2016•2017
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef
Lossless compression of RAW image data on the FPGA

Promotor :
Prof. dr. ir. Luc CLAESEN

Copromotor :
De heer Wout SWINKELS

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

Arno Libert

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

2016•2017
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterproef

Lossless compression of RAW image data on the FPGA

Promotor :
Prof. dr. ir. Luc CLAESEN

Copromotor :
De heer Wout SWINKELS

Arno Libert

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Foreword

Data traffic in multi-camera video- and vision systems is a growing concern since the number of pixels grows with every new camera model. The bandwidth and storage space needed to process images from multi-camera systems need to be high-tech or the data must be altered to allow more efficient processing. This is where data compression enters the picture. Numerous compression methods have been developed and perfected for commercial use. These methods are often based on the human perception of image quality. Data that is not essential for these images will be deemed unnecessary and ignored.

Video and vision systems require all information captured by their cameras to calculate the best image. Lossless image compression methods offer a solution but these methods often yield lower compression ratios. The best ratio is hence chosen according to the required specification selected by the developer. Lossless compression methods often require intensive calculations which results in a slow process that is difficult to implement on hardware platforms.

Most commercial compression methods perform their compression on RGB data. The RGB recalculation ignores noise and simply calculates it into the final image to denoise it later. Noise will take up extra storage space and bandwidth and must be removed prior to further compression.

During the creation of this work I learned many things about noise, compression and the difficulty to implement it on hardware.

Finally, I like to thank my professors without whom this work would not have been possible and my friends without whom this work would have been finished 3 months ago.

“Normal people don't understand this concept; they believe that if it ain't broke, don't fix it. Engineers believe that if it ain't broke, it doesn't have enough features yet.” - Scott Adams

Table of contents

List of abbreviations

RGB	Rood-Groen-Blauw, Red-Green-Blue
CoSenS	Computational Sensor Systems
PCA	Principle Component Analysis
PSNR	Peak Signal Noise Ratio
RLE	Run-Length Encoding
DPCM	Differential Pulse-Code Modulation
GAP	Gradient Adjusted Prediction
CALIC	Context-based Adaptive Lossless Image Coding
BBP	Bits Per Pixel
CFA	Color Filter Array
LMMSE	Linear Minimum Mean Square-error Estimation
MSE	Means Square-Error
PDS	Primary Difference Signals

Abstract

De CoSenS onderzoeksgroep aan de UHasselt focust zich op het toepasbaar maken van multicamerasystemen. Deze systemen genereren enorme hoeveelheden videodata. De belasting om gegevens samen te voegen valt op één verwerkingssysteem. De algemene oplossing voor dit probleem is een datacompressie. Om de coherentie tussen de beelden van verschillende cameras te kunnen bepalen is het wenselijk dat de compressiemethodes verliesvrij zijn. Eerst onderzoekt en vergelijkt deze masterproef verschillende methoden om videobeelden te comprimeren. Nadien volgt de implementatie van de meest toepasbare methode. De gebruikte methode moet een compressieratio van minstens 60% behalen en de beelden moeten aan een snelheid van minstens 40 Mb/s verwerkt kunnen worden.

Het onderzoek focust zich op verliesvrije methoden voor ruwe (RAW) en Rood-Groen-Blauw (RGB) beelden. De filtering van ruis uit de beeldinformatie wordt onderzocht. Criteria voor compressiemethoden zijn: snelheid, nauwkeurigheid, verlies en compressieefficientie. De implementatie gebeurt op het Altera DE2-70 bord met softwareversies Quartus II 8.1 en NIOS II in verilog. De resultaten worden geanalyseerd met Matlab R2016b. TRDB D5M is de gebruikte camera.

De geïmplementeerde methode maakt gebruik van Rice encoding met datapreparatie in de vorm van het GAP algoritme. Na calibratie om de ruis in te perken vindt er een compressie plaats op RAW data. Deze methode levert een compressieratio op van 55% aan een snelheid van 44 Mb/s. De methode is nog niet geoptimaliseerd en zou voor beelden in een normaalverlichte ruimte een ratio van onder de 50% kunnen behalen.

The CoSenS research group at UHasselt focusses on the implementation of multi-camera systems. These systems generate a large amount of video data. One central processing system calculates all data and becomes heavily loaded. The solution is data compression. To maintain good cohesion visual clues among images the method must be lossless. First, this master thesis evaluates and compares various compression methods on RAW and RGB data. This is followed up by the implementation of the most suitable method. The compression method used must reach a compression ratio of 60% and a processing speed of more than 40 Mb/s.

The research revolves around compression methods for RAW and RGB data. The filtering of noise is also considered. Criteria for compression methods are: speed, accuracy, loss and compression ratio. The implementation is realized on the Altera DE2-70 board with software Quartus II 8.1 and NIOS II using Verilog. The results are analyzed in Matlab R2016b. The camera TRDB D5M captures the video information.

The implemented method utilizes Rice encoding after data preparation using the GAP algorithm. After calibration to reduce noise, the data is compressed in RAW state. The method achieves a compression ratio of 45% at the speed of 44 Mb/s. In the future, this method can be optimized which can result in a compression ratio of above 50% in a normally lit room

1 Introduction

Multi-camera video and vision systems generate an enormous data traffic caused by the growth of the number of pixels with each new camera model. The bandwidth and storage space must grow alongside the number of pixels to prevent problems while processing images. A common solution to reduce the need for large storage capacity of a system is data compression. There exist numerous video compression methods utilized in commercial applications. These methods however often only take the human eye into account when processing the compressed data. Data irrelevant to picture detail visible to humans is then often ignored and discarded.

Multi-camera video and vision systems have need for all information captured for the calculation of the total image. Lossless compression methods are a solution to this problem but result in a lower compression ratio. The compression method needs to be chosen according to the specifications needed by the applications. The implementation of these lossless compression methods on hardware is often problematic and straining the system.

The most commonly used compression methods compress RGB values. The recalculation of RAW to RGB ignores noise from the camera and calculates this faulty data further into the pattern. This results in larger data values requiring storage. The compression method must filter out the noise prior to compression to increase the compression ratio and to improve the coherence between images from different cameras. The need for coherence between images is a requirement to make the calculation of the larger image easier.

The goal of this thesis is to develop a compression method easily implementable on hardware. The work builds on research done by previous students and myself. The compression will take place when the data is in its RAW state prior to RGB conversion. A benefit of RAW data compression is the ability to remove noise before generalization into the digital image. Correlation between pixels will not be influenced by the removal of noise.

The following work describes the method used to find solutions to problems stated previously. First, the theoretical aspects of the compression will be discussed in order of relevance in the compression chain. With each problem stated the chosen design for implementation will be summed up and explained in detail. Third, the implementation process will be presented and tests described. Finally results will be posted. The requirements for the compression method is a compression ratio of 60% at a speed of at least 40 Mb/s and completely lossless.

The creation of this work has been done in various stages. First the literary study has been done, researching and comparing various existing compression methods and algorithms. Second, the most adequate method and data preparation algorithm for the desired implementation has been selected. Third, the implementation of separate solutions to the problems previously stated and testing has been performed. Finally, the

grand result is studied. The implementation is realized on the Altera DE2-70 board with software Quartus II 8.1 and NIOS II using Verilog. The results are analyzed in Matlab R2016b. The camera TRDB D5M captures the data.

The master's thesis is in the context of the Engineering Technology faculty at the campus of Diepenbeek. The faculty is the result of a joint collaboration between the KU Leuven and the UHasselt. This master's thesis is part of the main subject of Electronics-ICT.

CoSenS (Computational Sensor Systems) is a research group at the UHasselt researching innovating architectures for computational multi-camera video and vision systems. The research group develops new hardware and software application for high resolution, high frame rate multi-camera distributed calculation compatible video and vision systems. These systems are implemented utilizing modern digital image sensors. The fields of application vary between omnidirectional video, 3D reconstruction, interpolation and the use of virtual cameras. There exist numerous practical applications namely real-time traffic control, assembly line control, medical applications, …

2 Design

2.1 Introduction

Digital cameras present in multi-camera systems utilize the Bayer pattern to collect color video information. This pattern consists of n red, n blue and 2n green pixels aligned in squares as illustrated in figure 1. Each pixel only registers the value of its respective color. The intensity is measured by the photo diodes in the image sensor array and stored. In other words, the measured intensity is the amount of incident photons in the specific photo diode. The color filters above photo diodes select the wavelengths of the light in the resp. photo diodes. The data registered by the Bayer pattern is named RAW data. The measured intensities are then recalculated into RGB images most commonly used in applications. Most existing and commercialized compression methods are based on the RGB pattern. The process of interpolating the RAW data into RGB data is named demosaicking [1]–[5].

The noise caused by crosstalk and camera electronics are simply ignored and further injected into the calculations of the RGB image during the demosaicking process. The noise is either removed afterwards or deemed irrelevant for the application. The process is visually illustrated in figure 2. In the context of multi-camera systems, the noise must be removed prior to the RGB calculation to maintain cohesion between images. Recalculation of the images from each camera into a virtual image requires exact values [1], [6]–[10].

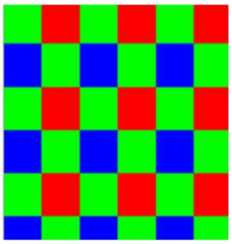


Figure 1: Bayer pattern [11]

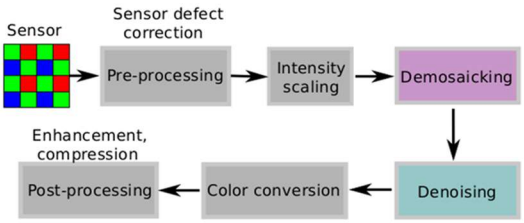


Figure 2: Traditional schematic [1]

A standard lossless compression method based on predictors is built in a several work-order steps. First, the prediction performed by the predictor. Based on surrounding pixels, the value of an upcoming pixel is predicted. Generally, the prediction will reach a value close to this of the upcoming pixel and is subtracted from the real value. A good predictor will cause a distribution of values around zero. Second, a form of entropy encoding is implemented on the transformed dataset. Entropy encoding methods work best with dataset having a high percentage of reoccurring values.

The compression methods studied in previous works [11], [12]are all based on the same principle. First a compression method is chosen. The compression method functions best on a specific type of data. Second an algorithm is selected to recalculate the image information into the specific data of the compression method. I.e. The work of W. Zhang [11]. Zhang chose to study the data reduction method most suitable for Huffman entropy encoding. This technique functions best with a high rate of recurring values. She

compares the DPCM, Paeth and Gap algorithms to prepare data. The compression has been fulfilled on both RAW and RGB images.

Huffman encoding, developed by D. Huffman [13], is a compression algorithm which optimally stores recurring values in a more efficient way. The algorithm requires the calculation of the probability that a value will be present in the data. Based on the highest probability the data is then restored in less or more bits depending on the amount of occurring values. The encoded values are mapped for later reconstruction.

The DPCM, Paeth and Gap recalculation methods are all based on the same principle. The value of the upcoming pixel is predicted based on the previous surrounding pixels. The prediction is then subtracted from the actual value and the error is later encoded by the entry coding compression algorithm. A good prediction method will cause a large recurrence of the value '0'. A visual representation of this system can be seen in figure 3. The effective result of the DPCM, Paeth and Gap methods is the shift of the prediction error into a Laplacian distribution [14].

A Laplacian distribution resembles the Gaussian distribution with a sharp peak in the center of the curve. A representation can be seen in figure 4. In this case, the peak represents the large amount of zeroes present. This gives us a unique opportunity to utilize a different lossless data compression algorithm namely Golomb-rice encoding. The Golomb-rice encoding method can reach the efficiency of the Huffman entropy encoding algorithm, which is optimal, when the dataset is specifically prepared for optimal Golomb-rice encoding. This form of encoding is more effective when the amount of low values increases. The more effective the predictor the better the compression ratio will be [11], [14]–[16].

This is where the problem with noise comes in. Noise will cause larger, faulty values which can ruin a prediction since not all pixels will have an increased value. A form of filtering must be implemented prior to the compression to counter this phenomenon.

To summarize: the following problems in order of occurrence must be solved. First the noise must be reduced. Second: the data must be prepared to make the compression algorithm the most efficient. Third: the actual compression must be performed. The result must be a compression adequately efficient, fast, lossless and implementable on hardware. A visual representation of the implementation is given in figure 5.

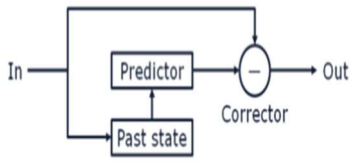


Figure 3: Schematic of predictor [11]

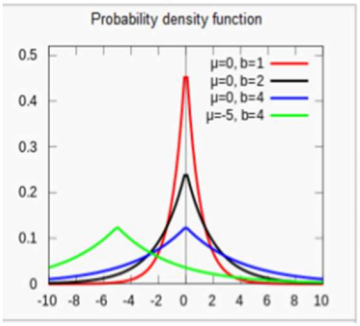


Figure 4: Laplacian curve [11]

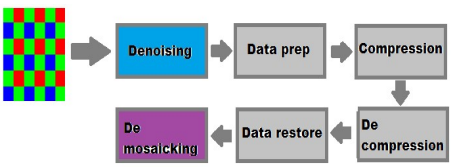


Figure 5: Implementation schematic

2.2 Noise reduction

2.2.1 Definition of noise

To design a way to get rid of noise originating from digital image sensor a definition of noise must first be set up. In [1], [17] a widely accepted, simplified formula relating the desired value and the actual value is described.

$$y = x + v$$

y describes the output value measured by the camera, x is the noiseless value and v represents the noise. The formula is applicable to each color channel.

Researchers in [1] concluded that noise can be divided in two parts namely fixed and random noise. A representation is given in the following formula

$$\eta_D = \eta_P + \eta_R$$

η_D describes total noise, η_P describes the fixed pattern noise and η_R describes the random noise. In other words, images created by digital cameras contain fixed noise present in every pixel and noise present at random places in the image. The random component of the noise causes an error in the prediction while the fixed component should be evened out by the predictor. Removing the random noise should be a priority before compression can start. The fixed noise will however cause an error between the images from different cameras since it is not a constant value over different image sensors. To simplify the creation of the virtual image, the fixed noise must also be removed. These formulas describe the model on which the noise removal will be designed.

2.2.2 Dark pixel calibration

As previously stated, digital cameras utilize a pixel array in Bayer pattern to capture image data. The sensor measures the intensity of light passing through each pixel. The intensity is the digital representation of the color grade of each respective pixel. The digital camera has a m x n pixel array which can be measured by the image sensor. However, not all m x n pixels are used to capture active image data. The pixel array is divided into three types of pixels:

1. Active image pixels that are responsible for capturing the actual image;
2. Dark image pixels that are shielded by a light blocking metal layer on the photodiodes and that do not directly receive photons from the incident light;
3. Active boundary pixels that capture any fallout to correctly capture the border of the active image.

In theory, the dark pixels will always measure an intensity of zero since no light is able to reach this part of the Bayer pattern. When the whole Bayer array is read out, the photon diodes in these dark pixels only generate electron-hole pares due to thermal noise. The higher the temperature the higher the signal resulting from thermal noise. In practice, the

dark pixels register a value slightly different from zero. Since no light can reach this part of the sensor it can be concluded that the measured value is the sum of fixed and random noise.

The intensity measured behind the dark pixels is the total noise in the image. Research in [1], [8], [18] stated that noise over color channels is universal. The presence of multiple rows and columns of dark pixels can be used to create an average total noise value. By subtracting the value of the intensity of the dark pixels from the measured value noise can, in theory, be effectively be removed completely.

2.2.3 Mathematical based solutions

2.2.3.1 Gray value noise filter

Noise filters for gray value need to be well designed since it is impossible to implement color correction after the removal of visible noise. If RAW data can be converted to a single dimension, the filter will be usable on the video data [17], [4].

Raw Data, in fact, only has one intensity value per pixel. The recalculation into an RGB image uses these values several times to achieve the same resolution. Loading the data directly into a gray value filter however will cause a loss of cohesion in the image resulting in an inadequate filtering. The data must be rearranged to maintain correlation. In common applications, the color channels are separated from the CFA image and denoised separately [8], [18]. Figure 6 gives a visual representation of the process. On these restructured images, various gray value noise filters can be applied, allowing the selection of the best filter. The previously described implementation can result in little loss in correlation. However, applying this method will result in a small loss of pixel value [18]. Researchers in [17] report a computationally low-cost result of 33% compared to classic RGB filtering methods while maintaining the image quality of traditional RGB noise filters. An interesting result of their research is the similarity of random noise in each color channel.

The benefits of applying gray value noise filters are an easy implementation and a gain in computational speed. Negative aspects are a loss of accuracy. The nature of the data output by the camera causes this method to be unsuitable. The camera prints the image line by line, most gray value noise filters require surrounding pixels from all sides to perform the filtering. Adjusting the filters to this input method is not an option either since this will result in an even greater loss of cohesion in the image.

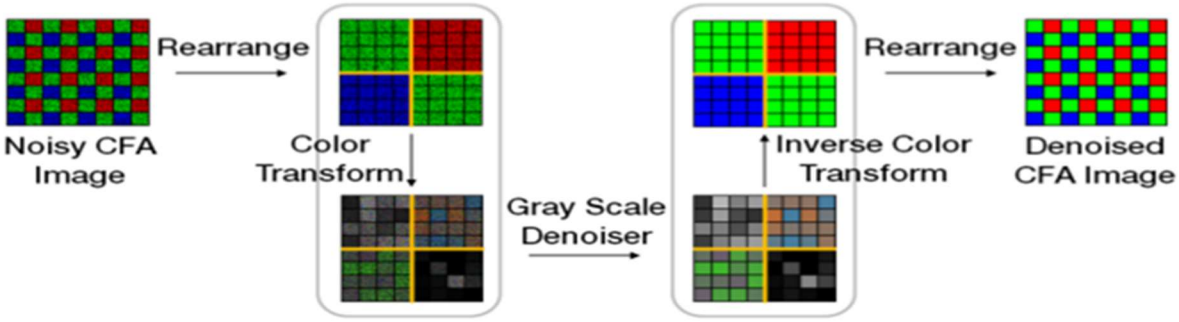


Figure 6: Gray filter on RGB images schematic [17]

2.2.3.2 Noise reduction based on block compare

In image processing a common way to reduce random noise is the median filter. The filter examines a block of pixels surrounding the pixel. Methods have been developed based on this principle. Various works have studied this type of method posting effective results [8], [18]. The noise reduction methods based on block compare can be implemented on the FPGA providing the dimensions of the matrix remains limited. If not for inspiration for a denoising method, these works can still be useful to conclude denoising results.

Principle component analysis (PCA) is a statistical procedure utilizing an orthogonal transformation. PCA converts a dataset which contains correlation into a set of linear uncorrelated variables named principal components. The method calculates the eigenvectors of a covariance matrix describing the axes which generally represent the variance in the data set [18]. Basically, PCA decorrelates a dataset making it representable in a reduced number of dimensions. The nature of random noise will cause an even distribution of intensity along the axes generated by PCA. The required data will however be concentrated alongside the most significant subsets. By removing the smaller subsets and a reverse transformation, a noise reduction can be achieved [8].

In [18] an algorithm is developed to use PCA to calculate the covariance matrix of training blocks and compares them with the matrices of surrounding blocks to remove noise. The result is a reduction in the color differences created by noise, thereby increasing the Peak signal noise ratio (PSNR).

[8] describes a method to create blocks in a 3-D structure. The researchers first group similar blocks with the same color configuration into one 3-D array. Afterwards a 3-D transform and shrinkage is applied. After the inverse transformation blocks are returned to their respective places. The last step is the calculation of output images by weighed averaging of the overlapping estimates.

2.2.3.3 Linear minimum mean square-error estimation (LMMSE)

Minimum mean square-error estimation is a lossy estimator used in signal processing designed to minimize the mean square error (MSE). MSE is a numeric representation of image quality. The use of LMSSE on imagery is to estimate the primary difference signals (PDS) to take advantage of spectral correlation in the image. Spectral correlation

between color channels is used in the demosaicking of the image and can be used for denoising sensor noise when the parameters are known prior. PDS is calculated with the following formula:

$$Xg, r = G1 - R$$

$$Xg, b = G2 - B$$

Xg,r and Xg,b represent the PDS, RG1G2B represent the color channels of the Bayer filter.

In [10] it is assumed that noise is channel dependent, additive and Gaussian distributed with zero mean. The formulas calculate the difference between the green channel in a row and the red or blue channel. The reconstruction of an image can be done providing the green channel can be reconstructed perfectly. For this purpose, LMSSE is used. The denoising is done using a wavelet algorithm and solely performed on the green channel. The linear model of the PDS is worked out in [10] and given by the formule:

$$\overline{X}_i^d = X_i + \varepsilon_i^d + v_i^d$$

$\overline{X}_i^d$represents the interpolated PDS of a color channel, X_i represents the PDS of a color channel, ε_i^d represents the directional interpolation error, v_i^d represents the additive Gaussian white noise. Refer to [10] for the explanation and deduction of the formula.

Based on this linear model, LMSSE is performed. An estimate of the green channel is calculated. The formula used for the calculation is:

$$\hat{x} = \mu_x + \frac{\sigma_x^2}{\sigma_y^2}(y - \mu_x)$$

Where μ_x is the autocorrelation function prescribed in [10], σ_x^2 = Var(x) and σ_y^2= Var(y). The formula calculates the directional estimates of the PDS whom are afterwards fused using a wavelet transform.

The implementation of LMSSE and use of wavelet transform is an interesting subject and warrants further research. LMSSE can be programmed on the FPGA and wavelet transforms in 2 dimensions have been performed by researchers in [19], [20]. Several works have been performed on the optimization of denoising using wavelets and LMSSE [22], [23].

2.3 Lossless compression method

2.3.1 Introduction

In signal processing, compression is the encoding of data into fewer bits than the original information. There exist two types of compression namely lossy and lossless compression. Lossy compression is used in applications where the exact reconstruction of data is unnecessary. The reduction of bits is achieved by removing irrelevant or less essential information which is not immediately observable by human viewers. Lossless compression reduces the number of bits by locating and removing statistical

redundancies. No information is lost in lossless compression. Where lossy compression methods are applicable to many sort of data series, lossless methods are designed for a specific form of information [1], [3], [11], [17], [7], [25], [9], [26].

There exist three main commercial uses for lossless data compression:

1. General data compression e.g. winRAR, 7zip,
2. Audio compression e.g. ALAC, WMA Lossless,
3. Graphic compression e.g. JPEG-LS, PNG.

This work solely studies the graphic compression methods. The following part describes Run-length encoding (namely Golomb and Golomb-Rice encoding) and Huffman encoding [13], [27], [28].

2.3.2 Run-length encoding

2.3.2.1 Introduction

Run-length encoding (RLE) is a lossless data compression method where data is stored in a value and a count. These methods have the highest efficiency when the same data value reoccurs often in the dataset. Golomb encoding and Golomb-Rice encoding are based on the same principle but vary in the criteria of the parameters used. A simple overview of the working method is described and the unique qualities of each method is given below [27], [28].

1. Assign an integer value to parameter m,
2. The value to be encoded, iDATA, is divided by m,
   a. quotient q = int(iDATA/m),
3. remainder r = int(iDATA%m),
4. Encoding (when unary encoding writes 0's),
   a. The integer value of q in written in unary coding: q bits of zero are written,
   b. A splitting bit 1 is written out,
   c. the remainder is written out in regular binary and stored in k bits ($\mathrm{k} = \log_2(\mathrm{m})$).

2.3.2.2 Golomb encoding

Golomb encoding utilizes the encoding algorithm described above. The criteria for the value of m is that it must be a real integer. The number of bits used to store the remainder is dependent on the value of the remainder [27].

$$\text{When } \mathrm{r} < 2^{\mathrm{k}} - \mathrm{m} : \text{the remainder is stored in } \mathrm{k} - 1 \text{ bits}$$

$$\text{When } \mathrm{r} \geq 2^{\mathrm{k}} - \mathrm{m} : \text{the remainder} + 2^{\mathrm{k}} - \mathrm{m} \text{ is stored in k bits}$$

The variation in the remainder storing method makes reconstruction more difficult. There exists a special type of Golomb encoding named Rice or Golomb-Rice encoding.

2.3.2.3 Golomb-Rice encoding

Golomb-Rice encoding has the same working principle as Golomb encoding with one difference. A criterion for the value m must be met. the value of the integer m must be a multiple of 2. When this criterion is met the variation in the storing method of the remainder is removed [28].

$$\text{When } r < 2^k - m : \text{the remainder is stored in } k - 1 \text{ bits}$$

Note that when m = 1, the whole data sequence is stored in unary coding. The choice of m is crucial for a decent compression ratio and must be adjusted to the histogram of the information to be encoded. Golomb-Rice encoding is more suitable for our application since it is easy to implement and decode without difficult calculations.

2.3.3 Huffman encoding

Huffman encoding is a compression method designed for the optimal and lossless storage of a series of symbols [13]. Applications for the algorithm are generally in the fields of data communication and the compression of digital imagery. The principle of Huffman encoding is simple;

1. A list of the probability of occurrence of symbols in the dataset is calculated and sorted from high probability of occurrence to low;
2. A Huffman tree is build;
   a. The two symbols with the lowest occurrence frequency are linked together, the frequency of the link is the sum of the symbols' probability;
   b. The links is reentered into the sorted list;
   c. Step one and two are repeated until one symbol remains;
3. Starting from this last symbol, the tree is encoded giving the highest frequency a 0 and the lowest a 1.

To decode the Huffman bitstream the tree must be sent to the processing system in advance. This method will be more effective when there are few differing symbols or a large rate of reoccurrence [13]. A visual representation of the creation of a Huffman tree is given in figure 7.

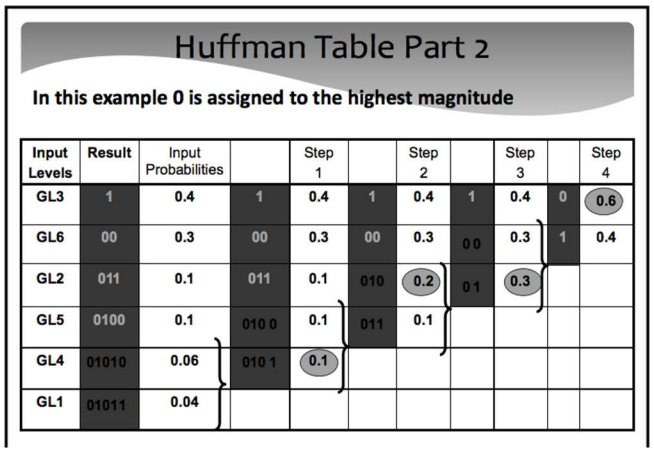


Figure 7: Huffman table [38]

2.4 Data preparation for Golomb-Rice encoding

2.4.1 Introduction

As previously stated, lossless compression methods are designed for a specific type of dataset. Intensity values from a digital camera can vary for example between 0 and 255 with each being equally likely to occur. The data needs to be rewritten to get a smaller number of symbols with a larger chance of occurrence. In the following part, several predictive-corrective coding filters will be described. The filters use the surrounding pixels to calculate a prediction for the upcoming pixel. This prediction is then subtracted from the actual measured value of the pixel and the error is what remains. A good predictor will result in a high compression ratio when using Golomb-Rice encoding since most of encoded values will be zero or close to zero

2.4.2 Differential pulse-code modulation (DPCM)

The DPCM algorithm functions as a predictor for the upcoming pixel value [14]. For raw images, the predictor uses the values of the same color channel to make a prediction. If the upcoming pixel in the Bayer pattern has the coordinates (x,y), the prediction is calculated:

$$prediction = B +$$

With $A = int(rawimage(x \quad 2, y-2)$ = northwestern pixel

$$B = int(rawimage(x, y-2) = northern\ pixel$$

$$C = int(rawimag \ \ (x-2, y) = western\ pixel$$

The prediction is then subtracted from the actual value and the error is used for further calculations. The error will have a value close to zero when there exists gradual change in color in the image. When the color change is sudden, the predictor will struggle. Sudden change is often the case in natural imagery. A visual representation of the used pixels is given in figure 9.

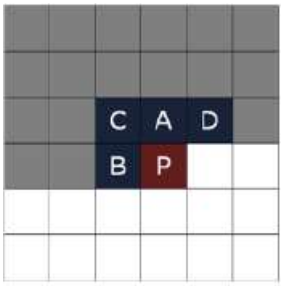


Figure 8: DPCM and Paeth method pixel pattern [11]

2.4.3 Paeth method

The Paeth method is based on the DPCM method but adds another layer of prediction correction. The initial prediction is calculated according to the formula:

$$iprediction = B + C - A$$

As was the case with DPCM. Paeth uses the prediction to try and estimate which of the surrounding pixels is the best estimate for the prediction.

$$Pa = abs(iprediction - A)$$

$$Pb = abs(ipredictio \quad - B)$$

$$Pc = abs(iprediction - C)$$

With $A = int(rawimage(x-2, y-2)$ = northwestern pixel

$$B = int(rawimage(x, y-2) = northern\ pixel$$

$$C = int(rawimage(x-2, y) = western\ pixe$$

The actual prediction used for the subtraction is calculated based on Pa, Pb and Pc.

$$aprediction =$$

$$A\ if\ Pa \leq Pb\ and\ Pa \leq Pc$$

$$B\ if\ Pb \leq Pa\ and\ Pb \leq Pc$$

$$C\ if\ Pc \leq Pb\ and\ Pc \leq Pa$$

In other words, the lowest probability will be chosen and the respective value will be used as the prediction [11], [29]. A visual representation can be seen on figure 9.

2.4.4 Gradient Adjusted Prediction (GAP)

The GAP algorithm is utilized in context-based adaptive lossless image coding (CALIC) [11], [14]. The Gap method uses the surrounding pixels to calculate the change in pixel value and then adjusts the prediction accordingly. Gap uses 7 surrounding pixels if the to be predicted pixel is (x,y) in the Bayer pattern then the used pixels are:

$$N = north = int(rawimage(x, y-2))$$

$$W = west = int(rawimage(x-2, y))$$

$$NW = northwest = int(rawimage(x-2, y-2))$$

$$NN = northnorth = int(rawimage(x, y-4))$$

$$WW = westwest = int(rawimage(x-4, y))$$

$$NE = northeast = int(rawimage(x+2, y-2))$$

$$NNE = northnortheast = int(rawimage(x+2, y-4))$$

The average prediction and standard deviation generally improve when using more pixels. Next in the algorithm is the calculation of the color shift in each color plane. First the change in edges is calculated.

$$dh = horizonta\ \ shift = int(abs(W - WW) + abs(N - NW) + ab\ \ (N - NE))$$

$$dv = vertical\ shift = int(abs(W - NW) + abs(N - NN) + abs(NE - NNE))$$

$$error = dv - dh$$

The value of the prediction is based on the error value. Based on the numerical value the best suitable prediction calculation is selected:

$$If\ erro \quad 80 \rightarrow pred = W \qquad If\ error \quad 80 \rightarrow pred = N$$

$$If\ 80 > error > 32 \rightarrow pred = \frac{\frac{N+W}{2} + \frac{NE -}{4} + W}{2}$$

$$If\ 32 > error > 8 \rightarrow pred = \frac{3 * (\frac{N+W}{2} + \frac{NE - N}{4}) + W}{4}$$

$$If\ 8 > error > -8 \rightarrow pred \quad \frac{N+W}{2} + \frac{NE - NW}{4}$$

$$If\ -8 > error > -32 \rightarrow pred \quad \frac{3 * (\frac{N+W}{2} + \frac{NE - N}{4}) + N}{4}$$

$$If \quad 32 > error > -80 \rightarrow pred \quad \frac{\frac{N+W}{2} + \frac{NE - NW}{4} + N}{2}$$

The Gap method considers more values and makes a detailed prediction for the upcoming pixel value. This method will make decent predictions for subtraction. The remainder will be a data set with values close to zero or zero. The GAP method will be used to prepare the data for rice encoding [11], [30], [31]. A visual representation of the pixels used is given in figure 8.

Figure 9: Gap method pixel pattern [11]

3 Material and method

3.1 Material and specifications

3.1.1 TRDB-D5M Camera

“The Micron Imaging MT39P031 is a 1/2.5-inch CMOS active-pixel digital image sensor with an active imaging pixel array of 2592 Horizontal and 1944 vertical pixels. It incorporates sophisticated camera functions on-chip such as windowing, column and row skip mod, and snapshot mode. It is programmable through a simple two-wire serial interface.” –D5M datasheet, general description p1 [32].

The sensor is used to capture high resolution images. The pixel sizes are 2.2μm x 2.2μm and images can be captured at a speed of up to 60 fps. The image sensor also has several useful features utilized in this work. The snapshot function allows the capture of single frame. The bulb exposure mode allows the increase of shutter time and creates brighter images when capturing.

The general description gives us the active image area. For this application, all pixels are used. The full Bayer filter is built in a RG1G2B pattern. The pixels for rows and columns are given in tables 1,2 and visually represented in figure 10.

The sensor has built in black pixel calibration. A feedback control system adjusts the value captured by the black pixels to fall within a specific threshold. In theory, the value for these pixels should always be close to 0. In practice, tests were done and it was concluded that there are intensities being captured by the black sensor of up to 0.5% of the maximum value [32]–[34] which is a decent result.

Table 1: Dark pixel columns [32]

Column	Pixel Type
0–9	Dark (10)
10–15	Active boundary (6)
16–2,607	Active image (2592)
2,608–2,617	Active boundary (10)
2,618–2,751	Dark (134)

Table 2: Dark pixel rows [32]

Row	Pixel Type
0– 49	Dark (50)
50–53	Active boundary (4)
54–1997	Active image (1944)
1,998–2,001	Active boundary (3)
2,002–2003	Dark (2)

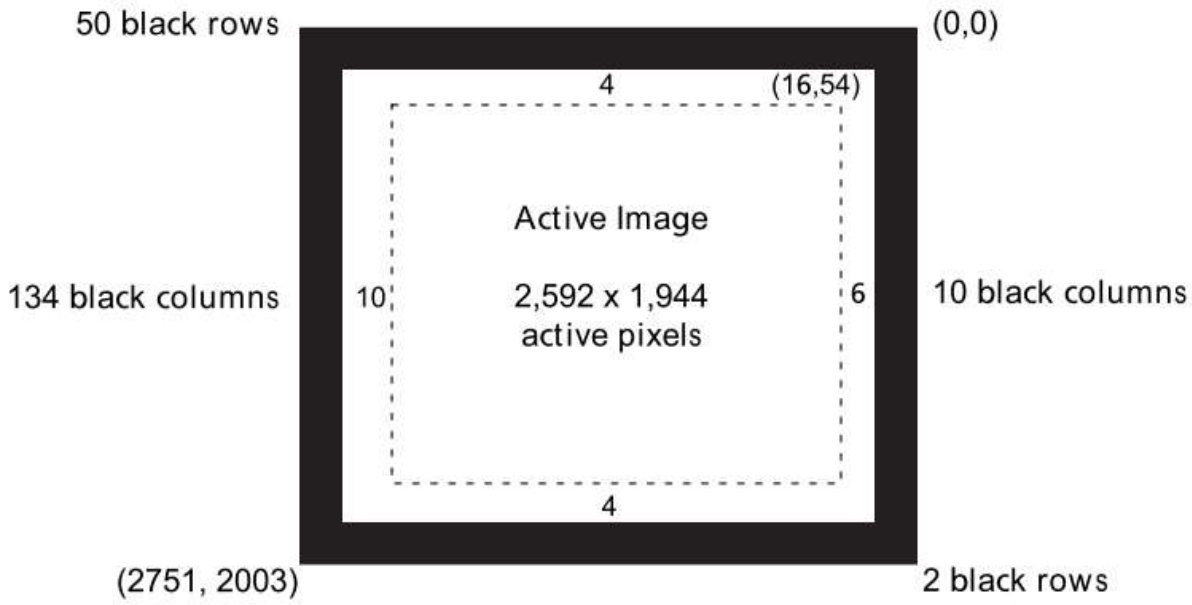


Figure 10: Visual representation of dark pixels and active image D5M [32]

3.1.2 The Altera DE2-70

DE2-70 is part of the cyclone II family. The DE2-70 board has many features that allow the user to implement a wide range of designed circuits. The board has 32-Mbyte SDRAM which is used to capture and save single frames. The GPIO's are used to connect the image sensor.

Programs for this board are written in Quartus II 8.1 as Verilog files. The decision to use this board and software language was simply past experiences and availability. The physical setup can be seen in figure 11.

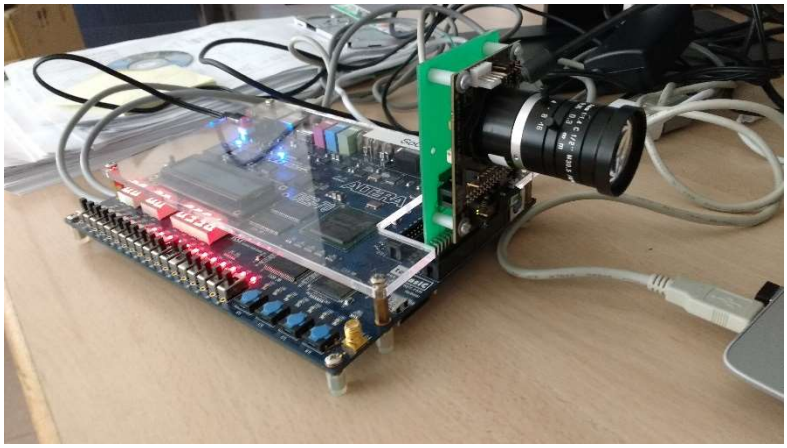


Figure 11: Hardware setup

3.1.3 Matlab R2016b

The images captured using the camera and registered by the FPGA were processed using Matlab.

3.2 Implementation

3.2.1 Source code

Altera has a tutorial for the use of the DE2-70 camera. This tutorial captures data, sends to VGA and allows you to capture images. The code of this tutorial was used as a base and adjusted or rewritten for this application.

The method to capture images was as following:

1. At 60 MHz, the registry of the camera is loaded into the FPGA;
2. The information of the registry is loaded into CCD_Capture to link data of the same frame together;
3. The data of the whole frame is loaded into RAW2RGB which transforms it from RAW to RGB;
4. The value of each color channel is send to the VGA output and the NIOS II processor for capturing.

3.2.2 Black pixel calibration

The first adjustment made was the removal of the RAW2RGB code so that RAW image value could be read out. This was accomplished by writing the data into a single bit stream. The initial data stream is loaded into a converter and the resulting RAW values are send to the processor for capture.

The data images proclaimed by the camera is set to a resolution of 1280x1024 pixels at 60 MHz. The pixels used for capture in the original program were part of the active image area. The first implementation was the adjustment of the I2C configuration of the sensor to read out an image of 1280x1024 pixels but containing the upper and right black pixel rows and columns.

The I2C configuration of the sensor was originally set to start at the active image pixels. An alteration of the sensor_start_row and sensor_start_column variables allowed for the capture of the black pixel values. The images are captured using a NIOS II processor which was made by Alterra for this specific sensor. To get satisfactory results, when capturing images, all information input into the processor must be structurally equal to the original.

When the adjustments are made the ability to capture the RAW images of the camera with a border of black pixels on the upper and right border now exists. The images captured will be used to test the code in Matlab. The NIOS II processor does not allow the implementation of the compression method and the processor due to lack of RAM.

```
assign sensor_start_row         = 24'h010000;// + (2004 - (PIXEL_SKIP + 1)*NR_OF_ROWS) / 2;
assign sensor_start_column      = 24'h020000;// + (2752 - (PIXEL_SKIP + 1)*NR_OF_COLS) / 2;
```

Figure 12: Loading in black pixels

3.2.3 GAP method

The Gap method utilizes the CFA capture data as input data, performs the GAP algorithm and sends it through to a histogram module. The histogram should show a peak near zero in the middle of the screen to confirm success. The prediction is calculated with incoming data, stored for one clock cycle and released as prediction. When the values are negative, an adjustment must be made to prevent data loss. See figure 13.

The histogram, as shown below, features the different values resulting from the GAP module. A Laplacian distribution with a peak around zero is the result of the histogram. The output data of the GAP method together with the control signals are read out into the Golomb-Rice encoding module. The clock controlling is 60 MHz to keep up with the incoming data. See figure 14.

```
assign dh1_sign = W - WW;
assign dh2_sign = N - NW;
assign dh3_sign = NE - N;
assign dv1_sign = W - NW;
assign dv2_sign = N - NN;
assign dv3_sign = NE - NNE;
assign dh_sign = W - WW + N-NE + N-NW;
assign dv_sign = W - NW + N - NN + NE - NNE;
assign error_sign = dv_sign - dh_sign;
```

Figure 13: GAP method preparing variables

```
always @(posedge iCLK) begin
   error <= error_sign[13] ? -error_sign : error_sign;
   dh <= dh_sign[13] ? -dh_sign : dh_sign;
   dv <= dv_sign[13] ? -dv_sign : dv_sign;
   if(dv > dh) begin
        if(error > 80) begin
            predict <= W;
        end else if(error > 32) begin
            predict <= ((N+W)/2 + (NE-NW)/4 + W)/2;
        end else if (error > 8) begin
            predict <= (3*((N+W)/2 + (NE-NW)/4) + W)/4;
        end else begin
            predict <= (N+W)/2 + (NE-NW)/4;
        end
   end else begin
        if(error > 80) begin
            predict <= N;
        end else if(error > 32) begin
            predict <= ((N+W)/2 + (NE-NW)/4 + N)/2;
        end else if (error > 8) begin
            predict <= (3*((N+W)/2 + (NE-NW)/4) + N)/4;
        end else begin
            predict <= (N+W)/2 + (NE-NW)/4;
        end
    end
end
```

Figure 14: GAP method implementation

3.2.4 Golomb-Rice encoding

Now the information is prepared for Golomb-Rice encoding. The data output of the GAP module is loaded into the data input of the encoding module. First, the data is loaded into a data2unary function which calculates the quotient and the remainder. The quotient is simultaneously transformed into unary coding and send back to the encoding module. The unary output is a register containing the quotient number of bits 1. The Remainder output contains the remainder of the division. See figure 15.

The unary encoding makes up the first part of the encoded word. Presenting the amount of times the divider m can fit into the incoming data. Second, the encoded register is shifted left by k bits to allow us to store the remainder and input the division bit. The length of the word is the value of the number of bits which will be transmitted to the processing system since this value is still stored in a set register. See figure 16.

```
data2unary(.iCLK(iCLK), .iDATA(iDATA), .oUnary(Unary), .oRem(Remainder));
```

Figure 15: Writing input data as unary

```
assign encoded = oData1;

always @(posedge iCLK) begin
    oData1 = Unary;
    oData1 = oData1 << k;
    oData1[k] = 0;
    oData1[k-1:0] = rem[k-1:0];

    oData_len = q+k;
end
```

Figure 16: GAP algorithm implementation

4 Results

The results were calculated using Matlab, pictures below show a grey value image captured by the D5M camera. The results of the implemented method can be seen in figure 17-21. Figures 17-19 represent respectively the original image, the prediction made by the GAP method and the error between them. Images are processed by the FPGA according to the clock controlling the camera. The FPS of the camera is the limiting factor. Results in the form of images of the compression performed on the FPGA can be found below as well as in attachments. The first image shows the histogram of the GAP method with the captured value on the background. The second image shows the value of the image through color:

- Red = low values, 0-750 (0-1024 but is dominated by green in higher values),
- Green = medium values, 751-1250,
- Blue = high values, 1251-2047 (1025-2047 but is dominated by green in lower values).

The image seems to be largely dominated by red and blue whilst the original image was clearly greener. The reason behind this is that low values become lower since they are stored in less bits and the remainder is defaulted to zero. The high values become higher since the start of each encoded word is many ones mimicking high intensity. This can effectively almost eliminate the medium values from the dataset since they will either be translated to a higher or lower value. The results from the compression on the FPGA can be seen in figure 20-21. Be mindful that this is merely a representation of the function since some large encoded words cannot be properly represented through VGA cable.

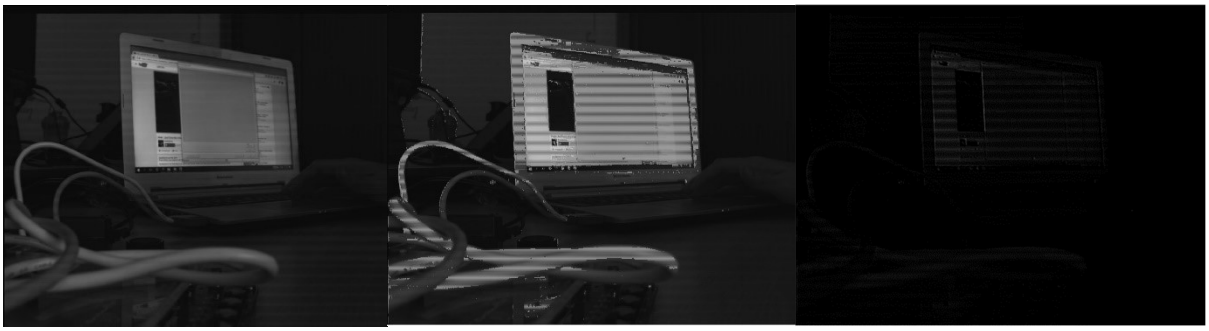


Figure 17: Input RAW image

Figure 18: Visual representation of prediction

Figure 19: Error = abs(input image - prediction)

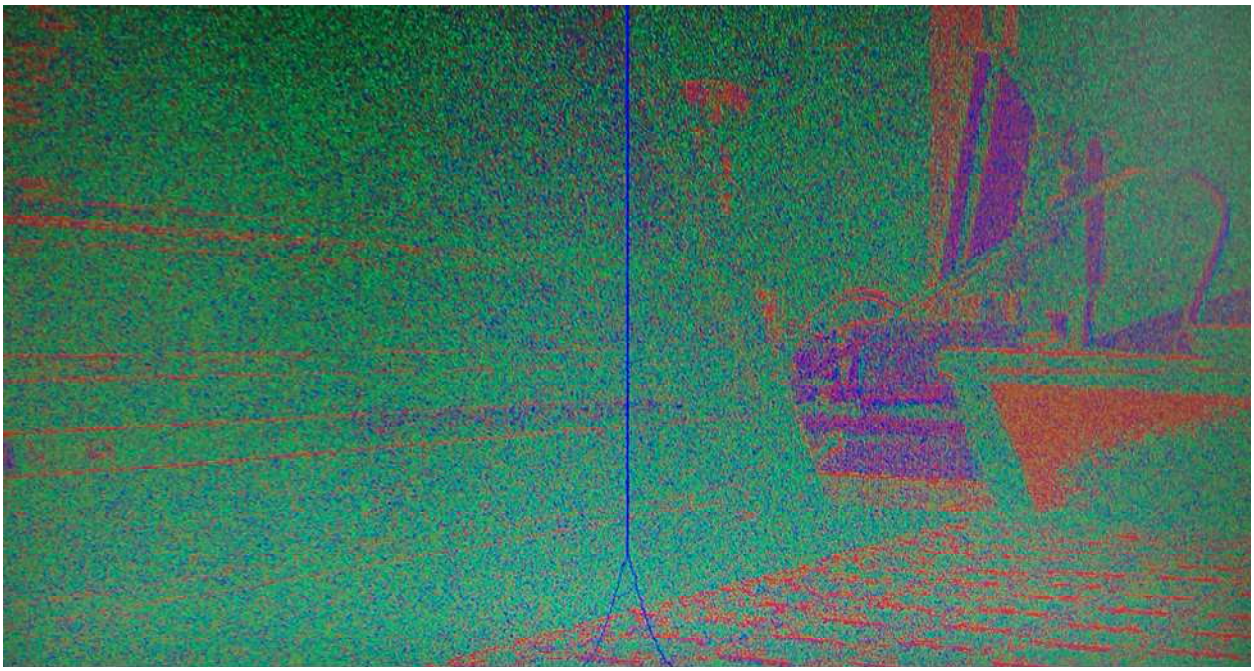


Figure 20:: Image prior to compression, color represents value of pixel. Blue line represents the histogram of data recalculated by the GAP algorithm.

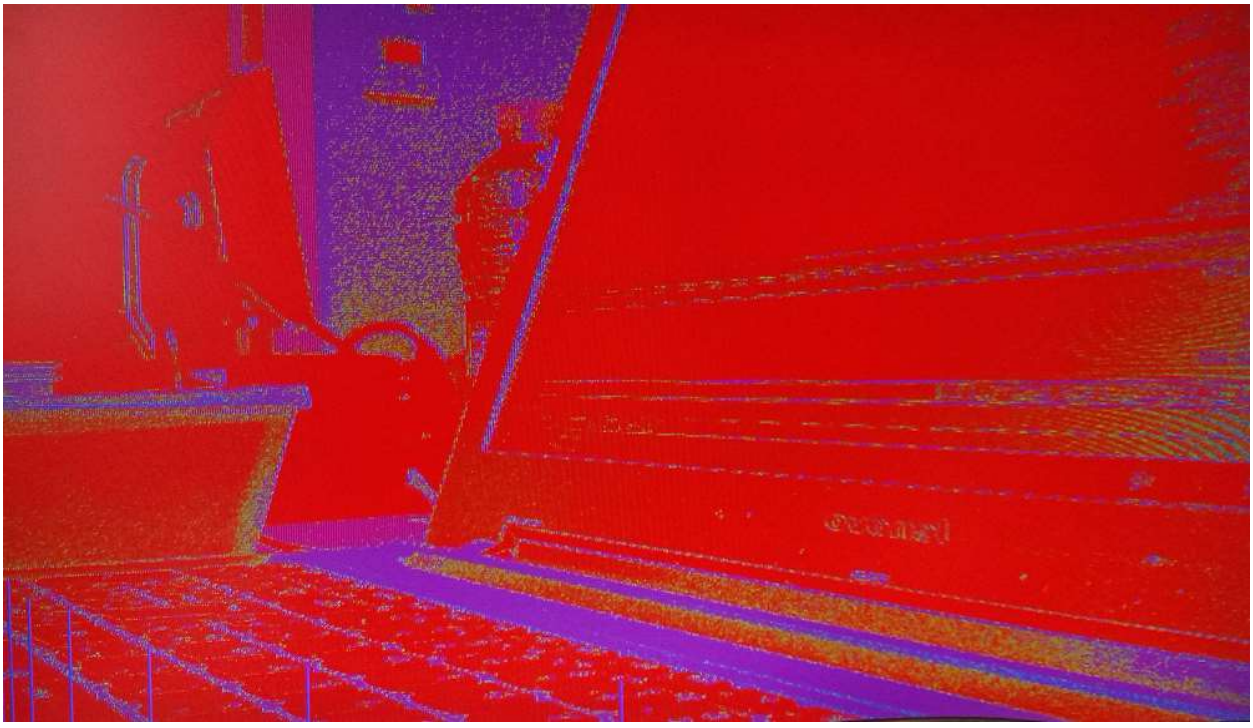


Figure 21: Image after compression, color represents value of pixel

4.1 Compression ratio

The achieved compression ratio is calculated with the following formula:

$$Compression\ ratio = \frac{Original\ file\ size}{Compressed\ file\ size}$$

The calculation is based on the number of bits needed to transfer the files as would be the case during compression. The encoding method requires the divider to be a multiple of 2. The divider, m, alternated between 2,4,8,16 and 32. The input image data is divided by m. The value of the intensities of incident light were originally stored in 8 bits. The rest storage bits, k, represent the number of bits used to store the remainder of the division.

First, the overall results will be discussed. The results show that the compression method is most effective when m = 2 and the remainder k = 1. A slightly worse result is achieved when m = 4 and k = 2. The compression ratio decreases steadily when m increases. The splitting value, which is 0 or 1, as previously stated is the bit which divides the unary coding of the Golomb-Rice encoding method from the remainder of the division. As expected the use of different splitting values does not affect the compression ratio. The bitstream resulting from the compression shows that there are less shifts between 1 and 0 required when the division bit equals 0. Utilizing 1 as splitting value would be preferable since this makes the values easier to decode. i.e. When zero is encoded with m = 4 and k = 2 would result in a rice-encoded value of 100 using 1 as divider bit and 000 using 0 as divider bit.

Second, the results between m = 2, k = 1 and m = 4 and k = 2 will be discussed. A decrease in compression ratio is caused by doubling the storage bits of the remainder. The decrease is quite significant and shows the fact that the results from the predictor are often zero or very close to zero. The results conclude an abundance of storage bits when m = 4 on the images taken by the camera.

Third, the results between m = 2, k = 1 and m = 8, k = 3 will be discussed. The number of values close to zero or zero is very predominant in the dataset. The bits needed to store zero in Golomb-Rice encoding for the first case is 2 while the second case is 4. The bits saved for higher values in the prediction are nullified by the bit added each time the prediction is correct since this will be more predominant in the dataset.

The value of the divider m should be kept low since a good working predictor will result in an overcompensation in storage bits. Most images have unpredictable sudden changes in surrounding pixels and can cause a bad prediction. The sacrifice of 1 bit per pixel will be justified in such cases.

Fourth, compression for all three methods is examined for several CFA stock images. The results show that the average bits per pixel (BBP) go down when the divider goes down. The BBP is calculated with the following formula:

$$\mathrm{BBP} = \text{Bits per pixel} = \frac{\text{Original bits per pixel}}{\text{Compression ratio}}$$

This fact reveals that the predictor works accordingly but struggles with some types of images. The best results seem to be achieved for darker images. The stock images can be found at the end of this work in the attachments section. The testchart and webcam stock images are generally dark while the lighthouse and testimage stock images show a much lighter contrast. The lighthouse images even result in a negative compression ratio meaning it will take more bits to compress this image then it would take sending the picture through in original form. The tests in the next section try to explain the decrease of compression ratio as intensity increases.

The following tests will be performed with the best method for compression namely

- m = 2
- k = 1

The results show the influence of different external factors on this form of data conversion.

Table 3: Different dividers and their influence on compression ratio

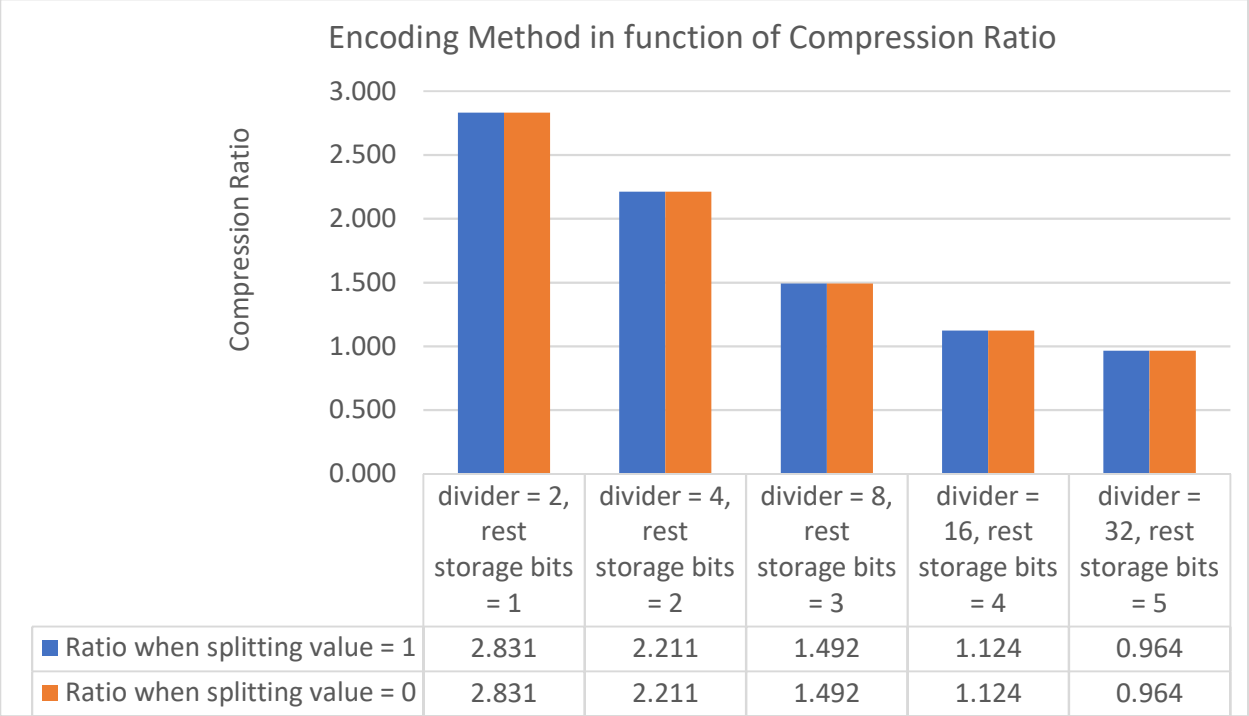
Golomb-Rice encoding	Bytes send, splitting value = 1	Compression ratio
divider = 2, rest storage bits = 1	3051749	2.831
divider = 4, rest storage bits = 2	3908333	2.211
divider = 8, rest storage bits = 3	6481187	1.333
divider = 16, rest storage bits = 4	7688108	1.124
divider = 32, rest storage bits = 5	8960885	0.964
Original	8640000	

Table 4: Different dividers and their influence on compression ratio with splitting value = 0

Golomb-Rice encoding	Bytes send, Splitting value = 0	Compression ratio
divider = 2, rest storage bits = 1	3051749	2.831
divider = 4, rest storage bits = 2	3908333	2.211
divider = 8, rest storage bits = 3	6481187	1.333
divider = 16, rest storage bits = 4	7688108	1.124
divider = 32, rest storage bits = 5	8960885	0.964
Original	8640000	

Table 5: Average compression ratio and Bits per pixel (BPP) for varying CFA images for encoding with different dividers and rest storage bits

Image/CR	m = 2, k = 1	m = 4, k = 2	m = 8, k = 3	m = 16, k = 4	m = 32, k = 5
Camera capture	**2.8312**	2.2107	1.3331	1.1238	0.9642
Lighthouse1	0.8558	1.1621	**1.3154**	1.3126	1.2203
Lighthouse2	0.8530	1.1591	**1.3139**	1.3116	1.2202
Lighthouse3	0.8718	1.1768	**1.3250**	1.3172	1.2227
Lighthouse4	0.8733	1.1778	**1.3257**	1.3172	1.2229
Testchart1	**2.1356**	1.9757	1.7456	1.4950	1.2856
Testchart2	**2.1336**	1.9761	1.7425	1.4950	1.2856
Testchart3	**2.1358**	1.9762	1.7453	1.4951	1.2856
Testchart4	**2.1334**	1.9759	1.7436	1.4950	1.2856
Webcam1	**2.3236**	2.0468	1.7520	1.4934	1.2853
Webcam2	**2.3251**	2.0474	1.7521	1.4935	1.2849
Webcam3	**2.3250**	2.0473	1.7521	1.4936	1.2853
Webcam4	**2.3252**	2.0474	1.7521	1.4934	1.2853
Testimage1	1.0739	1.3548	**1.4387**	1.3663	1.2393
Testimage2	1.0742	1.3546	**1.4386**	1.3664	1.2393
Testimage3	1.0730	1.3542	**1.4382**	1.3660	1.2393
Testimage4	1.0739	1.3547	**1.4387**	1.3664	1.2394
Average	**1.6716**	1.6705	1.5501	1.4001	1.4022
Average bpp	**4.7858**	4.7891	5.1608	5.7139	5.7054



	divider = 2, rest storage bits = 1	divider = 4, rest storage bits = 2	divider = 8, rest storage bits = 3	divider = 16, rest storage bits = 4	divider = 32, rest storage bits = 5
Ratio when splitting value = 1	2.831	2.211	1.492	1.124	0.964
Ratio when splitting value = 0	2.831	2.211	1.492	1.124	0.964

Figure 22: Encoding method in function of compression ratio

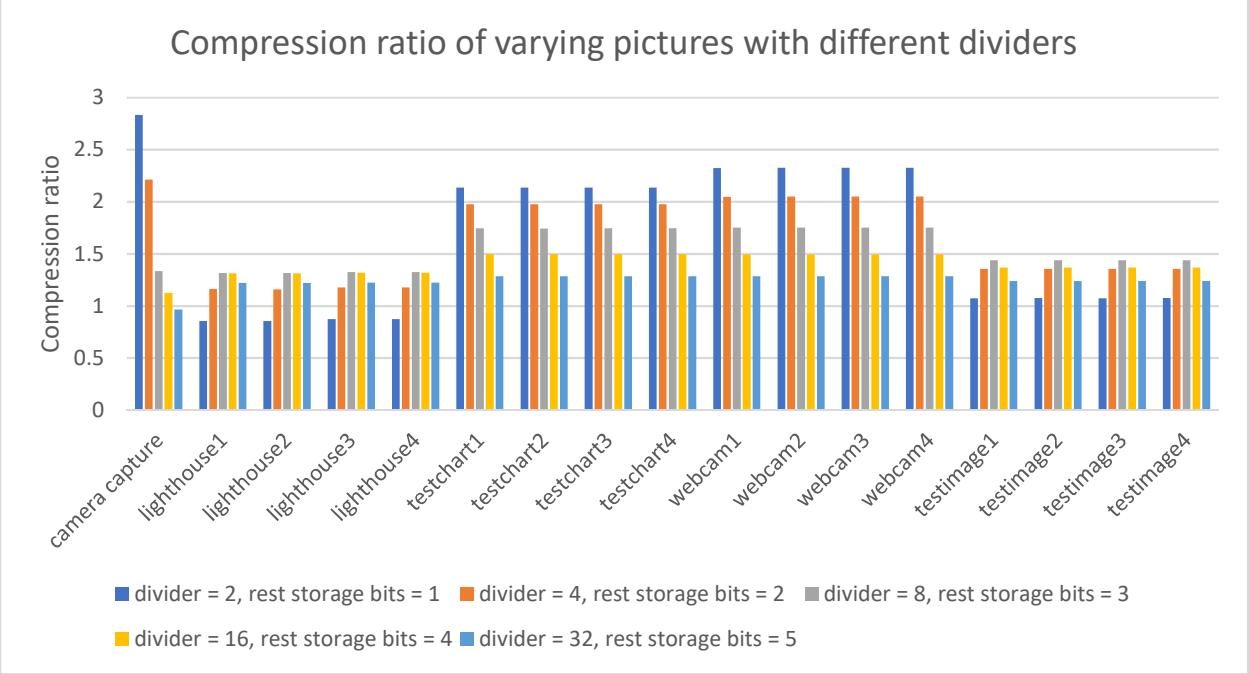


Figure 23: Compression ratio of varying pictures with different dividers

4.2 Influence of exposure

As stated previously, the further tests will be performed with a divider, m, = 2 and the number of storage bits, k, = 1. The tests were performed on several images taken at different exposure rates. The prediction was a decrease in compression with higher

exposure since previous results show darker images to have a better compression ratio. Results concluded the exact opposite. The compression ratio increases when the exposure increases.

First, the result of the standard image is explained. The standard was taken in an artificially lit room with various varying color objects present. The D5M camera has a function to increase exposure time allowing more photons to be captured by the camera per frame. The standard image shows a compression ratio of 1.979. The pictures relative to this test were taken minutes apart meaning their relation to one another are uncompromised.

Second, the darker the image becomes, the lower the compression ratio. The test pictures show more shadows and dark spots at low exposure times. The presence of change in intensity will cause the amount of errors in the prediction to increase. Shadow can cause the sudden shift in intensity where the shadow starts and ends.

Third, the bright image seems to have a better compression. The test pictures show an increase in intensity around the border. More pixels reach the maximum value of 255 when the exposure is increased and these pixels border each other. The prediction will be more accurate when block of pixels with the same value exist. The longer exposure allows more incident light to be captured and will increase compression ratio. The lack of these pixels explains why the compression ratio drops at low exposure since pixels will rarely reach the same exact value causing error increase.

The increased intensity of pixels does not affect the compression ratio negatively. An explanation for the bad compression ratios of the lighthouse and testimage stock images is now clear. The compression ratio is negatively affected by the range of pixel values. Light images that only contain high pixel values will have the same good compression ratio as dark images that contain only low pixel values. The existence of extremes, especially local extremes, will cause great errors in the prediction and lower the efficiency of compression dramatically.

Table 6: Influence of exposure on compression ratio for divider = 2 and rest storage bits = 1

Lighting		bytes send	Compression Ratio
	Original	11796480	
underexposure	Compressed	5631791	2.095
standard	Compressed	5548968	2.126
overexposure	Compressed	5365910	2.198

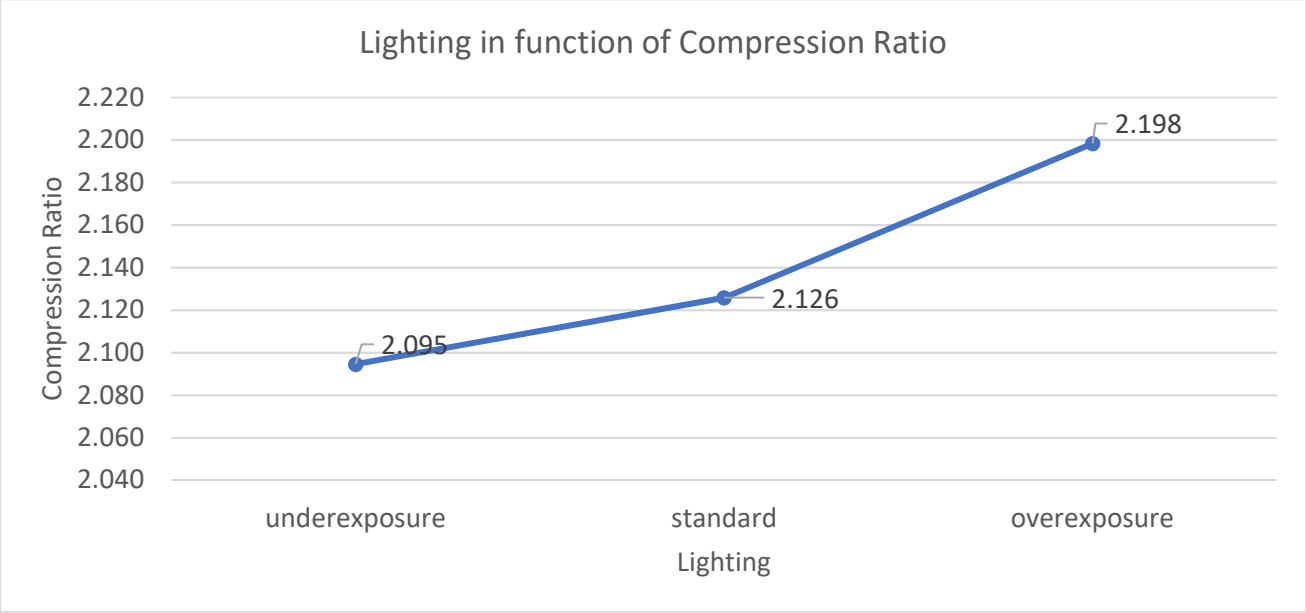


Figure 24: Lighting in function of compression ratio

4.3 Influence of intensity shifting

The tests were performed on several stock images with different colored objects and varying objects in the background. The prediction was an increase in compression with an increase in objects since the amount of unpredictable intensity changes increase when objects of varying color are located near one another. Results met the prediction. The compression ratio goes up when the number of objects decreases.

First, the test is explained. The stock images contained 4 slightly varying pictures of the same objects. Removing or shifting an object each image. The average of the compression ratio of each corresponding image is calculated, for m = 2, 4, 8, 16 and 32, and used to represent the overall compression ratio of their respective images.

Second, the relation between the nature of the images and the compression ratio is studied.

The lighthouse image shows an image with large intensity shifting in all part of the picture. The compression ratio is inadequate to expectations. The presence of shades and random patterns all over the picture makes calculation the upcoming pixel values difficult.

Next, the testchart image. The picture shows a dark largely reoccurring pattern with small shifts in intensity found in the picture. The compression ratio meets expectations for the image. The low variation in intensity causes errors but the value of the error is low since the surrounding pixel intensities are low.

Following is the discussion of the webcam image. The image shows a poorly lit room with varying objects and shapes with overall similar intensities save a few exceptions. The compression is adequate to expectations. The large amount of variation and

shadow in this picture cause errors but the number of generic color and intensity make up for these errors.

Last, the testimage is examined. The picture shows an enormous amount of varying color and intensity throughout the picture in a well-lit room. Compression of this image is below average because of the large amount of intensity variation. The image was taken in good lighting causing the compression to be better than the lighthouse image.

Table 7: Nature of pictures influence on the compression ratio for the average compression of the images

	Compression ratio
captured	1.6926
lighthouse	1.1777
testchart	1.7271
webcam	1.7806
testimage	1.2945

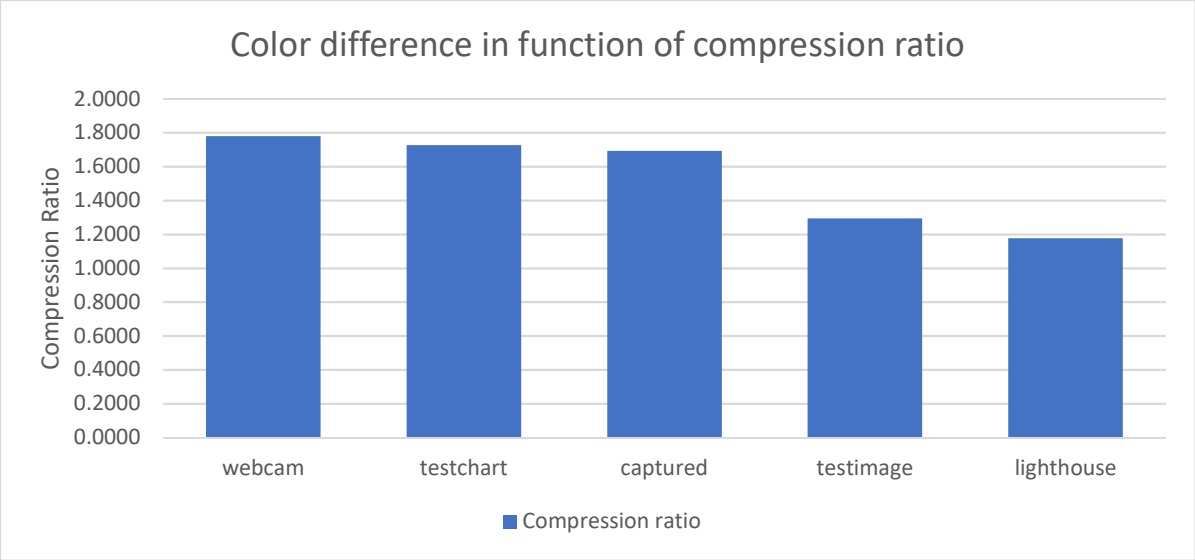


Figure 25: Color difference in function of compression data

4.4 Discussion

The testing of the implemented method concludes an average compression ratio of 1.6716 for varying images. for images in RAW format taken by the camera, the ratio increases to 2.8312 In other words, a standard pixel value can be stored in 3 to 4 bits where originally it was stored in 8. There is an effective gain of 4-5 bits per pixel (BPP). When comparing these values with the results of other lossless compression methods it can be concluded that commercialized compression methods like JPEG-LS and SLIC achieve a better ratio for RGB images.

When comparing the method to other lossless RAW image compression schemes it is shown that some methods reach a better BBP average. The JPEG-LS' results, having lost its RGB values, have dropped significantly. The BPP reached by this work has an

average of 4.7858 which is slightly worse. Two tables of results from other works are given for different lossless compression methods. Information was taken from [35] for RGB images and [36] for CFA images.

When comparing these to values found in [2], [3], [26], [35], [36], it can be concluded that this method is a valid alternative for lossless raw image compression. Better results can be achieved with a study into the best prediction algorithm for Golomb-Rice encoding or testing different compression algorithms for RAW images i.e. Huffman entropy encoding.

The decompression method reads the bitstream and counts the 1's until it reaches a 0. The 0 represents the end of the unary encoding signaling the start of the remainder. The remainder is read in as a value and added after the number of bits unary encoded times the divider m. The results of the decompression are loaded into the dePaeth module to restore the original value. The implementation on hardware was not optimized for speed or optimization.

For the implementation on FPGA, the suggest value for M and K is respectively 8 and 3 or 16 and 4. The reasoning is the matlab images contain intensity values from 0-255 while the FPGA has a maximum intensity value of 2048. Larger errors are possible and the implementation must be adjusted accordingly.

Table 8: Lossless compression method comparison for RGB images[35]

RGB	BPP
Huffman	5.295
Runlength	7.599
Huffman + Runlength	5.236
LZW	6.048
Arithmatic	4.625
JPEG-LS	4.190
SLIC	4.484

Table 9: Lossless compression method comparison for CFA images[36]

CFA	BBP
JPEG-LS	5.649
JPEG2K	5.953
LCMI	4.198
CMBP	4.025

5 Conclusion

Raw image compression can be useful for the implementation of multi-camera systems. Preserving the image quality and removing noise is crucial for the creation of the 3d image. The amount of applications previously designed on the joint denoising and compression of RAW images is limited. Research is focused on improving RGB image applications. The works shows that at least the same amount of compression and precision can be achieved by lossless RAW image compression.

Implementation on the FPGA can be achieved effectively when the right design choices are made with hardware implementation in mind. The divider m should be chosen according to the nature of the images taken. Larger image intensities will benefit from having a larger divider while images with little varying shades can be stored more efficiently in fewer bits. The work shows that the compression ratio is negatively affected by differing intensity extremes in the images. Features like shadows and sudden intensity changes should be avoided for this compression method to work effectively.

The compression method reached an effective compression rate of 1.6716 for natural images which translates to a BPP rate of 4.7858 almost matching general lossless compression methods for RGB and RAW imagery.

Future works can study the different compression algorithm and alter the predictors for varying applications. More work into the compression and denoising of RAW images can be performed. There have been few studies on the concept of RAW image denoising and compression since most works focus on one problem be it denoising or compression. Few works have previously been completed studying both. The study of using wavelet transformations and some joint denoising and demosaicking methods might achieve interesting results.

References

[1] S. H. Park, H. S. Kim, S. Lansel, M. Parmar, and B. A. Wandell, “A case for denoising before demosaicking color filter array data,” Conf. Rec. - Asilomar Conf. Signals, Syst. Comput., no. 4, pp. 860–864, 2009.

[2] K. Chung and Y. Chan, “A Lossless Compression Scheme for Bayer Color Filter Array Images,” pp. 1–24.

[3] K. Chung and Y. Chan, “A Lossless Compression Scheme for Bayer CFA Images,” no. Eusipco, pp. 649–652, 2007.

[4] B. Smolka and K. Martin, “Vector Filtering for Color Imaging [,” no. January 2005, pp. 74–86.

[5] H. J. Trussell and R. E. Hartwig, “W y W x W H W W f,” vol. 11, no. 4, pp. 485–492, 2002.

[6] D. Paliy, V. Katkovnik, R. Bilcu, S. Alenius, and K. Egiazarian, “Spatially adaptive color filter array interpolation for noiseless and noisy data,” Int. J. Imaging Syst. Technol., vol. 17, no. 3, pp. 105–122, 2007.

[7] D. Paliy, A. Foi, R. Bilcu, and V. Katkovnik, “Denoising and interpolation of noisy Bayer data with adaptive cross-color filters,” Proc. SPIE, vol. 6822, p. 68221K–68221K–13, 2008.

[8] A. Danielyan, M. Vehviläinen, A. Foi, V. Katkovnik, and K. Egiazarian, “Cross-color BM3D filtering of noisy raw data,” 2009 Int. Work. Local Non-Local Approx. Image Process. LNLA 2009, no. 118312, pp. 125–129, 2009.

[9] C. C. Koh, J. Mukherjee, and S. K. Mitra, “New efficient methods of image compression in digital cameras with color filter array,” IEEE Trans. Consum. Electron., vol. 49, no. 4, pp. 1448–1456, 2003.

[10] L. Zhang, X. Wu, S. Member, D. Zhang, and S. Member, “Color Reproduction From Noisy CFA Data of Single Sensor Digital Cameras,” vol. 16, no. 9, pp. 2184–2197, 2007.

[11] W. Zhang, B. Stukken, C. Chen, and L. Claesen, “Hardware Efficient Lossless Video Compression Methods for Low-Latency Multi-Camera Systems,” pp. 4–7.

[12] A. Libert and R. Palmans, “Compression techniques for computable multi-camera video systems.” .

[13] D. A. Huffman, “A Method for the Construction of Minimum-Redundancu Codes,” A Method Constr. Minimum-Redundancu Codes, pp. 1098–1102, 1952.

[14] J. Arvo, “IMAGE FILE COMPRESSION,” 1987.

[15] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, “Practical Poissonian-Gaussian noise modeling and Þ tting for single-image raw-data,” vol. 17, no. 1, pp. 1–18, 2007.

[16] A. Daaboul, “Local Prediction for Lossless Image Compression 1 Introduction 2 Predictors techniques,” pp. 44–50.

[17] Y. Yoo, S. Lee, W. Choe, and C.-Y. Kim, “CMOS image sensor noise reduction method for image signal processor in digital cameras and camera phones,” Digit. Photogr. III, vol. 6502, pp. 1–10, 2007.

[18] L. Zhang, R. Lukac, X. Wu, S. Member, and D. Zhang, “PCA-Based Spatially Adaptive Denoising of CFA Images for Single-Sensor Digital Cameras,” vol. 18, no. 4, pp. 797–812, 2009.

[19] A. M. Reza, R. D. Turneyi, L. Drive, and S. Jose, “FPGA Implementation of 2D Wavelet Transform.”

[20] C. S. Avinash, J. Sahaya, and R. Alex, “FPGA Implementation of Discrete Wavelet Transform using Distributed Arithmetic Architecture,” no. May, pp. 326–330, 2015.

[21] J. Portilla, V. Strela, and M. J. Wainwright, “Image Denoising using Scale Mixtures of Gaussians in the Wavelet Domain,” vol. 12, no. 11, 2003.

[22] L. Zhang, P. Bao, and X. Wu, “Multiscale LMMSE-Based Image Denoising With Optimal Wavelet Selection,” vol. 15, no. 4, pp. 469–481, 2005.

[23] S. G. Chang, S. Member, B. Yu, S. Member, and M. Vetterli, “Spatially Adaptive Wavelet Thresholding with Context Modeling for Image Denoising,” vol. 9, no. 9, pp. 1522–1531, 2000.

[24] A. Pi˘, “Estimating the probability of the presence of a signal of interest in multiresolution single- and multiband image denoising,” pp. 1–13.

[25] H. Shen and C. Fuh, “New Hierarchical Noise Reduction,” no. 3, pp. 1–8, 2009.

[26] S. H. Park, “Analysis on Color Filter Array Image Compression Methods.”

[27] S. W. GOLOMB, “Run-Length Encodings,” IEEE Trans. Inf. Theory, vol. IT-12, no. 3, pp. 399–401, 1966.

[28] A. Variable-length, “Adaptive Variable-Length,” 1971.

[29] “Paeth filter.” [Online]. Available: http://www.libpng.org/pub/png/spec/1.2/PNG-Filters.html.

[30] C. Wang, Y. Shen, Z. Zhu, and Y. Zhou, “Gradient-adjusted prediction of lossless image compression based on block-scan,” no. 1, pp. 1–4.

[31] K. S. Ng and L. M. Cheng, “LOSSLESS IMAGE COMPRESSION BY USING GRADIENT ADJUSTED PREDICTION,” vol. 45, no. 2, pp. 380–386, 1999.

[32] M. Confidential, “1/2.5-Inch 5-Megapixel CMOS Digital Image Sensor,” 2006.

[33] T. T. Hardware, “TRDB-D5M,” 2009.

[34] T. Trdb and D. C. Package, “Page Index.”

[35] D. Wu and E. C. Tan, “COMPARISON OF LOSSLESS IMAGE COMPRESSION ALGORITHMS,” pp. 718–721, 1999.

[36] S. Kim, S. Member, N. I. Cho, and S. Member, “Lossless Compression of Color Filter Array Images by Hierarchical Prediction and Context Modeling,” vol. 24, no. 6, pp. 1040–1046,

2014.

[37] Vanderwalle, “benchmark images.” [Online]. Available: lcav.epfl.ch.

[38] “Huffman picture,” 2013. [Online]. Available: http://stackoverflow.com/questions/14432503/how-is-this-huffman-table-created.

List of attachments

Attachments

Figure 26: varying exposure rate images [11]

Figure 27: lighthouse images 1 to 4 [37]

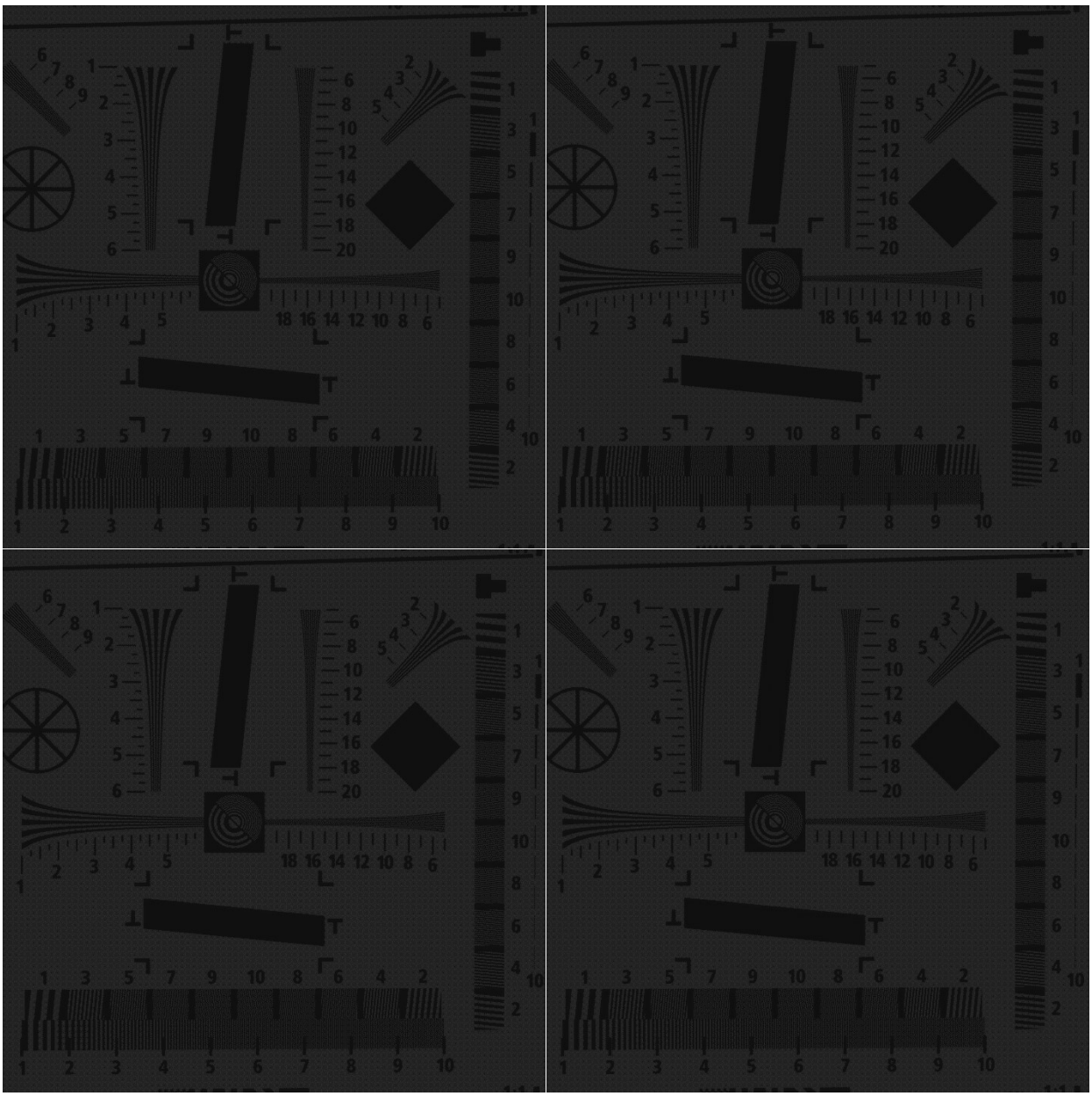


Figure 28: testchart images 1 to 4 [37]

Figure 29: testimage images 1 to 4 [37]

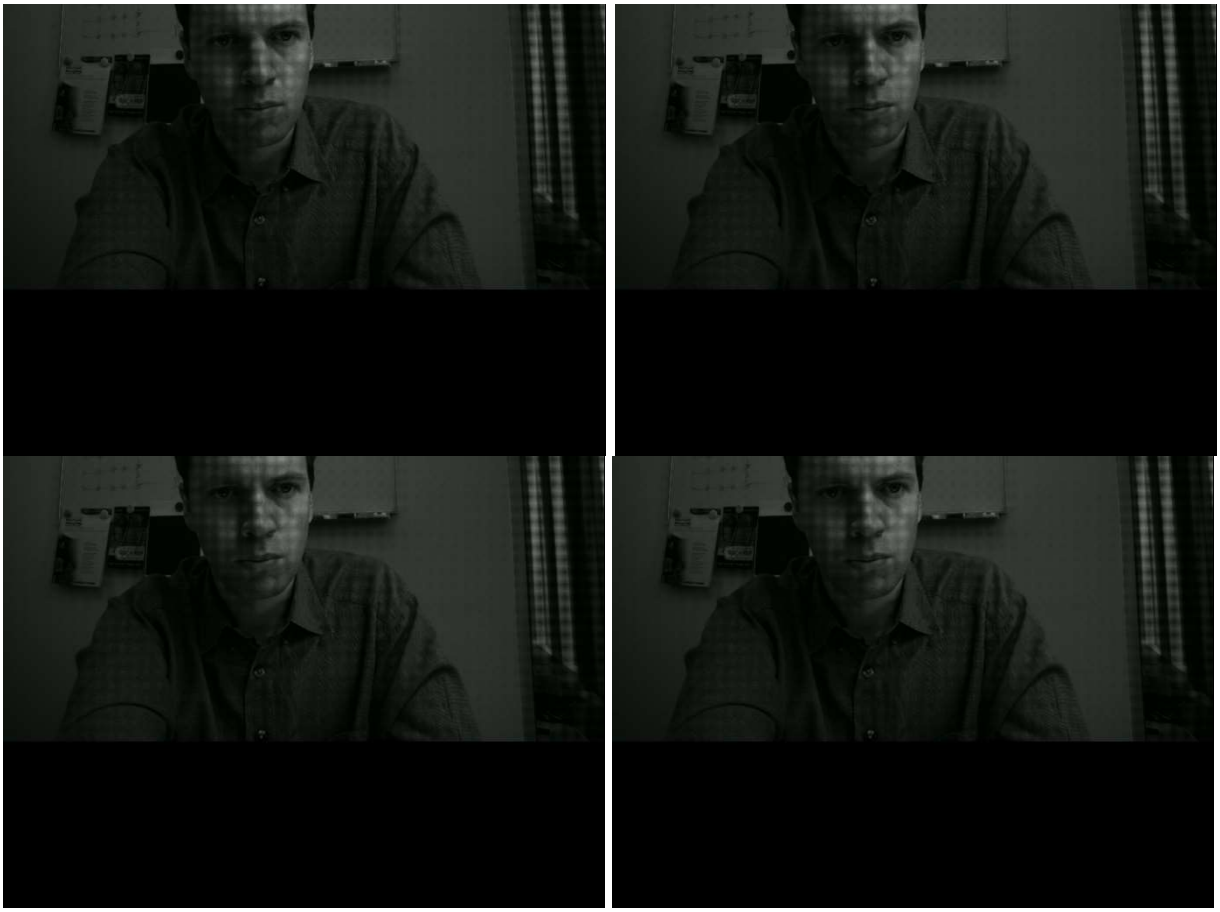


Figure 30: webcam images 1 to 4 [37]

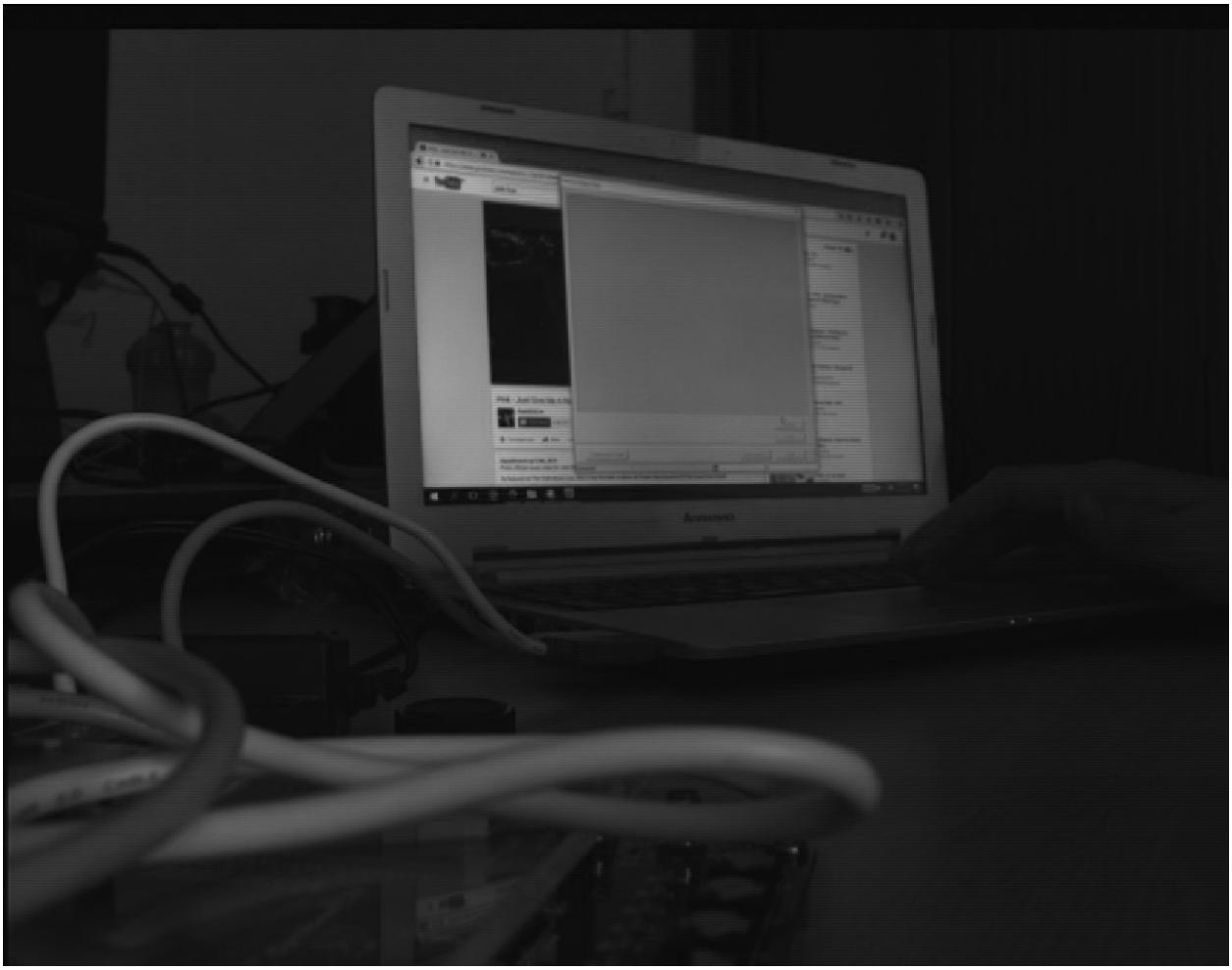


Figure 31: original image large

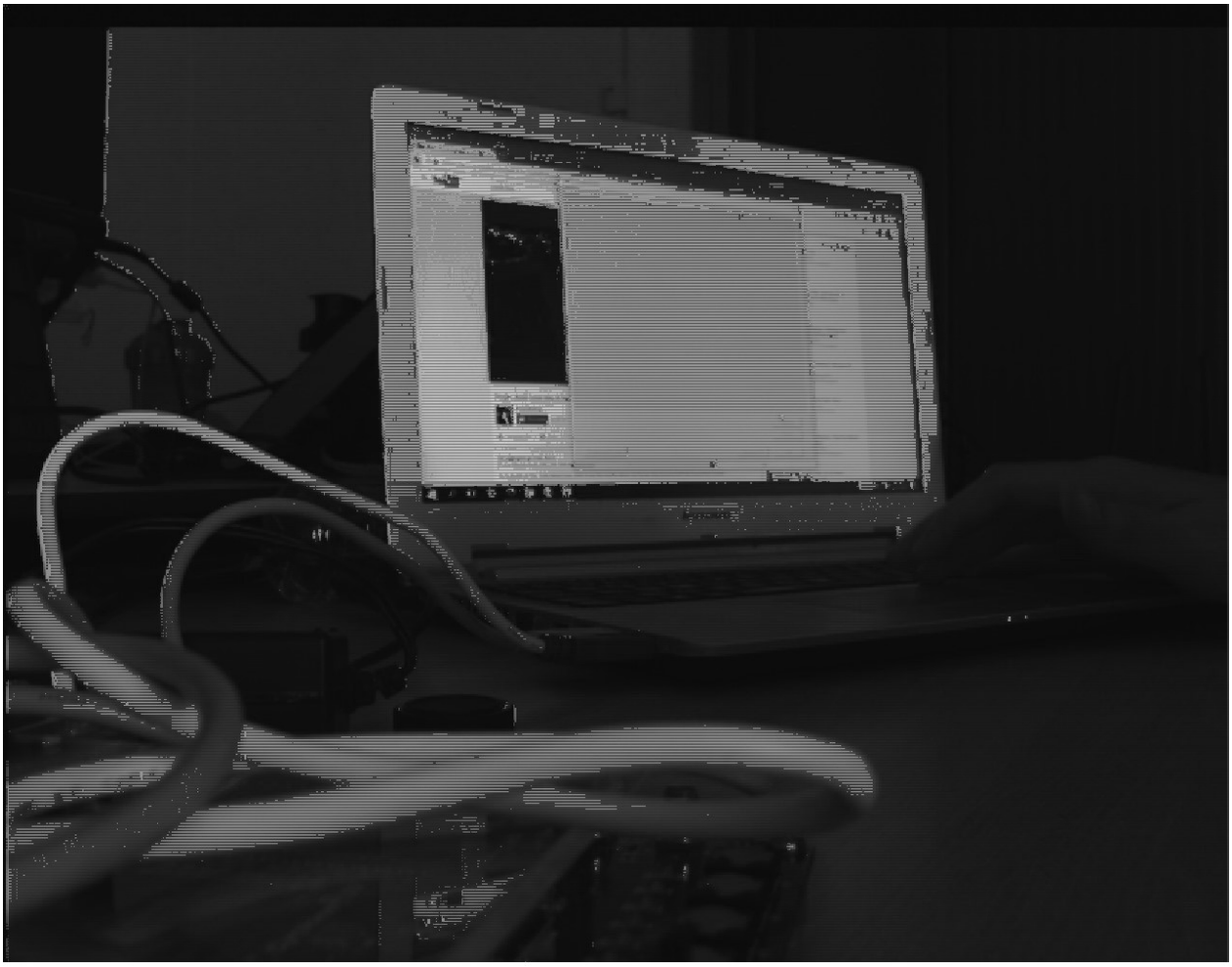


Figure 32: predicted image large

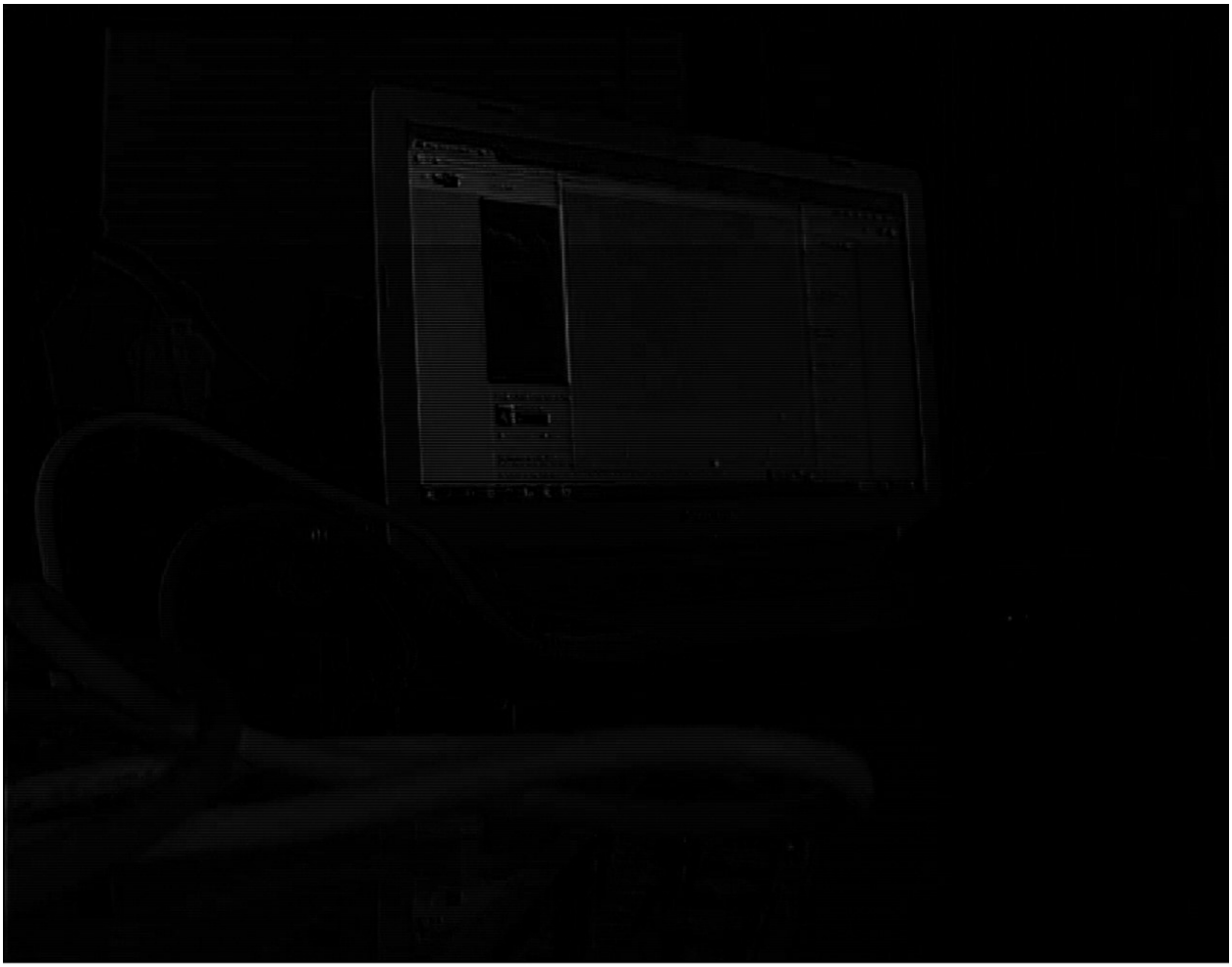


Figure 33: error image large

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Lossless compression of RAW image data on the FPGA

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2017**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Libert, Arno

Datum: **6/06/2017**