting, the amount of variation depends on the location in the image. The most common way to handle such brightness differences is to model them by two adaptive parameters α for contrast and β for brightness [74, 37, 35, 4]. Disregarding the misalignment, the intensities of the two images to register are assumed to be linearly related:

$$I_{t+1}(x, y) = \alpha I_t(x, y) + \beta \tag{2.21}$$

α and β can then be included as two additional parameters to estimate together with motion. The linearity of this relationship makes it fit naturally into linear least-squares optimization. Brightness and contrast are calculated on the same scale as the motion parameters, e.g., on pixel-level, block-level or globally.
Brightness variation can be modeled more generally as a linear combination of appearance variation images $A_i(x, y)$ with coefficients λ_i [8, 51]. Again disregarding misalignment, the two images are related as

$$I_{t+1}(x, y) = I_t(x, y) + \sum_i \lambda_i A_i(x, y). \tag{2.22}$$

The A_i are predetermined according to the expected variation, e.g., constant to model a brightness offset or containing linear image gradients. As above, the coefficients are estimated in conjunction with the motion parameters.
The linearity assumption only holds within the small range of typical brightness change as a side-effect of image capturing. It may break in the case of exposure bracketing for HDR creation where large brightness differences are intended. The value of a saturated pixel in an image taken with a long shutter speed is no longer linearly related to a pixel in a darker exposure. There is thus a need for registration techniques that are specific to the registration of exposure sequences for HDR. Kang et al. [60] propose one for video sequences that alternate between dark and bright exposures. In order to create an HDR video frame at time t, they combine the current LDR image with its differently

exposed neighbors (at times $t-1$ and $t+1$ respectively). The intermediate motion is compensated in an offline process. It is observed that the neighboring LDR frames are closer to each other with respect to exposure values than they are to LDR image t. The authors thus incorporate motion information gained from registering the two neighboring frames with each other into the registration between t and its neighbors. They compensate both camera and scene motion by using a combination of hierarchical perspective transformations and a variant of the optical flow in [74]. A sufficient amount of common structure in the neighbor images is assured by choosing the exposure values for the sequence appropriately.
The approach by Kang et al. is further improved by Troccoli et al. [109]. Here, image sequences with varying exposure, taken from slightly different camera positions are used to obtain radiance maps and depth information simultaneously. Contrary to the previous approach, this algorithm also works with unknown exposure values and camera response. The authors make use of the fact that the normalized cross correlation is – to a certain extent – invariant to exposure. However, their approach only compensates camera motion and assumes a static scene.
A notable method for the registration of image sequences with large brightness differences has been published by Ward in 2003 [120]. It is very fast and suitable for compensating camera motion in real-time. Computational complexity is drastically reduced by using a simple translational model and binary images. Before registration, the images are thresholded such that 50% of the resulting pixels are black and 50% white. This is achieved by using the median pixel values as the threshold. The resulting image is called a mean threshold bitmap (MTB). Despite the greatly different exposure values, roughly the same pixels remain above or below the median, making MTBs suitable for registration. The pixels of the MTB are stored efficiently as single bits in 64-bit words. A global translation vector is found in a hierarchical 2D search on a pyramid of MTBs. On each level, the similarity score for nine offsets (the eight directions and no motion) is calculated. Image shifting is reduced to bit shifts, and pixel differences are computed as an XOR operation on 64-bit words and subsequent counting of ones by using a lookup table. The shift vector is then refined on the next larger pyramid level. Ward's method has been extended to rotational motion in [56, 40]. In [48], the computational efficiency is further improved by splitting up the hierarchical 2D search into two full 1D searches, and it is extended to video sequences.

2.4.3 Ghost Removal

The previous two sections describe how camera motion and sometimes object motion that takes place between two images can be compensated. However, such an image registration is never perfect. Object motion is generally too complex to model accurately by block-based registration or optical flow and may sometimes be ignored for computational efficiency. The residual motion that remains in the images then appears as ghosting artifacts in the merged result. Depending on the amount of motion between the shots, ghosting can appear around the edges of moving objects, or objects may appear in the result image twice with half intensity (see Figure 2.7). This motivates the development

Figure 2.7: *Example for a ghosting artifact. Two images were taken with a static camera and then merged together. The waving hand is in different positions in the two images. Each instance is visible in the merged image with half intensity, creating a ghost-like effect.*

of techniques for ghost removal in merged images. The underlying observation they all have in common is that object motion creates inconsistency within the image set. For each pixel or region in the target image, they identify a consistent subset of the images to create the result from. They mainly differ in how consistency is measured and which subset of the source images is used.

The problem of ghost removal was first considered in the context of panoramic images where images taken under different angles are stitched into one large picture. For each target pixel in the panoramic image, a set of source images that map to it is found. The target pixel is computed as a weighted average of the source pixels. Moving objects in the overlapping areas thus lead to ghosting artifacts. This was addressed in [23]. The authors attempt to divide the overlap between two source images into two regions such that no object boundaries are cut. Moving objects cause high differences between the pixels of the two images. Static background pixels, on the other hand, are consistent among all images if the global registration is accurate. The authors thus compute a relative difference image of the overlapping area and find a minimal path cutting through it using Dijkstra's algorithm [21]. This divides the overlap into two regions without cutting through objects. Low-pass filtering the difference image before applying Dijkstra's algorithm assures that broad paths of low differences are preferred. Each region is then filled with pixels from only one of the images to avoid ghosting.

It is not obvious how to generalize the above to an arbitrary number of overlapping images as is the case for general panoramic images. Uyttendaele et al. [113] recognized this problem. They segment the difference image in the overlapping area into contiguous

regions of high difference. Then they group corresponding regions of high difference across the source images to find regions belonging to the same real-world object. The regions are then represented as vertices in a graph with correspondences as its edges. By removing the vertex cover from the graph, only non-corresponding regions (unconnected vertices) remain. That is, multiple instances of a moving object are eliminated. The remaining regions can then be used to synthesize a ghost-free final image.

Ghost removal takes on a slightly different form when applied in the context of high dynamic range imaging. The source images have a much larger overlap – ideally close to 100%. Cutting through the overlapping area is thus no longer feasible. Additionally, taking the large exposure variations into account is essential for calculating suitable difference images.

In their book on HDR imaging [96], the authors describe an HDR-specific ghost removal technique. LDR sequences are combined into HDR images as a weighted average, too (see the following section on HDR stitching for details). This suggests to also compute a weighted variance image to detect inconsistencies between the source images by detecting high variance values. This is done by thresholding the variance image, dilating it and finding contiguous ghosting regions. For each region, the non-saturated LDR image with the longest exposure time is chosen to derive radiance from. So, rather than using a weighted average, regions of moving objects are filled with values from only a single source image. This makes sense, because all images are self-consistent. The radiance values derived from the chosen image are then blended with the original HDR result to create a ghost-free image.

Instead of completely discarding all but one source image to avoid inconsistencies, the authors of [62] manipulate the weights each image receives during HDR stitching of the ghosting region. Additional weights are determined iteratively for each pixel depending on its probability of being a background pixel. This has the effect that pixels that are likely to be part of a moving object receive lower weights and become less visible in the averaged HDR result. This is clearly advantageous in situations where a moving object is only visible in one of the source images, leaving all other images available for stable radiance estimation.

The same is also true for the approach published in [36]. Here, an LDR image with a long exposure, but a low number of saturated pixels is chosen as a reference. Inconsistencies are then detected with respect to the reference image. This is done patch-wise instead of for every single pixel. A patch in an image is considered to be inconsistent with the reference if a certain percentage of it differs from the reference image. All but the inconsistent patches are then used to create the HDR image.

2.5 HDR Stitching

Section 2.3 introduced methods to capture images of a high dynamic range of scene radiance values. All except for those doing HDR on the sensor level produce an LDR image sequence $I_i, i = 0, \ldots, n$ with varying, but known exposure parameters. We limit ourselves to varying shutter speeds Δt_i in this section for simplicity. Note that everything

mentioned here is also directly applicable to varying aperture size and gain. After image registration has been performed as described in Section 2.4, we may assume that a certain pixel position (x, y) represents the same point in the scene for all LDR images. The value $I_i(x, y)$ of a pixel in any of the images is an imperfect measurement of the scene radiance. We can now use all available measurements to recover the entire radiance map $L(x, y)$ of the scene. This is done by first bringing the pixel values that were taken under different imaging conditions into a common domain and then computing a weighted average.
The common domain is radiance, or scaled radiance to be more precise, which is sufficient for our purpose. A point in the scene gives rise to the same radiance $L(x, y)$ in all of the LDR images. Radiance is integrated inside a pixel of a CCD sensor for the duration of the shutter speed Δt_i, which is different for each I_i. What a sensor cell actually measures is the *exposure* $L(x, y)\Delta t_i$, which is radiance integrated over time. Only industrial digital cameras that are meant for measuring light accurately map exposure linearly to pixel values. Twice the radiance then results in a pixel value twice as high. Cameras which take pictures intended to be viewed by humans introduce a nonlinear mapping function f. The radiance $L(x, y)$ captured with a shutter speed of Δt_i is then mapped to a pixel value

$$I_i(x, y) = f(L(x, y)\Delta t_i), \tag{2.23}$$

typically in the range of 0 to 255.
For now, we assume that f is known. Details on how to obtain it are given in Section 2.5.1. Every reasonable mapping function is (stepwise) monotonic, i.e., more exposure of the cell leads to a higher pixel value. This means that f is invertible. With the shutter values and the LDR image sequence available, the inverse of f allows to make an estimate $\tilde{L}_i(x, y)$ of the radiance at position (x, y) from image I_i:

$$\tilde{L}_i(x, y) = \frac{f^{-1}\left(I_i(x, y)\right)}{\Delta t_i}. \tag{2.24}$$

The real radiance map $L(x, y)$ is then recovered by computing a weighted average with weights w over the obtained radiance values $\tilde{L}_i(x, y)$:

$$L(x, y) = \frac{\sum_i w\left(I_i\left(x, y\right)\right) \tilde{L}_i(x, y)}{\sum_i w\left(I_i\left(x, y\right)\right)}. \tag{2.25}$$

Suitable weighting functions w are discussed in Section 2.5.2. Because of the increased dynamic range of the radiance map, its values should be stored with a higher precision than that of 8-bit integer values. Using 32-bit floating point values works well in practice.

2.5.1 Estimating the Camera Response Function

A large number of works have been published concerning the estimation of a camera's response function [76, 24, 82, 97, 110, 63, 41]. We outline the three most influential methods in this section.
A response function f maps the exposure q to pixel values p: $f(q) = p$. Without loss of generality, we assume 8 bits per pixel. If q is below or above a certain exposure threshold,

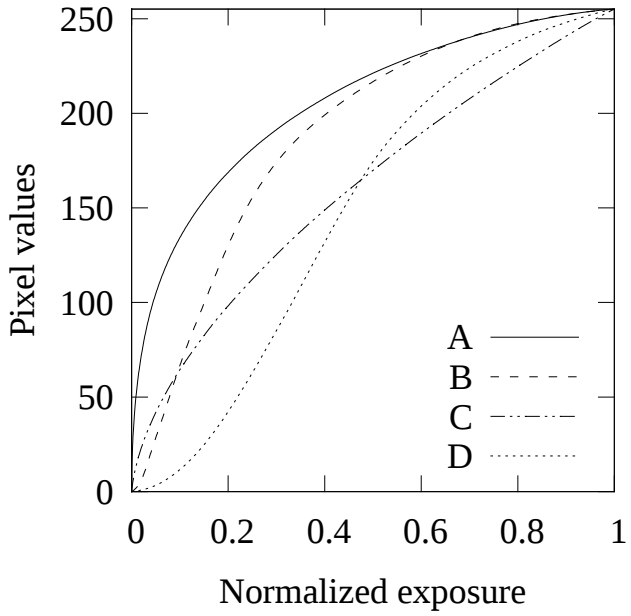


Figure 2.8: *Four examples of camera response functions that map exposure to pixel values. They are taken from the database of response functions (DoRF) [43]. Their names are given in the database as "F400CD Red"(A), "Agfachrome RSX2 100CD Red" (B), "Portra 160VCCD Red" (C), and "Kodachrome 25 Red" (D).*

it is mapped to 0 or 255, respectively. Otherwise it is mapped nonlinearly to the range of $[1, 254]$. See Figure 2.8 for exemplary camera response functions. It is safe to assume that any reasonable response function is monotonically increasing and thus invertible. In the context of recovering radiance, the inverse response function f^{-1} is of more interest. It maps a pixel value back to exposure, and ultimately to radiance.
Mann and Picard [76] suggest estimating the response function from two images I_0 and I_1. For this to be possible, the ratio $k = \Delta t_1/\Delta t_0$ between the shutter speeds under which the two images were captured must be known. They begin by choosing any dark pixel $I_0(x_0, y_0)$ in the first image. It is the result of an unknown exposure $q_0 = \Delta t_0 L$ being mapped by f:

$$I_0(x_0, y_0) = f(\Delta t_0 L) = f(q_0). \tag{2.26}$$

Now consider the same position in the second image $I_1(x_0, y_0)$. The pixel there is brighter as it is a mapping of the exposure $\Delta t_1 L$, which is a multiple of q_0 with the factor k:

$$I_1(x_0, y_0) = f(\Delta t_1 L) = f(k\Delta t_0 L) = f(kq_0) \tag{2.27}$$

Now search for a pixel in the first image that has exactly the same value. It is found at a new location (x_1, y_1), so $I_0(x_1, y_1) = f(kq_0)$. Again looking at the same position in the second image gives us an even brighter pixel, the mapping of exposure $k\Delta t_1 L$:

$$I_1(x_1, y_1) = f(k\Delta t_1 L) = f(k^2 q_0). \tag{2.28}$$

Continuing in this fashion, we obtain a sequence of explicit values for $f(q_0)$, $f(kq_0)$, $f(k^2q_0)$, ... They are values of the response function f at known multiples of the arbitrary exposure q_0. It is now possible to fit a function with the shape of $f(q) = \alpha+\beta q^\gamma$ to the sequence. It is invertible for nonzero β and γ.
Debevec and Malik [24] recover the inverse of the response function directly. By applying f^{-1} to both sides of Equation 2.23 and taking the logarithm, it can be rewritten as

$$log\ f^{-1}(I_i(x,y)) = log\ L(x,y) + log\ \Delta t_i. \tag{2.29}$$

The logarithm of the inverse response function is given a new symbol $g(\cdot) = log\ f^{-1}(\cdot)$ for simplicity. g maps the 256 pixel values to the logarithm of their corresponding exposures. The goal is now to recover the log exposures $g(p)$, $p = 0, \ldots, 255$ that fully describe the inverse response function. This is done by formulating an optimization problem from the above relationship:

$$\underset{g,L}{arg\,min} \sum_i \sum_{(x,y)} \Big(g(I_i(x,y)) - log\ L(x,y) - log\ \Delta t_i \Big)^2 + \lambda \sum_{p=1}^{254} g''(p)^2. \tag{2.30}$$

The first term minimizes the error of Equation 2.29 over all pixel positions (x, y) in all LDR images I_i using the known shutter speeds Δt_i. The unknowns are the 256 values of g as well as the original radiance map $L(x, y)$. The second term is a smoothness term with adjustable smoothness parameter λ. It minimizes the second derivative of g. In the discrete case, it can be written as $g''(p) = g(p-1) - 2g(p) + g(p+1)$. It relates the neighboring values of g to each other and controls the amount of variation in the response function. Only a small number of suitable pixel positions (x, y) are selected in practice. This is more efficient than summing over all pixels of all images. The optimization problem is quadratic in the values $g(p)$ and in the logarithm of radiance values. Calculating all partial derivatives with respect to the unknowns and setting them to zero thus yields a linear equation system which can be solved for the values of g.
Mitsunaga and Nayar [82] also consider the inverse camera response directly. They model it as a polynomial of degree M for which the coefficients c_j are to be determined:

$$f^{-1}(p) = \sum_{j=0}^{M} c_j p^j. \tag{2.31}$$

An LDR image sequence $I_i(x, y)$ forms the basis for the recovery of the coefficients. Only the ratios between the shutter speeds of two subsequent images must be available. They are denoted by $k_{i,i+1} = \Delta t_i / \Delta t_{i+1}$. Using Equation 2.23, it can be seen that $k_{i,i+1}$ is also the ratio between the inverse response function values of a pixel in images i and $i + 1$:

$$\frac{f^{-1}(I_i(x,y))}{f^{-1}(I_{i+1}(x,y))} = \frac{L(x,y)\Delta t_i}{L(x,y)\Delta t_{i+1}} = k_{i,i+1}. \tag{2.32}$$

Substituting the polynomial model for f^{-1} into this relationship, the authors obtain an equation for each pixel position in a pair of images:

$$\frac{\sum_{j=0}^{M} c_j I_i(x,y)^j}{\sum_{j=0}^{M} c_j I_{i+1}(x,y)^j} = k_{i,i+1}. \tag{2.33}$$

From the error of these equations, a least-squares optimization problem can be derived

$$\underset{c_j}{arg\,min} \sum_i \sum_{(x,y)} \left(\sum_{j=0}^{M} c_j I_i(x,y)^j - k_{i,i+1} \sum_{j=0}^{M} c_j I_{i+1}(x,y)^j \right)^2 \tag{2.34}$$

which can be easily solved numerically. The authors suggest solving it multiple times for different degrees $M < 10$ of the polynomial and choosing the degree resulting in the lowest total error.

2.5.2 Weighting Functions

The previously shown weighting function w in Equation 2.25 determines how much the radiance estimate $\tilde{L}_i(x,y)$ from a pixel $I_i(x,y)$ contributes to the corresponding HDR pixel $L(x,y)$. In other words, it judges a pixel's usefulness for recovering a radiance value based on its brightness value. Weighting functions are usually chosen to reflect noise characteristics of a camera, the derivative of its response function (i.e., the camera's sensitivity), and saturation effects. They are often found in the literature as parts of HDR creation techniques [24, 76, 82, 97]. Even though various weighting functions exist, they often share a few common properties. Most notably, the extremes of the pixel range are always assigned zero weight. This means that pixels with these values contain no useful information about the real radiance. As an example, a white sheet of paper and a reflection of the sun in a window can – under a certain exposure setting – both be represented by a pixel value of 255, even though the sun is several orders of magnitude brighter than the paper. The same reasoning applies to very dark pixels. Another common attribute of weighting functions is the location of their maximum, which is often in the middle or the upper half of the range. Pixels with a medium to high value are considered to be more faithful than dark pixels. This is due to the fact that a large portion of the image noise (e.g., quantization noise, fixed pattern noise) is independent of the amount of light falling onto the pixel. A bright pixel thus has a better signal-to-noise ratio than a dark one.

When the camera response curve is steep in a certain range, small changes in exposure result in relatively large changes of the mapped pixel value. This means that the camera is sensitive within this range. Mann and Picard [76] thus propose to use the derivative of the response curve as weighting functions, i.e., give more weight to pixels in the sensitive range. [96] suggests multiplying the derivative with a broad hat function to limit the support of the weighting function. The result is shown as the first example for a weighting function in Figure 2.9. The second example is taken from Debevec and Malik [24]. It assigns a maximum weight to the middle of the pixel range and decreases linearly towards the extremes. In practice, all choices adhering to the guidelines given in this section are equally well-suited for recovering radiance. The influence of the weighting function is rather small.

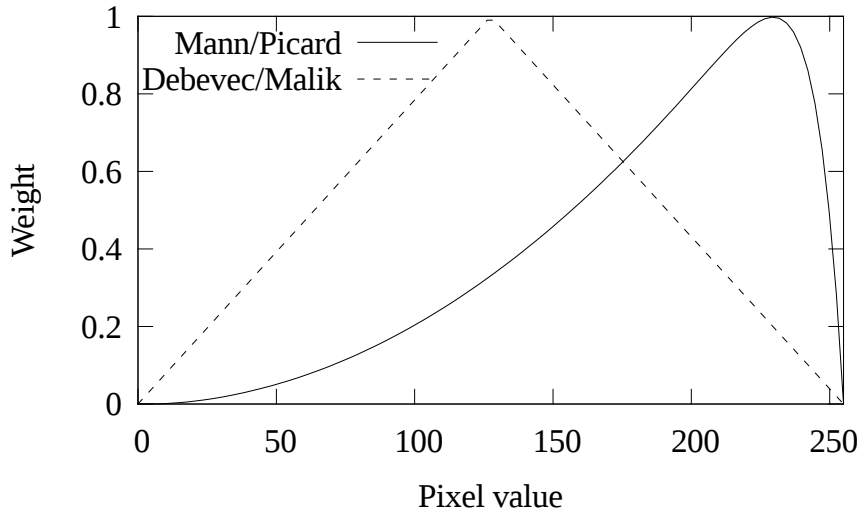


Figure 2.9: *The solid line is the pixel weighting function proposed by Mann and Picard for an assumed camera response curve. It is multiplied by a broad hat function to set the weight at the extremes to zero. The dashed line is the simple weighting function by Debevec and Malik.*

2.6 Tone Mapping and Display

HDR stitching is the last step in creating a frame in an HDR video. The resulting frame is a floating point valued map of (scaled) real-world radiance. It was created in a way that ensures that ratios of radiance values are maintained. For example, if the headlight of a car is 10^3 times more intense than the surrounding night scene, the pixel values in the HDR frame corresponding to the headlight are 10^3 times higher, too. Such an HDR frame may have a dynamic range spanning over four orders of magnitude, measured from the darkest shadows to the car's lights. Ideally, the ratios of the HDR pixels would be preserved during display. This however would require a display capable of producing the same range of output values. Such high dynamic range displays exist, but they are still not very common (see Section 2.6.1 for a discussion). Typical LCD and CRT displays only span a dynamic range of 2 to 3 orders of magnitude, which is not enough to reproduce an HDR video frame faithfully.

In addition to the inadequate dynamic range, the absolute luminous intensity produced by the headlight of a car exceeds that of an LCD by far. Consider the following rough calculation for illustration: A modern television set may emit a maximum luminance of 500 cd/m^2. Multiplied by the luminous area of a car lamp, we obtain a luminous intensity of 5 cd for the TV set. The luminous power of a halogen car light may be specified as 1500 lm. This results in a luminous intensity of 1500 $lm/(4\pi\ sr) \approx 120\ cd$ in all directions or 960 cd if focused into the solid angle of one eighth of a full sphere. It becomes clear that this intensity cannot be reproduced accurately. Similar arguments apply for the minimum reproducible luminance of an LCD screen. A pixel on such a screen selectively attenuates the backlight to produce contrast. The darkest luminance

is thus bounded by the luminance of the backlight multiplied by the lowest achievable transmittance of a pixel.
For these reasons, an HDR frame must be further processed in order to be displayable on a regular display device. Its dynamic range must be compressed to the limited output range of the screen. This final DR reduction step is called *tone mapping*. Simply scaling radiance down linearly would result in the loss of much of the detail obtained from HDR processing. The goal of tone mapping is to find a mapping between radiance and pixel values that preserves as much of the original appearance, color and detail as possible. This can be achieved by intentionally mapping very bright image areas to a darker output range to leave room for brightness variation. Section 2.6.2 introduces tone mapping operators that were designed for still images. Considerations for the extension to video are presented in Section 2.6.3.
The problem of tone mapping has been addressed even before high dynamic range imaging. In the context of computer graphics, using physical quantities like radiance to represent rays of light during rendering has been done for decades [39, 118]. The rendered output image is then a radiance map similar to an HDR image which needs to be tone mapped appropriately for display.

2.6.1 HDR Display Systems

With more and more HDR material emerging, there also arose a need for displaying HDR content. Consequently, prototypes for HDR still image viewers [119] and systems for viewing dynamic content were proposed [101, 89, 71, 20]. This section describes two of these systems. The latter of the two is the basis for a great portion of the subsequently developed displays. All HDR display systems are based on the same observation: Assume that any display system achieves a dynamic range of $c_1 : 1$ between the darkest and brightest intensity it can produce. Now add a second layer in front of the first display, for example an LCD panel with selectively attenuable pixels with a dynamic range of $c_2 : 1$. The theoretical contrast ratio of the combined system is then $c_1 \cdot c_2 : 1$. The dynamic range of a display can thus be increased by either selectively controlling the intensity of the backlight, or by using two layers of attenuators.
One of the first viewing systems dedicated to high dynamic range still images was proposed by Ward [119]. It combines the benefits of stereo vision and HDR imaging to permit viewing of an image with a high degree of realism. It is designed as a box with optics to look at a stereo HDR image inside. The optics provide a 120 degree field of view, which is approximately the field of view of the human eye. A strong light source in the back of the box providing a luminance of 20,000 cd/m^2 illuminates two layers of printed transparencies containing the image. One of the transparencies contains image detail like regular film, while the other is a blurred grayscale version to further darken the backlight where necessary. For regular film, the dynamic range is the ratio between maximum transparency and maximum opaqueness. The usable contrast ratio of film is estimated to be 100:1. Two layers thus achieve a dynamic range of 10,000:1.
Seetzen et al. [101] describe two different HDR display systems. Both are built as a combination of two regular displays. In each of them, the pixels of an LCD panel

selectively attenuate a backlight to create an image. A color pixel of such a panel typically only has a maximum transmittance of 3% to 8%, with a theoretical upper bound of 16.7% (half of the light is lost during polarization and two thirds are absorbed by the color filter array). The panel used in [101] only yields a contrast ratio of about 300:1. With the original backlight, this display produces luminance values ranging from 1 cd/m^2 to 300 cd/m^2. The dynamic range of the LCD panel is extended by replacing the static backlight with a dynamic light source.
In the first of the two systems, it is exchanged for a digital light projector that allows to spatially vary the light intensity. The image is projected onto the LCD panel and dark areas are further attenuated there. This combination allows to use a light source with a much higher luminous intensity while still maintaining a reasonable black level. This system can produce luminance values in the range of 0.05 cd/m^2 to 2700 cd/m^2, equaling a dynamic range of 54,000:1.
The second display uses a hexagonal array of 760 ultra-bright LEDs instead of a projector behind the LCD screen. The brightness of each LED can be controlled individually to create a low resolution version of the image. High frequency features are added by the high resolution LCD panel. To derive the two distinct image signals, the HDR image is factorized into an LCD and an LED component on a GPU. The maximum luminance this system achieves is 8500 cd/m^2 with a similar dynamic range as the one described above. The prototype was further improved into the BrightSide DR37-P display, which entered the market in 2005 as the first commercially available HDR display system.

2.6.2 Still Image Tone Mapping Operators

A variety of tone mapping (TM) operators for still images already exist. They are classified into spatially invariant, *global* operators and spatially variant, *local* operators [25]. Global operators are non-linear functions based on the content of an image as a whole, using statistical values such as minimum, maximum or average luminance to estimate optimal mapping parameters. When the optimal mapping function between luminance values and the displayable range is found, the same transformation is applied to each pixel in the luminance map. Psychophysical models of brightness and contrast perception as well as retinal properties serve as a basis for most approaches [81, 111, 31, 117]. These operators are simple and fast, but are limited in their ability to process very high dynamic ranges. A mapping of the full input range of luminance can be achieved by more engineering-oriented models such as logarithmic contrast compression [26] or histogram adjustment [121]. Due to their computational efficiency, global operators are the tone mappers of choice for real-time HDR video.
Local operators consider a set of neighboring pixels for estimation of the parameters of a local transformation function. Each pixel of an image is mapped differently, based on the local features of its neighborhood. Such tone mappers aim to enhance local contrast to create the impression of large brightness differences. Because the human visual system is only sensitive to local contrast, high quality images spanning high dynamic ranges are possible with these methods. Due to the more complex nature of these operators, computing time increases, making them badly suited for a real-time scenario. They are

also prone to artifacts such as halo effects.
Local operators can be further divided into different classes. Center-surround methods increase local contrast by computing the difference between a pixel's value and a weighted set of its neighboring pixels. This procedure is inspired by receptor properties of the human visual system [95, 5, 79, 94].
Frequency-based operators separate the low and high frequency bands of an image. Assuming that an image is a product of light intensity and reflection, only the part relevant to light intensity is compressed. This is the low frequency band, while the image details contained in the high frequency bands are kept. Oppenheim et al. published this technique known as the first TM operator which attenuates low frequencies more than high frequencies [92].
Gradient-based operators alter the gradient of an image. High frequency regions contain significant differences between neighboring pixels, while low frequency regions contain rather small differences. Fattal et al. observed that drastic luminance changes greatly increase gradients [29]. Thus the gradient field is compressed progressively and integrated back into an image by solving the corresponding Poisson equation.
There are three TM operators that are of particular interest in the later chapters of this thesis and are thus explained in more detail. They are Ward's *Contrast-Based Scale Factor* [117], the *Photographic Operator* by Reinhard et al. [95], and the *Histogram Adjustment* technique published by Ward et al. [121].
The *Contrast-Based Scale Factor* is a global operator. The primary goal of this method is the preservation of contrast. A constant of proportionality between display luminance $L_d(x, y)$ and world (scene) luminance $L_w(x, y)$ has to be found that yields a tone mapped result with roughly the same contrast visibility as the actual scene. Calculation of the scale factor m is based on the research of Blackwell, who defined a relationship between adaptation to luminance changes and the just noticeable luminance difference [106]. The value of m depends on the maximum luminance the display can produce as well as the average luminance in the image. It is then applied to pixel values to convert world luminance values into display values

$$L_d(x, y) = mL_w(x, y). \tag{2.35}$$

The *Photographic Operator* consists of a global prescaling step with subsequent local contrast enhancement. It is inspired by photographic development and printing techniques. First, a linear mapping reduces the range of world luminance $L_w(x, y)$ (i.e., range of the HDR pixels) to an intermediate displayable range of values $L_m(x, y)$ based on the average luminance $\bar{L}_w$ in the scene

$$L_m(x, y) = \frac{a}{\bar{L}_w} L_w(x, y). \tag{2.36}$$

[95] recommends setting the user parameter a to 18% of the display range. The intermediate values are then mapped nonlinearly by the following function:

$$L'_m(x, y) = \frac{L_m(x, y)\left(1 + \frac{L_m(x,y)}{L^2_{w,max}}\right)}{1 + L_m(x, y)}, \tag{2.37}$$

where $L_{w,max}$ is the maximum luminance contained in the HDR image. These two global mapping steps resemble deciding upon exposure settings when taking a picture. In the local step, the algorithm determines for each pixel position (x, y) a circular surrounding with maximum size that does not contain any sharp contrasts. The average luminance of this circular area is denoted by $L_{avg}(x, y)$. The intermediate values $L'_m(x, y)$ are then finally mapped to display luminance

$$L_d(x, y) = \frac{L'_m(x, y)}{1 + L_{avg}(x, y)}. \tag{2.38}$$

A pixel that is darker than its surrounding is mapped to a lower value, because $L'_m(x, y) < L_{avg}(x, y)$. If it is brighter, its brightness is further enhanced. Either way, the pixel's contrast relative to its surrounding is increased.
The third operator considered here is *Histogram Adjustment*. It globally applies a monotonic tone reproduction curve derived from the cumulative log histogram to all pixels. The idea is to allocate most of the displayable dynamic range to luminance ranges that are represented by many pixels. Thus, pixels in less frequent brightness levels are compressed more strongly. This is achieved by first computing a histogram H over the logarithm of the HDR pixels, i.e., over $log(L_w(x, y))$. Each histogram bin $H(j)$, $j = 1, ..., 100$ counts the number of pixels in its corresponding log luminance range b_j. The bins are summed up to a cumulative histogram $\bar{H}$ and normalized by the number T of pixels in the image as follows

$$\bar{H}(b) = \sum_{j}^{b_j<b} H(j)/T \tag{2.39}$$

$$T = \sum_{j} H(j). \tag{2.40}$$

The value of the cumulative histogram $\bar{H}$ at a luminance position b is thus the sum of all histogram bins $H(j)$ corresponding to a log luminance that is smaller than b. It satisfies $0 \geq \bar{H}(b) \geq 1$. The cumulative histogram could then be used directly in the following contrast equalization formula

$$logL_d(x, y) = logL_{d,min} + (logL_{d,max} - logL_{d,min})\bar{H}(logL_w(x, y)). \tag{2.41}$$

However, this mapping function has the flaw that contrast is *expanded* rather than compressed wherever there is a peak in the histogram. The authors thus present a number of algorithms for clipping of the histogram bins to prevent this from happening. Human visual limitations such as glare or visual acuity are also simulated in further processing steps. The log histogram created when applying this tone mapping operator to an HDR frame can be used to derive optimal shutter speeds for the next frame of an HDR video. This will be demonstrated in Chapter 5.
Among the many available tone mapping operators, it is difficult to find the one best suited for a given application. The choice needs to be made based on the computational efficiency of the operator, and also the quality of the LDR output. Quality rankings of

various tone mapping operators can be found in the literature as the results of subjective experiments. In these studies, the outputs of the TM operators are compared with each other [18], with the unmodified HDR image shown on an HDR display [68], or with the real-world scene itself [128, 6].

2.6.3 Tone Mapping for Video

A tone mapping operator typically considers the minimum, maximum or average luminance of a recorded HDR image and scales the image accordingly. However, the operators described so far are designed with still images in mind. That is, the dynamic range of each frame of an HDR video sequence is scaled independently. In situations where the minimum, maximum or average brightness of a frame differs greatly from its predecessor, the scaling function changes rapidly from one frame to the next, leading to visible image flicker. Such a situation can easily arise when a bright object, such as the headlight of a car or a specular reflection, enters the field of view. The tone mapper attempts to map the greatly increased dynamic range onto the same output range, often changing the brightness of the entire image in the process. This can be alleviated either by post-processing of the tone mapped image [45] or by employing a TM operator specific to HDR video content.

Pattanaik et al. [93] were among the first to model the gradual adaptation of the human visual system (HVS) to abrupt changes in scene intensity specifically in the context of temporal coherence in HDR sequences. Their tone mapping operator is built upon a solid basis of published quantitative measurements of the HVS. It tone maps HDR sequences frame by frame, but achieves temporal coherence by maintaining a number of state variables describing the current adaptation level. Scene radiance is first converted into photoreceptor responses – much like the conversion between irradiance and illuminance demonstrated in Equation 2.3. The responses are then modified in accordance with the current adaptation level and further processed into appearance parameters like whiteness/blackness and colorfulness. These are finally mapped to the display intensity.

An implementation of the Photographic TM operator [95] using programmable graphics hardware was first presented by Goodnight et al. [38] and later in a similar fashion by Wang et al. [116]. Goodnight et al. extend the TM operator to include a time-dependent model that is based on the adaptation model proposed by Durand and Dorsey [27]. It describes a multi-pass interactive rendering that computes the average luminance in a first pass and tone maps the scene in the second pass. Both multiplicative and subtractive light adaptation is simulated by applying a global multiplicative scale factor during the mapping from world luminance to display luminance.

[67] also implement the Photographic Operator for a GPU and extend it to react to temporal changes in luminance conditions in accordance with the human visual system. They add perceptual effects like glare and night vision as a post-processing step, which have not been considered in the previous work of Goodnight et al. [38].

An approach using gradient domain tone mapping in HDR videos was introduced by Lee et al. [69]. The gradient field of the HDR image is compressed as in [29]. Pixel-wise motion information is then incorporated into the Poisson equation used to create an

output image from the adjusted gradients. Incorporating motion information leads to a decrease of visible flicker in the resulting LDR video. This is measured by comparing the average luminance variation of the output frame over time.
More recently, Benoit et al. [11] proposed a model based on properties of the human retina. HDR video content is enhanced by a non-separable spatio-temporal filter with added temporal constancy. This is done by imitating the retina's luminance compression and additional temporal information processing.

Chapter 3

System Overview

In this chapter, we give an overview of the HDR video system presented in this thesis, with novel approaches for acquisition, registration and visualization. We introduce two separate methods for the fast capturing of the LDR sequences required to create an HDR frame. Both aim to reduce the redundancy when capturing multiple images of the same scene. Furthermore, we present an image registration technique that is both robust to extreme brightness differences and fast enough to be used on real-time video. For the visualization of HDR video, we show an extension of existing still-image tone mapping techniques to video. It mainly focuses on the removal of flickering artifacts arising in the case of video.
The HDR video pipeline, as implemented for this thesis, is shown in Figure 3.1. It consists of four modules: LDR image capture, image registration, HDR stitching, and video tone mapping. We contribute new ideas to the fields of capturing, registration and tone mapping and use existing techniques for HDR stitching.

3.1 LDR Image Capture

The capturing of LDR images constitutes the first step. We present two alternative methods for capturing with reduced redundancy. The decision about which one to use is based on the capabilities of the capturing camera as well as the preferred optimization. The first one minimizes the amount of image data by only re-capturing the potentially small badly exposed areas of a base LDR image. These areas are detected during the capturing process, and new images are triggered one by one. Our second approach is to only use the shutter speeds that contribute the most information to the HDR frame. In this case, the shutter sequence is determined in advance using the histogram of the previous frame. It is transmitted to the camera which then captures all images in one go.

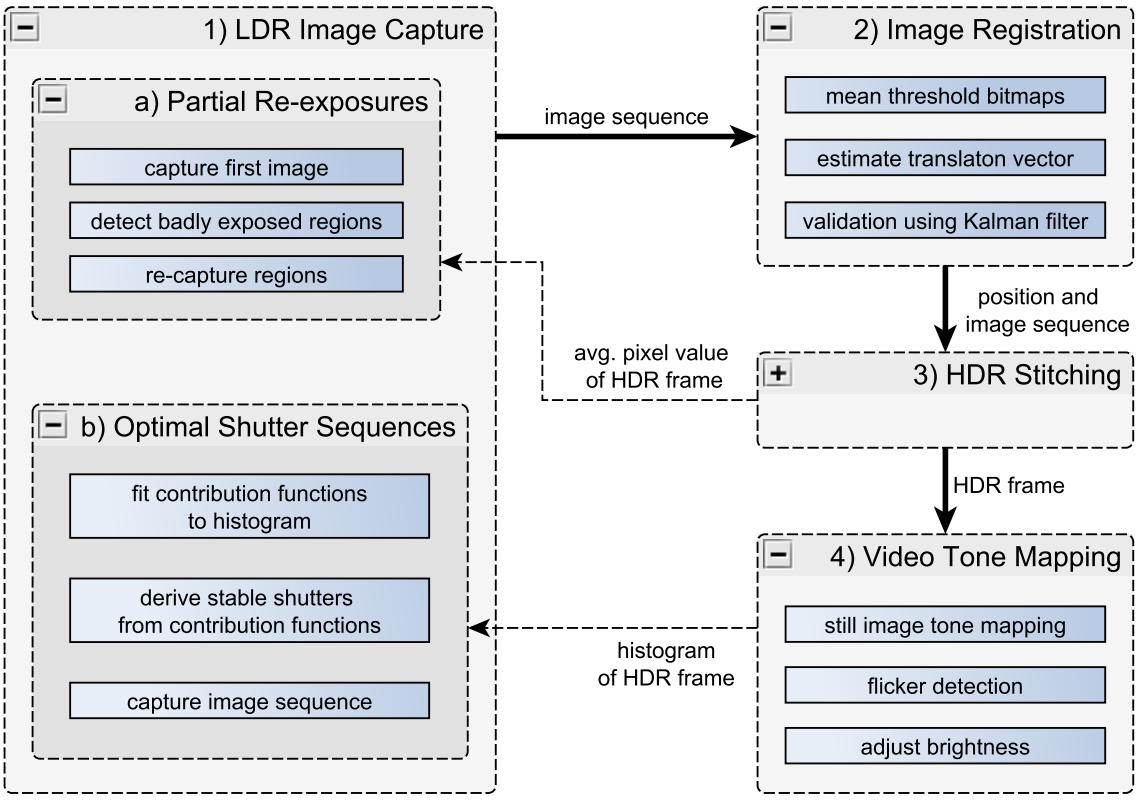


Figure 3.1: *Overview of the HDR video processing system introduced in this thesis. As a first step, a sequence of LDR images is captured using one of the two presented methods. The image sequence is passed to the registration module where camera motion in the sequence is compensated. The registered sequence is then stitched into a single HDR frame, which is finally tone mapped for display. HDR image statistics are passed back to the capturing module and used to determine the capture parameters for the next frame.*

3.1.1 Capturing with Partial Re-Exposures

Many industrial FireWire CCD cameras have a feature called *true partial scan*. It allows the definition of a rectangular sub-area of cells on the CCD sensor – a region of interest – to be read out while all other cells are being discarded. As a result, the time needed to read out the relevant parts of the sensor and to transmit the image data over the FireWire bus is reduced, leading to a higher frame rate at lower image sizes.

In our partial re-exposure approach, we do not capture a fixed number of LDR images with varying shutter speeds, but adapt the number to the dynamic range of the scene. Additionally, we make use of the idea that it might not always be necessary to capture a full image at another exposure setting if only a few image areas require a higher dynamic range. We developed an algorithm that detects badly exposed regions in an already

captured image and triggers the camera to re-capture only those regions. Reducing the image size decreases the overall capture time of an image significantly. Capturing partial images also reduces the amount of redundant data that is used to merge the LDR images into an HDR image which saves additional processing time.

3.1.2 Determining Optimal Shutter Sequences

Another way of speeding up capturing is to optimally choose shutter speeds at which to capture. The fewer images are captured, the less time is needed to process them, leading to higher frame rates. Yet at the same time, the dynamic range of the scene may necessitate a certain minimum number of exposures so that all detail is captured properly. So the goal is to get the most out of the recorded exposures.
In our HDR video system, the histogram of scene radiance values is a by-product of tone mapping the previous frames with Ward's histogram adjustment technique [121]. This second approach thus uses the available histogram to calculate a shutter speed sequence in real-time. The shutter speeds are chosen in a way such that frequently occurring radiance values are well-exposed in at least one of the captured LDR images. This increases the average signal-to-noise ratio (SNR) for a given number of exposures or minimizes the number of exposures required to achieve a desired SNR.
The shutter speed algorithm is based on our definition of *contribution functions* to specify precisely what we mean by "well-exposed". An image pixel is a noisy measurement of physical radiance. The quality of this measurement is a function of the pixel value, with higher values generally leading to a more accurate measurement. This circumstance is modeled by our contribution functions which are an extension to the weighting functions introduced in Section 2.5.2. They are a concept similar to the noise models used in other shutter speed methods.
In order to be applicable to video, we also consider bootstrapping and convergence to a stable shutter sequence. Additionally, we introduce a stability criterion for the shutter speeds to prevent flicker in the video.

3.2 Histogram-based Image Registration

We address the challenge of estimating the camera motion between two partial LDR images in an efficient way. We argue that a purely translational camera motion model is sufficiently accurate for high frame rates ($\geq$ 200 frames per second). This assumption is supported by [120]. The goal of the registration algorithm is thus to estimate a translation vector between two LDR images captured at different exposure settings. We improve the approach based on mean threshold bitmaps [120]. The hierarchical 2D search is replaced by two separate exhaustive 1D searches to speed up the computation. We start by counting the number of dark pixels in each column of both frames to be aligned to create column histograms. By using a normalized cross correlation between the two column histograms, we estimate the horizontal component of the translation vector. Repeating this process for image rows allows us to estimate the vertical component in

the same way. The resulting vector is then validated using a Kalman filter to incorporate knowledge of the prior motion into the estimation.

3.3 HDR Stitching

The registered image sequence is then merged into a single frame. We do not contribute own ideas to the field of HDR stitching and simply use the techniques introduced in Section 2.5.
The cameras we employ are industrial FireWire cameras with linear response. This means that the exposure of a sensor cell is mapped linearly to a pixel value. The camera response function f is then the identity function. For our cameras, this was verified using the response estimation technique by Mann and Picard [76] (see Section 2.5.1). The same weighting function w is used for HDR stitching and for determining optimal shutter sequences. Our weighting function of choice is discussed in Chapter 5 of this thesis.
It is computationally cheap to add a calculation of the average radiance of the HDR frame to the HDR stitching process where the frame is created. In the next frame, the average is used to calculate an initial shutter speed for the capturing with partial re-exposures.

3.4 Flicker Reduction in Tone Mapped HDR Videos

It is our goal to perform tone mapping of HDR videos using standard operators designed for still images. When doing so, temporal changes of the minimum, maximum, or average scene radiance lead to flicker in the tone mapped video. We propose a generic method for the automatic detection and removal of flicker. It is implemented as a post-processing step to be independent of the tone mapper used.
Essentially, the per-frame brightness difference (flicker) is smoothed over a number of frames to become less obtrusive. We first introduce a criterion to detect flickering frames. It is based on the difference of the average image brightness of two consecutive frames and on a threshold derived from Stevens' power law which relates the physical magnitude of a stimulus to its perceived intensity. Our assumptions are that: Image flicker is the most obtrusive artifact introduced by applying still image tone mappers to videos, and flicker can be detected sufficiently well by analyzing the average brightness of a tone mapped frame. These assumptions are supported by our experimental results.
To reduce the visibility of detected flicker, we adjust the average image brightness after tone mapping. This is done using image normalization and clamping to the output range, which is included as a last processing step in many tone mapping operators anyway. A frame that was mapped to a much darker average than its predecessor is adjusted to a level closer, but still darker than the previous frame. We always keep the brightness variation within a range that is tolerable according to our flicker detection. After a few frames of convergence, the same average brightness the operator would maintain without our intervention is reached again.

Chapter 4

Capturing with Partial Re-Exposures

4.1 Properties of "True Partial Scan"

In order to describe our partial re-exposure algorithm, we begin by analyzing the relationship between image capture parameters (e.g., image size and shutter speed) and the resulting capture time. From this relationship, our algorithm for selecting image regions of interest and doing re-exposure is derived.
Many industrial FireWire CCD cameras have a feature called "true partial scan". It allows the definition of a rectangular sub-area of cells on the sensor – a *region of interest* (ROI) – to be read out while all other cells are being discarded. As a result, the time needed to read out the relevant parts of the CCD sensor and to transmit the image data over the FireWire bus is reduced, leading to a higher frame rate at lower image sizes.
We first identify relevant camera parameters that influence capture speed and infer general rules for the choice of parameters in Section 4.1.1. Then we apply the model exemplarily to a specific camera in Section 4.1.2. The work described here was published in [46].

4.1.1 Parameters and General Rules

The relevant parameters in our scenario are:

- shutter speed setting,
- position of the ROI on the CCD sensor, and
- width and height of the ROI to be captured.

The *shutter speed setting* determines how long the sensor cells are exposed to light before being read out and transmitted to the PC. This value imposes a delay in the capturing process and is added to the overall capture time.
The *position of the ROI* on the CCD sensor has no significant influence on capture time. Any ROI of a given size requires the same amount of time to be read out and transmitted, no matter where it is located on the sensor.
The *ROI size* has the most interesting influence on capture speed. Increasing the height of the ROI leads to a linear increase in capture time. This is obvious because CCD sensors are usually read out row by row at a constant frequency, while rows that are not captured are discarded completely. Contrary to this, no time can be saved by decreasing the ROI width because the read-out time of a row on the sensor is constant. The ROI width merely determines the number of bytes per read-out cycle produced by the camera. This data is split into packets of a fixed size and sent over the bus at the bus' own cycle frequency. Since the bus packet size can only be set in discrete steps, a slight *increase* in capture time is perceived when decreasing the ROI width.
From these considerations, we can conclude the following:

- The position of ROIs to be captured is irrelevant.
- Given that the camera's data rate does not exceed the bandwidth of the FireWire bus, it is most efficient to capture images at full *width*.
- The *height* of a ROI to be captured should be chosen as small as possible.

Additionally, some CCD cameras require a minimum total image size to capture efficiently. Since the image width is fixed, this requirement results in a lower bound for the ROI height h_{min}.
At full image width, the total capture time in milliseconds at a given shutter setting Δt and image height h can be described as

$$T(\Delta t, h) = a + \Delta t + c + v \cdot h, \tag{4.1}$$

where a is the time to set up an image buffer of appropriate size and to allocate bandwidth on the bus. c and v are the camera-specific constant and variable capture costs. a, c and v need to be determined experimentally as described in Section 4.1.2.
Note that the constant cost of image capturing $(a + \Delta t + c)$ can exceed the variable cost $v \cdot h$ for small ROI heights by far. Instead of capturing two close but distinct ROIs, it can therefore be more efficient to capture both regions and the area in between in one step. According to the above considerations, the distance between two ROIs can be expressed by the number d of image rows between them. It is more efficient to merge two regions and capture d additional rows in between rather than capturing twice and doubling the constant cost if

$$v \cdot d < a + \Delta t + c \quad \Leftrightarrow \quad d < \frac{a + \Delta t + c}{v}. \tag{4.2}$$

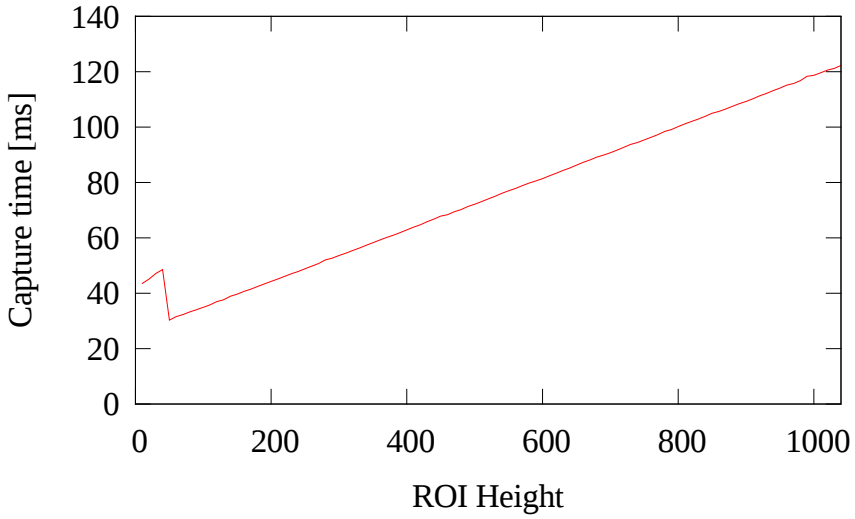


Figure 4.1: *Measured capture time in milliseconds for ROIs of a given height and full width.*

4.1.2 Estimating Capture Costs

In our experiments for this chapter, we used an AVT Marlin F-145B2 FireWire camera with a maximum resolution of 1392 by 1040 pixels, B/W. It allows shutter values ranging from 0.037 ms to 81.9 ms and features "true partial scan".

To estimate the camera-specific coefficients, we captured images at different heights and measured the time taken. We kept the image width constant at 1392 pixels and varied the height from 10 to 1040 in steps of 10, measuring each size five times for an average. The shutter speed was set to the minimum possible value of 0.037 ms and subtracted later. The time to allocate image buffers and bus bandwidth and to trigger the camera was averaged over all image sizes, resulting in a value of $a = 20.36$ ms. For each individual exposure, we started to measure the capture time after triggering and stopped when the image was fully received. The results are shown in Figure 4.1. As can be seen from the plot, it is inefficient to capture ROIs with a height of less than 48 rows – this is where the total image size falls below 64 kB. As a consequence of this characteristic trait, we set h_{min} to 48 for our camera. In order to estimate c and v, we fit a regression line to the sample data starting from $h = 50$ and obtained the values $c = 25.63$ ms and $v = 0.09778$ ms per row.

For our camera, the total time t_{capt} to capture an image of full width, height h and shutter speed Δt can therefore be calculated as

$$t_{capt}(\Delta t, h) = 20.36 + \Delta t + 25.63 + 0.09778 \cdot h. \tag{4.3}$$

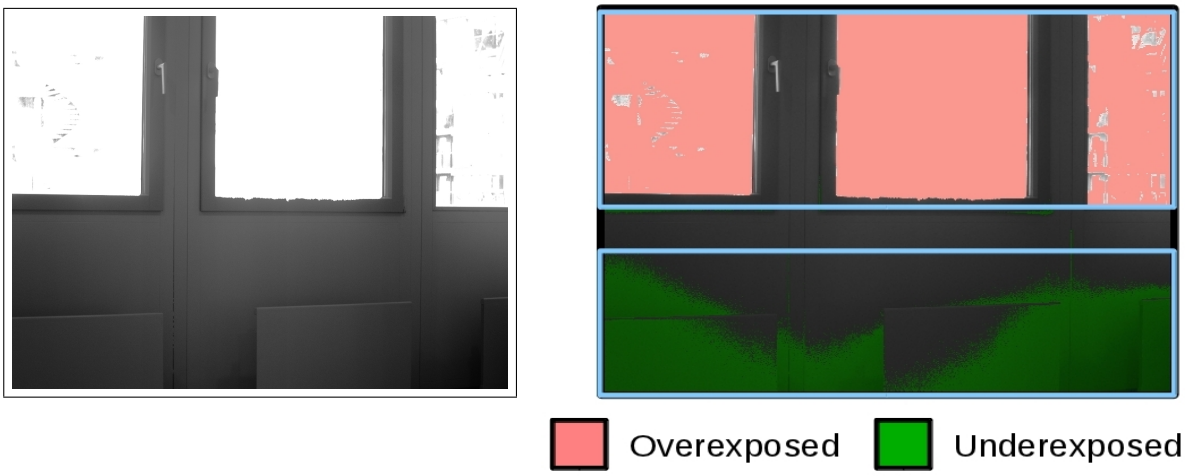


Figure 4.2: *The left image shows the base image of an LDR image sequence. Some areas of the base image are badly exposed. Only the rectangular ROIs are captured again with a shorter (top) and longer (bottom) shutter speed.*

4.2 The Partial HDR Algorithm

Our algorithm to capture HDR images using partial re-exposures of poorly exposed regions can be divided into the following steps:

1. Capture a base image of the scene at full resolution and an initial shutter setting,
2. Search the captured image for under- or overexposed pixels,
3. Group these pixels into ROIs for re-exposure and determine an appropriate shutter speed setting,
4. Re-Capture all ROIs from the previous step with a different shutter setting and repeat from 2 using each newly captured image,
5. If no more under- or overexposed regions are found, create an HDR image from the set of exposures.

This is illustrated in Figure 4.2.
The algorithm explores the base image and all subsequently captured partial images iteratively and captures at only as many shutter speeds as necessary to cover the full dynamic range of the scene. As a side effect, it is insensitive to changes of the initial shutter setting. Details on how to set the initial shutter speed are given in Section 4.2.2.
In order to search captured images for under- or overexposed pixels, we introduce a simple criterion to determine bad exposure based on the brightness value of an image pixel: A pixel is *valid* if its brightness value p lies within an interval $[p_{min}, p_{max}]$ and is *invalid* otherwise. In other words, very dark or very bright pixels are poorly exposed (invalid) and are considered for re-exposure. The camera used in this chapter is an 8-bit

industrial camera with very little dark noise. We thus choose $[p_{min}, p_{max}] = [10, 254]$. The choice for p_{min} is more or less arbitrary and can be adjusted to the needs of the particular application and capturing device.

4.2.1 Determining ROIs for Re-Exposure

This Section describes the analysis of an image for invalid pixels and the identification of rectangular image areas to be captured again at different shutter speeds. We derive the latter mostly from the results of Section 4.1.
The ROI detection process starts with the first captured image – the base image. It is the only image that is searched for both under- and overexposed pixels. All subsequently recorded images have either lower or higher shutter speeds than the original image and are only analyzed towards their corresponding direction. The two directions "higher shutter speeds" and "lower shutter speeds" are performed independently but in parallel to some degree, as can be seen later in Section 4.2.3.
Our considerations in Section 4.1 have shown that no performance gain can be achieved by capturing images at less than full width. We therefore restrict the set of possible ROIs to those with a width equal to the full width of the CCD sensor. Such a region is fully described by the location of the first row belonging to the ROI and its height. Thus, as a first step in determining areas for re-exposure, a histogram is created with as many bins as the number of rows in the image to be considered. Each bin stores the number of invalid pixels found in its corresponding image row. Note that two histograms must be created for the base image. For all later images, *one* histogram for either under- or overexposed pixel counts is sufficient.
From now on, only row histograms counting invalid pixels are considered, reducing the problem of finding ROIs to a one-dimensional one. As a preprocessing step, a morphological closing is done to the row histogram to achieve a preliminary grouping of nearby rows with high numbers of invalid pixels. A threshold r_{max} is then applied to the histogram, marking those image rows having an invalid pixel count of more than r_{max} percent. Marked rows in the histogram are the ones to be considered for re-exposure. By changing the parameter r_{max}, it is possible to adjust the trade-off between capture speed and image quality: Setting r_{max} to a lower value results in more rows to be marked for re-exposure, leading to a lower number of invalid pixels that remain in the final HDR image, but also to increased costs for capturing. The influence of r_{max} on the image capturing process is further examined in Section 4.3.
Next, the thresholded row histogram is searched for contiguous runs of marked rows. These constitute the basic ROIs for re-exposure. Before being pushed into the image capture queue, they are expanded to a minimum size of h_{min} rows, and ROIs that are closer together than d rows are merged into single regions to accommodate the properties of the camera's "true partial scan" feature.
Lastly, the detected ROIs are pushed into the queue of images to be captured. Depending on whether the image was analyzed for under- or overexposed pixels, the regions will be re-exposed with either longer or shorter shutter speeds, respectively. In our approach, we double or halve the shutter speeds. By doing so, there will be enough image pixels

that are valid in both of two consecutive exposures, so they can be used for image registration purposes. As soon as no more invalid pixels are found in any of the newly captured images or the required shutter speed exceeds the camera's limits, the algorithm terminates. Therefore, no more shutter speeds than necessary to capture the scene's dynamic range are used.

4.2.2 Setting the Initial Shutter

Once an HDR frame is created from the partial re-exposures, it can be used to estimate an optimal initial shutter setting for the next base frame. Optimal in this context means that the expected number of re-exposures in the next set is minimal. Our approach is to set the next initial shutter speed to a value that maps the average radiance of the scene to a central pixel value of the next exposure. The average radiance is conveniently obtained during HDR stitching.
A pixel is not under- or overexposed if its value lies within an interval $[p_{min}, p_{max}]$. Using equation 2.24, the radiance interval resulting in well-exposed pixels under a given shutter speed Δt is given by

$$\left[\frac{f^{-1}(p_{min})}{\Delta t}, \frac{f^{-1}(p_{max})}{\Delta t}\right]. \tag{4.4}$$

Taking the logarithm of the radiance values results in an interval of

$$\left[log\left(\frac{f^{-1}(p_{min})}{\Delta t}\right), log\left(\frac{f^{-1}(p_{max})}{\Delta t}\right)\right] \tag{4.5}$$

in the log HDR image. The size of this interval no longer depends on Δt because Δt becomes a scalar offset in the log domain. The center of the interval is

$$\frac{1}{2}\left(log\left(\frac{f^{-1}(p_{min})}{\Delta t}\right) + log\left(\frac{f^{-1}(p_{max})}{\Delta t}\right)\right). \tag{4.6}$$

Figures 4.3 and 4.4 illustrate the intervals.
The average L_{avg} of the logarithm of the scene's radiance values is calculated during HDR stitching. We now set the next shutter speed Δt so that the average log radiance L_{avg} is mapped to the center of the interval of well-exposed pixels:

$$\begin{aligned}
& L_{avg} = \frac{1}{2}\left(log\left(\frac{f^{-1}(p_{min})}{\Delta t}\right) + log\left(\frac{f^{-1}(p_{max})}{\Delta t}\right)\right) \\
\Leftrightarrow \quad & L_{avg} = \frac{1}{2}\left(log\left(f^{-1}(p_{min})f^{-1}(p_{max})\right) - 2log\left(\Delta t\right)\right) \\
\Leftrightarrow \quad & log\left(\Delta t\right) = \frac{1}{2}log\left(f^{-1}(p_{min})f^{-1}(p_{max})\right) - L_{avg} \\
\Leftrightarrow \quad & \Delta t = \frac{2^{\frac{1}{2}log\left(f^{-1}(p_{min})f^{-1}(p_{max})\right)}}{2^{L_{avg}}} \\
\Leftrightarrow \quad & \Delta t = \frac{\sqrt{f^{-1}(p_{min})f^{-1}(p_{max})}}{2^{L_{avg}}}
\end{aligned}$$

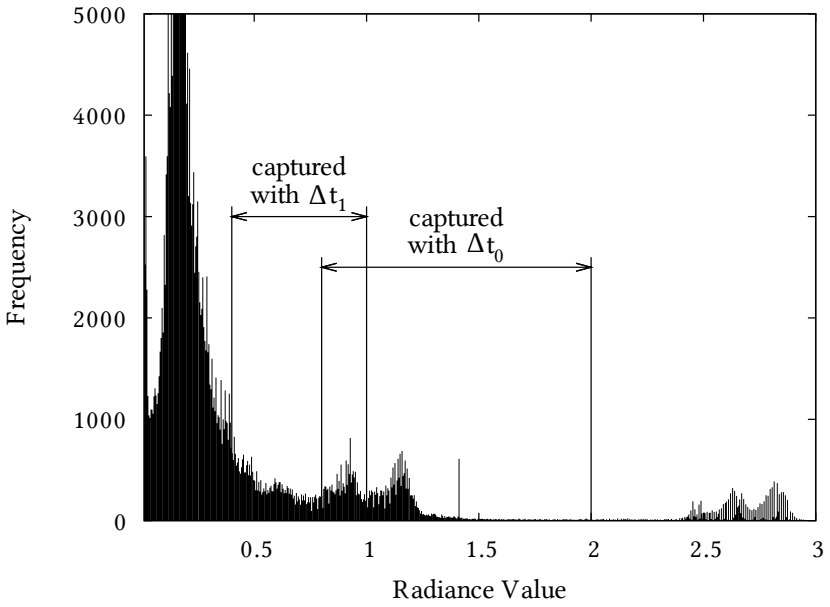


Figure 4.3: *Histogram of an HDR frame. The marked radiance intervals will be neither under- nor overexposed in images captured with a shutter speed of Δt_0 and Δt_1.*

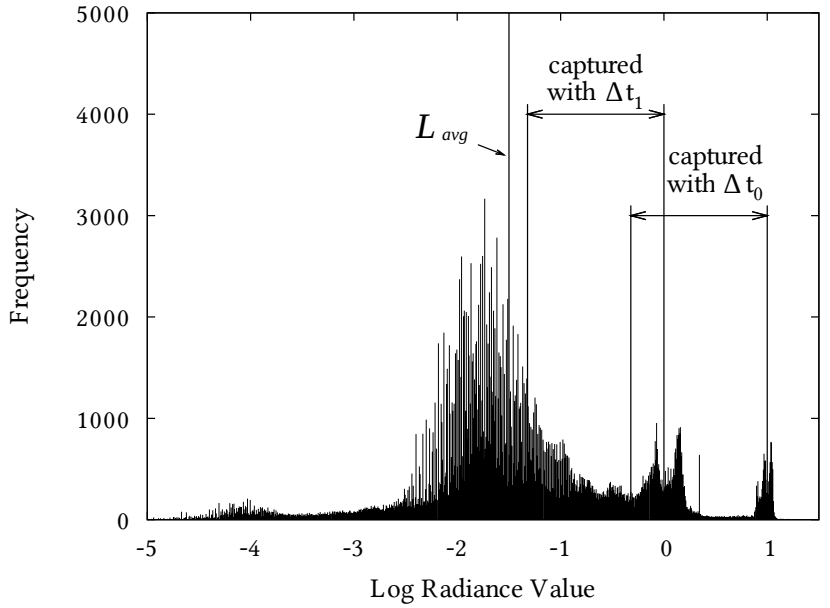


Figure 4.4: *Histogram of the logarithm of the same HDR frame as in Figure 4.3. The log radiance intervals are now the same size. L_{avg} is the average of the logarithm of the scene radiance.*

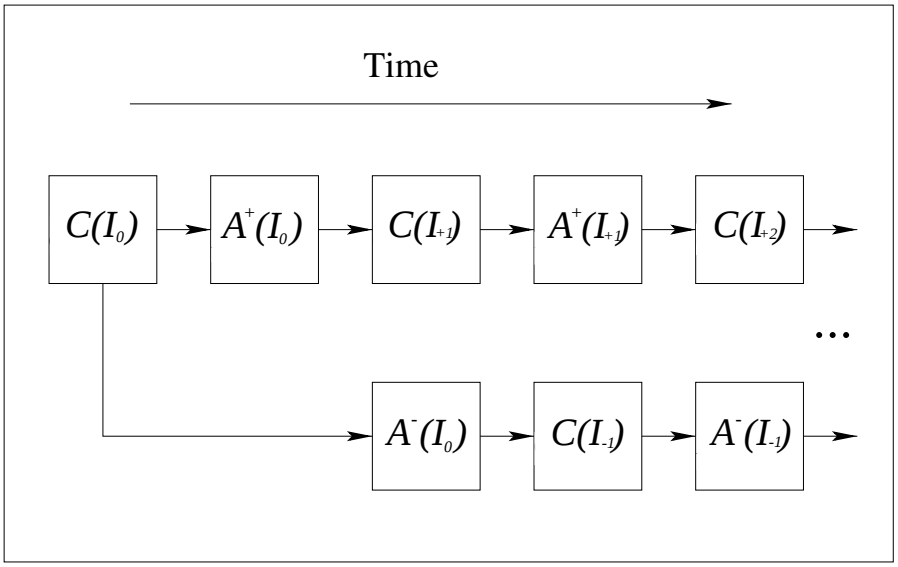


Figure 4.5: *Capturing and analyzing for badly exposed pixels can be interleaved to use the camera and the CPU in parallel. While a darker image is captured $C(I_{-1})$, a previously captured image I_{+1} can be searched for overexposed pixels $A^{+}(I_{+1})$.*

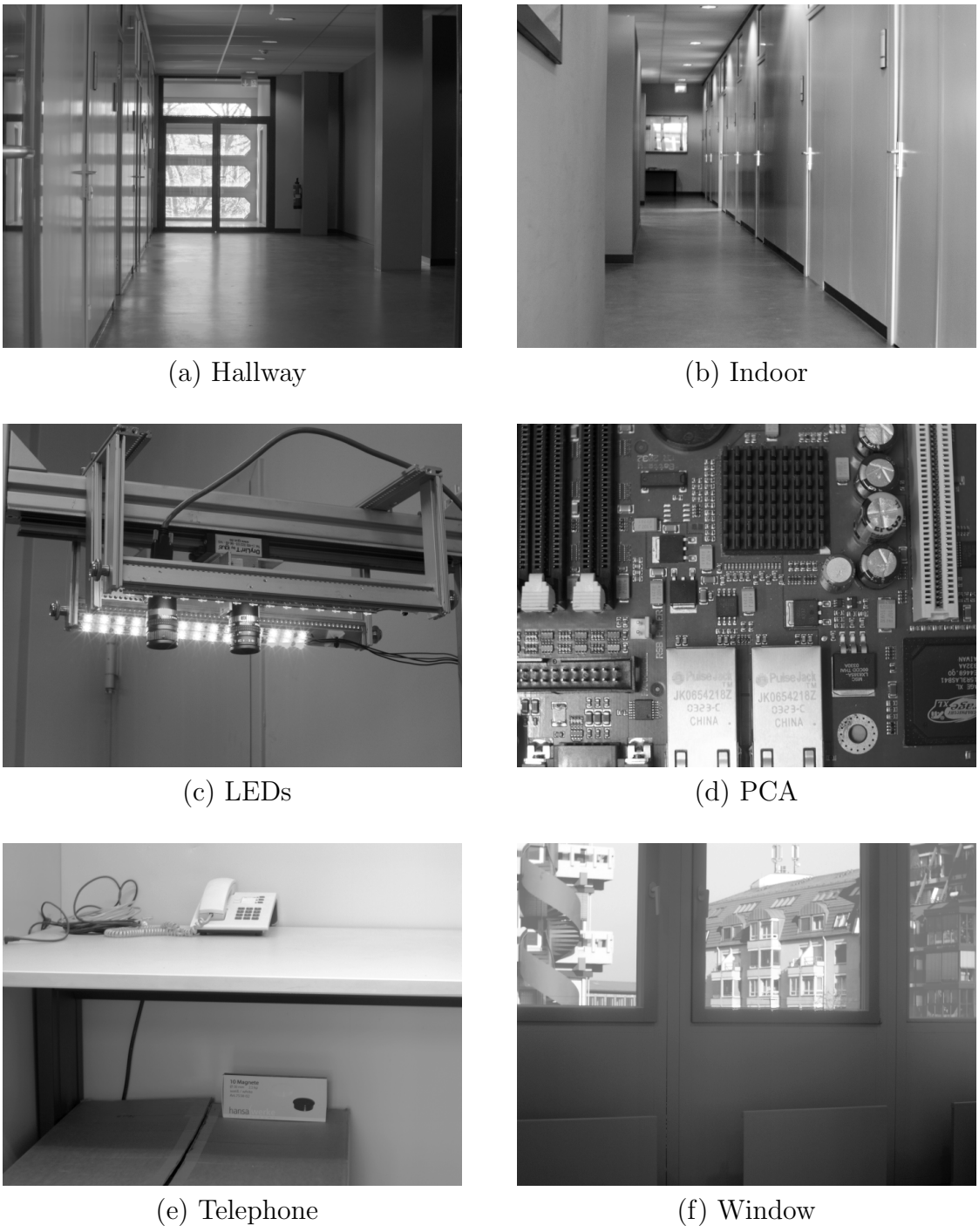
4.2.3 Implementation Issues

Our experiments in Section 4.1 revealed a phenomenon that can be utilized for an efficient implementation of the partial HDR algorithm. From an implementation point of view, the image capturing process can be divided into two distinct parts. The first part is the setup phase where an image buffer is allocated, bandwidth on the bus is reserved, and the camera is triggered to record an image. In the second phase, the camera exposes the sensor to the light of the scene, reads out the currents accumulated in its CCD cells, and sends the image data over the FireWire bus to the PC where it is written into memory via "Direct Memory Access" (DMA). During this second phase, until the image is fully received from the camera, the CPU is idle. We can therefore use this idle CPU time to analyze the captured images without adding to the overall capture time.
In our implementation, we make use of this fact in the following manner: First the base image I_0 is captured. We denote the process of capturing an image by $C(I)$. Next, the base image is analyzed for overexposed regions which are then put into a re-exposure queue. We denote the analysis for under-/overexposed regions by $A^{-}(I)$ and $A^{+}(I)$, respectively. After these two initial sequential steps, the algorithm can be parallelized: While the overexposed regions are re-captured with a shorter exposure time to create a new image I_{+1}, we use the idle CPU time to analyze I_0 again, this time for underexposed regions. Then, during the process of capturing the brighter image I_{-1}, the darker image I_{+1} is analyzed, and so on. This process is illustrated in Figure 4.5.
Our way of implementing this is by using two queues. One queue contains the images to be analyzed and the other contains camera settings for images to be captured. In each step, one element from *each* queue is considered, and both are processed in parallel. We

found that in our setup, capturing even the smallest possible image took longer than analyzing a full image. As long as there are images in the capture queue, the analysis can thus be performed for free and will not add to the overall capture time.

4.3 Experimental Results

We conducted experiments to evaluate the performance of our algorithm. The two performance criteria we considered were: 1) Time saved to create an HDR image from partial re-exposures, and 2) the quality of the resulting image with regard to the number of invalid pixels that remain in the final result. Both quantities were evaluated for our partial image HDR approach with adaptive numbers of exposures and compared to the traditional approach of creating HDR images using full images. For convenience we refer to the two approaches as "partial HDR" and "full HDR". To compare the results, we first ran our algorithm on the test data to determine the set of exposures used and then measured the full HDR approach using the same set. The camera was an AVT Marlin F-145B2 black and white camera with a maximum resolution of 1392 by 1040 pixels, capable of capturing 15 frames per second.
As test data, we created six scenarios we refer to as Hallway, Indoor, LEDs, PCA, Telephone and Window. Tone mapped HDR images of these can be seen in Figure 4.6. Each scenario consists of twelve saved exposures of the same static scene captured with a static camera. The shutter speeds were set to $2^i \cdot 0.02ms, i = 1, .., 12$ in accordance with all possible shutter settings requested by our algorithm.
A compromise had to be made between making our measurements reproducible and measuring time taken to capture images as realistically as possible. The former suggests using saved images while the latter requires capturing live images. We chose to conduct our experiments on saved image data. Instead of recording live images with the camera, we copy ROIs from saved images and use the results of Section 4.1 to assess the required capture time. As described in Section 4.2.3, the computation time that can be scheduled in parallel to an image acquisition is neglected.
Table 4.1 compares the results of the speed measurements in each of the six scenarios using full and partial HDR. In this experiment, the parameter r_{max} was set to 0.7%. It can be seen that using partial re-exposures in these scenarios saves 20-49% of the time to create an HDR image. This leads to an increase in achievable frame rates by 25-96%. As expected, the LED example (see Figure 4.6) leads to the largest performance gain because the bright areas cover only a small portion of the scene. As a "worst case" example, the Hallway scenario contains overexposed regions that comprise a large area of the scene (due to reflections on the floor). Throughout all scenarios, the overhead introduced through image analysis accounts for approximately 5% of the overall duration.
The parameter r_{max} influences the process of detecting invalid regions in an image. It determines the maximum allowed percentage of invalid pixels in an image row. Rows that contain at most r_{max} percent invalid pixels are not re-exposed and may generate under- or overexposed pixels in the final HDR image. As stated before, r_{max} is an optimization parameter allowing to adjust the trade-off between capture speed and image quality.



(a) Hallway (b) Indoor

(c) LEDs (d) PCA

(e) Telephone (f) Window

Figure 4.6: *Tone mapped HDR images of the six scenarios we used for our experiments. (a) and (b) were chosen as worst case examples for our approach: (a) has very large overexposed areas due to the window in the background and reflections on the floor, and (b) displays an indoor scene with a rather low dynamic range. A high performance gain is to be expected in scenario (c) where the bright LEDs cover only a small portion of the image. The metal parts in (d) reflect the light source and become overexposed. (e) and (f) are typical HDR examples with bright and dark regions.*

Scenario	Capturing	Analysis	Stitching	Total
Hallway (f)	1263	0	365	1628, 100%
Hallway (p)	954	77	274	1305, 80%
Indoor (f)	515	0	174	689, 100%
Indoor (p)	367	26	126	519, 75%
LEDs (f)	1936	0	545	2481, 100%
LEDs (p)	981	41	254	1276, 51%
PCA (f)	1493	0	451	1944, 100%
PCA (p)	1078	46	285	1409, 72%
Telephone (f)	897	0	256	1153, 100%
Telephone (p)	642	30	169	841, 73%
Window (f)	1074	0	327	1401, 100%
Window (p)	739	87	246	1072, 77%

Table 4.1: *Measured time in milliseconds taken for image capturing, image analysis and HDR stitching. For each scenario the two approaches full HDR (f) and partial HDR (p) were examined.*

Scenario	r_{max}	Invalid	Total time
Hallway	0%	0%	87%
	0.7%	0.02%	80%
	5%	0.31%	67%
	10%	3.68%	49%
PCA	0%	0%	89%
	0.7%	0.04%	72%
	5%	1.04%	42%
	10%	3.78%	37%

Table 4.2: *Influence of r_{max} on image quality and capture speed. The "Invalid" column displays the percentage of invalid pixels in the HDR image. "Total time" shows the total time taken to capture, analyze and stitch images in relation to the speed achieved by the full HDR approach as shown before.*

The influence of r_{max} on these two performance criteria is illustrated in Table 4.2 for the Hallway and PCA images. It shows the percentage of invalid pixels in the created HDR image and the overall time taken to capture, analyze and stitch images. The overall time is expressed as the percentage of the full HDR capture time, analog to Table 4.1. To avoid clutter, we chose two exemplary scenarios and four different settings for r_{max} each. The results in the other scenarios were similar.

4.4 Conclusions

We have shown a technique to capture HDR images more efficiently than by capturing images with varying exposure at full resolution. By capturing partial images and selecting the range of shutter settings used adaptively, we were able to increase the frame rate by 25-96%. We also showed how our algorithm can be parallelized, so that image analysis is performed while exposing a new image. Images can therefore be analyzed "for free" while waiting for the next image. Capturing partial images also reduces the amount of redundant data that is inputted to HDR stitching which allows for more time to be saved.
A limitation of our approach is the relatively high constant cost of the capturing process. In our scenario, only roughly one half of the total capture time was dependant on the image size, setting an upper bound to the achievable performance gain.

Chapter 5

Optimal Shutter Speed Sequences

In this chapter, we present the alternative capturing approach that makes use of the cameras sequence mode. A shutter sequence is determined and transmitted to the camera, which then captures the image sequence asynchronously. We begin by defining our concept of contribution functions and describe a useful relationship between these functions and histograms over the logarithm of scene radiance values. This relationship is then exploited in our optimal shutter sequence algorithm.

5.1 Contribution Functions and Log Radiance Histograms

Section 2.5.2 introduced weighting functions in the context of merging corresponding pixels of an LDR image sequence into one scaled radiance value. A weighting function w is defined for the 256 different values an 8-bit pixel can take on. It determines a weight for a pixel which reflects the pixel's suitability as an accurate measurement of radiance. Figure 5.1 shows the weighting function we use in our HDR video system. In our experiments, we found that the function shown in the plot gives the best results, but our approach also works with other choices.
For a given shutter speed Δt, we can calculate how well a radiance value L can be estimated from an image captured at Δt by combining the camera's response function and the weighting function. A radiance value L is mapped to a pixel value using the camera's response function f. The weighting function w then assigns a weight to the pixel value. We define

$$c_{\Delta t}(L) = w(f(L\Delta t)) \tag{5.1}$$

as the *contribution* of an image captured at Δt to the estimation of a radiance value L. In the special case of a linear response function, $c_{\Delta t}$ looks like a shifted and scaled version of w. An example of a contribution function in the log domain is shown in Figure

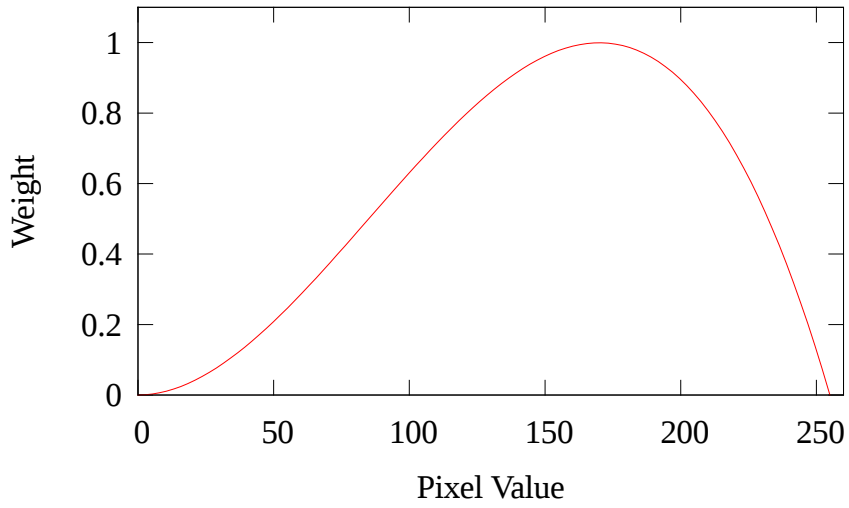


Figure 5.1: *The weighting function we use in our experiments. The weight of a pixel is its value multiplied by a hat function normalized to a maximum weight of 1.*

5.3. Representing radiance in the log domain is preferable in our scenario, because then the shape of $c_{\Delta t}$ no longer depends on the shutter speed Δt.

In our HDR videos system, the scene's brightness distribution is known from tone mapping of the previous frame. We can thus use the available log radiance histogram to calculate a sequence of shutter speeds Δt_i which allows the most accurate estimation of the scene's radiance. We do this by choosing the Δt_i such that the peaks of the contribution functions $c_{\Delta t_i}(L)$ of the LDR images coincide with the peaks in the histogram. That is, radiance values that occur frequently in the scene lead to LDR images to be captured which measure these radiance values accurately. This is illustrated in Figures 5.2 and 5.3.

The histogram over the logarithm of scene radiance has M bins. Each bin with index $j = 1, ..., M$ corresponds to the logarithm of a discrete radiance value: $b_j = log(L_j)$. Bin j counts the number $H(j)$ of pixels in the HDR image having a log radiance of b_j. The bins have even spacing in the log domain, meaning that for any j, the log radiance values b_j and b_{j+1} of two neighboring bins differ by a constant $\Delta b = b_{j+1} - b_j$. The non-logarithmic radiance values corresponding to two neighboring bins thus differ by a constant factor $exp(\Delta b) = exp(b_{j+1})/exp(b_j) = L_{j+1}/L_j$.

Equation 5.1 states that, for a given shutter speed Δt and an LDR image captured using Δt, the value of $c_{\Delta t}(exp(b_j))$ indicates how accurately log radiance b_j is represented in the LDR image. When considering log radiance histograms, the continuous contribution function is reduced to a discrete vector of contribution values. It has one contribution value for each radiance interval of the histogram. We can now exploit a useful relationship between the log radiance histogram and our contribution vector: Shifting the contribution vector by a number of s bins leads to

Figure 5.2: *Example of a tone mapped HDR image.*

$$\begin{aligned} & c_{\Delta t}(exp(b_j + s\Delta b)) \\ = \; & w(f(exp(b_j + s\Delta b)\Delta t)) \\ = \; & w(f(exp(b_j)exp(\Delta b)^s \Delta t)) \\ = \; & w(f(exp(b_j)\Delta t')) \\ = \; & c_{\Delta t'}(exp(b_j)), \end{aligned}$$

where

$$\Delta t' = exp(\Delta b)^s \Delta t. \tag{5.2}$$

This means that the contribution vector corresponding to shutter speed $\Delta t'$ is identical to a shifted version of the original vector. We thus easily obtain an entire series of contribution vectors for shutter speeds that differ by a factor of $exp(\Delta b)^s$. In other words, only the shift, but not the shape of the contribution function depends on the shutter speed in the log domain. This allows us to move the contribution function over a peak in the histogram and then derive the corresponding shutter speed using the above formula.

5.2 Optimal Shutter Sequence

In order to compute an optimal shutter speed sequence, we first calculate an initial contribution vector from the known camera response and a chosen weighting function. Camera response functions can be estimated as described in Section 2.5.1. The initial shutter speed Δt to compute $c_{\Delta t}$ can be chosen arbitrarily. For ease of implementation,

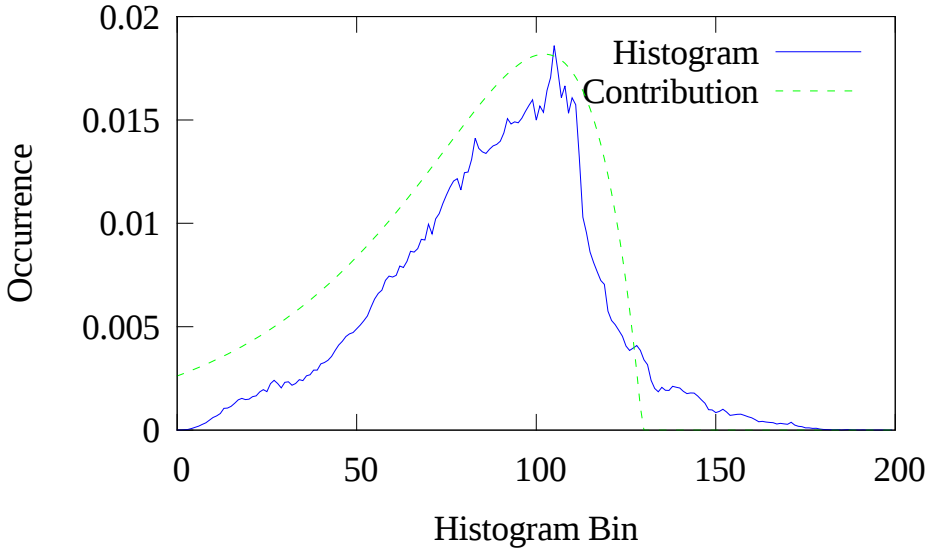


Figure 5.3: *The solid line depicts the log radiance histogram of our example scene (Figure 5.2). The dashed line is the contribution function in the log domain corresponding to a shutter speed chosen by our algorithm. The exposure was chosen such that it captures the most frequently occurring radiance values best.*

we choose Δt such that the first histogram bin is mapped to a pixel value of 1, that is $f(exp(b_1)\Delta t) = 1$. Note that $f^{-1}(0)$ is not uniquely defined in general. The size of the contribution vector depends on the dynamic range of the camera, reflected in its response function. Reaching a certain scene radiance $L_{N+1} = exp(b_{N+1})$, the camera's pixels will saturate, resulting in $f(exp(b_j)\Delta t) = 255$ for $j \geq N + 1$ in case of an 8 bit sensor. It is safe to assume that any reasonable weighting function assigns zero weight to this pixel value. Hence, the contribution vector $c_{\Delta t}(L_j) = w(f(exp(b_j)\Delta t))$ consists of N nonzero values. It can be shifted to $M + N - 1$ possible positions in the log radiance histogram. Each shift position s corresponds to a shutter speed Δt_i, which can be calculated using Equation 5.2: $\Delta t_i = exp(\Delta b)^s \Delta t$. This equivalence between shutter and shift is utilized later.

Here, we explain how a new shutter speed is added to an existing shutter sequence. The first shutter can be determined analogously. So we assume that the sequence already consists of a number of shutter speeds Δt_i. To each Δt_i belongs a contribution vector $c_{\Delta t_i}(L_j)$, with $L_j = exp(b_j)$ being the radiance values represented by the histogram bins. See Figure 5.3 for an example. We now need to decide whether to add another shutter to the sequence or not, and find out which new shutter brings the biggest gain in image quality. For this purpose, we define a *combined contribution vector* $C(L_j)$ that expresses how well the radiances L_j are captured in the determined exposures. We make the assumption that the quality of the measurement of a radiance value only depends on the highest contribution value any of the exposures achieves for it. The combined

contribution is thus defined as the maximum contribution for each histogram bin

$$C(L_j) = \max_i \left(c_{\Delta t_i} \left(L_j \right) \right). \tag{5.3}$$

This definition can now be used to calculate a single *coverage value* C to estimate how well exposed the pixels in the scene are in the exposures. C is obtained by multiplying the frequency of occurrence of a radiance value $H(j)$ by the combined contribution $C(L_j)$ and summing up the products:

$$C = \sum_{j=1}^{M} C(L_j) H(j). \tag{5.4}$$

This is essentially the same as the cross correlation between the two. The algorithm tries out all possible shifts between a new contribution vector and the log histogram. The shutter speed corresponding to the shift that leads to the biggest increase of C is added to the sequence.
The algorithm described so far is greedy in that it does not reconsider the shutter speeds it already chose. We added a second iteration over the shutter sequence to allow for some hindsight refinement. All shutters but the first one are refined in the same way. The first shutter is treated differently as described later. The shutter to be refined is first removed from the sequence. The algorithm for finding the next best shutter according to the maximum increase of C is then applied again. In most cases, the resulting shutter value is similar, but slightly better than the previous choice with respect to coverage. This is because the algorithm is aware of the rest of the sequence at this point. Our experimental results support this claim.

5.2.1 Stop Criteria

If the histogram is normalized such that its bins sum up to 1 and the weighting function has a peak value of 1, then C is in the range of [0..1] and can be expressed as a percentage. $C = 1$ then means that for each radiance value in the scene, there exists an exposure which captures it perfectly. However, perfect coverage is not achievable in a realistic scenario. It is more practical to stop adding shutters to the sequence once a softer stop criterion is met. We came up with three different stop criteria: the total number of exposures, a threshold for C and a maximum sum of shutter speeds. The sum of shutter speeds must not exceed the available time between two video frames. The criterion that limits the *total number of exposures* is always active. It guarantees that the algorithm terminates after calculating a finite number of shutter speeds. We also use this criterion to manually choose the number of exposures for our evaluation for better comparability. This is described in more detail in Section 5.3.
The *threshold for the coverage value* C is a quality criterion. A threshold closer to 1 allows for a better estimation of scene radiance, but requires to capture more exposures. We chose $C \geq 0.9$ for our running system.
For the type of camera we employ, the capture time of a frame is roughly proportional to the exposure time. And since we are interested in capturing real-time video at 25 frames

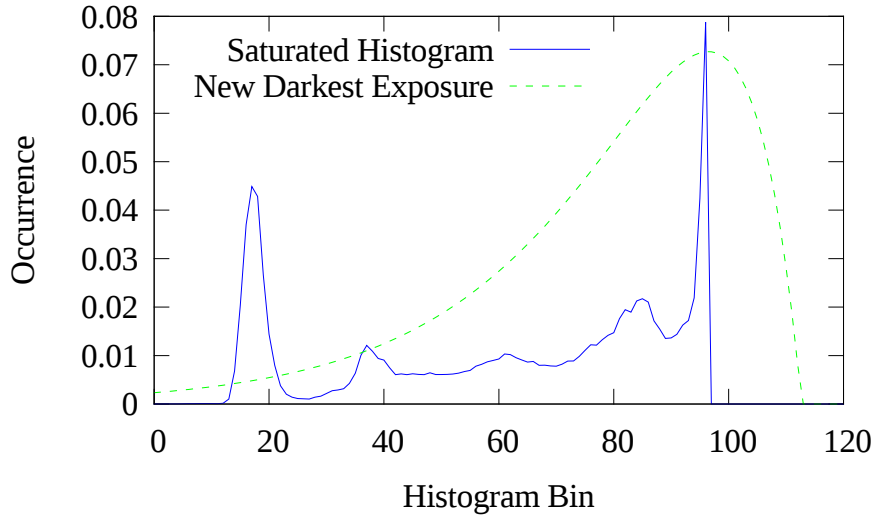


Figure 5.4: *Some areas of the scene are overexposed even in the darkest exposure. It shows up as a peak at the highest radiance value in the histogram. In the next frame, the algorithm chooses a shutter speed that covers the peak. By doing so, areas with a higher radiance than the previous maximum can still be captured faithfully.*

per second, the sum of all shutter speeds must not exceed 40 milliseconds. Our third stop criterion is an adjustable threshold for the *sum of shutter speeds*. However, it should be made clear that the algorithm has little control over meeting this requirement. In the example shots we took, only two exceeded the threshold. But they in turn overshot it by a large factor. We argue that it is the camera operator's responsibility to adjust aperture and gain or to use a different lens to cope with particularly dark scenes.

5.2.2 Adapting to Brightness Change

So far, we described the algorithm to determine a sequence of shutter speeds for a single HDR frame based on a perfect histogram of the scene. However, there are two major problems that arise when applying this algorithm to HDR video directly: imperfect histograms and flicker.

Perfect histograms are not available in a real video. The available histograms are created from the previous frame which generally differs from the current one. Furthermore, the dynamic range covered by the histogram is only as high as the range covered by the previous exposure set. For example if the camera pans towards a window looking outside, the bright outdoor scene may be saturated even in the darkest exposure. This shows up as a thin peak at the end of the histogram of the previous frame (see Figure 5.4). How bright are these pixels really? To find out, the algorithm needs to produce a shutter sequence that covers a larger dynamic range than the histogram of the previous frame indicates. This allows the sequence to adapt to changes in the scene.

We accomplish this by treating the first shutter in the sequence differently. The special

treatment is based on the observation that underexposed images contain more accurate information than overexposed ones. The dark pixels in an underexposed image are a noisy estimate of the radiance in the scene. However, this noise is unbiased. Saturated pixels, on the other hand, always have the maximum pixel value, no matter how bright the scene actually is. As a consequence of this observation, the first shutter is chosen such that its contribution peak covers the highest radiance bin of the histogram. The peak of a weighting function is usually not located at the highest possible pixel value. This means that radiances beyond the peak – if existing in the next frame – are still represented by a non-saturated pixel. See Figure 5.4 for an example. This allows to faithfully record radiance values that are higher by a certain percentage than the previous frame's maximum, and the sequence can adapt to brighter scenes. Change towards a darker scene is less critical, because underexposed pixels still contain enough information about the real radiance to calculate a new longer shutter time. With adaptation enabled, bootstrapping becomes straightforward. We can start with any set of shutter speeds and arrive at the correct values after a few frames. The speed of adaptation is evaluated in Section 5.3.2.

5.2.3 Avoiding Flicker

The second problem to deal with when applying our algorithm to HDR video is flicker. It is a side effect of changing the shutter sequence over time. Consider the following scenario: A bright saturated area like a white wall leads to a peak at the highest histogram bin. This gives rise to a darker exposure taken in the next frame as shown in Figure 5.4. The darker exposure causes the histogram peak to spread out over several bins. It may now cause too little extra coverage to justify the darkest exposure. In this situation, the algorithm oscillates between including the lowest shutter speed and omitting it. In the resulting video, the white wall would alternate between having texture and being completely saturated.

Another reason why stable shutter sequences are desirable is the way we operate our camera. A sequence of exposure parameters is sent to the camera. It then repeatedly captures exposures by cycling through the parameter list. This is done asynchronously, and the captured exposures are buffered. Changing the shutter sequence requires a costly retransmission of the parameters, and the buffers are used suboptimally.

For these reasons we impose a stability criterion upon the shutter sequence. We begin by defining whether two given shutter speed sequences are *similar*. If the number of shutters in the two sequences differs, then they are not similar. If it is the same, then we calculate the distance between their shutter values. The distance between two shutters is expressed as a percentage to model their exponential nature. For each value in the first sequence, the closest shutter speed in the second one is found. This search is necessary because the order of the lists is arbitrary. The distance between all closest shutter pairs is averaged. If the average is greater than a threshold (we use 20%), the sequences are not similar. Otherwise they are similar.

Using this definition, we achieve temporal stability by distinguishing between two states: *changing* and *static*. We always run our algorithm to determine a new shutter sequence.

In the *changing* state, this new sequence is used directly, and new parameters are transmitted to the camera. In the *static* state, the sequence is simply discarded, and the parameters of the previous frame are kept. A change between the states occurs according to the following rules:

- When in the *static* state and the newly determined sequence is not similar to the previous one, increase a counter.
- If more than certain number of non-similar sequences occur in a row (3 in our system), transition to the *changing* state.
- A sequence similar to the current one always brings the algorithm back to the *static* state and resets the counter.

These rules have the effect that small variations in the shutter speeds are ignored. Once the scene actually changes, it takes three frames to react. Then the algorithm retains its original flexibility. It is able to adjust in each frame until a stable shutter speed sequence is found again. For fast bootstrapping, the system starts in the *changing* state.

5.2.4 Reducing the Image Size

When capturing with partial re-exposures as described in the previous chapter, capturing and image analysis are interleaved, so that the algorithm can adapt the image sizes to the scene while acquiring the LDR sequence. To maximize throughput for capturing with optimal shutter speeds, we use the camera's sequence mode. This means that the camera parameters must be specified completely before the it starts exposing the first image. Ideally, the same parameters can be used for multiple HDR frames. Another requirement of using the sequence mode is that the DMA buffers for the image data and bandwidth on the FireWire bus are allocated beforehand. This is only possible when the exact image sizes are known in advance.
For these reasons, using the sequence mode with variable image sizes is difficult, and we always capture full images here. The LDR images are then cropped after acquisition to save processing time in the subsequent steps of the HDR pipeline. We begin by choosing the darkest image of the sequence as the base image. It is searched for underexposed areas as described in Section 4.2.1. The next brighter image is now cropped to the size of the underexposed regions of interest of the base. The cropped brighter image is then searched for underexposed ROIs in turn which are used to crop the next image in the sequence, and so on.
Finding badly exposed regions can be done in one pass over the image plus several operations one the one-dimensional column histogram. It is thus an inexpensive operation. Additionally, we do not actually crop the images, but store the cropping parameters and only do further processing on the remaining ROI. In our experiments, we found that the processing time saved during color conversion, image registration and HDR stitching outweighs the cost of reducing the image size after capturing.

5.3 Experimental Results

This section presents the evaluation of our algorithm for optimal shutter speed sequences. Section 5.3.1 describes a subjective user study we conducted to assess the HDR image quality our approach achieves compared to the traditional way of choosing evenly spread shutters. For reasons described later in the section – most notably the unavailability of a perfect reference HDR video – only still images are used in this study. Section 5.3.2 contains a number of experiments to investigate the algorithm's behavior in our live HDR video system. They include an analysis of the algorithm's adaptation to changing brightness conditions and of its processing time.

5.3.1 Subjective User Study

27 participants took part in our subjective user study. Five of them were familiar with HDR imaging algorithms. The study was done over a website that allows to rate the quality of HDR images.[1] See Figure 5.5 for a screenshot of the website. Its first page contains a brief introduction to HDR imaging and the problem of choosing suitable shutter speeds. The participants were told to base their rating on: The amount of under- and overexposure present, the amount of image noise and quantization effects in color gradients. An example for each type of artifact was given. Variations in overall image brightness, contrast or color saturation were to be ignored as they may occur as a side-effect of tone mapping. The subjects were then shown twelve datasets of various scenes (see examples in Figure 5.6). Each dataset consisted of three HDR images: a reference image, an image created using shutter speeds from our approach and one where evenly spread shutters were used. The reference was always shown on the left side while the two survey images were shown in random order to avoid a subjective bias. Each of the two images had to be rated using the five scores (numerical value in parentheses): Very Good (5), Good (4), Average (3), Poor (2), Very Poor (1).

We used an AVT Pike F-032C FireWire camera capable of capturing 208 VGA frames per second with an aperture of f/2.8. The twelve scenes we captured had dynamic ranges exceeding the camera's capabilities. To attain radiance values with high precision, we chose static scenes and used a tripod. Each scene was captured as a set of 79 LDR exposures with shutter speeds varying by a factor of $\sqrt[8]{2}$. An exposure set covers the entire range of our camera's shutter settings (37 μs to 81.9 ms). All 79 exposures were used to generate the reference image and the log radiance histogram of each scene. The reference image is assumed to be an accurate representation of the scene radiance.

To create our datasets, we manually selected a suitable number of LDR exposures to be used for the two survey HDR images of each scene. The number was chosen low enough for a discernible degradation of image quality to facilitate the rating process. For comparison, the default stop criterion for total coverage was set to $C \geq 90\%$, while the average coverage achieved for our datasets was 80.4% for optimal and 75.9% for equidistant shutters. The chosen number of exposures was used as the only stop criterion of our algorithm; a sequence of shutter speeds was created accordingly. Out of the 79

[1] http://pi4.informatik.uni-mannheim.de/~bguthier/survey/

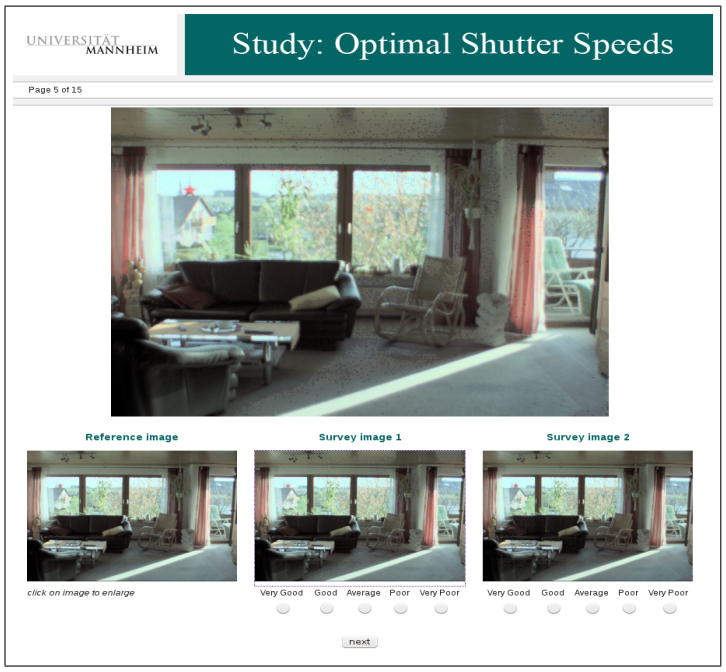


Figure 5.5: *Screenshot of the website we used for our subjective user study. A reference image and two survey images are shown, and participants can rate their quality.*

saved images of one dataset, those best matching the determined shutter speeds were merged to create the first HDR image. The second image was created using evenly spaced shutter speeds obtained in a way similar to [9]. To determine this sequence, the minimum and maximum scene radiance were considered, and the same number of exposures were spread evenly to cover the entire dynamic range. "Evenly" in this context means that the corresponding shutter speeds vary by a constant factor, i.e., a constant offset in the log domain. The shortest shutter speed was chosen in the same way as for our algorithm. The only exception are equidistant shutter sequences with only two shutters. For these, we found that choosing them closer to the center of the histogram gave better results. Due to the way we determined them, equidistant shutters also benefit from prior knowledge of the scene radiance, which is an advantage over plain exposure bracketing. This needs to be considered when comparing the achieved scores.

The main reason to use HDR still images instead of video for subjective quality assessment is the availability of a perfect reference image and with it the reproducibility of the results. Capturing 79 LDR exposures at varying shutter speeds allows to reconstruct the real scene radiance accurately. The shutter values are sufficiently close together to simulate arbitrary shutter sequences. Capturing the same amount of exposures for an HDR reference video is not feasible. Another reason is the difficulty to capture the optimal and the equidistant shutter video both at once. And lastly, HDR video may introduce various new artifacts like a misalignment of the exposures or a temporally inconsistent tone mapping. These additional artifacts may mask the difference between the two shutter speed choices.

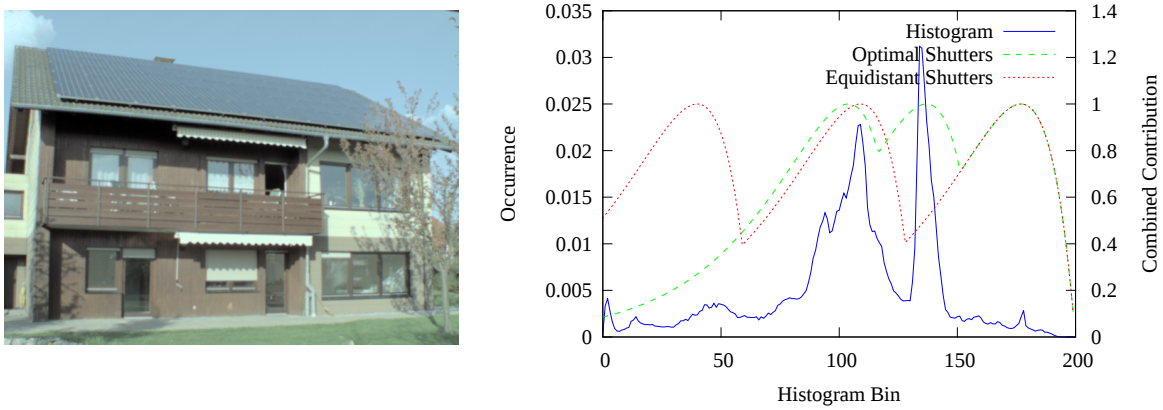
The 27 participants rating 12 datasets resulted in a total of 324 pairs of scores, one for optimal and one for equidistant shutters. Seven pairs were invalid because at least one score was not specified by the subjects. This was explicitly allowed in order not to encourage the participants to enter bogus scores when wanting to skip datasets. Averaging the 317 valid ratings results in a score of 3.73 for the optimal shutter algorithm and 2.83 for the equidistant approach. Note that the absolute value of the score is meaningless as the survey images were intentionally captured with a less than optimal number of exposures for better comparison. As a second aggregation of the results, we counted the instances where either of the approaches scored better than the other. This leads to our approach achieving a better score in 70%, the same in 16% and a worse score in 14% of the ratings. Our approach got rated worse most often in a dataset where it created a stronger quantization effect in the clouded sky. The sky only covers a relatively small area of the scene. It appears, however, that human observers pay more attention to it than its area indicates. We believe that this discrepancy between impact on the scene histogram and human attention poses a challenge for our algorithm. Tackling it exhaustively would require a costly visual attention analysis of the scene.
Figure 5.6 shows the reference images of all twelve scenes. The plot next to each image contains the log radiance histogram of the reference HDR image. It is normalized so that its bins sum up to 1. The plot also displays the combined contribution functions created by the two algorithms. It is calculated according to Equation 5.3. It can be seen that the equidistant shutters disregard the brightness distribution of the scene and sometimes exposures are captured that add little to the coverage value. The achieved coverage values and the calculated shutter speeds are presented in Table 5.1. Due to the special treatment of the first shutter in our algorithm, its achieved coverage can be lower than for equidistant shutters. This effect is most prominent in scenes where only two exposures are used.

5.3.2 Objective Measurements

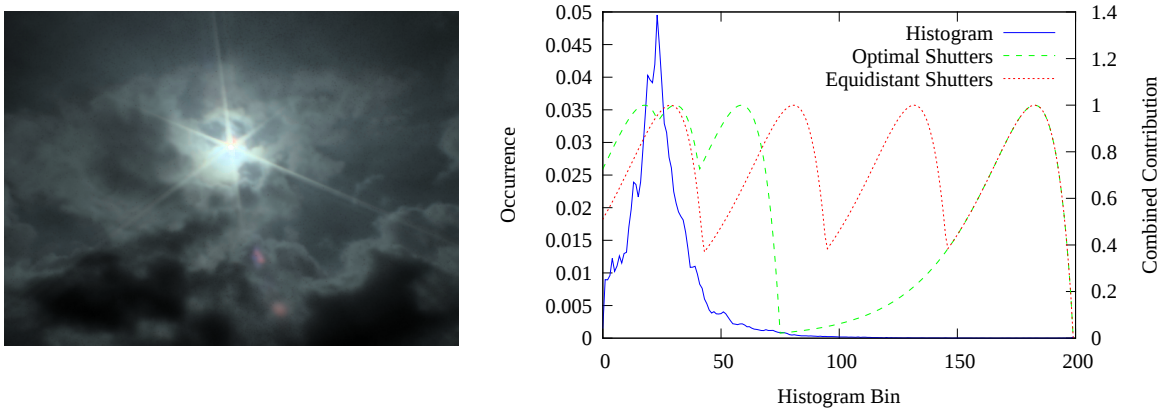
All experiments presented in this section were conducted in our real-time HDR video system. The shutter speed sequence algorithm uses the histogram of the current HDR frame as input. It was created during tone mapping of the frame. The calculated shutter values are then used to capture the LDR exposures for the next frame. An appropriate subset of the following three scenarios was used for the measurements.

1. Mostly static indoor scene with no camera motion.
2. A busy road with moving cars but no camera motion.
3. Moving scene with many camera pans between dark indoor and very bright outdoor areas.

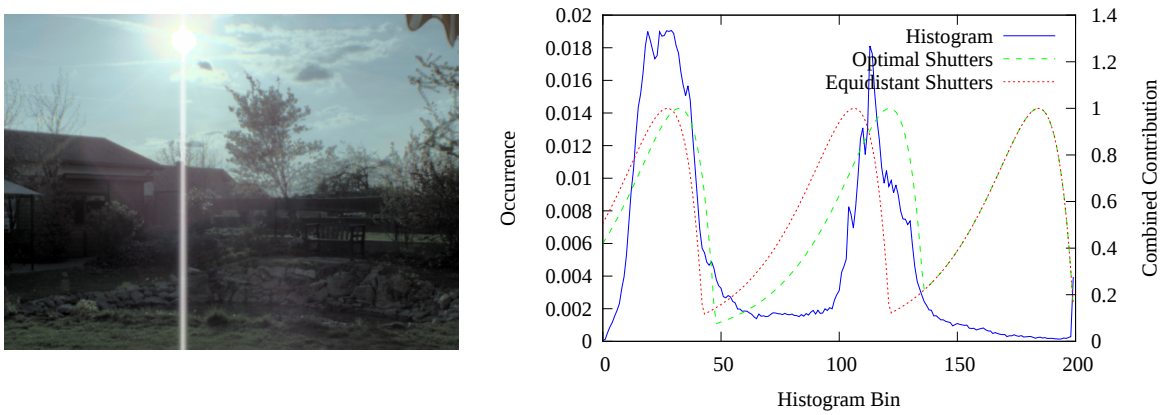
Unless stated otherwise, the measurements were taken over a period of 15 seconds (≈ 375 HDR frames).
As described in Section 5.2, the shutters that were determined greedily were refined in a second pass over the sequence. The goal of this is to improve the coverage value C which describes how well the chosen exposures overlap with the scene histogram. In order to



(a)

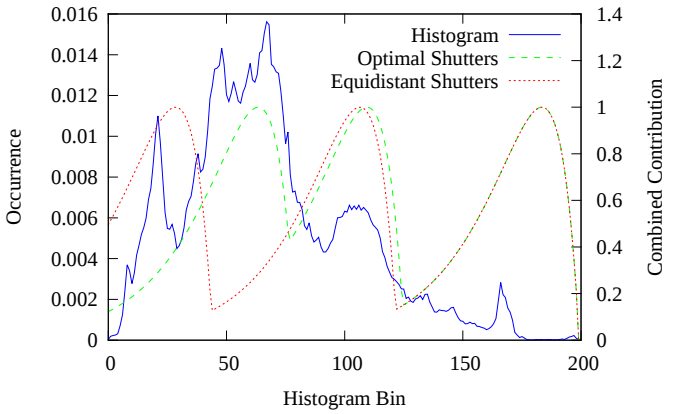


(b)

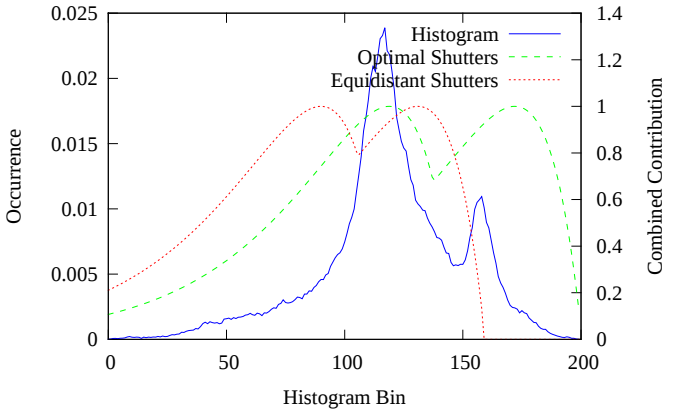


(c)

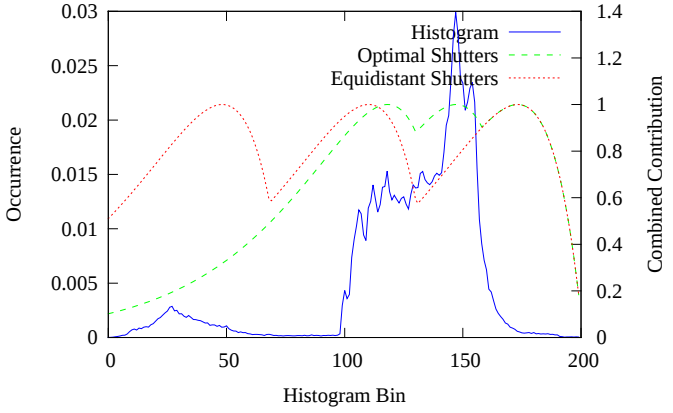
Figure 5.6: *The left column shows the reference images of the example scenes used in our subjective evaluation. The plots contain the corresponding normalized log radiance histogram. The dashed lines are the maximum of the contribution functions belonging to the shutter speeds determined by our algorithm and to the equidistant shutters.*


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

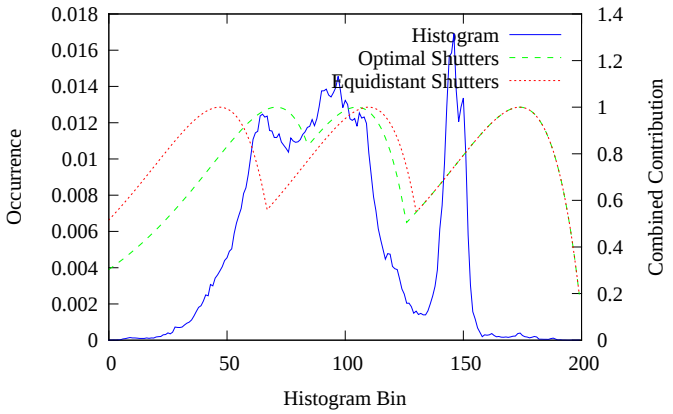
(d)


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.025
0.02
0.015
0.01
0.005
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

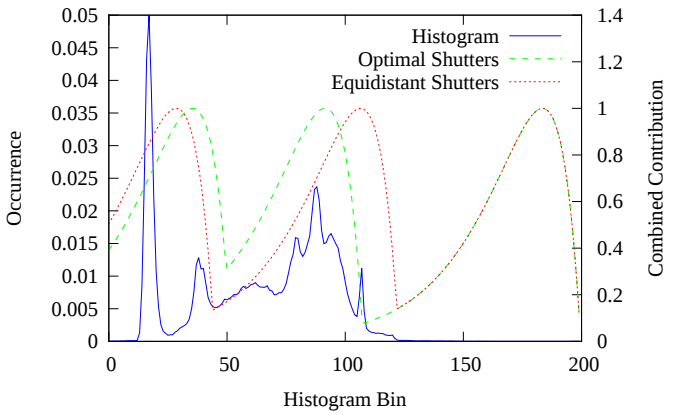
(e)


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.03
0.025
0.02
0.015
0.01
0.005
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

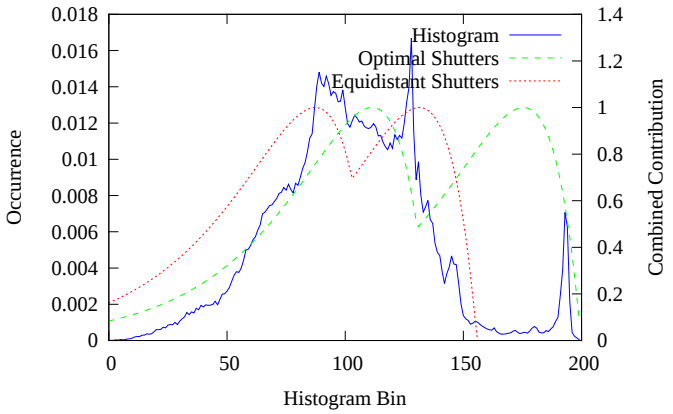
(f)


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

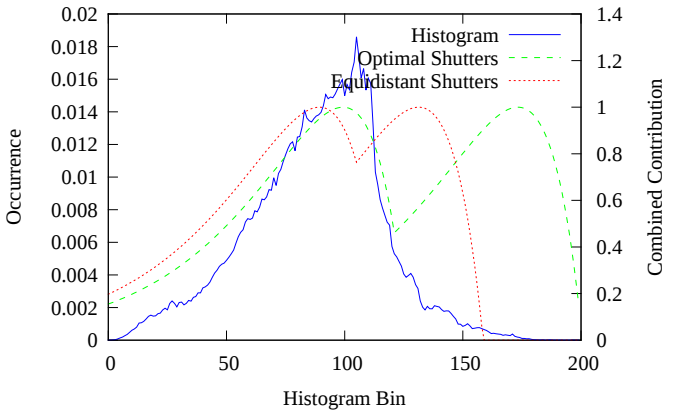
(g)


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

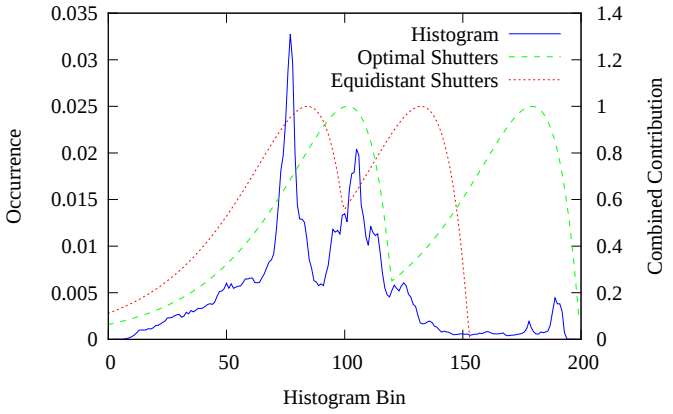
(h)


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

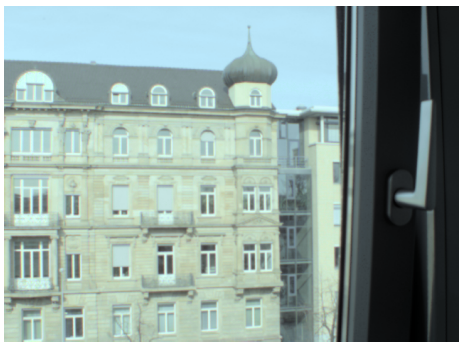
(i)

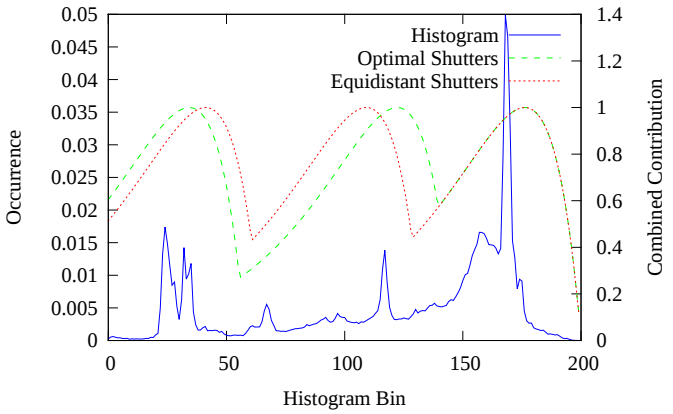

Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

(j)


Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

(k)




Histogram
Optimal Shutters
Equidistant Shutters
Occurrence
Combined Contribution
Histogram Bin
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200

(l)

Scene	C_{OPT}	C_{EQ}	Shutters (OPT)	Shutters (EQ)
(a)	84.0%	77.2%	0.26 2.36 0.87	0.26 2.07 16.5
(b)	92.9%	83.8%	0.02 12 3.8 21	0.02 0.18 1.5 13
(c)	77.3%	69.8%	0.02 17.4 0.34	0.02 0.67 21.8
(d)	68.5%	53.0%	0.06 12.1 1.70	0.06 1.90 57.2
(e)	86.1%	77.0%	0.96 3.45	2.60 7.06
(f)	90.9%	81.9%	0.64 1.20 2.47	0.64 3.04 14.5
(g)	88.8%	82.0%	0.78 4.40 10.4	0.78 3.88 19.3
(h)	74.7%	64.8%	0.03 1.59 17.4	0.03 0.85 23.8
(i)	72.4%	82.0%	0.68 4.19	2.40 8.46
(j)	77.9%	83.2%	1.81 11.7	5.25 15.3
(k)	66.4%	73.0%	1.26 18.4	6.41 32.7
(l)	85.0%	83.9%	0.61 37.4 2.92	0.61 4.33 30.8

Table 5.1: *The second and third column contains the coverage values C for the twelve scenes as achieved by the two algorithms: optimal shutters (OPT) and equidistant shutters (EQ). The third and fourth column show the calculated shutter speeds in milliseconds.*

evaluate the additional gain from the refinement step, we measured C before and after the refinement. This was done in the third (dynamic) scenario. Averaged over 15 seconds of video, the refinement achieved an increase of C by 1.5 percent points. To judge this result, one must consider two things: Firstly, the algorithm usually stops adding shutters to the sequence once $C \geq 0.9$. Because the maximum coverage is 1.0, there is not much room for improvement. Secondly, refinement does not add new shutters to the sequence, but adjusts the existing ones. Compared to capturing an extra frame to obtain a higher coverage, it is thus a rather cheap operation. We decided to include the refinement step into our running system, but omitting it is a viable option when processing time needs to be saved.

For our stability criterion, we defined the distance in percent between two shutter speed sequences. In order to get an understanding of this quantity and to decide upon a similarity threshold, we measured the distances between two sequences computed in two consecutive frames. This was done in all three scenarios, and the stability criterion was ignored. The results are listed in Table 5.2. When the size of two sequences differed, they were always classified as non-similar. So the first column of the table counts how often the size changed during the 15 seconds of the video. It is given as a percentage of the frames. The second column contains the average distance between two consecutive sequences. The standard deviation is given in the third column.

These values can be used to determine a suitable threshold for the distance to distinguish similar and non-similar sequences. We make the following observations. The first scene is completely static. Therefore, the shutter speed sequence should remain the same at all times. All measured distances should be considered as being similar. The second scene contains moving cars, and the shutter sequence needs to adapt occasionally. In the third scenario, the sequence needs to change a lot to accommodate the varying brightness

Scenario	Size Differs	Average Distance	Std. Dev.
1	0%	1.00%	0.89%
2	3.75%	2.79%	1.74%
3	18.87%	8.08%	9.11%

Table 5.2: *Percentage of sequences with differing number of shutters, average distance between the sequences and the standard deviation of the distance. They were obtained from 15 second shots in the three aforementioned scenarios.*

conditions. To meet these requirements, we set the threshold to 20%. Activating the stability criterion with this threshold, we repeated the experiments. During the 15 seconds, the algorithm was in the *changing* state 0% of the time in the first scenario, 0% in scenario 2, and 11.49% of the time in scenario 3. We found that these results were rather insensitive to changes in the threshold as long as it is high enough for a stable sequence most of the time. Once the scene's brightness actually changes noticeably, the size of the sequence often changes as well, and the distance between the sequences becomes very large.

In the experiment described in the following, we investigated the time it took for our algorithm to adapt to changes in the scene. We did this by keeping the scene and the camera static, choosing extreme shutter speed sequences and measuring the number of frames it took to stabilize. The scene and aperture of the camera were chosen such that the optimal shutter sequence consisted of four shutter values around the center of the camera's shutter range. By *center*, we mean the middle value in the log domain with the same *factor* to the lowest and to the highest shutter. For our camera, the shutter value of 1.74 ms is a factor of 47 higher than the minimum and lower than the maximum shutter. The algorithm was set to the *changing* state, and three different starting sequences were set: the sequence consisting of only the shortest possible shutter, the longest shutter and a sequence covering the full shutter range with one stop between the shutters. We then measured the number of frames the algorithm stayed in the *changing* state. The values are averaged over 375 runs for each of the three starting sequences.

As expected, the full coverage sequence adjusted the fastest. It took 2.07 frames to stabilize. This means that the stable sequence could be directly calculated from the first HDR frame in almost all of the iterations. From only the shortest shutter value, it took exactly 3 frames to stabilize. The algorithm already calculated three shutters in the second frame and reached the final sequence in the third. It then switched to the *stable* state in the fourth frame, because the calculated sequences were similar from then on. The worst adaptation speed was achieved when starting from only the longest shutter value, that is, from the brightest image. The lowest shutter in the sequence was approximately halved in every frame. On the average, the algorithm was in the *changing* state for 8.20 frames. This confirms our previous statement that convergence towards darker scenes (i.e., higher shutter values) is easier. It also justifies the special treatment of the first shutter in the sequence, as described earlier.

Since it is our goal to perform shutter sequence computations in real-time to create HDR videos, we measured the processing time taken by our algorithm. As mentioned earlier, the histogram of the previous HDR frame was computed during tone mapping. Histogram creation is thus not included in these measurements. The system we used for this experiment has an AMD Athlon II X2 250 dual-core CPU. The scenario with dynamic camera and scene was used to cover a large variety of shutter sequence lengths. The experiment showed that 96.5% of our algorithm's processing time is spent for trying out all possible shifts between the contribution vector and the histogram to find the next shutter speed with the best coverage value. As a consequence, the processing time is roughly proportional to the number of shutters in the sequence. We measured 0.30 ms per shutter value including refinement. For comparison, the entire process of creating a displayable HDR frame from an average of 3.6 base exposures takes 13.6 ms on a GPU. As mentioned above, in a 25 fps real-time HDR video system, there are 40 ms available for processing each frame. Our algorithm is thus fast enough to be used in this application.

5.4 Conclusions

We presented an approach to computing shutter speed sequences for temporally bracketed HDR videos. Our goal is to maximize the achieved HDR image quality for a given number of LDR exposures. This is done by consecutively adding shutters to the sequence that contribute to the image quality the most. Choosing evenly spread shutters wastes too much time for capturing exposures which contribute little to the HDR result. We are thus able to save capturing and processing time over the traditional approach by being able to reduce the number of LDR exposures without impairing quality. Analysis of the algorithm's behavior in a real-time HDR video system showed that it is suitable for such a scenario.

Using the histogram coverage as our criterion for optimization means focusing on the largest image areas first. We believe that being able to see as much as possible in a video is the main focus in surveillance. However, the user study showed that in certain situations, HDR images are also judged by *where* in the image the quality is achieved. We would like to take this into account in our future work.

Chapter 6

Histogram-based Image Registration

This Chapter describes our technique to compensate camera motion between the LDR exposures. It was published in [48]. So far, a sequence of LDR images has been captured with one of the two approaches introduced in Chapters 4 and 5, and the color format has been converted from the camera-dependent Bayer pattern into the Yxy color space. Our histogram-based algorithm for image registration solely operates on the brightness channel Y. Both approaches produce exposures with varying image size that exhibit a parent-child relationship: An LDR sequence consists of one full resolution base frame I_0 and n smaller re-exposures I_i for $i = 1, \ldots n$ captured with different exposure settings. Each re-exposure was initiated by badly exposed regions detected during the analysis of a previous image. An exposure I_i with $i > 0$ is thus a brighter re-exposure which was triggered by an underexposed area in I_{i-1}, making I_{i-1} the parent of I_i. The base frame is the root of the whole set. Each exposure is contained entirely in its parent. For simplicity we only consider the brighter re-exposures in this section. Registration of the darker re-exposures is done analogously.
Our algorithm performs no image registration on the base frame of an exposure set. Each re-exposure is registered with respect to its parent, i.e., a two-dimensional integer shift vector $\vec{s_i}$ between frames I_{i-1} and I_i is estimated. The absolute shift $\vec{v_i}$ between each exposure and the base frame can be easily calculated by

$$\vec{v_i} = \sum_{k=1}^{i} \vec{s_k}. \tag{6.1}$$

For estimating the translation vectors, we use *mean threshold bitmaps* (MTB) as described in [120]. A mean threshold bitmap is a black and white image that was created from a grayscale image such that 50% of the image pixels are white and 50% are black. The advantage of an MTB compared to a regular grayscale image is that – within certain limits – two exposures depicting the same scene captured at two different exposure

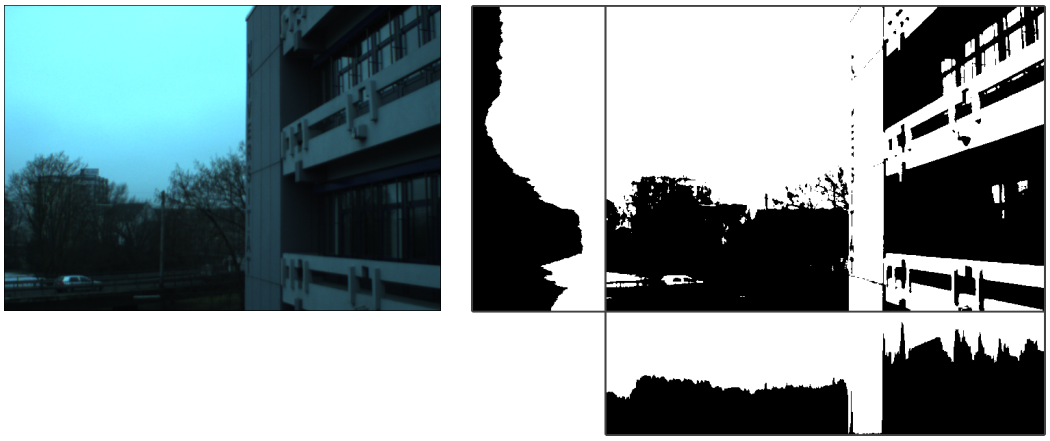


Figure 6.1: *An LDR frame (left) and its corresponding Mean Threshold Bitmap (right). The row and column histogram of the MTB count the number of black pixels in the image row and column.*

settings will result in approximately the same MTB. This fact is very desirable for image registration. The creation of MTBs is covered in Section 6.1.

Once the MTBs of two exposures to be registered are computed, we proceed by computing a *column* histogram, counting the number of black pixels in each column of the MTB. This is demonstrated in Figure 6.1. Such a column histogram is computed for both MTBs. By using a normalized cross correlation between the two column histograms, we estimate the horizontal component of the translation vector. Repeating this process for image *rows* allows us to estimate the vertical component, respectively. More details on the computation are given in Section 6.2.

As a last step, all resulting vectors are validated using a Kalman filter to incorporate knowledge of the prior motion into the estimation. A novel uncertainty criterion is used to determine the weighting between using the computed translation directly and extrapolating it from the preceding trajectory. This process is described in Section 6.3.

6.1 Mean Threshold Bitmap

Image registration starts with the creation of two MTBs for the two exposures I_{i-1} and I_i to be registered. Since a re-exposure is always contained entirely in its parent, processing time can be reduced by computing the full MTB of the re-exposure I_i, but only computing the MTB of the overlapping image area in the parent frame I_{i-1}.

The first step is to build a brightness histogram with 256 bins over the pixel values of I_i. From this histogram, we can deduce the median brightness value m_i to be used as a threshold so that 50% of the thresholded pixels are white and 50% black. At this point, the exposure values (e.g., shutter values) Δt_{i-1}, Δt_i at which the two frames were captured as well as the response function f of the capturing camera are known. We can

thus use these known values to calculate the unknown threshold m_{i-1} as follows:

$$m_{i-1} = f\left(\frac{f^{-1}(m_i)}{\Delta t_i}\Delta t_{i-1}\right). \tag{6.2}$$

This is an improvement over the original algorithm and saves the computation of a histogram over I_{i-1} [120].
In the original paper, ignoring pixels with a value near the median is suggested because they are unstable with regard to thresholding. In our experiments, we found that a noise threshold of $T_N = 2$ brightness steps below and above the mean leads to good results. Additionally, the original authors refrain from using medians that are closer than a threshold T_M to the margins of the 8-bit interval. If either of the computed median brightness values is less than T_M, they are computed again such that a higher percentage of pixels will be black in the thresholded image. The case $m_{i-1}, m_i > 255 - T_M$ is handled analogously.
For our algorithm, it is sufficient to calculate the two medians m_{i-1} and m_i which are then used to build the row and column histograms. The MTB itself is not built.

6.2 Row and Column Histograms

We estimate a two-dimensional shift $\vec{s_i} = (x_i, y_i)$ between two exposures I_{i-1} and I_i by estimating two one-dimensional shifts x_i and y_i separately. It is a greedy algorithm for image registration where each dimension of the shift vector is estimated independently of the other.
We start by estimating the horizontal shift x_i. The first step in doing so is to build column histograms over the full image I_i and the overlapping image area of I_{i-1}. A bin in the column histogram represents the number of black pixels in the corresponding column of the exposure's MTB. Since near-median pixels are ignored, as described in the previous Section, two individual histograms counting black and white pixels must be built for each exposure. Let w_i and h_i be the width and height of I_i. The column histogram $B_i^x(j)$ of exposure I_i counting black pixels is a function of the column index $j = 1, ..., w_i$ and is defined as

$$B_i^x(j) = |\{I_i(j,k) < m_i - T_N \ ; \ k = 1, ..., h_i\}| \tag{6.3}$$

where $I_i(j,k)$ is the pixel value at position (j,k) and $|\cdot|$ denotes the number of elements in the set. The histogram W_i^x counting white pixels and the two histograms for I_{i-1} are defined accordingly.
The horizontal shift x_i is now estimated using these four histograms. We let the shift s assume all possible integer values within a search range (e.g., -64 to 64 pixels) and compute the *normalized cross correlation* (NCC) between the histograms of exposures I_{i-1} and I_i under the given shift:

$$NCC(s) = \frac{C}{\sqrt{N_1 N_2}} \tag{6.4}$$

where C is the cross correlation value between the histograms of I_{i-1} and I_i

$$C = \sum_{j=1}^{w_i} \left(W_i^x(j) W_{i-1}^x(j-s) + B_i^x(j) B_{i-1}^x(j-s) \right) \tag{6.5}$$

and N_1 and N_2 are the two normalization values

$$N_1 = \sum_{j=1}^{w_i} \left(W_i^x(j)^2 + B_i^x(j)^2 \right) \tag{6.6}$$

$$N_2 = \sum_{j=1}^{w_i} \left(W_{i-1}^x(j-s)^2 + B_{i-1}^x(j-s)^2 \right). \tag{6.7}$$

The s producing the highest correlation value is then used as the estimate for x_i.
Using row histograms, the vertical shift y_i can be estimated analogously. The image regions over which the row histograms are computed are chosen according to the now known horizontal shift. Our experiments show that the choice of which dimension to start with has little effect on the final result. We also found that performing multiple iterations of the greedy algorithm does not improve the registration quality significantly. We therefore only estimate x_i and y_i once and set $\vec{s}_i = (x_i, y_i)$ as the resulting translation vector.

6.3 Kalman Filtering

A Kalman filter is used to incorporate the entire trajectory of the camera motion into the estimate of the current frame. Simply put, instead of using $\vec{s}_i$ directly, we essentially compute a weighted average over *all* preceding shift vector measurements, including those of the previous HDR frames. Each shift vector is weighted by the degree of uncertainty with which it was measured. This average is used as the shift of exposure I_i.
More precisely, the state of the Kalman filter represents the two-dimensional motion vector corresponding to the current exposure, i.e., the velocity of the camera translation. A 2×2 covariance matrix represents the uncertainty of the current state with respect to the real camera translation between exposures I_{i-1} and I_i. We assume zero acceleration of the camera, so our state transition model is the identity matrix. Changes in the camera motion are modeled by process noise instead. For each exposure, the new state of the Kalman filter is first predicted without including new information. The predicted state is identical to the previous state, but its uncertainty increases due to process noise. Next, a motion vector $\vec{s}_i$ is estimated as described in the previous section. This prediction is used as the starting point of the greedy search algorithm to increase its accuracy.
We developed a criterion that allows to judge the uncertainty of the estimate $\vec{s}_i$. From manually registered HDR test videos, we computed the mean μ and the standard deviation σ of the distances d between two consecutive motion vectors: $d = |\vec{s}_{i-1} - \vec{s}_i|$. With approximately zero mean and assuming that the distances are Gaussian distributed, over

99% of the motion vectors lie within 3σ from the previous vector. At the same time, erroneous measurements can be assumed to be uniformly distributed over the entire search range.
We thus use d as our criterion for the uncertainty of the measured shift. A $d > 3\sigma$ is likely to indicate an incorrect measurement, and the corresponding shift vector is discarded. If $d \leq 3\sigma$, the predicted state of the Kalman filter is corrected using $\vec{s}_i$ as the measured state and d as the variance of the measurement. In both cases, we use the new state of the filter as shift vector of the frame to be registered.
In our scenario, increasing the search range also increases the chance to detect errors in the shift measurement. Since computing the NCC is rather cheap, we set our search range to approximately $\pm 20\sigma$ to leave enough room for error detection.

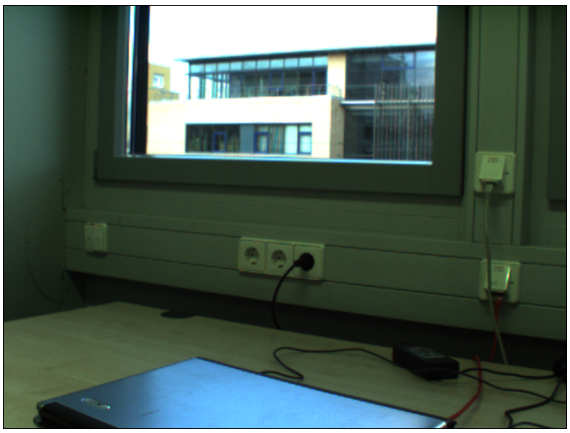
6.4 Experimental Results

For our experiments, we captured five HDR test videos. The videos are numbered from 1 to 5 and have the following characteristics:

1. Captured using a tripod. Mostly indoor scene with a window to the bright outside. Smooth horizontal camera rotation only. Static scene.
2. Handheld camera. Indoor with very dark areas and window. Completely random camera motion. Static scene.
3. Tripod. Outdoor scene with a relatively narrow dynamic range. Horizontal and vertical camera rotation. Static scene.
4. Tripod. A street with moving cars and trees. Horizontal and vertical camera rotation.
5. Tripod. Dark indoor scene with long exposure times and motion blur. Artificial light sources. Horizontal and vertical camera rotation. Mostly static scene.

Figures 6.2(a) – 6.2(e) show sample frames of the five videos.
All videos have a resolution of 640×480 pixels and an average of 87 HDR frames. Each HDR frame was created using one base frame and 3.35 re-exposures on the average. The first video has the most deterministic camera motion and the second the most random one. We thus use these two videos to fine-tune the parameters of the algorithm. This process is described in Section 6.4.1 All five videos are used for performance evaluation in Section 6.4.2. All frames were registered manually, and the resulting translation vectors constitute the ground truth for evaluating the accuracy of our automatic registration algorithm. As the criteria for our evaluation, we use the mean and the standard deviation of the distance between our estimate and the ground truth over all exposures of a video. Since it is our goal to capture and display HDR videos in real-time, the time taken for registration of a frame of a certain size is our second criterion. Both criteria are compared to our implementation of Ward's algorithm [120].

(a) Video 1



(b) Video 2

(c) Video 3

(d) Video 4

(e) Video 5

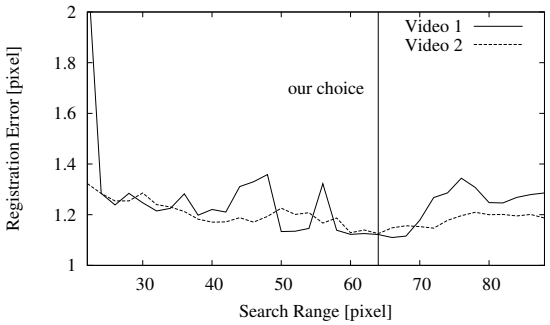
Figure 6.2: *Representative frames of the five HDR test videos used in our experiments.*

6.4.1 Setting the Parameters

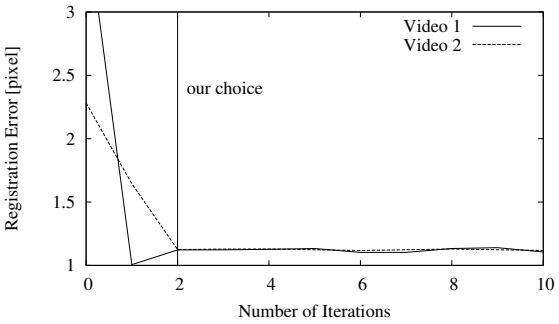
To optimize the settings of our algorithm's parameters, we use the first two test videos. One parameter is varied over the range of meaningful values, and all other parameters are kept constant. The average registration error is measured, and the parameter is set to the value leading to the smallest error. This process is repeated for all parameters until no further improvement is achieved. In this section, we show plots for the variation of each parameter while all other parameters are already set to their optimum. Our choice for the parameter's value is indicated by a vertical line in the plot.
The first parameter we consider is the search range. Figure 6.3(a) shows a plot of the registration error against the size of the search range. It can be seen that within reasonable bounds, changing the size of the search range does not influence the registration error by much.
Our algorithm determines the two components (horizontal and vertical) of the shift between two exposures separately. We chose to estimate the horizontal shift first. Starting with the vertical shift instead leads to an increase of the error by 0.069 pixels for video 1 and 0.002 for video 2.
An iteration of the algorithm is defined as the process of estimating one component of the shift. We first estimate the horizontal and then the vertical shift which counts as two iterations. Plot 6.3(b) shows the registration error against the number of iterations performed. It can be seen that increasing the number of iterations beyond two has no effect on the accuracy.
To decide whether a measured shift is reasonable or erroneous, we set an upper limit for the measurement uncertainty d. Its plot is shown in Figure 6.3(c). Setting this value too low results in misdetection of erroneous measurements, and a too large portion of exposures is registered using shift values extrapolated from the previous frames. Too large values lead to accepting too many bad shift measurements. Our choice is 10 which corresponds to the 3σ mentioned above.
The last three parameters influence the creation of the MTBs. See Section 6.1 for a description. Figure 6.3(d) depicts the effect of the noise threshold T_N on the accuracy of our algorithm. It controls the number of pixels that are ignored during MTB computation. If either of the computed means is too high or too low, both are computed again using a different percentage of black and white pixels in the thresholded image. Parameter T_M is the width of the illegal area at the extrema of the 8-bit brightness range. In the two considered videos, only very few of the means fall into this area. Thus, changing T_M has little effect on the registration error, as can be seen in Figure 6.3(e).
Similarly, changing the percentage of black pixels from the original 50% if the mean is too high or too low has little effect (see Figure 6.3(f)). We chose to change the percentage to 20% and 80% (50% $\pm$ 30%), respectively.

6.4.2 Evaluation

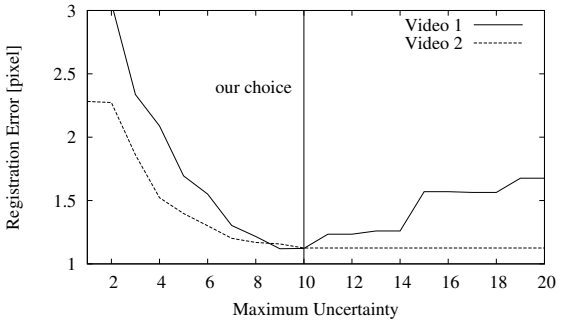
Ward's algorithm was developed for registering still images only. It does not make use of the history of motion vectors. So we start by comparing its accuracy to the one achieved



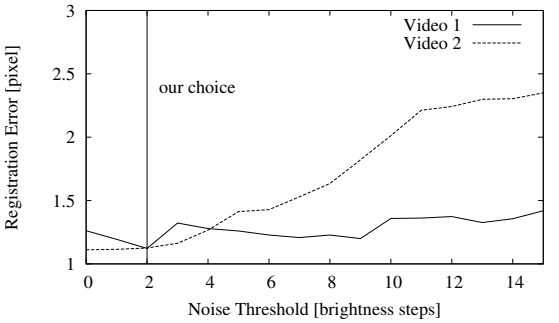
(a) Effect of changing the size of the search range on registration accuracy.



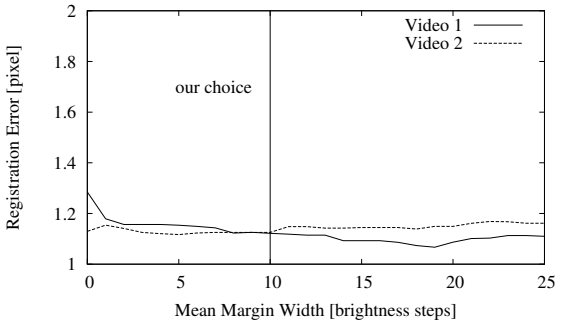
(b) Number of iterations performed by the greedy search algorithm.



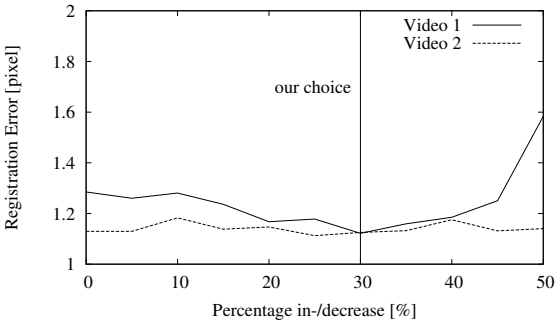
(c) Threshold for the uncertainty value d.



(d) Noise threshold T_N for pixels near the median.



(e) Width T_M of the margins of the 8-bit interval.



(f) Percentage of black pixels in case 50% yields a too high/low threshold.

Figure 6.3: *Effects of changing the parameters of the algorithm on the registration accuracy. One parameter is varied over the range of meaningful values while all other parameters are set to their optimum. Our choice for the parameter's value is indicated by a vertical line in the plot.*

Video #	Ward	without filtering	with filtering
1	1.56 (3.46)	5.21 (6.90)	1.12 (2.60)
2	1.05 (2.21)	1.31 (1.49)	1.13 (0.89)
3	1.37 (4.05)	1.12 (3.47)	0.78 (0.78)
4	2.27 (4.70)	1.78 (3.76)	1.38 (1.37)
5	3.96 (6.33)	4.52 (6.69)	2.77 (2.89)

Figure 6.4: *Average registration error in pixels (and standard deviation in brackets) for the five test videos. The algorithms compared are: Ward's algorithm, our approach without filtering and our full algorithm.*

by the still image version of our algorithm, excluding the filtering and prediction. The search range is set to 16 for both approaches. The second and third column of the table in Figure 6.4 show the results of this comparison. It can be seen that in this setup both algorithms perform similarly with respect to accuracy. The first video contains re-exposures with a height of only 48 pixels which is too small for our algorithm to handle properly. Video 5 is a dark indoor scene with motion blur in some of the long exposures. This explains the bad accuracy achieved by both.
In the second step, we add Kalman filtering to our approach and set the search range to 64. The rightmost column of the table indicates the accuracy improvement achieved. The effect of filtering outliers can be seen best in the reduced standard deviation. The motion vectors of the small frames of video 1 are now interpolated from the surrounding bigger frames, leading to a much better accuracy.
Figure 6.5 shows the time taken for both algorithms to perform image registration. As described in Chapter 4, the HDR capturing algorithm we employ always captures re-exposures at full width but with varying height [46], so the frame was cropped to heights from 100 to 480 pixels in steps of 10 before registration. Depending on the frame size, Ward's algorithm takes between 1.4 and 3 times longer than ours.
At full resolution, approximately 7.3 out of the 8 ms taken by our algorithm are due to the computation of means (1.4ms) and the construction of row and column histograms (5.9ms). Only 0.7 ms are taken for computing the normalized cross correlation. Filtering the results takes approximately 0.016 ms and is negligible.

6.5 Conclusions

We introduced an approach to the registration of LDR exposures for the creation of HDR videos in real-time. The focus was on improved registration speed compared to existing algorithms and suitability for the registration of differently exposed images. We believe that the accuracy achieved by our algorithm using Kalman filtering is acceptable in most viewing scenarios. The biggest obstacles for further accuracy improvement are the assumptions of purely translational motion and integer-valued motion vectors. The former is an inherent part of our algorithm. However in future work, we would like to add sub-pixel shift measurements to our approach to overcome the 0.5 pixels of average

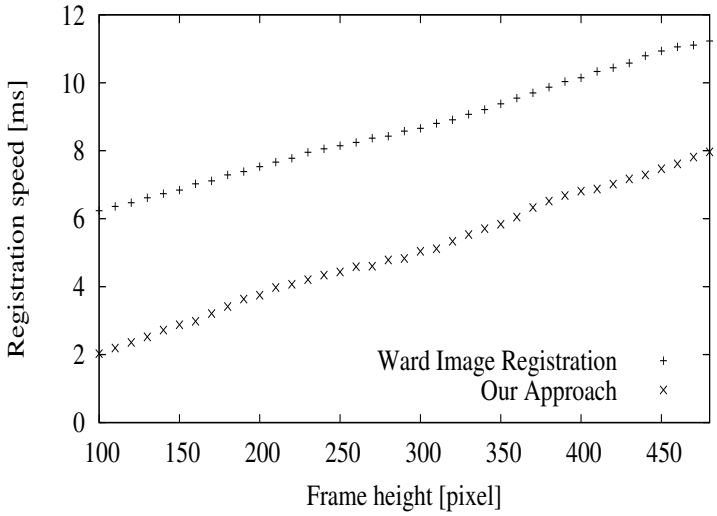


Figure 6.5: *Time taken for registration of images with a width of 640 pixels and the given height.*

quantization error immanent to our implementation. We would also like to explore the possibility of registering different pairs of exposures than just a frame and its parent.

Chapter 7

Flicker Reduction in HDR Videos

It is our goal to perform tone mapping on high dynamic range videos using standard operators designed for still images. When doing so, temporal incoherence of the minimum, maximum or average scene luminance leads to image flicker in the tone mapped result. We argue that image flicker is the most disturbing artifact introduced in this process. This is supported by two experiments described in Section 7.3. In the context of tone mapping, we thus focus entirely on the detection and reduction of flicker. The work described in this Chapter was published in [45].

7.1 Flicker Detection

We make the assumption that flicker is sufficiently well detected by computing the average image brightness of a tone mapped frame and comparing it to the average of the previous frame. The validity of this simple criterion is also backed up by our experiments described later in the chapter.
Figure 7.1 shows the average brightness of the frames of a tone mapped video over time. It illustrates well the rapid decrease and the rapid increase in average brightness between frames 52 and 53 and between frames 90 and 91. The respective frames are shown in Figure 7.2. They are taken from our HDR video *Turn* which contains a camera turn from a dark indoor area towards a window showing a light outdoor scene. The discontinuities in Figure 7.1 occur exactly when the bright window first enters the camera's field of view or leaves it, respectively.
To compute the average image brightness, we use the geometric mean. Over its arithmetic counterpart, the geometric mean has the advantage of being more resilient to outliers. Furthermore it more closely resembles the way average image brightness is perceived by the human eye. For this reason it is often used in tone mapping. It is given by

$$\tilde{I} = \prod_{(x,y)} (I(x,y) + \epsilon)^{1/n}, \tag{7.1}$$

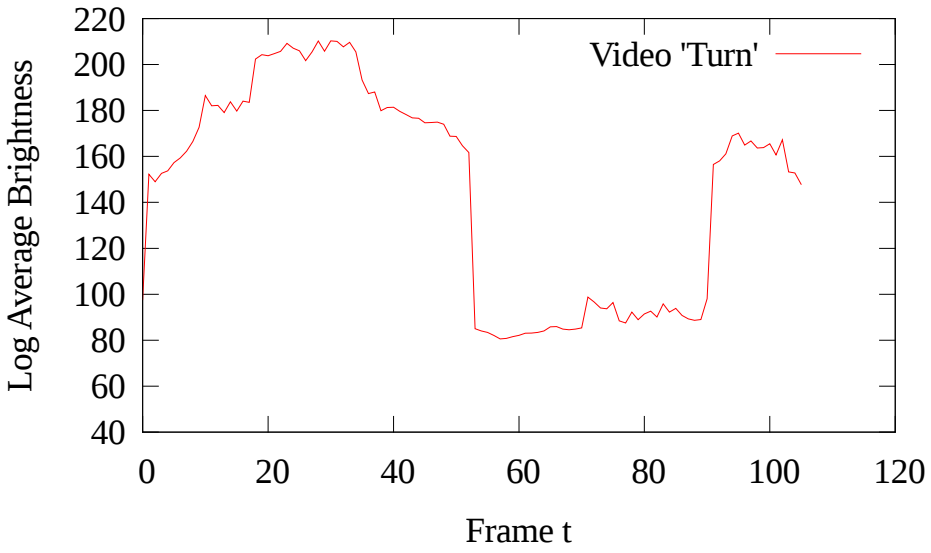


Figure 7.1: *Log average pixel value of an HDR video tone mapped with the Photographic Operator. From frame 53 to 90, a window showing the light outside is visible. This leads to a large increase of the scene's dynamic range and a large drop in brightness of the tone mapped result.*

where n is the number of pixels in the image and $I(x, y)$ is the value of the pixel at position (x, y). ϵ is a small number to prevent the product from becoming zero for black pixels. This equation can be rewritten as

$$\tilde{I} = exp\left(\frac{1}{n}\sum_{(x,y)} log(I(x,y) + \epsilon)\right). \tag{7.2}$$

The biggest challenge in developing a flicker detection criterion using the log average pixel value is finding a suitable threshold for the difference of the averages of two consecutive frames. We experimented with two different models found in the literature on the human visual system. They are Weber's law [30] and Stevens' power law [105]. Both use the notion of a *just noticeable difference* ΔR, which depends on a given background luminance R, and both introduce an adjustable parameter k. For a given luminance level (in our case the log average of the previous frame), these models allow the computation of a maximum luminance change that will remain unnoticed by a human observer. Even though the setting for which these laws were developed slightly differs from ours, they serve as a perceptional basis for our criterion.
Weber's law is a mathematical model for the human visual perception [30]. It states that the ratio between the minimum incremental amount of luminance required to be perceptible and the background luminance is a constant:

$$k = \frac{\Delta R}{R} \tag{7.3}$$

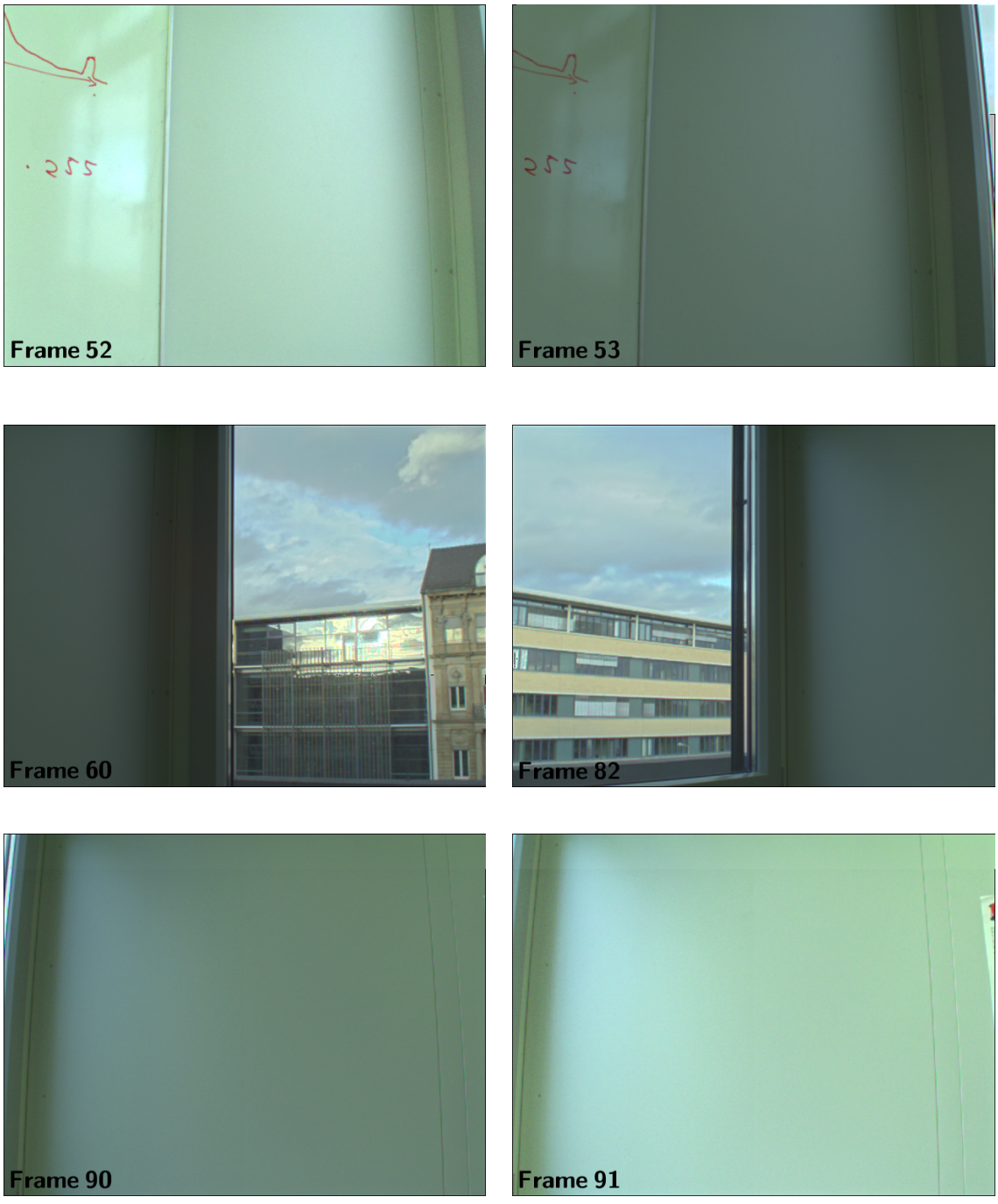


Figure 7.2: *Six frames of the video* Turn *tone mapped with the Photographic Operator. As soon as the window enters the camera's field of view, the scene's minimum and maximum luminance change considerably, leading to a visible brightness difference in the tone mapped frames. This can be seen between frames 52 and 53 and frames 90 and 91.*

In other words, the just noticeable difference is a fixed fraction of the predominant brightness. This equation can be solved for ΔR to yield the desired threshold.
Our second criterion is Stevens' power law [105]. It describes the relationship between the real magnitude of a general stimulus and the magnitude as perceived by a human. A number of tone mapping operators use the power law with the stimulus being the sensation of brightness [111, 81]. In its general form, it is given by

$$\Delta R = kR^{\alpha}, \tag{7.4}$$

where α is a constant specific to the type of stimulus considered. For brightness $\alpha \approx 0.33$. A suitable value for the parameter k is determined experimentally for each criterion. This is presented in Section 7.3. In our experiments, we also found that Stevens' power law allows for the most accurate detection of flickering frames. It is therefore our criterion of choice.

7.2 Flicker Reduction

In this section we demonstrate our flicker reduction algorithm, which makes use of flicker detection. It will be seen that a robust detection makes flicker removal straightforward. If flicker occurs in a frame, we iteratively adjust its brightness until it is within the tolerable threshold. To illustrate our algorithm, we use frames 52 and 53 of the test video *Turn*, as shown in Figures 7.1 and 7.2, as an example throughout this section: A part of the window showing the very bright outside suddenly moves into the camera's field of view, making the maximum scene luminance go up rapidly. The TM operator maps the now increased dynamic range of the scene onto the same display range, resulting in a much darker image. The case where the image brightness *increases* rapidly is handled analogously and not explicitly described here.
The algorithm is implemented as a post-processing step and works with any tone mapper. We tested our approach with Ward's Contrast-Based Scale Factor [117], the Histogram Adjustment by Ward et al. [121] and the Photographic Tone Reproduction by Reinhard et al. [95]. Since it was designed for live capturing of HDR material, it only adjusts the brightness of the current frame and only uses knowledge of the previous frames. A more optimal flicker reduction could be achieved if the whole video was known in advance. However, this is not the case in our scenario.
We start by tone mapping the current frame t with the chosen operator and settings. Next, the log average pixel value $\tilde{I}_t$ of the frame is computed using Equation 7.2. Then we calculate the maximum allowable brightness difference ΔR to the previous frame using Stevens' power law (Equation 7.4):

$$\Delta R = k(\tilde{I}_{t-1})^{0.33}, \tag{7.5}$$

where $\tilde{I}_{t-1}$ is the log average of the previous frame. Now we check whether $|\tilde{I}_{t-1} - \tilde{I}_t| > \Delta R$. If it is not, then the frame is likely not to be a flickering frame (see experimental results). In our example, however, it is assumed that the current frame is much darker than the previous one ($\tilde{I}_{t-1} > \tilde{I}_t$). The goal is now to increase the frame's brightness so

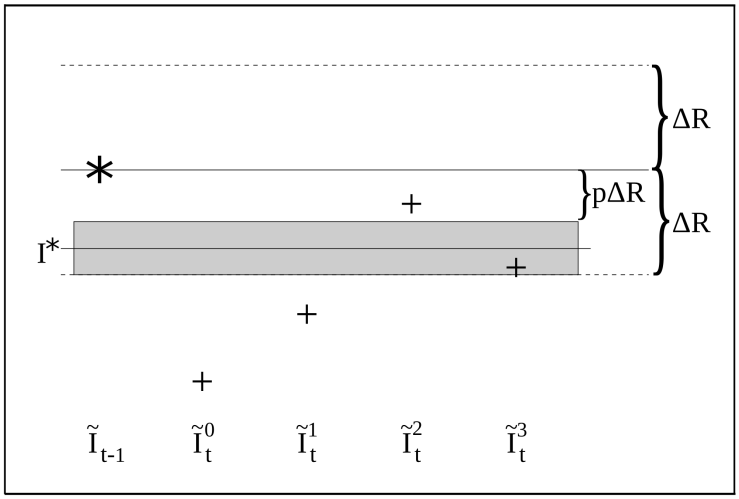


Figure 7.3: *Frame t is too dark after tone mapping ($\tilde{I}_t^0 < \tilde{I}_{t-1} - \Delta R$). Its brightness is thus iteratively adjusted towards the target value I^*. After three iterations, it falls within the tolerable brightness range drawn in gray.*

that it falls within a tolerable range. The lower end of this range is given by $\tilde{I}_{t-1} - \Delta R$, meaning that there shall be no detectable flicker. It is also desirable to maintain the original brightness produced by the TM operator as well as possible. After adjustment, $\tilde{I}_t$ should therefore be close to the lower end of the range. To accommodate this fact, we set the upper bound to $\tilde{I}_{t-1} - p\Delta R$, where p is a percentage which we set to 50% in our implementation. The next step is to iteratively adjust the frame's brightness, producing a sequence $\tilde{I}_t^0, \tilde{I}_t^1, ..., \tilde{I}_t^i$, until it falls into the desired range of $[\tilde{I}_{t-1} - \Delta R, \tilde{I}_{t-1} - p\Delta R]$. As an explicit target value I^*, we aim for the range's center. The process of iteratively approaching the desired brightness is depicted in Figure 7.3. It should be noted that adjusting the average brightness of an image directly is not possible due to saturation effects. This necessitates iterative brightness adjustment. Our experiment in Section 7.3.3 shows that less than two iterations are required on the average.
Most TM operators perform a final normalization and clamping of the resulting floating point values. The tone mapper may produce scaled luminance values in an arbitrary range. In the normalization step, these values are shifted to begin from zero and scaled to 255. Afterwards, the values may be clamped to $[0, 255]$ again for robustness against imprecision in the floating point operations. We modify these steps by introducing a scale parameter s. The values after tone mapping are then normalized to the range of $[0, 255 * s]$ instead. The subsequent clamping to a byte remains unchanged. In doing so, the change of the average output brightness will be approximately linear to the change of s (only approximately because of shifting and clamping). That is, for a certain range of scale changes, there is a slope parameter m^0 for which

$$\tilde{I}_t^0 - \tilde{I}_t^1 = m^0(s^0 - s^1) \tag{7.6}$$

holds. In the following, we use the superscript to denote the iteration index in our

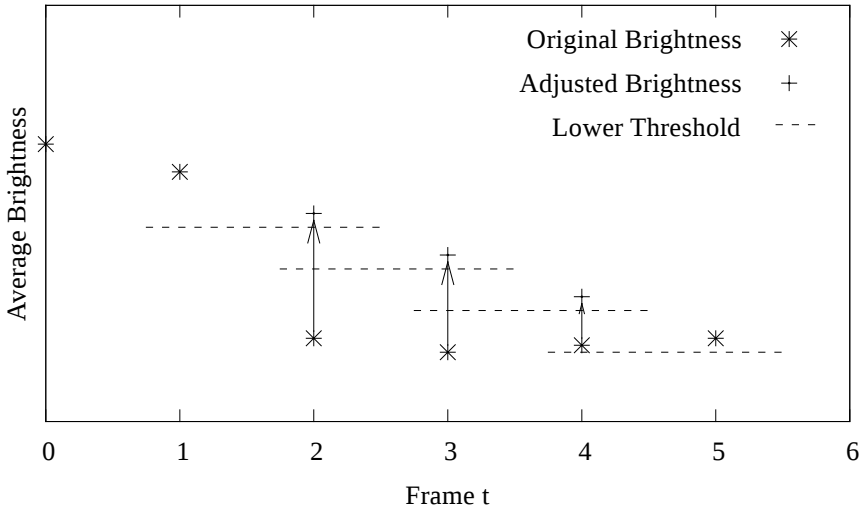


Figure 7.4: *This plot shows a rapid decrease of average brightness between frames 1 and 2. Frame 2 is adjusted to reduce the brightness gap. The amount of adjustment needed decreases with each subsequent frame. In frame 5, the adjusted brightness has converged towards the value achieved by tone mapping with a standard scale. No more adjustment is required.*

iterative brightness adjustment scheme. $\tilde{I}_t^0$ is therefore the log average pixel value of frame I_t as determined earlier (called $\tilde{I}_t$ above). $s^0 = 1.0$ is the original scale used for normalization. $\tilde{I}_t^1$ and s^1 are the respective values after adjusting the scale once. The initial slope m^0 is set to an arbitrary value of 50. It is refined during the iterations and re-used for later frames. In the following, we omit the time parameter t to avoid clutter. At this point, we know the frame's original average brightness $\tilde{I}^0$, the desired target brightness I^* to which we would like to adjust it, the original scale s^0 and a rough estimate of the coefficient m^0 to relate scale change to brightness change. From this, we can compute a new scale s^1 that maps the brightness closer to the target by rearranging Equation 7.6 and replacing $\tilde{I}_t^1$ with I^*:

$$s^1 = s^0 - \frac{I^0 - I^*}{m^0}. \tag{7.7}$$

The normalization step as described before is then re-done using the adjusted scale parameter s^1. This results in a new log average image brightness of $\tilde{I}^1$ in the first iteration. We now know that adjusting the scale from s^0 to s^1 changed the average from $\tilde{I}^0$ to $\tilde{I}^1$. Inserting these values into Equation 7.6 yields a more accurate approximation of the slope parameter which we denote by m^1.
In each iteration i, the scale parameter s^i is updated (Equation 7.7), and the original tone mapping output is normalized and clamped using s^i, resulting in $\tilde{I}^i$. A more accurate slope m^i is then estimated from

$$m^i = \frac{\tilde{I}^i - I^0}{s^i - s^0}. \tag{7.8}$$

The iteration ends as soon as $\tilde{I}^i$ falls within the tolerable target range given above, in which case the normalized image is the output of the post processing step. The iterative brightness adjustment is illustrated in Figure 7.3. See the next section for the mean number of iterations required. The final slope m^i from the last iteration is saved and re-used as an initial guess for the next flickering frame.
The adjusted brightness $\tilde{I}_t^i$ of frame t is now slightly lower than $\tilde{I}_{t-1}$ (more specifically: $\tilde{I}_t^i \approx I^* < \tilde{I}_{t-1}$). However, the difference is now within a range we consider to be unobtrusive. For the next frame $t+1$, we set the normalization scale to 1.0 again, i.e., we tone map with standard parameters. If there is no more rapid scene histogram change between frames t and $t+1$, $\tilde{I}_{t+1}$ is now closer to $\tilde{I}_t$ than $\tilde{I}_t$ was to $\tilde{I}_{t-1}$, and the amount of adjustment required is smaller. After a few frames, the difference approaches a value less than ΔR, and no further adjustment is necessary. Figure 7.4 illustrates this convergence towards standard parameters.

7.3 Experimental Results

7.3.1 Subjective Flicker Detection

Our experimental results are based on two subjective user studies we performed. For both studies, we used the same five HDR videos, tone mapped without flicker reduction. The lengths of the video clips range from 14 to 44 seconds. Here are the names and a brief description of the videos:

Turn:	Mostly indoor scene with a window showing the bright outside. Smooth horizontal camera rotation with a sudden transition from dark to bright and back.
Handheld:	Handheld camera. Indoor with very dark areas and window. Random camera motion.
Outdoor:	Outdoor scene with a relatively small dynamic range. Horizontal and vertical camera rotation.
Outdoor South:	A street with moving cars and trees.
Linux Room:	Dark indoor scene with long exposure times and motion blur. Artificial light sources. In some frames, the window to the outside is visible.

Since different TM operators produce different flicker artifacts, we tone mapped these videos with the three different operators: Ward's Contrast-Based Scale Factor [117] (referred to as *Scale* in the following), the Histogram Adjustment by Ward et al. [121] (*Histogram*) and the Photographic Tone Reproduction by Reinhard et al. [95] (*Photographic*). This resulted in 15 videos total. The URL where all our videos can be found is given in the References section [1].
Both experiments included the same ten test subjects. They had no prior experience with HDR video or tone mapping. Because there is no standardized quality evaluation method for tone mapping, our experiments were performed in accordance with the Subjective

Assessment Methodology for Video Quality (SAMVIQ) [28]. It is a subjective video quality measurement for motion pictures, generally used to evaluate video encoding techniques. We only differ from the recommendation in that we do not show an explicit reference video. This was not possible due to the lack of an HDR display to show a non-tone-mapped reference video on.
In the first experiment, the subjects were told to mark the video frames in which they perceive a general distortion. They were not informed of our endeavor to remove flicker artifacts in particular. From the resulting list of marked frames, we removed three frames where problems were clearly due to errors in the capturing process and unrelated to tone mapping (e.g., motion blur or ghosting due to exposure bracketing). In the second experiment, we told the subjects to mark frames with visible image flicker. We observed that more frames were marked in the second experiment than in the first. In particular, each frame that was marked in experiment 1 by anyone was also marked by half or more of the subjects in experiment 2, i.e., the first results were a subset of the second. The fact that each reported distortion turned out to be classified as image flicker implies that flicker is the most obtrusive artifact introduced when still image tone mappers are applied to video.
To compile a final list of frames that are considered flickering, we used those frames that were marked by 50% or more of the subjects in the second experiment. This includes everything that was marked in the first experiment. Table 7.1 shows our list. Some clips exhibit a large brightness difference between frame 0 and 1. This is an artifact of the capturing process, but was left in the list as it also constitutes flicker.

Video	Operator	Reported Flickering Frames
Turn	Scale	1, 53, 57, 71, 72, 90-92
	Photographic	1, 53, 91
	Histogram	19, 51-54, 91, 92
Handheld	Scale	12, 14, 34, 35
	Photographic	-
	Histogram	-
Outdoor	Scale	1, 13, 18
	Photographic	1, 13, 60
	Histogram	-
Outdoor South	Scale	6, 27, 28, 42, 54, 57
	Photographic	27, 28, 41, 42
	Histogram	-
Linux Room	Scale	10, 17, 50, 68
	Photographic	10, 18, 50, 78
	Histogram	14, 26, 78, 79

Table 7.1: *List of flickering frames for each video tone mapped with each operator. We included all frames that were reported by 50% or more of the subjects in our second experiment. 50 frames were reported as flickering in total.*

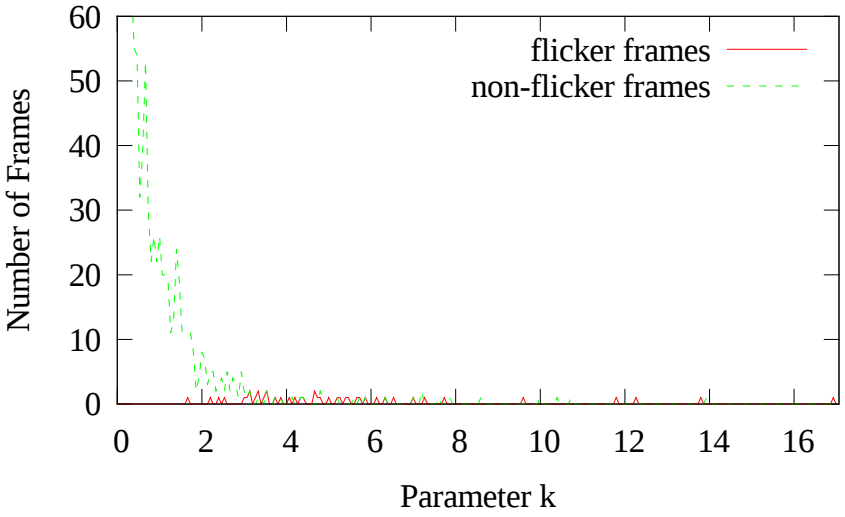


Figure 7.5: *The histogram shows the number of flicker and non-flicker frames for each value of k for Stevens' power law.*

7.3.2 Setting the Parameter k

In Section 7.1, we introduced two criteria for the detection of flicker: Stevens' power law and Weber's law. Both had an adjustable parameter k to control their sensitivity to best match the subjective impression of flicker. We now describe how the marked frames shown in Table 7.1 are used to determine the optimal values for k and to judge the performance of the two criteria. For this purpose, a data set consisting of all 1311 frames of all video clips is created. For each frame, we compute the difference ΔR of its log average brightness and its predecessor's R. Knowing ΔR and R, solving Equations 7.3 and 7.4 for k yields a minimum sensitivity value for which the frame would just be detected as flickering. We thus also record k_{weber} and $k_{stevens}$ for each frame. Additionally, we mark the flickering frames according to Table 7.1. Figure 7.5 shows the number of frames having a certain value $k_{stevens}$ for the two classes of flicker and non-flicker frames. For a given value of k, we can now count the number of correctly detected flickering frames (true positives, tp), frames incorrectly classified as flickering (false positives, fp), and missed frames (false negatives, fn). The metrics *precision* P and *recall* R are calculated as [90]:

$$P = \frac{tp}{tp + fp}, \tag{7.9}$$

$$R = \frac{tp}{tp + fn}. \tag{7.10}$$

Precision is defined as the fraction of correctly detected flicker frames among all detected frames, whereas recall is the fraction of correctly detected flicker frames among all actual flicker frames. Figure 7.6 plots precision and recall in our data set against the chosen value of k. A higher k generally increases the precision, but has a negative impact on recall.

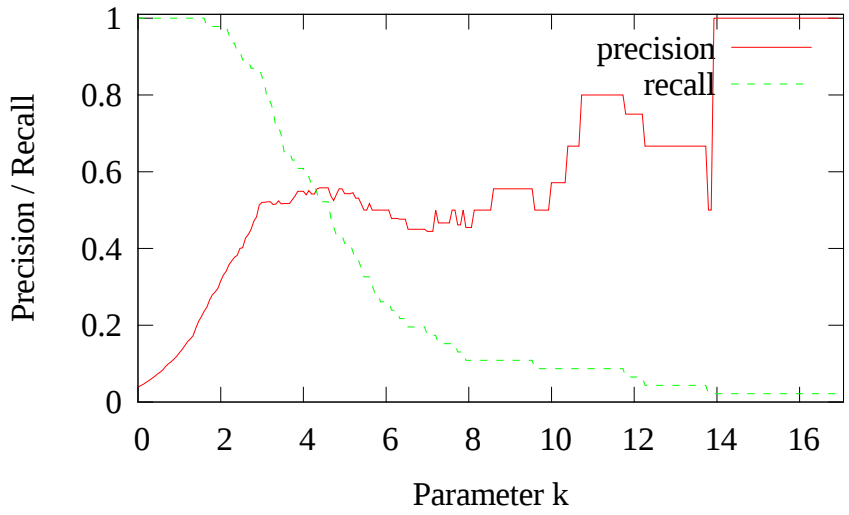


Figure 7.6: *Precision and recall of our detector plotted against parameter $k_{stevens}$.*

The F_β-score ($F_\beta \in [0, 1]$, a higher score is better) combines both precision and recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R}. \tag{7.11}$$

The parameter β expresses subjective preferences, e.g., whether the number of frames incorrectly classified as flickering is more relevant compared to flicker frames that are not detected (missed frames). In the case of $\beta = 1$, the F_1 score can be interpreted as the weighted average of precision and recall; $\beta = 2$ gives recall (missed frames) twice the weight of precision. Figure 7.7 shows the value of four different F-scores depending on the value of k. The global maximum of each F-score indicates the k which optimizes the subjective preferences. Higher β values lead to a lower k and reduce the number of missed frames at the expense of more false positives.

Our goal is to remove as much flicker as possible to generate high quality videos. Thus the number of missed flicker frames should be low. False positives are more acceptable, since additional tone mapping of a non-flicker frame merely increases the computational effort. Based on these considerations we chose the F_4-score as our quality measurement criterion which assigns four times as much importance to recall as to precision.

To derive an optimal value for the parameter k, we consider the full data set. The optimal values for Stevens' power law and Weber's law are $k_{stevens} = 2.133$ and $k_{weber} = 0.0846$, respectively.

We use tenfold cross validation [65] to judge the performance of our flicker detection in practice. The idea is to partition the video frames into ten subsets: In one iteration, the maximum F_4 score (i.e., the optimal value of k) is calculated using nine training sets. The remaining subset is used to calculate a realistic F_4-score and to judge the performance of the detection. This is repeated ten times, such that all frames are used for validation exactly once. The average over all F_4-scores is used as the final score for the respective detection method. Table 7.2 summarizes the results.

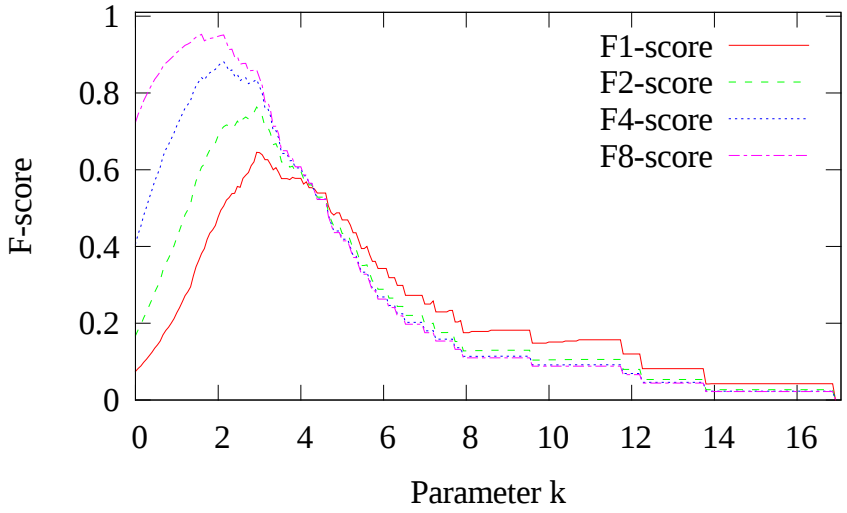


Figure 7.7: *Comparison of different F-scores based on Stevens' power law.*

Method	Avg. F_4-score	# false negatives	# true positives	# false positives
Stevens	0.8689	1	49	87
Weber	0.7774	2	48	149

Table 7.2: *Comparison of cross-validation results.*

It can be seen that the average F_4-score of Stevens' power law is higher, making it the better choice for our purpose. From the number of false negatives and false positives for our choice of k, we can conclude that a large difference of the log average pixel value between two frames is a necessary criterion for a flickering frame. That is, if our detector classifies a frame as non-flickering, it is very likely to actually be a non-flicker frame. This justifies our decision to adjust a frame's brightness until our detector stops reporting flicker: it is very likely that the flickering is now removed. In other words, we use flicker detection as an objective quality metric to prove the correctness of our flicker reduction algorithm.
Our results are presented as eight plots in Figure 7.8. They show the trend of the log average brightness before and after flicker reduction. We selected eight representatives out of the 15 videos we created. The original and the non-flickering versions of all 15 videos can be downloaded from the URL given in the References section [1].

7.3.3 Computational Effort of Flicker Reduction

We assume that the log average brightness of a frame can be obtained from the TM operator as a by-product or calculated with little extra effort. The cost of flicker detection is thus negligible, and so is the calculation of the new scale parameter in our iterative brightness adjustment scheme. Hence, the additional computational effort produced by

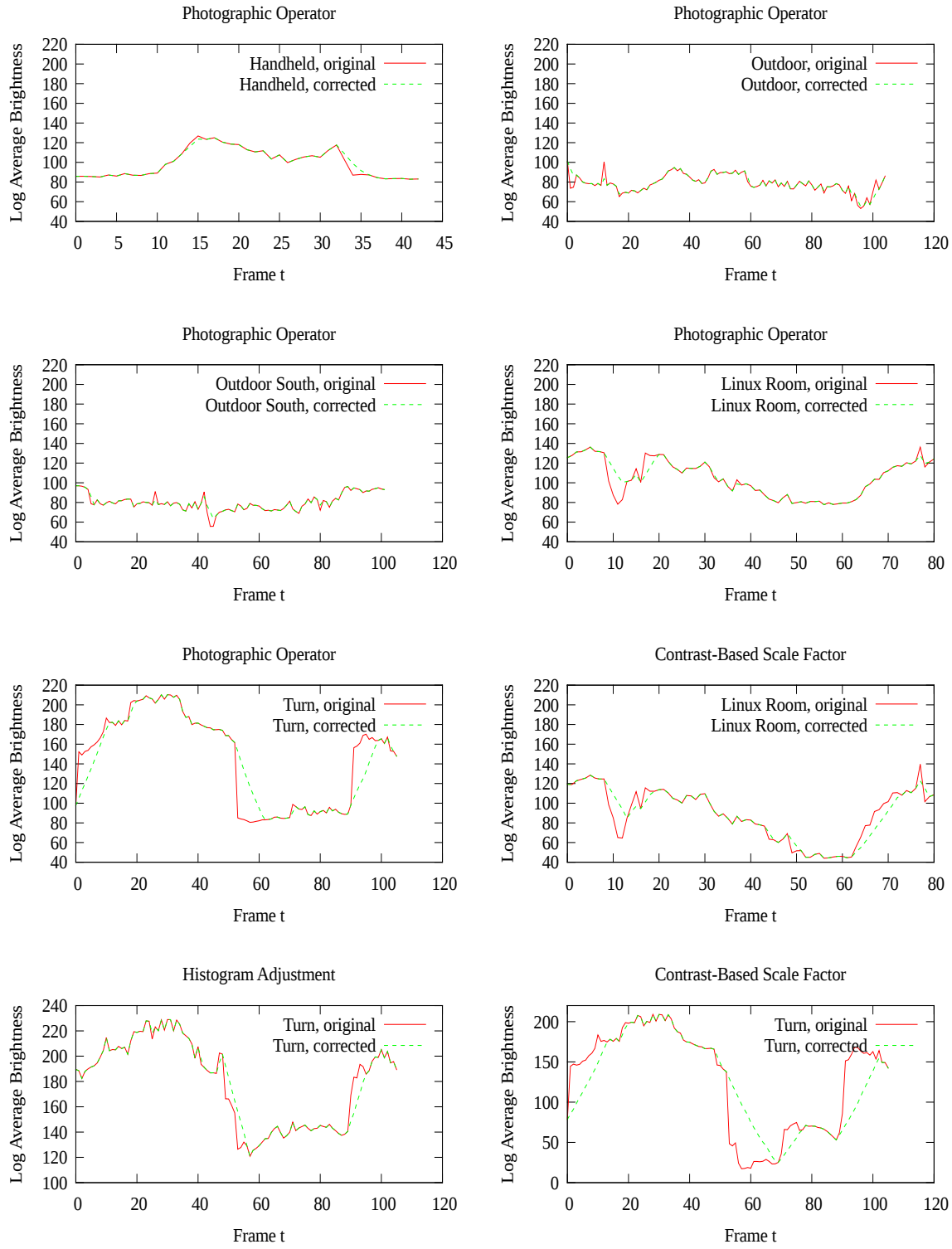


Figure 7.8: *Average brightness before and after flicker reduction. We chose to show all five videos, tone mapped with the Photographic Operator. Additionally, we show all three versions of* Turn*, since it contains the most noticeable flickering artifacts.*

our flicker reduction algorithm is mainly due to the repeated normalization of flickering frames. When processing the 1311 frames of our test videos, 274 additional normalizations were performed. They were caused by adjusting 208 flickering frames. This number differs from the 50 flickering frames in Table 7.1 because one flicker effect may have to be smoothed over an entire succession of frames, and we also process false positives. This gives an average number of 1.317 iterations per adjusted frame and 4.16 adjusted frames per subjective flicker artifact.

7.4 Conclusions

We presented an algorithm for the automatic removal of flicker in tone mapped high dynamic range video. It is based on a threshold for the maximum allowable difference between the log average pixel value of two consecutive frames. When the threshold is exceeded, the frame's brightness is adjusted until it falls into a tolerable range. The flicker artifact is thus smoothed over several frames and becomes unobtrusive.
We evaluated our work by first fitting the threshold for flicker detection to the results of a subjective study. Flicker reduction adjusts the brightness until it is within the threshold. Assuming that we can detect flicker reliably, our flicker reduction algorithm is very likely to correctly remove flicker from the videos.

Chapter 8

GPU Implementation

It is our goal to create high dynamic range video in real-time. In the previous chapters, we have described algorithms we use in our HDR video system that focus on minimizing the amount of data to process or perform the necessary computations as efficiently as possible. To further increase the performance of our system, we decided to make use of the parallel processing capabilities of a graphics processing unit (GPU). As opposed to a CPU with a small number of high-performance processor cores, optimized for sequential programs, a GPU has many simple cores that complete a large number of simple tasks in parallel. Implementing algorithms for a GPU requires a higher degree of understanding of the underlying platform than a serial CPU implementation. We thus begin this chapter by giving an introduction to Nvidia's Compute Unified Device Architecture (CUDA) which we use in our work. Details can be found in Nvidia's *CUDA C Programming Guide*. [1] We then analyze the steps in the HDR pipeline with respect to suitability for parallelization and show the required modifications to the algorithms. The end of this chapter evaluates the performance of the HDR system as a whole and gives a comparison between the runtimes of the CPU versus the GPU implementation. This Chapter contains algorithms and results taken from a diploma thesis on HDR video processing using GPUs which was supervised by the author [124]. It was also published in [50].

8.1 Considerations for a Parallel Implementation

When designing a parallel algorithm for a GPU implementation, the specific properties of the hardware must be understood. In our work, we use Nvidia's Compute Unified Device Architecture to create code for Nvidia GPUs. It classifies GPUs into categories with similar *compute capabilities*, allowing to abstract from hardware details. Between

[1] http://developer.nvidia.com/nvidia-gpu-computing-documentation/

the GPU classes, the differences are often just a matter of parameter adjustment, and all code is backward compatible so far. In this section, we introduce the architecture common to all CUDA devices, giving numbers that are specific to our graphics card where applicable.

The CUDA programming model is a C99 dialect with a minimum set of language extensions. At its core there are three key abstractions: a hierarchy of thread groups, shared memories and barrier synchronization. This model requires partitioning of the problem into many small sub problems that can be solved independently or by cooperation of the threads within a block. More precisely, for a problem to be shifted to the GPU, it must be expressible as a data-parallel algorithm. Data parallelism describes a programming paradigm that suggests the subdivision of a problem into smaller sub problems such that the same program (*kernel*) can be executed by a large number of threads working on many data elements in parallel.

The sum of all threads launched by a GPU within one kernel call is called a *grid*. A grid consists of several *blocks of threads* in a two dimensional structure. The elements contained in the blocks are light-weight *threads* executed by the GPU. In image processing, there is often a one-to-one correspondence between image blocks (e.g., 32×32 pixels) and blocks of threads. This means that the image is divided into pixel blocks which are then each processed by one thread block.

The conceptual structure of grids, blocks and threads is called *thread hierarchy*. Thread blocks are required to be executable independently, in any sequential order or in parallel. This requirement allows threads to be scheduled in arbitrary order to a flexible number of cores, as one block is always executed by one core. The graphics card we use contains 15 multiprocessors of which each can process 32 threads at once.

The data to be processed (e.g., the LDR exposures) must be copied from the host computer's memory to the memory of the graphics card. The GPU distinguishes between global, local, and shared memory, as illustrated in Figure 8.1. They differ in visibility, size and access time. *Global memory* is accessible by all threads running on all multiprocessors. It is typically several gigabytes in size, but accessing it can take up to 800 clock cycles. If data is read only and the access pattern exhibits locality, this is sped up by level-1 and level-2 caches. The access to *local memory* is restricted to a single thread and is very fast. It is comparable to the access to CPU registers. *Shared memory* is a user-managed cache that is shared among the threads of one block and invisible to all other blocks. Its size is 48 kilobytes on our GPU, and it can be read or written without latency, similar to a low-level cache or registers. Its main purposes are fast temporary data storage and communication between the threads of a block. For the sake of performance, it should be avoided to use the slow global memory wherever possible by keeping intermediate results in local or shared memory.

Special care must be taken to prevent race conditions when multiple threads write to the same address of the shared memory concurrently. The entire memory range is split up into 32 interleaved memory banks such that successive 32-bit words are assigned to successive banks. This means that the 32 threads of a block that are processed concurrently can all access the shared memory in parallel as long as the requested words lie in 32 different memory banks. When two concurrent threads access different words

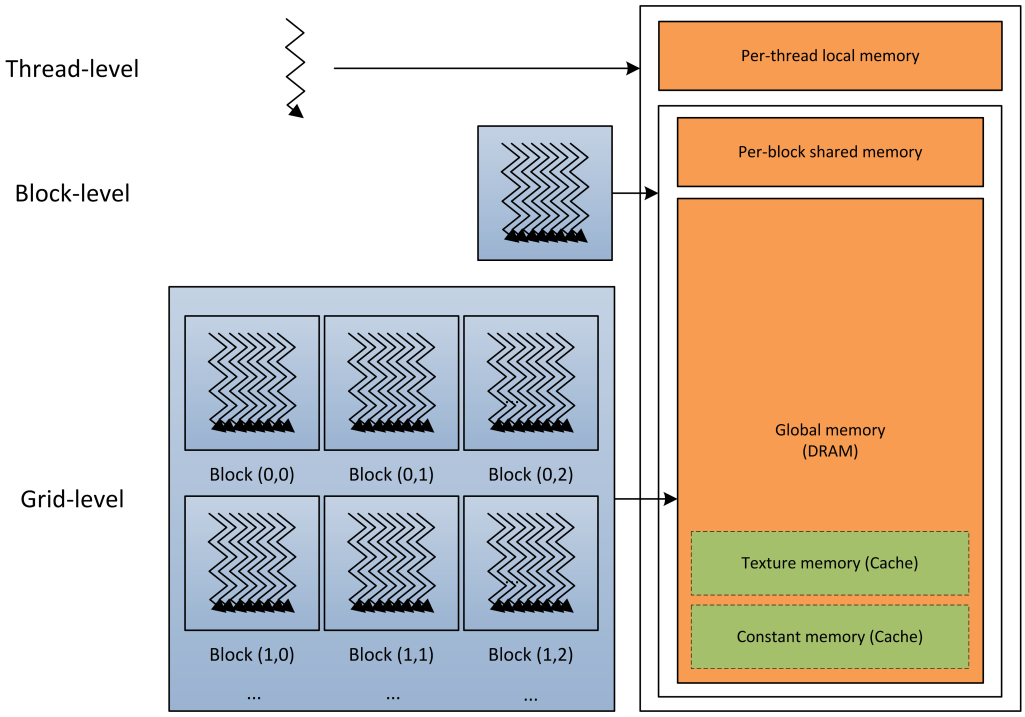


Figure 8.1: *Visualization of the memory hierarchy in CUDA. Single threads have access to every type of memory. Threads in a block can only communicate via shared and global memory. The whole grid uses global memory to exchange data.*

in the same memory bank at the same time, they can only be read sequentially, and the performance gain of parallel processing is lost. This needs to be considered when designing algorithms for CUDA.

There exist two additional read-only caches called *constant memory* and *texture memory*. Constant memory is used for broadcasting read-only values quickly to requesting threads. Texture memory is interesting in our scenario as it is an optimized cache for 2D access. When a thread accesses the texture cache, the hardware prefetches values from global memory that are close to the fetched value in 2D (e.g., neighboring pixels). This decreases cache misses in image processing applications, leading to a large performance gain. Additionally, the texture cache offers addressing modes like linear interpolation of values in hardware. Consequently, these operations are very fast: fetching a linear interpolated value does not take any longer than fetching a non-interpolated one.

Multiprocessors schedule and execute threads in groups of 32 parallel threads called *warps*. Warps each have their own instruction counter and registers. They always execute one common instruction at a time. If threads diverge due to data-dependent branching, the warp serializes which means that threads following the branch are executed together while all other threads are idle. When all threads are on the same path again, execution is merged for the whole warp. Such divergent branching thus slows down execution speed and is to be avoided in the code.

8.2 Parallelizability of the HDR Pipeline

Redesigning an algorithm for a parallel implementation takes considerable effort. It is also more difficult to assure correctness and to maintain such an implementation. We thus first analyze the individual steps of the HDR pipeline with respect to the computation time they require and their suitability for parallelization. The former is measured easily from the existing sequential code. The latter is judged by the amount of parallelism a problem exhibits and its arithmetic intensity: Parallelism is the percentage of instructions that can be executed concurrently; arithmetic intensity can be defined as the ratio between mathematical operations and memory access, where a higher arithmetic intensity is preferable for a GPU realization. Both criteria are somewhat vague but still sufficient for assessing the suitability for a parallel implementation.

The complete HDR video system we implemented consists of the following parts: Calculation of optimal shutters, capturing using the sequence mode, color conversion, histogram-based registration, HDR stitching, histogram adjustment tone mapping with flicker reduction, and color back conversion. These parts can be further divided into their computationally expensive subtasks. The most expensive step of determining shutter sequences is repeatedly calculating the *cross correlation* between the (existing) brightness histogram and the contribution vector. Capturing is done by the camera and can not be sped up. The Bayer pattern in the captured LDR images is first *interpolated to full RGB* and then *converted into Yxy*. For registration, a *brightness histogram* must be calculated for each LDR image and its *median* must be found. *Row and column histograms* are then created from a temporary MTB and the *normalized cross correlation* (NCC) between them is calculated repeatedly. The resulting shift vector is *Kalman filtered*. HDR stitching consists of computing each HDR pixel from a *weighted average* over the corresponding LDR pixels. Tone mapping requires the computation of a cumulative log radiance histogram, which consists of finding the *minimum and maximum* radiance, calculating a *log radiance histogram* and *cumulating* the bins. Each pixel is then *tone mapped* from radiance to pixel values. To reduce flicker, the *average brightness* of the tone mapped result must be calculated, and the image must be *normalized* iteratively. In the end, the tone mapped image is *converted back to RGB*.

In the following, we analyze all subtasks with respect to necessity and feasibility of a parallel implementation. The results of the analysis are summarized in Table 8.1. Refer to the respective previous chapters of this thesis for details. Note that subtasks that are similar to each other are discussed only once.

Cross correlation: The normalized cross correlation between all row or column histograms for all possible shift values can be calculated independently of each other. One thread can be started to calculate the NCC between each histogram pair for each possible shift. A high cache hit rate is expected, because the same row/column histogram bins are *read* repeatedly. Additionally, a high arithmetic intensity makes cross correlation well-suited for parallelization. On the other hand, it is a rather cheap operation overall. We decided to implement cross correlation on the GPU only for image registration and not for shutter calculation: The row and column histograms were created on the GPU and thus already reside in the graphic card's main memory. Furthermore,

HDR Pipeline Subtasks					
Pipeline Step	Operators	Cost	P	AI	GPU/CPU
Optimal Shutter Seq.	Cross Correlation	low	med.	high	CPU
Bayer Pattern Interp.	-	high	high	med.	GPU
Color Space Conversion	-	high	high	high	GPU
Image Registration	Brightness Hist.	high	med.	low	GPU
	Median	low	med.	low	CPU
	Row/Column Hist.	high	high	low	GPU
	NCC	low	high	high	GPU
	Kalman Filter	low	low	high	CPU
HDR Stitching	Weighted Average	high	high	high	GPU
Tone Mapping	Min / Max	high	med.	low	GPU
	Brightness Hist.	high	med.	low	GPU
	Hist. Cumulation	low	med.	low	CPU
	TM Operator	high	high	high	GPU
	Avg. Brightness	high	med.	low	GPU
	Normalization	high	high	med.	GPU
Color Back Conversion	-	high	high	high	GPU

Table 8.1: *Overview of subtasks of the HDR pipeline. Shown are the relative computational cost, the amount of parallelism (P), and the arithmetic intensity (AI). "high" entries indicate factors that suggest a GPU implementation. Our decision for the type of implementation is given in the rightmost column.*

they are constant during the entire cross correlation, enabling efficient caching and data independence. This is not the case when determining optimal shutters, as can be seen in Equations 5.3 and 5.4. The combined contribution, which is repeatedly correlated with the brightness histogram, changes after each determined shutter speed. This adds a sequential dependence to the calculation, making it less suitable for parallelization. Since it is a cheap operation, we kept the existing sequential code.

Bayer pattern interpolation: For bilinearly interpolating a pixel's RGB value from its neighbors, four cases need to be differentiated based on the pixel's location on the color filter array. This means that without further modification, this method leads to massive branching in the kernel. On the other hand, an interpolation kernel can benefit from texture caching, because the access pattern is highly local in 2D. It is also a highly parallel problem. Its arithmetic intensity varies with the pixel position with an average of 0.65 arithmetic operations per memory access.

Color conversion, tone mapping, normalization: Color space conversion, as well as applying the tone mapping operator and image normalization are ideally suited for a GPU implementation since they fully comply with the data parallelism paradigm. No branching takes place as each element is treated in the same way. Each image pixel is read and written exactly once. Additionally, all three operations have a high arithmetic intensity caused by the multiplication and addition of pixel values.

Brightness histogram: During the creation of a brightness histogram, data must be written to a small set of memory addresses (the histogram bins) for all of the pixels. This induces a certain data dependence and leads to thread collisions and sequential writing. Additionally, there is very little computational work between the memory accesses. Despite these difficulties, we considered it for a GPU implementation because it is a costly operation overall.

Median computation: There exist parallel sorting algorithms, so the problem of finding the median of a histogram can be parallelized. However, the problem size of searching through a number of histogram bins is too small to justify the effort.

Row and column histograms: The creation of an MTB can be viewed as an intermediate step to the computation of row and column histograms. Both operations have a low arithmetic intensity. MTB creation has high parallelism as each pixel can be converted separately. Computation of row and column histograms brings up a similar issue as brightness histogram computation: An entire row/column accesses the same histogram bin. However, in this case, this access is predictable and can be optimized.

Kalman filtering: Filtering only takes about 16μs on a CPU, so its computational effort is negligible, and we thus leave it on the CPU.

Weighted average: HDR Stitching complies well with the data parallelism paradigm. The radiance value of each HDR pixel can be obtained independently without a need for synchronization. A thread is started for each pixel, iterating through the corresponding LDR pixels to compute the weighted average. The arithmetic intensity of this operator is high due to the addition and multiplication of pixel values and the evaluation of the weighting function. The adaptive number of LDR exposures prevents the usage of the texture cache, because the number of textures needs to be known at compile time.

Minimum, maximum and average: Calculating a cumulative log radiance histogram for tone mapping starts by finding the minimum and maximum log radiance in the HDR image. This is similar to calculating the average image brightness for flicker reduction. It can be done by so-called parallel reduction. The minimum, maximum or average is first (trivially) calculated in parallel for each pixel. The values of each pair of two neighboring pixels is then merged in the second step, also in parallel. In each further step, the number of values is halved until a single value for the entire image is reached.

Cumulative log radiance histogram: The same considerations as for general brightness histograms described above apply to the computation of the cumulative log radiance histogram. The only exception is the summing up of the histogram bins. It has negligible computational costs and does not require a GPU implementation.

8.3 Parallel Implementation

This section describes a number of adjustments that were made to the subtasks of the HDR pipeline for their parallel GPU implementation. For similar subtasks, we limit our description to one sample of the respective category.

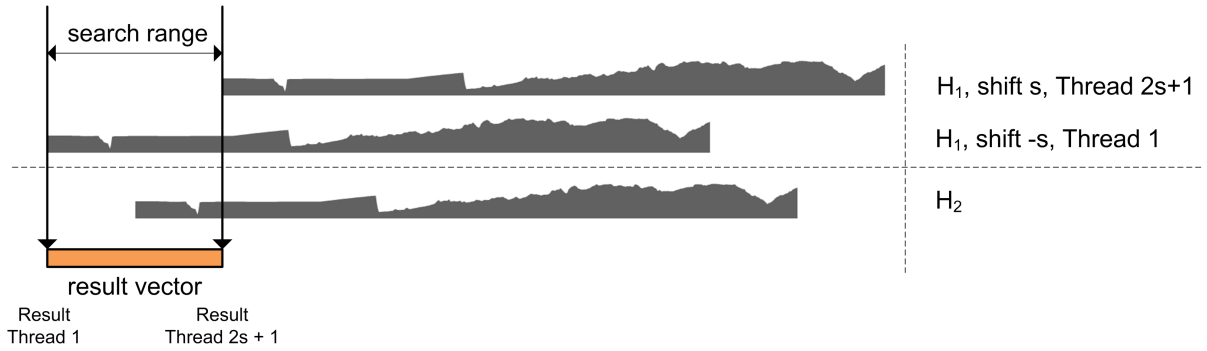


Figure 8.2: *Normalized cross correlation between two column histograms H_1 and H_2 to determine the horizontal shift s between two images. A thread is started for each value in the result vector to be calculated. The values represent the correlation for a specific shift.*

8.3.1 Normalized Cross Correlation

To compute the NCC between two row or column histograms, one thread is started for each shift s in the search range (see Equation 6.4). Each thread calculates the normalized cross correlation for its shift and writes the result of the calculation to its corresponding position in the result vector. This approach allows a high cache hit rate if the arrays to be correlated are read-only and bound to the texture cache. In the end, the result vector is downloaded into the host memory. A sequential search on the CPU finds the position of the highest correlation value. Figure 8.2 illustrates the process.

8.3.2 Bayer Pattern Interpolation

Bayer pattern interpolation benefits strongly from texture caching because of its high access locality. It can be implemented with a few lines of code on a CPU. A naive implementation iterates through the image and interpolates the missing pixel values differently depending on the four possible locations in the Bayer grid. The obvious problem with this implementation for a GPU is the massive branching induced by the four cases. In order to avoid branching, a *thread relocation* mechanism was implemented which is illustrated in Figure 8.3. It changes the relationship between a pixel and the thread which does the interpolation. Normally, a thread would be responsible for interpolating the RGB values for a pixel matching the thread's position in the 2D grid. Neighboring threads would then be executed at the same time sharing the same instruction counter. In this situation, the different branching of the threads would lead to a serial execution and low performance. Relocating the threads so that those corresponding to pixels with matching location on the color filter array are executed simultaneously avoids branching. For example, in the Figure, every thread in block 4 can now calculate its blue component from its left and right neighbors.

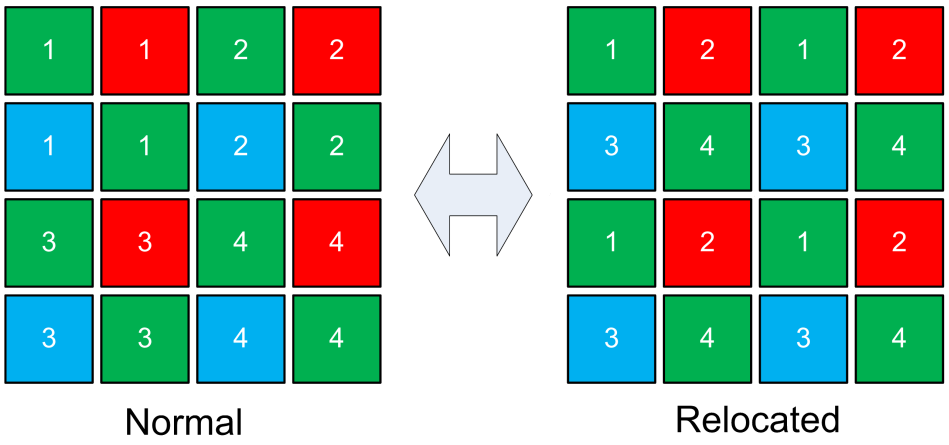


Figure 8.3: *Each pixel is assigned one thread to interpolate its RGB values from the surrounding. The threads with the same number belong to the same block and are executed at the same time. This leads to threads with different colors in their neighborhood running simultaneously, and branching becomes necessary. After relocating the threads, each thread in the block can interpolate in the same way.*

8.3.3 Color Conversion

The description of GPU color conversion given here also applies to using a global tone mapping operator and to image normalization. The implementation of the kernel for color conversion from RGB to XYZ is the translation of Equation 2.9 into code. Again, one thread is started per pixel. The RGB values for the conversion are read from the pixel corresponding to the thread. These values are multiplied by the color transformation matrix. The resulting vector $(X, Y, Z)^T$ is then normalized to obtain the chromaticity xy and the brightness Y. These values are written back to the three channels of the pixel in the output image. Afterwards, the result can either be passed on to the next kernel (e.g., image registration), or it can be displayed in the case of the final back conversion to RGB.

8.3.4 Brightness Histogram

The image over which the histogram is computed is first subdivided into rectangular areas of size 32×64 pixels. To calculate a histogram, each block has to perform 2048 read and write operations on the histogram bins in global memory. This would take many clock cycles, and the operations would be strictly serial. Instead, we apply the paradigm of parallel reduction. That is, we first create histograms for small image areas and then successively merge them into one final histogram over the entire image.
32 threads are started per block. Each thread computes a separate histogram with 64 bins over one row of the block which is written to the fast shared memory. We use a histogram with only 64 bins for efficiency purposes. Interleaving of the histograms in shared memory such that a whole histogram resides on the same memory bank allows for

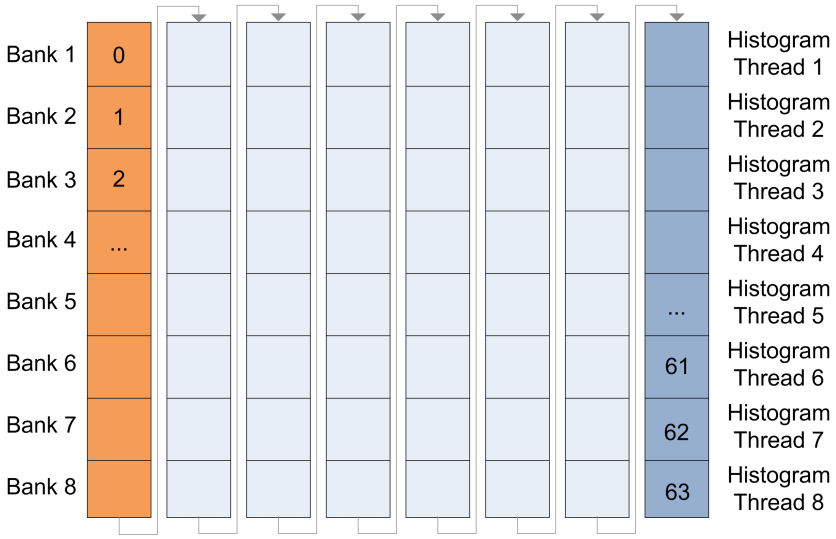


Figure 8.4: *Simplified illustration of shared memory with 8 memory banks and 64 addresses. Consecutive addresses lie on consecutive banks. The histograms are interleaved such that each lies on its own bank. Eight threads can write concurrently.*

conflict-free memory access by the threads, and true parallelism can be achieved. The banks of shared memory and the way the histograms are stored is illustrated in Figure 8.4.
Next, the 32 histograms of the block are summed up into a single histogram for the block. Since the content of the shared memory expires when the threads terminate, the same 32 threads must be re-used for summation. Each thread is assigned two bins of the total histogram. It loops through the 32 histograms (vertically in Figure 8.4) and maintains two sums. It must be noted that each histogram resides on its own memory bank. If all threads started with the first histogram, the 32 read operations to the same bank would be serialized, leading to bad performance. Instead, summation loops start with a different histogram for each thread (shifted by one relative to its predecessor thread). Like this, all summations can be done in parallel without bank conflicts. The final sums (i.e., the histogram for the block) are then written to global memory by an atomic add function provided by CUDA.
With this approach, we reduce the number of write operations to global memory from 2048 to 64 per block. Pixel data is read from global memory as a texture which allows for efficient caching.

8.3.5 Row and Column Histogram

For the registration of an image pair, a total of eight row or column histograms are created. All histograms are calculated separately by similarly implemented method calls. On a GPU, this is more efficient than running one parametrized method with code branching. For simplicity, we restrict our description to the creation of a column

histogram counting black pixels for every column of an image.
We subdivide the image into blocks of 32×32 pixels. This time, one thread is started for each pixel in the block. Each thread computes the MTB of its respective pixel, that is, the thread checks if its pixel is darker than the threshold and writes a 1 into shared memory if it is. The MTB is thus created in shared memory only. Care is taken that the set of 32 threads that are executed concurrently on a multiprocessor write the bit to 32 separate memory banks. Pixel data is again read from global memory as a texture.
32 of the threads are then re-used to count the black pixels of each column. Each thread is assigned one of the columns of the block. The thread loops through all rows to count the 1s in shared memory. An entire row resides in the same memory bank (see Figure 8.4). So, in order to prevent bank conflicts, the threads each start counting from a different row so that all 32 read operations can be done in parallel. The sum of black pixels in a column is then added to the column histogram in global memory using an atomic add operation.

8.3.6 HDR Stitching

HDR stitching cannot use the GPU's texture cache to access the LDR exposures to be stitched. This is because the CUDA compiler needs to know the number of texture cache bindings in advance before compilation; however, the number of LDR exposures is dynamically chosen as described earlier. The LDR sequence can be viewed as a 3D stack of images with varying size. One thread is started for each HDR pixel which iterates through the corresponding pixels in the stack of exposures and calculates the weighted average. During one iteration, the current radiance and chrominance values and the cumulated weight of all pixels must be held in local memory (registers). The radiance and the two chrominance values are then written to the output image in global memory as floating point values. Contrary to the CPU version, the GPU implementation does not precalculate the weighting function or stores it in memory. Our experiments showed that calculating only the required weights on the fly is cheaper than accessing the precalculated array from all threads.

8.3.7 Minimum, Maximum, Average

Calculating the minimum, maximum or average value of an image on a GPU is done by parallel reduction. We use the calculation of the maximum of a one-dimensional array here for illustration. Parallel reduction is an iterative divide-and-conquer approach. In the first iteration, one thread is started for each element in the array. Each thread calculates the (trivial) maximum of the element, which is simply the element itself, and writes it to shared memory. Every other thread is then discarded. In the next iteration, the remaining threads calculate the maximum of their element and its neighbor and again write it to shared memory. In each iteration, the number of threads is halved, and maxima are combined with their neighbors. This is repeated until only one global maximum is left which can then either be written to global memory or copied back to the host system.

8.4 Experimental Results

In this section, we present the results of the experiments we conducted to assess the performance of our HDR video system with GPU support. The following aspects are analyzed:

- influence of the shutter speeds and the number of exposures on the time taken to capture the LDR image sequence,
- processing times of the subtasks of the HDR pipeline when changing the image size,
- processing times when changing the number of exposures,
- average capturing and processing times in a 30 seconds HDR video under realistic conditions,
- comparison of CPU and GPU processing times.

The individual subtasks were grouped together as seen fit for the analysis. Displaying the processed video frames means copying them into the memory of the graphics card. Since the last step in the pipeline – color conversion from Yxy back to RGB – is performed on the GPU anyway, displaying the result is a free operation and thus ignored in the following.
For all experiments, we used a desktop PC with an AMD Athlon II X2 250 64-bit CPU with two cores running at 3 GHz and a total of 4 GB of RAM. The installed graphics card is an Nvidia GeForce GTX 480 with 15 multicores running at a clock rate of 1.4 GHz and 1.5 GB of dedicated memory. Each multicore can process 32 threads at once. Our camera is an AVT Pike F-032C FireWire camera capable of capturing 208 frames per second in VGA resolution. It can capture in sequence mode, and it uses a Bayer color filter array to acquire color images.

8.4.1 Analysis of the Capturing Time

In order to determine how the chosen shutter speeds and the number of required LDR images affect capturing, we varied these parameters and measured the time between triggering the images and fully receiving them from the camera. Trying all combinations of parameters exhaustively is not feasible as there is a large number of possible shutter sequences. We thus limited ourselves to sequences where every image in the sequence is taken using the same shutter speed. 24 different shutter settings in the range of 4 ms to 80 ms were chosen. The number of exposures was varied between one and ten. For each of the 240 combinations, we captured five sequences and averaged the time taken. In the first iteration of the experiment, the parameter sequence was transmitted to the camera, and bandwidth on the FireWire bus and image buffers were allocated for every measurement. This simulates the situation where the parameters change, and they are newly transmitted to the camera. We found that within the range of considered shutter speeds Δt and number of exposures n, the capture time t_{capt} in milliseconds can be closely modeled as:

$$t_{capt}(\Delta t, n) = \Delta t \cdot n + 2 \cdot n + 25.8. \tag{8.1}$$

Here, $\Delta t \cdot n$ is simply the sum of shutter times of all images. It is the minimum amount of time capturing can take since each of the n images must be exposed for the desired time span Δt. $2 \cdot n + 25.8$ ms is the time it takes to transmit the parameters to the camera, allocate resources, and trigger the images. This is a very slow process, because the FireWire camera interface IIDC only allows to set parameters one by one with an acknowledgment after each value is set.
In the second iteration of the experiment, the sequence was set up only once beforehand, and then the capturing of the same sequence was triggered five times. This corresponds to the situation where parameters only changed very little from one frame to the next, and the currently used parameter sequence is kept. The rest of the experimental setup is the same as before. Now, the capturing time in milliseconds is

$$t_{capt}(\Delta t, n) = \Delta t \cdot n + 6.9. \tag{8.2}$$

The sum of shutter times is still the lower bound. Now, only the constant 6.9 ms for triggering the sequence are added to it. It can be seen that re-using the same sequence is much cheaper than setting up a new one, which is a major reason for introducing the stability criterion in Section 5.2.3. Both equations are valid for shutter speeds in the tested range. However, for shutter speeds much lower than 4 ms, $\Delta t \cdot n$ does not shrink proportionally anymore. Capturing time is then bounded by the time it takes to transfer the image data from the camera to the PC.

8.4.2 Analysis of the Processing Time

The computation time of most of the steps in the HDR pipeline depends on the size of the images. In the experiment described here, we analyze the relationship between processing time and image size. Most parts of our GPU implementation are optimized specifically for the image width of 640 pixels. Changing this in our implementation would bias the results of this test. However, different image heights were considered in the implementation to accommodate the outputs of the capturing method with partial re-exposures. This fact is used to investigate the relationship between image size and processing time. The images in this experiment all had the full width of 640 pixels and a height varying from 100 to 480 pixels. For each size, a sequence of five exposures was captured once and processed 20 times by the entire HDR pipeline to obtain stable average processing times. The steps from Bayer pattern interpolation to the computation of row and column histograms were thus performed five times in each iteration, cross correlation and filtering was done four times, and HDR stitching needs to take five exposures into account. All the subsequent steps work on just one HDR image. The content of the images has no significant influence on the processing times. We thus set the shutter speeds to an arbitrary value that exposes the recorded indoor scene well. Most steps of the pipeline depend on the image size in an obvious way as it influences the number of pixels or blocks to process. Only the processing steps of shutter speed computation and Kalman filtering are completely unaffected. Figure 8.5 shows the measured processing times versus image height. Debayering and the initial color conversion are grouped

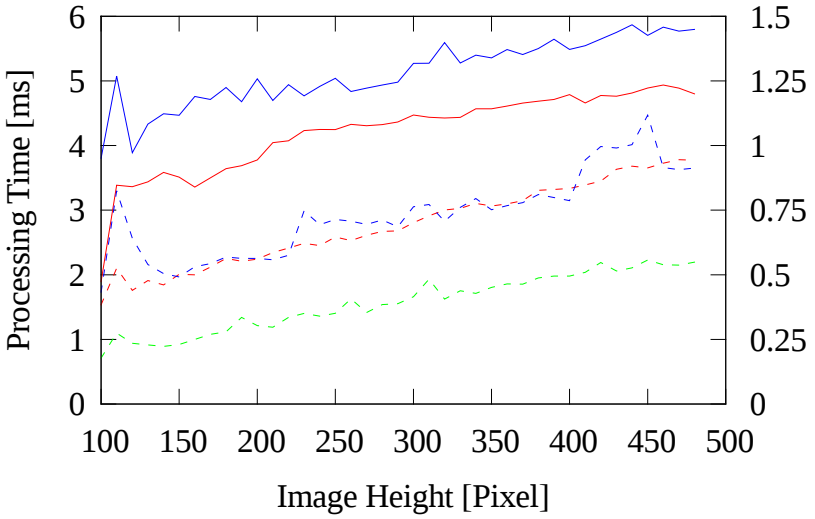


Figure 8.5: *Processing time versus image height for a fixed number of 5 exposures. The two solid lines are image registration (blue) and color conversion (red). They both use the left scale ranging from 0 to 6 ms. The dashed lines use the right scale. They are tone mapping (blue), HDR stitching (red) and color back conversion (green).*

together. Excluding apparent measurement noise, the computation times of the pipeline steps grow linearly in the number of pixels, as expected.
Now, we keep the image size at its maximum of 640×480 pixels and vary the number of LDR exposures instead. The only steps that need to process a varying number of exposures are color conversion from RGB to Yxy, Bayer pattern interpolation, image registration and HDR stitching. The initial color conversion and debayering are again grouped together. In order to perform registration, at least two images must be present in the sequence. We thus varied the number of exposures from two to ten and, again, repeated the measurement 20 times on the same sequence for a stable average. The shutters were chosen to match the given scene. Figure 8.6 shows the measured processing times versus the number of exposures. Again, the dependency is linear, as expected.

8.4.3 Performance in a Realistic Scenario

For the final test, we captured 30 seconds of HDR video material in a realistic scenario using our system. The camera was situated inside a room illuminated only by sunlight shining through a window on a sunny day. During the 30 seconds, the camera pans from the bright window towards the darker room and eventually towards a door leading to an even darker hallway. The video thus includes very bright, very dark, and mixed lighting conditions. Four example frames are shown in Figure 8.7. We loosely refer to the sections of the video as *window*, *indoor*, and *door*. The window is visible from frames 0 to 311, and the door enters the field of view in frame 583. The *window* section is a typical high dynamic range example with large brightness differences between the sunny

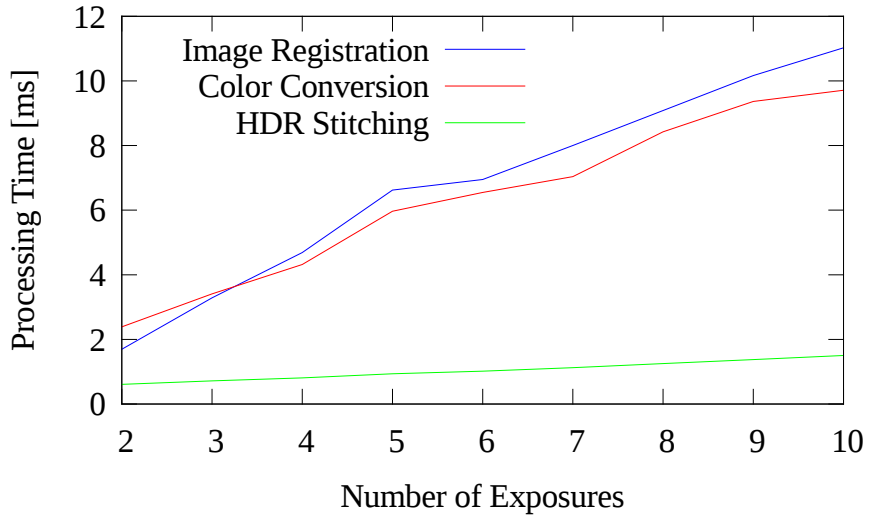


Figure 8.6: *Processing time versus number of exposures for full images. Only those steps that process multiple LDR exposures are considered.*

outside and the dark inside. While slowly panning towards the inside of the room, the dynamic range decreases. The *indoor* section has the lowest dynamic range of the video. It increases again as the door to the dark hallway becomes visible. Here, long shutter speeds are used to capture the very dark parts of the scene properly.
The video consists of 733 HDR frames. Averaged over the entire video, an HDR frame was created from 3.62 LDR exposures. On the average, 29.8 ms per frame were spent for capturing and 13.6 ms for processing. This results in a total time per frame of 43.4 ms and an average frame rate of 23 fps.
Figure 8.8 illustrates, how the number of exposures, the sum of shutter values and the capturing time change over the course of the video. As can be seen, there is a strong relationship between the sum of shutter values and the time taken for capturing. This means that the total exposure time has the greatest influence on the capture time. Total exposure time in turn strongly depends on the number of exposures, but also on the illumination level of the scene. The Figure also clearly shows the peaks in the capturing time whenever the number of exposures or the shutter speeds change. This is due to the retransmission of the capturing parameters, i.e., changing from Equation 8.2 to Equation 8.1. During the *door* section of the video, the sum of shutter times required to properly expose the dark hallway is already higher than the available frame time of 40 ms, making the frame rate drop.
Figure 8.9 shows the total processing time and the achieved frame rate for the video. When comparing it to Figure 8.8, it can be seen that the frame rate drops when the parameter sequence changes. The rate is highest in the *indoor* section with a low DR (45 to 50 fps) and goes down to 13 fps when the door to the hallway becomes visible. It is interesting to note that the impact of the processing time on the frame rate is rather low.
The time to create an HDR frame from beginning to end is more closely inspected in

Figure 8.7: *Four tone mapped frames of the HDR video taken in our experiment. The camera pans from left to right. The top left frame belongs to the section* window *(frame number 100). Top right shows the transition to section* indoor *(300). The bottom row frames represent* indoor *(500) and* door *(620) respectively.*

Figure 8.10. The top left chart shows the fractions of the computation time of the steps of the HDR pipeline. As mentioned above, capturing took 29.8 ms and processing 13.6 ms in the average. Color conversion from RGB to Yxy, back to RGB and debayering is grouped together. The other sub-figures show how computation time is further divided among the subtasks for color conversion, image registration and tone mapping.

For comparison, the entire video was processed again using our pure CPU implementation. The computation times were now 56 ms for color conversion, 24 ms for image registration, 43 ms for HDR stitching and 80 ms for tone mapping. This leads to an average processing time of 203 ms per frame. Our GPU implementation is thus faster by a factor of 15 on the average.

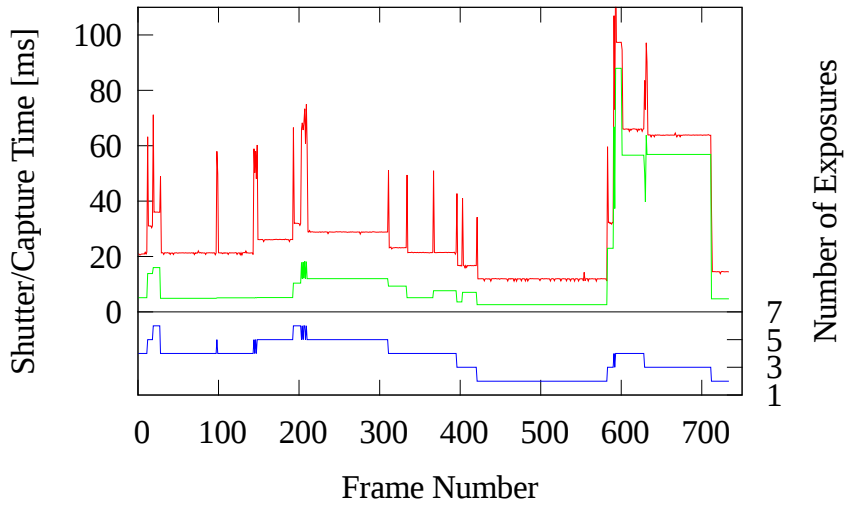


Figure 8.8: *The red line shows the capturing time for each frame in milliseconds. It strongly depends on the sum of shutter times of the LDR exposures shown in green. The blue line is the number of exposures. Peaks in the capture time occur when exposure parameters change.*

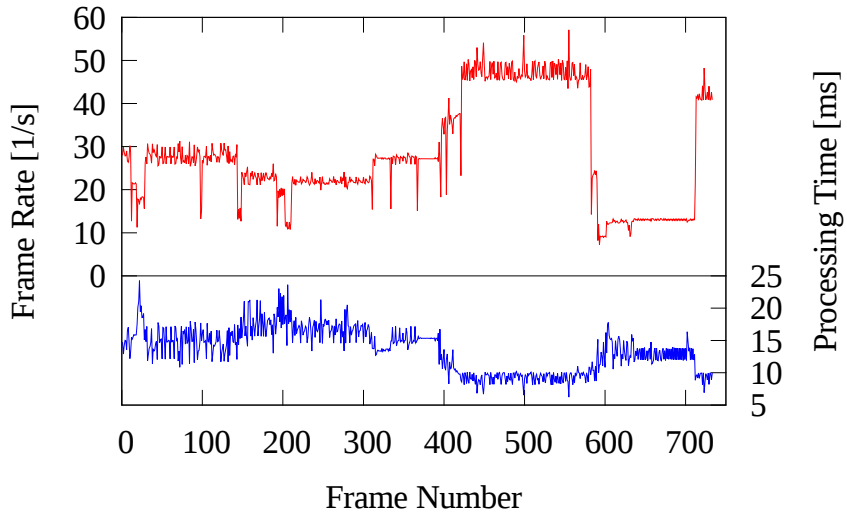


Figure 8.9: *Frame rate (red) and total processing time (blue) for the frames of the test video. The processing time has less variation than the capturing time.*

8.5 Conclusions

From the experiment described above, we conclude that capturing is the bottleneck of the HDR video pipeline. Setting up the camera parameters over the FireWire bus is a costly operation. Especially when the scene is very dark, the exposure times alone can already exceed the available frame time. This is a well-known problem even for LDR

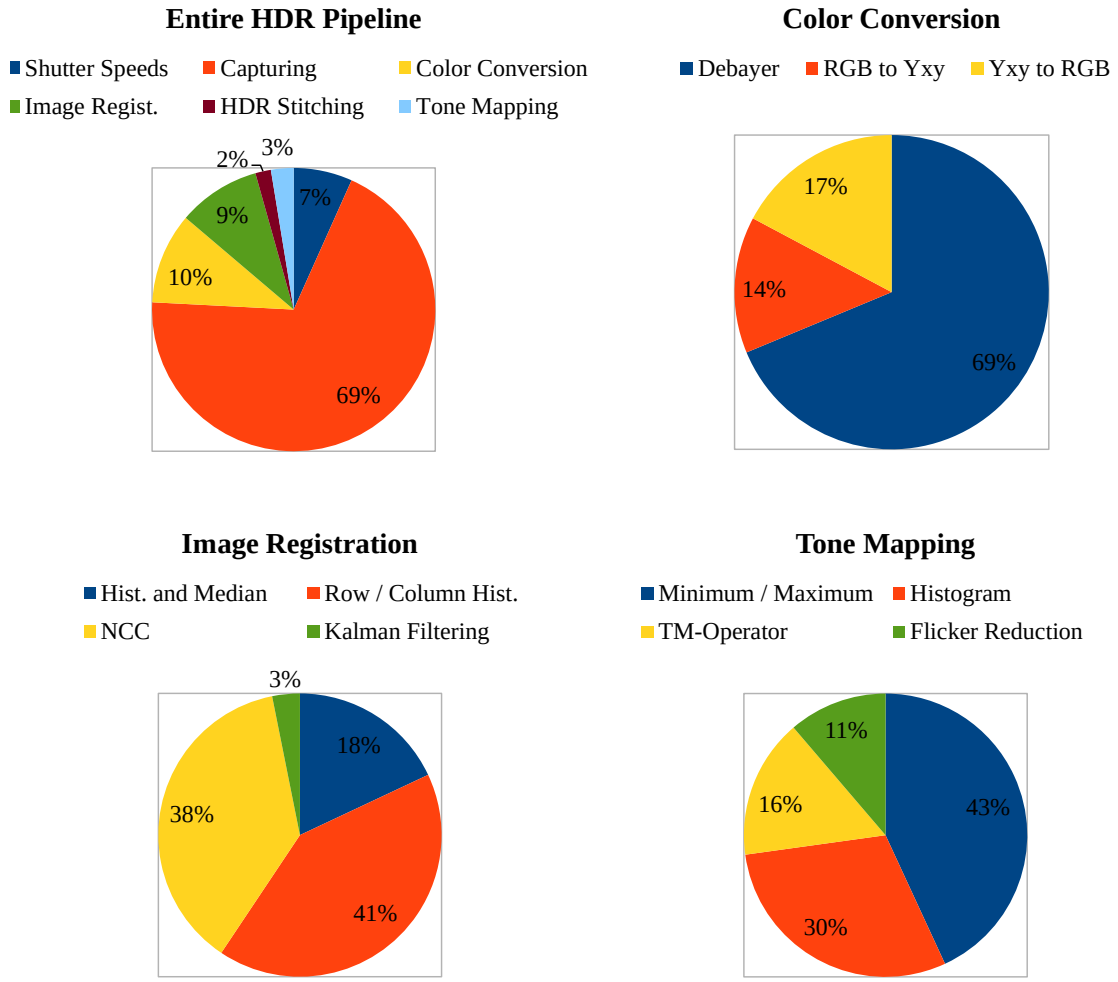


Figure 8.10: *Percentage of the time taken to perform the steps of the HDR pipeline in our test video. The steps of color conversion, image registration and tone mapping are further subdivided into their individual tasks.*

video equipment. The low-cost solution is to increase the sensor's gain to amplify the image brightness and noise at the same time. A better solution with respect to image quality is to employ a bigger lens which allows to collect more light during the same time span. Despite this limitation, we can also conclude that our system is capable of capturing high dynamic range video in real-time with a sufficiently high frame rate when the lens is adequate for the illumination level of the scene. In particular, the frame rate averages to 24.7 fps in the high dynamic range *window* section of the video. The processing times are consistently lower than 25 ms. In our current implementation, capturing and processing is done sequentially. Large portions of these two parts could be done in parallel, which would further increase the frame rate. Since capturing always takes longer than processing, processing could then be done almost for free.

Chapter 9

Video Automatic Optical Inspection

Flaws in the process of populating *printed circuit boards* with electronic components often lead to malfunctioning of the resulting *printed circuit assembly* (PCA). To guarantee that the right components were placed at the correct positions and work as expected, PCAs must be tested subsequent to the assembly. Various methods for testing exist. The most straightforward approach is to do functional testing. This means running a sequence of tests on the assembly as a whole and monitoring the results. However, there may be faults that are more subtle and not covered by the functional tests. It is therefore common to perform in-circuit tests to gain a more fine-grained understanding of the components' functionality. They allow for checks at any level of granularity varying from a single component to the entire assembly. In-circuit testing is done by connecting electrical probes to a PCA that allow for measuring conductivity, resistance, capacity and other electrical properties.

The two major techniques used are the so-called *bed of nails testers* and the *flying probe testers*. While the former uses a static arrangement of connectors that is pressed against the PCA, the latter uses a typically much smaller set of movable probes to connect. In practice, creating a specialized bed of nails adapter for one particular type of board is expensive. Only for high volumes, the initial cost is redeemed by the high testing speed achieved through extensive parallelization. On the other hand, the flexibility of flying probe testers takes effect for smaller volumes and prototyping. These testers can be easily configured to test new types of boards as needed. Their main drawback is the lower testing speed due to the smaller number of probes and the time it takes to re-position them between the individual checks.

An alternative to functional and in-circuit testing is *automatic optical inspection* (AOI) [83, 64, 107, 44]. In this approach, line scan cameras or area scan cameras with strong magnification are used to capture high-resolution images of the PCA. The digital images are then processed by machine vision algorithms to search for faults. This technique reveals faults that are hard or impossible to detect by the other two approaches. Examples

of such faults are bent pins on a connector, badly aligned components, bad solder joints or faults in regions that are unreachable by probes. Another advantage of optical inspection is contact-less testing. Assemblies tested with in-circuit tests often show traces of the pointy tips of the connectors on the soldering pads. In-circuit testing damages the tested PCA to some extent which can be avoided by AOI.

The approach we chose in this chapter is the enhancement of a flying probe tester by AOI. This combination achieves a high coverage of fault classes as it joins the sets of detectable faults of flying probe testers and AOI. Additionally, increased confidence in the test results can be gained by checking crucial elements redundantly. When testing speed is of major concern, individual checks can be completely shifted to the AOI and performed in parallel to the electrical tests, speeding up the entire process significantly. Depending on the particular application, an optimal weighting between electronic testing and optical inspection can be determined.

As a novel idea, we use video sequences instead of still images in this scenario. We call our technique *Video-AOI*. Our definition of *video* in this context is that we continuously capture images of a moving PCA, resulting in a large number of images, such that each electrical component on the PCA is contained in multiple images, taken under varying viewing conditions. Having multiple images of each component can then be exploited for inspection. If the images are captured at known camera positions, the height of a component can be determined using *stereo vision* and *structure from motion* approaches, facilitating the detection of missing components. Another way of taking advantage of Video-AOI is by switching light sources on and off between the individual shots, controlling the casting of shadows and light reflections. Similarly, the shutter speed of the camera can be varied to capture images at varying exposure that each emphasize a different dynamic range of the object under consideration. The necessity for applying HDR video techniques in the context of optical inspection become apparent when looking at Figure 9.1. It shows the top of a capacitor with a highly reflective surface. Choosing the shutter speed high enough for the components on the board to be detectable results in saturation of the pixels showing the capacitor. Due to specular reflection of the light source, the markings on the capacitor are no longer recognizable. Creating an HDR frame from a dark and a bright image alleviates this problem.

Applying Video-AOI inside a flying probe tester gives rise to new difficulties to overcome. Optical inspection of PCAs requires an image capturing system with a very high resolution. The image may need to span up to $40cm$ of a board while still resolving details with a size of $100\mu m$. Such a resolution can only be achieved by line scan cameras, but only area scan cameras are capable of capturing video. It is therefore necessary to employ several area scan cameras at once, leading to increased costs of the system, high data rates to be processed and shortness of space inside the narrow-built tester. The narrowness of the tester is also a challenge for the lighting used. Additionally, due to the moving probes, it is not possible to capture a video of the PCA during the electronic tests. In order to not add overhead for capturing to the overall duration of the test, the video must be captured inline while the board is transported into the tester via a conveyor belt. In an industrial environment, the speed of the conveyor must be considered a given constant. Only by using short shutter times and cameras capable of

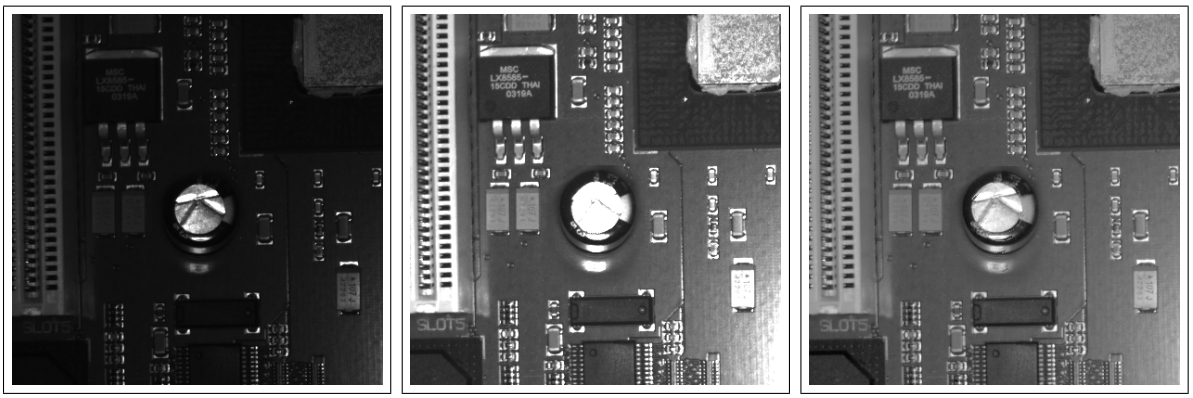


Figure 9.1: *The surface of the capacitor in the image reflects light specularly. When setting the shutter to a high enough value for the PCA to be well-exposed, the markings on the capacitor already saturate. The rightmost image is a tone mapped HDR image showing the components on the PCA and the markings on the capacitor at the same time.*

capturing video at high frame rates the capturing system can keep up with the conveyor speed. And lastly, the capturing system will produce a large number of images taken under varying lighting conditions and from differing angles. Before they can be used for inspection, they need to be aligned with respect to each other. Having a fully registered set of captured images of a PCA, the Video-AOI system is capable of determining the set of images containing a component to inspect and its exact pixel position in the images. It should be noted that calibrating the cameras, capturing image sequences and computing the offsets between the images so they can be used for inspection is the main focus of this chapter. Going into the details of a particular inspection algorithm is not our goal. In Section 9.1 we describe the prototype of a Video-AOI system and analyze its parameters and their relationship to each other. At the end of the section, we give the parameters of our prototype as an example. Section 9.2 focuses on the capturing and preprocessing of image sequences with our system and explains the steps necessary to capture videos that can be used for AOI. They include camera calibration, coordinate system transformations and four forms of image registration specific to this context. The Section ends with a demonstration of how HDR video can be created with our system. Section 9.3 contains experimental results. Section 9.4 concludes this Chapter. The work described here was published in [47].

9.1 Parameters of the System

Before building a Video-AOI system to be used inside a flying probe tester, one must first examine the system's parameters, understand their interrelation and ultimately decide upon the values to be used. We begin this section with an overview of the system as a whole and its relevant components. We then analyze the parameters by grouping them into three categories: Constraints by the tester (Section 9.1.2), parameters determined by the application (9.1.3) and the freely adjustable parameters (9.1.4). The

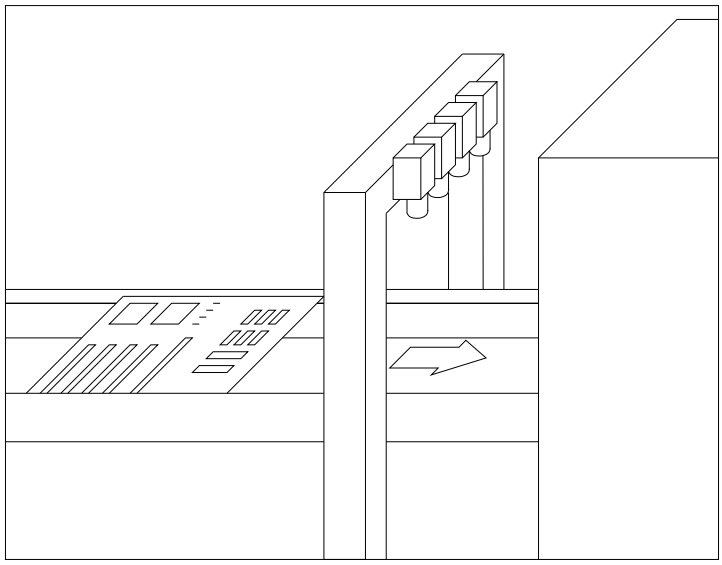


Figure 9.2: *Simplified representation of the flying probe tester and its preceding Video-AOI unit. The camera array captures images of the PCA as it is transported into the tester.*

hardware constraints and the application requirements are the starting point for the choice of adjustable parameters. The currently available camera hardware, optics and bus technology then determine how well the requirements can be met. At the end of this section, we list the choice of parameters we made when building our prototype.

9.1.1 System Overview

Figure 9.2 depicts the arrangement of conveyor, camera array and flying probe tester in our Video-AOI prototype. The system is built as an inline facility that can be directly connected to the production line. The assembled board is transported into the flying probe tester on a conveyor band. On its way in, it passes an array of area scan cameras. A photo sensor below the conveyor detects the approaching board and starts a clock generator to trigger the cameras. The cameras then synchronously capture a sequence of images until the board has completely passed the array. Finally the captured images are processed while the PCA is tested electronically inside the flying probe tester.
For the remainder of this chapter, we will refer to the board axis perpendicular to the board motion and parallel to the camera array as the horizontal axis. The direction of board movement defines the vertical axis respectively.

9.1.2 Hardware Parameters

The first category of parameters of the Video-AOI system for flying probe testers consists of the hardware parameters. Their values are usually given by the production environment and we assume them to be fixed. The following parameters are relevant:

Maximum PCA width. The size of the biggest PCA to be inspected is determined by the width of the conveyor. Its value has an impact on the number of cameras used, their resolution and the required optics. For our further considerations, we will assume that a PCA to be inspected has the maximum width which we denote by w.

Minimum/maximum vertical camera position. The camera array may not always be as freely positionable as shown in Figure 9.2. It may be tightly integrated into the production line or the tester itself. In these cases, constraints regarding the distance to the conveyor at which the camera array can be installed apply. They limit the achievable depth of field and determine the optics to be used for the cameras. The lower and upper bound for the distance of the camera to the PCA will be denoted by d_{min} and d_{max} respectively.

Conveyor band speed. Making changes to the speed at which a PCA is transported into the tester is difficult in an existing production line. We therefore assume it to be constant and assign it the letter v. As a result, the cameras' frame rate, shutter speed and lighting must be chosen carefully to allow the capturing of motion blur-free images under the given board movement speed.

9.1.3 Application Parameters

The second parameter set consists of those that are determined by the particular inspection application. Variations may be due to the type of boards and components to be inspected as well as the inspection task to be performed. The reconstruction of 3D data through stereo vision or the creation of HDR video for example require the cameras' fields of view to overlap largely. Application parameters are in general more flexible than hardware parameters. The parameters to be considered are:

Color. Many industrial cameras are available in the two variants color and monochrome. A common trick is to apply a color filter array to an image sensor to make the sensor cells color sensitive. While color cameras using this technique are similar in price to the corresponding b/w models, they effectively trade off resolution for the ability to detect colors. Color cameras should thus only be used if color information is relevant to the application at hand.

Depth of field. The height of the highest inspectable component on a PCA determines the depth of field required for the Video-AOI system. The achievable depth of field mainly depends on the camera's focal length, lens aperture and the distance to the PCA. It is measured in units of length and we denote it by d_f.

Image brightness. Assuming that the lighting illuminating a scene was chosen to be as bright as possible, the brightness of the captured images only depends on the shutter speed and lens aperture used. Increasing the exposure time by adjusting the shutter speed leads to brighter images but also to motion blur when capturing a fast moving board. Its upper bound is therefore determined by the conveyor

speed. If the desired brightness cannot be achieved by adjusting the shutter speed alone, depth of field must be traded off for image brightness by widening the lens aperture.

Spatial resolution. The size of the smallest structure to be inspected through Video-AOI is a parameter determined by the application. If we assume that a fixed number of pixels is required to resolve a structure, dividing this number of pixels by the size of the smallest structure directly leads to the required spatial resolution of the optical system. Hence the spatial resolution r is the number of pixels required per unit of dimension of the PCA.

Image multiplicity. We refer to the number of captured images in which a PCA component is contained as image multiplicity. Multiplicity can be achieved by capturing images that overlap horizontally or vertically. Horizontal overlap is the result of overlapping fields of view of the cameras in the array. In the vertical direction, increasing the cameras' frame rate increases multiplicity. The horizontal and vertical multiplicity factor m_h and m_v must be chosen according to how the captured images are to be processed. Capturing images of a component under varying lighting conditions for example can only be achieved through vertical overlap, since all cameras are triggered synchronously.

9.1.4 Adjustable Parameters

The final category of parameters are those of the optical system that must be chosen to meet the hardware and application requirements determined before. In practice, not every combination of parameters is possible and restrictions of the available camera hardware must be considered. It is then necessary to review the hardware and application parameters and to relax the constraints until they can be met. In this section we give guidelines and formulae for their choice.

Number of cameras and resolution. When deciding on the camera type and the number of cameras to be used for the Video-AOI system, the important values to consider are the maximum PCA width w, the spatial resolution r and the horizontal image multiplicity m_h. The horizontal camera resolution, i.e. the number of cells per row on the camera's sensor, and the number of cameras used must be big enough to achieve the desired spatial resolution over the entire width of a PCA at the desired multiplicity. Mathematically, this can be expressed as follows: Let n be the number of cameras and p_h the number of camera pixels per row. n and p_h must be chosen so that $wrm_h \leq np_h$. A suitable compromise between number of cameras and resolution is one that minimizes the overall cost. Typical values for p_h range from 500 to 2000 for current industrial cameras.

Focal length. Once the type and number of cameras are determined, the cameras need to be configured with suitable optics. We found that for optical inspection, it is desirable to employ telephoto lenses to keep the distortion due to short focal

lengths to a minimum. We therefore position the camera array at d_{max} within reasonable bounds. With the width of the desired field of view of a camera being p_h/r and knowing the width of the camera's sensor w_s, the required focal length f can be approximated by

$$f = \frac{d_{max}}{1 + p_h/(rw_s)}. \tag{9.1}$$

Lens aperture. If the chosen lens has an adjustable f-number N, it can be set to achieve the desired depth of field. Giving objective directives for setting the camera's f-number is difficult, since the definition of depth of field depends on the maximum size of the *acceptable circle of confusion* c, which is strongly subjective. A suitable value for c must be chosen for the given application (see Section 9.1.6 for our choice). If the camera's focus is set to the surface of the PCA, the achieved depth of field d_f can be roughly estimated by

$$d_f = d_{max} - \frac{d_{max} f^2}{f^2 + Nc(d_{max} - f)}. \tag{9.2}$$

Given the desired depth of field, and values for the other parameters from the previous considerations, this equation can be used to estimate the required f-number setting:

$$N = \frac{d_f f^2}{c(d_{max} - f)(d_{max} - d_f)}. \tag{9.3}$$

Shutter speed. The shutter speed Δt – also called *exposure time* – is the duration for which the camera's sensor is exposed to the light of the scene. It is measured in microseconds. As stated before, we assume a constant conveyor band speed v in our context. The longest usable shutter speed is therefore limited to avoid motion blur. In other words, the distance in pixels a point on the PCA moves during one exposure period must be lower than a threshold τ

$$vr\Delta t \leq \tau \tag{9.4}$$

leading to an upper bound for the shutter speed of

$$\Delta t \leq \frac{\tau}{vr}. \tag{9.5}$$

Frame rate. The last parameter to be chosen is the frame rate s of the cameras in the array. It is a crucial limiting factor of the attainable capture speed. Most industrial cameras have an adjustable frame rate, so the question is: What is the lowest frame rate sufficient to capture images of the moving PCA with the desired vertical multiplicity m_v? The cameras must then be chosen to support at least this rate. More precisely, this requirement can be formulated as

$$s \geq \frac{vr}{p_v} m_v \tag{9.6}$$

where vr is the conveyor band speed in pixels per time unit and p_v the frame height in pixels. It should be noted, that in theory the time between two frames cannot be shorter than one exposure period, so the frame rate must also be less than $1/\Delta t$. In a VAOI scenario though, the illumination is bright enough to use very short exposure periods, so that this restriction does not apply.

9.1.5 Further Considerations

Capturing videos with the VAOI system described above produces high data rates and large amounts of data. For example, capturing 1392×1040 pixels at 15 frames per second results in a data rate of roughly 175 $MBit/s$. By employing several of these cameras, the bandwidth quickly exceeds the limit of a single FireWire bus. Special care must be taken when choosing the image processing hardware to cope with the occurring data.
As a result of basing the design of the VAOI system and its parameters on the constant conveyor band speed, the total time to capture a video of a PCA only depends on the speed of the conveyor. Capturing ends once the PCA has passed the camera array completely. Though before the captured video can be used for inspection, the individual frames need to be registered. Time taken to perform this step is evaluated in Section 9.3.
Achieving proper lighting for the VAOI system is a challenge. Little advice on its choice can be given here as it strongly depends on the availability of space, mounting and power inside the tester. Generally speaking, the lighting should be as bright as possible to attain more freedom in choosing other parameters like shutter speed and f-number. We chose to use an array of LED light sources that are triggered synchronously to the cameras. By using the LEDs in a pulsed mode rather than operating them continuously, a higher luminous power can be achieved by the same LEDs without damaging them. The pulse only needs to be as long as the exposure time, giving the LEDs time to cool down while the CCD sensors are read out.

9.1.6 Our Choice of Parameters

Our VAOI prototype is built as box separate from the flying probe tester and is preceding the tester in the conveyor line. The box is opaque to allow for constant lighting conditions, independent of the surrounding light. Upon entering the VAOI prototype, the PCA triggers a light barrier that will start the capturing process. The same light barrier will then signalize the end of the capturing process as the PCA exits.
In our scenario, the hardware requirements are as follows: The widest PCA to be inspected by Video-AOI has a width of $w = 400\ mm$. PCAs are transported on the conveyor at a speed of approximately $v = 100\ mm/s$. Our Video-AOI prototype allows a maximum height of the camera array above the surface of the PCA of $d_{max} = 500\ mm$. The upper limit for the height of an inspectable component on a PCA – and thus the required depth of field – was set to $d_f = 10\ mm$. Our application requires a resolution of $r = 40$ pixels per millimeter. The horizontal image multiplicity was set to $m_h = 1.05$ for roughly 5% of overlap as tolerance. In the vertical direction, we capture with a

multiplicity of $m_v = 2.1$ to get two shots of each component with some tolerance that can be used for image registration. The cameras we use are monochrome 1394b FireWire cameras with a resolution of 1392×1040 ($p_h \times p_v$) pixels and 1/2" sensors.

Using the formulae described in Section 9.1.4, we get the following values for the adjustable parameters: We need at least $n = 12$ cameras. With a sensor width of $w_s = 6.4\ mm$, Equation 9.1 gives a focal length of $f = 77.7\ mm$. For reasons of availability, we used a lens with a fixed focal length of 75 mm and moved the camera array to a distance of $d = 483\ mm$ from the PCA to achieve the desired resolution. In Equation 9.5, we require that the PCA moves at most one pixel during one exposure period which results in an upper limit for the allowed shutter speed of $\Delta t \leq 250\ \mu s$. The shutter speed can be varied under this constraint to capture High Dynamic Range videos using varying exposure settings for example. And finally the cameras must be capable of capturing images at a rate of at least $s = 8$ frames per second.

The total data rate produced by the twelve cameras in our setup is 1111.8 $MBit/s$ over a duration of up to 5 seconds. This data rate can be handled by two 1394b interface cards and the amount of data produced conveniently fits into the main memory of a modern PC.

9.2 Capturing Videos for Inspection

This section describes in detail how high-resolution videos of PCAs can be captured and preprocessed in order to be used for video-based automatic optical inspection. We start with an overview of the coordinate systems involved and their relationships in terms of mathematical transformations in Section 9.2.1.

In a first offline step, the cameras in the array need to be calibrated with respect to each other and the conveyor band. For this step, we use a calibration board tailored to our camera array. The board and the calibration process are described in Section 9.2.2.

In the online phase of the VAOI system, a PCA to be tested is transported into the system where it triggers a light barrier and starts the capturing process. Periodic trigger signals are sent to all cameras in the array and all light sources until the PCA exits the VAOI unit. In each cycle, a row consisting of n images is captured by the n cameras. We denote the number of total rows captured by m. It is important that all cameras are triggered at exactly the same time so the relative positions of the images in one row correspond to those determined in the calibration process. The video consisting of $m \cdot n$ frames is first captured into the main memory of the PC the cameras are connected to. Preprocessing of the video starts once capturing is completed. The preprocessing mainly consists of estimating the transformations between images of the video. After this, one last transformation needs to be computed that relates the "big picture" with the CAD description of the PCA. This registration step is described in full detail in Section 9.2.3.

Once capturing and preprocessing is done, the video can be used for inspection. How this can be done is beyond the scope of this thesis. We end this Section with an example of how our VAOI system can be used to capture high dynamic range videos of PCAs.

9.2.1 Coordinate Systems and Transformations

A multitude of two-dimensional coordinate systems are involved in capturing videos of a moving PCA. Each type of PCA has its own coordinate system called the *CAD coordinate system*. It is used to describe positions and sizes of components placed on the PCA, which is important for AOI. Its unit is usually a physical unit of length and its axes and origin can be arbitrarily placed on the PCA.

Every camera of the array has a pixel coordinate system with the origin residing in the top left pixel of the camera image and the positive horizontal and vertical axes pointing right and down respectively. We refer to them as *camera coordinate systems*.

For the sake of understandability, we imagine the PCA to be standing still on the conveyor and the camera array moving once over the entire PCA while capturing $m \cdot n$ images. It then becomes clear that each image has its own *image coordinate system*. The coordinate systems of the first row of images are identical to the camera coordinate systems. The coordinate systems of each subsequent row of captured images are then related to the camera coordinate systems by a Euclidean transformation. A point given in image coordinates is denoted by a homogeneous 2D coordinate (a 3-tuple) x. It should be noted that contrary to the conventions in the previous chapters where x generally was a scalar value representing the horizontal coordinate, it is a homogeneous vector in this chapter. We omit explicitly writing homogeneous coordinates as 3-tuples $(x, y, 1)$ to avoid clutter and simply use x instead.

We introduce an additional virtual 2D coordinate system between CAD and the images which we call the *tester coordinate system*. It lies in the plane defined by the conveyor band and is established during the camera calibration process. It serves as an intermediate coordinate system to simplify image registration. Points given in tester coordinates are denoted by x'.

Using homogeneous coordinates, a point in any of these coordinate systems is a tuple with three components. Let $\bar{x}$ be a point on a PCA, specified in CAD coordinates. It is then represented in tester coordinates as *one* 3-tuple x', since we imagine the board to be standing still. For $i = 1, \ldots, n$ and $j = 1, \ldots, m$ the 3-tuple $x_{i,j}$ represents the same point in the coordinate system of image $I_{i,j}$, where $I_{i,j}$ is the jth image captured by camera i.

Transformations between the various coordinate systems can be expressed by 3×3 matrices. In the whole process of capturing and aligning $m \cdot n$ images, there are $m \cdot n + n + 1$ matrices involved. The *image matrix* $M_{i,j}$ transforms a point from tester coordinates into the coordinate system of image $I_{i,j}$:

$$M_{i,j}x' = x_{i,j},\ i = 1, \ldots, n,\ j = 1, \ldots, m. \tag{9.7}$$

Once during the calibration process, the *camera matrices* N_i are established, that transform tester coordinates to the coordinate system of camera i (see Section 9.2.2). The first row of image matrices are set to the camera matrices:

$$M_{i,1} := N_i,\ i = 1, \ldots, n. \tag{9.8}$$

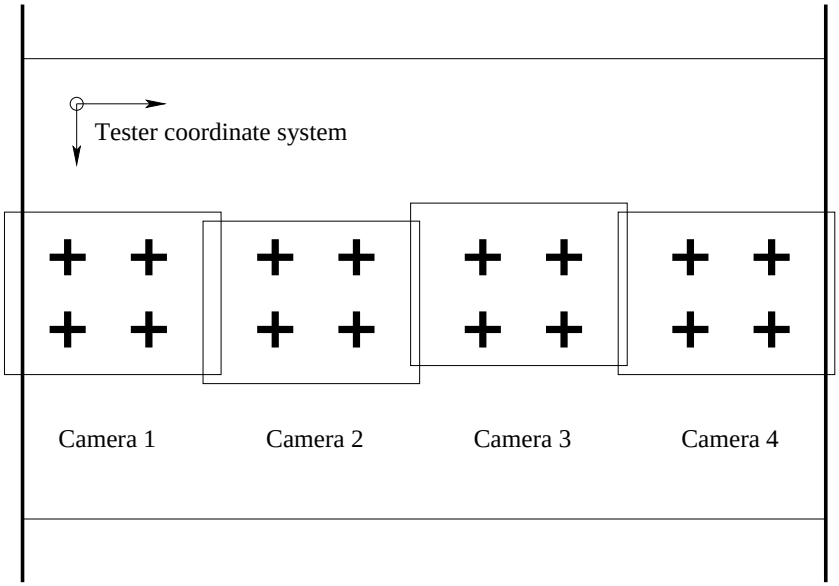


Figure 9.3: *Calibration board placed on the conveyor belt underneath the camera array. The cross-shaped fiducial marks are printed onto the board so that each camera can see at least four marks. Their coordinates are specified in an arbitrary coordinate system which later constitutes the tester coordinate system.*

All further matrices must be estimated in the registration process as described later. One more matrix, the *CAD matrix* C, transforms CAD coordinates into tester coordinates:

$$C\bar{x} = x'. \tag{9.9}$$

It is estimated in the final step of image registration.
The matrices $M_{i,j}$ and N_i represent projective transformations. The $M_{i,j}$ are interrelated by Euclidean transformations, which becomes clear when imagining the cameras to be moved over the fixed PCA. C is a similarity transformation, consisting of a Euclidean transformation with scaling. Its Euclidean part can be explained by the PCA residing fixed in the tester, translated and rotated relative to the tester coordinate system. The scaling is due to the potentially differing units of length used in the CAD and tester coordinates. Note that all matrices are invertible and can also be used to transform coordinates in the opposite direction.
The results of the entire capturing and registration process described throughout this chapter are thus $m \cdot n$ images $I_{i,j}$ with corresponding image matrices $M_{i,j}$ and a CAD matrix C. Using the matrices $M_{i,j}$, a single high-resolution image of the PCA similar to a panoramic image can easily be obtained. However for AOI, this step is unnecessary.

9.2.2 Camera Calibration

Before any videos can be captured with the VAOI system, the array of cameras must be calibrated once. This means to estimate the camera matrices N_i, that relate the pixels

of all cameras to positions in the common tester coordinate system. The matrices N_i are saved and used in image registration later.
For calibration, we place a *calibration board* on the conveyor band, which is similar to a PCA in size. A number of fiducial marks is printed onto the calibration board and it is positioned in a way that allows each camera to see at least four marks. An example calibration board with cross-shaped marks and the fields of view of four cameras is shown in Figure 9.3. The positions $x_{i,1}$ to $x_{i,4}$ of the four fiducial marks in pixel coordinates of camera i can be accurately detected by template matching, thresholding and computation of centers of gravity.
Let $x'_{i,1}$ to $x'_{i,4}$ be the coordinates of the marks on the calibration board in a fixed coordinate system with arbitrary origin and scale. These coordinates must be known prior to calibration. The arbitrary coordinate system constitutes the intermediate tester coordinate system.
For each camera i, the eight parameters of the camera matrix N_i are calculated by solving the system of equations

$$N_i x'_{i,k} = x_{i,k}, \ k = 1, \ldots, 4. \tag{9.10}$$

9.2.3 Image Registration for Video-AOI

As mentioned before, we imagine the PCA to be standing still on the conveyor while the camera array is moved for scanning. From a mathematical point of view, this scenario is identical to a moving PCA and stationary cameras, so the choice between the two philosophies is arbitrary. We believe that our view helps the comprehensibility.
The problem of registration can be formulated as the estimation of the image matrices $M_{i,j}$ and the CAD matrix C. The matrices $M_{i,j}$ relate the pixel coordinates of image $I_{i,j}$ to tester coordinates. For the first row of images, the image matrices are identical to the calibration matrices. Every additional row of image matrices can then be obtained by multiplying the matrices of the previous row by a Euclidean matrix which is estimated from two subsequent images from the same camera. Three approaches to the problem of estimating $M_{i,j}$ will be introduced in the following sections. In the end, only *one* mapping C between tester and CAD coordinates needs to be estimated.

Fiducial-based Registration

We now show how the image matrices $M_{i,j}$ can be estimated by setting $M_{i,1} := N_i$ for $i = 1, \ldots, n$ and showing a method for estimating $M_{i,j+1}$ from already known $M_{i,j}$ for $j = 1, \ldots, m-1$. For this, we assume that in each pair of subsequent image rows $(I_{1,j}, \ldots, I_{n,j})$ and $(I_{1,j+1}, \ldots, I_{n,j+1})$ there are two fiducial marks that are visible in any two images of row j and the corresponding two images of row $j+1$. Let k be the camera index so that the images $I_{k,j}$ and $I_{k,j+1}$ contain the first mark, and let l be the index so that $I_{l,j}$ and $I_{l,j+1}$ contain the second. The pixel coordinates $x_{k,j}, x_{k,j+1}, x_{l,j}, x_{l,j+1}$ of the marks in the four images are determined through template matching just like during calibration. We must now estimate the matrices $M_{k,j+1}$ so that

$$M_{k,j+1}^{-1} x_{k,j+1} = M_{k,j}^{-1} x_{k,j}. \tag{9.11}$$

Index l likewise. Equation 9.11 means that the pixel coordinates of a mark in two images must both map to the same position in the tester coordinate system.
We accomplish this by first transforming $x_{k,j+1}$ and $x_{k,j}$ by the known matrix $M_{k,j}^{-1}$ and transforming $x_{l,j+1}$ and $x_{l,j}$ by the known matrix $M_{l,j}^{-1}$ into tester coordinates. We then estimate a Euclidean transformation T that maps the transformed coordinates onto each other:

$$TM_{k,j}^{-1}x_{k,j+1} = M_{k,j}^{-1}x_{k,j} \quad \wedge \quad TM_{l,j}^{-1}x_{l,j+1} = M_{l,j}^{-1}x_{l,j}. \tag{9.12}$$

This system of four equations has three variables and can be solved by non-linear least-squares fitting. We now set $M_{k,j+1} := M_{k,j}T^{-1}$, so that $M_{k,j+1}^{-1} = TM_{k,j}^{-1}$. It follows, that

$$M_{k,j+1}^{-1}x_{k,j+1} = TM_{k,j}^{-1}x_{k,j+1} = M_{k,j}^{-1}x_{k,j}. \tag{9.13}$$

It can be seen that $M_{k,j+1}$ fulfills Equation 9.11 as required. Index l likewise. Using the same matrix T, all other $M_{i,j+1}$ are now calculated as: $M_{i,j+1} := M_{i,j}T^{-1}$
Adding fiducial marks that can be used for fiducial-based registration to a PCA can be done by putting the PCA into a fixture that already contains the required fiducials. Mounting PCAs in such a way has to be done manually, which can be an unacceptable drawback in some production lines. The advantage of fiducial-based registration is clearly its speed. The indices k and l of the cameras that can see the fiducial marks as well as their approximate position in the camera's field of view are usually known. In a relatively small search area properly printed marks on a fixture can be detected quickly and robustly. Estimating the $M_{i,j}$ is even faster, as is shown in Section 9.3.

Feature-based Registration

The feature-based registration is similar to the fiducial-based version. Again, we set $M_{i,1} := N_i$ for $i = 1, \ldots, n$ and show a method for estimating $M_{i,j+1}$ from already known $M_{i,j}$ for $j = 1, \ldots, m-1$.
For the feature-based registration, we relax the requirement of having fiducial marks and only assume that detectable features like corners and dots are present in the images. We use Harris feature points [52], SIFT [73] and RANSAC [32] for feature detection and matching. For each pair of subsequent images $I_{i,j}$ and $I_{i,j+1}$, $\forall i$, we obtain is a list of coordinate pairs $(x_{i,j}^{(k)},\ x_{i,j+1}^{(k)})$. For each k, this pair represents the coordinates of a feature that has been detected in two subsequent images. We refer to this pair as a *feature match*. Similar to Equation 9.11, $M_{i,j+1}$ must be estimated, such that both coordinates of a match are transformed to the same tester coordinates:

$$M_{i,j+1}^{-1}x_{i,j+1}^{(k)} = M_{i,j}^{-1}x_{i,j}^{(k)},\ \forall i, k. \tag{9.14}$$

In practice, only a small number of feature matches in only two distant images must be considered. The more features and images are considered, the higher the accuracy.
Again, we transform both coordinates of a match $(x_{i,j}^{(k)},\ x_{i,j+1}^{(k)})$ by the same known matrix $M_{i,j}^{-1}$ and estimate *one* Euclidean transformation T, that approximates

$$TM_{i,j}^{-1}x_{i,j+1}^{(k)} \approx M_{i,j}^{-1}x_{i,j}^{(k)},\ \forall i, k. \tag{9.15}$$

This system of three variables and two equations per feature match can be approximated using non-linear least-squares fitting. Again, by setting $M_{i,j+1} := M_{i,j}T^{-1}$, we get

$$M_{i,j+1}^{-1}x_{i,j+1} = TM_{i,j}^{-1}x_{i,j+1} \approx M_{i,j}^{-1}x_{i,j} \tag{9.16}$$

and approximate Equation 9.14. The quality of this approximation is evaluated in Section 9.3.
Note that the detection and matching of features is less accurate and robust than the detection of fiducial marks. We therefore use a higher number of matches (in the magnitude of tens or hundreds) for an average, and the correspondence in Equation 9.14 cannot be achieved perfectly for each of the matches. In order to be able to detect enough feature matches, sufficiently structured PCAs and a higher vertical multiplicity m_v than for fiducial-based registration are required. The latter leads to a higher frame rate requirement to retain the same conveyor velocity as can be seen in Equation 9.6. In addition, the process of detecting, matching and selecting suitable features is computationally expensive. As an advantage, the overall accuracy is higher than for fiducial-based registration due to averaging over all the feature matches considered. Another major advantage is the independence of fiducial marks on the PCA, allowing feature-based registration to be used without fixture and inline.

Reference-based Registration

In order to cope with high conveyor speeds using a small m_v while still being mostly independent of a fixture, we developed a third alignment mechanism called reference-based registration.
Here, we use a fixture with fiducial marks only *once* for each type of PCA to be inspected. A reference PCA is manually mounted to a fixture, and a reference video is captured and registered based on the fiducial marks on the tray. Its images $J_{i,j}$ and image matrices $R_{i,j}$ are saved and later used as a reference for alignment. For a fixed i and j, the image $I_{i,j}$ captured of a PCA to be inspected overlaps with $J_{i,j}$ by nearly 100%. The difference is only a Euclidean transformation. We therefore estimate $M_{i,j}$ using $J_{i,j}$ and $R_{i,j}$. Since only images and matrices with the same i and j are used at a time, we omit the indices here for simplicity's sake.
We first detect feature matches $(x_{(k)},\ y_{(k)})$ in the images I and J as was done in the previous section. M must be estimated, so that

$$M^{-1}x_{(k)} = R^{-1}y_{(k)},\ \forall k. \tag{9.17}$$

Using the same method as before, we estimate a Euclidean transformation T, such that

$$TR^{-1}x_{(k)} \approx R^{-1}y_{(k)},\ \forall k. \tag{9.18}$$

Setting $M := RT^{-1}$ yields

$$M^{-1}x_{(k)} = TR^{-1}x_{(k)} \approx R^{-1}y_{(k)},\ \forall k, \tag{9.19}$$

and Equation 9.17 is approximated. In practice, T is approximately equal for all i and must be computed only once using a small set of images.
This registration method allows for conveyor speeds as fast as those achieved by fiducial-based registration while taking as much processing time as the feature-based approach. No additional fiducial marks need to be put on the PCAs in the running system. The overall accuracy is limited by the accuracy achieved by the initial registration of the reference video.

Mapping to CAD Coordinates

So far we introduced different approaches of obtaining the matrices $M_{i,j}$ that transform tester coordinates into pixel coordinates. As a last step once this is done, a similarity transformation C is computed that performs the final mapping between CAD coordinates and tester coordinates (see Equation 9.9). The four parameters of C denote the position of the PCA inside the tester (in our view of a moving camera array), its rotation and the difference in scale of the two coordinate systems.
To define the CAD coordinate system, a PCA always has at least two fiducial marks with known CAD coordinates, which are also used for populating the PCA. The indices of the images in which they appear are manually selected once when examining the captured video of a reference PCA. Since all future PCAs will be captured under similar conditions, knowledge about potential search areas for the fiducial marks gained from the reference video can be used to facilitate the fiducial detection mechanism during operation.
Let $\bar{x}_{(k)}$, $k = 1, 2$ be the CAD coordinates of two fiducial marks. Let $x_{(k)}$ be the pixel positions at which the fiducials have been detected in the images $I_{(k)}$. We transform them into tester coordinates $x'_{(k)}$ using the image matrices $M_{(k)}$:

$$x'_{(k)} := M_{(k)}^{-1} x_{(k)}, \ \forall k. \tag{9.20}$$

We can now calculate the four parameters of C by solving the following system of four equations:

$$C\bar{x}_{(k)} = x'_{(k)}, \ \forall k. \tag{9.21}$$

9.2.4 Using Videos for Inspection

Knowing C and all matrices $M_{i,j}$, coordinates can be freely transformed between the various systems. This permits the inspection of PCA components using the video captured as images $I_{i,j}$. Each component to be inspected will be visible in $m_h \cdot m_v$ images in the average. We obtain a component's bounding box from the CAD data of the PCA. For inspection, we create roughly $m_h \cdot m_v$ temporary images containing exactly the component, captured under varying application-dependent viewing conditions like angle, time, camera settings and lighting. The size of the images is easily obtained by multiplying the bounding box size by the resolution r. For each pixel in the temporary image, we calculate the corresponding CAD position by linear interpolation of the bounding box coordinates.

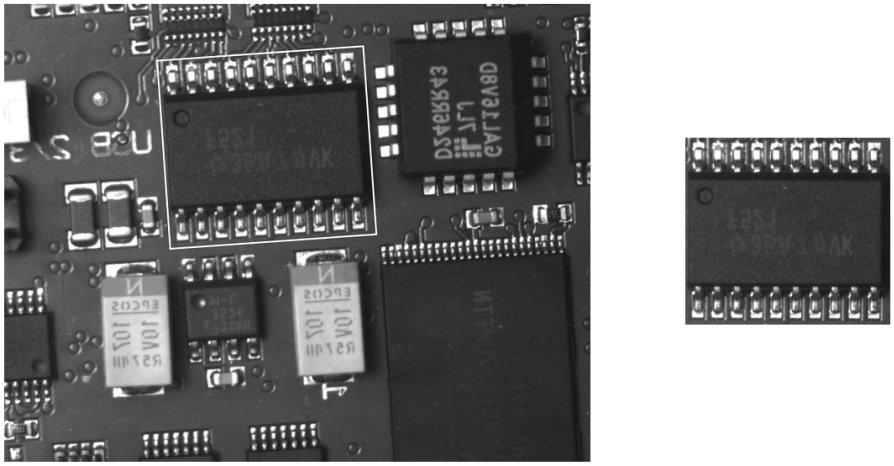


Figure 9.4: *The left image shows a part of a PCA as seen by a camera. The camera's rotation was exaggerated for clarity. The white box represents the bounding box of a component. It can be seen that the CAD coordinate system is rotated and translated with respect to the camera's pixel coordinates. The right image is a temporary image created for inspection. Its coordinates are aligned with the bounding box.*

Let $\bar{x}$ be the CAD coordinate corresponding to a pixel position. We transform it into tester coordinates $x' = C\bar{x}$. The selection of a source image from which the color value is retrieved is highly application dependent. Generally speaking, indices i and j must be determined, so that

$$M_{i,j}x' = x \in [0,\ p_h - 1] \times [0,\ p_v - 1]. \tag{9.22}$$

The color value at x in image $I_{i,j}$ is calculated using bi-linear interpolation and inserted into the temporary image. This process must be repeated for each pixel of each temporary image.
In the new images, pixel positions can be easily mapped to CAD coordinates and vice-versa using linear interpolation. This allows for efficient AOI. See Figure 9.4 for an exemplary camera image and a temporary image that was created for inspection.

9.2.5 Capturing HDR Video in a Video-AOI System

HDR video techniques can be applied to assist optical inspection. When capturing videos of a PCA for inspection, achieving proper lighting is difficult. An integrated circuit for instance may have highly reflective metal pins and a dark gray label on a black surface. For an industrial camera with a CCD sensor with linear response, it may be difficult to find a suitable shutter speed that shows details in dark and bright areas at the same time. A similar example was the capacitor shown in Figure 9.1. We thus use our Video-AOI prototype to capture two videos of the PCA using two different shutter settings and combine them into one video covering a higher dynamic range.
For this purpose, we set the horizontal multiplicity to $m_h = 2.2$ and set m_v to an arbitrary high value. We set each camera with an even index to a short shutter speed and cameras

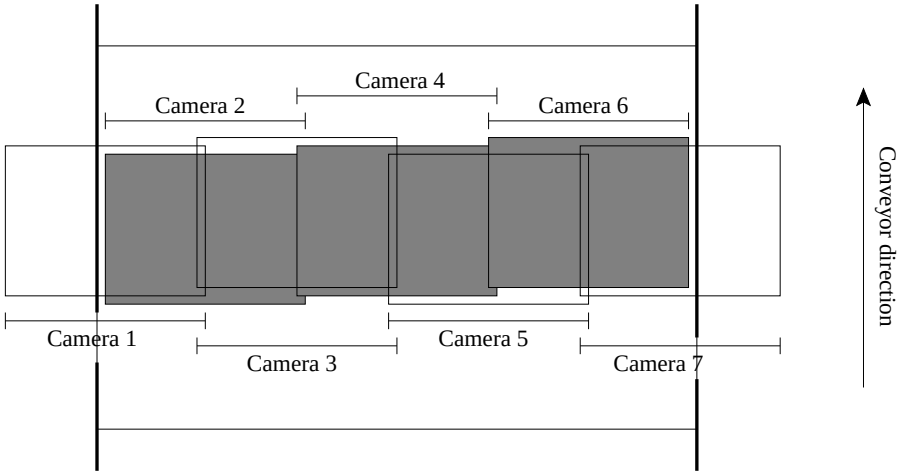


Figure 9.5: *A row of images of a PCA on the conveyor is captured by seven cameras. The horizontal multiplicity is $m_h = 2.2$. Cameras with an even index have a lower shutter speed setting, resulting in a darker image. This assures that each position on the PCA is visible in one bright and one dark image.*

with odd index to a longer value within the upper bound specified in Equation 9.5. Like this, two subsequent images captured by the same camera have the same brightness level which increases the robustness of feature-based registration. This is illustrated in Figure 9.5.
Due to the horizontal image overlap of $(1 - \frac{1}{m_h}) \approx 55\%$, each position on the PCA is guaranteed to be contained in at least one bright and one dark image. When creating a temporary image as described in the previous section, Equation 9.22 will be satisfied for an odd and an even index i for each pixel. We retrieve both the dark and the bright pixel value and combine them into a HDR pixel according to Section 2.5 in the technical introduction of this thesis. m_v temporary HDR images are created for each component to be inspected.

9.3 Experimental Results

We used our VAOI prototype to perform measurements of the time taken for registration and the accuracy achieved. The tests were done with a camera resolution of 1032×776 and a spatial resolution of $r = 12$ pixels per millimeter. The other parameters remained unchanged.
Four PCAs of the same type were used. Their width is 45 mm and their height 215 mm. The width was small enough to be captured with a single camera in this setup. With a vertical multiplicity of 2.1, this resulted in seven rows of one image per row. The first board was used as a reference and registered using feature points. The other three were registered based on the reference video. In each image, we selected five components that were visible in the top left and right corner, the bottom left and right corner and

the center of the image. The real position of each component as seen in the images was selected manually using a mouse and compared to the estimated position obtained by transforming the component's CAD position by the estimated matrices. The average error over all 35 components was calculated. Out of the four videos, the reference video had the lowest total error, as expected. Its total registration error was 0.53 mm. For the other three videos, the error was 0.59 mm, 0.64 mm and 1.23 mm respectively. Since registration is a pixel-based operation, this error is inversely proportional to the resolution r.
We processed the captured video on a PC with an Intel Quad Core CPU with 2.4GHz. Detecting Harris feature points in a full image and computing SIFT keys took 300 ms. The feature threshold was set to a value so that roughly 600 features were detected. Matching them with the same number of features in another image took 220 ms.
For reference, detecting a fiducial mark in a full image took 135 ms. The computation time is proportional to the size of the search area. Prior knowledge about fiducial positions will thus speed up the process significantly.
Estimating a Euclidean transformation from k feature matches took about $k \cdot 0.05$ ms with a lower bound of 0.5 ms for exactly two features, as is the case for fiducial-based registration.

9.4 Conclusions

We showed how a prototype for video-based optical inspection of PCAs can be built. We gave an overview of all the parameters involved and gave advice on how they can be set. The process of capturing high-resolution videos for AOI was described. The focus was put on preprocessing the videos in a way that allows to locate parts of the PCA in the captured images.
Future work includes the development of inspection techniques that benefit from the video aspect of our system. We also aim to conduct more detailed measurements of the performance of our prototype. In this process, we hope to speed up the stitching and increase its accuracy.

Chapter 10

Conclusions and Outlook

We presented a real-time high dynamic range video system that creates HDR frames from multiple low dynamic range exposures, captured under varying shutter speeds. The steps necessary to create an HDR frame are: determining suitable shutter values, capturing a sequence of LDR exposures, compensating intermediate camera motion, merging the LDR images into an HDR frame, and tone mapping the result for display. We contributed to the fields of shutter speed calculation, capturing, image registration, and tone mapping, while existing techniques were used for frame merging. Our proposed techniques focus on saving capturing and processing time and achieving temporal consistency between the frames of an HDR video. They benefit from knowledge gained from the previous frames.

The shutter values were calculated from the log radiance histogram of the scene. They were chosen such that frequently occurring radiance values are covered by an LDR image that exposes the radiance range well. Taking the entire histogram of the scene into account is an improvement over existing techniques which only consider minimum, maximum or average brightness. Being adaptive to the scene allows to capture only as few exposures as necessary. The introduced stability criterion for the shutter speeds prevents oscillation of the values which would otherwise lead to flicker. It was also shown that stable shutter sequences are desirable for more efficient capturing. Our criterion allows to compromise between stability and adaptability to changes in the scene. A subjective user study showed that our shutter values result in HDR images with a higher quality than those created from the traditionally used equidistant shutters for the same number of exposures. Likewise, fewer exposures are required for a desired image quality which saves capturing time.

We made the observation that the parts of a scene requiring HDR often only cover small image areas. When acquiring multiple LDR image of the scene, a large amount of time can be saved by only selectively re-exposing these small areas. This avoids transmitting and processing image material that does not contribute to the quality of the HDR result

anyway. By interleaving capturing and image analysis, badly exposed areas are identified and only re-exposed as needed. 20% to 49% of the time to create an HDR frame was saved by using this method.

Motion compensation for an exposure sequence is a computationally costly operation. We introduced modifications to an existing still-image registration technique to simultaneously achieve faster computation times and better accuracy. This was made possible by simplifying the still-image registration to a fast heuristic while exploiting knowledge gained from the camera motion of the previous frames to keep the accuracy high. Our algorithm is based on the normalized cross correlation between horizontal and vertical projection profiles of the two images to register. Robustness to the large brightness difference among the images of an exposure sequence is accomplished by thresholding them such that 50% of the pixels are white and 50% are black. Compared to the still-image approach, computation time was improved by a factor of 1.4 to 3 while reducing the average registration error by 30% in our tests.

In a subjective user study, we found that flicker is the most disturbing artifact introduced when applying still image tone mapping operators to the frames of an HDR video. Flicker occurs when the scene brightness changes drastically from one frame to the next. The changed radiance range is then mapped to the same display range, resulting in a sudden change of brightness of the tone mapped result. This is perceived as flicker. We showed how flicker can be detected by checking the variation of the log average image brightness against a threshold based on Stevens' power law. Flicker was then removed by smoothing large brightness differences over several frames. This was done as a post-processing step to tone mapping by modifying the image normalization which is often included as a last step of such operators. In our test scenario, 49 out of 50 flickering artifacts reported by viewers were removed.

To further speed up the creation of HDR frames, we analyzed the employed HDR algorithms with respect to necessity and suitability for a GPU implementation using CUDA. We demonstrated the adjustments to the algorithms that were necessary for their parallelization. In a 30 second demo HDR video, the processing time was sped up by a factor of 15 over its pure CPU counterpart. We achieved an average frame rate of 23 frames per second. Almost 70% of the time taken to create HDR frames was spent for capturing. This justifies the effort we put into improving the acquisition. Large portions of the capturing time were in turn caused by the exposure time itself. As a consequence the frame rate depended more strongly on the illumination level of the scene than on its dynamic range. Our system accomplished frame rates between 20 and 50 fps in a very bright HDR setting and in an LDR setting with medium brightness. On the other hand, the rate dropped down to 13 fps in a low-light HDR situation where the exposure time of the LDR frames alone already constituted 55 ms (corresponding to an upper bound for the frame rate of 18 fps). Using a camera lens that collects more light shifts the bottleneck of HDR frame creation back towards capturing overhead and processing time.

As an application of HDR video, we presented a prototype for the automatic optical inspection of printed circuit assemblies. It uses multiple cameras with overlapping fields of view to capture HDR video of a PCA. HDR has the potential to improve optical

inspection results for highly reflective components which would be saturated otherwise.
During our work, we recognized a limitation of the IIDC standard, the FireWire interface for digital cameras. Transmitting exposure parameters to the camera one by one and waiting for acknowledgments makes the setting of an exposure sequence inefficient. As future work, we would like to look into the details of the frame grabber library and the FireWire driver implementation we used. By modifying the standard, it may be possible to reduce the overhead for setting the acquisition parameters. Furthermore, employing a "smart camera" may permit to perform the shutter sequence calculation on the capturing device directly.
Obtaining HDR video from a camera that needs to be connected to a PC is cumbersome in practice. In the future, more and more parts of the HDR pipeline should be shifted into the camera. Ideally, the camera would determine shutter speed sequences and capture images on its own. Image registration may become mostly obsolete due to the low delays between capturing. HDR stitching is expected to be well-suited for an on-chip implementation. The camera could then output HDR video directly which would ideally be displayed on an HDR screen without the need for tone mapping.
We would like to investigate the possibilities offered by applying our HDR acquisition techniques to 3D video. In particular, igital representations of real actors are often used for special effects in situations where filming a real actor is too expensive, too dangerous or outright impossible. This can be done by a combination of 3D geometry scanning and photography under a very large number of different lighting conditions. Specular reflections on the skin surface cause problems for scanning under certain lighting angles, necessitating HDR. Since a large number of HDR images must be taken to capture a face under as many different lighting conditions as possible, the algorithms presented in this thesis may prove useful to quicken the scanning process.

References

[1] http://ls.wim.uni-mannheim.de/de/pi4/research/projects/projekte/videos/.

[2] P.M. Acosta-Serafini. *Predictive multiple sampling algorithm with overlapping integration intervals for linear wide dynamic range integrating image sensors.* PhD thesis, Massachusetts Institute of Technology, 2004.

[3] M. Aggarwal and N. Ahuja. Split aperture imaging for high dynamic range. *Int. Journal of Computer Vision*, 58(1):7–17, 2004.

[4] P.M.Q. Aguiar. Unsupervised simultaneous registration and exposure correction. In *Proc. of the IEEE Int. Conference on Image Processing*, pages 361–364, 2006.

[5] M. Ashikhmin. A tone mapping algorithm for high contrast images. In *Proc. of the 13th Eurographics Workshop on Rendering*, pages 145–156, 2002.

[6] M. Ashikhmin and J. Goyal. A reality check for tone-mapping operators. *ACM Trans. Appl. Percept.*, 3(4):399–411, 2006.

[7] A. Bab-Hadiashar and D. Suter. Robust optic flow computation. *Int. Journal of Computer Vision*, 29(1):59–77, 1998.

[8] S. Baker, R. Gross, and I. Matthews. Lucas-kanade 20 years on: A unifying framework: Part 3. Technical Report CMU-RI-TR-03-35, Carnegie Mellon University Robotics Institute, 2003.

[9] N. Barakat, A. N. Hone, and T. E. Darcie. Minimal-bracketing sets for high-dynamic-range image capture. *IEEE Trans. on Image Processing*, 17(10), 2008.

[10] S.S. Beauchemin and J.L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–466, 1995.

[11] A. Benoit, D. Alleysson, J. Herault, and P. Callet. *Spatio-temporal tone mapping operator based on a retina model*, pages 12–22. Springer-Verlag, Berlin, Heidelberg, 2009.

[12] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. of the ECCV*, pages 237–252, 1992.

[13] M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[14] R. Bogart, F. Kainz, and D. Hess. OpenEXR image file format. *ACM SIGGRAPH, Sketches & Applications*, 2003.

[15] V. Brajovic and T. Kanade. A sorting image sensor: An example of massively parallel intensity-to-time processing for low-latency computational sensors. In *Proc. of the IEEE Int. Conference on Robotics and Automation*, volume 2, pages 1638–1643, 1996.

[16] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *Int. Journal of Computer Vision*, 61(3):211–231, 2005.

[17] P.J. Burt and R.J. Kolczynski. Enhanced image capture through fusion. In *Proc. of the 4th Int. Conference on Computer Vision*, pages 173–182, 1993.

[18] M. Cadik, M. Wimmer, L. Neumann, and A. Artusi. Evaluation of HDR tone mapping methods using essential perceptual attributes. *Computers & Graphics*, 32(3):330–349, 2008.

[19] T. Chen and A. El Gamal. Optimal scheduling of capture times in a multiple capture imaging system. In *Proc. of the SPIE Electronic Imaging Conference*, 2002.

[20] H. Cho and O.K. Kwon. A backlight dimming algorithm for low power and high image quality LCD applications. *IEEE Trans. on Consumer Electronics*, 55(2):839–844, 2009.

[21] T.H. Cormen. *Introduction to algorithms*. The MIT Press, 2001.

[22] E. Culurciello, R. Etienne-Cummings, and K. Boahen. Arbitrated address-event representation digital image sensor. *Electronics Letters*, 37(24):1443–1445, 2001.

[23] J. Davis. Mosaics of scenes with moving objects. In *Proc. of the CVPR*, pages 354–360, 1998.

[24] P.E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proc. of the 24th Conference on Computer Graphics and Interactive Techniques*, 1997.

[25] K. Devlin, A. Chalmers, A. Wilkie, and W. Purgathofer. Tone reproduction and physically based spectral rendering. In *State of the Art Reports, Eurographics*, pages 101–123, 2002.

[26] F. Drago, K. Myszkowski, T. Annen, and N. Chiba. Adaptive logarithmic mapping for displaying high contrast scenes. In *Computer Graphics Forum*, volume 22, pages 419–426, 2003.

[27] F. Durand and J. Dorsey. Interactive tone mapping. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, pages 219–230, 2000.

[28] European Broadcasting Union. SAMVIQ - Subjective assessment methodology for video quality. Report by the EBU Project Group B/VIM, May 2003.

[29] R. Fattal, D. Lischinski, and M. Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, 21(3):249–256, 2002.

[30] J.A. Ferwerda. Elements of early vision for computer graphics. *IEEE Computer Graphics Applications*, 21(5):22–33, 2001.

[31] J.A. Ferwerda, S.N. Pattanaik, P. Shirley, and D.P. Greenberg. A model of visual adaptation for realistic image synthesis. In *Proc. of the SIGGRAPH*, pages 249–258, 1996.

[32] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[33] D.J. Fleet and Y. Weiss. Optical flow estimation. *Handbook of Mathematical Models in Computer Vision*, pages 239–258, 2005.

[34] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[35] C.S. Fuh and P. Maragos. Motion displacement estimation using an affine model for image matching. *Optical Engineering*, 30(7):881–887, 1991.

[36] O. Gallo, N. Gelfandz, W.C. Chen, M. Tico, and K. Pulli. Artifact-free high dynamic range imaging. In *Proc. of the IEEE Int. Conference on Computational Photography (ICCP)*, pages 1–7, 2009.

[37] M.A. Gennert. Brightness-based stereo matching. In *Proc. of the 2nd International Conference on Computer Vision*, pages 139–143, 1988.

[38] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys. Interactive time-dependent tone mapping using programmable graphics hardware. In *ACM SIGGRAPH Courses*, 2005.

[39] C.M. Goral, K.E. Torrance, D.P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 213–222, 1984.

[40] T. Grosch. Fast and robust high dynamic range image generation with camera and object movement. In *Vision, Modeling, and Visualization*, pages 277–286, 2006.

[41] M. Grossberg and S. Nayar. What can be known about the radiometric response from images? *Computer Vision ECCV*, pages 393–413, 2006.

[42] M.D. Grossberg and S.K. Nayar. High dynamic range from multiple images: Which exposures to combine? In *Proc. of the ICCV Workshop on Color and Photometric Methods in Computer Vision (CPMCV)*, 2003.

[43] M.D. Grossberg and S.K. Nayar. What is the space of camera response functions? In *Proc. of the CVPR*, volume 2, pages 602–609, 2003.

[44] E. Guerra and J.R. Villalobos. A three-dimensional automated visual inspection system for SMT assembly. *Computers & Industrial Engineering*, 40(1-2):175–190, 2001.

[45] B. Guthier, S. Kopf, M. Eble, and W. Effelsberg. Flicker reduction in tone mapped high dynamic range video. In *Proc. of IS&T/SPIE Electronic Imaging (EI) on Color Imaging XVI: Displaying, Processing, Hardcopy, and Applications*, volume 7866, pages 78660C:01 – 78660C:15, 2011.

[46] B. Guthier, S. Kopf, and W. Effelsberg. Capturing high dynamic range images with partial re-exposures. In *Proc. of the IEEE 10th Workshop on Multimedia Signal Processing (MMSP)*, 2008.

[47] B. Guthier, S. Kopf, and W. Effelsberg. High-resolution inline video-aoi for printed circuit assemblies. In *Proc. of IS&T/SPIE conference on Image Processing: Machine Vision Applications II*, volume 7251, 2009.

[48] B. Guthier, S. Kopf, and W. Effelsberg. Histogram-based image registration for real-time high dynamic range videos. In *Proc. of IEEE Int. Conference on Image Processing (ICIP)*, pages 145–148, September 2010.

[49] B. Guthier, S. Kopf, and W. Effelsberg. Optimal shutter speed sequences for real-time hdr video. Technical report, University of Mannheim, 2011. http://pi4.informatik.uni-mannheim.de/∼bguthier/optshutter-TR.pdf.

[50] B. Guthier, S. Kopf, M. Wichtlhuber, and W. Effelsberg. Parallel algorithms for histogram-based image registration. In *Proc. of IEEE Int. Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 182–185, April 2012.

[51] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[52] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[53] S.W. Hasinoff, F. Durand, and W.T. Freeman. Noise-Optimal Capture for High Dynamic Range Photography. In *Proc. of the 23rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[54] K. Hirakawa and P.J. Wolfe. Optimal exposure control for high dynamic range imaging. In *Proc. of the 17th IEEE International Conference on Image Processing (ICIP)*, 2010.

[55] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[56] S. Hua and L. Wang. Photographs alignment and high dynamic range image composition based on varying exposure levels. In *16th Int. Conference on Artificial Reality and Telexistence (ICAT)*, volume 4282, pages 1146–1155, 2006.

[57] Eiichiro Ikeda. *Image data processing apparatus for processing combined image signals in order to extend dynamic range.* U.S. Patent 5801773, September 1998.

[58] D. Ilstrup and R. Manduchi. One-shot optimal exposure control. In *Proc. of the 11th European Conference on Computer Vision (ECCV)*. 2010.

[59] International Telecommunication Union (ITU). Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange. *ITU-R Recommendation BT.709*, 1990.

[60] S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High dynamic range video. *ACM Transactions on Graphics (TOG)*, 22(3):319 – 325, 2003.

[61] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts. A logarithmic response CMOS image sensor with on-chip calibration. *IEEE Journal of Solid-State Circuits*, 35(8):1146–1152, 2000.

[62] E.A. Khan, A.O. Akyüz, and E. Reinhard. Ghost removal in high dynamic range images. In *Proc. of the ICIP*, pages 2005–2008. IEEE, 2006.

[63] S.J. Kim and M. Pollefeys. Radiometric alignment of image sequences. In *Proc. of the CVPR*, pages 645–651, 2004.

[64] S. Kishimoto, N. Kakimori, Y. Yamamoto, Y. Takahashi, T. Harada, Y. Iwata, Y. Shigeyama, and T. Nakao. A printed circuit board (PCB) inspection system employing the multi-lighting optical system. In *Electronic Manufacturing Technology Symposium*, pages 120–129, 1990.

[65] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. of the 14th Int. Conf. on Artificial intelligence*, pages 1137–1143, 1995.

[66] C. Kolb, D. Mitchell, and P. Hanrahan. A realistic camera model for computer graphics. In *Proc. of the 22nd conference on Computer Graphics and Interactive Techniques*, pages 317–324. ACM, 1995.

[67] G. Krawczyk, K. Myszkowski, and H.P. Seidel. Perceptual effects in real-time tone mapping. In *Proc. of the 21st spring conference on Computer graphics*, pages 195–202, 2005.

[68] P Ledda, A Chalmers, T. Troscianko, and H. Seetzen. Evaluation of tone mapping operators using a High Dynamic Range display. *ACM Transactions on Graphics*, 24(3):640–648, 2005.

[69] C. Lee and C.S. Kim. Gradient domain tone mapping of high dynamic range videos. In *Proc. of the IEEE Int. Conference on Image Processing (ICIP)*, volume 3, pages 461–464, 2007.

[70] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.*, 2:431–441, 1944.

[71] F.C. Lin, Y.P. Huang, L.Y. Liao, C.Y. Liao, H.P.D. Shieh, T.M. Wang, and S.C. Yeh. Dynamic backlight gamma on high dynamic range LCD TVs. *Journal of Display Technology*, 4(2):139–146, 2008.

[72] X. Liu and A. El Gamal. Simultaneous image formation and motion blur restoration via multiple capture. In *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1841–1844, 2001.

[73] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the 7th Int. Conf. on Computer Vision*, volume 2, pages 1150–1157, 1999.

[74] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, pages 674–679, 1981.

[75] B.C. Madden. Extended intensity range imaging. Technical Report 248, University of Pennsylvania, December 1993.

[76] S. Mann and R.W. Picard. Being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures. In *Proc. of the IS&T 48th Annual Conference*, 1995.

[77] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[78] M. Mase, S. Kawahito, M. Sasaki, Y. Wakamori, and M. Furuta. A wide dynamic range CMOS image sensor with multiple exposure-time signal outputs and 12-bit column-parallel cyclic A/D converters. *IEEE Journal of Solid-State Circuits*, 40(12):2787–2795, 2005.

[79] L. Meylan and S. Süsstrunk. High dynamic range image rendering with a retinex-based adaptive filter. *IEEE Transactions on Image Processing*, 15(9):2820–2830, 2006.

[80] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[81] G.S. Miller and C.R. Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments. *SIGGRAPH: Course Notes for Advanced Computer Graphics Animation*, pages 1–12, 1984.

[82] T. Mitsunaga and S.K. Nayar. Radiometric self calibration. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[83] M. Moganti, F. Ercal, C.H. Dagli, and S. Tsunekawa. Automatic PCB inspection algorithms: a survey. *Computer Vision and Image Understanding (CVIU)*, 63(2):287–313, 1996.

[84] H.P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. *Doctoral Dissertation, Stanford University*, 1980.

[85] S.K. Nayar and V. Branzoi. Adaptive dynamic range imaging: Optical control of pixel exposures over space and time. In *Proc. of the 9th IEEE Int. Conference on Computer Vision*, volume 2, pages 1168 – 1175, 2003.

[86] S.K. Nayar and T. Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 472–479, 2000.

[87] S.A. Nene and S.K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(9):989–1003, 1997.

[88] M. Niskanen. View dependent enhancement of the dynamic range of video. In *Proc. of the 18th Int. Conf. on Pattern Recognition (ICPR)*, volume 1, pages 984–987, 2006.

[89] E.Y. Oh, S.H. Baik, M.H. Sohn, K.D. Kim, H.J. Hong, J.Y. Bang, K.J. Kwon, M.H. Kim, H. Jang, J.K. Yoon, et al. IPS-mode dynamic LCD-TV realization with low black luminance and high contrast by adaptive dynamic image control technology. *Journal of the Society for Information Display*, 13:215, 2005.

[90] D.L. Olson and D. Delen. *Advanced Data Mining Techniques*. Springer, Berlin Heidelberg, 1 edition, 2008.

[91] E.P. Ong and M. Spann. Robust optical flow computation based on least-median-of-squares regression. *Int. Journal of Computer Vision*, 31(1):51–82, 1999.

[92] A.V. Oppenheim, R.W. Schafer, and Jr. Stockham, T.G. Nonlinear filtering of multiplied and convolved signals. *Proceedings of the IEEE*, 56(8):1264–1291, August 1968.

[93] S.N. Pattanaik, J. Tumblin, H. Yee, and D.P. Greenberg. Time-dependent visual adaptation for fast realistic image display. In *Proc. of the 27th Conference on Computer Graphics and Interactive Techniques*, pages 47–54, 2000.

[94] Z.U. Rahman, D.J. Jobson, and G.A. Woodell. Retinex processing for automatic image enhancement. *Journal of Electronic Imaging*, 13(1):100–110, 2004.

[95] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. on Graphics*, 21(3):267–276, 2002.

[96] E. Reinhard, G. Ward, S. Pattanaik, and P. Debevec. *High dynamic range imaging: Acquisition Display and Image-Based Lighting*. Morgan Kaufmann, 2006.

[97] M.A. Robertson, S. Borman, and R.L. Stevenson. Dynamic range improvement through multiple exposures. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 3, pages 159–163, 1999.

[98] H. Samet. Applications of spatial data structures. Addison-Wesley, 1990.

[99] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or how do I organize my holiday snaps?. *Computer Vision ECCV 2002*, pages 414–431, 2002.

[100] C. Schnörr. Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class. *Int. Journal of Computer Vision*, 6(1):25–38, 1991.

[101] H. Seetzen, W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, and A. Vorozcovs. High dynamic range display systems. *ACM Transactions on Graphics (TOG)*, 23(3):760–768, 2004.

[102] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. of the 9th Int. Conference on Computer Vision*, pages 750–757, 2003.

[103] J. Shi and C. Tomasi. Good features to track. In *Proc. of the Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[104] S.M. Smith and J.M. Brady. SUSAN – a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[105] S. S. Stevens. The surprising simplicity of sensory metrics. *American Psychologist*, 17(1):29–39, 1962.

[106] Technical Committee 3.1. An analytic model for describing the influence of lighting parameters on visual performance, Vol. 1: Technical foundations. *CIE 19/2.1*, 1981.

[107] E.K. Teoh, D.P. Mital, B.W. Lee, and L.K. Wee. Automated visual inspection of surface mount PCBs. In *16th Conference of the Industrial Electronics Society (IECON)*, volume 1, pages 576–580, 1990.

[108] A. Tomaszewska and R. Mantiuk. Image registration for multi-exposure high dynamic range image acquisition. In *Proc. of the WSCG*, pages 978–80, 2007.

[109] A. Troccoli, S.B. Kang, and S. Seitz. Multi-view multi-exposure stereo. In *3rd Int. Symposium on 3D Data Processing, Visualization, and Transmission*, pages 861–868, 2006.

[110] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of the CCD imaging process. In *Proc. of the ICCV*, pages 480–487, 2001.

[111] J. Tumblin and H. Rushmeier. Tone reproduction for realistic images. *Computer Graphics and Applications*, 13(6):42–48, 1993.

[112] J. Unger, S. Gustavson, M. Ollila, and M. Johannesson. A real time light probe. In *Proceedings of the 25th Eurographics Annual Conference*, pages 17–21, 2004.

[113] M. Uyttendaele, A. Eden, and R. Skeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *Proc. of the Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2001.

[114] L. Van Gool, T. Moons, and D. Ungureanu. Affine/photometric invariants for planar intensity patterns. *Computer Vision ECCV'96*, pages 642–651, 1996.

[115] H. Wang, R. Raskar, and N. Ahuja. High dynamic range video using split aperture camera. In *Proc. of the IEEE 6th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, 2005.

[116] T.H. Wang, W.S. Wong, F.C. Chen, and C.T. Chiu. Design and implementation of a real-time global tone mapping processor for high dynamic range video. In *Proc. of the IEEE International Conference on Image Processing (ICIP)*, pages 209–212, 2007.

[117] G. Ward. A contrast-based scalefactor for luminance display. *Graphics Gems IV*, pages 415–421, 1994.

[118] G. Ward. The RADIANCE lighting simulation and rendering system. In *Proc. of the 21st Conference on Computer Graphics and Interactive Techniques*, pages 459–472, 1994.

[119] G. Ward. A wide field, high dynamic range, stereographic viewer. *Journal of Vision*, 2:2, 2002.

[120] G. Ward. Fast, robust image registration for compositing high dynamic range photographs from hand-held exposures. *Journal of Graphics Tools: JGT*, 8(2):17–30, 2003.

[121] G. Ward, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4), 1997.

[122] G. Ward and R. Shakespeare. Rendering with radiance: The art and science of lighting simulation. *Morgan Kaufman*, 1998.

[123] A.M. Waxman and K. Wohn. Contour evolution, neighborhood deformation, and global image flow: Planar surfaces in motion. *The International Journal of Robotics Research*, 4(3):95, 1985.

[124] M. Wichtlhuber. Real-Time Generation of HDR Videos Using GPUs. Master's thesis, Praktische Informatik IV, Prof. Dr. W. Effelsberg, University of Mannheim, Germany, 2010.

[125] B. Wilburn, N. Joshi, V. Vaish, E.V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM Trans. on Graphics*, 24(3):765–776, 2005.

[126] O. Yadid-Pecht and E.R. Fossum. Wide intrascene dynamic range CMOS APS using dual sampling. *IEEE Trans. on Electron Devices*, 44(10):1721–1723, 1997.

[127] S.H. Yang and K.R. Cho. High dynamic range CMOS image sensor with conditional reset. In *Proc. of the IEEE Custom Integrated Circuits Conference*, pages 265–268, 2002.

[128] A. Yoshida, V. Blanz, K. Myszkowski, and H.P. Seidel. Perceptual evaluation of tone mapping operators with real-world scenes. In *Human Vision and Electronic Imaging X, IS&T/SPIE's 17th Annual Symposium on Electronic Imaging*, volume 5666, pages 192–203, 2005.