Streamed High Dynamic Range Imaging

Donald G Bailey

School of Engineering and Advanced Technology, Massey University
Palmerston North, New Zealand
D.G.Bailey@massey.ac.nz

(*Demonstration Paper*)

***Abstract*—The problem of mapping from a high dynamic range image, or a series of images spanning a wide dynamic range, to a standard 8-bit image is considered. Several techniques for dynamic range compression are reviewed. FPGA implementations are developed and demonstrated, with a focus on directly processing the video stream from the camera.**

I. Introduction

Many scenes contain a very wide dynamic range of light intensities. Within this environment, the human visual system is remarkable in its ability to detect subtle contrast variations, and is consequently able to reliably interpret such scenes. The dynamic range of most image displays is limited to 8 bits per colour channel, limiting what can reasonably be reproduced [1]. Since display is limited, many image sensors and image processing systems are also often restricted to 8 bits, severely limiting its ability to make similar subtle distinctions. Obtaining a high quality image in an environment with a wide range of contrast can be challenging. If the image is exposed for the bright areas then there is a loss of detail in the darker regions of the image. If the image is exposed for the darker regions, then the bright areas become saturated, and lose detail.

One standard approach is to capture a sequence of images with different exposure times, and fuse the resulting images into a single composite image [2, 3]. The exposure of each of the different images is optimised for the different regions of the image, corresponding to different portions of the dynamic range. When these images are fused, the wide dynamic range is compressed into fewer bits, with the fine detail retained. One limitation of this approach is the scene must be static during the series of exposures. Any motion within the scene will result in ghosting artefacts as objects appear in multiple places, combined with the background.

One approach to overcome this is to use multiple sensors to simultaneously capture the different exposures. This is similar to a 3-chip colour camera where a separate sensor is used for the red, green and blue spectral components. Instead, each sensor has different light sensitivity. Nayar and Mitsunaga [4] implemented a single chip version, where different pixels were given different exposures by masking them appropriately with neutral density filters.

Many more modern sensors have a wider dynamic range (10 to 12 bits is typical now), although most image processing and image storage algorithms still work with 8-bit pixels. A single 12-bit image can be thought of as providing a series of simultaneous bracketed exposures simply by selecting different bit ranges. The problem then becomes one of how to compress the wide dynamic range to a form suitable for display without loss of significant detail.

Typically two approaches are taken to dynamic range compression [5]. The first is to apply a compressive mapping of the input pixels to the pixel values displayed. Such a tone reproduction curve is a pixelwise mapping, making the implementation very simple. The second approach also considers the spatial context. Effectively the image is decomposed into two or more bands; the low spatial frequency bands are compressed, while the higher spatial frequency bands are retained, to retain strong local contrast. While such images may be visually more appealing, the direct relationship between the pixel value and light intensity is lost, limiting their utility in applications where intensity is being measured.

This paper considers the implementation of several dynamic range compression algorithms on a field programmable gate array (FPGA). The key constraint applied is to perform the dynamic range compression directly on the images as they are streamed from the camera. Working with such streamed data minimises the data handling.

Section II describes the implementation system. The different range compression schemes are presented in Section III, outlining their FPGA implementation for streamed processing between the camera and display. Section IV discusses the implications of different filter choices and how to extend the work for colour image compression. Section V gives an overview of the demonstration.

II. Demonstration System Setup

To compare the different methods for dynamic range compression, and demonstrate their practical application, the Terasic tPad platform was chosen [6]. This kit consists of a development and education board DE2-115 integrated with an 800×600 touch panel display and 5 megapixel CMOS camera module. The camera has 12 bit pixels, and can be configured to read out a window, enabling 800×600 images to be captured. The display takes 18 bit RGB pixels (6 bits for each channel). The same data can be displayed to an external monitor at 8 bits per channel.

Handel-C has been used as the hardware description language, compiled to EDIF by Mentor Graphics DK Design Suite version 5.5. The EDIF is mapped to the FPGA by Quartus II web edition, version 11.0.

The first step in the design is to synchronise the camera with the display. This enables the images to be streamed directly from the camera to the display, without the need for

978-1-4673-2845-6/12/$31.00 ©2012 IEEE

frame buffering. The pixel clock for the display is 40 MHz, which results in a row period of 26.4 μs (including the blanking period at the end of a row). The camera, with an 800 pixel wide window, has a minimum row period of 1700 clock cycles. This gives an awkward pixel clock frequency (64.394 MHz). However, extending the row length to 1716 clock cycles corresponds to a 65 MHz clock, which can be derived using a phase locked loop with a simple integer ratio. The difference in clock rates between the camera and the display is managed by a dual-port row buffer, with one port in each clock domain.

The camera uses a Bayer pattern colour filter array. The raw image from the camera is converted to an RGB colour image through bilinear interpolation. From this, a monochrome image is produced:

$$M = \frac{R + 2G + B}{4} \tag{1}$$

The overall system setup is shown in Fig. 1. The row synchronisation timing from the camera is used to synchronise the timing generator for the display. The timing generator also provides advanced trigger signals to the other blocks to account for their latency.

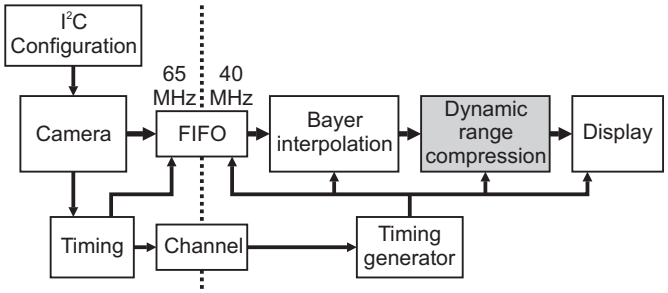

Fig. 1: Hardware configuration of the demonstration. The different dynamic range compression algorithms are substituted for the shaded box.

III. Implementation of Dynamic Range Compression

The architectures of several different dynamic range compression techniques are described. The goal is to compress the 12-bit pixels from the camera to an 8-bit image for the display. The 12-bit image can also be considered as a series of bracketed exposures by selecting different 8-bit ranges from the 12-bit image. Pixels which exceed a particular 8-bit range are saturated at 255.

A. Baseline – No Dynamic Range Compression

The baseline is to perform no dynamic range compression. A selected 8-bit range is directly displayed. Since the camera image is exposed for the highlights, selecting the top 8 bits is not good at representing details within the low-light regions of the image. Selecting a middle range gives the best average exposure, at the cost of saturating the highlights.

B. Point Transform – Power and Logarithmic Curves

The simplest form of dynamic range compression is to apply a monotonic mapping to each pixel. To compress the highlights, without seriously compromising the details in the low light regions, the mapping should be steeper for low pixel values and flatter for high values, as shown in Fig. 2.

Two classic curves follow this general shape. The first is a power law curve (gamma transformation):

$$Q_N = M_N^{\gamma} \tag{2}$$

where M_N and Q_N are normalised input and output pixel values (in the range 0 to 1) and γ is the transformation parameter (<1). A greater or lesser compression of the highlight details can be achieved by decreasing or increasing γ respectively.

The second is a logarithmic curve:

$$Q = k \log(M + 1) \tag{3}$$

where the scale factor, k, is chosen to match the maximum values of the range. The +1 avoids an undefined mapping for a pixel value of 0.

The problem with the logarithmic curve is that it tends to over-enhance the dark regions, and significantly compress the contrast in the mid and light regions. As a result the image appears quite flat. The power curve is better in this regard, as it allows the contrast between the dark and light regions to be better balanced (with selection of appropriate γ).

Being a point operation, the mapping can be pre-calculated and stored in a lookup table. This can easily be implemented using a memory block on the FPGA.

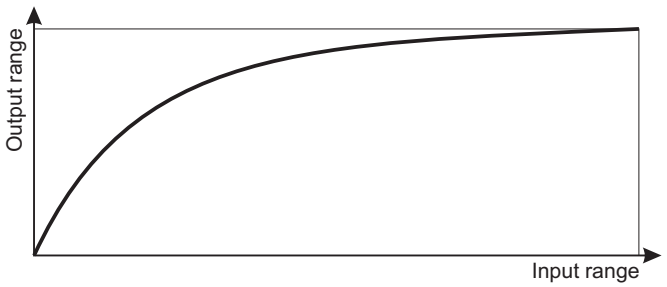

Fig. 2. Monotonic mapping for dynamic range compression.

C. Histogram Equalisation

Rather than preset the balance between the light and dark regions in the output image, this can be determined dynamically through histogram equalisation. Implicit in using histogram equalisation for contrast adjustment is that for tones there are a large number of pixels, the contrast is enhanced, and where there are fewer, the contrast is reduced. A useful feature of histogram equalisation is that it is non-parametric and that it automatically adapts to the image content.

Histogram equalisation is based on the statistics of a complete frame. Therefore, the frame must be buffered while the histogram is built. However, for many images, the statistics do not change significantly from frame to frame. This suggests that the histogram gathered for the previous frame will be very similar to that for the current frame. Therefore a simpler alternative is to use the histogram gathered from the previous frame to equalise the current frame [7], allowing a streamed implementation.

The mapping produced by histogram equalisation corresponds to the scaled cumulative histogram. The

monotonic nature of the mapping enables an efficient hardware implementation using repeated subtraction to perform the scaling [8].

D. Point Fusion – Simple Average

Rather than operate from a single high dynamic range image, most traditional algorithms work from a series of images captured with different exposures. The simplest fusion algorithm is to simply average the individual input images.

This works better than expected, because the images exposed for the darker regions are saturated in the bright regions. Therefore averaging results in a piecewise linear approximation to the gamma curve of (2), with the knees corresponding to the intensities where the individual images saturate [4].

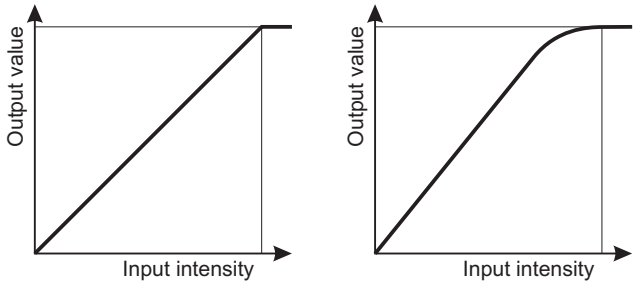

Fig. 3: Adjusting the clipping function to smooth the mapping. Left: hard clipping for saturation; Right: soft clipping for saturation.

The knees can be smoothed by soft clipping rather than hard clipping as illustrated in Fig. 3. This has an added advantage of increasing the contrast slightly within the region of interest.

Simply adding the images together requires only a small amount of logic. If smooth clipping is used, this is easily implemented using a lookup table to map each input image. If the input images are derived by selecting bit ranges from a higher dynamic range image, then the complete mapping from the original image can be implemented more simply with a larger lookup table.

E. Point Fusion – Weighted Average

Rather than give equal weight to each of the different input images, they could be weighted based on which image has the best contrast [3]. In the extreme, this could simply select from the different exposures based on "image quality" at each point. If based purely on individual pixels, such a simplistic selection results in noisy discontinuities and even contrast reversals in the boundaries between selections.

One simple "image quality" measure is how close each pixel was to mid range (128). The corresponding a discrete selection function would be

$$f = \arg\min_i \left| M_i - 128 \right| \tag{4}$$

where M_i is the pixel value of the i th exposure, arranged in an ordered sequence. The contrast artefacts can be reduced by spatially smoothing this selection function with a linear filter (for example a moving average or box filter). From the resulting smoothed value, $\bar{f}$, the output is selected by weighting the two nearest exposures:

$$Q = \left(\lceil \bar{f} \rceil - \bar{f} \right) M_{\lfloor \bar{f} \rfloor} + \left(\bar{f} - \lfloor \bar{f} \rfloor \right) M_{\lceil \bar{f} \rceil} \tag{5}$$

where $\lceil\ \rceil$ and $\lfloor\ \rfloor$ are ceiling and floor operators respectively. Such a smoothing filter is relatively easy to implement on an FPGA [8], with the resulting weighting implemented using DSP blocks on the FPGA.

An alternative "image quality" measure is based on local dynamic range. This weights exposures which have a better local contrast more highly. Under-exposed images have a lower dynamic range simply as a result of reduced scaling. Increasing the exposure increases the local dynamic range until the image begins to saturate, at which point the dynamic range begins reducing again. Correctly exposed images therefore receive the highest weight. A simple measure of local dynamic range is the standard deviation within a window. The weighted output is then

$$Q = \frac{\sum M_i \sigma(M_i)}{\sum \sigma(M_i)} \tag{6}$$

where $\sigma(M_i)$ is the result of applying a standard deviation filter to M_i .

Implementing the standard deviation filter requires accumulating both M and M^2 within the window. The standard deviation is then given as

$$\sigma(M) = \frac{\sum M^2}{N} - \left(\frac{\sum M}{N} \right)^2 \tag{7}$$

where N is the number of pixels within the window. Equation (7) requires quite a lot of logic to evaluate for every pixel. An alternative is to use the local range within the window as a measure of contrast. This has the advantage that the constituent MIN and MAX filters are separable and have efficient implementations for streamed data [9]. The normalising division in (6) can be implemented efficiently using a simple non-restoring division algorithm.

There are potentially many other weighting functions which could be applied to fuse the individual exposure images.

F. Image Scaling

In the input high-dynamic range image, the dynamic range could be improved if there was less light in the highlight areas and more light in the darker areas. A reduced dynamic range is achieved by scaling the different parts of the image differently to simulate changing the lighting or the exposure.

The light level at each point can be approximated by a low pass filter, with the scaling function derived from this:

$$Q = M \cdot s(\bar{M}) \tag{8}$$

where $\bar{M}$ is the low pass filtered image and $s()$ is the scaling function.

An advantage of this scaling approach is that the local contrast ratio is retained because adjacent pixels are given similar weight. A scaling function which maps the smoothed image using the gamma transformation of (2) is therefore

$$s(\bar{M}) = \frac{\bar{M}_N^{\gamma}}{\bar{M}_N} = \bar{M}_N^{\gamma-1} \tag{9}$$

Such a mapping can easily be implemented using a lookup table on the smoothed image to obtain the scale factor.

Since the scale factor has a negative slope with intensity, a simple alternative would be to apply the negative of the image as the scale factor:

$$s(\bar{M}) = m - k\bar{M} \tag{10}$$

where m and k are chosen to give a suitable range of scale factors between the highlights and shadows.

One of the limitations of simple image scaling is the halo effect around sharp contrast edges [5]. This halo results from the sharp contrast edge being blurred, and regions on either side of the edge receiving the same scaling. To overcome this requires using non-linear filters (for example a trimmed filter or a morphological filter).

In terms of implementation, a linear blurring filter is straight forward to implement, although a non-linear filter can require significantly more resources depending on its complexity. A DSP block can be used to scale the image.

G. Multi-channel Fusion

The principle behind multi-channel fusion is to split the input image or images into low and high frequency components. Separate compression curves are then used for the different components. Typically wavelets [10] or Gaussian pyramids [2] are used to decompose the image into frequency bands. The dynamic range of the low frequencies can be compressed using a gamma transformation or other related mapping. The high frequency components contain the details within the image. These can be mapped with a steeper transformation (less compression) to retain the local differences and give good local contrast.

IV. Discussion

In the descriptions above, simple moving average filters were used, both for their simplicity and for their separability. Unfortunately, simple moving average filters have a poor frequency response. The performance could be improved by using Gaussian weighted filters. These are also separable, but have two disadvantages from an implementation perspective. First, the non-uniform weights requires more complex calculation. These can be implemented with DSP blocks, or by reusing sub-expressions to reduce the number of additions. An alternative is to approximate a Gaussian by repeated iteration of smaller uniform filters [11]. The second disadvantage is that to achieve the same level of smoothing as a uniform weight filter, the filter length has to be considerably longer. This requires more resources, particularly in terms of row buffers to implement the filter.

The descriptions in this paper have focused on monochrome images. Extending to colour is not trivial. Applying a non-linear map individually to each of the RGB channels results in colour distortion. This is made worse if any of the input images are saturated, because the true colour is lost for those pixels. The simplest approach is to separate the image into its luminance and chrominance components, and apply the mapping to the luminance components. It is also necessary to scale the chrominance components in proportion to the luminance to maintain good colour representation. For this reason, using the HSV colour space would be preferable because the saturation is already normalised with the value or intensity.

V. Demonstration Overview

The demonstration at the conference uses the setup described in Section II. Each of the algorithms described in Section IV, is demonstrated for monochrome images.

It is shown that while the point based approaches are moderately effective at compressing the dynamic range, they are limited in that the local contrast is also, by necessity, compressed. The key advantage of the point based approaches is their computational simplicity and low resource requirements.

It is shown that filtering based approaches are able to take into account the local contrast, and can maintain the contrast while compressing the overall dynamic range. However, this comes at the expense increased logic resources to implement the filters.

One limitation of using a single wide dynamic range image as input is that the darker regions are noisier than if a series of images was captured, each optimally exposed for a particular range.

The demonstrations compare the visual effectiveness of the different techniques.

References

[1] G. Ward, "High Dynamic Range Imaging," in *Ninth Colour Imaging Conference*, Scottsdale, Arizona, USA, 2001, pp. 9-16.

[2] P. J. Burt and R. J. Kolczynski, "Enhanced image capture through fusion," in *Fourth International Conference on Computer Vision*, Berlin, Germany, 1993, pp. 173-182.

[3] S. Mann and R. W. Picard, "On being 'undigital' with digital cameras: extending dynamic range by combining differently exposed pictures," in *Proceedings of IST's 48th Annual Conference*, Cambridge, Massachusetts, USA, 1995, pp. 422-428.

[4] S. K. Nayar and T. Mitsunaga, "High dynamic range imaging: spatially varying pixel exposures," in *IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina, USA, 2000, pp. 472-479.

[5] J. M. DiCarlo and B. A. Wandell, "Rendering high dynamic range images," in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications*, San Jose, California, USA, 2000, pp. 392-401.

[6] Terasic, *tPad User Manual* Version 1.0.3: Terasic Technologies, 2010.

[7] A. J. McCollum, C. C. Bowman, P. A. Daniels, and B. G. Batchelor, "A histogram modification unit for real-time image enhancement," *Computer Vision, Graphics and Image Processing,* vol. 42, no. 3, pp. 387-398, 1988.

[8] D. G. Bailey, *Design for embedded image processing on FPGAs*. Singapore: John Wiley & Sons (Asia) Pte. Ltd., 2011.

[9] D. G. Bailey, "Efficient implementation of greyscale morphological filters," in *International Conference on Field Programmable Technology (FPT 2010)*, Beijing, China, 2010, pp. 421-424.

[10] G. Pajares and J. M. de la Cruz, "A wavelet-based image fusion tutorial," *Pattern Recognition,* vol. 37, no. 9, pp. 1855-1872, 2004.

[11] F. M. Waltz and J. W. V. Miller, "An efficient algorithm for Gaussian blur using finite-state machines," in *Machine Vision Systems for Inspection and Metrology VII*, Boston, Massachusetts, USA, 1998, pp. 334-341.