Low-latency histogram equalization for infrared image sequences – a hardware implementation

Volker Schatz
Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung, Ettlingen, Germany
firstname.lastname∂iosb.fraunhofer.de

This is the author-generated version of a paper published in the Journal of Real-Time Image Processing on 15 June 2011 (received 22 November 2010, accepted 13 May 2011).

This work describes a hardware implementation of the contrast-limited adaptive histogram equalization algorithm (CLAHE). The intended application is the processing of image sequences from high-dynamic-range infrared cameras. The variant of histogram equalization implemented is the one most commonly used today. It involves dividing the image into tiles, computing a transformation function on each of them, and interpolating between them. The contrast-limiting is modified to facilitate the hardware implementation, and it is shown that the error introduced by this modification is negligible. The latency of the design is minimized by performing its successive steps simultaneously on the same frame and by exploiting the vertical blank pause between frames. The resource usage of the histogram equalization module and how it depends on its parameters has been determined by synthesis. The design has been synthesized and tested on a Xilinx FPGA. The implementation supports substituting other dynamic range reduction modules for the histogram equalization component by partial dynamic reconfiguration.

Keywords: **Histogram equalization, CLAHE, infrared images, FPGA design, partial dynamic reconfiguration**

1 Introduction

Histogram equalization is an image enhancement technique that consists in a grey-level transform designed to equalize the frequency of occurrence of different grey values. It has its origins in the information theoretic problem of maximizing the information content of discretized data by applying a transformation before discretization. These concepts were applied to images by Hummel [Hum75]. Histogram equalization has since been successively refined. Most importantly, each pixel can be transformed based on the histogram of a contextual region [KLW74, KLW76, Hum77, Piz81], and the contrast amplification can be limited by clipping the histogram [PAA+87]. The latter variant, called contrast-limited adaptive histogram equalization (CLAHE), is in most widespread use today and is the method that has been implemented here.

Specialized hardware for computing histogram equalization also has a long history. The paper that introduced contrast limiting [PAA+87] also discussed how to implement histogram equalization on special processor architectures. Since then, implementations as a specialized multiprocessor machine [EPA90] and as parallel software on commercially available hardware [Kur91] have been provided. More recently, several implementations of global histogram equalization have been developed that either partly [SS99, DND+01] or exclusively [LNC+98, JCR03, AA05] rely on FPGA logic. Reza has presented a hardware implementation of CLAHE with an application in medical imaging in mind [Rez04]. Ferguson *et al.* have developed an FPGA implementation of CLAHE intended for video processing [FAEP08]. Both closely follow the software algorithm.

The intended application of this implementation is the processing of image sequences from an infrared (IR) camera with a dynamic range of 14 bits or more. Reducing the dynamic range is necessary before the images can be displayed on a standard computer monitor. Most displays support a dynamic range of 8 bits in each colour channel, resulting in a dynamic range of 8 bits for greyscale images as well. Displays with a higher dynamic range are sometimes recommended, such as for medical applications [NEM09], but the available dynamic range should always be utilized as effectively as possible. Histogram equalization is the method of choice for this purpose.

Besides visualization, there are other reasons to use histogram equalization. The FPGA implementation that is presented in this paper is to be part of a reconfigurable hardware image processing system briefly described in [BGP+09]. Reducing the dynamic range of pixel values allows to reduce the chip area required for downstream image processing algorithms if necessitated by resource constraints. Employing histogram equalization for this purpose allows to do this in a way that minimizes image degradation. As one important application of the image processing system is the investigation of object tracking methods, an important requirement for this histogram equalization implementation is minimal latency, so that the reaction time of tracking algorithms is largely unaffected.

This paper is structured as follows. The following section will review histogram equalization methods and present an optimization of the chosen method for a hardware implementation. The implementation will be presented in section 3 and the appendix. Section 4 will describe the application and the test setup and demonstrate the benefits of partial dynamic reconfiguration. The final section will summarize the paper.

2 THEORY

2.1 Transformation formula

This section will present a brief derivation of the histogram equalization transformation formula; for a more detailed version, see [Hum75] or textbooks such as [GW02].

Histogram equalization, as its name suggests, refers to transforming an image in such a way that the histogram of the resulting image is flat. It is not generally possible to do this exactly in the case of a real-world discretized image. In the following derivation, we will therefore work with an idealized greyscale image, which has continuous coordinates and a continuous brightness value at each point. Without loss of generality, we will assume the range of brightness values is the unit interval $I = [0, 1]$.

Then the histogram of the image is also a continuous function, the probability density function (PDF) of the image values x:

$$\begin{aligned} p : I &\to I \\ x &\mapsto p(x) \end{aligned}$$

If y is the image value x is transformed to, and q is its PDF, the requirement of histogram equalization is expressed by $q(y) \equiv 1$. From the principle of probability conservation

$$|p(x)\,\mathrm{d}x| = |q(y)\,\mathrm{d}y|$$

and the requirement that we seek a monotonically increasing transformation function, one derives the transformation:

$$y = \int_0^x p(x')\,\mathrm{d}x'$$

Expressed in words, the histogram equalizing transformation function is the cumulative distribution function (CDF) of the image values.

This transformation formula is discretized for use with real-world discretized images:

$$\begin{aligned} x &\in \{0, 1, \ldots, 2^\xi - 1\} \\ y &\in \{0, 1, \ldots, 2^\eta - 1\} \\ y &= \left\lfloor \frac{2^\eta - 1}{N} \sum_{x'=0}^{x} N_{x'} \right\rfloor , \end{aligned} \tag{1}$$

where N is the total number of pixels in the image or image region, N_x is the number of pixels with value x, and $\lfloor \cdot \rfloor$ is the floor function. Usually $\xi \geq \eta$, i.e., y has no more bits than x. If some of the bin counts $N_{x'}$ are too large (larger than $N/(2^\eta - 1)$), there will be gaps in the histogram of y. This is a consequence of the fact that values that have been lumped together in the same x bin cannot be picked apart again. So in the discretized case, the histogram of the image resulting from a histogram equalization is only flat when averaged across the gaps. Nonetheless, the resulting image shows improved contrast.

y can also be computed from a histogram that contains more than one value of x in each bin. This requires that the bit width of y is smaller than that of x, or the accuracy of the result will suffer. The formula for y then takes the form:

$$y = \left\lfloor \frac{2^\eta - 1}{N} \sum_{b=0}^{\beta(x)} \bar{N}_b \right\rfloor , \tag{2}$$

where $\bar{N}_b$ is the bin count of bin b and $\beta(x)$ stands for the bin which the original pixel value x belongs to.

2.2 Variants

Since its inception, histogram equalization has evolved into a number of variants, each of which redresses some of the drawbacks of the previous one. Applying histogram equalization globally to an image has the disadvantage that bright and dark regions of the image are treated equally. This may cause the contrast in very bright or dark regions to remain bad or even to deteriorate. This problem is solved by adaptive histogram equalization (AHE) [KLW74, KLW76, Hum77, Piz81]. AHE transforms each pixel according to the histogram of a neighbourhood region. As a consequence, contrast is enhanced locally, with physically separated bright and dark regions treated independently. The size of the neighbourhood region serves as a scale parameter. Variations in the image data on length scales larger than this parameter are suppressed, those on smaller length scales are enhanced.

The downside of AHE from a quality perspective is that it can overamplify noise: when the neighbourhood lies completely within a nearly homogeneous region, a very small range of values is mapped to the whole output range. This is remedied by CLAHE [PAA+87]. This method clips the histogram at a predetermined value before computing the transformation function, thereby limiting the contrast amplification. It was found [PAA+87] that it is advantageous to redistribute the excess rather than discard it. This is done in a recursive process that, while it can be implemented in hardware [Rez04, FAEP08], is not particularly suitable for it. The following subsection will present an equally good alternative that lends itself well to a hardware implementation.

The second drawback of AHE (and the straightforward version of CLAHE) is that it is computationally expensive (in software) or quite complex (in hardware).

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing

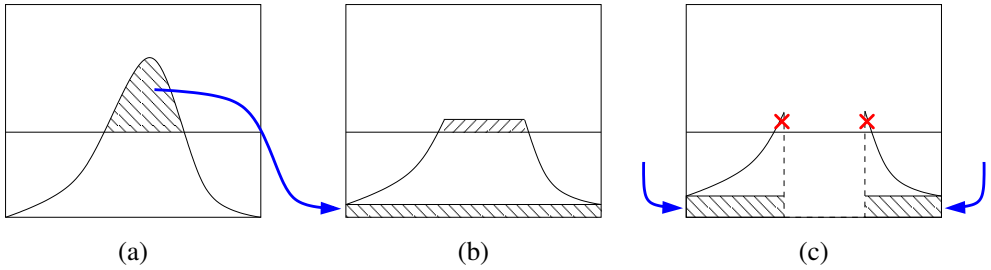

Figure 2. The redistribution procedure that is part of the histogram clipping for CLAHE. (a), (b) One step in the conventional recursive method. (c) The new single-step method: only non-exceeding bins receive the redistributed excess, and the secondary excess is discarded (see text).

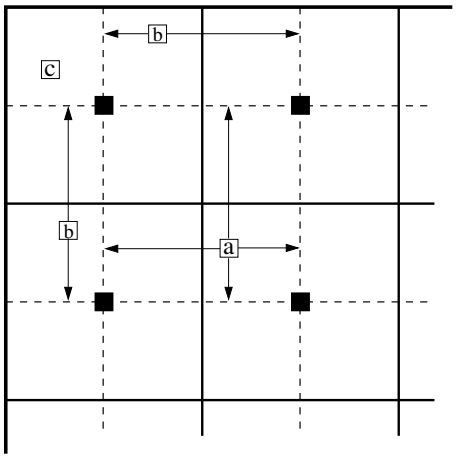

Figure 1. Tile-interpolated CLAHE: Transformations computed from the tile histograms are appropriate for the tile centre pixels (black squares). Pixels in the bulk of the image (a) are bilinearly, margin pixels (b) linearly interpolated between the transformations of the nearest centre pixels. Corner pixels (c) transform as the corner centre pixel.

This is also redressed in [PAA+87]. Rather than compute the histograms and transformation functions for the neighbourhood of each pixel, this is done for each tile of a rectangular grid into which the image is divided up. For the centre pixel of a tile, that transformation function is obviously the correct one. For the others, a linear interpolation between the transforms belonging to the nearest tile-centre pixels is performed as displayed in figure 1. Most pixels lie in the bulk of the image and are surrounded by four tile centre pixels. They are transformed with the corresponding four different transformation functions and interpolated bilinearly to obtain the result pixel value. Pixels on the image margins are linearly interpolated between transformation results from the two neighbouring centre pixel transformation functions. Finally, corner pixels are not interpolated at all but simply transformed with the transformation function of their tile.

2.3 Excess redistribution without recursion

The key element of CLAHE is the limiting of the contrast amplification by clipping the histogram before computing the transformation function. The conventional way of doing this is by the following recursive procedure [PAA+87]. The histogram is clipped at the predefined clip limit, which is a parameter of CLAHE. The total excess from those bins that exceed the clip limit is distributed equally over all histogram bins, as depicted in figure 2. This once again pushes some bin counts over the clip limit (figure 2 (b)). The resulting excess is redistributed again, and the procedure is repeated until the limit is not exceeded by any bin any more.

Implementing recursion in hardware can be complex, necessitating the implementation of control flow and of storage for intermediate results, and time-consuming, as recursions are performed sequentially. In particular, in our case each recursion requires a complete pass over the histogram RAM even if redistributing the excess and determining the excess for the next round are combined [Rez04]. This means that the time for the whole redistribution procedure is of the order of the number of iterations times the number of histogram bins.

Fortunately, the redistribution can be well approximated by a single-step procedure. It is based on the observation that the histogram bins that initially exceed the clip limit will again do so in every recursion. This is because those bins are clipped to exactly the clip limit, so every redistribution step will push them over the limit again. What is more, only a few additional bins will be clipped in the later recursions, namely those that do not initially exceed the limit but are close enough to it that a redistribution step makes them exceed it. The single-step redistribution procedure then consists of redistributing the initial excess only once, and only among the bins that do not initially exceed the limit, as displayed in figure 2 (c). The relatively few bins that are pushed over the limit are clipped again, and the secondary excess, marked by red crosses in figure 2 (c), is discarded.

2.4 Validity of the approximation

The following calculation will estimate the error incurred by discarding the secondary excess. To that end, we will utilize the continuous formulation introduced in the first part of section 2.1. Since the histograms of IR images consist of one or several peaks (see figure 3), a Gaussian function will be used as an exemplary continuous histogram.

It seems unlikely that the precise shape of the peak would influence the error introduced by the modified

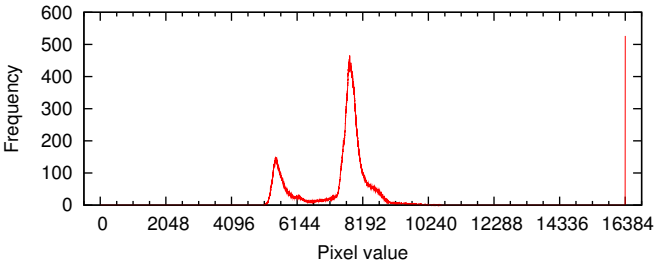

Figure 3. Example histogram of a 14-bit infrared image. This histogram corresponds to the scene displayed in figure 9. The narrow peak to the right represents defective pixels which always take the maximum value.

redistribution procedure to a significant extent. The fact that the Gaussian is a symmetric function does not play a part. By combining two halves of Gaussians with different σ, one can approximate an asymmetric peak. The discarded excess for the resulting function lies between that for the two Gaussians. A histogram with several peaks can be constructed similarly, by putting two Gaussians next to each other.

So in the following, the image PDF is assumed to be a Gaussian function centred on $\frac{1}{2}$:

$$p(x) = \mathcal{N} \exp\left(-\frac{(x-\frac{1}{2})^2}{2\,\sigma^2}\right), \tag{3}$$

$$\mathcal{N} = \left[\int_0^1 \frac{p(x)}{\mathcal{N}}\,\mathrm{d}x\right]^{-1} = \left[\sqrt{2\pi}\,\sigma\,\operatorname{erf}\left(\left(2\sqrt{2}\,\sigma\right)^{-1}\right)\right]^{-1}$$

Here, erf is the error function, the normalized integral over the Gaussian, defined as follows [AS64, GR00]:

$$\operatorname{erf} x = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-t^2\right)\,\mathrm{d}t$$

As the first step, we have to compute the locations x_1 and x_2 where the histogram crosses the clip limit ℓ. Because p is centred on $\frac{1}{2}$, $x_{1/2}$ have to be symmetrical with respect to $x = \frac{1}{2}$. The result is:

$$x_{1/2} = \frac{1}{2} \pm x_0(\ell) = \frac{1}{2} \pm \sigma\sqrt{2\,\log\frac{\mathcal{N}}{\ell}} \tag{4}$$

If ℓ is larger than the maximum of the PDF, $\mathcal{N}$, the term under the square root becomes negative, and no real solutions exist because the histogram is everywhere below the limit, $p(x) < \ell\ \forall x$. Otherwise, the primary excess to be redistributed is

$$\begin{aligned} E(\ell) &= \int_{x_1}^{x_2} \left(p(x) - \ell\right)\mathrm{d}x \\ &= 2\int_0^{x_0(\ell)} p(x+\tfrac{1}{2})\,\mathrm{d}x - 2\,x_0(\ell)\,\ell \\ &= \frac{\operatorname{erf}\sqrt{\log(\mathcal{N}/\ell)}}{\operatorname{erf}\left(\left(2\sqrt{2}\sigma\right)^{-1}\right)} - 2\,x_0(\ell)\,\ell \end{aligned} \tag{5}$$

This amount is redistributed to the regions outside the interval $[x_1, x_2]$, so the PDF is increased in these regions by the constant

$$S = \frac{E(\ell)}{1 - 2\,x_0(\ell)} \tag{6}$$

The quantity we want to compute is the fraction of the histogram that we ignore by discarding the secondary excess after redistribution. Because our PDF is normalized, this is equal to the integral of the PDF over the discarded regions. These are the regions above the clip limit between the intersection of the shifted PDF with the limit and the old intersection, as indicated in figure 2 (c). The new intersection points are the same as the points at which the original PDF intersects the limit $\ell - S$. They are therefore given by (4) with ℓ replaced by $\ell - S$. So the amount neglected by the single-step redistribution procedure is

$$\begin{aligned} D &= 2\int_{x_0(\ell)}^{x_0(\ell-S)} \left(p(x+\tfrac{1}{2}) - (\ell - S)\right)\,\mathrm{d}x \\ &= \frac{\operatorname{erf}\sqrt{\log(\mathcal{N}/(\ell-S))} - \operatorname{erf}\sqrt{\log(\mathcal{N}/\ell)}}{\operatorname{erf}\left(\left(2\sqrt{2}\sigma\right)^{-1}\right)} \\ &\quad - 2\,\sigma\left(\sqrt{\log(\mathcal{N}/(\ell-S))} - \sqrt{\log(\mathcal{N}/\ell)}\right)(\ell - S) \end{aligned} \tag{7}$$

Some values of D for typical parameter values are displayed in table 1. ℓ is necessarily larger than 1, since that is the average of the normalized PDF. On the other hand, if ℓ is chosen much larger than 1, it exceeds the height of the PDF peak, and no clipping happens. Table 1 shows cases with a reasonable value of ℓ where the peak is high enough (σ is small enough) that clipping occurs. As the table shows, the secondary excess is less than one percent of the total in all cases.

Table 1. Error introduced by neglecting the secondary excess in figure 2 (c) for a Gaussian PDF.

σ	ℓ	D
0.05	4	0.001
0.05	3	0.003
0.1	3	$5.7 \cdot 10^{-4}$
0.1	2	0.006

In addition to the above general derivation, the single-step redistribution procedure was tested on real-life images. The images were IR images available at the author's institute as well as standard test image series from the Signal and Image Processing Institute of the University of Southern California[1] and imagecompression.info[2]. The latter is available in a bit depth of 16 bits. The colour images from USC-SIPI were converted to greyscale before processing. Five images from the USC-SIPI Miscellaneous series and one image from imagecompression.info were left out because they do not depict natural scenes. The results are shown in table 2. They have the same order of magnitude as the results

1. http://sipi.usc.edu/database/index.html
2. http://www.imagecompression.info/test_images/

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing

Table 2. Error introduced by neglecting the secondary excess in figure 2 (c), averaged over various test image series.

Source	Series	Bit depth	Number of images	$D\ (\ell = 3)$	$D\ (\ell = 4)$
This author	IR	14	11	0.0017	$8.7 \cdot 10^{-4}$
imagecompression.info	Gray 16 bit	16	14	$9.5 \cdot 10^{-4}$	$5.7 \cdot 10^{-4}$
USC-SIPI	Aerials	8	38	0.001	$2.6 \cdot 10^{-4}$
USC-SIPI	Miscellaneous	8	39	0.001	$4.2 \cdot 10^{-4}$

of the formal calculation and are consistently better for $\ell = 4$.

2.5 Preprocessing

In this implementation, the actual histogram equalization is preceded by a preprocessing step. This is commendable due to some characteristics of the IR images that are to be processed. Because of the challenges involved in manufacturing IR imaging sensors, such sensors are not perfect but contain a significant number of bad pixels which always have the minimum or maximum grey value. Also, in most situations the occupied range of grey values is smaller than the maximum dynamic range. Both effects can be seen in the example histogram in figure 3. These effects would reduce the dynamic range of the result image if histogram equalization were performed immediately. In attempting to stretch the bad pixels accumulated at either end of the input dynamic range, CLAHE would create a gap next to the minimum or maximum value. In addition, the contrast limiting would prevent the empty regions between the minimum or maximum and the occupied range from being squashed, causing the corresponding region of the output range to be wasted.

Both these problems are solved by applying a windowing transformation to the grey values before performing histogram equalization: Based on the global histogram of the image, a certain share of the largest and smallest values are discarded as bad pixels. The remaining pixels are scaled linearly to occupy the full range of an intermediate value, which is then input into the CLAHE step. The share of pixels to discard is a characteristic of the camera, which has to be determined manually.

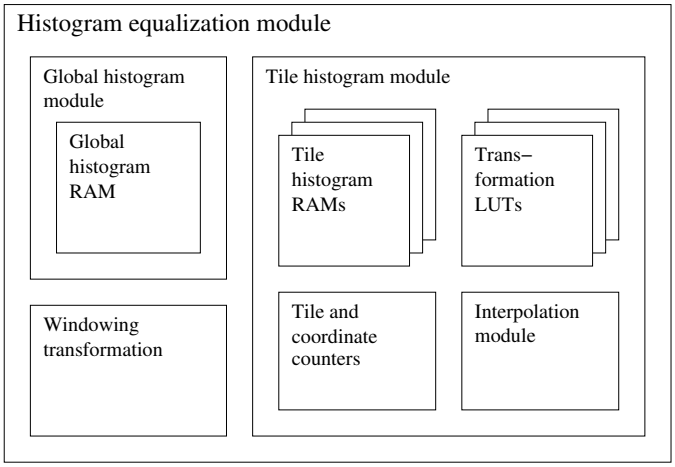

Figure 4. Structural overview of the CLAHE implementation

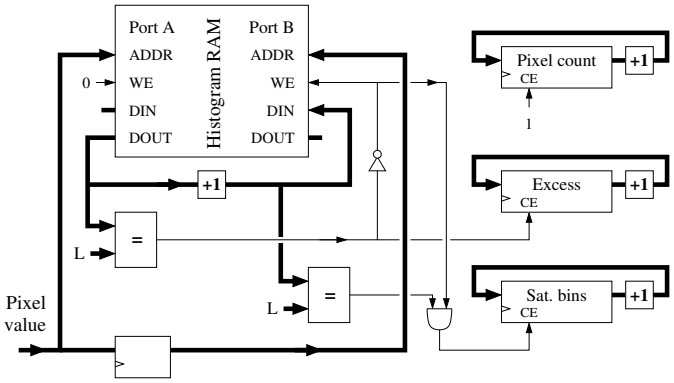

Figure 5. Simplified schematic of the tile histogram generation circuit. The input data valid and in-range signals that qualify all write operations are omitted. Histogram bins are incremented until they saturate. L stands for the clip limit. The total pixel count, excess count and number of saturated bins are generated simultaneously with the histogram.

3 Implementation

3.1 General

This and the following sections will concentrate on new aspects of the implementation and describe its structure very briefly. Readers aiming to reproduce the implementation should refer to the appendix and to the description of the hardware implementation in [Rez04].

The structure of the CLAHE implementation is displayed in figure 4. Two submodules are instantiated, which handle the global histogram for the preprocessing step and the tile histograms and transformations, respectively. The global histogram submodule consists primarily of the global histogram RAM, in which the histogram is built up as pixels are available. After the end of each frame, the histogram is accumulated both from the bottom and from the top until the prescribed fraction of bad pixels is exceeded. The histogram bins where this happens are taken as the limits of the range of pixel values to be processed further with the CLAHE method.

In the top module, the global linear transformation of the preprocessing step is performed on the basis of the range from the global histogram submodule. Pixels that are outside the range are marked as such.

The tile histogram submodule contains most of the CLAHE implementation. It instantiates two sets of

RAMs, for the histograms and transformation functions of each image tile. One submodule contains coordinate counters which track the tile and coordinate within the tile of the current pixel. Another submodule performs the bilinear interpolation between the four transformation results as described in section 2.2.

The input data of the tile histogram module are the intermediate pixel values resulting from the preprocessing step. As in the case of the global histogram, the tile histogram bins are incremented as pixels arrive at the module (see figure 5). Only the pixel values inside the range determined in the preprocessing step are taken into account. The clipping is implemented simply by stopping incrementing once a bin count has reached the limit. Besides avoiding a separate clipping pass over the histogram, this has the additional benefit that the width of the histogram RAM can be reduced significantly, as the maximum value that has to be stored is the clip limit. Simultaneously to histogram generation, a total pixel count and an excess count are computed for each tile. The former is counted up on every in-range pixel, the latter only if the corresponding bin is not incremented because it has saturated. A further per-tile counter keeps track of the number of bins that have saturated. The total number of bins minus the end result of that counter gives the number of bins among which the excess has to be redistributed (see section 2.3).

In this way the clipping is performed in the same process as the creation of the histograms, and all data necessary for redistribution are available when the histogram is complete. Only one additional pass over the histogram is necessary to redistribute the excess and compute the transformation function. This is performed for each row of image tiles in parallel. First the amount to be redistributed to each bin is computed. Then each bin count is increased by that amount, clipped to the clip limit and added to the sum of previous bin counts. This is done indiscriminately to all the bin counts — for those that are already saturated, the clipping undoes the effect of the redistribution. The partial sum is then multiplied with a normalization factor[3] to obtain the transformation function value, which is stored in the look-up table.

Finally the interpolation submodule transforms each intermediate pixel value according to the transformation functions corresponding to the four nearest tile centre pixels. The tile counter submodule provides the tile numbers and the distances from the centre pixels. Linear interpolation is performed in the horizontal and vertical direction as required (cf. section 2.2 and figure 1). Pixel values that are outside the range of the initial windowing bypass the whole transformation process and are set to the minimum or maximum output value, respectively.

3. The normalization factor is derived from the pixel count excluding bad pixels but including the secondary excess, which slightly biases the transformation function towards smaller pixel values. This could be corrected by deferring normalization until the transformation functions are used, when the secondary excess is known.

3.2 Minimizing latency

To reduce the latency of this CLAHE implementation, two means are employed: performing different steps of the algorithm in parallel on the same frame and making use of the vertical blank interval between frames.

The preprocessing step requires knowledge of all pixel values before the appropriate linear transformation can be determined, which gives it a latency of a whole frame. The CLAHE transformation of a pixel may require all the pixel values of the tile below and to the right of its own, resulting in a latency of $1\frac{1}{2}$ times a tile row. This is not acceptable for our purposes. The problem is solved by using the data range from the previous frame for the windowing transformation in the preprocessing step and by transforming pixel values with the tile transformation functions from the previous frame. This allows to perform the three steps of which the algorithm consists — value range determination, tile histogram generation and transformation of pixel values — in parallel on the current frame, as shown in figure 6. The total latency of the transformation path (bottom of figure 6) is nine clock cycles.

The validity of this procedure is based on the assumption that the content of successive frames and of corresponding tiles in them is similar. This assumption was verified in IR sequences available at the author's institute and surveillance video sequences[4] from the conference PETS2007 [Fer07]. The tile histograms of successive frames were computed, and their normalized RMS differences were found to be of the order of 10^{-3}. The following factors can affect the validity of the assumption: Other things being equal, a high camera frame rate and a large field of view are beneficial because they reduce differences between frames. Fast panning of the camera may be detrimental, but is also likely to degrade image quality through motion blur.

Those operations that do not process pixel values as they arrive — finding the value range of the global histogram and generating the tile transformation functions — are designed to be performed during the vertical blank gap between successive frames. Their results are available for processing of the first pixel of the following frame. This places a constraint on the duration of the vertical blank interval of the camera used. The number of pixel clock cycles taken up by the vertical blank must be at least

$$T_{\mathrm{vb}} \quad \geq \quad \max\left\{B_{\mathrm{G}} + 72\,,\ (35 + B_{\mathrm{T}})\,N_{\mathrm{T,x}} + 39\right\}\,, \quad (8)$$

where B_{G} and B_{T} are the number of bins of the global histogram and of the tile histograms, respectively, and $N_{\mathrm{T,x}}$ is the number of tile columns. The cameras with which the CLAHE module is to be used fulfil this requirement.

4. Available from http://www.cvg.rdg.ac.uk/datasets/index.html

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing

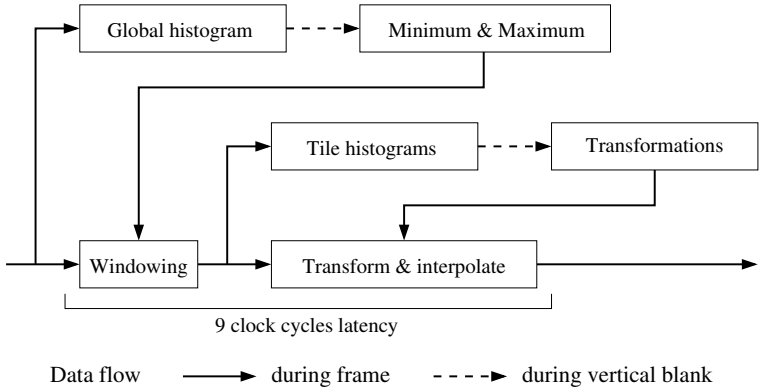

Figure 6. Data flow graph of the CLAHE implementation. Paths represented by dashed arrows do not affect the latency of the transformation data path at the bottom.

3.3 Choice of design parameters

This implementation of CLAHE is systematically parametrized by constants in the top module which are propagated to the submodules through the use of VHDL[5] generics. The most fundamental parameters are the width and height of the frames, the number of tiles in the horizontal and vertical direction, and the input and output bit widths. In addition, the range determination for the preprocessing step has two parameters, the amount to discard at each end of the global histogram. These are given as real-valued parameters interpreted as the share of the total number of pixels to discard. By not choosing the explicit number of pixels as parameters, they are made independent of the frame size, which determines the total number of pixels.

The last but not least parameter is the CLAHE clip limit. This is also specified in a way which makes it independent of the other parameters, in the following sense: If the accuracy of the input image data is modified (by changing the image size or input data bit width), the same clip limit parameter should still produce the same output image (within the limits of numerical precision). This is achieved by dividing the clip limit by a quantity proportional to the average bin count, as follows.

The tile histogram total approximately equals the number of pixels in the tile, so the average bin count is proportional to it. The average bin count also is inversely proportional to the number of bins and therefore to the number of different input data values of the tile histogram module. So we choose the clip limit parameter ℓ as

$$\ell \;=\; \frac{B_\mathrm{T}}{N}\,L = \frac{2^{\xi'}}{N}\,L\,, \tag{9}$$

where B_T is the number of bins of the tile histograms, ξ' the bit width of the intermediate pixel values after preprocessing, N the number of pixels in the tile and L the actual limit used to clip the tile histograms. The parameter ℓ is the same variable as has been used for the continuous-valued computations in section 2.4 and can be viewed as the maximum slope of the continuous transformation function. The implementation computes L from the real-valued ℓ at compile time using VHDL constants.

5. VHDL is one of the major hardware description languages. It stands for "VHSIC hardware description language", where VHSIC stands for "very high speed integrated circuit".

3.4 Some details of the design

This section presents some details of the CLAHE implementation which help ensure numerical accuracy while not wasting any resources.

Histogram bin size Minimizing the number of histogram bins saves storage space for the histogram and time for accumulating the histogram values. In this implementation, the number of tile histogram bins is chosen as four times the number of possible output pixel values. This introduces an error of up to $\ell/4$ of the least significant bit in the output pixel values. For reasonable values $\ell \leq 4$, this is no worse than the precision of the output data. Importantly, the errors do not add up across bins because a partial sum over whole bins is always exact.

Similarly, the global histogram has four times as many bins as the number of distinct values for the intermediate pixel values, 16 times the number of output pixel values. This results in an inaccuracy for the minimum and maximum of the global value range, but it is small compared to the size of that range. The minimum and maximum values are rounded towards the middle of their bins.

To sum up, the histogram bin sizes are

$$\begin{array}{lll} B_\mathrm{T} & = 2^{\xi'} & = 2^{\eta+2} \\ B_\mathrm{G} & = 2^{\xi'+2} & = 2^{\eta+4} \end{array} \tag{10}$$

Intermediate quantities Three parts of our algorithm require a division: the windowing of the preprocessing step, the computation of the excess to be redistributed per eligible bin, and the computation of the transformation function normalization factor. In order to save resources while maintaining accuracy, these divisions

were replaced by multiplications with the inverse of the denominator, which is computed only once during the vertical blank gap. So as not to introduce numerical errors, the bit widths of these inverses and other intermediate quantities are computed at compile time. This is done by value range propagation based on the bit widths of the histogram inputs and the output data.

Exploiting dual-port RAMs Most of today's FPGA architectures offer dual-port on-chip RAM blocks. Their most important application in this CLAHE implementation is in generating the histograms at a throughput of one pixel per cycle, as displayed in figure 5. One port is used for reading the current bin count, the other for writing the new count after incrementing. Writing the new count and reading the (usually different) next bin to be incremented is done simultaneously. The existence of two RAM ports also allows to save storage space by packing all tile histograms of one tile row into one set of RAM blocks. Every two of the four transformation results needed for interpolation come from two ports of the same RAM.

3.5 Resource usage

The histogram equalization module was synthesized, placed and routed for five different FPGA architectures, the Virtex-II Pro, Virtex-4, Virtex-5, Spartan3 and Spartan6, all made by Xilinx. The three Virtex chips are members of the high-end FPGA series from Xilinx that are in use at the author's institute. The Virtex-II Pro in particular was used for testing and demonstrating the histogram equalization module (see next section). The Spartan6 and Spartan3 are the newest and previous generations of Xilinx low-cost FPGAs.

To determine its resource requirements, the module alone was synthesized without any logic interfacing a camera or data sink. The synthesis software used was Xilinx ISE. The effort level of the mapper and placer and router was set to "high". The clock frequency was slightly overconstrained to determine the maximum frequency that could be achieved.

Table 3 shows the resource usage and maximum clock frequency of the histogram equalization module on the five architectures. Its parameters were a frame size of 640 by 480 pixels (VGA resolution), an input bit width of 14 bits and output bit width of 8 bits, a clip limit of 4, and 8 by 8 tiles. The module fits comfortably on the large Virtex chips, which makes using it as part of a larger image processing chain on an FPGA feasible. The two Spartan devices in table 3 are the smallest of their series on which the module fits. The limiting element is the number of RAM blocks required for the histograms and transformation functions. The Spartan3 4000 is the second largest Spartan3, but the Spartan6 LX45 is medium-sized for its series [Xil09].

That the clock frequencies in table 3 are relatively low is mainly due to multiplication operations on numbers that are wider than the built-in multipliers of the architecture. In addition, the critical paths sometimes contain multiplexers or adders or subtractors. If clock frequency was a concern, additional pipeline stages could be added. However, the pixel clock frequencies of the cameras currently used together with the histogram equalization module range from 9 to 16 MHz, so the frequencies achieved are quite sufficient. Also, the devices chosen for the comparison in table 3 are not the fastest speed grades of their respective series.

figure 7 shows how the resource usage depends on various parameters of the module. The slices, multipliers and RAM blocks used are displayed relative to the resources used by the module with the set of parameters above, which is consequently displayed as 1 in all plots. The parameters that were not varied in each of the comparisons were kept as above. The target device was the Virtex-5 FX130T.

Figure 7 (a) shows that the size of the histogram equalization module depends only moderately on the frame size. Adjacent frame sizes differ by a factor of two in both width and height, and therefore by a factor of four in the number of pixels. That the required resources do not vary much is due to the fact that the pixels are streamed through the module and no part of the image is stored.

The most important parameters on which the resource usage does depend are the output bit width and the number of tiles, as shown in figure 7 (b) and (c). The output bit width determines the width of the transformation function look-up table (LUT); hence the strong dependence of the number of RAM blocks required. The number of tiles has the largest influence on the resource usage. This was to be expected, as large parts of the module exist for each tile: the histogram RAM, the transformation function LUT, and the circuits that generate the histogram and the transformation function.

Lastly, the dependence of required resources on the clip limit is slight, as shown in figure 7 (d). That a small dependence exists at all is due to the fact that the clip limit determines the width of the histogram RAM. The resource usage shows almost no dependence on the input bit width (not shown in figure 7). This is because the size of the global histogram and the bit width of all logic after the initial windowing are determined by the output bit width and are thereby independent of the input bit width.

The results of synthesis runs as those displayed in figure 7 have to be taken with a pinch of salt. There are threshold effects involved in all three types of resources: RAM blocks may be partly utilized, not all bits of a multiplier may be needed, and slices may be underused due to imperfect packing. But that the major resource usage dependencies could be ascribed to features of the histogram equalization module suggests strongly that the results above are valid.

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing

Table 3. Resource usage for the fully placed and routed histogram equalization module, not including any interfacing or auxiliary logic. The parameters of the module are a frame size of 640 by 480 pixels, 14 bits input data width, and 8 by 8 tiles. The small table on the right is to remind the reader that the multipliers and RAM blocks of the Virtex-5 architecture and the slices of the Virtex-5 and Spartan-6 are larger than those of the others.

Device	Virtex-II Pro		Virtex-4		Virtex-5		Spartan-3		Spartan-6	
Part number	xc2vp100-6		xc4vfx100-10		xc5vfx130t-1		xc3s4000-4		xc6slx45-2	
Slices	9045	20%	9037	21%	3691	18%	9130	35%	3396	49%
Multipliers	27	6%	27	16%	18	5%	27	28%	27	46%
RAM blocks	61	13%	61	16%	32	10%	61	63%	61	52%
Max. frequency	72.9 MHz		71.8 MHz		101.3 MHz		38.4 MHz		44.7 MHz	

Odd-one-out resource sizes		
	Virtex-5 + Spartan-6	others
LUTs / slice	4×6 bit	2×4 bit
	Virtex-5	others
Mult. width	18×25	18×18
BRAM size	36 kb	18 kb

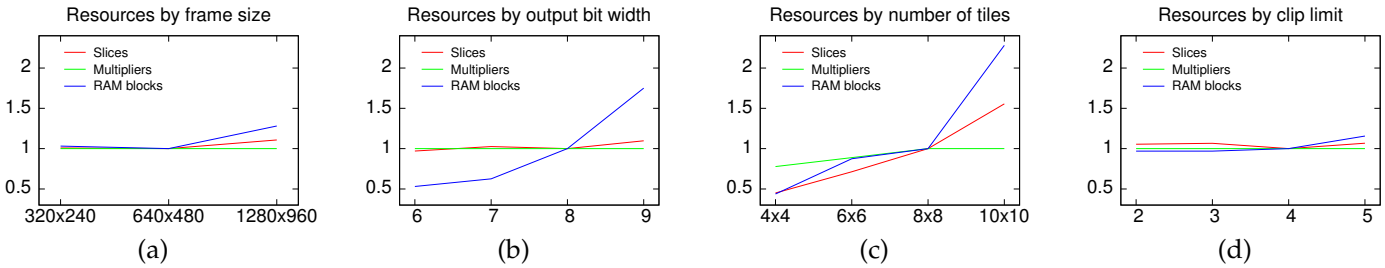

Figure 7. Dependence of the resource usage of the histogram equalization module on the frame size, the output bit width, the number of tiles, and the normalized clip limit ℓ. The plots show the number of slices, multipliers and RAM blocks relative to the resources used for a frame size of 640 by 480 pixels, 8 bit output data, 8 by 8 tiles, and clip limit 4. The input data bit width was always 14 bits. The y axes in all plots are scaled equally for easy comparison. This comparison was performed for the Virtex-5 FX130T.

3.6 Comparison with prior work

There exist two previous hardware implementations of the tile-based CLAHE algorithm, by Reza [Rez04] and Ferguson *et al.* [FAEP08]. Reza's work is aimed at the real-time visualization of medical X-ray images. He is careful to keep the latency of his design within the bounds required for that application and achieves a latency of half a frame interval, typically 1/60 of a second [Rez04]. As the implementation presented in this work in addition envisages use in feedback loops such as object tracking, it was designed to have a latency that is much lower still, of the order of microseconds. It is therefore completely transparent to downstream processing with respect to latency, not just to the human eye.

Ferguson *et al.* have developed a CLAHE implementation for video streams with a bit depth of 8 bits per pixel. In order to minimize size and power consumption, they sacrifice quite a bit of accuracy. The tile histograms are limited to 8 bins, and it is not discussed whether and how the resulting 8-step transformation function LUTs are interpolated to utilize the 8-bit range of result values. The implementation given above does not trade off size for accuracy and therefore can serve as an accurate baseline. As a consequence it is considerably larger than that presented in [FAEP08].

This work improves upon both prior implementations in several ways. The single-step redistribution procedure presented in section 2.3 is novel and allows a more efficient hardware implementation by avoiding recursion. Unlike prior work, this design is extensively parametrized, which allows adapting it to different situations without touching the HDL code. In particular, it can be synthesized to process input data with an arbitrary bit depth. Finally, the preprocessing step makes the implementation tolerant of faulty pixels and increases accuracy if not all of the input data range is used.

4 Test system

4.1 Overall design

In order to be able to test and demonstrate the CLAHE module, it was implemented together with a camera interface on a Xilinx Virtex-II Pro FPGA. The design consists of the camera interface, the histogram equalization module, a FIFO buffering the transformed images and a local bus interface. The local bus data transfers are translated to PCI bus transfers by a bridge chip on the FPGA card. The image data are then displayed by a program running on the host PC. The FPGA card used was an ADM-XP board from the company Alpha Data [Alp04].

Because the CLAHE module has a throughput of one pixel per clock cycle, it can run with the same clock as the camera interface, and buffering the raw camera data is not required. There needs to be only one clock domain transition in the design, between the camera clock and the local bus clock.

The histogram equalization parameters in the test system were as follows. The image size was 640 by 486 pixels, and the input data bit width was 14 bits, as required by the attached camera. The CLAHE clip limit was chosen as 4, and the output bit width was 8. Eight by eight tiles were used. The preprocessing step was configured to discard 1% of the pixels at either end of the global histogram.

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing

(a) (b) (c)

Figure 8. (a) Test setup with the camera AIM 640 QLW. (b) Photograph of the scene viewed. (c) Infrared image of the scene after being processed by the CLAHE module. The missing upper left corner is due to a patch of bad pixels on the sensor.

Table 4. Reconfiguration times for the dynamically reconfigurable CLAHE module and speedup relative to a reconfiguration of the complete Virtex-II Pro FPGA. The results for the first two configuration interfaces were measured, while those for the ICAP controllers were calculated based on the data rates measured in [CZS+08].

Interface	Partial reconfiguration time	Speedup
JTAG	2.5 s	5.2
PCI API	203 ms	1.6
OPB_HWICAP	182 ms	n/a
PLB_ICAP	9.7 ms	n/a

(a) (b)

Figure 9. The results (a) of a manual global contrast adjustment and (b) of the conventional tiled CLAHE with recursive redistribution for an infrared image of the same scene as in figure 8.

4.2 Partial dynamic reconfiguration

In preparation for integration into a reconfigurable image processing toolkit, the CLAHE module in the test design was implemented as a dynamically reconfigurable module.

The Virtex-II Pro FPGA supports one-dimensional partial dynamic reconfiguration [LBM+06]. This means that reconfigurable modules extend over a range of configuration columns. The reconfigurable region containing the histogram equalization module was constrained to the slice columns 0 to 53 and the corresponding block RAM and multiplier columns. This amounts to 28% of the chip area (excluding the space taken up by the PowerPC hard cores). Two alternative dynamic range reduction algorithms were implemented which can replace the histogram equalization module: A global contrast adjustment (the same used for the preprocessing step of the CLAHE module) and a trivial method that merely discards the least significant bits.

The other components of the design — camera interface, FIFO and local bus interface — were wrapped in a static design module for which the remainder of the chip was available. The static and reconfigurable parts communicate through hard-placed LUT-based bus macros, which were developed by Hübner *et al.* [HBB04] and are now provided by Xilinx. The design was synthesized and implemented using the Xilinx ISE command-line tools version 9.2 with the Early Access Partial Reconfiguration patches [LBM+06].

Table 4 shows the time needed for reconfiguring the histogram equalization module using different interfaces. Also shown is the factor by which the configuration time is reduced relative to a reconfiguration of the complete Virtex-II Pro FPGA. The results in the first two rows were measured. They apply to the JTAG interface and the SelectMAP interface accessed via the PCI bus, board hardware and API provided by the board manufacturer. The reconfiguration time for the latter is adversely affected by the fact that the API does not allow PCI bus DMA transfers for partial bit streams. For full bit streams this is possible, resulting in the comparably low speedup factor.

The lower two rows in table 4 show reconfiguration times calculated for the Internal Configuration Access Port (ICAP). These were computed using the throughput measured by Claus *et al.* [CZS+08]. The first relates to the ICAP controller provided by Xilinx, which receives configuration data from an on-chip processor via the On-chip Peripheral Bus (OPB). The last row applies to the ICAP controller presented in [CMZS07], which is attached as a master to the Processor Local Bus (PLB) and can access a memory controller directly. The small reconfiguration time achieved with this approach makes it the preferred choice for a future image processing system.

4.3 Test setup

The histogram equalization design presented above has been operated together with a camera and a visualization program running on the host PC. Figure 8 (a) shows the test setup. The FPGA board was plugged into the server PC on the left. The infrared camera used was of the type AIM 640 QLW and is sensitive to wavelengths between 8 and 10 μm. It has a frame resolution of 640 by 486

pixels and outputs 14 bit data. The pixel clock of its digital interface, with which the histogram equalization module operates, was 16 MHz.

The result of the hardware histogram equalization is displayed in figure 8 (c). For comparison figure 9 shows the result of a manual global brightness and contrast adjustment and the result of a software implementation of CLAHE with recursive excess redistribution. The image in figure 9 is not exactly the same, because the hardware implementation only allowed recording the processed image, but it displays the same scene a short time later. The superior contrast of the CLAHE can be seen in the tree branches, the windows of the building and the round window in the tower.

5 Summary

This report has presented a hardware implementation of the tile-interpolated contrast-limited adaptive histogram equalization method. This implementation has been optimized for latency. Extremely low latency is achieved by executing successive steps — preprocessing, generation of transformation functions, and application of the transformation — in parallel on successive frames. An important novelty is the replacement of the traditional recursive redistribution of the excess from histogram clipping by a single-step procedure. This allows clipping the histogram while accumulating it and thus saving much time and storage space.

An exploratory synthesis has revealed that the histogram equalization module, when used on its own, fits on today's low-cost FPGAs such as the Xilinx Spartan6. When implemented on a large FPGA such as the Xilinx Virtex-5 series, it leaves most of the chip area for other functionality. Its resource usage has been shown to depend primarily on the desired result precision and the number of tiles to use for the algorithm. It depends only slightly on the frame size, the clip limit and the input precision.

The histogram equalization module has been synthesized together with a camera interface for a Xilinx Virtex-II Pro FPGA. It has been shown to be functional by operating it with an infrared camera. The test design incorporates partial dynamic reconfiguration and allows replacing the histogram equalization module by other dynamic range reduction algorithms without reconfiguring the whole FPGA. The histogram equalization module will in due course be integrated into a partially dynamically reconfigurable image processing toolkit that is under development at the author's institute.

Acknowledgements

The author thanks Diana Göhringer for vital tutoring in the use of the Xilinx Partial Reconfiguration design flow. Furthermore, he thanks Diana Göhringer and Thomas Perschke for proofreading the manuscript, and Lars Braun for helping overcome our FPGA board's resistance to partial bit streams.

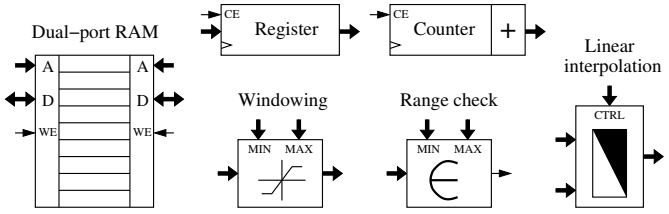

Figure 10. Key to the component symbols used in figure 11 and 12. The windowing component maps the valid input data range to the complete output data range. The range check component outputs a logical 1 if the input value is in the valid range.

Appendix

This appendix presents an overall functional description of the CLAHE implementation and shows how the components described in the main part of the paper work together. Readers intending to reproduce the design should also consult [Rez04], where an implementation of the unmodified CLAHE algorithm is described in great detail.

Figure 11 displays a functional schematic of the operations performed while pixel values arrive at the module. Thick lines represent numerical quantities, while thin lines are boolean signals. Figure 10 defines the utilized components. As pixel values arrive, they are added to the global histogram necessary for the range determination of the preprocessing step. The windowing transformation based on the range from the previous frame is applied to the values. The range check component determines which pixel values are within the valid range. Its output prevents bad pixels from being processed by the CLAHE.

In the right part of the figure, the pixel value is added to the appropriate tile histogram. This task is performed by the circuit displayed in figure 5 the operation of which is qualified by the data valid and in-range signals. Simultaneously, it is transformed using transformation functions computed from the tile histograms of the previous frame. The transformation results are then interpolated to produce the result pixel value. See below for how the right tiles are selected.

The dotted boxes in figure 11 represent submodules within which other operations are performed during the inter-frame gap. These operations are shown in figure 12. The pixel value counters in that figure start from zero and count up to the maximum once during each vertical blank interval. The circuit on the left accumulates the global histogram and keeps updating the range registers until the corresponding boundary of the valid range has been passed. The quantities BadPixLow and BadPixHigh are the numbers of pixels at the bottom and the top of the histogram to discard as bad pixels of the camera.

The submodule on the right which contains the tile histograms and transformation functions operates only on pixels which are within the range of the preprocessing step. L is the clip limit. The registers for the number of in-range pixels, the excess and the number of satu-

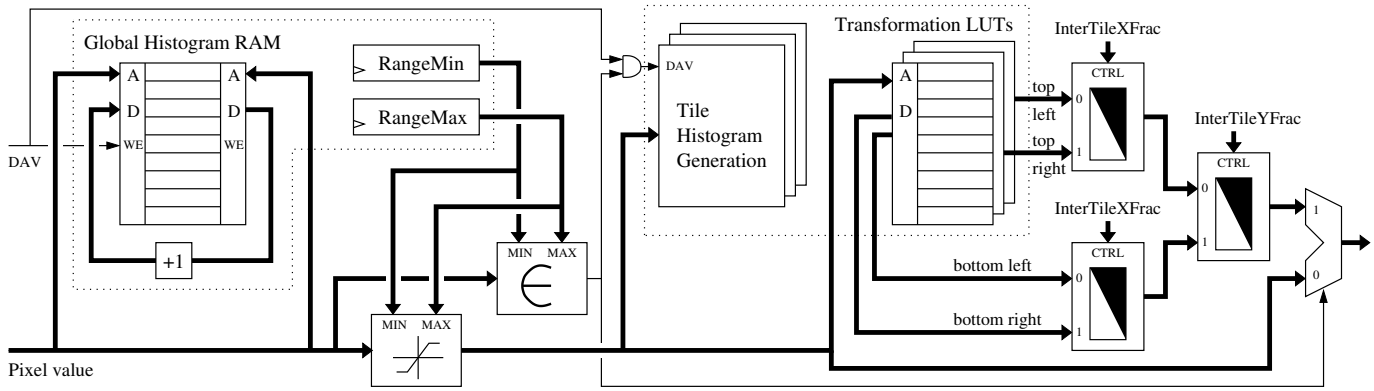

Figure 11. Operation of the CLAHE circuit while pixel data arrive during the frame. See figure 5 for the tile histogram generation circuit.

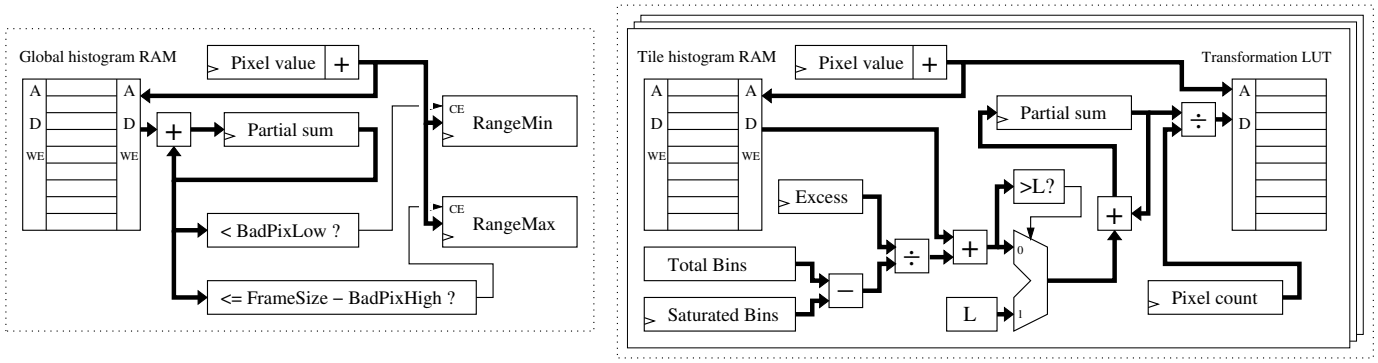

Figure 12. Determination of the valid pixel range and generation of the tile transformation functions during the vertical blank gap. The dotted boxes represent the same submodules as in figure 11. A copy of the circuit on the right exists for each tile.

rated bins are provided by the tile histogram generation circuit, see figure 5. The amount to be redistributed per bin is computed, it is added to the bin value, and that value is clipped again. The result is accumulated, normalized by division by the pixel count and written to the transformation LUT.

Selecting the right tiles for tile histogram generation and interpolation requires auxiliary quantities that are derived from the pixel coordinates. These are more concisely expressed as formulas than block diagrams. The horizontal tile indices are derived from the X coordinate as follows:

$$\begin{aligned} \text{HistTileX} &= X/\text{TileWidth} \\ \text{InterTileLeft} &= (X - \text{TileWidth}/2)/\text{TileWidth} \\ \text{InterTileRight} &= \text{InterTileLeft} + 1 \\ \text{InterTileXFrac} &= (X/\text{TileWidth}) \bmod 1.0 - 0.5 \end{aligned} \tag{11}$$

HistTileX is the horizontal index of the tile histogram to which a pixel value is added. InterTileLeft and InterTileRight are the column indices of the tile transformation functions to use for interpolation. All divisions in the first two formulas are truncating integer divisions. InterTileXFrac is the fractional horizontal interpolation coefficient. Vertical tile indices and the vertical interpolation coefficient are computed analogously from the Y coordinate. Pixels in the image margins or corners have to bypass one or both of the interpolation steps, see figure 1 in section 2.2.

REFERENCES

∼ confer • reference ⇿ related ◁ background

[AA05] ⇿ A. M. Alsuwailem & S. A. Aishebeili: *A new approach for real-time histogram equalization using FPGA*. In Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems. Dec 2005 pp. 397–400.

[Alp04] ◁ Alpha Data Parallel Systems Ltd.: ADM-XP data sheet. 2004. Online.

[AS64] • M. Abramowitz & I. A. Stegun: Handbook of Mathematical Functions. Dover Publication, New York. 1964. ISBN 0-486-61272-4.

[BGP+09] ◁ L. Braun, D. Göhringer, T. Perschke et al.: *Adaptive real-time image processing exploiting two dimensional reconfigurable architecture*. Journal of Real-Time Image Processing **4** (2) (2009) 109–125. doi:10.1007/s11554-008-0095-8.

[CMZS07] ◁ C. Claus, F. H. Müller, J. Zeppenfeld & W. Stechele: *A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration*. In 21st IEEE International Parallel and Distributed Processing Symposium, Reconfigurable Architectures Workshop. 2007 pp. 1–7.

[CZS+08] ◁ C. Claus, B. Zhang, W. Stechele et al.: *A multi-platform controller allowing for maximum dynamic partial reconfiguration throughput*. In 2008 International Conference on Field Programmable Logic and Applications. 2008 pp. 535–538.

[DND+01] ⇿ G. Dongsheng, Y. Nansheng, P. Defu et al.: *DSP+FPGA-based real-time histogram equalization system of infrared image*. In Society of Photo-Optical Instrumentation Engineers (SPIE) Conference 2001, vol. 4602. 2001 pp. 160–165. doi: 10.1117/12.445721.

[EPA90] ⇿ J. P. Ericksen, S. M. Pizer & J. D. Austin: *MAHEM: A Multiprocessor Engine for Fast Contrast-Limited Adaptive Histogram Equalization*. In SPIE Vol. 1233 Medical Imaging IV: Image Processing. 1990 pp. 322–333.

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing

[FAEP08] ∼ P. D. Ferguson, T. Arslan, A. T. Erdogan & A. Parmley: *Evaluation of contrast limited adaptive histogram equalization (CLAHE) enhancement on a FPGA*. In SoCC. 2008 pp. 119–122.

[Fer07] ◁ J. M. Ferryman (ed.): Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2007). IEEE Computer Society. October 2007. Online.

[GR00] • I. S. Gradshteyn & I. M. Ryzhik: Table of integrals, series and products. Academic Press, 6th edn. 2000. ISBN 0-12-294757-6.

[GW02] • R. C. Gonzalez & R. E. Woods: Digital Image Processing. Prentice-Hall, second edn. 2002.

[HBB04] ◁ M. Hübner, T. Becker & J. Becker: *Real-time LUT-based Network Topologies for dynamic and partial FPGA Self-Reconfiguration*. In 17th Symposium On Integrated Circuits And Systems Design (SBCCI04). Sep 2004 pp. 28–32.

[Hum75] ◁ R. A. Hummel: *Histogram modification techniques*. Computer Graphics and Image Processing **4** (1975) 209–224.

[Hum77] ◁ R. A. Hummel: *Image Enhancement by Histogram Transformation*. Computer Graphics and Image Processing **6** (1977) 184–195.

[JCR03] ⇿ I. J. Jeon, B. M. Choi & P. K. Rhee: *Evolutionary Reconfigurable Architecture for Robust Face Recognition*. In International Parallel and Distributed Processing Symposium. IEEE Computer Society, Los Alamitos, CA, USA. 2003 p. 192a. doi:10.1109/IPDPS.2003.1213356.

[KLW74] ◁ D. J. Ketcham, R. W. Lowe & J. W. Weber: *Image enhancement techniques for cockpit displays*. Tech. rep., Hughes Aircraft. 1974.

[KLW76] ◁ D. J. Ketcham, R. W. Lowe & J. W. Weber: *Real-time image enhancement techniques*. In Seminar on Image Processing, pp. 1–6. Hughes Aircraft. 1976.

[Kur91] ⇿ C. W. Kurak: *Adaptive Histogram Equalization: A Parallel Implementation*. In Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems. 1991 pp. 192–199.

[LBM+06] ◁ P. Lysaght, B. Blodget, J. Mason, J. Young & B. Bridgford: *Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs*. In 2006 International Conference on Field Programmable Logic and Applications. 2006 pp. 12–17.

[LNC+98] ⇿ X. Li, G. Ni, Y. Cui, T. Pu & Y. Zhong: *Real-time image histogram equalization using FPGA*. In L. Zhou & C.-S. Li (eds.), Society of Photo-Optical Instrumentation Engineers (SPIE) Conference 1998, vol. 3561. Aug 1998 pp. 293–299.

[NEM09] ◁ *Digital Imaging and Communications in Medicine (DICOM) Standard, Part PS 3.14*. 2009. Online.

[PAA+87] • S. M. Pizer, E. P. Amburn, J. D. Austin et al.: *Adaptive Histogram Equalization and Its Variations*. Computer Vision, Graphics, and Image Processing **39** (1987) 355–368.

[Piz81] ◁ S. M. Pizer: *Intensity mappings for the display of medical images*. In Functional Mapping of Organ Systems and Other Computer Topics. Society of Nuclear Medicine. 1981.

[Rez04] ∼ A. M. Reza: *Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for Real-Time Image Enhancement*. Journal of VLSI Signal Processing **38** (2004) 35–44.

[SS99] ⇿ Z. Salcic & J. Sivaswamy: *IMECO: A Reconfigurable FPGA-based Image Enhancement Co-Processor Framework*. Real-Time Imaging **5** (6) (1999) 385–395. doi:10.1006/rtim.1998.0134.

[Xil09] ◁ Xilinx Inc.: Spartan-6 Family Overview. 2009. Online.

Author-generated version of DOI 10.1007/s11554-011-0204-y published in the Journal of Real-Time Image Processing