HDR Image Composition From Multiple Exposures on the Cell Processor

by

Thomas Heade, BSc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science

University of Dublin, Trinity College

September 2009

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Thomas Heade

September 8, 2009

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Thomas Heade

September 8, 2009

Acknowledgments

Thanks to my friends and family for all their support during the year. I'd also like to thank my supervisor Michael Manzke for his help and guidance in completing this project.

Thomas Heade

University of Dublin, Trinity College
September 2009

HDR Image Composition From Multiple Exposures on the Cell Processor

Thomas Heade
University of Dublin, Trinity College, 2009

Supervisor: Michael Manzke

High dynamic Range Imaging (HDRI) techniques have existed in one form or another for over one hundred years as photographic tools to aid in the creation of realistic, scene referred images. Since the advent of ubiquitous digital imaging, HDRI has become a wide ranging and continually evolving field, presenting many opportunities for interesting research.

This body of work seeks to examine and evaluate the applicability of the Cell Processor for HDRI while exploring some of the main concepts in this field. Techniques in the areas of Image Alignment, Response Curve Recovery, HDR Image Composition and Tonemapping, proposed by key figures in the field, are explored in depth with particular focus given to both Image Composition and Tonemapping for the Cell Processor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

High dynamic range imaging (HDRI) is a set of techniques which allow for the representation of a greater range of luminance values that which is afforded with traditional image reproduction and scene rendering methods.

This high dynamic range of luminance enables images produced using HDRI techniques to more accurately represent the luminance ranges found in the real world.

Currently, digital images are generally encoded in formats which specify one byte per colour channel (RGB) and three bytes per pixel, allowing for a maximum of 1.6 million unique colours at 24-bits per pixel. This equates to just 256 different values per channel which is inadequate for the representation of luminance ranges present in the greater majority of real world scenes. The dynamic range afforded by current image capture and display technologies is limited to just two orders of magnitude dynamic range, compared with the ten orders of magnitude range from darkness to brightness present in the real world.

The problem is that this lossy compression of real world luminance ranges to the limited range displayable happens directly following image capture. This means that much of the luminance data available in the scene is discarded. This results in images which can only accurately represent real world luminance for a portion of the image, with the remaining areas appearing overexposed.

This gross loss of luminance data during the imaging process quickly becomes a troublesome issue to anyone seeking to capture accurate scene referred images.

In contrast to regular digital images (LDR images), HDR images reflect the true luminance range present at the point of capture. As such they are described as scene referred images. With the full luminance data available, a user, rendering device, or image processing application is afforded the choice on how to represent the HDR data for a particular purpose on a particular display device (whether it be a future HDR display device, LCD, OLED, CRT or even paper).

The advent of the Cell as an emergent digital media processing platform [10] presents this opportunity to examine the applicability of Cell Processor programming paradigms in the HDRI field. Computationally the processes involved in creating HDR images are slow and, apart from a few notable exceptions [34][33], must be performed by the end user via an image editing package [22][32][11] on a standard PC. This limits the proliferation of HDR technologies to the majority of end users, notably the amateur photographer.

The focus of this body of work is the development and implementation of a HDR image composition based technique to produce tonemapped HDR images in such a way as to take advantage of the massive parallel processing power afforded by the Cell Processor. As the Cell continues to evolve and more Cell enabled consumer devices reach the marketplace, it is not a wild assumption to foresee the advent of Cell enabled digital capture devices, capable of producing full HDR images on device without the need for additional hardware.

In the following chapters the full range of processes involved in creating HDR images from image sets of differently exposed images, beginning with initial photographic image capture to the final tonemapping operations for LDR display, are presented in detail, along with associated implementations, complete evaluation and results.

1.1 Overview

Chapter two of this dissertation covers the most pertinent areas relating to the field of HDRI research, with both seminal and recent papers in each area presented and discussed. The Cell Processor is examined and a detailed overview presented.

Chapter three details the main body of work; design and implementation. The Image alignment, photographic response curve recovery, image composition, and tonemapping methods presented in the preceding chapter are described at an algorithmic and mathematical level, with several modifications and enhancements presented. For both the image composition and tonemapping processes an implementation designed to take advantage of the Cell Processor is presented along with a traditional x86 implementation, this forms the basis for the comparisons presented in the fourth chapter.

Chapter four concerns performance analysis and comparisons between The Cell and x86 based HDRI for different sets of image input data. Additionally pertinent results from the methods examined in the previous chapter are included.

Finally chapter five presents the conclusions brought about by the analysis of HDRI methods on the Cell Processor, including discussions on the effectiveness of the chosen HDRI algorithms for this project. The dissertation comes to a close with thoughts towards future work involving HDRI and the Cell Processor, extending the research presented.

Chapter 2

State of the Art

In this chapter key topics relating to the current state of the HDRI field are discussed. HDRI consists of many broad interrelated area's but for the purpose of this work the main topics of interest are response curve recovery, image alignment, image composition, and tonemapping.

The paper continues this pattern in later chapters.

2.1 Historical Overview

The roots of high dynamic range image generation techniques lie in the mid 1800's with a technique known as "combination printing"[27]. Generally this technique involved the manual composition of two differently exposed black and white film negatives, one giving the sky proper exposure and one for the remainder of the scene. This extended the luminosity range of the final composited image (see figure 2.1)

What we call HDRI today can be traced back to about 1995 with the publication of a paper by S. Mann and R. W. Picard [17]. In this paper (which extended upon work detailed in a paper Mann presented two years previous [16]) the mathematical theory behind the composition of multiple, differently exposed images of the same scene to produce an image with extended dynamic range, was presented. An outline was also given of a process to recover the response function of the imaging device.

This work formed the basis for Debevec and Malik's seminal HDRI paper presented in 1997 [9]. In this paper Debevec-Malik refined previous HDR image composition implementations by producing a robust method of recovering the response function of an imaging system coupled with a simple and effective image composition method based on a weighting function which gave greater precedence to correctly exposed elements of an image in the final HDR composition.

Since then HDR techniques have progressed and spread to many area's of imaging research. Techniques such as HDR rendering and image based lighting have become accepted as useful tools in the mainstream graphics industry [21]. Meanwhile HDRI processing for photographic applications has continued to evolve, with different approaches being explored in each interrelated field of research.

Presented below is a breakdown of each of the main issues in HDRI for photographic applications with reference to some of the published research in each area.

2.2 Response Curve Recovery

When an image of a scene is captured using some form of mechanical imaging device, whether the capture device is a digital camera, scanning device (in this case the scene is whatever object is lying in the path of the scanners optics) , or even an analog film based camera, the final digital representation of the original scene obtained is not a true measurement of the relative radiance present in the original real world scene. This is due to a number of unknown non-linear mappings which map scene radiance to the final image radiance dictated by the imaging device. These non-linear mappings happen at various stages, from image capture to final digital representation. This means that in order to retrieve the original radiance values of a scene using a digital representation of that scene, the compounded series of non linear mappings need to be somehow resolved and quantified in order to determine an inverse (recovery) function.

For an example of the non linear mappings which take place during image acquisition, we look at the image capture process for a digital camera.

Figure 2.1: River Scene, Camille Silvy. An example of combination printing from the 1860's. Note that both the land and the sky are correctly exposed[28].

First, light enters the lens and hits the charge coupled device (CCD) of the camera for the length of the exposure. This actually produces a mapping proportional to the irradiance of the scene, however, at the point where CCD data is composed into a digital image and written to storage a non-linear mapping is generally applied. This mapping is employed to give a digital image some of the aesthetic characteristics traditionally associated with analog film stock. Namely the characteristic curve of analog film(see figure2.2). Additionally the 12 or 14-bit output from the camera's CDD may be converted to gamma-compressed 8bit for final storage (eg. JPEG).

The most notable and troublesome non-linear mapping from a HDR composition point of view, occurs approaching the saturation point of the capture medium, whether film stock, or CCD. That is, the point at which the greatest possible amount of light that can be absorbed by the medium has been absorbed by that medium and any greater amount will have no effect. In traditional sensitometry terms (see figure 2.2) when referring to photographic film this area approaching the saturation point is known as

the shoulder (due to its shape). When the saturation point is reached any value exceeding this point is mapped to same max value, in 8bit terms (for example, JPEG) this would mean a scene radiance value exactly twice as bright as a neighboring value corresponding to "240" will be mapped to the value "255". This corresponds to over exposure and is clearly an inaccurate representation of the original scene. Significant non-linearity also exists with pixels of low radiance, at the base of the response curve. This area is known as the toe of the curve and corresponds with underexposure. The area between the toe and the shoulder is the most linear area of the curve, corresponding with correctly exposed pixels.

The technique of compositing a series of temporally coherent (not a necessity, but preferable for dynamic scenes) images of the same scene, separated by differing exposure times, encompassing the full range of radiance values present requires some way to relate the pixels of one image to the pixels of another image. This is where the problem of non-linear mapping from scene radiance to pixel value effects HDR image composition. By recovering the response function of the imaging process used to capture the images, a look up table can be created. Using this table, pixel values can be resolved to actual scene radiance facilitating the composition process.

Several techniques exist for recovering the response curve of the imaging process. The Debevec and Malik [9] technique offers a technique which can recover an arbitrary response curve from a series of differently exposed images. Final output consists of a weighted average of the input pixels at different exposures. The recovery technique involves generating an over-determined system of linear equations and solving using a singular value decomposition method (SVD). This technique assumes no relationship between colour channels and must be performed for each colour channel present, producing separate response curves.

An alternative technique proposed by Mitsunaga and Nayar [19] uses a parametric model to recover response data. Contrasting with the Debevec and Malik technique Mitsunaga-Nayar assume a relationship between the three (RGB) response curves which preserves chromaticity. This technique can potential recovery a more accurate response function than the Debevec-Malik technique.

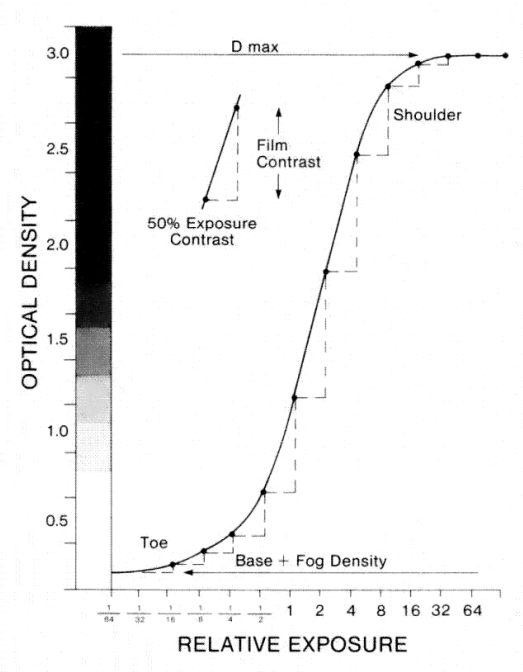

Figure 2.2: A general characteristic film curve[29].

2.2.1 The RAW image format

As an aside, it is of interest to note that the usage of the RAW image format (a format which generally stores unprocessed data directly from the CCD in the CCD's native bit format. Therefore constituting a true representation of the scene as seen by the CCD sensor.) can potentially provide a work around to the problem of non-linear response. Unfortunately this cannot be relied upon as a practical solution for the majority of applications, reasons for which are detailed below:

- File Size: RAW files tend large in comparison with JPEG files due to the amount of data they contain meaning that less images can be captured and stored on a given storage device.

- Write Speed: Large files take longer to write to storage, increasing the period of time between shots. When shooting an image sequence for HDR composition

each image in the series need to be temporally close to its neighbors to avoid changes in the scene which will effect the composition process (changing light conditions, clouds, people moving through a scene).

- Format Standardization: There is no standardized format for RAW. Each manufacturer specifies their own internal format which need to be treated differently from a processing point of view. Formats can even differ between camera models produced by the same manufacture. Certain RAW formats may even include lossy compression or other such alterations to CCD data which eliminate the benefit of using the RAW format from the perspective of avoiding non-linear mappings.

With JPEG practically being the standard for image capture today it seems logical to attempt to make up for its faults rather than use a non-standard format such as RAW, despite potential benefits. Additionally, conversion from an analog process photograph to a digital image (via scanning, or otherwise) cannot benefit from the use of the RAW format.

2.3 Image Alignment

Image alignment is generally required for any HDR image composition technique. This is because misalignment errors are introduced into the image set due to camera movement between each image capture. For this reason it is generally recommended to capture LDR image sets intended for HDR composition with the use of a tripod. However, even with a tripod alignment errors can appear in an image set.

HDR images produced using misaligned image sets generally result in blurry images, with more extreme misalignment causing a clear double image (see figure 2.3). For this reason, a robust HDR image generation technique should include an image alignment operation as a pre-processing step.

Figure 2.3: A comparison between a HDR composition obtained with an unaligned data set and an aligned data set

Image alignment is a very broad field but few alignment techniques have been purposefully developed for HDR applications.

For example, Greg Ward [35] has produced an implementation of an alignment technique which boasts a fast, exposure independent, translational alignment technique based on multilevel image pyramid alignment. This technique offers performance benefits and exposure independence which negates the need for a response function of the imagine process to align images. However, it can only align images via x,y axis adjustments and does not compensate for rotational misalignments.

In comparison Kang et al. [14] has produced an image warping technique using motion estimation for frame interpolation as part of a HDR video implementation. This technique offers a robust alignment technique at a performance trade off.

2.4 Image Composition

In scenes which have a high dynamic range, LDR photography produces images which contain overexposed areas. By varying exposure we can capture the overexposed areas correctly, but the other areas of the image become underexposed.

HDR through image composition of differently exposed LDR images, is based on the idea that by capturing a scene at a range of different exposures, portions of the overall dynamic range can be captured in each image. That is, different parts of the scene will

Figure 2.4: An short exposure set showing different areas of correct exposure

be correctly exposed in different images in the set (see figure 2.4).

By recovering the response function of the imaging process we can resolve pixel values to scene radiances and composite the images to recover a HDR image of the original scene.

Key to this concept is the assumption that each pixel has a correctly exposed counterpart in at least one image in the set.

The majority image composition methods are derivatives of the method introduced by Debevec et al. [9]

2.5 Tonemapping

As discussed previously the dynamic range of luminance in the real world is very high, about 10 orders of dynamic range from darkness to brightness. An every day example of a real world scene with a very high dynamic range is one comprising an outdoor

sunlit area and an indoor area illuminated by an internal light source. Using the techniques described in the previous Image Combination section the full dynamic range of such a scene can be captured, however, the issue of how to display such an image becomes a problem.

Current display devices operate on about two orders of dynamic range, with LCD monitors allowing for a slightly higher range than traditional CRT's. Printed paper meanwhile has an even lower range, somewhere around 50:1. These limitations of traditional display devices necessitate a process to map the recovered real-world HDR values to the limited range of the display device while retaining as much of the detail (or at least an aspect of the detail) from the HDR image as possible.

Tonemapping is such a process, of which there are many, many variations. Two broad categories of tonemapping operators exist; those that use global operators, and those that use local operators. Other categories exist such as gradient domain and frequency domain operators, but global and local operators are the most common. Global operators distinguish themselves by compressing an image based on the use of the same non-linear compression curve to map each pixel, while local operators apply compression according to the luminance values of each individual pixel and the values pixels in the local neighborhood.

Global operators tend to be better for images without extreme variations in dynamic range. Used on such images the tonemapped result can present with blown out highlights in high brightness areas, reducing visibility. However, different global operators produce different results with the majority producing good, aesthetic pleasing results across a vast spectrum of image types.

Local operators are useful in circumstances where an image has extreme variations in dynamic range located through an image, multiple high intensity light sources in a darkened room for example. The application of a local operator based tonemapping algorithm generally results in a very distinctive hyper-real image due to the lack of a global evaluation of the range present. This distinct look is preferred by some photographers for aesthetic reasons and shunned by others for presenting an unrealistic

representation of the original scene. However, these are just opinions based on personal perception and it is important to remember that there is no right, or wrong when it comes to tonmapping operators.

Performance wise global operators tend to be appreciably faster [24] than any other form of tonemapping. This quality means that global operators are the preferable tonemapping method for high performance real time applications. Local operators, although slower, still present generally better results than global operators in scenes with extremely high dynamic range, as well as a unique aesthetic equality. As such global and local tonemapping operators can be considers complimentary algorithms.

2.6 A Brief Overview of The Cell Processor

The Cell Broadband Engine (also commonly known as the Cell Processor, or simply "The Cell"') was developed jointly by a collaboration between Sony, Toshiba and IBM. The Cell is the first processor in what is intended to be a new family of microprocessors conforming to the Cell/B.E. Architecture (CBEA) model which was laid down during the development of the Cell.[23]

Originally conceived by Sony Computer Entertainment inc. with the intention of developing a new CPU architecture as the basis for their then in development, "next-gen", games console - the Playstation 3, the architecture as it exists today is capable of high performance general purpose computation and is thus suitable for a wide range of applications in many fields.

This first generation Cell Processor is a single-chip asymmetric design comprising nine individual processing cores, or elements loosely connected via a shared memory model facilitated by an EIB, or Element Interconnect Bus. Eight of these nine processing cores are Synergistic Processor Elements (SPE's) with the remaining one being a Power Processor Element (PPE). These two elements are distinct in their design and function, being intended for use on different types processing tasks.

The PPE consists of a conventional PowerPC Architecture (RISC design) core ca-

pable of 64-bit and 32-bit computation paired with a 64KB L1 cache and a 512KB L2 cache. [23] In comparison the SPE's are self contained independent single-instruction-multiple-data (SIMD) vector processors , and thus optimized for compute intensive tasks. Each is paired with small 256KB local stores, lacks a cache, and has full access to shared memory via multiple direct memory access (DMA) units.

The PPE is a conventional processor and as such is intended to handle the operating system and schedule tasks for the eight SPE's while the SPE's are designed to perform the computationally intensive tasks. This relationship between the PPE and the SPE's forms the basis for CBEA.

The biggest difference between the PPE and the SPE's is the way in which each accesses memory. The PPE is able to read directly from main memory and store data into available local cache memory as required, just as a regular processor would. The SPE's however access main memory with DMA commands which move data in 1Kb to 16Kb chunks into what is termed local storage. Each SPE's DMA controller can have a maximum of 16 transfers in operation simultaneously. Instruction fetch and load and store operations act on this local store rather than main memory. Additionally the lack of an SPE cache and associated coherency logic removes a level of complexity from a cache management perspective, increasing the rate at which data can be processed.

Effective use of the SPE's local store is key to the performance gains inherit in the Cell design as transfer operations between an SPE's local store and an SPE's registers can be performed at an incredible sustained rate of 64Gigabyte per second, additionally, effective management of DMA transfers between main memory and each SPE's local store by exploiting the bandwidth of the EIB is vital to maintaining a high rate of computational throughput.

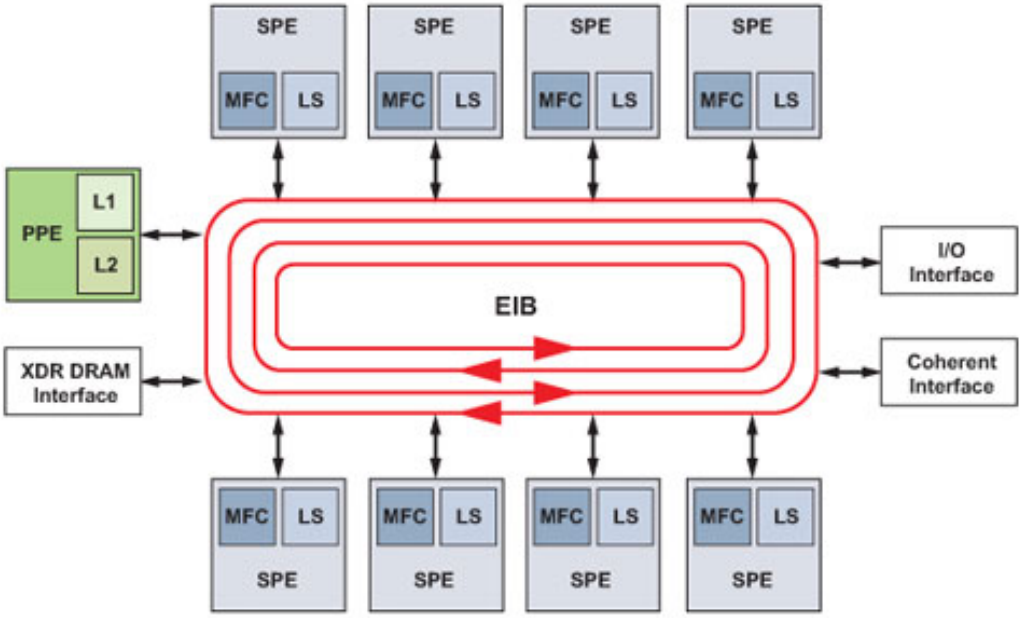

Figure 2.5: An overview diagram of the Cell Processor architecture [20]

Chapter 3

Design and Implementation

This chapter details the design and implementation of a HDRI program encompassing the concepts introduced in the previous chapter. Both Cell Processor and x86 implementations are described where appropriate.

3.1 Design Overview

The over arching design philosophy behind this project was to produce a modular chain of algorithms which took a set of LDR images of different exposures as input and produced tonemapped HDR output (see figure 3.1). This design allowed for a particular algorithm to be swapped out and replaced with another algorithm without affecting the rest of the project, making for a robust experimental HDRI platform.

For each of the HDRI area's selected for implementation a suitable algorithm needed to be sourced. Suitable, both in terms of effectiveness in their related area, and from the point of view of deployment to the Cell Processors SPE's.

At the beginning of the project it was decided that several versions, of what would become the final implementation, would be created and deployed on both the x86 and Cell Processor based platforms. One of the main reasons for this approach was to provide a solid basis for performance comparison between HDRI algorithms on the Cell Processor and HDRI on the x86 platform.

The second reason for this was to allow for initial algorithm prototyping, development, and modification to take place on the familiar x86 platform rather than the more esoteric Cell Processor platform. In effect algorithms were implemented on the x86 platform, analyzed with the intention of deriving a suitable Cell implementation, and finally ported and rewritten to take advantage of the Cell's unique architecture.

This approach saved time in the initial stages of development where various avenues of approach could be explored quickly and rapid prototypes developed in a familiar setting.

Over the course of the project three different code bases were developed and utilized to create the final Cell Processor SPE enabled implementation. Each was developed with a specific need in mind.

Initial algorithm testing and rapid prototyping was conducted in the C++ programming language with the use of the CImg [7] image processing library for both file IO and pixel manipulation operations.

An improved C based HDRI implementation was then developed as a basis for the eventual porting of the code base to the Cell's PPE. CImg was replaced with a very light weight public domain JPEG/PNG reader. This functionally complete x86 C code base was used as the performance measure for HDRI algorithms on the x86.

Porting this C implementation to the Cell PPE was surprisingly simple. Effectively, the only code that needed to be rewritten for the platform was due to conflicting operating systems. The x86 C implementation was developed under Windows, while the Cell enabled platform was Unix based. This necessitated a rewrite of the file IO code to suit Unix directory manipulation.

The final implementation was a combination of the PPE only version and a full SPE implementation of the image composition and tonemapping algorithms. These two area's of HDRI were selected for SPE deployment due to the vital importance they have in

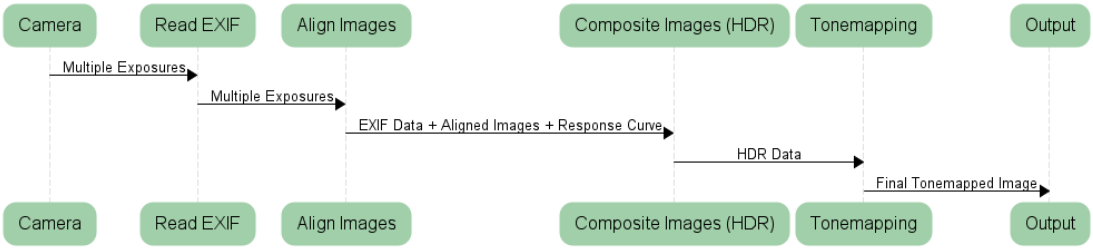

Figure 3.1: Sequence of operation, assuming response curve data has been recovered and saved to file

the HDRI process. For instance response curve recovery is only required on a per device basis. Generally after recovering a response curve the algorithm will not need to be executed again as the initial curve is saved to an external file and queried on future executions. Implementing an SPE based response curve recovery system, while useful, was not seen as vital to the project. Additionally Image Alignment is not required for images which do not possess a significant spacial disparity within an image set. Any image set captured with the use of a tripod will, in most circumstances, not require image alignment. Thus for the majority of inputs to the HDRI algorithm chain, image composition and tonemapping present the greatest computational expenses. Thus it was logical to derive an implementation which offloaded both of these tasks to the Cell SPE's.

To facilitate this SPE implementation a suitable Cell programming paradigm was sought. Studying the structure of the image set input data suggested a technique based on parallel processing of partitioned data sets (See figure 3.3). A pipelined (See figure 3.2) approach was also considered but due to time constraints this model was not implemented.

3.2 Taking Photographs for HDR image composition

From a conceptual standpoint the process of capturing the images for high dynamic range image composition is a simple process. An image set consisting of photographs

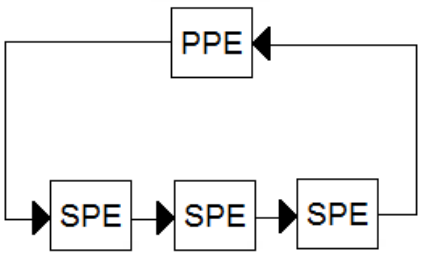

Figure 3.2: A pipelined approach to cell processing.

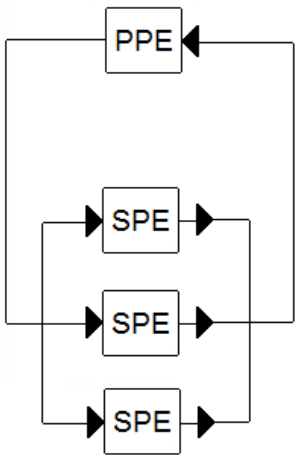

Figure 3.3: A data partitioned sequential processing approach to cell processing.

of the same scene at different exposure values (the more images the better, 10-13 images works well in most cases, but it can work with 2 images and above) taken from a steady camera is all that is required. A tripod is recommended, clamping the camera to a surface works equally well, with a steady hand only as a last resort.

However, capturing this image set can prove difficult given the limited settings afforded by standard "point-and-shoot" consumer level digital cameras. The camera used for the majority of this project was a Panasonic DMC-TZ1. This is a mid-range consumer level digital camera, and as such offers little in the way of the manual control necessary for capturing images for high dynamic range composition. Values such as f/stop [8] and exposure time only allow for very minor adjustments, such as setting the minimum shutter speed (1/8,1/2,1,2,4,8), and altering the Exposure Value (+2 - -2 in 1/3 increments).

As such, the user is mostly at the mercy of the cameras auto-exposure software. Various techniques must be employed both at an algorithmic and image capture level to create viable input image sets.

An example of auto-exposure interfering with the image capture processes presents itself when attempting to purposefully take exposures which are largely over or underexposed, in this case the camera will usually default back to a lower/higher exposure time to prevent the user from taking a "bad" photograph. In these cases it can be useful to use a technique known as Auto Exposure compensation or AE Lock. An example of this would be attempting to take a 1 second exposure in a very bright scene. In this case the camera would detect the overexposure and default back to a lower exposure time. To force the camera to take the shot with the desired exposure time the user can focus the camera on a much darker scene making the camera receptive to the desired exposure time. At this point the user can move the camera back to the scene without refocusing and take the overexposed shot. (More expensive digital SLR's and some higher grade consumer level digital cameras include an AE Lock button enabling the user to use the same forced setting between shots).

However, this can cause the f/stop to change between exposures since this is dictated

mostly by the cameras auto-exposure system. Because we use exposure time as an input for image composition and response curve recovery, it is necessary to normalize exposure values, with respect to f/stop, across the image set.

3.2.1 Preprocessing for Exposure Time Normalization

In order to normalize the exposure values across a range of varying f/stop's the "Law of Reciprocity" must be employed.

The law of Reciprocity refers to the inverse relationship between intensity and duration of light that determines the reaction of light-sensitive material. With regards to photography this refers to the interaction between the aperture size and the shutter speed to determine the volume of light hitting the CCD/film stock, ie. aperture and shutter speed are inversely linear, total exposure = intensity x time. Therefore the same response can result from reducing exposure time and increasing aperture size and vice versa. For example the same exposure time can also be achieved by doubling the aperture to f/2 and halving the shutter speed to 1/250 s or by halving the aperture to f/4.0 and doubling the shutter speed to 1/60 s.

Since the law of reciprocity defines f/stop and exposure time as inversely linear, and we know that f/stop is a logarithmic scale of known factor; exposure times can be calculated given a change in f/stop. eg. If at f/stop f/1.0, exposure time is 1/2000 what is the exposure time at f/stop f/5.6? (or any other f/stop exposure time combination).

The formula for this is as follows.

Shorter exposure time -

$$A = B/2^C$$

Longer exposure time -

$$A = B * 2^C$$

Where A is the new exposure time, B is the original exposure time and C is the f/stop difference.

F/stop difference is calculated as follows.

$$\frac{log(f1)-log(f2)}{log(\sqrt{2})}$$

The normalizing of exposure times is done as a pre-processing step, just after the EXIF data is extracted and the images are loaded into memory. This is followed by a sort operation of the input image set, from brightest image (highest exposure) to darkest image(lowest exposure).

3.3 File Handling and Data Storage

File handling is, of course, integral to the overall HDRI implementation. No HDRI processing can proceed without an input image set.

To facilitate the storage of image data and pertinent variables related to each image, a dynamic array of struct's was created. With each struct containing data relevant to the source image, including pixel values, f/stop, exposure time and original dimensions.

A simple directory crawling method was used to iterate over the input image files for a given directory path. For image input only images with the extension ".JPG" are recognized, other file types and nested directories are ignored. Therefore it is important for the input directories to be free of any superfluous JPEG's unrelated to the current HDRI composition operation.

When valid input is found the pixel data is read from the image into the image list data structure and separated by colour channel. The width, height, exposure time, and f-stop data is then extracted from the EXIF header of the JPEG. EXIF support was implemented with the use of excerpts from the public domain EXIF manipulation tool "jhead" [13].

Initially the CImg library was used for extracting pixel data from images. However, due to a switch to pure C for the final Cell and x86 implementations a change to a C image parsing solution was required. The "stb_image.c" [4] JPEG/PNG reader was

selected for this task due to its lightweight design and minimal dependencies.

File output for tonemapped images was achieved using the same "stb_image" solution, with uncompressed 24-bit Bitmaps selected as the output format. The use of a Bitmap was necessitated due to format limitations of "stb_image", however, the only tangible drawback to this is an increased file size.

3.4 Response Curve Recovery

This section gives an overview of the chosen response curve recovery method followed by implementation details.
The method selected for response curve recovery was detailed by Debevec and Malik[9].

3.4.1 Selected Technique

The Debevec and Malik technique is based on the concept of Photographic Reciprocity. Photographic Reciprocity refers to the inverse relationship between intensity and duration of light that determines the reaction of light-sensitive material.

Specifically this refers to the interaction between the aperture size and the shutter speed to determine the volume of light hitting the CCD/film stock, ie. aperture and shutter speed are inversely linear (CCD charge = light intensity x time).

Therefore the same response can result from reducing exposure time and increasing aperture size (effectively increasing the irradiance at the CCD) and vice versa. For example the same level of total exposure can also be achieved by doubling the aperture to f/2 and halving the shutter speed to 1/250 s or by halving the aperture to f/4.0 and doubling the shutter speed to 1/60 s. The product of intensity*time is the only thing that determines the total exposure, neither value has more influence.

Given this, we can recover the light intensity hitting the CCD at the point of capture for pixel X once we know the composite non-linear function, via:

$$E = X/\Delta t$$

Therefore the goal of this method is derive this find this composite non-linear function and apply its inverse to each pixel value in the 24-bit range, thus enabling the creation of a look up table of original exposures. This inverse function can be recovered and applied because the response curve is monotonic in nature.

Describing the final pixel value as a function of non-linear mappings being applied to scene irradiance values we can write the following:

$$Z_{ij} = f(E_i \Delta t_j)$$

Where Z_{ij} is a pixel value where i is the index of a particular pixel and j is the index of that pixels exposure, f is the non-linear function, E_i is the irradiance value for pixel at index i, and Δt_j is the exposure time at index j.

Since we have assumed that f is a monotonic function it is possible to obtain the inverse the and apply it to retrieve the total exposure of pixel i at exposure j.

$$f^{-1}(Z_{ij}) = E_i \Delta t_j$$

Breaking this down further we can apply the natural logarithm to both sides, giving:

$$lnf^{-1}(Z_{ij}) = lnE_i + ln\Delta t_j.$$

We already know the exposure time as it has been extracted from the JPEG's EXIF header. This allows us to obtaining the original scene irradiance value for pixel i at exposure duration j in isolation.

Given that function $g = lnf^{-1}$, we now have.

$$g(Z_{ij}) = lnE_i - ln\Delta t_j$$

At this point we know everything except the function "g" and the irradiance values E_i. The idea is to find the function g and the irradiances E_i that satisfy the equation above in a least-squared error sense.

Given that we have only a limited number of pixel values present in the encoding

(usually 0-255), we only need to solve for that limited rage in order to create a look up table of scene irradiances.

Therefore, assuming P is the number of photographs we are using as input to this algorithm and N being the number of sample points we have chosen from the image (50 points for 11 images is recommended by the paper) the problem can be refined to one where we need to find the range of pixel values of g() and the N values of lnE_i that minimizes the following quadratic objective function:

$$O = \sum_{i=1}^{N} \sum_{j=1}^{P} [g(Z_{ij} - lnE_i - ln\Delta t_j]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g^{''}(z)^2$$

The first part of the function has already been described above, and ensures that the final solution satisfies the set of equations in the least squares sense. The second part of the function is a smoothing term ensuring that the resulting response curve is smooth and provides coherence between the values in the pixel range. The smoothing term specified by the paper is of the following form.

$$g^{''}(z) = g(z-1) - 2g(z) + g(z+1)$$

Lambda is an input to the smoothing function specifying the amount of smoothing to apply.

Finally, since we know that the response curve will generally have a distinctive "S" shape to it, with a toe, straight and shoulder as discussed earlier. We can introduce a weighting function w(z) to further increase smoothness and fit terms towards the center of the curve which corresponds to correctly exposed pixels (z).

This weighting function suggested by the paper is a simple hat function.

$$w(z) = \lceil Z_{max} - z for z > (Z_{min} + Z_{max})/2$$
$$\lfloor z - Z_{min} for z \leq (Z_{min} + Z_{max})/2$$

Other weighting functions have been proposed in various similar approaches [26]. These alternative weighting functions conform to the shape of the expected response

curve more closely.

This gives the final equation

$$O = \sum_{i=1}^{N}\sum_{j=1}^{P}\{w(Z_{ij})[g(Z_{ij} - lnE_i - ln\Delta t_j]\}^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1}[w(z)g^{''}(z)]^2$$

This overdetermined system of linear equations can then be solved using the mathematical technique of singular value decomposition (SVD) to derive the response values for each pixel in the range. Plotting these values reveals the response curve of the imaging system.

3.4.2 Implementation

Input to this function consisted of set of images separated by exposure time. To ensure that the recovered response curve accurately represented the range of pixel values available (0-255) measures were taken to ensure a good spread of colour values in the input image set.

According to the original paper response curves can be retrieved with as few as two images with no upper limit. After extensive testing the an image set consisting of between four and six images seemed to produce the most accurate response curves. It is important not to use too many input images as this can cause memory problems given that the size system of linear equations is on the order of N x P + P-range. Where N is number of samples and P is the number of images.

The first operation performed as part of the recovery technique is to obtain a list of sample pixel index's from the image list as input to the equation system. Basically, pixel locations need to be chosen which give a good distribution of pixels in the pixel value domain, which are well separated spatially, and are sampled from area's with low internal variance. This processes is fully detailed with alternate implementations in the sampling section.

The linear system of equations is formed using the pixel values determined through the

sampling function across the image set as specified by the method, and the smoothing function applied. This set of linear equations is expressed in matrix form.

At this point the system of equations requires solving using SVD. In the original paper SVD was accomplished through the use of Mathlab. Therefore it was necessary to source a compatible linear algebra library to perform this function.

For the initial C++ implementation the JAMA/TNT [31] linear algebra package was utilized to good effect. However, no C port of this package existed. Some C linear algebra packages were explored as potential solutions but in the end it was decided to rewrite the SVD, and related portions of the JAMA/TNT package for C compatibility.

Through initial use of SVD it was discovered that for the QR decomposition technique used, the default weighting function provided by the original paper was inadequate. This is due to the weighting function returning weights which equal 0 for pixel values of 0 and 255. QR decomposition requires matrices which are full rank [36], for this to hold true, the weighting function must never return zero. The simple solution to this is to shift the weighting function up by 1, so that a pixel value of 0 or 255 produces a weighting of 1.

The SVD function returns an array containing the response function.

This technique is repeated for each colour channel yielding three response curves.

3.4.3 Sampling Techniques

Effective sampling is key to the response curve recovery method outlined above. In the original paper useful sampling points were selected by hand from the input image set. Although automating the process was proposed and some of the criteria for effective sampling were briefly discussed; an implementation was not detailed. To this end, several sampling algorithms were developed and tested for effectiveness in retrieving valid pixel data from an image set. Valid pixel data is assumed to be data which results in the generation of a monotonic response curve similar in appearance to a film response

curve.

The first sampling technique employed involved using random sampling to achieve the specified number of sampling points. This technique has obvious advantages in terms of computation time and code complexity but could not be considered a serious point sampling solution due to its lack of robustness. Points sampled generally lack good spacial distribution and the possibility exists of sampling points in regions which remain over or underexposed for the lifetime of the image set. Additionally points sampled randomly generally do not encompass the full range of intensity values from 0-255. All of these issues contribute to a poor representation of the true response function of a particular camera. Random sampling does produce usable, albeit highly variable, results (see figure 3.4).

A second iteration of this sampling method was developed To improve upon random sampling and address some of the criteria suggested for valid sample points. Random points were selected from the median image of the sorted response curve image set, this makes the task of picking a set of pixels with a wide distribution of intensity values easier because the median image should have less extremely over and underexposed areas. Half the points were selected in the lower half of the luminance range (0-128) and the other half in the upper range (128-255), this ensures that the sampled points cover a wide range of luminance values.

Points were then tested to see if they lay in a spatially homogeneous areas. Points in the immediate surrounding area are tested against the chosen pixel for high levels of variance. This helps to reduce the effects of chromatic aberration and other fringing artifacts as well as CCD sensor noise which commonly presents as randomly dispersed purple hued pixels in isolation.

The rejection criteria for points passing this test checked the individual colour channels for high levels of internal variance, pixels with a high internal variance (by default a value of 10 was used) were rejected. This was done with the intention of preventing any particular colour channel exuding an unbalancing effect on the final response curve.

Finally, in order to prevent inordinately long sample times and the possibility of entering into an infinite loop when sampling images which have few pixels which meet the outlined criteria, a contingency was introduced which increased the tolerance of the internal variance test after a given number of rejections.

Results for this sampling method were improved over the simple random sampling method and gave uniformly similar results over multiple runs (see figure 3.5). However, some concerns were raised regarding the validity of testing for pixels with low levels of internal channel variance, effectively sampling grey scale pixels, might cause a misrepresentation of the true response of a particular colour channel.

To address this concern One final sampling variation was tested. For this sampling variant the internal variance test for the sampled pixel was removed and fully one third of the requested number of samples was reserved for each colour channel with each half of this third for low luminance values and the other half for high luminance (as detailed above). This effectively means each component of each sampled pixel is treated as an individual sample independent of sibling components. For example a pixel presenting with RGB values of (100,50,0) would constitute a valid sample for the red and green channels but the blue component would be rejected for extreme underexposure. This increases the acceptance rate of selected pixels.

Results for this final sampling method were very similar to the previous sampling method. Resulting response curves did not appear to be any more or less accurate and the tonemapped HDR images composed with these RGB curves did not present any major colour imbalance (see figure 3.6).

The final two sampling methods outlined are both valid techniques for sampling pixel values for the Debevec and Malik recovery technique. Both methods are superior to purely random untested sampling in terms of resulting response curves, consistency, and robustness.

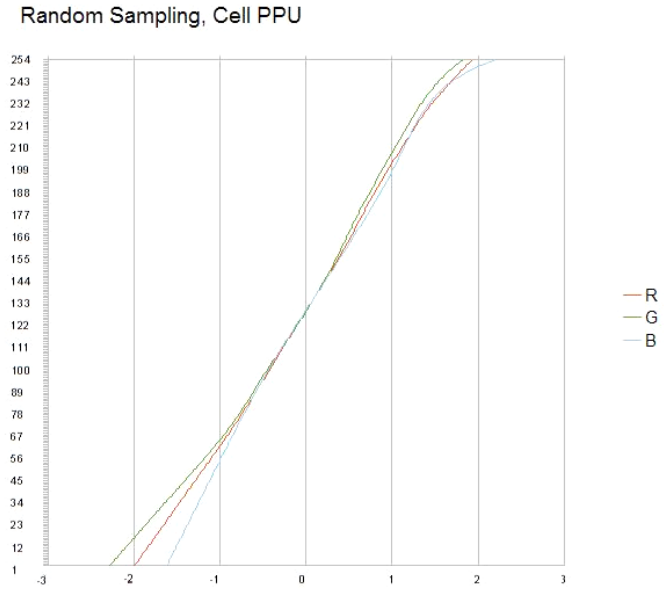

Figure 3.4: Random sampling method. Pixel value vs luminance

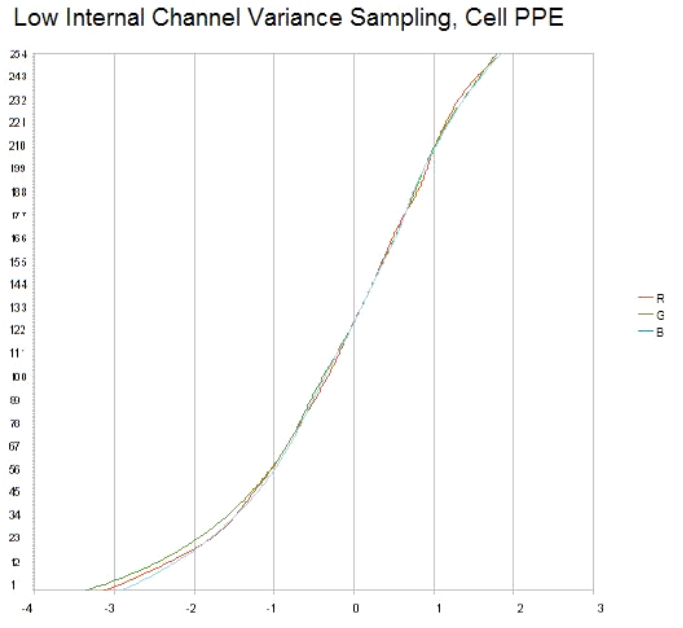

Figure 3.5: Sampling with regard to internal channel variance. Pixel value vs luminance

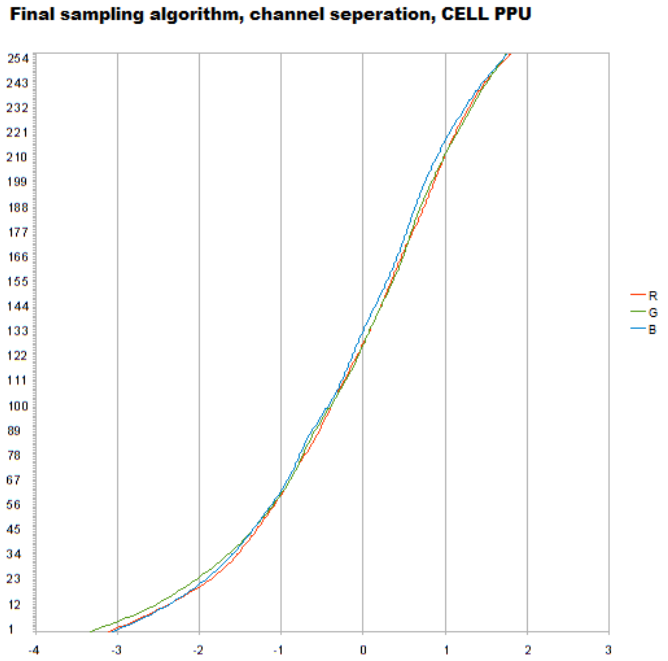

Figure 3.6: Points sampled at sub-pixel level. Each curve is sampled individually. Pixel value vs luminance

3.5 Image Alignment

In many cases input image sets are misaligned, generally resulting from camera movement at the point of image capture. For this reason it is generally recommended to use a tripod, however, in situations where a tripod is not available or an image set becomes significantly misaligned despite precautions image alignment becomes necessary. This section gives an overview of the chosen image alignment method.

The method selected for this operation was detailed by Greg Ward [35]

3.5.1 Median Threshold Bitmap Alignment

The median threshold bitmap (MTB) alignment technique was developed specifically to facilitate the composition of high dynamic range photographs from hand held LDR exposures. This was one of the main reasons this technique was selected.

The MTB technique is based on the creation of threshold bitmaps to act as an alignment proxy, reducing the computational complexity of the alignment operation. These bitmaps are aligned horizontally and vertically with the use of an image pyramid. The threshold bitmap also theoretically avoids the problems associated with aligning images of different exposure levels. This is a point of note because traditional edge detection algorithms don't respond well when applied to image sets consisting of differently exposed images (as found in the HDR composition process). This is because the resulting edge detection bitmaps vary heavily with exposure. Additionally, this exposure invariance negates the need for an exposure normalization technique necessitating the use of the response curve of the imaging process. This is important to note because the input to the response curve recovery step consists of an aligned image set differing by exposure time. Without prior knowledge of the response function, aligning this image set would pose major difficulty.

The first step in the algorithm is covert each image in the input image set to 8bit greyscale images through the application of the following formula

```
grey =  0.2125 * red + 0.7154 * green + 0.0721 * blue
```

The paper then suggests selecting an arbitrary image from the image set as the reference image to form the basis by which every other image in the set will be aligned according to. However, to increase the robustness of the algorithm an alternative technique is to select the center image in the image list corresponding to a moderately exposed image. This prevents the use of massively over or underexposed images which can exist in the input image set. These images are unsuitable as the basis for alignment because they contain very little usable pixel data.

The titular median threshold bitmap is constructed from the greyscale input images by creating a 1-bpp bitmap where 0's represent pixels from the input image which are less than the median grey value present in the image and 1's represent pixel which are greater than the median value. This technique produces exposure invariant bitmaps. By computing the XOR between the MTB of the image to be aligned and the reference image shows the alignment difference between the two images.

The alignment approach proposed by the paper involves the use of an image pyramid [3]. Here the grey scale approximation for the reference image and the current input image are sub-sampled by a factor or two for each level of the image pyramid. The number of levels in the pyramid corresponds to the base two log of the maximum alignment offset. So for a maximum offset of 64 pixels the image pyramid for each image will have 6 levels.

Starting from the top of the pyramid (smallest image subsample) the threshold bitmap is computed for both images and the minimum difference offset between the input image and the comparison image is calculated within plus or minus 1 pixel in each dimension. Moving down the pyramid the previous offset is doubled to account for the sub-sampling operation and the offset is computed on the new level with respect to the previous levels offset. This continues to the bottom of the pyramid. At this point the final offset has been computed, here it can be noted that each level of the pyramid corresponds to a single binary bit, from least significant to most significant, in the overall offset. This is because, as stated previously, the number of levels in the pyramid is calculated via base two log.

To increase the robustness of the approach the paper proposes an additional operation to reduce the amount of noise present in the threshold bitmaps. This noise is introduced by the computation of the threshold bitmap where there are a large number of values close to the median value.

The approach taken in this step is quite straight forward and involves the creation of an additional "exclusion bitmap". This bitmap is specified in the same way as the threshold bitmap except that all values within a specified distance of the given threshold are represented by 0's and all other values consist of 1's. This exclusion bitmap is calculated for each image at each level on the pyramid. The bitmap is used by taking the XOR difference for the current offset calculation and AND'ing it with both offset exclusion bitmaps. This has the effect of removing the pixels which are classified as noise by the threshold value thus eliminating the potential problem of local minima during the alignment process.

3.5.2 Implementation

The MTB alignment technique was implemented as a recursive algorithm which took an image to be aligned and a reference image as input and returned the integer amount of x,y shift required to align the input image.

Each level of recursion in the algorithm corresponds with a level of the image pyramid. Starting from the base of the pyramid, the algorithm recursively subsamples the input image and the reference image until the base of the pyramid (specified as an input parameter which controls maximal shift) is reached.

At this point the threshold and exclusions bitmaps are calculated. The threshold value is calculated by taking the average luminance value from a computed greyscale of the input images. While the exclusion bitmap is calculated using a tolerance to this threshold value. The tolerance value suggested by the paper is 4, it is not mentioned how this value was arrived at so it is assumed that it is empirically derived.

The alignment operation then takes place by searching for the best alignment match

between the -1/+1. For each possible alignment, a copy of the threshold and exclusions bitmap are shifted by that amount and the logical operations applied to remove the noise (as specified by the exclusion bitmap) in threshold bitmap copy. The shift which gives the best match to the reference input image is selected.

On the next level up the current shift value is scaled up by a factor of 2 and the operation continues.

This operation is applied for each image in the image set with the median image in the set selected as the reference image.

3.6 HDR Image Composition

Image composition is the heart of any HDRI implementation, as it is the process by which LDR image data is processed to create a HDR image.

At this point in the algorithm chain we have an aligned image set and the relevant response information recovered via the previous step. These are the inputs required for this image composition step. Our final expect output is a series of float point HDR data arrays, one for each colour channel (RGB).

3.6.1 Selected Technique

The composition method selected for implementation was proposed by Debevec and Malik[9].

This method was introduced in the same paper as the response curve recovery technique detailed earlier in this chapter. As such it shares some similarities carry on from that method, such as the use of the same weighting function.

The general idea behind the technique is one of converting LDR pixel data across the image set into weighted relative radiance values through the use of the recovered response curve data. This radiance data is then composited an average obtained. This

in essence is HDR composition.

This is expressed in an algorithmic sense as follows,

$$lnE_i = \frac{\sum_{j=1}^{P} w(Z_{ij})(g(Z_{ij}) - ln\Delta t_j)}{\sum_{j=1}^{P} w(Z_{ij})}$$

Where lnEi is recovered as follows,

$$lnE_i = g(Z_{ij}) - ln\Delta t_j$$

As discussed in the response recovery section, this is possible because we know the exposure time and the response function "g".

For the composition process the weighting function from the response recovery method is reused. This weights pixel values closer to the linear section of the curve higher than pixels in the toe and shoulder regions. This means that pixels of nominal exposure are weighted higher than pixels which are over or under exposed, therefore these errant pixel values have less influence over the final HDR output.

Across each image in the image set, the weighted sum of pixels which share the same spatial location is calculated and divided by the sum of the pixel weights. This gives the composite HDR value for this pixel. This process is performed for each sub-pixel value of every pixel in each image. The result is a set of HDR data arrays representing each colour channel.

3.6.2 x86 Implementation

The x86 implementation closely followed the algorithm as described. However, two issues were encountered. Firstly, examination of the final tonemapped image (processes described in the following section) revealed oddly coloured pixels appearing in sections of overexposure (see figure 3.7). These area's corresponded to locations of extreme exposure in the LDR image set, such as the center of the sun, or glare bouncing from a reflective surface. This issue was quite perplexing and remained unsolved until quite late in development. This stemmed from an inability to resolve whether the actual composition method was causing the problem, as the HDR data produced by

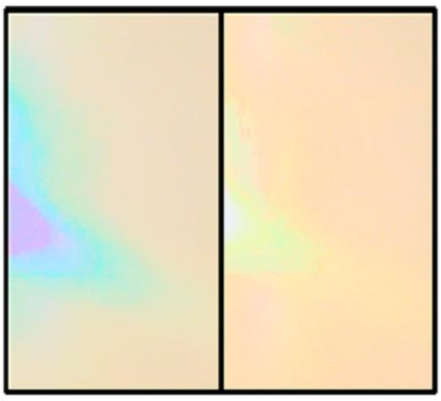

Figure 3.7: Composition error due to weighting method.

the method cannot be viewed without tonemapping (which could have been causing problem).

Eventually the problem was traced back to the weighting function, which is used by the both response curve recovery and the HDR composition method. As explained in the response recovery section the weighting method required modification, amounting to adding a 1 to the result, in order to create a full rank matrix for SVD. Since this weighting method is the same as that used in the HDR image composition process, extreme over and underexposed values were being assigned a weight of 1 where they should have been assigned a 0. This meant that these pixels were being included in the final image where they should have been ignored. This is what caused the appearance of the incorrectly coloured pixels. Creating a weighting function solely for image composition solved the issue.

The second problem arose when compositing a HDR image which contained a section of overexposed pixels in each image section corresponding to the sun. This in effect means that there was no data available through the use of the method as described to represent this area as the HDR composition technique relies on the assumption that every over and under exposed section has a properly exposed representation in at least one image in the exposure set. This problem was solved quite easily by checking at

the end of the composition for HDR pixel values equal to 0. These pixels were then assigned the value of their pixel counterpart in the last image in the ordered image set (the darkest image).

3.6.3 Cell Processor Implementation

The cell Processor implementation of this algorithm was substantially more involved than its x86 counterpart.

Initial work to facilitate this Cell Processor implementation involved ensuring that each variable and data structure which would eventually be processed by the Cell's SPE's was aligned along the 128 byte boundary to maximize the performance of data transfer [23][6] and avoid potential bus errors.

This was accomplished through the use of the __attribute__ (aligned()) function

This alignment operation facilitates optimal DMA because the MFC (Memory Flow Controller) contains five cache lines of 128 bytes each.

A struct was then defined to encompass the idea of a "work unit".

This work unit contains the addresses to the assigned partitioned section of each RGB data channels for the current image undergoing processing, the address of each of the three response curves, the address for each of the three HDR output arrays, the address of each of the three weighted sum arrays, and finally scalar values specifying the length of the current work unit, the length of the response curve arrays and the exposure time of the current image. The data type long long was used to store this address value ensuring the full address was stored correctly. Since these work unit structures were aligned along the 128 byte cache line, a char array was used (char is 1 byte in size) to pad the structure to a byte value divisible by 128

Effectively this work unit encapsulates the data required by an SPE to process one image as part of the image composition algorithm.

Using the "libspe2" library [5] an SPE context is created for each SPE, SPE code loaded via this context, a thread created for each SPE, and finally each SPE thread is executed. From this point on a total of nine synchronous threads were in operation.

The PPE then populates each work unit structure with the addresses to the partitioned image data as it exists in main memory. For example, the first work unit contains the address data required by each SPE to process their segment of the first image in the image set. (see figure 3.9). When each work unit is populated the PPE sends each SPE a message to its inbound mailbox indicating the work unit is available for DMA. The PPE then waits for a reply indicating the composition operation for the first image is complete. When the PPE receives this message it populates the work units with the data for the next image in the set. This process continues until each image in the set has been composited by the SPE's.

On the SPE side some initial set up is performed. This involves creating several storage vectors for intermediate values, and constant values for composition calculations (such as the max 8bit value 255, and the median value 128, both used for the weighting calculations). When each SPE has completed its initial set up operations, it enters the outer processing loop which iterates so long as there are images left to process in the image set. Each SPE then waits for a PPE message via its inbound mailbox, this message indicates that the first work unit is available for DMA.

When this message is received each SPE DMA's its work unit into its local store, followed by entering into a loop which iterates through each RGB colour channel for the current image. Here the relevant response curve data is DMA'ed into a local store buffer, and an inner processing loop is entered. This inner loop iterates so long as there is data remaining to be processed in the current RGB colour channel. Here the pixel data for the current channel is DMA'ed into the local store along with the radiance sum data and the HDR composition array for the same channel. The amount of data transferred during this step is constrained by the limited size of each SPE's local store.

Finally the inner most loop which performs the actual composition operation is entered. Here the current colour channel data is iterated through, four variables at a time (corresponding to the use of SIMD)

Each iteration the following branchless SIMD composition operation is executed:

```
//perform the weighting operation
wselect = spu_cmpgt(vmidpoint,vimage[i]);
vtrue = spu_add(vimage[i],vone);
vfalse = spu_sub(vmax,vimage[i]);
vfalse = spu_add(vfalse,vone);
vweight = spu_sel(vweight,vtrue,wselect);
vweight = spu_sel(vfalse,vweight,wselect);

//add pixel weight to sum
vsum[i] = spu_add(vsum[i],vweight);

//look up the pixel values in the response curve
vcurveVals = (__vector float) {curve[image[i*4]],curve[image[(i*4)+1]],
curve[image[(i*4)+2]],curve[image[(i*4)+3]]};

//add weighted luminance to hdr array
vtemp= spu_sub(vcurveVals,vexposure);
vtemp = spu_mul(vweight,vtemp);
vhdr[i] = spu_add(vhdr[i],vtemp);
```

When the processing operations of the inner most loop are completed, the sum and HDR data is DMA'ed from the local store back into main memory. The addresses for the current colour channel's pixel value, sum and HDR arrays in main memory is then incremented by the amount of bytes processed. The addresses now point to the next

segment of that SPE's piece of the partitioned data arrays.

This processing continues until every value in each of the three colour channels has been subjected to the composition operation.

At this point the SPE's send a message to the PPE via their outgoing mailboxes to indicate that they have completed processing on the current image, each SPE then waits for a reply from the PEP. The PPE populates the work units with addresses to data from the next image in the image set and sends a message to each SPE to indicate the work unit is available for DMA. Again, the PPE waits for a reply to indicate that the image has been processed.

This process continues until each image in the image set has been processed by the SPE's.

Finally, the PPE populates a work unit for the final image composition step where the HDR data arrays are tested for 0's values indicating an overexposed pixel in the same area in each of the images in the set. These errant pixel values are set to the value present in the final image in the image set.

To facilitate this operation the work unit for each SPE contains addresses to the image data for the final image in the set.

Processing proceeds as above except without the image loop. This is because there is effectively only a single image to process (the composite HDR image). The following SIMD code performs the operation.

```
//check for hdr values which are 0 (overexposed in every image)
wselect = spu_cmpeq(vhdr[i],vzero);

//look up the input image pixel values in the response curve
vcurveVals = (__vector float) {curve[image[i*4]],curve[image[(i*4)+1]],
```

```
curve[image[(i*4)+2]],curve[image[(i*4)+3]]};

vtrue = spu_sub(vcurveVals,vexposure);

//for hdr values which are 0, set associated sum to 1
vsum[i] = spu_sel(vsum[i],vone,wselect);
//set hdr values which are 0 to the radiance data values
//in the final image in the image set
vhdr[i] = spu_sel(vhdr[i],vtrue,wselect);

//divide hdr values by the sum of their weights
vhdr[i] = divf4(vhdr[i],vsum[i]);
//convert from logarithmic scale.
vhdr[i] = expf4(vhdr[i]);
```

3.7 Tonemapping

Given our current LDR constrained display technologies, tonemapping is necessary for the display of the HDR image data computed via the previous composition step. This is the final stage in our HDRI implementation, and requires the computed HDR data for each channel (RGB) and the associated parameters for the tonemapping method.

3.7.1 Reinhard et al Global Tonemapping operator.

The Reinhard et al. Tonemapping operator is based on the photographic concept of the zone system [1]. Basically, the zone system is a method that allows a photographer to determine the best exposure level for a particular scene making measurements of the scene with a light meter and adjusting exposure with reference to the middle-grey value of a twelve zone grey card. The zone system involves additional processing at the development stage.

Reinhard's method begins calculating the log-average luminance as an approximation to the key of the scene. The term key refers to the subjective brightness of a scene. A bright room would be high-key while a dark room would be low-key.

$$\overline{L}_w = \frac{1}{N} exp\left(\sum_{x,y} log(\delta + L_w(x,y))\right)$$

where N is the number of pixels in the image, $L_w(x,y)$ is the real world luminance (taken from the HDR input to the tonemapping algorithm) of pixel at co-ordinates x,y and δ is a small constant value (say, 0.00001) used to prevent NaN errors for black pixels (black pixels are effectively 0's, log(0) is negative infinity).

This value is then mapped to the middle-grey of the HDR image by scaling the log-average luminance with the following

$$L(x,y) = \frac{a}{\overline{L}_w} L_w(x,y)$$

Where L(x,y) is the scaled luminance and a = 0.18 which is the middle grey value on a scale of zero to one. For low-key (dark images) or high-key (bright images) this value need to be modified. A value between 0.18 to 0.36 up to a max of 0.72 is recommended for low-key images and values from 0.18 down to 0.09 and a minimum of 0.045 are recommenced for high-key images.

Finally the following simple tonemapping operator is applied to the scaled luminance values.

$$L_d(x,y) = \frac{L(x,y)}{1+L(x,y)}$$

This tonemapping operator compresses high luminances by approximately $\frac{1}{L}$ while low luminances are scaled by 1. This guarantees to bring all luminance values into a displayable range of 0-1 while preserving detail in low contrast areas.

This is all that is required for the global tonemapping operator. The local tonemapping operator is similar up to the final step above. The complete process for the local operator is explained below.

3.7.2 Reinhard et al Local Tonemapping Operator

Although the global tonemapping operator produces good results in most cases, however results for images of very high dynamic range tend to lack detail in high contrast areas. For these situations Reinhard continues the motif of applying traditional photographic technique to digital imaging proposing an automated process of dodging and burning [2] to restore contrast in these areas.

Dodging and burning is a technique employed in the dark room during the development process. By blocking light from the photographic negative (usually with a piece of shaped opaque material) in a dark area lacking in detail, the total exposure time for the blocked area is lessened lightening the area revealing the detail; this process is known as dodging. Conversely the process of burning involves increasing the exposure time to an overly bright area by masking out every other area of the negative, this darkens the exposed area. Exposure time during the development process works inversely to exposure time during capture; less light results in a brighter image, more light results in a darker image.

Reinhard's local tonemapping operator seeks to emulate these results though a similar technique.

Area's on which dodging and burning are applied are bounded by sharp contrasts to the surrounding scene. Reinhard operator uses this fact to determine the local neighborhood of each pixel through a series of center-surround computations based on the subtraction of two Gaussian blurred images. If the difference between the Gaussian weighted center and the surround is significantly large this indicates that the surround lies outside the local neighborhood (overlaps into a different contrast area), otherwise the contrasts beneath the center-surround are reasonably homogeneous.

The Gaussian profile used is empirically derived and takes the form.

$$R_i(x, y, s) = \frac{1}{\pi(\alpha_i s)^2} exp(-\frac{x^2+x^2}{(\alpha_i s)^2})$$

Where s indicated the scale of the center-surround at location x,y. The use of this function across an image results in a set of responses V_i comprising the luminance at

location x,y.

$$V_i(x, y, s) = L(x, y) \otimes R_i(x, y, s)$$

The center-surround function at scale s is given by:

$$V(x, y, s) = \frac{V_1(x,y,s) - V_2(x,y,s)}{2^{\phi} a/s^2 + V_1(x,y,s)}$$

Where V_1 is the center, V_2 is the surround (defined by the above equations) and $2^{\phi} a/s^2 + V_1$ serves to normalize the gaussian difference to prevent V from becoming too large for small values of V_1 and V_2.a is the key value, while ϕ is the parameter which specifies sharpness (8 being the value recommended by the paper).

Applying this center-surround function for each pixel in the image, starting with the smallest scale s0 increasing in radius until the scale Sx is found which satisfies the following:

$$|V(x, y, s_m)| < \epsilon$$

Where ϵ is a threshold value.

V_1 may then be used as the local operator by replacing L in the final global tonemapping formula as follows

$$L_d(x, y) = \frac{L(x,y)}{1 + V_1(x, y, s_m(x,y))}$$

The difference between L and V means that dark pixels in bright regions will cause a decrease in luminance L_d, increasing the contrast for that pixel (dodging), while bright pixels in dark areas result in a lesser decrease in luminance. (burning). This has the effect of increasing pixel contrast respect to the local neighborhood.

3.7.3 x86 Implementation

Reinhard's global tonemapping operator was implemented successfully on the x86 test platform with reference to the original paper. However, some aspects of the algorithm as described caused difficulties which required modifications to various formula.

Firstly the calculation for the log average luminance of the scene proposed -

$$\overline{L}_w = \frac{1}{N} exp\left(\sum_{x,y} log(\delta + L_w(x,y))\right)$$

Suggests the following -

```
double Lw = exp(luminanceSum)/N;
```

However, the exponential of the luminance sum for the vast majority of images far exceeds the highest value that can be stored with any regular data type. In the accompanying source code for Reinhard's own implementation [25] the following is used

```
double Lw = exp(luminanceSum/N);
```

which would suggest the following formula for log average luminance

$$\overline{L}_w = exp\left((\sum_{x,y} log(\delta + L_w(x,y)))/N\right)$$

This formula is not equivalent to the original however it makes more sense as a log average calculation.

Secondly the paper does not suggest a method for extracting the luminance values from colour pixels. In this implementation the following was used to approximate the luminance given the RGB values of a pixel:

```
luminance[i] =  (0.2125 * r_hdr[i] + 0.7154 * g_hdr[i] + 0.0721 * b_hdr[i]);
```

An alternative would have been to perform a colour space transform and extract the luminance channel, this would allow operations to be performed on the luminance values in isolation and negate the need to recover RGB colour values following the tonemapping procedure.

The colour recovery goes as follows

```
r_channel = r_channel/luminanceMap * scaledLuminace;
```

simply following the formula

$$L_d(x,y) = \frac{L(x,y)}{1+L(x,y)}$$

Figure 3.8: Reinhards Global Tonmapping Operator

without taking into account the individual colour channels results in a greyscale image

Alternatively the following can be used:

```
r_channel = pow(r_channel/luminanceMap, saturation) * scaledLuminace;
```

This can be used with an input saturation parameter to control the final saturation levels in the tonemapped image.

The tonemapped data is then sanity checked for values exceeding the range 0-1. All values > 1 are clamped. Finally the values are multiplied by 255 to covert the range to 0-255.

Output from the tonemapping function is saved to the hard disk as a colour bitmap.

3.7.4 Cell Processor Implementation

The cell implementation of Reinhard's global tonemapping operator runs on the same SPE threads as the image composition operation. This was done to eliminate the associated thread initialization cost.

When image composition is completed a message is sent to the PPE from each SPE via the SPE outbound mailbox, each SPE then waits for a reply from the PPE via the SPE inbound mailbox to signal that the first tonemapping work unit is available for DMA.

While the SPE threads complete their composition work the PPE populates the work unit struct with the addresses of the tonemapping inputs in main memory. For each SPE the associated struct is passed the address of the first value of that SPE's segment of the R channel HDR data, the G channel HDR data, the B channel HDR data (RGB) and the related estimated luminance values for those channels (one array). Each SPE is assigned one eight of each input array (see figure 3.9). Additionally each struct also contains the size of each SPE's data segment, the brightness or gamma value, a

saturation value and the sum of the estimated luminances.

The PPE then waits for each completion message to arrive from the SPE before continuing, this is necessary because the previous image composition operation produces the HDR input for the tonemapping step.

When each SPE message is received the PPE sends a message to each SPE's inbound mailbox indicating that the new work unit is available. In this implementation the tonemapping work unit is the final stage of the program so the PPE calls pthread_join() for each SPE thread. This has the effect of stalling the PPE until each SPE thread ends.

On the SPE side each SPE has received the go message from the PPE. The first task the SPE's perform is to read get the work unit struct from main memory via DMA. This list of addresses and tonemapping parameters are place in local storage in the same form of struct as on the PPE side.

Scaler inputs for contrast, gamma and luminance sum are then converted to vector values via the spu_splats SIMD instruction. This is necessary for the use of these values in the SIMD tonemapping calculations.

At this point the SPE's enter the main processing loop, which consists of three nested loops.

The most outer loop iterates while data remains to be processed and DMA's luminance data into the local store. The amount of data which can be DMA'ed is controlled by the size variable read in from the work unit struct, this variable decreases with each iteration as work is completed. At the start of each iteration the size variable is checked against the maximum size of the local storage buffer. If the size is greater than the buffer the amount of data that can be brought in by DMA is equal to the maximum size of the storage buffer, otherwise the amount of data that can be read is set to the remaining size of the data to be processed.

The inner loop index's through the RGB channels and, knowing how much data can

be read this iteration, DMA's the HDR data for the current channel into the storage buffer for HDR data in the SPE's local store.

The inner most loop indexes through the HDR and luminance values stored in the local store buffers four variables at a time (corresponding to the use of SIMD intrinsics). Each iteration the following branchless SIMD tonemapping operation is executed

```
//store the current luminance values
luminance = vlum[i];

//calculate the scaled luminance values
tmp = divf4(vbrightness,vkey);
vlum[i] = spu_mul(vlum[i],tmp);
tmp = spu_add(vlum[i],vonef);
vlum[i] = divf4(vlum[i], tmp);

//tonemapping starts here
vtonemap = divf4(vhdr[i],luminance);
vtonemap = powf4(vtonemap,vcontrast);
vtonemap = spu_mul(vtonemap,vlum[i]);

//check for values exceeding 1
wselect = spu_cmpgt(vtonemap,vonef);

//for values exceeding 1, set to 1
vhdr[i] = spu_sel(vtonemap,vonef,wselect);

//multiply by 255, scale is now from 0-255
vhdr[i] = spu_mul(vhdr[i],vmaxf);
```

When the inner most loop completes, the tonemapped data is DMA'ed from the local store into main memory at the address of the current HDR channel input array, origi-

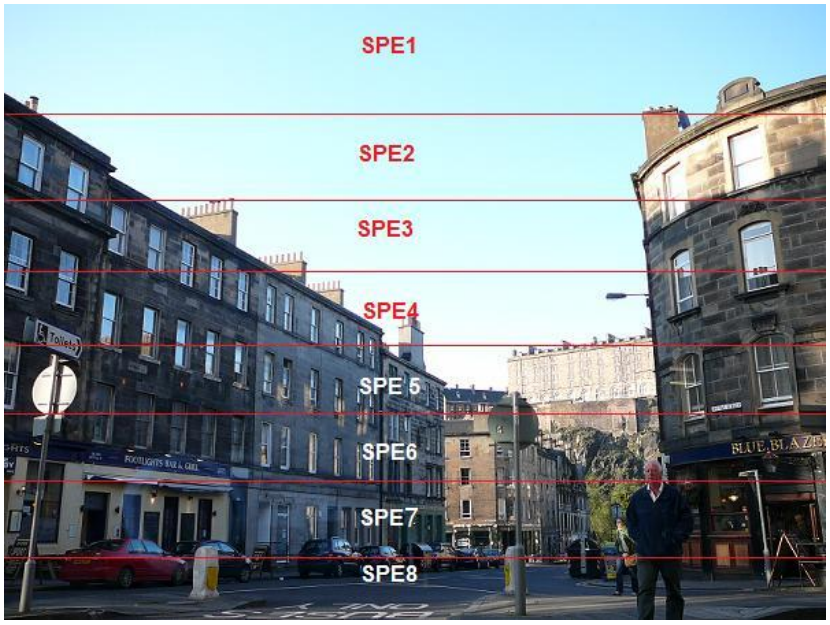

Figure 3.9: Graphical representation of data partitioning for a single image.

nally read in from the work unit struct. Effectively overwriting the input array in main memory. The amount of data put into main memory is the amount of data which has been processed that iteration. This process continues for each RGB channel.

Upon exiting the RGB channel loop, the address of the luminance array, and each HDR channel array in main memory, is incremented by the number of bytes processed this iteration (number of data values * sizeof(data type)). The size variable is then reduced, indicating the remaining values to be processed. This continues until the amount of data specified by the work unit struct is processed and DMA'ed back into main memory.

On the PPE side, on completion of each SPE thread, the tonemapped data is composed into an RGB data array suitable for image output to storage device by the stbi_write_bmp function.

Execution of the program ends at this point.

Chapter 4

Evaluation and Results

The performance testing for the Cell was carried out on a IBM BladeCenter QS20[12], while an dual core Intel E6600 machine with 4GB of ram was used as the x86 testing platform. For the BladeCenter machine tonemapping tests were performed with both 8 SPE's and 16 SPE's by accessing the additional SPE's afforded by the second Cell Processor.

Data sets for performance testing consisted of input image sets of exponentially increasing length, from two up to thirty two images with the image resolution varying from 2560x1440, 1280x720 down to 640x360. In all fifteen data sets were used with each set being used multiple times to gain an average run time result for each set. Runtime was measured at millisecond accuracy.

Performance was measured between four points. For image composition on the x86 the start point for timing was placed just after the response curve has been generated/read from file and just before the composition function call. On the Cell platform the start point is, again, placed after response recovery but just before SPE thread initialization. For global tonemapping the start point was placed after luminance data has been recovered from the HDR image but before the actual tonemapping operation as presented by the original paper. This holds true for both platforms.

The tests were designed to examine the performance of the Cell for different data

sets which would occur in regular HDRI composition and tonemapping operations. The main questions apart from how much performance benefit can be achieved via the Cell Processor, are those that relate to scaling. How does the implementation scale with changes in image set size, resolution, and SPE count?

4.0.5 Performance

Overall, scaling appears to be broadly linear across image set length and image resolution on both x86 and Cell. As the total number of pixels to be processed increases exponentially, runtime increases exponentially. (See page 62 for performance graphs and tables)

For the image composition process performance on the Cell is best for large sets of low resolution images rather than small sets of high resolution images even when the total number of pixels in the image sets are the same (2 images of 2560x1440 takes 0.28sec vs 0.19sec for 8 images of 1280x720) . Outside of experimental error this indicates that some portion of the SPE code is acting as a bottleneck. This is suggested by the fact that the difference between the processing of these two data sets is that the PPE performs less work in the low image count high resolution set. The opposite result was expected. DMA from main memory to the SPE local store is a possible suspect, double buffering could alleviate the issue by hiding the DMA latency. Additional experimentation and testing is required to trace this issue to its source. As it is, the current Cell image composition implementation represents a significant performance improvement over an equivalent x86 implementation. Additional refinement and optimization is required to unlock the full performance potential of the platform for image composition.

For the global tonemapping implementation performance results were more clear cut. Tonemapping a 2560x1440 HDR image on the Cell represents an approximate 23x performance increase over a similar x86 implementation. Interestingly scaling the implementation to 16 SPE's almost halves the runtime. This result is surprising because

the additional 8 SPE's are part of a separate Cell within the same machine. The process of accessing these external (from the point of view of a single Cell) SPE's should present a significant performance cost, however ignoring the additional setup cost of creating 8 additional threads, this was not observed. Another interesting observation is noted when tonemapping a 640x360 HDR image. Performing this operation on 8 SPE's is faster than using all 16 SPE's, this is most likely due to the set up cost for 16 threads outweighing the potential computational benefit. With this level of performance increase it should be possible to perform tonemapping operations in real time without modification to the general algorithm.

Overall these results show that the Cell Processor brings tangible performance benefits to the field of HDRI.

4.0.6 Image Composition

Image composition for static scenes produced good results overall. The lack of practical HDR displays precluded direct analysis of the HDR output from the composition operation, all outputs required tonemapping before display. Theoretically this could have caused issues resolving artifacts to a particular operator. However, since the tonemapping operator implemented was global it was generally obvious which operation was to blame when artifacts appeared in the final output.

One of the most prevalent artifacts that appeared in the final tonemapped image was ghosting. Ghosting is caused by misaligned in the input image set. Usually this misaligned is resolved through image alignment as a post processing step, however, in dynamic scenes (such as a crowed street) image alignment becomes ineffectual. Thus, the final HDR image produced by the composition operation contains faint images of the moving objects at each position they appear at in the image set (See figure 4.1).

Another artifact which appears due to image composition is lens flare. Lens flare can become quite apparent in the composited HDR image if several images in the set have lens flare to begin with (See figure 4.2).

Figure 4.1: Ghosting in a tonemapped HDR composition

Artifacts aside, it was observed that image composition has a positive effect on general image quality through noise reduction. In each tonemapped HDR image a near complete elimination of sensor noise was noted. Due to the random nature of noise in an image the same pixel of noise is generally not in the same location for each of the images in the set, thus it fades into the background. This is the same issue which causes the ghosting artifacts . However, since it occurs with isolated pixels which are usually quite close in value to their neighborhood, the effect is that they merge into the background. See figure 4.3

4.0.7 Mean Threshold Bitmap Alignment

Unfortunately the image alignment implementation failed for the majority of input image sets, resulting in gross misalignment. Greater success was noted with image sets which contained images with a low dynamic range or were close together in terms of exposure time. This is probably due to a error in selection of the threshold for generating the threshold bitmap. The recursive implementation of this algorithm caused

Figure 4.2: Lens flare in a tonemapped HDR composition

Figure 4.3: The affect of image composition on sensor noise. Left - LDR Image. Right - Tonemapped HDR

Figure 4.4: MTB alignment artifact caused by cloud movement

difficulty in debugging and the root cause of the ineffectualness of the algorithm was not discovered.

However, artifacts were still present in "correctly" aligned images. One can conclude from this that the artifacts are part of the algorithm and not due to the actual implementation.

If a dynamic object and a static object are present in a scene where the dynamic object takes up the majority of the space, the image will be aligned according to the dynamic object. This means that everything else in the scene becomes misaligned (see figure 4.4. Aligning these images with translational alignment is impossible, a more robust alignment technique is required in these situations. An alignment solution based on localized warping of the image set should prove to be an more successful in these situations.

Chapter 5

Conclusions and Future Work

The original goal of this project was to examine the applicability of the Cell Processor to the field of HDRI via a Cell based HDR through image composition implementation.

The work presented detailed both Cell SPE based image composition and tonemapping implementations which took advantage of the symmetric SIMD processing power afforded by the architecture. Image composition and tonemapping were shown to perform many times better than equivalent x86 based implementations, with the global tonemapping implementation showing particular improvement in performance to the point of becoming viable for real time applications dealing with very high resolution imagery.

These results clearly show the benefit of a Cell based approach to HDRI.

5.0.8 Future Work

The results presented show a clear improvement in the performance of HDRI methods on the cell processor. However, with additional research and development the work presented could be further improved to yield additional performance benefit.

For example, the image composition process in particular contains a clear bottleneck in its implementation. Further experimentation with alternative approaches in regard to programming model would be especially useful in this area. Experimentation with

a pipelined approach would be most likely to produce improved results. Additionally, optimizations such as double buffering, loop unrolling and software pipelining could improve performance further.

To increase the robustness of the image composition process in regards to removing artifacts in the final composition a post processing operation would prove useful. In this additional step ghost and lens flare removal functions could be employed to improve the final tonemapped image. Techniques currently exist for ghost removal such as that proposed by Khan et al. [15] which seem to offer a robust solution to the ghosting problem. Lens flare removal techniques also exist, such as those proposed by McCann et al. [18] and Talvala et al. [30]. Evaluating these techniques and building a robust postprocessing step around them should serve to produce some high quality images. Ghost removal in particular could be used to compensate for alignment errors, possibly to the extent of using ghost removal to deal with the double images introduced by misaligned images rather than using image alignment.

Disregarding the potential of ghost removal as a replacement for image alignment; it has been shown that translational alignment does not produce good results in certain dynamic scenes. Therefore, a more localized approach to image alignment would be prudent. Kang et al. [14] proposes an image warping technique using motion estimation for frame interpolation as part of a HDR video implementation. In this work Kang also noted the difficult of compositing images which involve the dynamic movement of cloud cover with respect to the ground.

Further evaluation of alternate tonemapped procedures for their applicability to the Cell architecture could suggest alternate high performance avenues for the representation of HDR data produced through image composition. Additionally, multiple tonemapping operators would increase the options available from an artistic perspective.

Finally, it would be of interest to apply the performance increases gained through Cell HDR image composition to the field of HDR video. Real time HDR video capture and tonemapping via the composition of multiple LDR video streams would represent

a significant leap toward a viable HDR video capture technology.

Appendix A

Appendix

A.1 Performance Tables and Graphs

	2560x1440 Cell	1280x720 Cell	640x360 Cell	2560x1440 86	1280x720 x86	640x360 x86
2	0.287	0.07	0.024	2.25	0.54	0.156
4	0.44	0.11	0.034	2.80	0.67	0.171
8	0.76	0.19	0.055	3.79	0.94	0.279
16	1.41	0.35	0.095	5.86	1.47	0.406
32	2.70	0.674	0.176	10.55	2.56	0.656

Table A.1: Image Composition. Cell vs x86. Images per set, resolution per image

	2560x1440	1280x720	640x360
8 SPE	0.09	0.025	0.0087
16 SPE	0.05	0.0160	0.0095
x86	2.13	0.55	0.14

Table A.2: Tonemapping. Cell vs x86. Resolution, platform

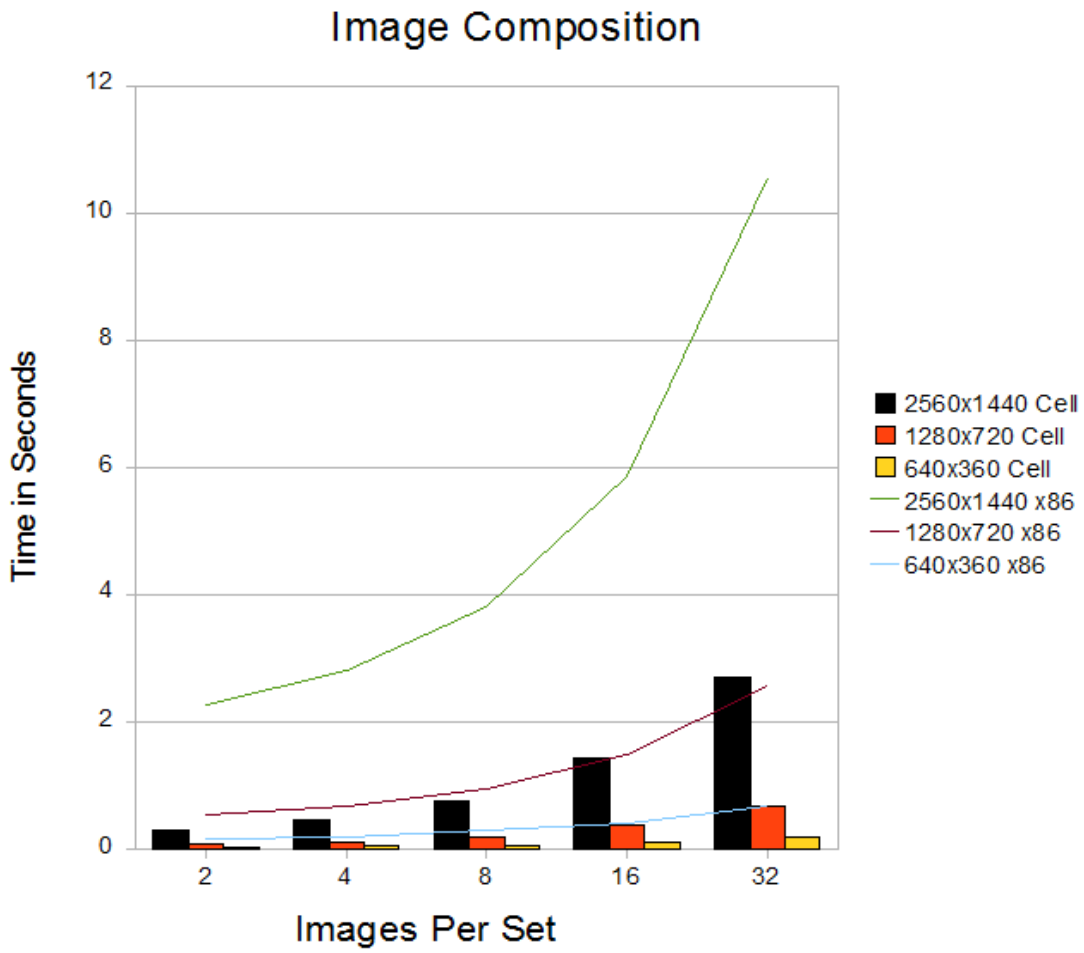

Figure A.1: Image Composition. Cell vs x86

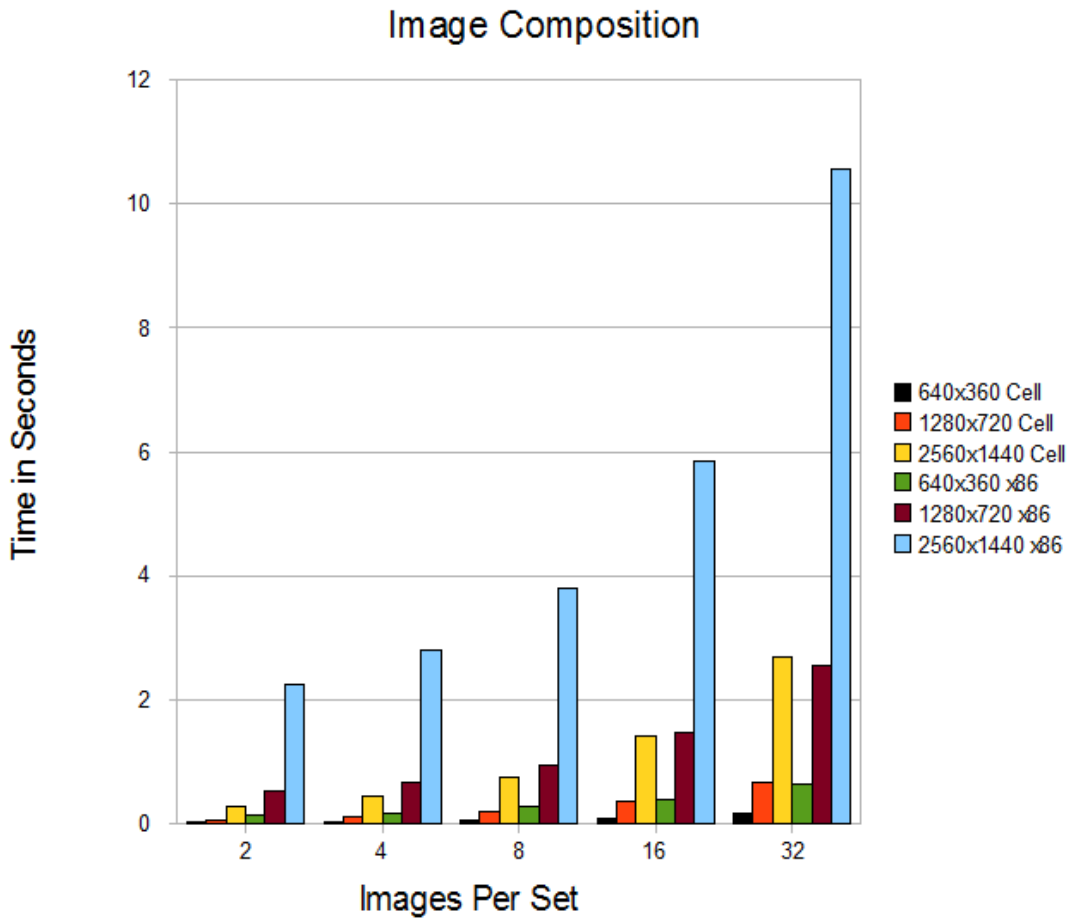

Figure A.2: Image Composition. Cell vs x86

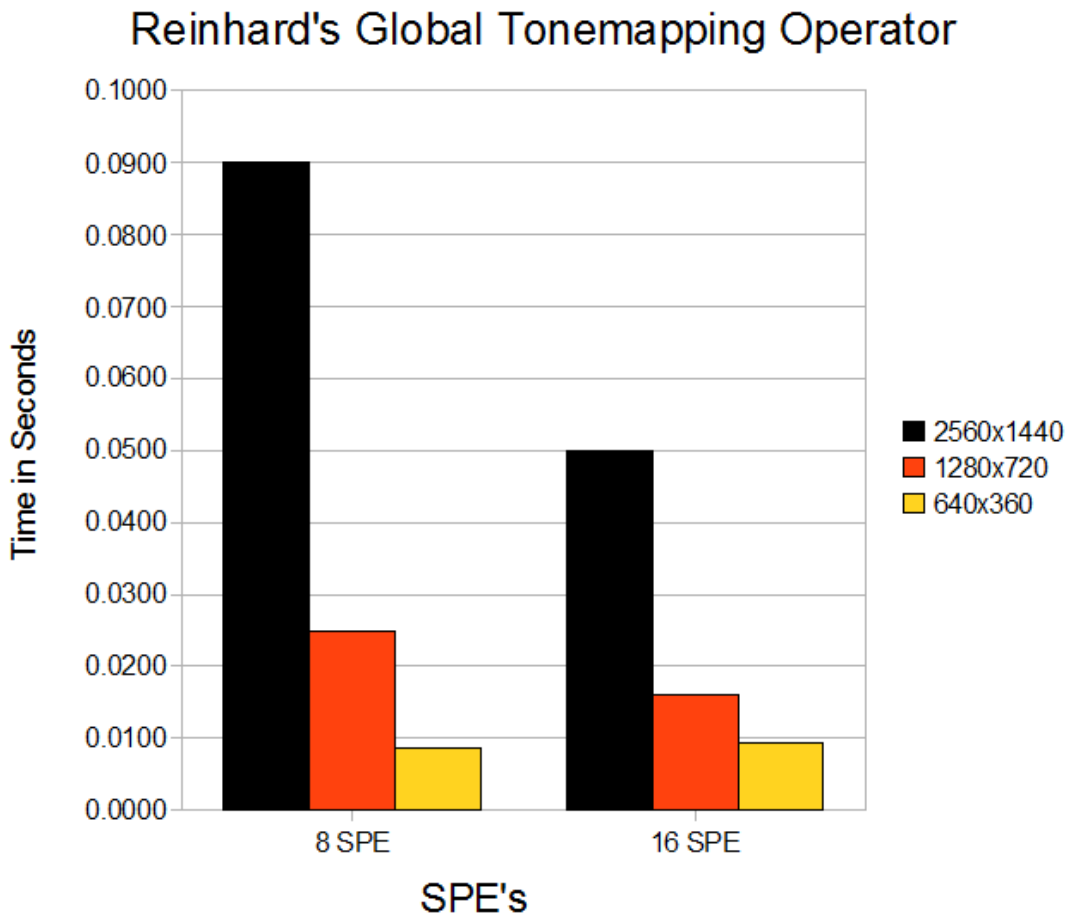

Figure A.3: Tonemapping. SPE scaling

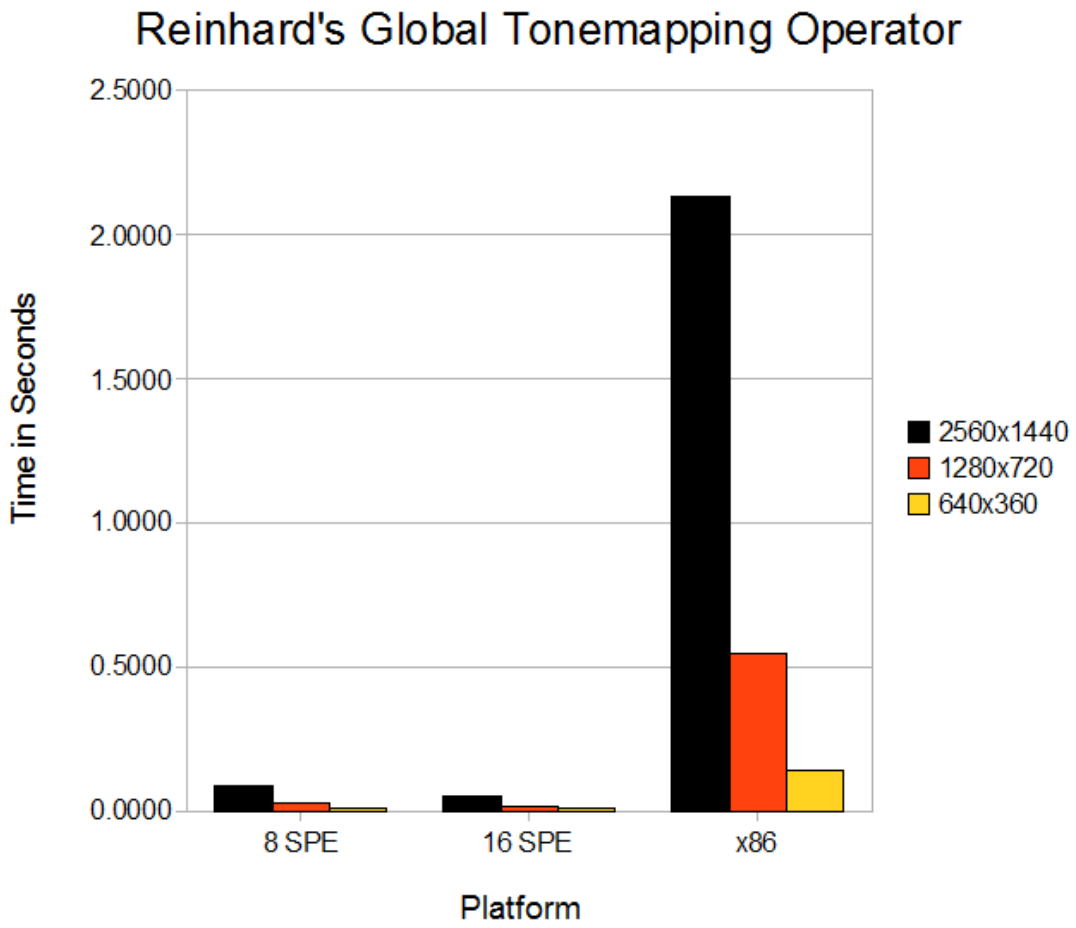

Figure A.4: Tonemapping. Cell vs x86

Bibliography

[1] A. Adams and R. Baker. *The negative*. Bulfinch Press, 1981.

[2] A. Adams and R. Baker. *The print*. Little, Brown, 1983.

[3] E.H. Adelson, CH Anderson, JR Bergen, P.J. Burt, and JM Ogden. Pyramid methods in image processing. 2004.

[4] Sean Barrett. `http://www.nothings.org/`.

[5] Jonathan Bartlett. Changes in libspe: How libspe2 affects cell broadband engine programming. `http://www.ibm.com/developerworks/library/pa-libspe2/`, Jul 17, 2007.

[6] Daniel A. Brokenshire. Maximizing the power of the cell broadband engine processor: 25 tips to optimal application performance. `http://www.ibm.com/developerworks/power/library/pa-celltips1/`, 27 Jun 2006.

[7] CImg. The cimg library - c++ template image processing toolkit. `http://cimg.sourceforge.net/`.

[8] Matthew Cole. A tedious explanation of the f/stop. `http://www.uscoles.com/fstop.htm`.

[9] P.E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 1997.

[10] Hiroki Eda. Toshiba demonstrates cell microprocessor simultaneously decoding 48 mpeg-2 streams. `http://techon.nikkeibp.co.jp/english/NEWS_EN/20050425/104149/`, Apr 25, 2005.

[11] hdrsoft. Photomatrix. http://www.hdrsoft.com/.

[12] IBM. Ibm support - overview - ibm bladecenter qs20. http://www-947.ibm.com/systems/support/supportsite.wss/docdisplay?brandind=5000008&lndocid=MIGR-5074153.

[13] jhead. Exif jpeg header manipulation tool. http://www.sentex.net/~mwandel/jhead/, Mar 02, 2009.

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

[15] EA Khan, AO Akyuz, and E. Reinhard. Ghost removal in high dynamic range images. In *2006 IEEE International Conference on Image Processing*, pages 2005–2008, 2006.

[16] S. Mann. Compositing multiple pictures of the same scene. In *Proceedings of the 46th Annual IS&T Conference*, 1993.

[17] S. Mann and R.W. Picard. Being undigital with digital cameras: Extending dynamic range by combining differently exposed pictures. In *Proceedings of IS&T 46th annual conference*, pages 422–428. Citeseer, 1995.

[18] JJ McCann and A. Rizzi. Veiling glare: the dynamic range limit of HDR images. *B. Rogowitz, T. Pappas, S. Daly, Proc. SPIE, Bellingham WA*, pages 6492–41, 2007.

[19] T. Mitsunaga and S.K. Nayar. Radiometric self calibration. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 374–380, 1999.

[20] NASA. Nasa high-end computing: Cell study feature. http://www.hec.nasa.gov/news/features/2008/cell.074208.html.

[21] Microsoft Developer Network. Hdr lighting (direct3d 9). http://msdn.microsoft.com/en-us/library/bb173486(VS.85).aspx.

[22] Qtpfsgui. Qtpfsgui-news. http://qtpfsgui.sourceforge.net/, 2009.

[23] IBM Redbooks. Programming the Cell Broadband Engine Architecture: Examples and Best Practices. 2008.

[24] E. Reinhard and P. Debever. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2006.

[25] Erik Reinhard. Photographic tone reproduction for digital images. `http://www.cs.ucf.edu/~reinhard/cdrom/source.html`.

[26] M.A. Robertson, S. Borman, and R.L. Stevenson. Estimation-theoretic approach to dynamic range enhancement using multiple exposures. *Journal of Electronic Imaging*, 12:219, 2003.

[27] J.B. Schriever. *Complete self-instructing library of practical photography: Volume IV*. 1909.

[28] Camille Silvy. Silvy, camille leon louis: Szene an einem fluss in frankreich. `http://www.zeno.org/Fotografien/B/Silvy,+Camille+Leon+Louis:+Szene+an+einem+Fluss+in+Frankreich`.

[29] P. Sprawls. *Physical principles of medical imaging*. Lippincott Williams and Wilkins, 1987.

[30] E.V. Talvala, A. Adams, M. Horowitz, and M. Levoy. Veiling glare in high dynamic range imaging. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2007.

[31] Template Numberical Toolkit. Tnt download page. `http://math.nist.gov/tnt/download.html`.

[32] FDR Tools. Full dynamic range tools broadening the horizon of photography. `http://www.fdrtools.com/`.

[33] Thomson Grass Valley. Viper filmstream digital cinematography camera. `http://www.thomsongrassvalley.com/products/cameras/viper/`, 2009.

[34] Spheron VR. Spherocam hdr. `http://www.spheron.com/en/intruvision/solutions/spherocam-hdr.html`.

[35] G. Ward. Fast, robust image registration for compositing high dynamic range photographs from handheld exposures. *Journal of Graphics Tools*, 2003.

[36] D.S. Watkins. *Fundamentals of matrix computations*. Wiley-Interscience, 2004.