University of Alberta

REAL-TIME PROCESSING FOR LOGARITHMIC CMOS IMAGE SENSORS

by

Jing Li

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

in

Digital Signal & Image Processing

Department of Electrical and Computer Engineering

Spring 2012
Edmonton, Alberta

Abstract

This thesis proposes a real-time Digital Signal Processing (DSP) design for logarithmic CMOS image sensors. The design contains novel Fixed Pattern Noise (FPN) correction and tone mapping methods suitable for fixed-point operation. Logarithmic CMOS image sensors offer high Dynamic Range (DR) at video rate but suffer from nonlinear FPN. FPN, due to parameter variation across pixels, results in lower image quality. A new method based on the Taylor series is introduced to correct nonlinear FPN effectively and efficiently. After FPN correction of a high-DR scene, reproducing it for display is challenging. Subjective DR needs to be communicated to human observers while objective DR must be compressed to suit the DR of a standard display. A new method maps tones of high-DR scenes for standard displays while limiting the visibility of camera noise. The new FPN correction and tone mapping methods both exhibit low computational complexity, which make them ideal for real-time processing. A fixed-point design of the proposed DSP is developed to further reduce computational complexity, enabling lower power consumption. Although experiments were done with a standard logarithmic CMOS image sensor, the proposed methods may be applied to other nonlinear image sensors thanks to their inherent generality.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Dileepan Joseph. He is the first professor I met after I came to the University of Alberta, and he introduced me to this pretty interesting research area: electronic imaging. During the past two years, he taught and guided me in how to do research, helped me to improve my writing and presentation skills by editing my thesis and slides repeatedly, and inspired me to overcome difficulties and gain confidence. Without his help and supervision, it would have been impossible for me to finish my thesis on time with good quality.

I want to thank my parents especially. They encouraged me to go to Canada, a wonderful and beautiful country, to receive a high-quality education. Otherwise, this memorable journey would have been impossible. With their care and support, I was able to focus on my study and research.

Also, I thank my aunt and her family in Calgary. They helped me when I first applied, and made me feel warm and welcome when I just arrived. And I thank my girlfriend, who read as a master's student in parallel. Although our majors were totally different, our discussions always inspired in me some new ideas. Moreover, she made my life in Edmonton enjoyable and colourful, instead of boring and lonely!

Furthermore, my friends in the Electronic Imaging Lab deserve great gratitude. Dr. Orit Skorka provided me a camera prototype she designed, and taught me how to use it. The camera prototype, with my modifications, was used for all tests in the thesis. Dr. Kamal Ranaweera provided a Visual C++ framework, which I modified with the assistance of Adam Harrison, who helped me learn object-oriented programming. Additionally, both Orit and Adam read parts of my thesis and gave me valuable feedback. Cindy Wong, who was always glad to help me, gave me a lot of assistance with LaTeX typesetting. And Dr. Alireza Mahmoodi, who is now working at PMC Sierra, guided me when I started Altera FPGA design. Discussions with him also helped me review fundamentals of digital circuit design.

I extend special thanks to all my friends in the department, in Edmonton, and in China. They helped me a lot when I confronted technical and personal difficulties. All of them helped and encouraged me to keep learning and to live happily!

I also thank the Natural Sciences and Engineering Research Council and the Graduate Students' Association for financial support. Additionally, I thank IMRIS, especially Dr. Mark Alexiuk of IMRIS, for in-kind contributions to my research.

Finally, I acknowledge the contributions made by Jamie Hon, Lane Mitchelmore, and Jesse Chen. Although they finished before I started, they worked on undergraduate research projects with my supervisor that influenced our work.

Table of Contents

List of Figures

List of Acronyms

ADC Analog-to-Digital Converter

APS Active Pixel Sensor

ASIC Application-Specific Integrated Circuit

CCD Charge-Coupled Device

CDF Cumulative Distribution Function

CMOS Complementary Metal-Oxide-Semiconductor

DOF Degrees of Freedom

DPS Digital Pixel Sensor

DR Dynamic Range

DSP Digital Signal Processing

FPGA Field-Programmable Gate Array

FPN Fixed Pattern Noise

GLS General Least Squares

GPU Graphics Processing Unit

HVS Human Visual System

IPR Inverse Polynomial Regression

LPF Low-Pass Filter

LUT Look-Up Table

OLS Ordinary Least Squares

PCB Printed Circuit Board

PDF Probability Density Function

PPS Passive Pixel Sensor

PR Polynomial Regression

PSNR Peak SNR

PSNDR Peak SNDR

RMS Root Mean Square

SNR Signal-to-Noise Ratio

SNDR Signal-to-Noise-and-Distortion Ratio

VI Vertically-Integrated

Chapter 1

Introduction

Solid-state image sensors are widely used to capture visual information for different goals, including scientific research, medical diagnosis, and consumer use. The development of such image sensors has experienced two generations, which comprise Charge-Coupled Device (CCD) and Complementary Metal-Oxide-Semiconductor (CMOS) Active Pixel Sensor (APS) technology [1]. There are many different designs for CMOS image sensors. Logarithmic CMOS image sensors have a great advantage in Dynamic Range (DR) [2]. They are able to capture over six decades of luminance in one frame. For typical linear CCD or CMOS image sensors, the DR can only span three decades. Moreover, logarithmic response is a natural way to achieve high DR because encoding on a logarithmic scale is similar to the human perception model [3]. Because of its superiority in DR, logarithmic CMOS image sensors are an attractive alternative to replace conventional linear image sensors to overcome the low DR limitation.

Unfortunately, logarithmic image sensors have been unable to compete with linear image sensors in terms of image quality. The main drawback of logarithmic image sensors is low Signal-to-Noise-and-Distortion Ratio (SNDR) because of low Signal-to-Noise Ratio (SNR) and nonlinear Fixed Pattern Noise (FPN) that has been difficult to correct efficiently. Integration of an Analog-to-Digital Converter (ADC) with each pixel is an approach to improve the SNR [4]. Nonlinear FPN can be corrected by subsequent DSP [5, 6, 7, 8]. Existing FPN correction algorithms trade off between performance and complexity. The algorithm should be able to correct FPN effectively. Yet, its computational efficiency is also very important.

Images and videos are often observed by humans. Logarithmic CMOS image sensors are able to capture high-DR scenes. But capturing is only the first step of the electronic imaging process. How to display high-DR scenes for human observers is the second part. Emerging high-DR display equipment, with DR closer to possible scene DR, is a direct solution [9]. However high-DR displays are unlikely to become widespread in the near future for multiple reasons, including price and power consumption. An algorithm-based approach is another solution, which has attracted a large amount of research interest. The process of converting scene luminance to display luminance is known as tone mapping [10]. The DR of standard displays spans only two decades [11]. Tone mapping algorithms compress the world DR of high-DR scenes to suit the display DR but also try

to keep perceived DR. Existing tone mapping algorithms have limited applicability to logarithmic image sensors, especially because there may be significant noise and distortion in the images and videos in comparison to linear image sensors.

DSP is a key component for image sensors. It can improve the quality of images and videos both during capture and display. This thesis proposes a complete DSP solution for logarithmic image sensors, including FPN correction, tone mapping, and corresponding fixed-point design. A background on logarithmic CMOS image sensors is given in Section 1.1. Section 1.2 reviews the state of the art on real-time processing for logarithmic pixels. Finally, the scope of the thesis is described in Section 1.3.

1.1 Logarithmic CMOS Image Sensors

Since 1970, CCD technology dominated the market of image sensors for three decades. During that period, CMOS image sensors could not compete with CCD image sensors because CCD had high SNDR and small pixels [12]. However, the disadvantages of CCDs, including high power consumption and the need for a specialized fabrication process, restricted their applications [13]. With technology developments, CMOS was able to rival CCD technology and even take over market share by overcoming the limitations of CCDs.

The advantages of CMOS image sensors are several. Lower power consumption makes CMOS image sensors suitable for portable imaging devices such as digital cameras in mobile phones [12]. Standard fabrication technologies enable high integration and low price [14]. With CMOS sensors, multiple signal processing blocks can be integrated readily, including ADCs and amplifiers [4]. Using these blocks, the function of a CMOS image sensor comes closer to that of a human eye, which can process light information before relaying it to the brain [3]. CCD and CMOS APS technologies represent first and second-generation image sensors, respectively. A third generation, based on Vertically-Integrated (VI) CMOS technology, is emerging [1]. Thanks to high compatibility, both the second and third-generation technologies benefit from the scope of this thesis. For logarithmic CMOS image sensors, low SNDR is still the main drawback when compared to linear image sensors. High-DR displays are another challenge. Existing methods for SNDR improvement and high-DR display are reviewed in this section.

Based on their response model, image sensors can be divided into three categories: linear, logarithmic, and others [2]. Compared with linear and other image sensors, logarithmic image sensors are able to provide high DR at video rates but they suffer from low SNDR. However, no matter the category, the performance of all image sensors still has difficulty competing with the human eye, although electronic imaging technologies have had more than four decades of development [15]. Compared to the human eye, a deficiency of modern image sensors is a trade off between SNDR and DR. In Fig. 1.1, the SNDR and/or SNR versus DR of different image sensors are compared to the same for the human eye. Commercial CCD and CMOS image sensors are mainly linear image sensors. Many rival the human eye in SNDR while DR is worse. Academic logarithmic CMOS image sensors have high DR, which is close to the human eye; SNDR is the main drawback. Therefore, it is easy to understand there are two basic approaches to make

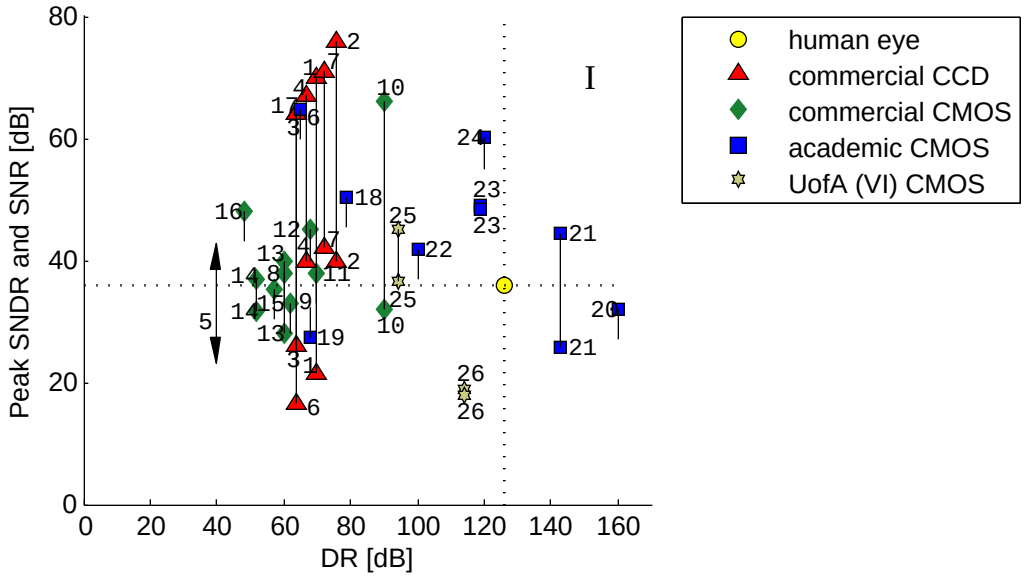

Figure 1.1: Linear sensors offer high SNR or SNDR while logarithmic sensors offer high DR. Sensors 25 and 26 are logarithmic CMOS DPS and logarithmic VI-CMOS APS prototypes, respectively, that were designed and tested at the University of Alberta. Logarithmic response offers high DR while pixel-level ADCs and nonlinear FPN correction provide high SNDR. The figure is provided by Skorka [1], incorporating data by Mahmoodi [16] and methods developed in this thesis.

image sensors with both high SNDR and high DR. One is to expand the DR of linear image sensors to achieve high DR. On the other hand, improving the SNDR of logarithmic image sensors is another feasible approach. Research in the Electronic Imaging Lab at the University of Alberta focuses on the latter approach.

There are several important steps before captured light signals achieve final display intensities. They include data conversion, FPN correction, and tone mapping. Data conversion, from an analog signal to a digital signal, in image sensors is an important process. With CMOS technology, data conversion can be achieved on-chip at three levels, which are chip-level, column-level, or pixel-level. In data conversion methods, Nyquist-rate ADCs and oversampling ADCs [17] are possible. Both the conversion level and method can determine performance, including SNDR and frame rate, of the image sensor. Low SNDR in logarithmic CMOS image sensors is caused by low SNR and high FPN. To deal with low SNR, pixel-level ADCs are a promising solution [4]. Instead of employing an ADC for the whole chip or one for each column, an ADC is integrated inside each pixel. Besides high SNR, such Digital Pixel Sensor (DPS) technology provides other advantages, such as low analog power consumption because of the low sampling frequency in the pixel. In addition, due to parallel conversion, high-speed digital readout is possible. At the University of Alberta, Mahmoodi [16] successfully designed and fabricated a logarithmic CMOS DPS prototype. The tests showed a Peak SNDR (PSNDR) of at least $36\,\mathrm{dB}$, which rivals the SNDR of the human eye. This result, along with a Peak SNR (PSNR) of at least $45\,\mathrm{dB}$, proved that DPS technology based on oversampling ADCs is able to

improve the SNR and SNDR of logarithmic CMOS image sensors significantly.

Although pixel-level ADCs improve the SNR of the logarithmic CMOS image sensors, SNDR must also be improved for the image quality to be improved. SNDR equals SNR only when FPN is perfectly corrected. Previously, researchers developed various analog and digital methods to reduce high FPN in logarithmic pixels. Although analog methods exhibit low delay [18, 19, 20], their performance is unsatisfactory with logarithmic pixels due to complex FPN. Compared to analog methods, digital methods are more accurate and flexible. Unfortunately, good performance methods, which can improve SNDR to approximate SNR, have required high computational complexity. Unless high power consumption is acceptable, this complexity makes the delay in FPN correction too long. Although many works about digital correction methods have been developed [5, 6, 7, 21], the restriction between performance and complexity still exists. Another main drawback of existing methods is they are tied to a specific response model. The model-specific methods may confront problems because different logarithmic pixel designs are possible. Moreover, the model of the standard design has evolved for accuracy reasons [21]. An ideal FPN correction method should have good and stable performance to make the SNDR close to the SNR over a high DR. At the same time, the computation complexity should be feasible for low-power real-time processing. Finally, the generality for different response models is a significant factor too.

After data conversion and FPN correction, digital images and videos of high quality are captured. Then, they are often reproduced on display equipment or print paper for human consumption. The conversion process from scene luminance to display luminance is called tone mapping [10]. Logarithmic CMOS image sensors are able to capture high-DR scenes. Yet, the DR of both standard display equipment and printed paper is much lower than what is possible in real scenes. This limitation makes high-DR tone mapping a challenging problem. Various methods have been developed from diverse ideas. The intuitive method is DR extension of the display equipment. If the DR of display equipment is close to possible scene DR, then display of high DR scenes is easier. Projector-based and LED-based high-DR displays have been reported [9]. Other high-DR display systems are in development [22, 23]. Although performance is good, factors such as price and power consumption prevent the widespread use of high-DR displays in the near future. On the other hand, software-based tone mapping methods are widely used. Tone mapping algorithms are mainly divided into global and local operators based on mapping functions. Global operators employ spatially-invariant mapping functions. In contrast, mapping functions may vary spatially for local operators. Popular operators have been reported in both categories [10, 24, 25]. Generally, local operators may have superior performance but suffer from high-computational complexity. Global operators can provide good performance for most cases and enjoy simplicity. For image sensors operating at video rates, global operators are preferred because they are relatively easy to implement in real time. Previous tone mapping methods always assume that images and videos are free of noise. However, this assumption is unsuitable for logarithmic image sensors. Therefore, although good methods have been reported in the literature, a new method needs to be developed for logarithmic image sensors.

Thanks to CMOS technology, logarithmic image sensors have advantages, such as high integration, low power consumption, and low cost. Additionally, they are able to offer high DR at video rates. They are a promising alternative to overcome the problem of low DR with linear image sensors. However, they suffer from low SNDR because of low SNR and high FPN. SNR can be improved through pixel-level ADC while digital correction can reduce FPN. Then, a tone mapping algorithm maps the luminance in a captured high-DR scene to display intensity. DSP performance directly affects the competitiveness of logarithmic CMOS image sensors with linear image sensors.

1.2 Real-Time Processing

DSP can be performed on different platforms, such as a desktop computer, a digital signal processor, an Field-Programmable Gate Array (FPGA), or an Application-Specific Integrated Circuit (ASIC). In any platform, DSP computation costs a period of time to get results, which introduces a delay between the original signal input and processed signal output. DSP applications can be divided into two types based on the constraint of delay, including real-time processing and non-real-time processing [26]. In this thesis, non-real-time processing is called offline processing. Offline DSP focuses on digital signals stored beforehand. With offline processing, what matters is the correctness and accuracy of the computation. The computational duration is not a significant factor because there is no strict constraint on the delay between input and output. On the other hand, stringent time demands exist with real-time processing. The DSP component must complete processing tasks within a certain duration, with a required accuracy, because the input signal will not wait. Therefore, real-time DSP entails high demands on the design of DSP algorithms and hardware. The computational complexity of a DSP algorithm should be feasible for real-time processing in the chosen hardware. Although the hardware where real-time processing is achieved should be able to provide high-computational speed and abundant-computational resources [26], hardware performance comes at a cost, which includes power consumption during operation. Besides functional performance, low power consumption is an important factor. It is especially appreciated for portable devices such as digital cameras. Logarithmic image sensors need real-time processing when they work at video rate. Existing designs are reviewed and limitations are analyzed below.

Logarithmic CMOS image sensors can provide high DR at video rate. Instead of still-image cameras, high-DR video cameras have more extensive applications in different fields. Machine vision can employ high-DR video cameras. Applications such as driver assistance may benefit from the high-DR of logarithmic image sensors [8]. Similar applications include safety surveillance [27] and quality control [28]. All of these applications propose stringent requirements on the DSP. Long delays can have serious consequences. Besides functional performance, reasonable power consumption is an important factor too because high power consumption may bring inconvenience. Real-time processing is also very important to the consumer market. For a high-DR video camera based on a logarithmic image sensor, it is difficult to imagine consumers accepting a long delay, say if fluid output is not visible during video capture. Moreover, short battery life is very

annoying. Therefore, for wider application, efficient real-time processing is necessary for logarithmic image sensors in video cameras.

Although research has been done on correcting the FPN of logarithmic image sensors, it has focused on offline processing. The trade off between performance and complexity has not been overcome. Some methods, such as Otim's correction [21] and Joseph's one or two-parameter correction [5], can be implemented in real time [7]. However, the correction performance is not good enough. High performance correction, like Joseph's three-parameter correction [5], suffers from high computational complexity, which impedes real-time implementation. Schneider reported a FPN correction method that was implemented in an FPGA evaluation board based on fixed-point operation [8]. In that method, piecewise linear functions instead of a nonlinear model made the real-time process straightforward. Yet, the simplified model resulted in degraded performance in the dark and at the piecewise knots. An ideal real-time FPN correction method should not sacrifice performance for efficiency and vice versa. In addition, high compatibility to evolving models is preferred.

With the developments of high-DR rendering and photography technologies, the issue of high-DR reproduction has attracted a large amount of interest in academia and industry. Some high-DR displays have been developed and marketed but only in a high-end market [9, 22, 23]. Before high-DR displays replace standard displays in the market, software-based methods for tone mapping will continue to be important. However, the conditions for real-time tone mapping are similar to those for real-time FPN correction. Onerous computation and a complicated structure would make real-time implementation of tone mapping difficult. At present, real-time tone mapping is achieved by implementing existing methods in a Graphics Processing Unit (GPU), an FPGA, or an ASIC. These platforms can process signals at high frame rate to make video fluid. Goodnight *et al.* realized "photographic tone reproduction" in a GPU [29]. Yet, the size and power consumption make the GPU approach infeasible for portable imaging devices. Hassan *et al.* [30] presented a FPGA-based architecture for the same algorithm. Wang *et al.* [31] designed and tested an ASIC implementing part of the same method. Moreover, Wang *et al.* also proposed an ASIC design for another tone mapping algorithm [32]. They reported a fluid frame rate for high resolution videos. However, all of these works are not developed for logarithmic image sensors. In contrast, real-time tone mapping for logarithmic image sensors is nearly non-existent [11]. Besides good DR-compression performance and real-time feasibility, residual noise and distortion in captured images and videos needs to be considered. Therefore, a specific method needs to be developed for better performance.

Real-time DSP raises strict demands on the algorithm design and hardware implementation. The computational complexity of the DSP algorithm must be feasible, while the required accuracy is ensured. Low computational complexity can reduce the power consumption, which is very significant for portable devices. Existing logarithmic FPN corrections mainly focus on offline processing. The methods which can be achieved in real time do not have good enough correction performance. On the other hand, real-time tone mapping is nearly non-existent for logarithmic image sensors, although a few general tone mapping algorithms were implemented in a GPU, FPGA, or ASIC to meet real-time

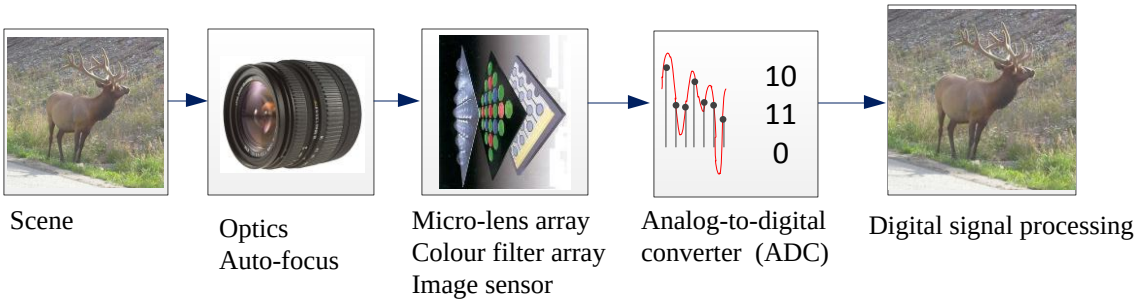

Figure 1.2: The components of a digital camera comprise mainly a lens, a color filter, an image sensor, one or more ADCs, and DSP. The ADCs in CMOS digital camera may be at chip level, column level, or pixel level. The work in this thesis focuses on the DSP part. (This figure was created by Orit Skorka and modified by Jing Li.)

demands [29, 30, 31]. Logarithmic image sensors cannot work at video rates with high quality without real-time processing. For more extensive applications, real-time DSP with high performance is needed specifically for logarithmic image sensors.

1.3 Scope of the Thesis

In Fig. 1.2, the main imaging components of digital cameras are shown. This thesis, which focuses on the final component of Fig. 1.2, provides a completed real-time processing design for logarithmic CMOS image sensors. The design improves the image quality in real time. First, a novel algorithm for FPN correction is introduced. Tests prove that the proposed method can overcome existing limitations. After high DR scenes are captured, videos need to be rendered for standard DR display equipment for human consumption. A new tone-mapping algorithm is designed to compress the DR while preserving the perceived DR and restricting noise magnification. FPN correction and tone mapping methods are both developed first for floating-point operation. To facilitate initial prototyping, a high-level programming language, namely Matlab, is used. However, floating-point operation and Matlab are not ideal decisions for a real-time system due to their computational complexity. High complexity implies high power consumption. Therefore, a corresponding fixed-point design implemented in C is provided in the thesis. It brings low complexity, which is especially important for portable devices where power consumption is critical. The rest of this section gives brief descriptions of following chapters. A prototype digital camera, which is used in experiments of this thesis, is introduced too.

Chapter 2 presents the novel FPN correction method. The new method is developed based on a Taylor series expansion and polynomial regression. It overcomes a limitation between correction performance and computational complexity. Response linearization, which is included, renders metric scene stimulus from corrected digital response using spline interpolation. No circuit response models are needed, which means the proposed method is not tied to a specific model. Such flexibility is a consequence of using numerical

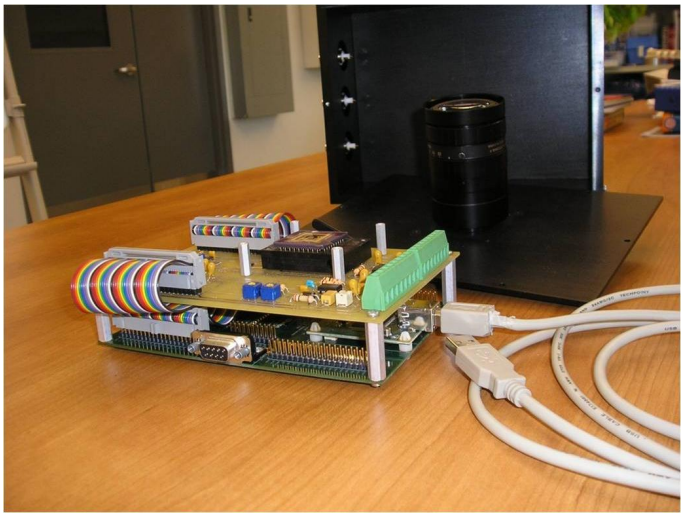

Figure 1.3: Prototype of a digital camera with a logarithmic CMOS image sensor. The PCB board and QuickUSB board are connected by a ribbon cable. The boards are put inside the camera body. (This picture was taken by Orit Skorka.)

methods.

The details of noiseless tone mapping are introduced in Chapter 3. The goal of this tone mapping algorithm is the real-time reproduction of high-DR scenes for standard DR display equipment with high fidelity. It prevents the camera noise after tone mapping based on histogram equalization from exceeding a visibility threshold. The adaptation process of the human eye is also considered. A Low-Pass Filter (LPF) is applied to frame information, which approximates the natural adaptation process. Abrupt DR changes are avoided in video output.

Chapter 4 describes a fixed-point design of the novel FPN correction and noiseless tone mapping methods. For FPN correction, the correction coefficients are first scaled. Then, a correction is computed through pipelined fixed-point operation. Fixed-point error is modeled and analyzed. For noiseless tone mapping, Look-Up Table (LUT)s are an ideal method for fixed-point implementation. Instead of staying constant, the LUT that stores the mapping function updates with each frame based on fixed-point operations. A program is coded to compute the performance of the proposed methods versus wordlengths. Then, an optimal design is determined considering both performances and wordlengths. Besides offering low complexity, the fixed-point design also ensures the proposed methods will have low power consumption with a planned FPGA implementation.

The proposed methods are suitable for a variety of nonlinear image sensors, including various logarithmic [33] and linear-logarithmic sensors [7, 20]. All the tests presented in this thesis were performed with a CMOS APS logarithmic image sensor, the schematic for which is well known [5]. This sensor was designed by Orit Skorka, in her PhD research. It was fabricated in a $0.35\,\mu\mathrm{m}$ TSMC process through CMC Microsystems [33]. The array tested includes 90×120 pixels, having a $10\,\mu\mathrm{m}$ pitch. A prototype digital camera, also designed by Orit Skorka, was fabricated to test the image sensor. It includes a camera body, a

lens (Fujinon CF25HA-1), a custom-made Printed Circuit Board (PCB), and a QuickUSB board. A photo of the disassembled prototype is shown in Fig. 1.3. The custom-made PCB accommodates the image sensor, a 16-bit ADC (Texas Instruments ADS8411), and other off-the-shelf electronic components that are needed for power supply, biasing, and digital communication. The QuickUSB board includes an Altera Cyclone II FPGA that operates at $48\,\mathrm{MHz}$ [34]. Address and control signals are generated by the FPGA for the image sensor and ADC. The FPGA reads data from the ADC and sends it to a PC, which has an Intel Pentium D $2.80\,\mathrm{GHz}$ CPU and $2\,\mathrm{GB}$ of DDR2 memory. The PC processes the captured data in real time. Kamal Ranaweera and Adam Harrison, a postdoctoral fellow and a PhD student, developed the Visual C++ framework for camera setup, data readout, and video display. Then, the proposed DSP algorithms, programmed in C, were embedded in the framework to form the completed real-time system.

Finally, Chapter 5 summarizes the novelty and significance of contributions in this thesis. Some relevant future work is discussed, including FPN correction considering the factor of temperature, the performance of the proposed methods with a DPS array, and noiseless tone mapping implementation in an FPGA. The potential application of the presented methods to the invisible band is also mentioned.

Chapter 2

Fixed Pattern Noise Correction

Fixed pattern noise (FPN) is caused by parameter differences going from one pixel to another in an image sensor. Both linear pixels and nonlinear pixels are susceptible to FPN. Compared with CCD image sensors, CMOS image sensors suffer from more serious FPN. The readout buffers and amplifiers are different for each pixel in a CMOS image sensor, so it causes relatively high FPN [3]. For FPN in linear pixels, many relevant papers have been published and some excellent methods have been proposed [35]. Similarly, a large amount of work has been done for FPN in nonlinear pixels. Some researchers prefer using analog techniques [18, 19, 20], which have an important advantage, namely speed. The correction part is in the pixel, and the corrected value is output with little delay. However, most analog FPN correction is only able to correct offset FPN. FPN in nonlinear pixels is much more complicated than offset FPN. For example, the response model of a logarithmic pixel keeps getting more accurate and more complex; it is now a four-parameter model [21]. So the correction performance of analog methods are far from good.

Digital processing of acquired images can reduce the FPN in nonlinear pixels. Compared with analog techniques, the digital method is able to correct FPN more accurately. Meanwhile, it is more flexible since it is usually achieved in an embedded system. Joseph and Collins [5] used images of multiple uniform scenes to calibrate parameters of logarithmic pixels. In three-parameter correction, response linearization was included with the FPN correction. This method has restrictions between correction performance and calibration complexity while it is specific for the logarithmic response model. Schneider [6] developed a method to approximate a logarithmic response with three different lines in different domains. Although this method is realizable in a real-time system, approximating a nonlinear model with a piecewise linear function is problematic. Also, the performance of this method strongly depends on the specific model and its accuracy. For a combined linear-logarithmic CMOS image sensor, Storm *et al.* [7] used a reference current for two-parameter logarithmic calibration. Otim *et al.* [21] developed a four-parameter model for the logarithmic response, which is more accurate than the three-parameter model of Joseph and Collins [5]. Nevertheless, Otim *et al.* only used a two-parameter method for FPN correction, where the correction method approximates the nonlinear model with a linear model.

Compared with FPN correction, response linearization is a relatively new problem since only nonlinear-response pixels need it. Hoefflinger and Schneider mentioned the response linearization problem and proposed a method [8]. Their algorithm is specific to the logarithmic response model of Joseph and Collins [5]. Moreover, it is closely tied to the associated FPN correction method. As mentioned before, the model keeps changing and the pixel configuration may change too. For example, this method will not work for the four-parameter model. A similar method for a new model is difficult to develop.

Existing methods can reduce FPN in nonlinear pixels but they always have restrictions in correction performance, calculation complexity, and generality. They try to simplify the nonlinear model to a linear model, or to approximate the nonlinear model with a linear model. Few of them try to correct FPN directly based on a nonlinear model. In addition, little research about response linearization has been done.

This chapter describes a new method for nonlinear FPN correction and response linearization. It is based on a Taylor series, Polynomial Regression (PR), and spline interpolation. The existing restrictions, including low performance, high computation complexity, and low generality, are overcome. Developing a new method for calibration and correction mainly includes two steps. The first step is based on PR, which is general for different monotonic nonlinear models, and easy for low-order correction but difficult or impossible for high-order correction. The second step is switching to Inverse Polynomial Regression (IPR), which is general for different monotonic nonlinear models too. More importantly, the IPR method only needs arithmetic operations for correction of any order. For response linearization, no model analysis is needed, thanks to a numerical method.

2.1 Polynomial Regression

The method of Joseph and Collins [5] for calibration and correction is divided into three kinds based on the number of model parameters that vary spatially. The one (offset) and two (offset and gain) parameter methods only need low-complexity calculations. However, the performance of these FPN corrections is not good enough. The three (offset, gain, and bias) parameter method has good performance on FPN correction but calibration needs iteration. Also, the non-arithmetic three-parameter correction causes difficulty when implementing it in hardware to meet a real-time demand.

Based on previous work in an image of N pixels, the actual response y_{ij} of the jth pixel to stimulus x_i is

$$y_{ij} = f_j(x_i) + \epsilon_{ij}. \tag{2.1}$$

Let the difference between the *actual* response y_{ij} and the *estimated* response $f_j(x_i)$ be the residual error ϵ_{ij}. This residual error is assumed to be statistically independent for different observations and pixels, and is further assumed to follow a zero-mean Gaussian distribution. Unlike previous work, the response model is expressed here by an abstract function f_j, instead of, for example, a logarithmic model, because the new method is general for different models.

The average response of all pixels in an image sensor to the same luminance is regarded as an ideal response without FPN. FPN correction, as described by Joseph [3], is about obtaining an ideal response for each pixel from its actual response. Calibration is categorized based on the number of parameters in the response model. For one and two-parameter models, the estimated responses $f_j(x_i)$ are linear functions of the average response $\bar{y}_i$ of all pixels to luminance x_i, and only a linear regression is needed even for a nonlinear pixel. Three-parameter calibration needed an iterative approach, which is very complex.

These low-order polynomial models raise the question of whether higher-order polynomials can replace the original nonlinear model for three-parameter calibration, and even for a response model that is nonlinear but not logarithmic. Because averaging filters the residual error, the average response $\bar{y}_i$ of all pixels to a luminance x_i can be expressed as

$$\bar{y}_i = \frac{1}{N}\sum_{j=1}^{N} y_{ij} \approx \frac{1}{N}\sum_{j=1}^{N} f_j(x_i) \equiv F(x_i). \tag{2.2}$$

Assuming that each response function f_j is monotonic to luminance x_i, the average response $\bar{y}_i$ is likewise monotonic. So the inverse function

$$x_i = F^{-1}(\bar{y}_i) \tag{2.3}$$

exists, whereby luminance x_i is a function of average response $\bar{y}_i$. Response y_{ij} in (2.1) is a function of luminance x_i, so the actual response y_{ij} is a function of average response $\bar{y}_i$ as

$$y_{ij} = f_j(F^{-1}(\bar{y}_i)) + \epsilon_{ij}. \tag{2.4}$$

With the Taylor series theorem, the actual response y_{ij} can be approximated through any order polynomial of the average response $\bar{y}_i$. If the order is high enough, the Taylor series error is very small compared to the residual error ϵ_{ij} and can be neglected. For example, the P order polynomial is

$$y_{ij} = a_{j0} + a_{j1}\bar{y}_i + a_{j2}\bar{y}_i^2 + \cdots + a_{jP}\bar{y}_i^P + \epsilon_{ij}. \tag{2.5}$$

Equation (2.5) can be expressed using matrix-vector notation as

$$\begin{bmatrix} y_{1j} \\ y_{2j} \\ \vdots \\ y_{Mj} \end{bmatrix} = \begin{bmatrix} 1 & \bar{y}_1 & \bar{y}_1^2 & \cdots & \bar{y}_1^P \\ 1 & \bar{y}_2 & \bar{y}_2^2 & \cdots & \bar{y}_2^P \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \bar{y}_M & \bar{y}_M^2 & \cdots & \bar{y}_M^P \end{bmatrix} \begin{bmatrix} a_{j0} \\ a_{j1} \\ \vdots \\ a_{jP} \end{bmatrix} + \begin{bmatrix} \epsilon_{1j} \\ \epsilon_{2j} \\ \vdots \\ \epsilon_{Mj} \end{bmatrix}, \tag{2.6}$$

which is equivalent to the simpler equation:

$$\mathbf{y}_j = \mathbf{Y}\mathbf{a}_j + \boldsymbol{\epsilon}_j. \tag{2.7}$$

So this is an Ordinary Least Squares (OLS) problem. It can be solved in Matlab for the best fit parameter vector of each pixel by

$$\hat{\mathbf{a}}_j = \mathbf{Y} \setminus \mathbf{y}_j. \tag{2.8}$$

Calibration has built the relationship between the ideal response (average response) and the actual response. This method for any order calibration is general and Matlab can solve (2.8) easily. Correction will use the relationship to recover an ideal response from an actual response, given an image y_j to be corrected. For P order calibration, the corrected image $\hat{y}_j$ is defined by the roots of

$$y_j = \hat{a}_{j0} + \hat{a}_{j1}\hat{y}_j + \hat{a}_{j2}\hat{y}_j^2 + \cdots + \hat{a}_{jP}\hat{y}_j^P. \tag{2.9}$$

If P = 1, this is a trivial problem, because solving N linear equations is very easy. If P = 2, now the problem becomes solving N quadratic equations. They can be solved using square root calculations, which are not arithmetic operations. If $P \geq 3$, the problem becomes very difficult analytically. There are no analytic solutions for getting the roots of $P \geq 5$ polynomials. Solving $P \geq 3$ polynomials is usually done iteratively, and this must be done for every pixel independently.

This PR *calibration* has very low calculation complexity since $\mathbf{Y}$ in (2.8) is fixed for each pixel. Moreover, the calibration needs to be done only once for each image sensor since the parameters are fixed. However, correction is different because it needs to meet a real-time demand. High-order *correction* with this approach cannot meet a real-time demand. Therefore, the PR method is still an imperfect solution to FPN correction.

2.2 Inverse Polynomial Regression

The restriction of the PR method, which was discussed in Section 2.1, inspires developing a better method for correction. PR correction recovers an ideal response from an actual response using relationship (2.9). Solving this high-order equation is difficult so we return to (2.4). If it is inverted as

$$\bar{y}_i = F(f_j^{-1}(y_{ij} - \epsilon_{ij})) \approx F(f_j^{-1}(y_{ij})) + \epsilon'_{ij}, \tag{2.10}$$

where

$$\epsilon'_{ij} = -\frac{dF(f_j^{-1}(y_{ij}))}{dy_{ij}}\epsilon_{ij}, \tag{2.11}$$

then $\bar{y}_i$ can be expressed as any order polynomial of y_{ij} through a Taylor series. As with (2.5), the P order polynomial is

$$\bar{y}_i = a'_{j0} + a'_{j1}y_{ij} + a'_{j2}y_{ij}^2 + \cdots + a'_{jP}y_{ij}^P + \epsilon'_{ij}. \tag{2.12}$$

Similar to PR, this can be expressed as

$$\begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_M \end{bmatrix} = \begin{bmatrix} 1 & y_{1j} & y_{1j}^2 & \cdots & y_{1j}^P \\ 1 & y_{2j} & y_{2j}^2 & \cdots & y_{2j}^P \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & y_{Mj} & y_{Mj}^2 & \cdots & y_{Mj}^P \end{bmatrix} \begin{bmatrix} a'_{j0} \\ a'_{j1} \\ \vdots \\ a'_{jP} \end{bmatrix} + \begin{bmatrix} \epsilon'_{1j} \\ \epsilon'_{2j} \\ \vdots \\ \epsilon'_{Mj} \end{bmatrix}, \tag{2.13}$$

which is equivalent to the simpler equation:

$$\bar{\mathbf{y}} = \mathbf{Y}_j \mathbf{a}'_j + \boldsymbol{\epsilon}'_j. \tag{2.14}$$

For this relationship, the parameter vector in (2.14) needs to be estimated. In PR, the residual error ϵ_{ij} is assumed to be statistically independent from observation to observation and pixel to pixel, with a zero-mean Gaussian distribution. This assumption has been confirmed as reliable for logarithmic CMOS image sensors [3]. So with PR, parameter estimation by the OLS method is equivalent to maximum likelihood estimation. However, with IPR, the residual error ϵ'_{ij} would depend on observation and pixel index. In theory, General Least Squares (GLS) should be used instead of OLS. Prewhitening will be the first step of GLS. However, determining the statistics of residual error ϵ'_{ij} is not easy, although (2.11) is the starting point. For now, prewhitening is neglected and OLS is used to solve the problem.

Matlab can find the parameter vector $\hat{\mathbf{a}}'_j$ that best fits the data through

$$\hat{\mathbf{a}}'_j = \mathbf{Y}_j \setminus \bar{\mathbf{y}}. \tag{2.15}$$

Compared to PR calibration, IPR calibration will be a little more complex since the matrix $\mathbf{Y}_j$ will be different for each pixel. However, since the calibration only needs to be done once for each image sensor and not in real time, this increase in complexity is insignificant.

Although the IPR calibration is a little more complex, the IPR correction is much simpler. Given an image y_j for P order correction, the corrected image $\hat{y}_j$ can be expressed as

$$\hat{y}_j = \hat{a}'_{j0} + \hat{a}'_{j1} y_j + \hat{a}'_{j2} y_j^2 + \cdots + \hat{a}_{jP} y_j^P. \tag{2.16}$$

This correction only needs arithmetic calculations. However, calculating (2.16) directly is not a good idea. It is numerically inefficient to take integer powers by repeated multiplication. If (2.16) is expressed in another form, i.e.,

$$\hat{y}_j = (\ldots((\hat{a}'_{jP})y_j + \hat{a}'_{j(P-1)})y_j + \hat{a}'_{j(P-2)}\ldots)y_j + \hat{a}'_{j0}, \tag{2.17}$$

the power calculation can be avoided and the calculations can be pipelined in hardware. This method of FPN correction is very suitable for real-time implementation.

2.3 Median Filtering

Although the new FPN correction algorithm is able to correct FPN efficiently, it is unable to correct "dead" pixels. Dead pixels may cause salt-and-pepper noise. In image processing, the median filter is a very popular order-statistic filter. Compared with linear smoothing filters of similar size, median filters can reduce salt-and-pepper noise effectively with less blurring [36]. In our system, we use the cross median filter, whose neighborhood only has three or five pixels. For most pixels, the filter uses the center pixel and four nearest neighbors. For border pixels, the filter uses the center pixel and the two nearest pixels along the same border. For the four corner pixels, the filter uses three pixels too. They are the center pixel and the two nearest pixels.

The cross median filter has two main advantages. Firstly, it has a very small neighborhood. Few calculations ensures high execution efficiency. The second advantage is that an odd-point median filter can be equally placed before or after any pixel-wise and monotonic image processing function. Although the cross median filter is introduced in this section because its function is correction, it is actually implemented after response linearization and simple tone mapping, which are explained next.

2.4 Response Linearization

Response linearization is about rendering the scene stimulus (luminance) from the pixel response, which may be nonlinear. The problems with existing methods for response linearization are similar to the problems with existing FPN calibration and correction methods: they are tied to a specific model and are difficult to calculate. Now we have a new method for response linearization, which is based on spline interpolation. Compared to polynomial interpolation, spline interpolation is preferred due to its ability to avoid round-off error and oscillations. In practice, cubic splines are most frequently used. The reason is that they show desired smoothness while keeping the simplest representation [37]. Similar to the proposed method for FPN calibration and correction, the new method for response linearization is a general method suitable for different response models.

The mission of response linearization is to recover a scene estimate $\hat{x}_j$ from the corrected image $\hat{y}_j$. Considering (2.3), if we know $F^{-1}(y)$ then, given corrected image $\hat{y}_j$, we calculate

$$\hat{x}_j = F^{-1}(\hat{y}_j). \tag{2.18}$$

However, the response model is abstract and a general method is preferred. Instead of circuit analysis to model $F^{-1}(y)$, we develop an empirical model using spline interpolation. It is called *inverse* spline interpolation because we model an inverse function.

In the inverse spline interpolation, the logarithm of luminance, $\ln(x)$, is used instead of the luminance x. The relationship between $\ln(x)$ and response y is approximately linear for a logarithmic or logarithmic-like pixel. However, the relationship between x and y is highly nonlinear. So the response linearization will be from y to $\ln(x)$ and then

to x. Other kinds of nonlinear pixels can use a similar approach, with or without an intermediate mapping to $\ln(x)$.

Assuming x_i is measured during calibration, we have data $(x_i, \bar{y}_i)$, where $1 \leq i \leq M$. The inverse spline model is:

$$\ln(\hat{x}_j) = S(\hat{y}_j), \tag{2.19}$$

where

$$S(y) = \begin{cases} S_1(y), & y \leq \bar{y}_2 \\ S_2(y), & \bar{y}_2 < y \leq \bar{y}_3 \\ \vdots & \vdots \\ S_{M-1}(y), & \bar{y}_{M-1} < y \end{cases}$$

For the linear spline, the expression for $S_i(y)$ is very simple since there are only two parameters. The equation is

$$S_i(y) = b_{i0} + b_{i1}(y - \bar{y}_i). \tag{2.20}$$

For the cubic spline, which is a cubic polynomial, the equation is

$$S_i(y) = b_{i0} + b_{i1}(y - \bar{y}_i) + b_{i2}(y - \bar{y}_i)^2 + b_{i3}(y - \bar{y}_i)^3. \tag{2.21}$$

There are four parameters in (2.21) so the calculation complexity is higher than in (2.20). As with (2.16) and (2.17), it is better to rewrite (2.21) as:

$$S_i(y) = b_{i0} + (y - \bar{y}_i)(b_{i1} + (y - \bar{y}_i)(b_{i2} + b_{i3}(y - \bar{y}_i))). \tag{2.22}$$

With a linear spline, the function after interpolation will always be monotonic. This is desirable since F^{-1} is expected to be monotonic. Coefficients for parameters b_{ik} of the linear spline are calculated in the standard manner for linear spline interpolation [37]. However, cubic spline interpolation will not ensure that the function is monotonic even if the data points are monotonic. Therefore, cubic Hermite spline interpolation is actually employed [38]. As with FPN calibration, the coefficients b_{ik} are calculated once, and are then used to calculate (2.19) repeatedly.

After the inverse spline interpolation, we have $\ln(\hat{x}_j)$. To complete the linearization, exponentiation is required, i.e.,

$$\hat{x}_j = \exp(S(\hat{y}_j)). \tag{2.23}$$

2.5 Simple Tone Mapping

The advantage of nonlinear pixels, such as logarithmic pixels, is mainly to capture a high DR image. Tone mapping, discussed further in Chapter 3, is required to represent captured scenes appropriately on a display device, such as a monitor. A simple tone-mapping algorithm, based on the IEC sRGB standard [39] is presented here. In an image, we need

to define a white point x_{white}. Usually, the white point is the highest pixel stimulus. However, in a high DR image, assigning the white point as the highest pixel stimulus may cause serious underexposure in parts that are relatively darker. So the white point x_{white} will be kept as a parameter and not be automatically computed.

Equations (2.24)–(2.26) give the details of this simple tone mapping. After defining the white point, each pixel stimulus is normalized as

$$\hat{x}'_j = \frac{\hat{x}_j}{x_{\text{white}}}, \tag{2.24}$$

where $\hat{x}_j$ is the estimated stimulus, as in (2.23). After we get a normalized stimulus, a normalized gray value is calculated according to the sRGB standard:

$$w_j = \begin{cases} 0, & \hat{x}'_j \leq 0 \\ 12.92 \cdot \hat{x}'_j, & 0 < \hat{x}'_j \leq 0.00304 \\ 1.055 \cdot \hat{x}_j'^{1/2.4} - 0.055, & 0.00304 < \hat{x}'_j < 1 \\ 1, & \hat{x}'_j \geq 1 \end{cases} \tag{2.25}$$

Because the white point x_{white} may not be the maximum value of pixel stimulus, the normalized gray value w_j is assigned 1 when the normalized stimulus $\hat{x}'_j$ is greater than 1. Finally, we compute an 8-bit gray value:

$$W_j = \text{round}(255 \cdot w_j). \tag{2.26}$$

Our proposed methods, including FPN correction and response linearization, are general for various nonlinear pixels. To use it with a logarithmic image sensor, the response linearization needs exponentiation, as in (2.23). However, because we have $\ln(\hat{x}_j)$ after the *inverse* spline interpolation, a small modification can make the performance better. Exponentiation will not be used during response linearization. It will be calculated during simple tone mapping. Equation (2.24) can be transfered to logarithmic form, i.e.,

$$\ln(\hat{x}'_j) = \ln(\hat{x}_j) - \ln(x_{\text{white}}). \tag{2.27}$$

Therefore, (2.25) is also rewritten:

$$w_j = \begin{cases} \exp(\ln(12.92) + \ln(\hat{x}'_j)), & \ln(\hat{x}'_j) \leq \ln(0.00304) \\ \exp(\ln(1.055) + \frac{1}{2.4}\ln(\hat{x}'_j)) - 0.055, & \ln(0.00304) < \ln(\hat{x}'_j) < 0 \\ 1, & \ln(\hat{x}'_j) \geq 0 \end{cases} \tag{2.28}$$

Equation (2.26) will not change. This form avoids calculating unnecessary logarithms and exponents (power calculations) in the simple tone mapping. It reduces the computation complexity and improves the accuracy.

2.6 Look-Up Table Implementation

The computational complexity of simple tone mapping is low. However, the piecewise functions in both response linearization and simple tone mapping need selection operators that adversely affect computational efficiency. The simple tone mapping is a pixel-wise operator. Moreover, the interpolated function is fixed for an image sensor while the

mapping function stays constant. So employing a LUT is an ideal alternative to direct computation.

The simple tone mapping follows the response linearization. Because we modified the sRGB tone mapping to avoid unnecessary calculations, only response normalization is needed. Two LUTs are adopted for achieving the required two steps. The whole process is:

$$\hat{y}_j \rightarrow \mathrm{LUT}_1 \rightarrow \ln(\hat{x}_j) \rightarrow \ln(\hat{x}'_j) \rightarrow \mathrm{LUT}_2 \rightarrow W_j. \tag{2.29}$$

First, the calculation of (2.19) is achieved by LUT_1. Next, the estimated stimulus in logarithmic scale $\ln(\hat{x}_j)$ is normalized by the white point $\ln(\hat{x}_{\text{white}})$, which only needs subtraction. The result is the input to LUT_2, which maps the normalized response $\ln(\hat{x}'_j)$ to the display intensity W_j, a composition of (2.26) and (2.28). Because the spline interpolant and mapping function are fixed for each image sensor, the two LUTs can be built once offline. No updates are needed during real-time processing, even with a tunable white point.

2.7 Results

Experiments were done using a digital camera prototype with a logarithmic CMOS image sensor, which has been described in Section 1.3. The proposed algorithms were first programmed in Matlab. Then they were programmed in C, which is embedded in a Visual C++ framework. Programming in Matlab helps to analyze the results of FPN correction and response linearization quantitatively. For real-time applications, computational complexity is a key factor. Languages with high efficiency, such as C or C++, are more suitable. Therefore, the experiments are divided in two parts. Matlab experiments focus on offline performance. Real-time performance is tested in Visual C++. The experiment methods and results are discussed below.

2.7.1 Matlab Experiments

In the experiment, we took the sun as the light source. A sheet of white paper was illuminated by the sun. Fifteen effective luminances were achieved by changing the aperture of the camera. At each luminance, ten uniform images were captured for calibration. The tests in Matlab mainly focused on quantitative performance through offline analysis.

In Matlab, the ten uniform images at each luminance were averaged to reduce the temporal noise. The calibration program used the average image at each luminance as the ideal response to calculate the correction coefficients. Both PR and IPR calibration and correction were done. Fig. 2.1 compares the performance of three-parameter PR and IPR correction for three pixels. The corrected response of each individual pixel approximates the ideal response very well. Both methods have nearly the same performance, which is difficult to distinguish by inspection.

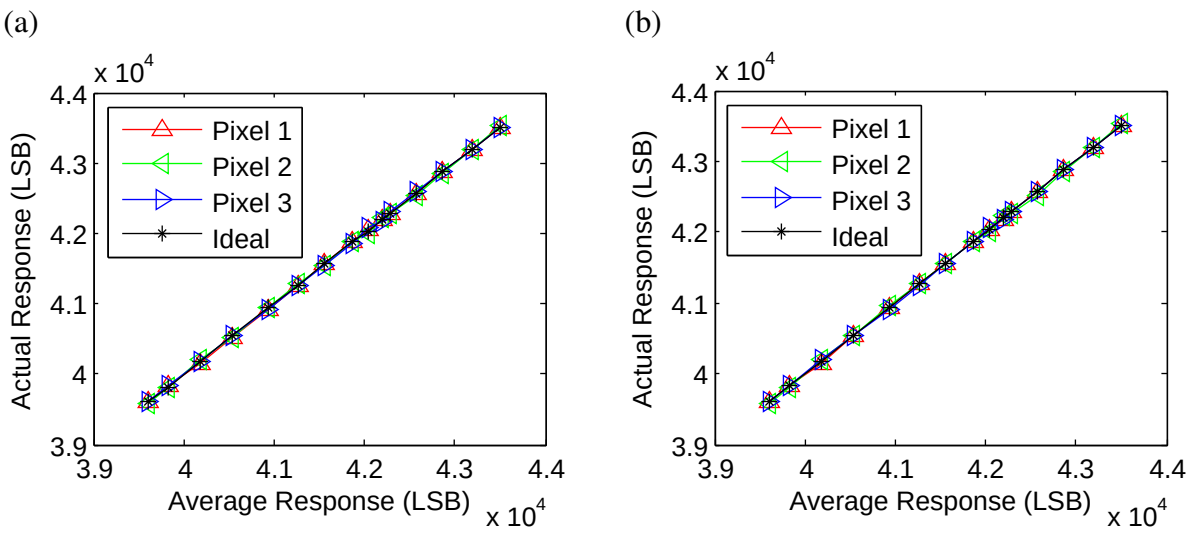

Figure 2.1: Three-parameter (a) PR and (b) IPR FPN correction of three pixels. The actual (corrected) responses of both methods approximate the average (ideal) response of all $10\,800$ pixels very well in a dynamic range of 5.3 to $33\,000\,\mathrm{cd/m^2}$.

Individual responses of corrected pixels determine imaging performance. For overall analysis, Root Mean Square (RMS) residual error is a suitable parameter for evaluation [5]. After FPN correction, each pixel has a different residual error. The residual error for a specific pixel varies with luminance. In general, FPN correction may be represented by a function $C(y_{ij})$. Given that $\bar{y}_i$ is the average response of all pixels, the residual error ϵ'_{ij} for pixel j at luminance level i is:

$$\epsilon'_{ij} = \bar{y}_i - C(y_{ij}). \tag{2.30}$$

Given a total pixel number of N, and that calibration and correction were done at M different luminances, the RMS residual error $\sigma_{\epsilon'}$ is defined as follows:

$$\sigma^2_{\epsilon'} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(\bar{y}_i - C(y_{ij}))^2}{DOF}, \tag{2.31}$$

where the denominator is the Degrees of Freedom (DOF) [40]. For the PR and IPR methods, one can show:

$$DOF = MN - M - PN, \tag{2.32}$$

where P is number of parameters per pixel. For any luminance level i, the RMS residual error $\sigma_{\epsilon'_i}$ is defined similarly:

$$\sigma^2_{\epsilon'_i} = \frac{M\sum_{j=1}^{N}(\bar{y}_i - C(y_{ij}))^2}{DOF}. \tag{2.33}$$

Fig. 2.2 compares the RMS residual error after one, two, and three-parameter IPR FPN correction, as well as Otim *et al.*'s FPN correction [21]. Three-parameter IPR correction

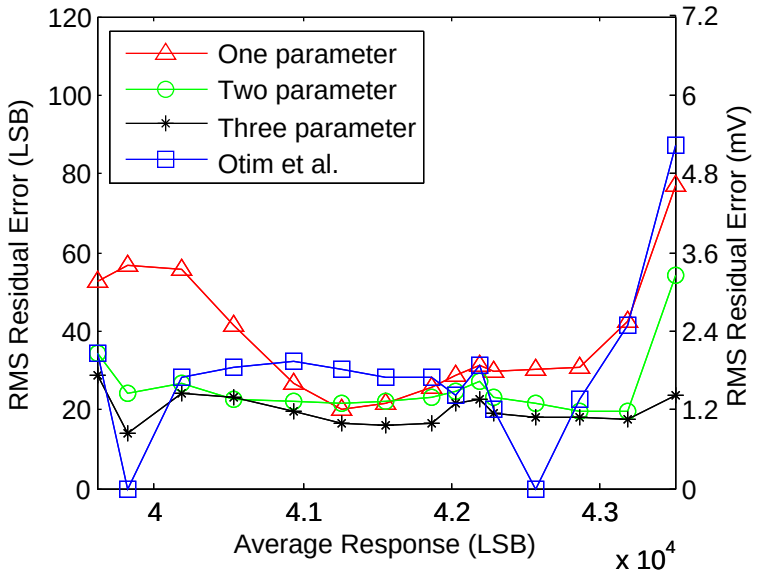

Figure 2.2: RMS residual error of various FPN correction methods, including one from Otim *et al.*'s FPN correction [21]. Three-parameter IPR correction provides the best performance among these corrections, considering all luminance levels.

offers better and more stable performance than one or two-parameter IPR correction. Although RMS residual error after Otim *et al.*'s correction equals zero at two luminance levels, the performance is less stable, and worse at all other luminance levels, than the proposed three-parameter IPR correction. Further analysis showed that four-parameter correction offered no further improvement.

The result of response linearization is demonstrated in Fig. 2.3. As expected, the interpolated curve approximates a straight line because scene luminance is plotted on a logarithmic scale. Fig. 2.4 compares images captured by our digital camera prototype before and after three-parameter IPR FPN correction. Simple tone mapping is applied to all images before display.

In summary, offline performance has been tested in Matlab. Quantitative and qualitative analysis proved the good performance of the proposed methods.

2.7.2 C++ Experiments

The testing with C++ focuses on real-time performance. The three-parameter IPR correction and simple tone mapping were programmed in C++. For simple tone mapping, an LUT is employed for computational efficiency. When the proposed methods were executed in C++, the system frame rate reaches $50\,\mathrm{Hz}$, which makes the video fluid. Fig. 2.5 exhibits two series of frames before and after IPR correction. The IPR correction reduced the FPN in each frame effectively. The image quality was improved significantly.

Real-time performance of the proposed method was proved with a C++ implementation. The method is ideal for real-time processing because only arithmetic operations are needed for correction. The restriction between complexity and performance for existing

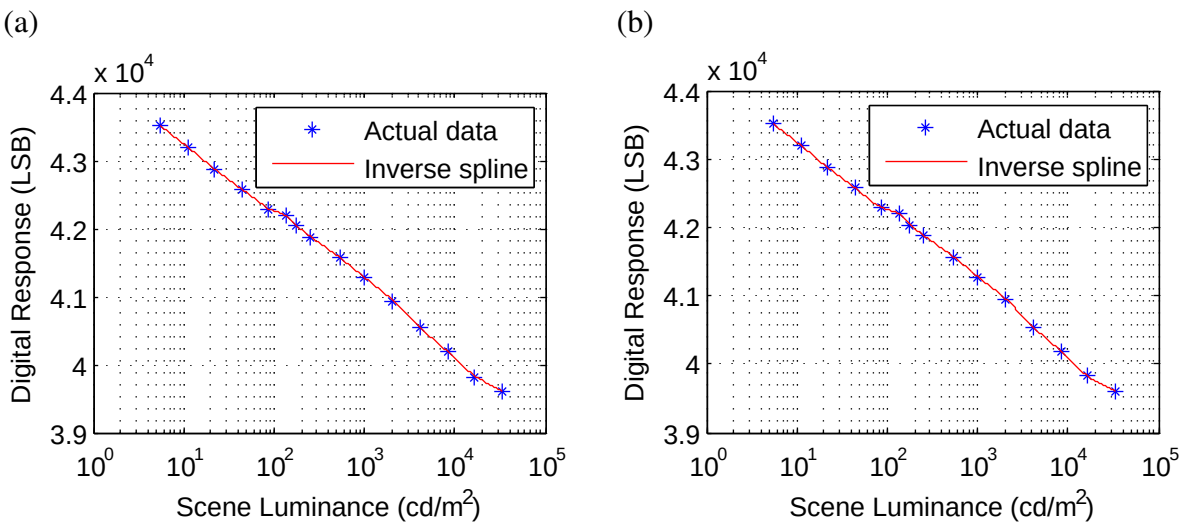

Figure 2.3: Response linearization based on (a) linear interpolation and (b) piecewise cubic Hermite spline interpolation using the calibration data. The difference between them is not obvious because of the log-scale X axis.

FPN correction methods has been overcome.

2.8 Conclusion

This chapter introduced a new FPN correction and response linearization method for nonlinear-response CMOS pixels. Most existing methods for nonlinear FPN correction try to simplify the nonlinear model to a linear model or approximate it through a piecewise linear function. They have to trade off complexity with performance. Also, little work has been done on response linearization. Moreover, because a concrete expression for the response model was previously necessary, another problem of existing FPN correction and response linearization methods is they are tied to the given model.

The new FPN correction has two important advantages compared to existing methods. The first one is high generality. Although tests were only done with a standard logarithmic CMOS APS array, the proposed method did not require a specific response model. Another advantage is low computational complexity during correction, which only needs arithmetic operations. This ensures that it can correct in real time while keeping high performance. At the same time, a new response linearization algorithm has been developed. Similar to the new FPN correction, response linearization is general for different response models but retains simplicity. By incorporating the sRGB display standard [39] into the framework, we developed a simple tone mapping.

Experiments were done using a digital camera prototype having a logarithmic CMOS image sensor. The RMS residual error after the new FPN correction showed superior performance over a wide dynamic range. Real-time implementation demonstrated system performance at a frame rate sufficient for fluid video. It proved that the new algorithm

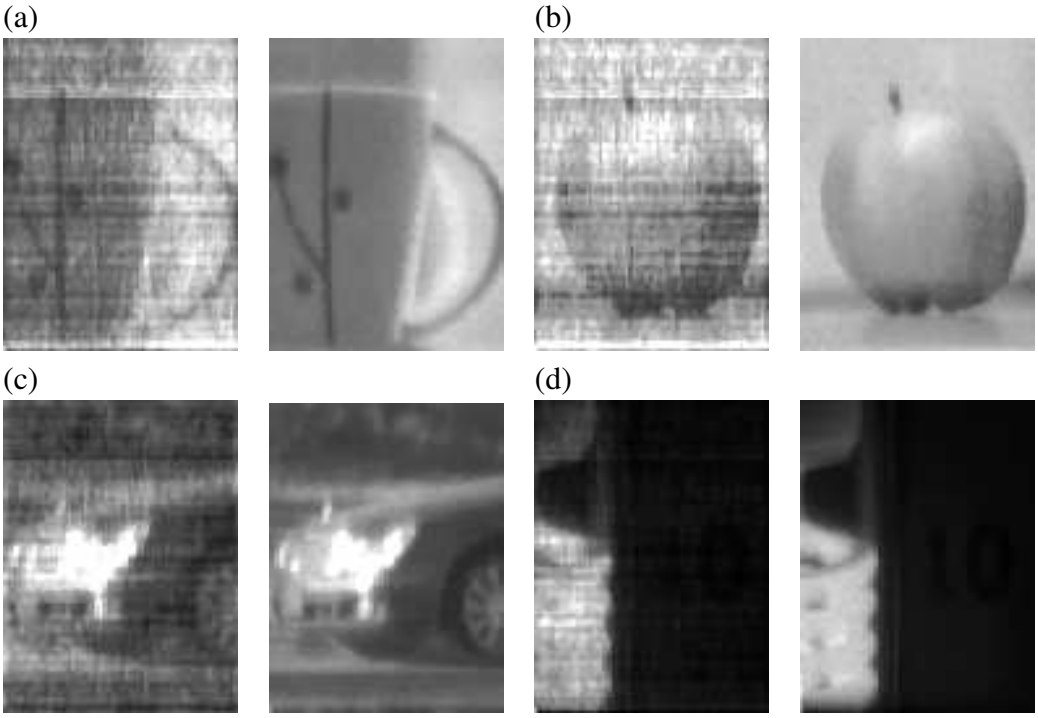

Figure 2.4: Original image (left) and image after three-parameter IPR FPN correction (right) for four scenes. The proposed correction reduces FPN effectively, which improves the image quality. Simple tone mapping makes it difficult to render high-DR scenes for a standard display. In (c), the headlight of the car is overexposed. In (d), the dark part of the scene is underexposed. Nevertheless, FPN is corrected.

provides high image quality in real time. Through experiments, the new FPN correction has overcome the existing limitation between performance and complexity. The proposed method is ideal for real-time and high-performance digital FPN correction.

The only issue with the efficiency of the new method is that it has been developed based on floating-point operations in a PC. Compared to fixed-point operation, floating-point operation has higher precision but lower computational efficiency. Such higher computation complexity would make it difficult to realize the method at high speed with low-power consumption. A fixed-point design will be introduced in Chapter 4, after tone mapping is improved in Chapter 3.

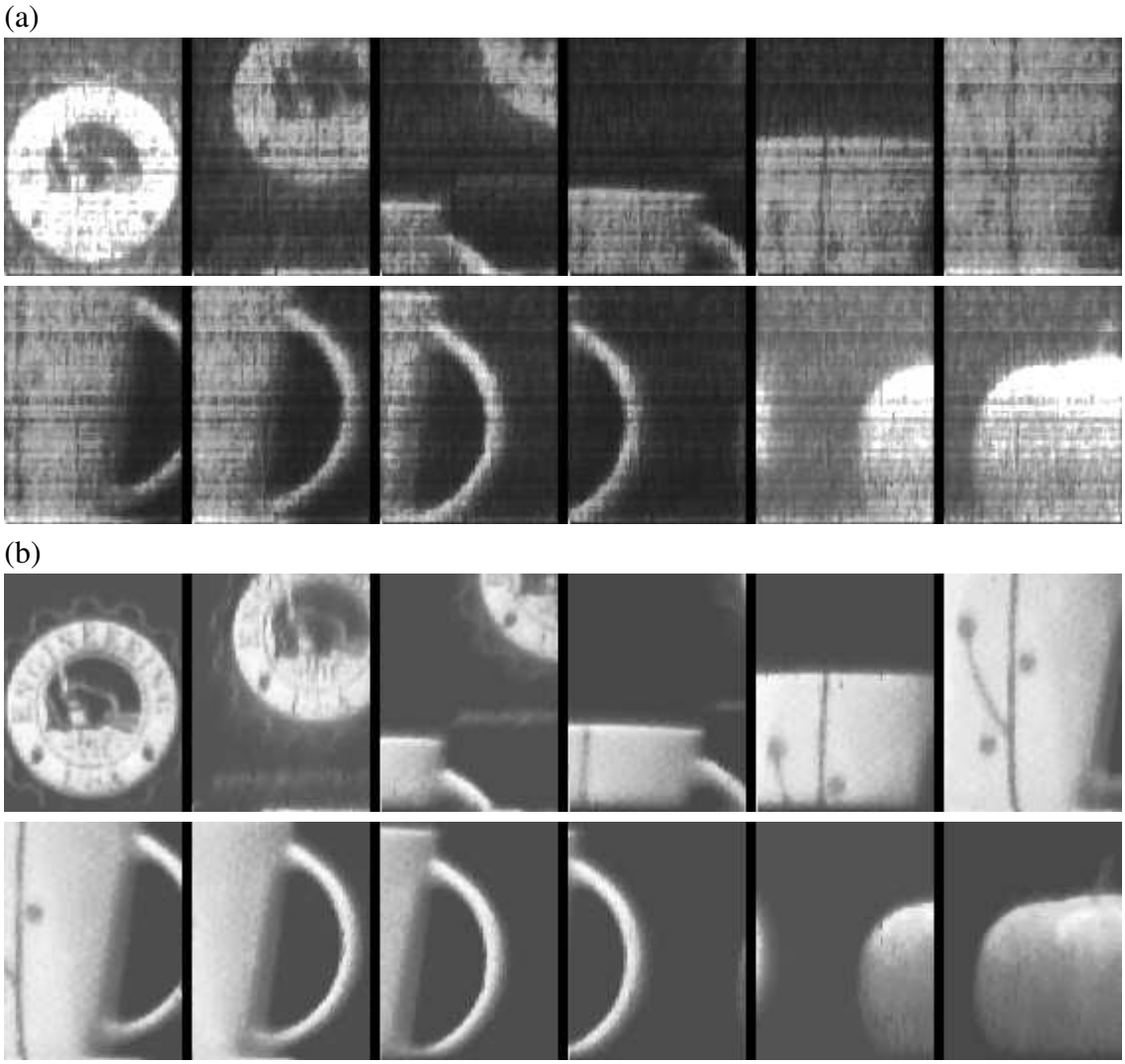

Figure 2.5: Video of a scene (a) before and (b) after three-parameter IPR FPN correction. These examples demonstrate the suitability of the method for real-time processing. Although the frame interval is $20\,\mathrm{ms}$, every tenth frame is shown to highlight dynamics.

Chapter 3

Noiseless Tone Mapping

High-DR image sensors are capable of capturing high-DR scenes in one exposure. For research purposes, captured signals, including images and videos, may be recorded on a metric scale for computer analysis. In most cases, images and videos are simply displayed for human analysis. Tone mapping is a necessary process in the latter case for mapping scene stimulus to display brightness [10]. High-DR tone mapping is challenging because the DR of standard display equipment is just above two decades [11]. This may be much lower than real-world DR for common scenes. Reproducing high-DR scenes with high fidelity has therefore attracted an increasing amount of attention. Both hardware and software-based methods are widely investigated.

Hardware-based methods are a direct solution. If the DR of display equipment is about the same as possible world DR, the high-DR scene can be displayed easily. Brightside Technologies (formerly Sunnybrook Technologies) developed projector-based and LED-based high-DR displays [9]. They report a projector-based display that is able to give a DR of 65000:1. The performance of LED-based high-DR displays is even better. They are able to display high-DR images and videos linearly. Other high-DR display systems are being developed using different ideas [22, 23]. However, multiple reasons such as price and power consumption still make it difficult for high-DR displays to replace standard displays in the near future. Compared to hardware-based methods, software-based methods will continue to be important.

The human eye is a subjective organ rather than an objective one. It is sensitive to relative brightness not absolute luminance. Tone-mapping algorithms are based on this feature. Many different tone mapping techniques have been reported in the literature. Existing techniques mainly include two categories: local operators and global operators. Local operators are developed based on the human eye's response to local contrast. Tones at different pixel locations are mapped through different local functions. In most approaches, high-DR images are decomposed, DR is compressed, and components are recombined. Durand and Dorsey [41] modified bilateral filtering [42] to form fast bilateral filtering. It decomposes the image into base and detail layers. DR compression is only applied to the base layer. Other approaches [43, 44] use similar ideas. Gradient domain DR compression [24] is another popular approach. The concept of this approach is relatively simple, and it is able to preserve local contrasts while bringing few visible artifacts. Re-

cently, a new approach that combines global and local operators has been developed [45]. It does not need scale decomposition, layer separation, or image segmentation. Experiments showed the algorithm has good off-line performance but it is time consuming. High computation complexity makes local operators infeasible or very expensive for real-time processing.

Unlike local tone-mapping techniques, global operators map each pixel's response through a global function. Research on global operators [46, 47, 48] is older than local operators. Among global tone-mapping techniques, the most impressive approach is the algorithm that Larson developed [10] based on histogram equalization and the Human Visual System (HVS). Much research has been done to modify the approach. Instead of local operators, we return to global operators for efficiency.

Most tone-mapping works focus on still images. Few of them try to process video and fewer still try to process it in real time. For video tone mapping, image tone mapping operated on each frame is not ideal. Coria and Nasiopoulos [49] processed video using temporal correlation. Image sequences are divided into groups. In each group, a block-matching motion estimation is applied to frames. Lee and Kim [50] combined gradient-domain tone mapping and motion information to develop a new approach to video processing. Hoefflinger reported [8] that Durand and Dorsey modeled the temporal adaptation process of the human eye as an exponential decay function [51]. This is a simple adaptation model but produces reasonable results [29]. For real-time tone mapping, the frame rate should be high enough to make delay tolerable and video fluid. This problem comes back to a trade off between complexity and performance.

As mentioned previously, nonlinear image sensors are able to capture high-DR scenes in one exposure. High-DR scenes cannot be displayed appropriately without a tone-mapping algorithm. Although tone mapping is not a new topic, real-time tone mapping for nonlinear high-DR image sensors is relatively unexplored [11]. In this chapter, we propose a new method developed from histogram equalization. The inspiration comes from camera and display noise. Existing tone mapping methods assume that images and videos are pure without noise. Unfortunately, this assumption does not hold for high-DR images captured by nonlinear image sensors. Not only does the proposed method compress the DR, it also prevents noise exceeding the tolerance of standard displays. In addition, a human-eye adaptation model is incorporated to handle abrupt DR changes across video frames.

3.1 Histogram Equalization

The DR of standard display equipment only spans two decades, which may be much smaller than the DR of real-world scenes. Simple tone mapping for a high-DR scene may cause underexposure of the dark part, overexposure of the bright part, or a combination thereof, no matter how the white point is set. In this chapter, we use a LogLuv-encoded TIFF image from Larson [52] to introduce our method. Scene luminance can be calculated from LogLuv-encoded TIFF images [53]. Fig. 3.1 presents the image, a bathroom illuminated by a lamp, after simple tone mapping.

(a) (b)

Figure 3.1: Bathroom image, taken from Larson [52], demonstrating (a) underexposure of parts and (b) overexposure of parts. Simple tone mapping cannot show details in dark and bright parts of the bathroom simultaneously.

Tone-mapping methods compress the DR of high-DR scenes to fit within the display DR, while retaining the subjective feeling of the original. Histogram equalization is a commonly used method in image processing, which redistributes pixel intensity to get a more uniform distribution. Because nearly all captured scenes do not have a uniform histogram, histogram equalization shrinks the sparsely-populated portions of the histogram to enable DR compression. In addition, the DR of pixels in densely-populated portions of the histogram is expanded.

Fig. 3.2(a) shows the histogram of the original bathroom image. For histogram equalization, the global tone-mapping function is:

$$Y = (Y_{\max} - Y_{\min})\, P_x(X), \tag{3.1}$$

$$P_x(X) = P(x < X). \tag{3.2}$$

Here, X is the input value (scene luminance) and $P_x(X)$ is the Cumulative Distribution Function (CDF) of the input value. Correspondingly, Y is the output value (display intensity). $Y_{\max}$ and $Y_{\min}$ define the range of Y. According to probability theory, the Probability Density Function (PDF) is related to the CDF as follows:

$$p_x(X) = P'_x(X). \tag{3.3}$$

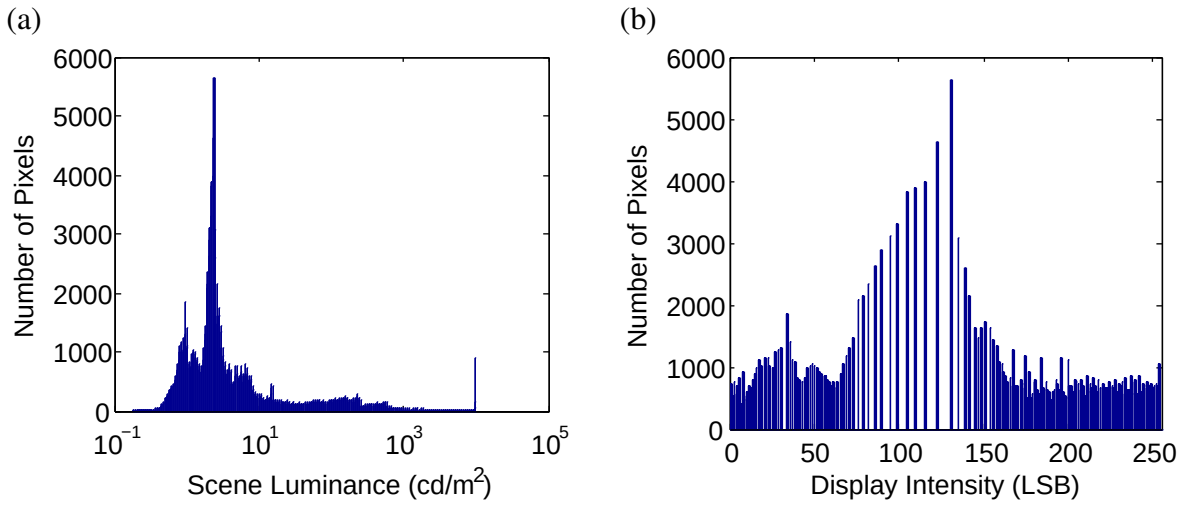

Figure 3.2: Histogram of the bathroom image. (a) The original histogram is not uniform. (b) After histogram equalization, pixel intensities are distributed more uniformly. Gaps in the new histogram occur because equalization is done on discrete data.

With images, the PDF may be approximated with a histogram:

$$p_x(X) \approx \frac{h(X)}{N\Delta x}, \tag{3.4}$$

where $h(X)$ is the histogram of X, N is the number of pixels in the image, and Δx is the step size for the histogram bin.

Intuitively, tone mapping should operate on luminance x because tone mapping exists to transform scene luminance to display intensity. On contrast, the human eye perceives brightness more on a logarithmic scale [3]. This inspires us to use $\ln x$ as a measure of brightness, as with Larson *et al.* [10]. Since e^x is monotonic, (3.2) may be rewritten as follows:

$$\begin{aligned} P_x(X) &= P(e^{\ln x} < e^{\ln X}) && (3.5)\\ &= P(\ln x < \ln X) && (3.6)\\ &= P_{\ln x}(\ln X). && (3.7) \end{aligned}$$

So tone mapping operating on luminance or brightness will bring the same results. In the proposed method, we take brightness as the input. With the high-DR image sensors descried in Chapter 2, this is convenient because the initial output is in the form $\ln X$.

Finally, histogram equalization can be divided into two steps. The first step is building a tone-mapping function based on the histogram of the brightness image. The next step is transforming brightness using the mapping function. Histogram equalization removes empty portions in the histogram to achieve effective DR compression. The DR is expanded in highly-populated regions of the histogram. Fig. 3.2(b) shows the bathroom histogram after histogram equalization, which results in a more uniform distribution while also mapping scene luminance to display intensity.

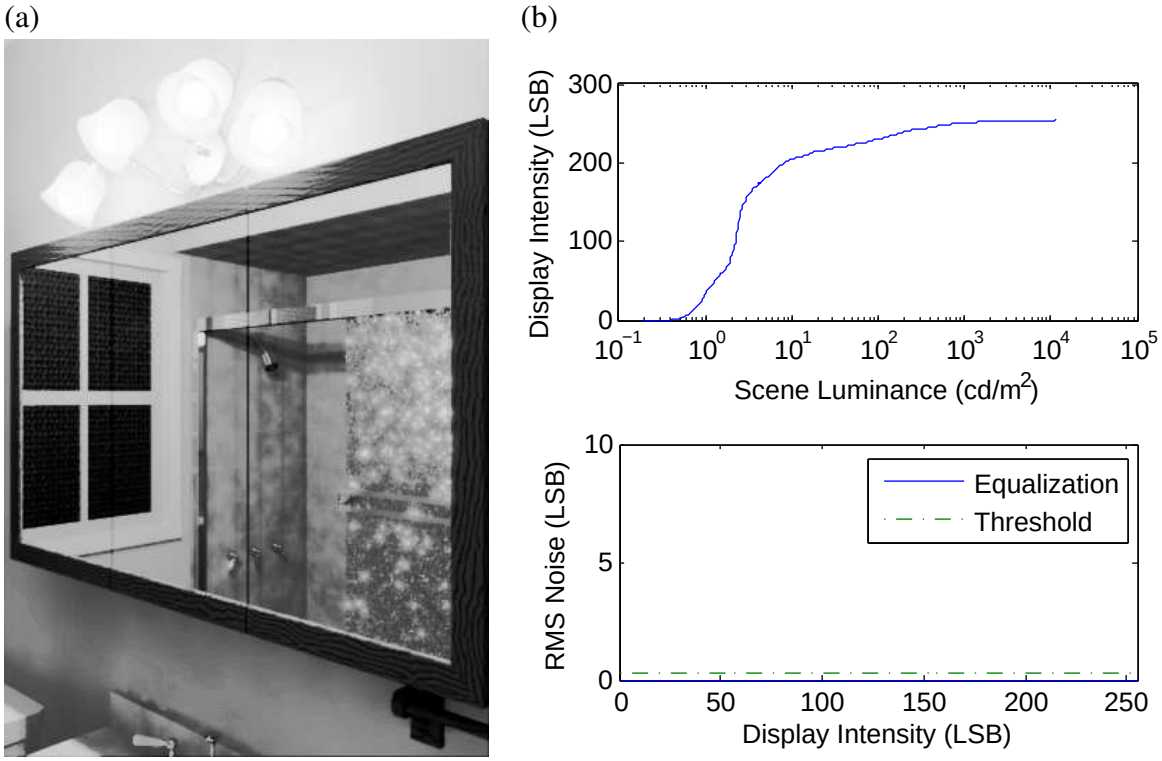

Figure 3.3: (a) Bathroom image after histogram equalization. (b) The mapping function (top) and displayed RMS noise (bottom) due to the camera, which is zero for this simulation.

3.2 Noise Ceilings

Our goal is to develop a tone-mapping algorithm for high-DR image sensors, which is able to process captured video in real time. Video is composed of a sequence of frames. A tone mapping algorithm for single frames may be applied in sequence to video data.

Fig. 3.3(a) shows the bathroom image after histogram equalization, as well as the corresponding mapping function and the RMS noise versus display intensity. Because camera noise was not included in the simulated image, RMS noise is zero at all display intensities after tone mapping. The computational complexity of histogram equalization is low, which makes it possible to process captured video in real time.

Unfortunately, images and videos captured by high-DR image sensors may be noisy. We simulated noise based on a human eye model [15], where RMS noise is a function of brightness. The bathroom image is mixed with this simulated noise. Fig. 3.4(a) shows the noisy bathroom image after histogram equalization. The noise is visible.

After histogram equalization, not only is the DR of highly populated intensities expanded, the noise in the corresponding pixels is magnified too. The worst case is the magnified noise exceeds the tolerance of the display format, becoming visible. Naive histogram equalization may corrupt the quality of images and videos, and therefore should be modified. We have said that histogram equalization includes building a tone-mapping

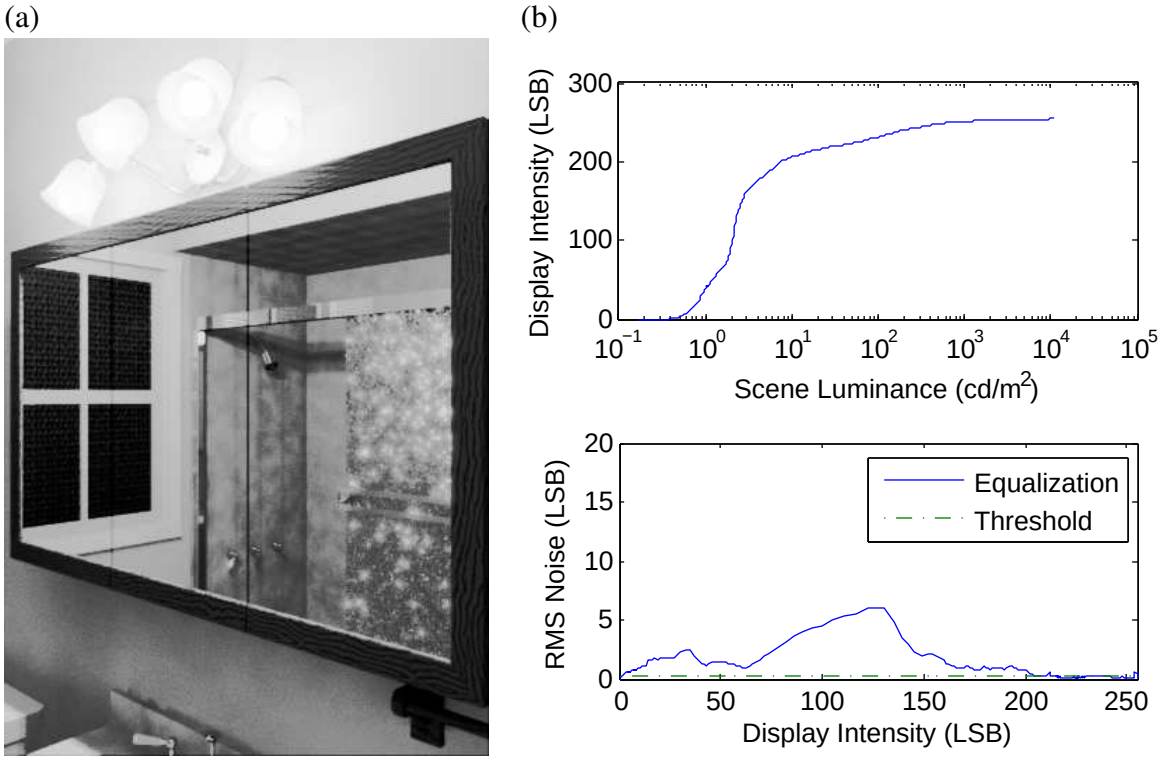

Figure 3.4: (a) Bathroom image with simulated camera noise after histogram equalization. (b) The mapping function (top) and displayed RMS noise (bottom). Camera noise is visible on the wall below the mirror.

function based on the histogram of the input image and applying the mapping function to produce the output image. So we should modify the mapping function to prevent noise magnification beyond a visibility threshold.

Chapter 2 described high-DR image sensors, which are modelled here as follows:

$$\ell_j = \ln x_j + \epsilon_j^{\mathrm{c}}, \tag{3.8}$$

where j indexes pixels, ℓ_j is the measured brightness, x_j is the scene luminance, and ϵ_j^c is the camera noise, which follows a zero-mean Gaussian distribution. Although RMS noise may depend on index as well as brightness, we assume dependence on brightness only. For the image sensors of Chapter 2, median RMS noise versus brightness, denoted $\sigma_{\mathrm{c}}(\ln x)$, can be computed from the calibration data. Therefore, $\sigma_{\mathrm{c}}(\ln x)$ is utilized as the RMS noise for all pixels in the image sensor.

A global tone mapping operator T maps measured brightness ℓ_j to display intensity W_j as follows:

$$W_j = \mathrm{round}(T(\ell_j)) \tag{3.9}$$

$$= T(\ell_j) + \epsilon_j^{\mathrm{d}}. \tag{3.10}$$

In (3.10), rounding is modelled by the addition of display noise ϵ_j^{d}. For sRGB displays [39], there are 256 gray levels. Display noise can be modeled with a uniform distri-

bution from -0.5 to $0.5\,\mathrm{LSB}$, so the RMS display noise in each pixel is:

$$\sigma_{\mathrm{d}} = \frac{1}{\sqrt{12}}. \tag{3.11}$$

This display noise is independent of gray level.

Through (3.8), the signal captured by a camera is mixed with camera noise ϵ_j^{c}. Therefore, a "tone-mapping noise" ϵ_j^{t} is introduced after tone mapping, which originates from the camera noise ϵ_j^{c}. So (3.9) can be written as:

$$W_j = T(\ln x_j) + \epsilon_j^{\mathrm{t}} + \epsilon_j^{\mathrm{d}}. \tag{3.12}$$

Meanwhile, we use a first-order Taylor series to approximate the tone mapping noise ϵ_j^{t} in terms of the camera noise ϵ_j^{c}. Therefore, we can get:

$$W_j \approx T(\ln x_j) + T'(\ln x_j)\epsilon_j^{\mathrm{c}} + \epsilon_j^{\mathrm{d}}. \tag{3.13}$$

Assuming that camera and display noise are uncorrelated, the total RMS noise that is displayed, versus brightness, is as follows:

$$\sigma_W(\ln x) = \sqrt{\sigma_{\mathrm{t}}^2 + \sigma_{\mathrm{d}}^2} \tag{3.14}$$

$$\approx \sqrt{(T'(\ln x)\sigma_{\mathrm{c}}(\ln x))^2 + \sigma_{\mathrm{d}}^2} \tag{3.15}$$

Therefore, under any brightness, if the RMS noise of tone mapping σ_{t} is smaller than or equal to the display RMS noise σ_{d}, i.e.,

$$\sigma_{\mathrm{t}} \leq \sigma_{\mathrm{d}}, \tag{3.16}$$

then the camera noise after tone mapping will be barely visible.

Under typical brightness $\ln x$, using (3.11) and (3.14)–(3.16), we have:

$$T'(\ln x)\sigma_{\mathrm{c}}(\ln x) \leq \frac{1}{\sqrt{12}}. \tag{3.17}$$

In sRGB format, the display intensity ranges from 0 to 255. Through equation (3.1):

$$T(\ln x) = (255 - 0)\, P_{\ln x}(\ln x), \tag{3.18}$$

where $P_{\ln x}(\ln x)$ is the CDF of the pixel responses (proportion of captured brightnesses less than or equal to $\ln x$). Therefore, $T'(\ln x)$ equals:

$$T'(\ln x) = 255\, p_{\ln x}(\ln x). \tag{3.19}$$

Through (3.4), the PDF is approximated by the histogram function $h(\ln x)$. By combining (3.4) and (3.19), we obtain:

$$T'(\ln x) \approx 255 \frac{h(\ln x)}{N \Delta \ln x}. \tag{3.20}$$

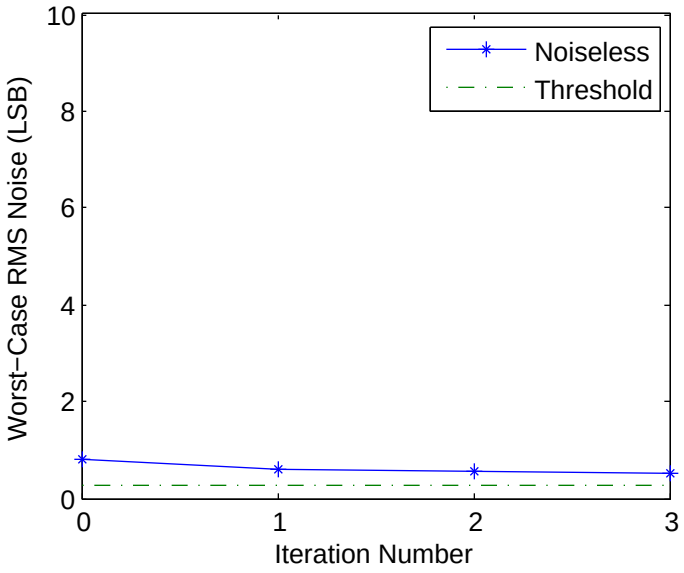

Figure 3.5: RMS noise of noiseless tone mapping versus number of iterations. Ideally, the RMS noise should be below the visibility threshold but it is close enough, even with zero interations.

However, when we count the histogram, the response is mixed with noise. Instead of counting $h(\ln x)$, we can only count $h(\ell)$ to approximate $h(\ln x)$:

$$T'(\ln x) \approx T'(\ell) \approx 255 \frac{h(\ell)}{N \Delta \ell}. \tag{3.21}$$

Finally, putting (3.21) into (3.17), we get:

$$h(\ell) \leq \frac{N \Delta \ell}{255 \sqrt{12} \sigma_{\mathrm{c}}(\ell)}. \tag{3.22}$$

This equation specifies a ceiling for each bin in the histogram. The ceiling prevents camera noise from exceeding the visibility threshold after tone mapping. When the pixel count in a bin of the histogram exceeds the ceiling, truncation of the count is a simple and reliable approach [10].

After truncation, the total number of pixels N in the histogram decreases because truncation removes pixels. Total number changes lead to ceiling changes. Therefore, the ideal condition cannot be reached after truncation. For still images, iteration can be employed until some condition is satisfied. Fig. 3.5 shows the tone-mapping worst-case RMS noise σ_{t} of all pixels in the bathroom image after the proposed tone mapping with iteration.

Through Fig. 3.5, the performance does not improve much with iteration. However, employing iteration will seriously affect efficiency. Therefore, iteration is omitted in our approach. Fig. 3.6(a) shows the bathroom image after noiseless tone mapping with zero iterations. The mapping function and RMS noise are plotted in Fig. 3.6(b). Comparing to the mapping function in Fig. 3.4(b), the slopes of the curve are changed because the

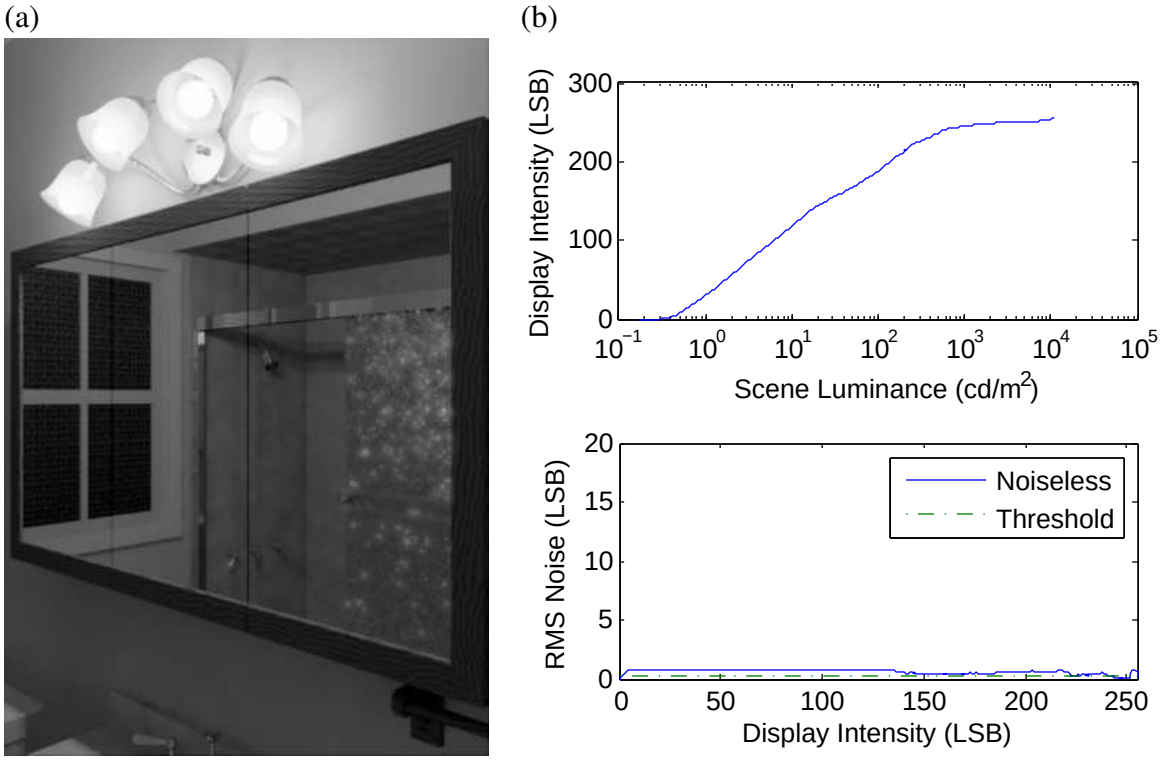

Figure 3.6: (a) Bathroom image with simulated camera noise after noiseless tone mapping. (b) The mapping function (top) and displayed RMS noise (bottom). Camera noise is invisible on the wall below the mirror.

histogram is changed by truncation. In Fig. 3.6(b), the RMS noise drops significantly but still exceeds the visibility threshold by a small amount. Yet, it is difficult for the human eye to detect the camera noise in Fig. 3.6(a). Fig. 3.7 depicts the process of the proposed tone mapping with noise ceilings.

3.3 Temporal Adaptation

Video is composed of a sequence of frames. The simplest method is to operate the tone mapping on each frame independently. Yet, this single-frame approach brings an unexpected effect when it encounters an abrupt DR change. The intuitive solution is to employ a multi-frame solution. Using a LPF, abrupt changes are eliminated.

Humans do not see each frame in a video independently. Our eyes take a period of time to adapt to the current DR if the DR changes. As reported by Hoefflinger [8], Durand and Dorsey modeled the temporal adaptation process of the human eye with an exponential decay function:

$$x_{\mathrm{p}}(t) = x_{\mathrm{p}}(t-T) + (x_{\mathrm{s}}(t) - x_{\mathrm{p}}(t-T))(1 - e^{\frac{-T}{\tau}}), \tag{3.23}$$

where $x_{\mathrm{s}}(t)$ is scene luminance at time t and x_{p} is perceived luminance. T is a time period between measurements of perceived luminance, and τ is a time constant. For

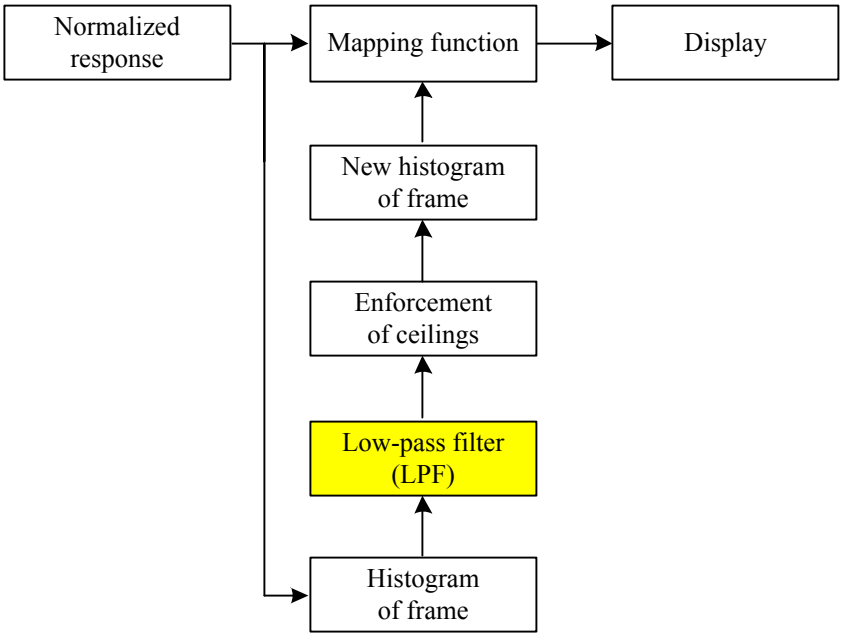

Figure 3.7: Flow chart of noiseless tone mapping. The highlighted block is added for noiseless tone mapping with temporal adaptation. Ceilings are calculated once beforehand. The mapping function is built from the new histogram.

a digital imaging system whose interval T between consecutive frames is constant, the exponential decay function in the discrete time domain becomes:

$$x_{\mathrm{p}}[n] = (1-\alpha)x_{\mathrm{s}}[n] + \alpha x_{\mathrm{p}}[n-1], \tag{3.24}$$

where

$$\alpha = e^{\frac{-T}{\tau}}. \tag{3.25}$$

Goodnight *et al.* [29] used this model on the logarithm of luminance (brightness), as a scale factor for each frame. This approach gave acceptable results.

Fig. 3.8 shows the process of temporal adaptation operated in luminance and brightness respectively, where both the scene and perceived luminance are $10^2\,\mathrm{cd/m^2}$ at the beginning, where the scene luminance changes to $10^3\,\mathrm{cd/m^2}$ abruptly, and where τ equals 0.4. This time constant is the worst-case one possible [25].

The above model coincides with our initial intuition. It shows perceived luminance may be approximated by passing scene luminance through a first-order LPF. Although there are differences when the LPF is operated on luminance or brightness, as shown in Fig. 3.8, both approaches are acceptable. In our method, the operator applied to brightness instead of luminance. Also, it is infeasible to apply a LPF to the response of each pixel. As display intensity is computed using a histogram of the current frame, we apply the LPF to the histogram:

$$h_{\mathrm{p}}(\ell)[n] = (1-\alpha)h_{\mathrm{s}}(\ell)[n] + \alpha h_{\mathrm{p}}(\ell)[n-1], \tag{3.26}$$

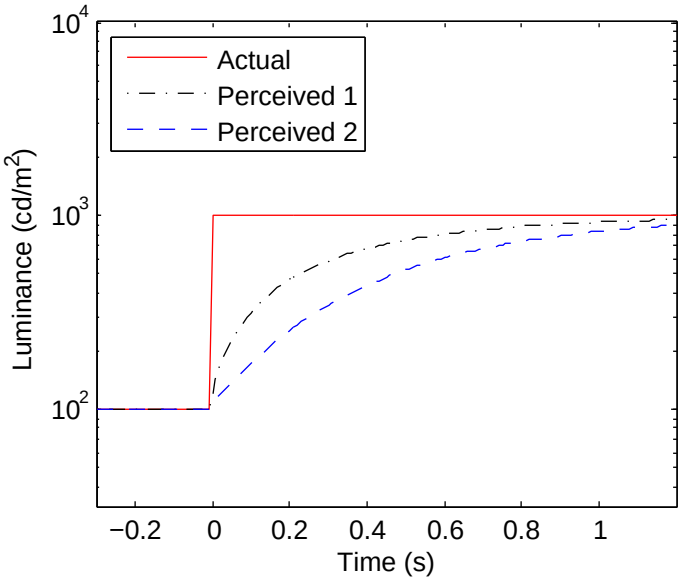

Figure 3.8: The process of temporal adaptation during luminance change. The Perceived 1 curve applied a LPF to luminance while a LPF is applied to brightness for the Perceived 2 curve. Both perceived luminances model temporal adaptation but the latter is easier to compute.

where α is calculated based on a time constant of the human eye and the frame period. $h_{\mathrm{s}}(\ell)$ represents the histogram of the current frame or scene. Because of the LPF, the filter output $h_{\mathrm{p}}(\ell)[n]$ may have a fractional part, which is meaningless. Therefore, we round it (not shown) to get the modified or perceived histogram $h_{\mathrm{p}}(\ell)$. The modified histogram $h_{\mathrm{p}}(\ell)$ is compared to the ceilings. A mapping function for the current frame in the video is derived from the modified histogram after truncation. Fig 3.7 presents the completed process of noiseless tone mapping, which employs both noise ceilings and a LPF.

3.4 Results

Unlike the evalution of FPN correction in Chapter 2, the evaluation of tone mapping is more subjective. The noiseless tone mapping is programmed in Matlab and C. Notwithstanding temporal adaptation, it suffices to test noiseless tone mapping offline with still images. On the other hand, noiseless tone mapping with temporal adaptation needs to be tested in real time with videos. Matlab was first used to process the captured images offline. Afterward, the complete noiseless tone mapping was programmed in C and embedded in a visual C++ framework. C++ tests focused on real-time performance. All the tests were done with the prototype imaging system introduced in Chapter 1, which employs a logarithmic CMOS APS image sensor. Section 3.4.1 discusses the Matlab experiments. The C++ experiments are discussed in Section 3.4.2.

3.4.1 Matlab Experiments

Before the experiments, FPN calibration was done for the image sensor. Through the calibration method introduced in Chapter 2, the correction coefficients were computed. Then the RMS residual noise of each pixel in the image sensor was computed. The median RMS noise was used in the noiseless tone mapping.

Images captured by the prototype imaging system were FPN corrected and tone mapped in Matlab. Matlab tests focused on offline performance for still images. Therefore, temporal adaptation is ignored. Fig 3.9 compares four images captured by our logarithmic image sensor after simple tone mapping, histogram equalization, and noiseless tone mapping, respectively. Two of the examples involve low-DR scenes while the other two involve high-DR scenes. Compared to simple tone mapping, histogram equalization provides a wider subjective DR. However, magnified noise degrades the quality of images while the bright parts may be overexposed. Noiseless tone mapping performs better in both low-DR and high-DR images. For low-DR images (mug and apple), the DR of the standard display is utilized more fully. In the high-DR image of a car, the headlight that was overexposed with simple tone mapping become visible. In the last example, a separator stands in the middle. At left, a bulb shines while a "10" printed on cardboard is on the right. The cardboard is underexposed with simple tone mapping. It becomes visible with noiseless tone mapping while the ring of the bulb can still be seen clearly. Performance in the dark part is worse because camera noise is relatively worse in dim lighting.

Good offline performance was demonstrated in Matlab experiments. Noiseless tone mapping is suitable for both low-DR and high-DR scenes. Real-time experiments are discussed below.

3.4.2 C++ Experiments

FPN correction and noiseless tone mapping were programmed in C to achieve high-speed processing. After implementation in a pre-existing Visual C++ framework, the frame rate of the digital camera was $45\,\mathrm{Hz}$. Fig. 3.10 presents a sequence of frames rendered by simple tone mapping, histogram equalization, and noiseless tone mapping with and without temporal adaptation. Except for the fourth operation which was computed in real time, the first three are all frame-based operations and were computed offline using logged data. As shown in Fig. 3.10, bulb switching introduces an abrupt DR change. The cardboard in (a) is underexposed after the bulb turned on. In (b), magnified noise degrades the quality of frames. The DR changes suddenly in (a), (b), and (c). In (d), a period is taken to adapt to the new DR, which is more natural considering human adaptation .

C++ experiments showed the proposed methods are suitable for real-time processing. Subjective DR is displayed while the visibility of camera noise is restricted. A LPF is employed on frame histograms to approximate temporal adaptation of the human eye. A low computational complexity makes real-time implementation feasible. This enables high frame rate and fluid video.

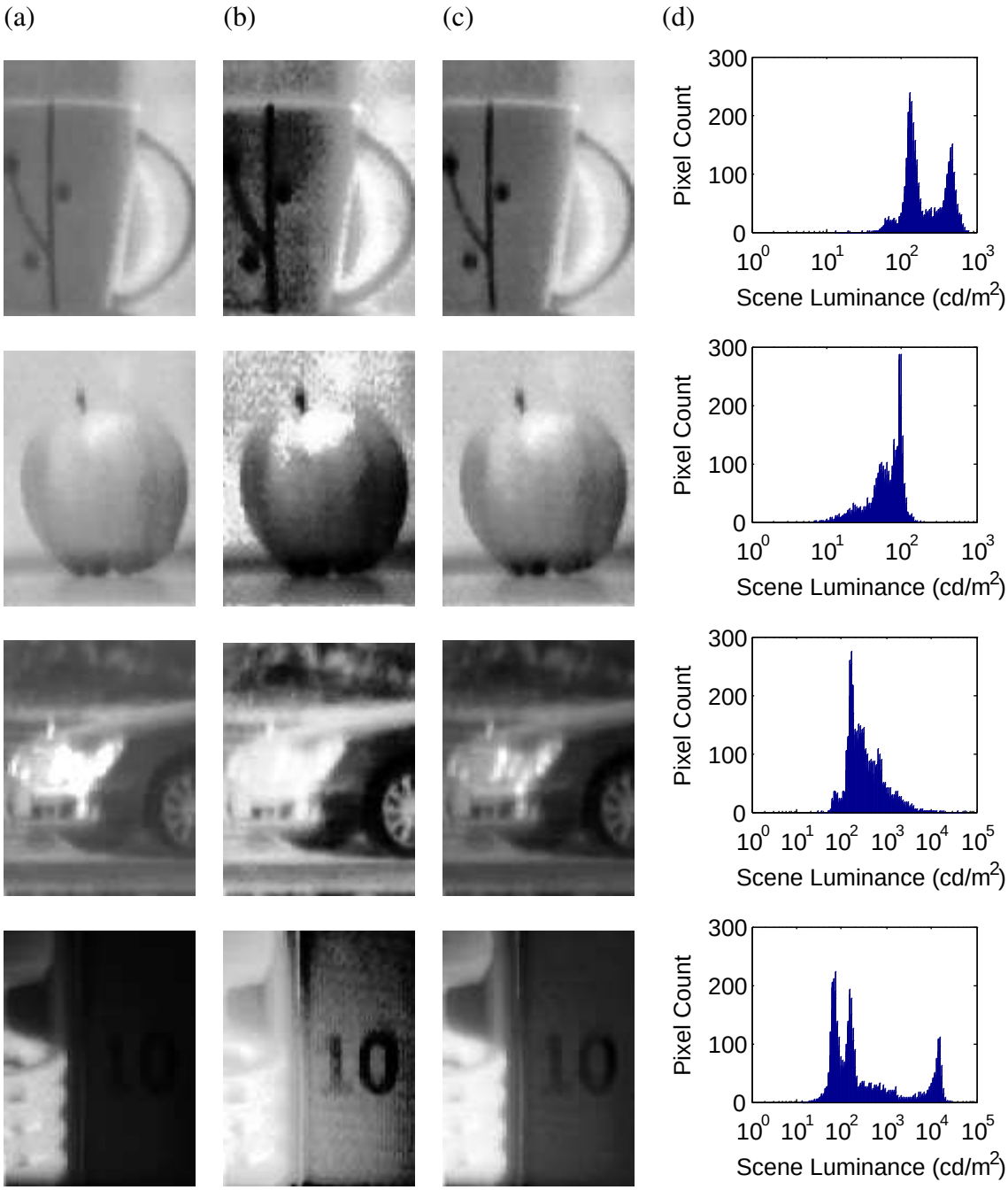

Figure 3.9: Low-DR (top two) and high-DR (bottom two) images after (a) simple tone mapping, (b) histogram equalization, and (c) noiseless tone mapping, respectively. Scene histograms, based on the original images, are given in (d).

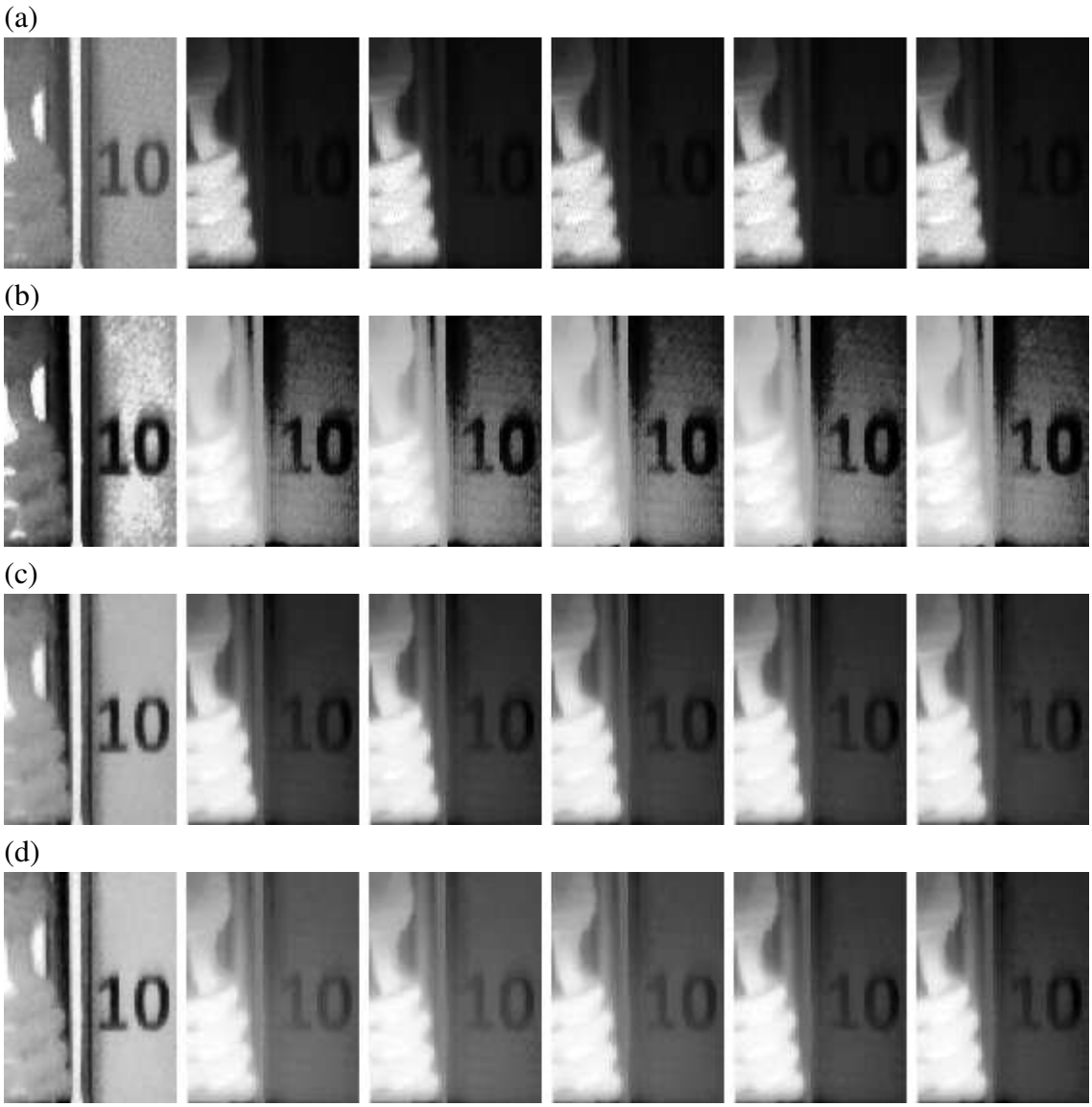

Figure 3.10: Video of scene changed from low DR to high DR. At left, a bulb is turned on while a printed "10" is on the right shielded by a separator. From (a) to (d) respectively, we use simple tone mapping, histogram equalization, noiseless tone mapping without temporal adaptation, and noiseless tone mapping with temporal adaptation. Although the frame rate is $45\,\mathrm{Hz}$, every ninth frame over a $1\,\mathrm{s}$ period is shown for brevity.

3.5 Conclusion

High-DR tone mapping has attracted wide interest. Both hardware-based and software-based methods have been proposed. Hardware-based methods try to expand the available DR of display devices. On the other hand, software-based methods compress the DR of high-DR images and videos to make them suitable for standard displays. Although high-DR display equipment is able to display high-DR scenes linearly, multiple reasons, including price and power consumption, prevent high-DR display equipment from replacing standard displays in the near future. Software-based methods are still mainstream.

Existing tone-mapping algorithms are divided into two categories: global operators and local operators. Local operators suffer from high computational complexities, which make them difficult to realize for real-time processing. Therefore, we return to global operators for efficiency. The tone mapping proposed in this chapter, which is developed from histogram equalization, targets videos captured by high-DR image sensors in real time. Existing tone-mapping methods assume that visual signals are free of noise. Unfortunately, this assumption is invalid for captured videos, especially with nonlinear image sensors. Previous tone mapping methods may therefore magnify noise to exceed the visibility threshold of displays, which is undesirable. Noiseless tone mapping solves this problem through adoption of noise ceilings. The noise is prevented from exceeding the visibility threshold of the sRGB display format.

Video is composed of a sequence of frames. Operating tone mapping on each frame independently is the simplest approach for video. Yet, the single-frame operator may produce unnatural effects when there is a sudden DR change. The temporal adaptation process of the human eye is modeled by an exponential decay function. A first-order LPF is adopted across frames to approximate the temporal adaptation.

Experiments were done with a prototype digital camera employing a logarithmic CMOS APS image sensor. The proposed method was programmed in Matlab and C++. The experiments in Matlab demonstrated good offline performance without considering temporal adaptation. C++ experiments showed the proposed method is also suitable for real-time processing because of its low-computational complexity. Although experiments were done with a standard logarithmic image sensor, the pixel response model does not affect the method. The proposed method can be applied for different nonlinear pixel designs. Currently, noiseless tone mapping was developed based on floating-point operations. The corresponding fixed-point design will be introduced in the next chapter.

Chapter 4

Fixed-Point Design

Digital imaging is widely used in research, medical, and commercial domains. Nonlinear image sensors are able to overcome the limit of low DR but suffer from high FPN, which degrades the quality of signals seriously. Moreover, displaying high DR scenes on standard DR displays is a difficult problem because of the lower DR. The previous two chapters introduced DSP solutions to these problems for nonlinear image sensors. This includes FPN correction and noiseless tone mapping. FPN correction reduces the FPN significantly to improve the SNDR of image sensors. Noiseless tone mapping maps scene luminance to display brightness while keeping perceptual DR and limiting noise visibility. Test results proved that the proposed methods are feasible and have good imaging performance based on floating-point operations.

For a digital camera, many parameters besides imaging performance need to be considered, such as compactness and power consumption. Previously, floating-point designs were programmed in Matlab and C++, which are high-level programming languages. Programming was realtively easy and precision was high. Not only that, wordlength limitations were negligible. Unfortunately, floating-point operations may be undesirable for a real-time system. High computational complexity makes it difficult to achieve a high frame rate. Unlike floating-point operations, fixed-point operations enable lower power consumption and higher compactness. With proper design, the precision loss can be minimized given the limitation of wordlength.

Fixed-point design for either FPN correction or tone mapping has not been widely investigated. Schneider proposed a fixed-point design for her FPN correction method [8]. A piecewise linear function is used for approximating a nonlinear model, which makes the corresponding fixed-point design straightforward. However, existing FPN correction methods [5, 21] suffer a trade-off between performance and complexity. On the other hand, tone mapping research focuses on offline processing instead of real-time processing. For offline usage, fixed-point design is unnecessary. Wang *et al.* designed a tone-mapping processor for a global tone-mapping algorithm called photographic [31]. Although the processor can achieve high frame rate and high resolution, it was not designed to limit the visibility of noise, which is important with nonlinear image sensors.

In this chapter, we provide fixed-point designs for the methods proposed in the two previous chapters. Compared with double-precision floating-point operations, fixed-point

operations bring inevitable errors. Yet, the error can be minimized through proper design. For FPN correction, we model the fixed-point errors. With a model of error, we program a method to compute the residual FPN, after fixed-point FPN correction, versus coefficient wordlength. Afterward, the optimal design is determined considering both the demands of performance and wordlength. For noiseless tone mapping, LUTs are implemented in the fixed-point design. Instead of staying constant, the LUT that stores the mapping function is updated each frame using only fixed-point operations. A fixed-point LPF is also adopted to approximate temporal adaptation.

4.1 FPN Correction

In Chapter 2, the method of FPN calibration and correction was introduced. IPR correction maps original responses to corrected responses through a polynomial function that is built during calibration. Three-parameter correction was tested in Matlab. The results showed that three-parameter correction were enough for the digital camera prototype under test. Without loss of generality, we present a fixed-point design for three-parameter correction. In this case, the polynomial function for the jth pixel is:

$$\hat{y}_j = (\hat{a}_{j2} y_j + \hat{a}_{j1}) y_j + \hat{a}_{j0}, \tag{4.1}$$

where y_j is the original response, $\hat{y}_j$ is the corrected response, and $\hat{a}_{j*}$ are the three parameters. Because they come from an ADC, the original responses are integers. The wordlength is decided by the specification of the ADC. We do not consider the quantization error in the original response, as it exists equally for both fixed-point and floating-point operation. There is no need for fixed-point calibration because calibration is done offline once. Therefore, calibration is still performed based on floating-point operation.

After calibration, the original coefficients for FPN correction all have fractional parts. Correction coefficients have different magnitude distributions as shown in Fig. 4.1. For fixed-point operation, the simplest method is to scale and round the original coefficients, which are then stored as integers. Coefficient scaling can be done by right or left shifting of the "decimal" point, in binary, of the original coefficients. Rounding eliminates the fractional parts.

Unlike floating-point addition, fixed-point addition of scaled numbers requires attention to scale factor and wordlength. "Decimal" point shifting and length adaptation are necessary to ensure numbers match before addition. With left shifting, fractional parts are simply truncated.

Both rounding with coefficient scaling and truncation before fixed-point addition introduce round-off error. Yet, there are differences between these two types of round-off error. The error caused by coefficient scaling and rounding is fixed for a specific pixel. Therefore, we call it static round-off error. Similarly, shifting in coefficient scaling is called static shifting. On the contrary, error caused by truncating before fixed-point addition changes with the original response even for a specific pixel. So it is called dynamic round-off error. Correspondingly, shifting before addition is called dynamic shifting. We

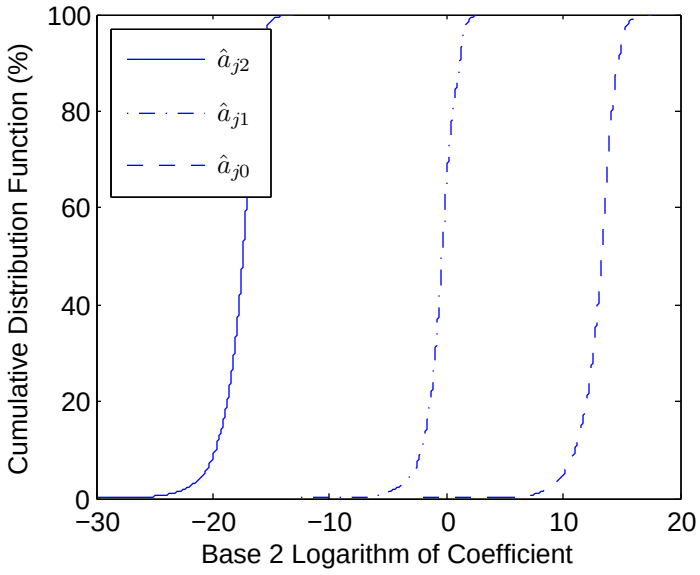

Figure 4.1: After FPN calibration, the magnitude distribution of each coefficient across pixels indicates a need for different scale factors. Using the same scale factor for each coefficient may result in saturation, depending on wordlength.

analyze the static and dynamic errors separately in the following subsections. Fig. 4.2 exhibits the whole process of fixed-point design for three-parameter FPN correction.

4.1.1 Static Round-Off Error

Because scaling is done with binary shifting, the scaling factor is always a power of two. For any pixel j, the kth static round-off error $\triangle A_{jk}$ depends on the original coefficient $\hat{a}_{jk}$ and scaling factor 2^{s_k}. After scaling and rounding, we obtain fixed-point coefficients:

$$A_{jk} = \text{round}(2^{s_k}\hat{a}_{jk}). \tag{4.2}$$

The rounding operation may introduce error. Static round-off error is:

$$\triangle A_{jk} = A_{jk} - 2^{s_k}\hat{a}_{jk}. \tag{4.3}$$

Therefore, scaling and rounding may be represented by the arithmetic process in Fig. 4.3(a). The static round-off error $\triangle A_{jk}$ has a limited domain, owing to the binary representation:

$$-\frac{1}{2} \leq \triangle A_{jk} \leq \frac{1}{2}. \tag{4.4}$$

For any pixel j, $\triangle A_{jk}$ is a constant given s_k.

Like A_{jk}, $\triangle A_{jk}$ is in effect scaled. Unscaling is required for a proper analysis. For static round-off error $\triangle A_{jk}$, there is a corresponding unscaled error $\triangle \hat{a}_{jk}$. Fig. 4.3(b) gives this model of rounding and scaling. The relationship between error $\triangle A_{jk}$ and unscaled error $\triangle \hat{a}_{jk}$ is:

$$\triangle A_{jk} = 2^{s_k} \triangle \hat{a}_{jk}. \tag{4.5}$$

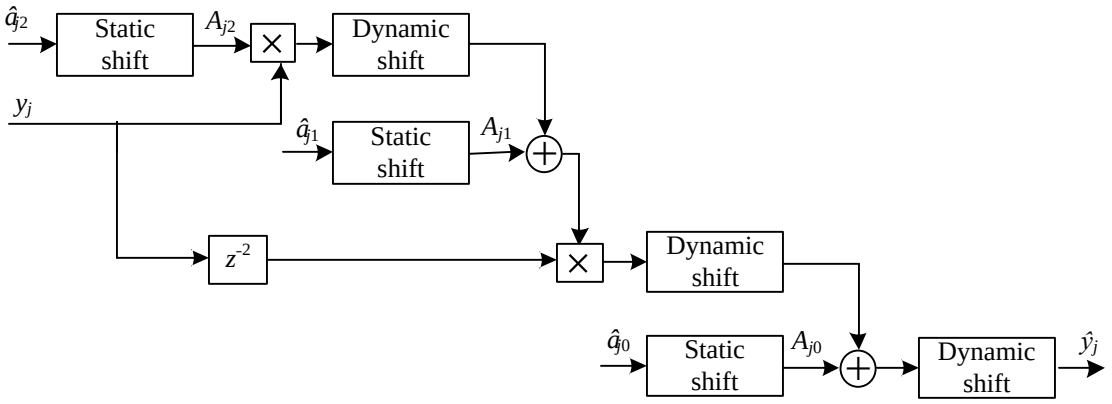

Figure 4.2: In this fixed-point design, static and dynamic shifting will introduce round-off errors. The total fixed-point error is the output-referred sum of round-off errors. We need to minimize it based on performance demands and wordlength limits.

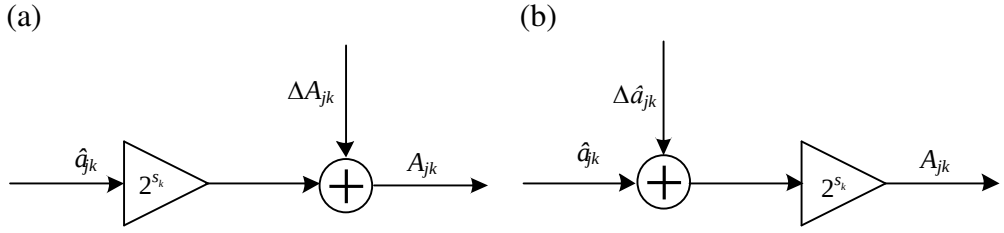

Figure 4.3: Coefficient scaling and rounding may be modeled in two ways: (a) original coefficient is scaled and static round-off error is added; or (b) original coefficient is added to unscaled static round-off error and the sum is then scaled.

Consequently, the domain of unscaled static round-off error is:

$$-\frac{1}{2^{s_k+1}} \leq \triangle \hat{a}_{jk} \leq \frac{1}{2^{s_k+1}}. \tag{4.6}$$

For any pixel j, we can calculate the total static round-off error $\triangle y_j^{\mathbf{s}}$ after fixed-point FPN correction as follows:

$$\triangle y_j^{\mathbf{s}} = ((\hat{a}_{j2} + \triangle \hat{a}_{j2})y_j + (\hat{a}_{j1} + \triangle \hat{a}_{j1}))y_j + (\hat{a}_{j0} + \triangle \hat{a}_{j0}) - \hat{y}_j \tag{4.7}$$

$$= \triangle \hat{a}_{j2} y_j^2 + \triangle \hat{a}_{j1} y_j + \triangle \hat{a}_{j0}. \tag{4.8}$$

For a specific pixel j, $\triangle y_j^{\mathbf{s}}$ is a constant given s_k.

From (4.6) and (4.8), we understand that increasing the scaling factor reduces the total static round-off error. However, increasing the scaling factor means less left shifting or more right shifting of "decimal" points before rounding. This increases the wordlengths needed to store coefficients without saturation. Moreover, there is residual error even with floating-point FPN correction. Total static round-off error need not be smaller than one LSB.

After coefficient scaling, the scaled coefficients are stored in RAM with fixed wordlength. Scaled coefficients below or above the minimum or maximum integer values, respectively,

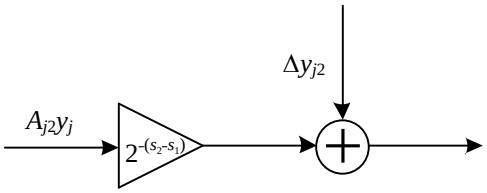

Figure 4.4: The model for first-level dynamic shifting can be represented as follows: the result of multiplication is scaled and then added to round-off error, which is introduced by a truncation operation.

are saturated. For example RAM words may be 16-bit signed integers, which ranges from $-32\,768$ to $32\,767$. Even though coefficient scaling may be optimized for round-off error, fixed-point performance is also limited by wordlength.

4.1.2 Dynamic Round-Off Error

In Fig. 4.2, the static shifts are done once offline after FPN calibration. FPN correction may be done using a pipeline structure. Three-parameter correction is completed with a four-delay pipeline. During each delay period, a multiplication or addition operation is completed. The first delay serves to compute $A_{j2}y_j$. Then the result is added to A_{j1}. As mentioned previously, fixed-point addition requires numbers to have both the same scale factor and wordlength. A_{j2} and A_{j1} are scaled by different factors 2^{s_2} and 2^{s_1}, respectively. With no loss of generality, assume s_2 is greater than s_1 because the magnitude of $\hat{a}_{j2}$ is much smaller than $\hat{a}_{j1}$. Before the addition, we can shift the "decimal" point of $A_{j2}y_j$ left $s_2 - s_1$ bits and truncate the fractional part, or shift the "decimal" point of A_{j1} right $s_2 - s_1$ bits. In theory, the left shifting approach introduces round-off error while the right shifting approach does not. However, left shifting is employed because it reduces wordlength, which lowers computational complexity. Reducing wordlength after multiplication is important because wordlength grows with multiplication, assuming no saturation.

Truncation after left shifting during FPN correction introduces dynamic round-off errors. After the first dynamic shift, the error is:

$$\triangle y_{j2} = \left\lfloor 2^{-(s_2-s_1)} A_{j2} y_j \right\rfloor - 2^{-(s_2-s_1)} A_{j2} y_j, \tag{4.9}$$

where $-1 \leq \triangle y_{j2} \leq 0$. The error varies with scene stimulus even for a fixed pixel. Because $A_{j2}y_j$ is an integer, $\triangle y_{j2} = 0$ when $s_1 \geq s_2$ (no left shift or no fractional part). Fig. 4.4 shows the error model for the first dynamic shift in Fig. 4.2. Similar models apply for the other dynamic shifts. The purpose of the final dynamic shift, which does not occur after multiplication, is to make the final result $\hat{y}_j$ have the same scale as the original response y_j.

Ignoring round-off errors, the following relation holds according to Fig. 4.2 and the preceding discussion:

$$\hat{y}_j = 2^{-s_0}(2^{s_0-s_1}(2^{s_1-s_2} A_{j2} y_j + A_{j1}) y_j + A_{j0}). \tag{4.10}$$

Therefore, the total dynamic round-off error $\triangle y_j^{\mathrm{d}}$ for any pixel is:

$$\triangle y_j^{\mathrm{d}} = 2^{-s_0}(2^{s_0-s_1}(2^{s_1-s_2}A_{j2}y_j + \triangle y_{j2} + A_{j1})y_j + \triangle y_{j1} + A_{j0}) + \triangle y_{j0} - \hat{y}_j \quad (4.11)$$

$$= 2^{-s_0}(2^{-(s_1-s_0)}\triangle y_{j2}y_j + \triangle y_{j1}) + \triangle y_{j0}. \quad (4.12)$$

For any pixel, the fixed-point error $\triangle y_j^{\mathrm{fixed}}$ is the sum of the static round-off error $\triangle y_j^{\mathrm{s}}$ and the dynamic round-off error $\triangle y_j^{\mathrm{d}}$, i.e.,

$$\triangle y_j^{\mathrm{fixed}} = \triangle y_j^{\mathrm{s}} + \triangle y_j^{\mathrm{d}}. \quad (4.13)$$

Let the residual error of the floating-point implementation be $\epsilon_j^{\mathrm{floating}}$, which is mainly due to camera noise and distortion. Then the residual error of the fixed-point implementation $\epsilon_j^{\mathrm{fixed}}$ will be:

$$\epsilon_j^{\mathrm{fixed}} = \epsilon_j^{\mathrm{floating}} + \triangle y_j^{\mathrm{fixed}}. \quad (4.14)$$

If $\epsilon_j^{\mathrm{fixed}} \gg \epsilon_j^{\mathrm{floating}}$ then image quality will be affected by the fixed-point design. On the other hand, trying to make $\epsilon_j^{\mathrm{fixed}}$ equal $\epsilon_j^{\mathrm{floating}}$ may lead to an inefficient fixed-point design. Ideally, $\epsilon_j^{\mathrm{fixed}} \approx \epsilon_j^{\mathrm{floating}}$.

4.2 Noiseless Tone Mapping

In Chapter 2, we introduced a LUT to realize simple tone mapping. The LUT is an ideal alternative to avoid complex computation and keep accuracy. Fixed-point noiseless tone mapping also adopts the LUT approach. With simple tone mapping, the LUT stores the mapping results for all possible inputs. Because the mapping function is fixed, the LUT can be built once beforehand and stays constant during processing. Yet, the mapping function of noiseless tone mapping changes from frame to frame. Therefore, the LUT that stores the mapping function is updated in each frame. The algorithm of noiseless tone mapping was introduced in Chapter 3. The problem here is how to build and update the mapping function with fixed-point operations. Chapter 3 developed noiseless tone mapping in two steps. First, noise ceilings were introduced. Afterwards, it described temporal adaptation. In this section, we follow the same sequence to develop a fixed-point design, firstly, for tone mapping with noise ceilings only and, secondly, with temporal adaptation also.

4.2.1 Noise Ceilings

Noiseless tone mapping without temporal adaptation works as follows. First, a histogram computation counts the number of pixels in predefined bins of intensity values. Noise ceilings, which are functions of image sensor and sRGB display noise, are computed once beforehand. In some bins of the histogram, the pixel count, which exceeds the corresponding noise ceiling, is truncated. This leads to a new histogram. All of the real-time steps readily compute with fixed-point operations when pixel intensities are integer

values, which they are after fixed-point FPN correction. Therefore, the only problem left is how to build a mapping function from the new histogram. The mapping function is built from a CDF, which is approximated from the new histogram, as follows:

$$W(\ell) = \text{round}\left(\frac{255\sum_{\ell_{\min}}^{\ell} h_{\text{new}}(\ell)}{N_{\text{new}}}\right). \tag{4.15}$$

Therefore, division is needed in this step. With fixed-point operation, division should be avoided because it is difficult to implement and suffers from low precision. This encourages us to consider an alternative to avoid division.

The total pixel count N_{new} is accumulated from the new histogram after bin truncation. It varies from frame to frame because the new histogram is different in each frame. For the total pixel number N_{new}, the range depends on the original histogram and the noise ceilings. The maximum value occurs when each bin count equals its noise ceiling. In contrast, the minimum value is difficult to determine. Therefore, we simply take 1 as the minimum value, which will be less than the true minimum value. Assuming the histogram has L bins, there are L ceilings corresponding to each bin. The range of N_{new} is:

$$1 < N_{\text{new}} \leq \sum_{\ell=1}^{L} N_{\text{ceil}}(\ell). \tag{4.16}$$

Now, (4.15) is able to be rewritten as follows:

$$W(\ell) = \text{round}\left(a \cdot \sum_{\ell_{\min}}^{\ell} h_{\text{new}}(\ell)\right), \tag{4.17}$$

$$a = \frac{255}{N_{\text{new}}}. \tag{4.18}$$

Therefore, parameter a is a function of N_{new}. As described previously, the ceilings are constant for a specific image sensor. Therefore, we can compute a for all possible N_{new} beforehand. The range of a is:

$$\frac{255}{\sum_{\ell=1}^{L} N_{\text{ceil}}(\ell)} \leq a \leq 255. \tag{4.19}$$

In the next step, all possible a are scaled by a factor 2^{s_a} and rounded, which is similar to the scaling in FPN correction:

$$A = \text{round}(2^{s_a} a) \tag{4.20}$$

Therefore, the unscaled round-off error is:

$$\triangle a = \frac{A - 2^{s_a} a}{2^{s_a}}, \tag{4.21}$$

where the range is:

$$-\frac{1}{2^{s_a+1}} \leq \triangle a \leq \frac{1}{2^{s_a+1}}. \tag{4.22}$$

The error is determined by the scaling factor. The scaling factor choice is made based on precision requirments. Finally, all the parameters A are stored in a LUT using a fixed-point format.

To build a mapping function, the A corresponding to N_{new} is read out from the LUT. Afterwards, the cumulative histogram $\sum_{\ell_{\min}}^{\ell} h_{\text{new}}(\ell)$ multiplies with A to get gray intensity W:

$$W(\ell) = \text{round}\left(2^{-s_a} A(N_{\text{new}}) \sum_{\ell_{\min}}^{\ell} h_{\text{new}}(\ell)\right) \tag{4.23}$$

Finally, these gray intensities are stored in a LUT that defines a function to tone map each pixel. The error brought by the round operator in (4.23) is the display noise ϵ^{d} introduced in Chapter 3. Therefore, it exists with both the floating-point and fixed-point operations, and has already been considered. Using an LUT for multiplication simplifies the fixed-point design for noiseless tone mapping because the fixed-point division is avoided while precision can be kept.

4.2.2 Temporal Adaptation

In noiseless tone mapping, a first-order LPF is used to approximate temporal adaptation of the human eye. Chapter 3 described the complete process. The same scaling method used in previous sections can be used here. Additionally, the LPF needs to be realized with fixed-point operations.

From Chapter 3, the first-order LPF is defined by:

$$h_{\text{p}}(\ell)[n] = \text{round}(\beta h_{\text{s}}(\ell)[n] + \alpha h_{\text{p}}(\ell)[n-1]), \tag{4.24}$$

where $\beta = 1 - \alpha$. The two coefficients α and β, which are calculated based on time constants, both range from 0 to 1. First, they are scaled and rounded:

$$\alpha' = \text{round}(2^{s_h}\alpha) \tag{4.25}$$

$$\beta' = \text{round}(2^{s_h}\beta). \tag{4.26}$$

Therefore, the unscaled round-off error for α is:

$$\triangle\alpha = \frac{\alpha' - 2^{s_h}\alpha}{2^{s_h}}, \tag{4.27}$$

and the range is:

$$-\frac{1}{2^{s_h+1}} \leq \triangle\alpha \leq \frac{1}{2^{s_h+1}}. \tag{4.28}$$

The case of β is similar.

During the processing, a left shift of the "decimal" point scales the result back to the original magnitude:

$$h_{\text{p}}(\ell)[n] = \text{round}(2^{-s_h}(\beta' h_{\text{s}}(\ell)[n] + \alpha' h_{\text{p}}(\ell)[n-1])). \tag{4.29}$$

The round operation removes the fractional part. As with (4.23), the error caused by the round operation is ignored; it exists with both the floating-point and fixed-point designs. The design of the fixed-point LPF is presented in Fig. 4.5.

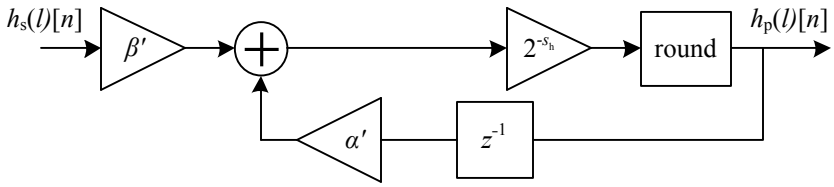

Figure 4.5: Fixed-point first-order low-pass filter, which is used for noiseless tone mapping with temporal adaptation. Filter coefficients are first scaled and then stored in fixed-point format. Finally, the result is scaled back.

4.3 Results

Tests of the fixed-point design are done using the prototype digital camera described in Chapter 1. The fixed-point design of FPN correction and noiseless tone mapping was programmed in both Matlab and C. The performance of the fixed-point implementation was evaluated by comparing its results to those of the floating-point implementation. In Matlab, we wrote a program to compute the RMS residual error after fixed-point correction versus wordlength of scaled coefficients. Using different wordlengths for parameters in the CDF computation and in the LPF, the performance of noiseless tone mapping was also computed. Such analysis helped to understand the trade off between performance and wordlength to determine the optimal design. Afterwards, the optimal design was programmed in C for a real-time test. In this manner, the feasibility and good performance of the fixed-point design in real time was demonstrated.

4.3.1 Matlab Experiments

For the performance evaluation of FPN correction in all pixels of the image sensor, the RMS residual error is a suitable criterion. Uniform images are captured under M different luminances and are used for FPN calibration. The number of pixels is N. After FPN correction, the residual error for pixel j under luminance level i is:

$$\hat{\epsilon}_{ij} = \bar{y}_i - \hat{y}_{ij}, \tag{4.30}$$

where $\bar{y}_i$ is the ideal response and $\hat{y}_{ij}$ is the corrected response. The RMS residual error $\sigma_\epsilon^{\text{fixed}}$ of the fixed-point correction is:

$$\sigma_\epsilon^{\text{fixed}} = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(\epsilon_{ij}^{\text{fixed}})^2}{DOF}}, \tag{4.31}$$

where the DOF for three-parameter correction is [5]:

$$DOF = MN - M - 3N. \tag{4.32}$$

In Section 4.1, we showed that fixed-point error depends on scaling factors, coefficient values, and original responses. On the other hand, scaling factors and the range

of original coefficients determine the wordlength of scaled coefficients. Therefore, the performance of fixed-point correction varies with wordlength. The total wordlength for scaled coefficients of one pixel is preferred to be an integer number of bytes to improve storage efficiency. For multi-parameter correction, different wordlengths for each coefficient will impact correction performance. The optimal wordlength distribution can minimize the RMS residual error for a specific total wordlength. Fig. 4.6 shows the minimum RMS residual error versus total wordlength and the corresponding optimal wordlength distribution, as determined by the computer program we wrote.

In Fig. 4.6(a), the RMS residual error decreases as wordlength increases. This means better correction performance. When the total wordlength reaches 52 bits, the fixed-point correction has equal performance to floating-point correction. Yet, besides correction performance, storage efficiency is another important factor. Although a difference still exists, the performance with 48 bits already approximated floating-point performance very well. In addition, 48 bits is a wordlength that occupies exactly six bytes. For wordlength distribution, the program decides the optimal configuration based on RMS residual error minimization. The optimal distribution versus total wordlength is plotted in Fig. 4.6(b). Therefore, a 48-bit wordlength and its corresponding wordlength distribution are the optimal FPN correction design for our logarithmic CMOS image sensor.

Testing of the fixed-point noiseless tone mapping is different. This is because the mapping of uniform images is meaningless. Instead, four captured images are mapped, which include two low-DR images and two high-DR images. However, the evaluation method is similar. The performance is evaluated through a comparison between the fixed-point and floating-point designs. For noiseless tone mapping without temporal adaptation, the performance only depends on the wordlength of parameter A in the CDF computation. Multiple images were mapped by the fixed-point design adopting different wordlengths. The RMS difference between images mapped by floating-point and fixed-point designs was computed. The RMS difference versus wordlength and mapped images are shown in Fig. 4.7. As shown, RMS differences are lower than or close to 1 LSB with a 15-bit wordlength. Considering storage efficiency and C data types, a 16-bit wordlength is an optimal choice and offers a little extra precision.

For temporal adaptation, the fixed-point LPF was employed. Two 200-frame sequences, which record motion and an abrupt DR change, were mapped. The fixed-point tone mapping design adopts a 16-bit multiplication-parameter LUT and stores two LPF coefficients in 8 bits each. Figs. 4.8(a) and (b) show selected frames in these sequences. Afterwards, each frame in the sequence was compared. The RMS difference of each sequence in entirety, versus wordlength of LPF coefficients, is shown in Fig. 4.8(c) and (d). As shown, RMS differences are lower than 1 LSB with the 8-bit wordlength. Since there are only two coefficients in the LPF, wordlength in terms of an integer number of bytes is not critical. This means storage efficiency is not as important a factor here as with the FPN correction coefficients and tone mapping LUTs. However, considering readily-available data types in C, an 8-bit wordlength is ideal.

Matlab tests help us to determine the optimal fixed-point design, including FPN correction and noiseless tone mapping, for our prototype image sensor. The optimal designs

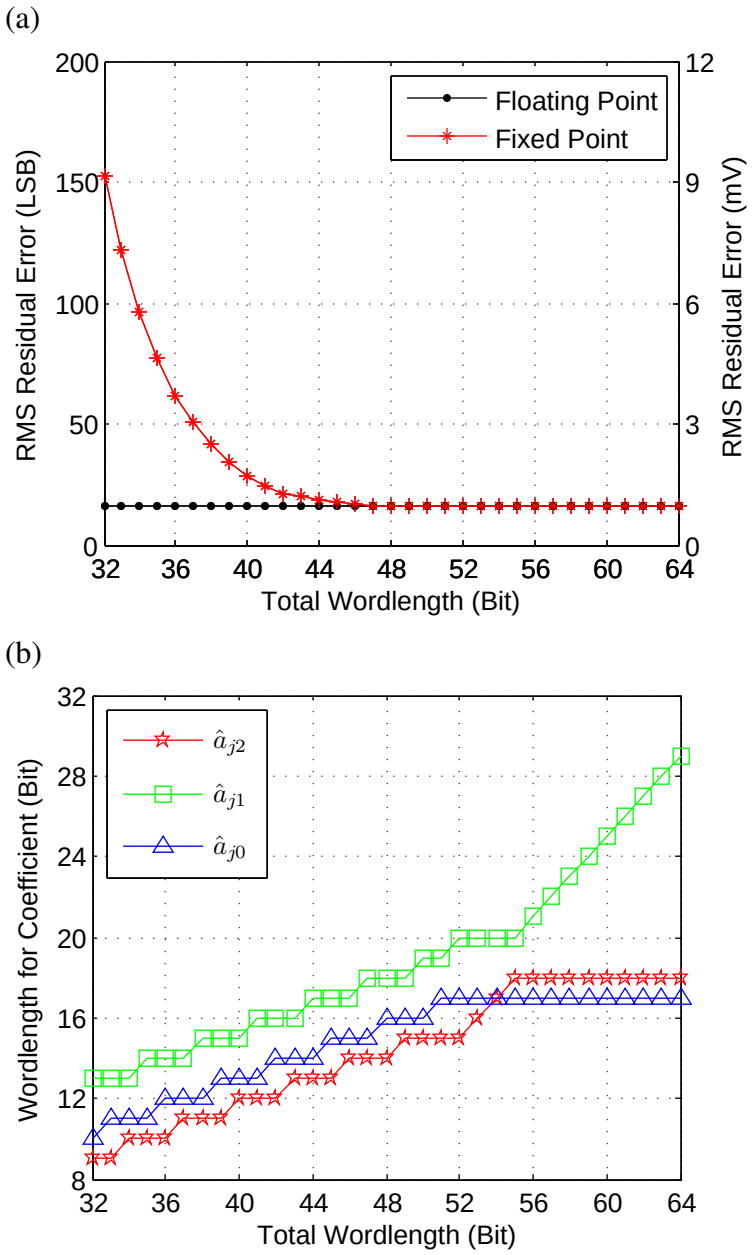

Figure 4.6: (a) RMS residual error is represented in two ways: digital response (left Y axis) and voltage (right Y axis). (b) Optimal wordlength distribution for coefficients, which is decided by minimizing RMS residual error for each total wordlength.

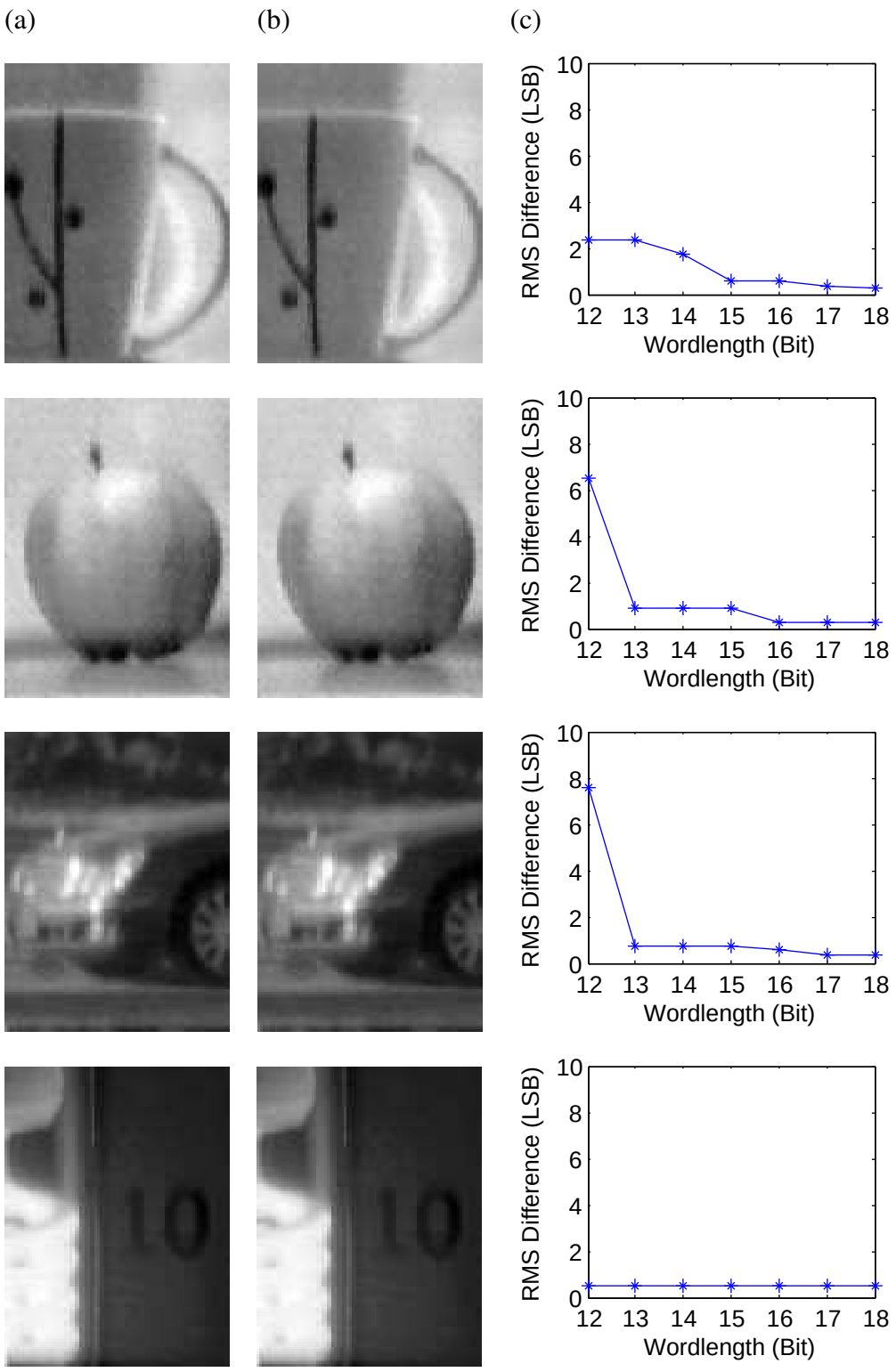

Figure 4.7: Images after (a) tone mapping by floating-point operation and (b) tone mapping by fixed-point operation (optimal design). (c) RMS difference versus parameter wordlength. Visual differences are quantified on the right. The RMS difference reduces with wordlength increase. Differences are difficult to see when the images are mapped by the optimal fixed-point design.

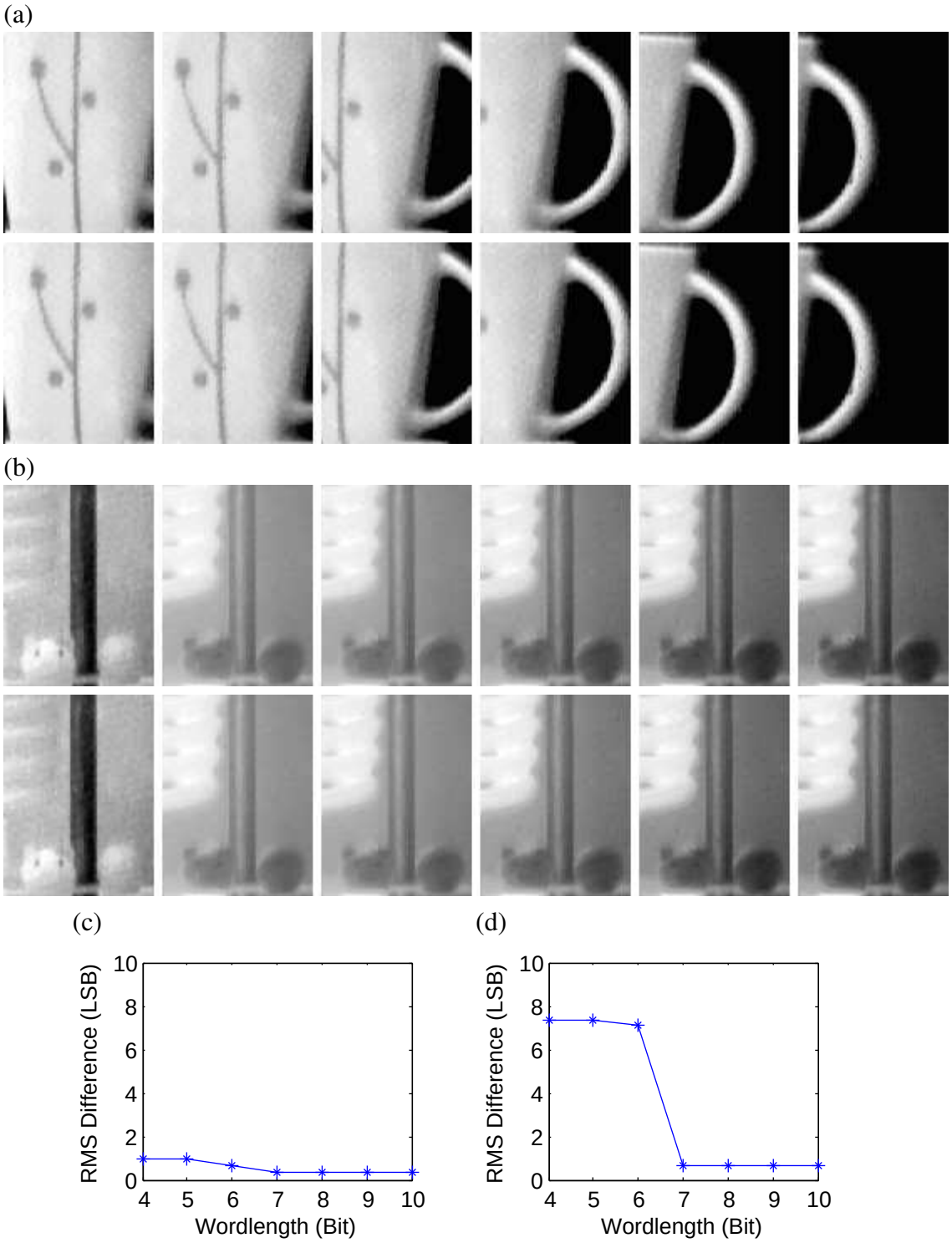

Figure 4.8: Frames in two sequences show (a) motion and (b) an abrupt DR change (when the bulb is turned on). The top frames use floating-point noiseless tone mapping (with temporal adaptation) while the bottom frames use the fixed-point design. RMS differences of sequences (a) and (b) are shown in (c) and (d) under different wordlengths for LPF coefficients. Although there are 200 frames over a 4 s period, every tenth frame is shown over a 1 s period.

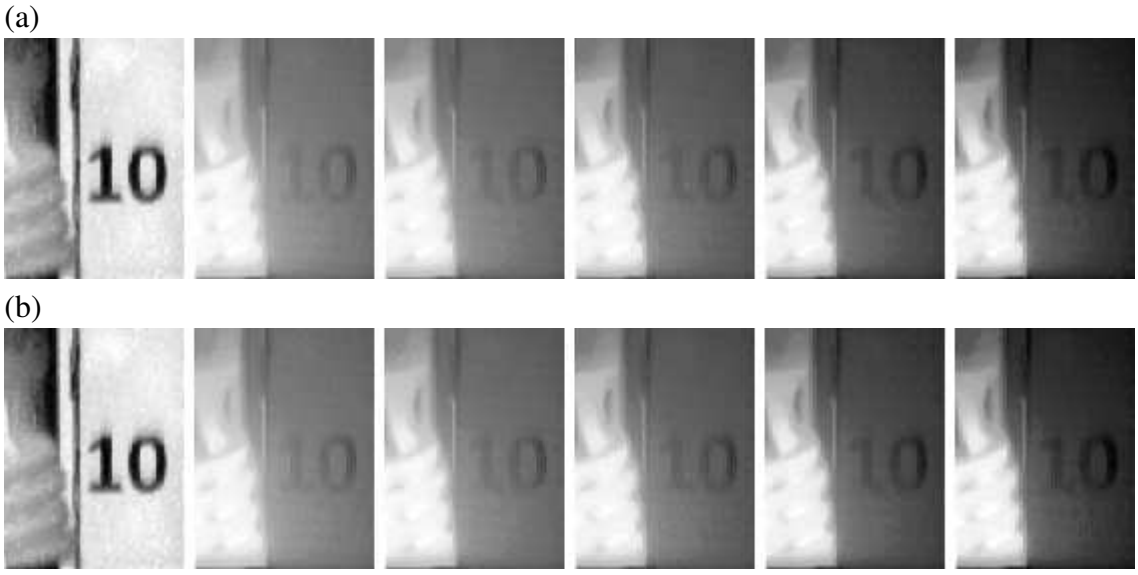

Figure 4.9: Corresponding frames of a video after (a) floating-point operation in Matlab and (b) fixed-point operation in C (real time). There is an abrupt DR change happening when the bulb is turned on. The difference between (a) and (b) is difficult to detect by eye. Although the frame interval is $20\,\mathrm{ms}$, every tenth frame is shown over a $1\,\mathrm{s}$ period.

were programmed in C and implemented in a Visual C++ framework to test the performance in real time.

4.3.2 C++ Experiments

In the C++ experiments, both FPN correction and noiseless tone mapping were programmed in C based on the optimal fixed-point design, which was decided in Section 4.3.1. Fig. 4.9 exhibits the frames in a video, which were processed using floating-point and fixed-point operations separately. The fixed-point design was computed in real time while the floating-point design was computed offline using logged data. Through comparison of the frames in Fig. 4.9, differences are not obvious to detect by eye. The result proves the optimal fixed-point design works in real time with good performance, which only brings minor error that is essentially imperceptible.

After implementing the fixed-point design, the prototype digital camera works at $50\,\mathrm{fps}$ instead of $45\,\mathrm{fps}$. Although the floating-point performance of the desktop PC prevented more obvious improvement in frame rate, the value of fixed-point design cannot be underrated. The low computational complexity makes the design simpler for an FPGA, which is a good platform to achieve an embedded system. In addition, it implies power consumption is reduced, which is especially important for portable devices.

4.4 Conclusion

FPN correction and noiseless tone mapping for logarithmic CMOS image sensors were introduced in Chapter 2 and Chapter 3. Both of them were based on floating-point operations. In this chapter, the corresponding fixed-point design was provided. The simplicity and minor precision loss make it a better choice for real-time processing.

For FPN correction, the fixed-point design brings inevitable fixed-point error. The fixed-point error is caused by static and dynamic round-off errors. Both error types were modeled and analyzed. Static round-off error results from a rounding after static shifting during coefficient scaling. Truncation after dynamic shifting leads to dynamic round-off error. The error model for each error type was derived. Although we did not estimate RMS fixed-point error through the error model directly, the theoretical analysis was still valuable. It helped us program code to achieve an optimal fixed-point design for FPN correction.

For fixed-point noiseless tone mapping, LUT usage is an ideal method for its simplicity and minor precision loss. Two constant LUTs, which store noise ceilings and multiplication parameters respectively, are required. The histogram and mapping function are stored in two LUTs also. These are updated frame by frame. A fixed-point LPF is introduced to approximate temporal adaptation of the human eye. Abrupt DR changes are avoided.

In the tests, a prototype digital camera employing a logarithmic CMOS image sensor was utilized. The performance of the fixed-point design varies with specified wordlength. The codes programmed in Matlab compute RMS residual error for fixed-point FPN correction and RMS difference for fixed-point noiseless tone mapping, under different specified wordlengths. Then, we determine the optimal fixed-point design by considering performance and wordlength. Therefore, Matlab tests help us to determine the optimal designs. Correspondingly, the optimal fixed-point designs were programmed in C++ for real-time tests. Fluid and good-quality videos demonstrated good performance of the optimal fixed-point designs and, therefore, the proposed methods.

After implementation of the fixed-point design in a Visual C++ framework, the frame rate of the system increased from 45 to $50\,\mathrm{Hz}$, an 11% increase despite the high floating-point performance of a desktop computer. More importantly, the fixed-point design enables a lower power consumption, which is especially important for mobile applications.

Chapter 5

Conclusion

Low DR is an important weakness for modern digital cameras compared with human eyes [1]. The DR of CMOS image sensors can be widened through different methods [2]. Linear image sensors cannot achieve high DR directly but they benefit from technologies like frequency-based sensors [54, 55], time-to-saturation sensors [56, 57], and multiple exposure sensors [58, 59, 60]. Multiple exposure sensors are most popular among them. An algorithm is needed to integrate multiple images under different exposures but captured from the same scene to get the final high-DR image [61]. However, the mechanism is complex and it may have trouble to capture dynamic scenes or to record videos. On the other hand, logarithmic CMOS image sensors are able to capture high-DR scenes in one exposure [3, 18, 2]. High-DR imaging can be utilized in research, medical, and commercial domains. Because of its built-in advantage in DR, employing logarithmic response instead of linear response is a promising alternative to overcome the limitation of low DR.

Logarithmic image sensors have a great advantage in DR, but they suffer from low SNDR. This weakness makes it difficult for them to compete with linear image sensors which are good at achieving high image quality. The low SNDR results from serious FPN and low SNR. Through digital pixel sensor (DPS) [16] and vertical integration [1] technology, the SNR of logarithmic image sensors is improved while achieving a wider DR that even exceeds the human eye. However, high FPN prevents logarithmic image sensors owning high SNDR even with high SNR. Using the new FPN correction algorithm proposed in Chapter 2, the SNDR of logarithmic image sensors approximates the SNR very well in the whole DR. The test result certifies good off-line performance. In addition, real-time correction ensures a high frame rate for video applications.

Visual signals, such as images and videos are often intended for human consumption. Tone mapping is a process that converts the digital responses from scene stimulus to display intensity [10]. Logarithmic image sensors can capture high-DR scenes with high quality. However, tone mapping for high-DR scenes is still difficult since the standard display equipment offers only low DR. Linear tone mapping does not work well for high-DR scenes. They cannot make both the dark part and bright part visible. The proposed method of noiseless tone mapping compresses DR effectively and efficiently to make the high-DR scenes fit the standard display range, while also considering the noise tolerance of the sRGB display format.

Because of their low computation complexity, the proposed FPN correction and noiseless tone mapping both have the potential for real-time processing. For real-time processing, frame rate is a key parameter. Fixed-point operations replace floating-point operations for higher efficiency and lower consumption. Although fixed-point operations introduce inevitable errors, the precision loss is minimized through the design methodology of Chapter 4.

This thesis developed the DSP required to improve the performance of nonlinear image sensors so that they can be used in digital video cameras. This includes FPN correction, noiseless tone mapping, and corresponding fixed-point design. Tests of the proposed methods were done using a custom digital camera employing a logarithmic CMOS image sensor. The results proved the performance of the presented methods. The contributions of FPN correction, noiseless tone mapping, and fixed-point design are presented in Section 5.1. Future work is discussed in Section 5.2.

5.1 Contributions

High FPN is an important factor that degrades the signal quality captured by logarithmic image sensors. Existing correction methods [5, 21] cannot achieve both good image quality and real-time performance. A novel method proposed in the thesis breaks through this limitation. Moreover, its generality makes it applicable to other nonlinear image sensors. Tone mapping is difficult for high-DR scenes because of a limitation in the DR of standard display equipment. Moreover, tone mapping also confronts new challenges with logarithmic image sensors because previous assumptions regarding images and videos are no longer valid. The key novelty of the proposed tone mapping method is it prevents camera noise from exceeding the tolerance of the standard display format after processing. The novel FPN correction and tone mapping both have a low computation complexity. The corresponding fixed-point design ensures the proposed methods are achievable in an embedded system for real-time processing. The work proposed in the thesis has been used in a poster presentation [62] and a technical report [63]. Additionally, two PhD theses [1, 16] employed the proposed FPN correction method in their tests.

We categorize the contributions of this thesis into three parts following the same structure of the whole thesis. In Section 5.1.1, we review the contribution of the novel FPN correction method, which overcomes the limitation between performance and complexity. Additionally, high generality makes it work for different image sensor designs. Section 5.1.2 reviews the contribution of noiseless tone mapping. This method is designed for logarithmic image sensors because it considers camera noise in the captured images and videos and restricts noise magnification. Finally, the contribution of fixed-point design is reviewed in Section 5.1.3. Not only did it provide a fixed-point design for a specific logarithmic CMOS image sensor, but it also offered a method to get an optimum fixed-point design for different image sensors.

5.1.1 Fixed Pattern Noise Correction

Both analog and digital methods have been investigated to correct FPN in logarithmic CMOS image sensors. The main advantage of analog methods is high speed. Because responses are corrected inside pixels, corrected responses can be output with little delay [18, 64]. Unfortunately, the logarithmic pixel circuit is complicated and analog methods do not work well. Correction performance is far from good. On the other hand, digital methods have became popular because of their accuracy and flexibility. Most previous methods either approximate the nonlinear FPN model with a single linear model or with a piecewise linear model [6, 7, 21]. A method that corrects logarithmic FPN directly has high computation complexity [5]. Generally, the limitation between performance and complexity is difficult to overcome for existing methods. In addition, because the correction methods were developed based on a specific response model, they are tied to logarithmic pixels instead of being a general method for nonlinear pixels.

The response of any nonlinear high-DR image sensor may be mapped to a logarithmic scale. The scene stimulus, such as visible-band luminance, needs to be rendered from the logarithmic scale for linear analysis. This step is called response linearization. Different from FPN correction, response linearization is a relatively unexplored problem of particular relevance to nonlinear image sensors. An existing method [8] was developed from a specific response model. The problems of this method are similar to the ones of FPN correction. Accuracy may be low and generality is weak.

The novel method presented in this thesis, which covers FPN correction and response linearization, breaks through the existing limitations using Taylor series, inverse polynomial regression, and spline interpolation. The FPN correction is inspired by the relation between average response and individual response. Although the response of a high-DR pixel is a nonlinear function, the relation between average response and individual response is close to linear instead of highly nonlinear. Therefore, a truncated Taylor series may be used to build a function that either maps the average response to individual response or vice versa. This calibration step is done once offline. In contrast, FPN correction must be done repeatedly in real time. Inverse mapping moves heavy computation from correction to calibration. FPN correction calculates a polynomial for each pixel, which only needs arithmetic operations. Moreover, the polynomial calculation can be done with a pipeline structure. This property is very suitable for achieving the correction in an FPGA. Previously, the response model was built through circuit analysis. In contrast, the proposed FPN correction built the polynomial function through a numerical method instead of an analytic method. Circuit analysis is not needed because a concrete response model is unnecessary. This makes the correction method general for different response models.

For response linearization, the situation is similar. The previous method does not work because there is no analytic model. Another numerical method, namely spline interpolation, is employed with the same calibration data used for FPN correction. The complicated nonlinear model is replaced by a piecewise polynomial function, which is built through spline interpolation. Computation complexity is low because only polynomial calculation is needed. It is also a general method because no analytic model is

required.

The tests were done in Matlab and C++ using a prototype digital camera with a logarithmic CMOS image sensor. Matlab tests focused on qualitative offline performance. The performance was proved by quantitative analysis and qualitative of images. C++ is suitable for real-time tests, which provided high frame rate to make the video fluid. In general, the proposed FPN correction and response linearization is a novel method that owns both high performance and low complexity. Both the off-line and real-time performances are good. In addition, the method is general for different nonlinear response models, although tests were done only with the logarithmic image sensor. Both generality and simplicity are obtained with the numerical methods.

5.1.2 Noiseless Tone Mapping

Tone mapping for high-DR scenes has been widely investigated in the literature. It can be categorized into global operators and local operators depending on the mapping functions. Compared to global operators, local operators require more sophisticated computations. Some local operators process high-DR images on different layers [43, 25, 65] while other methods process gradient fields [24] or radiance maps [45] instead. Although some methods reported good performance, high computation complexity restrict them only to off-line processing. The performance of global operators are good enough in most cases. In addition, a relatively low complexity makes the global operators preferred when real-time processing is necessary.

Most tone mapping methods are focused on high-DR images while few of them are designed for high-DR videos specifically. Previous methods are built based on an assumption that high-DR images and videos are free of camera noise. This assumption is valid for synthetic images and videos obtained through computer graphic techniques. Unfortunately, it is invalid for captured images and videos. All cameras have a limited SNDR, which indicates that captured signals are still noisy even after FPN correction. Neglecting this camera noise may result in display noise after tone mapping, which degrades the quality of displayed images or videos.

The noiseless tone mapping presented in the thesis originates from histogram equalization. Camera noise after FPN correction is determined from the calibration data. Noise changes after tone mapping are modeled. Ceilings are enforced to prevent noise after tone mapping from exceeding the tolerance of the display format. DR is compressed or expanded but noise is kept below a tolerance to guarantee quality. For video tone mapping, a luminance adaptation model for the human eye is considered [51]. A low-pass filter is applied to frame histograms to avoid sudden DR changes.

Theoretically, iteration should be employed because the ideal map cannot be reached in one step. However, it is abandoned in the proposed method for efficiency. A low computation complexity derives from non-iterative operation, fixed ceilings, and the global operator. This makes our method suitable for real-time processing with a nonlinear high-DR image sensor. Similar with FPN correction, tests were done in Matlab and C++ with a prototype imaging system. Only high frame-rate processing in C++ is suitable to test

the luminance adaptation model in real time. The noiseless tone mapping works well in Matlab and C++.

5.1.3 Fixed-Point Design

In the thesis, the proposed methods were developed at first based on floating-point operations. Floating-point operations are easy to program in high-level computer languages, such as Matlab and C++. Moreover, double-precision floating-point operations provide high precision and range so the wordlength limits may be ignored. However, the goal of this thesis is to provide a real-time processing solution for nonlinear video cameras. The frame rate is a key parameter that must be high enough to ensure fluid video. In this case, fixed-point operations are preferred. Moreover, fixed-point operations offer high power efficiency because of simplicity. Although fixed-point operations bring inevitable errors, precision loss can be minimized and the wordlength can be optimized through careful design. The optimum design considers both precision and wordlength demands.

Neither the fixed-point design of FPN correction nor tone mapping have been widely investigated. For FPN correction, most previous methods still focus on improving the off-line performance. Schneider reported a corresponding fixed-point design for her FPN correction method [8]. The fixed-point design is straightforward because she employed piecewise linear functions to approximate nonlinear functions. Similarly, existing tone mapping methods are used for offline processing. Lack of strict time and power constraints has made offline fixed-point design unnecessary. Wang *et al.* designed a tone-mapping processor for a global tone-mapping algorithm called photographic [31], but the work did not model noise so it may not be suitable for nonlinear high-DR image sensors.

In this thesis, we focused on providing a method to determine an optimum fixed-point design instead of just providing a fixed-point design for a specific image sensor. For FPN correction, we modeled two types of errors: static and dynamic. This guided us to appropriate coefficient scaling and fixed-point calculation. The code we programmed in Matlab can compute the residual error versus wordlength, which enables us to determine the optimum design. The method corresponds to our FPN correction, as applied to any suitable image sensor. So the code can be used for any nonlinear image sensor that employs the proposed method to do FPN correction. In noiseless tone mapping, dynamic LUTs are utilized to store the mapping function and histogram. The LUTs are updated each frame through fixed-point operation. During noiseless tone mapping, some operations, such as histogram counting and histogram modification based on ceilings, were already fixed point. However, we replaced division with fixed-point multiplication and employed a fixed-point LPF.

The tests for fixed-point design were done in Matlab and C++. After implementing fixed-point design in a pre-existing Visual C++ framework, the frame rate increased by 11%. The fixed-point design replaces the floating-point design for higher computational efficiency, implying lower power consumption. Additionally, it also makes the design feasible for an FPGA, which will be discussed in future work. The optimum design keeps the precision while improving the storage efficiency. The real-time DSP required by

nonlinear image sensors benefits from both the low complexity algorithms and optimum fixed-point design.

5.2 Future Work

In this thesis, a complete real-time DSP solution, including FPN correction, noiseless tone mapping, and corresponding fixed-point design has been proposed. Nevertheless, we could extend the thesis work in different ways. The performance of the proposed FPN correction has been tested. However, the factor of temperature, which affects the FPN, still needs to be investigated. In the thesis, all the tests have been done in a prototype digital camera with APS technology and a board-level ADC. Performance of the method for a DPS imaging system needs to be tested. Although noiseless tone mapping and corresponding fixed-point design has been provided, the optimum computation structure for noiseless tone mapping needs to be considered before implementation in an FPGA. Finally, the potential application of this work to invisible-band imaging system is discussed.

5.2.1 Fixed Pattern Noise and Temperature

Digital cameras can be used in different environments where temperature may vary greatly. The FPN in logarithmic CMOS image sensors depends on temperature [3]. In the thesis, we did not consider temperature effects in the proposed FPN correction algorithm. However, temperature affects the performance of FPN correction.

Schneider considered the temperature influence and proposed a correction algorithm [8]. Her method was developed from the three-parameter logarithmic-response model [5] and she modeled the parameters varying as a linear function of temperature with an offset at $0^{\circ}\mathrm{C}$, through empirical observation. Although the performance of algorithm was reported to be good over a typical temperature domain, the calibration in the method does not have a clear statistical basis [66]. Moreover, the temperature measurement required for correction brings difficulty and errors.

Joseph and Collins developed an FPN correction algorithm that includes temperature, which was based on semiconductor physics [66]. Unlike Schneider's method, this method does not need temperature and luminance measurement for calibration and correction. The maximum-likelihood estimation ensures the calibration has a clear statistical basis. Good performance has been shown through experiments.

Unfortunately, both of these methods are model-specific. As mentioned previously, the response model may keep changing with deeper research. Different nonlinear image sensor designs are possible too. So the temperature-aware FPN correction methods need to get rid of the specific model to own high generality, which is similar to the proposed method in the thesis. Additionally, the correction should own both good correction performance and low computation complexity. Although the correction method proposed by Joseph and Collins [66] only needs arithmetic operators, it actually simplified the nonlinear model to a linear model. A method that can correct all variations of the nonlinear model but still use only arithmetic operations needs to be investigated.

5.2.2 Experiments with Digital Pixel Sensors

In theory, the high generality ensures the proposed methods would work for different nonlinear image sensor designs. However, tests were done only with a prototype digital camera system having a logarithmic APS array and commercial board-level ADC. Success with the logarithmic CMOS APS was proved by the experiment results. With the technology developing, DPS arrays are expected to replace APS arrays in future for higher SNR [16]. The feasibility of FPN correction and noiseless tone mapping for DPS arrays still need to be tested through similar experiments, although Mahmoodi [16] began the process.

5.2.3 Noiseless Tone Mapping in an FPGA

In this thesis, noiseless tone mapping is achieved in C++ with a fixed-point design. For higher resolution and higher frame rate, the noiseless tone mapping should be achieved in high-speed hardware like an FPGA instead of a computer. Besides the fixed-point design, design of the timing structure is another important factor that determines the algorithm performance in an FPGA implementation. Different from FPN correction, implementing the noiseless tone mapping in an FPGA is difficult because it needs the whole frame information instead of pixel-wise operation. This means a pixel-level pipeline structure is difficult for the noiseless tone mapping.

The noiseless tone mapping includes two steps. The first step is building a mapping function while the second step is mapping the response to display intensity based on the mapping function. Response mapping is a pixel-wise operation, which can be achieved through a pipeline structure. There are two basic ideas for the timing structure of noiseless tone mapping. The first idea is assumes that the DR will not change sharply from one frame to the next in a fluid video sequence. This assumption becomes more valid after considering the temporal adaptation process in tone mapping. So the previous mapping function can be used for the current frame. In Wang's work [31], a similar approximation was employed. This approximation makes the mapping step in the noiseless tone mapping achievable through a pixel-level pipeline structure, which can shorten the display latency. Another idea does not employ this approximation, which maps the current frame using the previous frame mapping function. Tone mapping for the previous frame and FPN correction for the current frame can be done simultaneously. Latency will be longer with this idea but still shorter than without pipelining. Both the optimum fixed-point design and the timing structure design make the FPN correction and noiseless tone mapping achievable in an FPGA to form a complete embedded system, which includes the image sensor and the DSP.

5.2.4 Low-Dose X-Ray Imaging System

In this thesis, the proposed methods are developed for visible-band logarithmic CMOS image sensors. Moreover, all the tests were done in the visible band. However, the

presented methods can be easily transferred to the invisible band, for example, in medical X-ray imaging.

X-ray imaging for interventional use needs high SNDR, high DR, and fluid video too [67, 68, 69]. Logarithmic CMOS pixels can provide high DR and high frame rate [2], while pixel-level Delta-Sigma ADCs improve SNR [16]. The proposed FPN correction method can reduce the FPN effectively in resultant X-ray image sensors to make SNDR close to SNR, benefiting from high performance and high generality. Noiseless tone mapping is also useful for high-DR scene display. High-DR reproduction with high fidelity brings benefits for patients and medical staff. Combination of these technologies may produce an innovative X-ray imaging system with high SNDR, high DR, and fluid video at lower dosage.

Unlike visible band imaging, X-ray imaging needs a large area [70] in the direct approach [71] and in some indirect approaches [72]. A huge pixel number brings a heavy burden for post-DSP if the frame rate is high. An FPGA-based embedded system is a good approach to overcome the difficulty of high computation. This potential application in the invisible band benefits from the generality and significance of the proposed work, which was demonstrated in the visible band. Future work will focus on modifying the methods to ensure them they are more suitable for a low-dose X-ray imaging system.

References

[1] O. Skorka, *Vertically-Integrated CMOS Technology for Third-Generation Image Sensors*. PhD thesis, University of Alberta, Edmonton, AB, Canada, 2011.

[2] A. Spivak, A. Belenky, A. Fish, and O. Yadid-Pecht, "Wide-Dynamic-Range CMOS Image Sensors–Comparative Performance Analysis," *IEEE Transactions on Electron Devices*, pp. 2446 –2461, Nov. 2009.

[3] D. Joseph, *Modelling and calibration of logarithmic CMOS image sensors*. PhD thesis, University of Oxford, Oxford, United Kingdom, Sept. 2002.

[4] A. El Gamal and H. Eltoukhy, "CMOS Image Sensors," *IEEE Circuits and Devices Magazine*, vol. 21, pp. 6 – 20, May/June 2005.

[5] D. Joseph and S. Collins, "Modeling, Calibration, and Correction of Nonlinear Illumination-Dependent Fixed Pattern Noise in Logarithmic CMOS Image Sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, pp. 996 – 1001, Oct. 2002.

[6] V. Schneider, "Fixed-pattern correction of HDR image sensors," in *PhD Research in Microelectronics and Electronics*, pp. 99 – 102, 2005.

[7] G. Storm, R. Henderson, J. Hurwitz, D. Renshaw, K. Findlater, and M. Purcell, "Extended dynamic range from a combined linear-logarithmic CMOS image sensor," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 2095 – 2106, Sept. 2006.

[8] B. Hoefflinger, ed., *High-Dynamic-Range (HDR) Vision*. Stuttgart: Springer, 2006.

[9] P. Ledda, "High Dynamic Range Displays," *Presence*, vol. 16, pp. 119 – 122, Jan. 2007.

[10] G. Larson, H. Rushmeier, and C. Piatko, "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 291 – 306, Oct. 1997.

[11] S. O. Otim, *Simplified fixed pattern noise correction & image display for high dynamic range CMOS logarithmic imagers*. PhD thesis, University of Oxford, Oxford, United Kingdom, Oct. 2007.

[12] S. Kempainen, "CMOS Image Sensors: Eclipsing CCDs in Visual Information?," *EDN Europe*, pp. 101 – 102, Oct. 1997.

[13] M. Bigas, E. Cabruja, J. Forest, and J. Salvi, "Review of CMOS image sensors," *Microelectronics Journal*, vol. 37, pp. 433 – 451, May 2006.

[14] H. S. P. Wong, "CMOS Image Sensors - Recent Advances and Device Scaling Considerations," in *IEEE Electron Devices Meeting*, pp. 201 –204, Dec. 1997.

[15] O. Skorka and D. Joseph, "Toward a digital camera to rival the human eye," *Journal of Electronic Imaging*, vol. 20, pp. 033009 1–18, Aug. 2011.

[16] A. Mahmoodi, *Low-Area Low-Power Delta-Sigma Column and Pixel Sensors*. PhD thesis, University of Alberta, Edmonton, AB, Canada, Jan. 2011.

[17] Y. Chae, J. Cheon, S. Lim, D. Lee, M. Kwon, K. Yoo, W. Jung, D.-H. Lee, S. Ham, and G. Han, "A 2.1Mpixel 120 frame/s CMOS Image Sensor with Column-Parallel ADC Architecture," in *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 394 – 395, 2010.

[18] 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*, vol. 35, pp. 1146 – 1152, Aug. 2000.

[19] E. Labonne, G. Sicard, and M. Renaudin, "An on-pixel FPN reduction method for a high dynamic range CMOS imager," in *Proceedings of the 33rd European Solid-State Circuits Conference*, pp. 332 – 335, 2007.

[20] K. Hara, H. Kubo, M. Kimura, F. Murao, and S. Komori, "A linear-logarithmic CMOS sensor with offset calibration using an injected charge signal," in *IEEE International Solid-State Circuits Conference*, vol. 1, pp. 354 – 603, 2005.

[21] S. Otim, B. Choubey, D. Joseph, and S. Collins, "Characterization and Simple Fixed Pattern Noise Correction in Wide Dynamic Range Logarithmic Imagers," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, pp. 1910 – 1916, Oct. 2007.

[22] 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*, vol. 23, no. 3, pp. 760 – 768, 2004.

[23] J. Zhai and J. Llach, "Non-uniform backlighting computation for high dynamic range displays," in *IEEE International Conference on Image Processing Proceedings*, pp. 4005 – 4008, 2009.

[24] R. Fattal, D. Lischinski, and M. Werman, "Gradient Domain High Dynamic Range Compression," *ACM Transactions on Graphics*, vol. 21, pp. 249 – 256, Jul. 2002.

[25] F. Durand and J. Dorsey, "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images," *ACM Transactions on Graphics*, vol. 21, pp. 257 – 266, Jul. 2002.

[26] S. Kuo, B. Lee, and W. Tian, *Real-Time Digital Signal Processing: Implementations and Applications*. John Wiley, second ed., 2006.

[27] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "Computer vision system for in-house video surveillance," in *IEEE Proceedings-Vision, Image and Signal Processing*, vol. 152, pp. 242 – 249, Apr. 2005.

[28] D. Ponsa, R. Benavente, F. Lumbreras, J. Martinez, and X. Roca, "Quality control of safety belts by machine vision inspection for real-time production," *Optical Engineering*, vol. 42, pp. 1114 – 1120, Apr. 2003.

[29] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys, "Interactive Time-Dependent Tone Mapping Using Programmable Graphics Hardware," in *Eurographics Symposium on Rendering. 14th Eurographics Workshop on Rendering*, pp. 26 – 37, 2003.

[30] F. Hassan and J. E. Carletta, "A real-time FPGA-based architecture for a Reinhard-like tone mapping operator," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pp. 65–71, 2007.

[31] 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 *IEEE International Conference on Imaging Processing*, vol. 6, 2007.

[32] T. H. Wang, W. M. Ke, D. C. Zwao, F. C. Chen, and C. T. Chiu, "Block-Based Gradient Domain High Dynamic Range Compression Design for Real-Time Applications," pp. 561 – 564, 2007.

[33] O. Skorka, "Hybrid Image Sensors with High Dynamic Range," candidacy exam report, University of Alberta, Edmonton, AB, Canada, 2007.

[34] Bitwise SYSTEMS, *QuickUSB User Guide*, Apr. 2007.

[35] S. Lim and A. El Gamal, "Gain Fixed Pattern Noise Correction via Optical Flow," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, pp. 779 – 786, Apr. 2004.

[36] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. United States of America: Pearson Education, third ed., 2008.

[37] S. C. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists*. Boston: McGraw-Hill Higher Education, second ed., 2008.

[38] R. L. Burden and J. D. Faires, *Numerical Analysis*. Boston: Cengage Learning, 2011.

[39] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes, "Proposal for a standard default color space for the Internet - sRGB," 1996.

[40] R. Scheaffer, M. Mulekar, and J. McClave, *Probability and Statistics for Engineers*. Florida: Cengage Learning, 2011.

[41] B. Weiss, "Fast Median and Bilateral Filtering," *ACM Transactions on Graphics*, pp. 519–526, 2006.

[42] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in *International Conference on Computer Vision*, pp. 839 – 846, 1998.

[43] D. J. Jobson, Z. Rahman, and G. A. Woodell, "A Multiscale Retinex for Bridging the Gap Between Color Images and the Human Observation of Scenes," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 965–976, 1997.

[44] S. Pattanaik, J. Ferwerda, M. Fairchild, and D. Greenberg, "A multiscale model of adaptation and spatial vision for realistic image display," in *Computer Graphics Proceedings*, pp. 287 – 298, 1998.

[45] Q. Shan, J. Jia, and M. Brown, "Globally Optimized Linear Windowed Tone Mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 663 – 675, Jul. 2010.

[46] J. Tumblin and H. Rushmeier, "Tone Reproduction for Realistic Images," *IEEE Computer Graphics and Application*, vol. 13, pp. 42 –48, Nov. 1993.

[47] F. Drago, K. Myszkowski, T. Annen, and N. Chiba, "Adaptive Logarithmic Mapping for Displaying High Contrast Scenes," in *Computer Graphics Forum*, vol. 22, pp. 419 – 426, Sept. 2003.

[48] W. Li, S. Zhang, and M. He, "An Optimal Tone Mapping Algorithm for the Display of High Dynamic Range Images," in *International Conference on Information Engineering and Computer Science*, pp. 1–5, 2009.

[49] L. Coria and P. Nasiopoulos, "Using Temporal Correlation for Fast and High-detailed Video Tone Mapping," in *IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1–4, 2010.

[50] C. Lee and C.-S. Kim, "Gradient Domain Tone Mapping of High Dynamic Range Videos," in *IEEE International Conference on Image Processing*, pp. 461 – 464, 2007.

[51] F. Durand and J. Dorsey, "Interactive Tone Mapping," in *Proceedings of the Eurographics Workshop Rendering Techniques*, pp. 219 – 231, 2000.

[52] G. Ward, "Computer Graphics Renderings, Bathroom: Nighttime (mir) after simulation." http://www.anyhere.com/gward/pixformat/tiffluvrend.html, 1997.

[53] G. Larson, "LogLuv Encoding for Full Gamut, Hgh-Dynamic Range Images," *Journal of Graphics Tools*, vol. 3, pp. 15 – 31, 1998.

[54] E. Culurciello, R. Etienne-Cummings, and K. Boahen, "A Biomorphic Digital Image Sensor," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 281 – 294, Feb. 2003.

[55] X. Wang, W. Wong, and R. Hornsey, "A High Dynamic Range CMOS Image Sensor with Inpixel Light-to-Frequency Conversion," *IEEE Transactions on Electron Devices*, vol. 53, Dec. 2006.

[56] D. Stoppa, M. Vatteroni, D. Covi, A. Baschirotto, A. Sartori, and A. Simoni, "A 120-dB Dynamic Range CMOS Image Sensor With Programmable Power Responsivity," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 1555 – 1563, Jul. 2007.

[57] A. Bermak and Y.-F. Yung, "A DPS Array with Programmable Resolution and Reconfigurable Conversion Time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 15 – 22, Jan. 2006.

[58] O. Yadid-Pecht and E. Fossum, "Wide Intrascene Dynamic Range CMOS APS Using Dual Sampling," *IEEE Transactions on Electron Devices*, vol. 44, pp. 1721 – 1723, Oct. 1997.

[59] D. Yang, A. Gamal, B. Fowler, and H. Tian, "A 640×512 CMOS Image Sensor with Ultrawide Dynamic Range Floating-Point Pixel-Level ADC," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1821–1834, Dec. 1999.

[60] A. Belenky, A. Fish, A. Spivak, and O. Yadid-Pecht, "Global Shutter CMOS Image Sensor with Wide Dynamic Range," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 54, pp. 1032 – 1036, Dec. 2007.

[61] P. Debevec and J. Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," in *Computer Graphics Proceedings*, pp. 369 – 378, 1997.

[62] O. Skorka, J. Li, K. Ranaweera, and D. Joseph, "Canadian Vertically-Integrated CMOS Image Sensors," in *TEXPO, CMC Annual Symposium*, 2010.

[63] O. Skorka, J. Li, A. Harrison, M. Alexiuk, and D. Joseph, "Design of a low-dose X-ray imaging system using vertically-integrated CMOS circuits," tech. rep., University of Alberta and IMRIS, 2011.

[64] L.-W. Lai and Y.-C. King, "A Novel Logarithmic Response CMOS Image Sensor With High Output Voltage Swing and In-pixel Fixed Pattern Noise Reduction," in *IEEE Asia-Pacific Conference on ASIC Proceedings*, pp. 105 – 108, 2002.

[65] J. Tumblin and G. Turk, "LCIS: A Boundary Hierarchy for Detail-Preserving Contrast Reduction," in *Computer Graphics Proceedings*, pp. 83 – 90, 1999.

[66] D. Joseph and S. Collins, "Temperature Dependence of Fixed Pattern Noise in Logarithmic CMOS Image Sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, pp. 2503 –2511, Aug. 2009.

[67] M. Izadi and K. Karim, "High Dynamic Range Pixel Architecture for Advanced Diagnostic Medical X-Ray Imaging Applications," *Journal of Vacuum Science & Technology A (Vacuum, Surfaces, and Films)*, vol. 24, pp. 846 – 849, May 2006.

[68] S. Boyce, A. Chawla, and E. Samei, "Physical evaluation of a high frame rate, extended dynamic range flat panel detector for real-time cone beam computed tomography applications," in *Proceedings of the SPIE*, vol. 5745, pp. 591 –599, Apr. 2005.

[69] W.-C. Cheng and A. Badano, "A Gaze-contingent High-dynamic Range Display for Medical Imaging Applications," in *Proceedings of the SPIE*, vol. 7627, pp. 76270A 1–6, 2010.

[70] U. Rapp-Bernhardt, F. Roehl, R. Gibbs, H. Schmidl, U. Krause, and T. Bernhardt, "Flat-Panel X-ray Detector Based on Amorphous Silicon versus Asymmetric Screen-Film System: Phantom Study of Dose Reduction and Depiction of Simulated Findings," *Radiology*, vol. 227, pp. 484 – 492, Mar. 2003.

[71] M. Yaffe and J. Rowlands, "X-ray detectors for digital radiography," *Physics in Medicine and Biology*, vol. 42, pp. 1 – 39, Jan. 1997.

[72] S. K. Heo, S. K. Park, S. H. Hwang, D. A. Im, J. Kosonen, T. W. Kim, S. Yun, and H. K. Kim, "Development of a large-area CMOS-based detector for real-time x-ray imaging," in *Proceedings of the SPIE*, vol. 7622, pp. 76223T 1–10, 2010.