Comparison of Predictive-Corrective Video Coding Filters for Real-Time FPGA-based Lossless Compression in Multi-Camera Systems

Comparison of Predictive-Corrective Video Coding Filters for Real-Time FPGA-based Lossless Compression in Multi-Camera Systems

Bart Stukken
Hasselt University
Diepenbeek
Belgium

Yimu Wang
Hasselt University
Diepenbeek
Belgium

Yu Bao
Communication Eng.
Yangzhou University
China

Caikou Chen
Communication Eng.
Yangzhou University
China

Luc Claesen
Hasselt University
Diepenbeek
Belgium

ABSTRACT
Combining multiple cameras in a bigger multi-camera system give the opportunity to realize novel concepts (e.g. omnidirectional video, view interpolation) in real-time. The better the quality, the more data that is needed to be captured. As more data has a direct impact on storage space and communication bandwidth, it is preferable to reduce the load by compressing the size. This cannot come at the expense of latency, because the main requirement is real-time data processing for multi-camera video applications. Also, all the image details need to be preserved for improving the computational usage in a later stage. Therefore, this research is focused on predictive-corrective coding filters with entropy encoding (i.e. Huffman coding) and apply these on the raw image sensor data to compress the huge amount of data in a lossless manner. This technique does not need framebuffers, nor does it introduce any additional latency. At maximum, there will be some line-based latency, in order to combine multiple compressed pixels in one communication package. It has a lower compression factor as lossy image compression algorithms, but it does not remove human invisible image features that are crucial in disparity calculations, matching, video stitching and 3D model synthesis. This paper compares various existing predictive-corrective coding filters after they have been optimized to work on raw sensor data with a color filter array (i.e. Bayer pattern). The intention is to develop an efficient implementation for System-on-Chip (SoC) architectures to improve the computational multi-camera systems.

Categories and Subject Descriptors
B.6.3 [**VHDL, Verilog**]: Language Constructs and Features – *lossless compression, image processing, system-on-chip.*

General Terms
Algorithms, Performance, Design, Experimentation, Theory.

FPGAWorld'15, September 8-10, Stockholm and Copenhagen.

Keywords
multi-camera, lossless compression, Bayer pattern, system-on-chip, SoC, entropy encoding, VLSI.

1. INTRODUCTION
Nowadays, CMOS image sensors are abundantly available due to the rapid evolution in VLSI technology. Many multi-media devices (e.g. smartphones, tablets) contain one or more high-quality cameras. By combining multiple of these cost-efficient cameras, it is possible to realize all kinds of novel concepts such as omnidirectional video and virtual viewpoint interpolation [5,6,7]. If we would extrapolate this trend, hundreds of cameras could work closely together to capture scenes of television plays or sports events in real-time from several fixed positions. The various viewpoints can collect all the information that is needed to synthesize any "virtual" viewpoint between the real cameras.

In a multi-camera system, it is possible to improve the end result by not only improving the recorded quality of each camera, but also to improve the number of (cheaper) cameras. However, this has a direct impact on the storage space and communication bandwidth. For example, if you have five 2MP cameras with 30 fps, the total raw data size that is captured reaches around 900 MB per second. Imagine the impact if the amount of cameras would increase to a hundred, unless the bandwidth requirement is not reduced by some compression, a bottleneck will arise.

Traditional compression standards (e.g. MPEG, H.264) reduce the size by removing redundant information for the human observer. Unfortunately, high frequency content or other image features that are invisible to the human eye, are crucial for computational purposes in multi-camera systems. These systems need to combine the visual data from several image sensors and their quality relies a lot on the accuracy of such high spatial frequency clues for accurate and detailed matching of features captured by the different cameras. For future improvements to disparity calculations, matching, video stitching and 3D model synthesis, all the images features need to be kept intact. This limits this research to lossless compression techniques, although the compression factor is lower than the previously mentioned lossy compression algorithms.

Furthermore, to achieve the lowest possible latency in a future System-on-Chip (SoC) architecture, this research is focused on

predictive-corrective coding filters as it does not need framebuffers to do the lossless compression of the image sensor data. The first step of the encoding sequence can be calculated directly from the active sensor data. The second step using entropy encoding as the compression technique, needs multiple pixels to compress it to one communication package and to gain in the long run. This, however, will introduce line-based latency, but there is no other known technique that preforms better for lossless compression with regards to latency. Lossless image compression algorithms that make use of image tiling will have a higher latency, because the entire tile needs to be captured before it can be send. This puts the minimum latency as high as the height of the tile in scan line numbers, thus latency = tile height × image width.

There are already several well developed predictive-corrective coding filters (e.g. DCPM [1], Paeth [2], GAP [3], JPEG-LS [4]), but they are designed and optimized for full color images. As the goal is to develop a SoC architecture, the filter can be applied on raw sensor data with a color filter array and improve the compression factor for each individual camera in the multi-camera system. In this work, the predictive-corrective coding filters will be adjusted to work on a color filter array (i.e. Bayer pattern) as these kind of cameras are going to be used in future work.

This paper is structured as follows. In the next section the next predictive-corrective coding filters for lossless image compression are explained : DCPM, Paeth, GAP and JPEG-LS. In section III the method for compression, entropy encoding, is elaborated. The results from the applied Huffman coding on the error-corrective values given by the predictive-corrective coding filters, are also displayed in section III. These can be used to compare the results in section IV, in which the algorithms have been updated to work with a color filter array (i.e. Bayer pattern). It is noticeable that the prediction algorithms on Bayer patterned mosaic images work better (i.e. a compression factor of 5.3:1) than the normal RGB compression can reach (i.e. a compression factor of 2:1). Section V concludes with a discussion on the lossless image compression results.

2. PREDICTIVE-CORRECTIVE CODING FILTERS

All the predictive-corrective coding filters try to find the best estimation of a certain pixel, based on previous known pixel values. As the image sensor will transmit the 2D image in a pixel serial order (i.e. left to right for each row and from top to bottom), a prediction can only be done for the next pixel in line. Therefore, none of the future information (i.e. on the right and below the current pixel) can be used, unless there is a framebuffer. But this research will not look into the techniques that use framebuffers, as that is much more memory intensive. Predictive-corrective coding filters only need a limited amount of line buffer, depending on the chosen technique.

Once a estimation of the currently transferred pixel value is done, the correction with the actual sensor value can be calculated. By only transferring these error-corrective values, the original image can still be recalculated lossless in a later stage. This process is fully illustrated in figure 1.

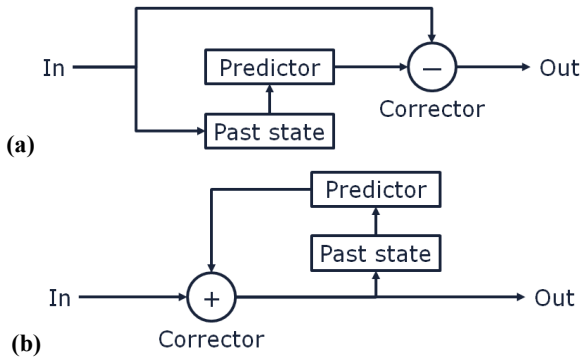


Figure 1. Predictive-corrective coding filter schematic diagram : (a) encoding image to error-corrective data stream, (b) decoding an error-corrective data stream back to the original image.

The better the prediction, the closer the prediction is to the actual value, and the smaller the error is, and the closer the error-corrective value is to zero. By gathering a lot of the same values (i.e. all of the values distributed near zero), it is possible to apply an entropy encoding technique (e.g. Huffman coding) to reduce the overall size.

2.1 DPCM

The 2D form of the Differential Pulse Code Modulation (DPCM) is one of the simplest forms of a predictive-corrective coding filter used for image compression. This technique has been added to the JPEG standard as Lossless JPEG in 1993 [1]. It makes a prediction of the next pixel by applying just one formula that calculates the interpolation of the three previous and neighboring pixels. As shown in figure 2, the prediction value is calculated as follows P = B + C – A. Due to this formula, it is possible to calculate over- and underflow. To keep the values in range, these are clipped within the possible range.

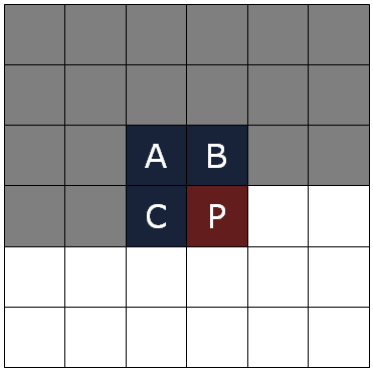


Figure 2. Predictive-corrective coding filters, DPCM and Paeth, use only the three previous and neighbouring pixels as window to calculate a prediction for P.

2.2 Simple predictive-corrective coding

For the PNG Working Group, Alan W. Paeth has proposed an algorithm that has a better prediction result than DPCM [2]. It uses the same window and formula as DPCM (see figure 2) to do the prediction (i.e. three previous and neighboring pixels), but the predicting algorithm is different. It will select a previous pixel value (i.e. A, B or C) that is closest to the calculated DPCM value. In pseudo code, this algorithm would look like the following sequence for each pixel :

```
P = B + C – A;
IF |P – A| ≤ |P – B| AND |P – A| ≤ |P – C|
    P = A; //A is closest to P
ELSE IF |P – B| ≤ |P – A| AND |P – B| ≤ |P – C|
    P = B; //B is closest to P
ELSE
    P = C; //C is closest to P
END
```

2.3 Context-based, adaptive lossless image coding

In response to the request for new techniques for lossless image compression from the JPEG Working Group, a new algorithm has been invented. Xiaolin Wu has proposed a Context-based, Adaptive Lossless Image Coding (CALIC) [3] that also uses a window of previous and neighboring pixels, but it is expanded up to seven previous known values to do a more substantiated prediction. As depicted in figure 3, the prediction has a much bigger window to make a prediction.

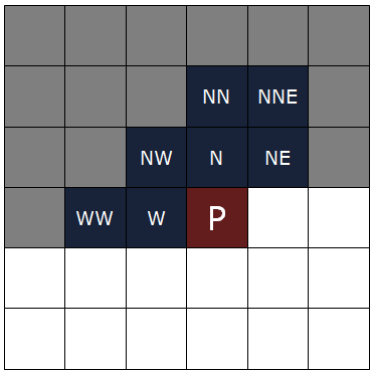


Figure 3. The window that is used by GAP algorithm uses the seven previous and neighbouring pixels to make a prediction for P.

By using more aligned pixel values, it is possible to use a Gradient-Adjusted Prediction (GAP). The GAP algorithm that is used in the CALIC design, also tries to take several different edge types into account. Therefore, it accumulates all absolute vertical changes and compares it with the accumulation of all the absolute horizontal changes. The difference between these two sums of absolute differences, has been parameterized to use different formulas on different types of edges. This is visible in the following pseudo code sequence that will be used for every pixel prediction :

```
Dh = |W – WW| + |N – NW| + |N – NE|; //horizontal
Dv = |W – NW| + |N – NN| + |NE – NNE|; //vertical
Edge = Dh – Dv;
IF Edge > 80 //sharp horizontal edge
    P = W;
ELSE IF Edge > 32 //horizontal edge
    P = W / 2 + (W + N) / 4 + (NE – NW) / 8;
ELSE IF Edge > 8 //weak horizontal edge
    P = W / 4 + 3 * (W + N) / 8 + 3 * (NE – NW) / 16;
ELSE IF Edge < –80 //sharp vertical edge
    P = N;
ELSE IF Edge < –32 //vertical edge
    P = N / 2 + (W + N) / 4 + (NE – NW) / 8;
ELSE IF Edge < –8 //weak vertical edge
    P = N / 4 + 3 * (W + N) / 8 + 3 * (NE – NW) / 16;
ELSE
    P = (W + N) / 2 + (NE – NW) / 4;
END
```

2.4 JPEG-LS

After the first proposition of JPEG-LS as a new Lossless JPEG technique that only detects horizontal and vertical edges and uses a different formula for the prediction, improvements where proposed to implement a diagonal edge-based prediction algorithm [4]. The window that is used for each edge-based prediction, is defined by the four previous and neighboring pixels as displayed in figure 4.

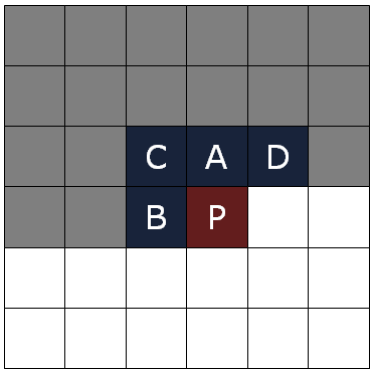


Figure 4. JPEG-LS uses four previous and neighbouring pixel to make an edge-based prediction for P.

Depending on the predefined thresholds, the algorithm will detect different types of edges. It is therefore important to determine thresholds that work well on a broad range of pictures. In this research, the two thresholds (T1 = 60 and T2 = 8) are founded on the results in [4]. Only when the top left pixel C is higher or lower than the two closest neighboring pixels A and B, an edge is detected. In all other cases, old DPCM formula is used to do a prediction. When de difference between C and its neighbors A and B is bigger than threshold T1, and the difference between A and B is not that bigger than threshold T2, a diagonal DPCM formula is used (i.e. P = B + D – A). For the vertical edges, one of the neighboring pixel values A or B is used. The following pseudo code clarifies the logic behind the algorithm, its edge detection and the different prediction formulas :

```
IF C ≥ max( A, B ) //light to dark edge
    IF C – max( A, B ) > T1 AND |A – B| ≤ T2 //diagonal
        P = B + D – A;
    ELSE
        P = min( A, B );
    END
ELSE IF C ≤ min( A, B ) //dark to light edge
    IF min( A, B ) – C > T1 AND |A – B| ≤ T2 //diagonal
        P = B + D – A;
    ELSE
        P = max( A, B );
    END
ELSE //no clear edge
    P = A + B – C
END
```

3. ENTROPY ENCODING

In previous predictive-corrective coding filters, the prediction errors are distributed near zero, which is well-suited for the subsequent coding. To compress these error-corrective values, the most frequent value must use the least amount of space at the expense of the values that are very rare. Therefore, in this research the Huffman coding is used as it is not the focus to compare entropy encoding techniques. Any optimization can always be obtained once the best predictive-corrective coding filters is found. Just using one entropy encoding technique is enough to demonstrate and compare the compression results. Figure 5 shows the histogram of pixel values before and after a predictive-corrective coding filter has been applied.

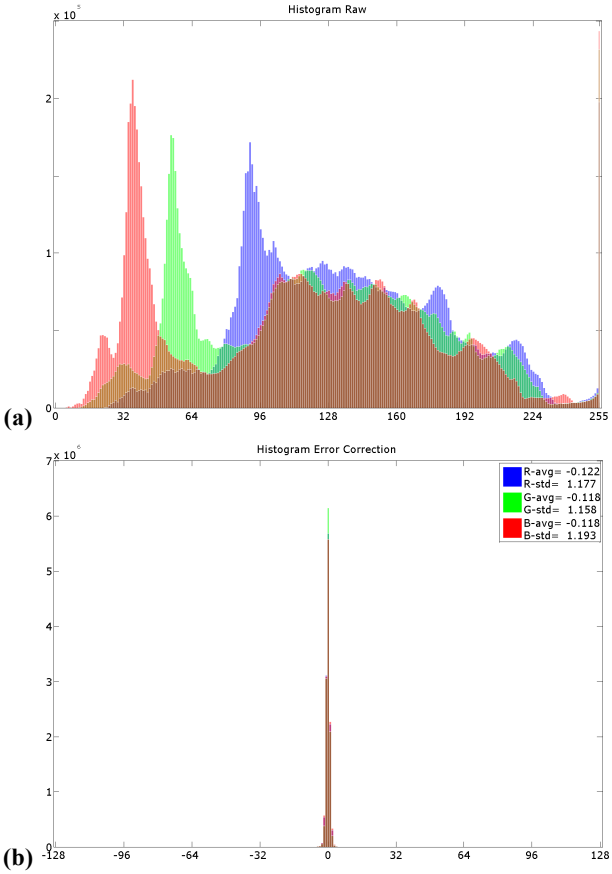


Figure 5. Pixel value histogram : (a) the full color of an image are distributed over the entire value spectrum, (b) the error-corrective values are close to zero.

In order to make a good comparison, all the predictive-corrective coding filters have been applied to the high resolution benchmark images from The *New Image Compression Test Set*[1]. The results of the compression factors after the Huffman coding are shown in table 1. To have a correct baseline the results from only applying Huffman coding is also included (i.e. the "none" filter). The compression factor = original size / compressed size, making the highest number the best compression algorithm.

Table 1. Compression factors of the benchmark images after lossless compression

Image	None	DPCM	Paeth	GAP	JPEG-LS
artificial	1.271	4.848	5.077	4.775	**5.078**
big_building	1.059	1.845	1.877	1.901	**1.916**
big_tree	1.133	1.679	1.729	**1.776**	1.760
bridge	1.075	1.660	1.684	**1.739**	1.716
cathedral	1.206	1.877	1.942	1.967	**1.974**
deer	1.266	1.362	1.440	**1.489**	1.451
fireworks	2.242	3.461	3.597	**3.650**	3.641
flower_foveon	1.122	2.987	3.197	**3.387**	3.277
hdr	1.135	2.617	2.833	**2.949**	2.885
leaves_iso_1600	1.089	1.574	1.589	**1.630**	1.618
leaves_iso_200	1.083	1.797	1.781	**1.839**	1.829
nightshot_iso_100	1.240	2.886	3.035	3.066	**3.083**
nightshot_iso_1600	1.292	1.692	1.771	1.780	**1.782**
spider_web	1.040	3.917	3.879	4.017	**4.022**
TOTAL	**1.151**	**1.971**	**2.026**	**2.070**	**2.062**

In the comparison above, this research is only comparing the technique for the inside of the picture. All the special border algorithms are being dropped as these small optimizations should not have any influence on the end result. To conclude, the GAP and JPEG-LS are very good candidates for compression, with a small preference to the GAP. The lossless compression factor is around 2:1 for a various kind of images, but as this research is focused on the compression of the hardware image sensor data, the current techniques need to be adjusted.

4. COLOR FILTER ARRAY

In order to capture full color images with an image sensor, many different techniques arose. One of the most used techniques is the use of a color filter array on top of the image sensor. By filtering a specific color for a single light sensor, only that color is captured. Using a fixed color pattern, like the Bayer pattern, the raw sensor data will transmit mosaic images. Afterwards, the full color images can be obtained with the use of demosaicing algorithms. As portrayed in figure 6, each pixel contains originally only one color value.

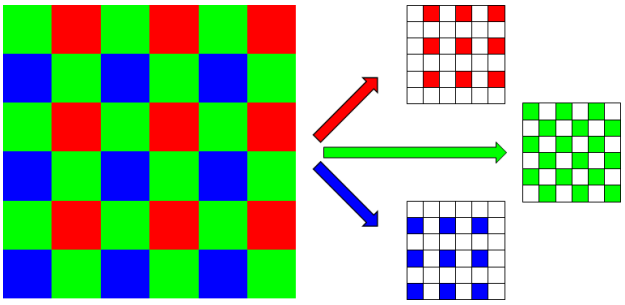


Figure 6. Bayer pattern as color filter array, only one color value is captured per pixel on the raw image sensor before the demosaicing blends them into a full color picture.

This would suggest that the images could have a lossless compression factor of 3:1 when saving the raw mosaic image. The demosaicing algorithm can always be redone and thus makes previous compression algorithms look useless. However, in this research both techniques are being combined. Therefore, the predictions have to be separated for each color and adjusted to

[1] http://www.imagecompression.info/test_images

use the correct neighboring pixels. A primary and easy approach would be to just use the same technique, but jump two pixels in each direction. For the red and blue color, this is sufficient. However, the green color could have used its diagonal neighboring pixels to do a better prediction. The results shown in table 2, indicate that the compression can be higher than 2:1 or 3:1 by just using this first approach. Also the "none" filter has been used to provide a better baseline for comparison.

Table 2. Compression factors of the benchmark images after lossless compression with Bayer pattern

Image	None	DPCM	Paeth	GAP	JPEG-LS
artificial	3.805	11.606	**12.754**	11.433	12.624
big_building	3.177	4.496	4.672	**4.740**	4.732
big_tree	3.368	4.142	4.340	**4.474**	4.385
bridge	3.219	4.137	4.302	**4.439**	4.350
cathedral	3.618	4.552	4.769	**4.901**	4.816
deer	3.795	3.877	4.125	**4.283**	4.152
fireworks	6.622	8.457	8.926	**9.211**	8.994
flower_foveon	3.489	7.681	8.230	**8.518**	8.365
hdr	3.426	6.602	7.170	**7.449**	7.256
leaves_iso_1600	3.238	3.958	4.051	**4.148**	4.097
leaves_iso_200	3.228	4.224	4.327	**4.425**	4.383
nightshot_iso_100	3.682	6.783	7.228	**7.421**	7.299
nightshot_iso_1600	3.826	4.507	4.703	**4.840**	4.738
spider_web	3.121	7.566	8.026	**8.406**	8.204
TOTAL	**3.442**	**4.868**	**5.099**	**5.225**	**5.155**

In a first simple approach, it is noticeable that GAP is the better algorithm in combination with the Bayer pattern. It has an average compression factor of 5.2:1 which is much better than the 3:1 or 2:1 discussed before. To be correct, it must be mentioned that for this research the GBRG Bayer pattern (as in figure 6) has been chosen and reverted from the benchmark images. Therefore, the results only give a good indication, but are not the most correct values. Future work will be done on the raw sensor data which has the correct color filter array.

In attempt to study the influences of improving the green prediction algorithm, a few changes has been made to the existing predictive-corrective coding filters. For the DPCM and Paeth, the window has been upgraded to also use the diagonal neighboring pixel as demonstrated in figure 7. The DPCM prediction value is also calculated differently as the average between the 2D and diagonal interpolation. The formula is as follows $P = (B + C - A) + (2 * D - A)$. As Paeth uses this to find the closest neighboring pixel value, the only improvement is that it also can pick the pixel value of D, if that one is closest.

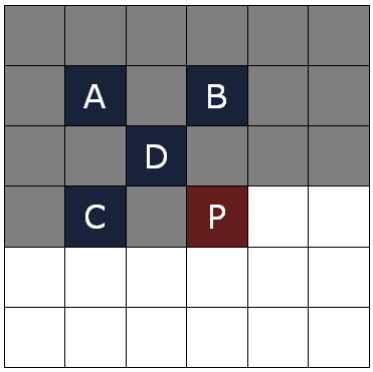


Figure 7. The improved window for the green pixels in mosaic image with a Bayer pattern, used for the predictive-corrective coding filters DPCM and Paeth.

For GAP and JPEG-LS a simple shift has been applied to the rows. While the horizontal neighboring pixels still require a two pixel jump, the vertical pixels are diagonal and require an additional jump to the left when going an odd number upwards. Figure 8 gives a clear view how these two algorithms windows have changed for the green color in a mosaic image.

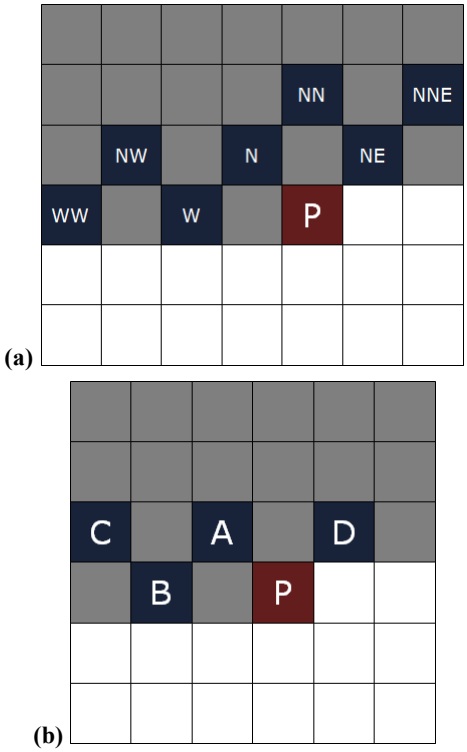


Figure 8. A simple shift applied to the green color windows of mosaic images with Bayer pattern in predictive-corrective coding filters : (a) GAP, (b) JPEG-LS.

These improved predictive-corrective coding filters are being tested on the Bayer patterned mosaic images of the benchmark set. The results are shown in table 3 and can be compared with all previous results.

Table 3. Compression factors of the benchmark images after improved lossless compression with Bayer pattern

Image	None	DPCM	Paeth	GAP	JPEG-LS
artificial	3.805	11,586	**13,210**	11,714	12,867
big_building	3.177	4,559	4,733	**4,857**	4,721
big_tree	3.368	4,206	4,403	**4,556**	4,432
bridge	3.219	4,184	4,337	**4,472**	4,331
cathedral	3.618	4,630	4,849	**5,021**	4,892
deer	3.795	3,918	4,135	**4,290**	4,153
fireworks	6.622	8,613	9,097	**9,490**	9,090
flower_foveon	3.489	7,835	8,346	**8,740**	8,456
hdr	3.426	6,722	7,326	**7,631**	7,284
leaves_iso_1600	3.238	4,032	4,124	4,226	**4,405**
leaves_iso_200	3.228	4,329	4,432	**4,533**	4,119
nightshot_iso_100	3.682	6,879	7,338	**7,555**	7,230
nightshot_iso_1600	3.826	4,563	4,748	**4,893**	4,756
spider_web	3.121	7,791	8,213	**8,720**	8,166
TOTAL	**3.442**	**4,943**	**5,170**	**5,326**	**5,170**

GAP is again the best overall algorithm and has like most other filters a compression factor gain of 0.1, making the end result 5.3:1. Although, this is a better result than previous reached, much improvements can be found in working directly with the color filter arrays. Future work should be done in the direction of improving the predictor with values from the other colors in the color filter array.

5. FPGA IMPLEMENTATION

To put the theory into practice, a real-time implementation is made on an Altera Cyclone-II EP2C70 FPGA equipped with a 5 mega pixel camera module inserted into the GPIO slot. This can be seen on figure 9. This camera module streams its raw Bayer pattern image data to the FPGA core with an implementation of the simple predictive-corrective coding algorithm. Each consecutive clock cycle during the image transfer, a pixel value is retrieved from the image sensor, a prediction is made by using the algorithm of Paeth. The error correction is then obtained by subtracting these two values.

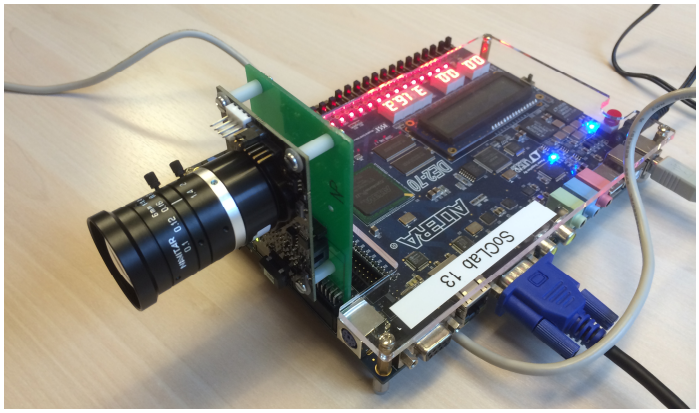
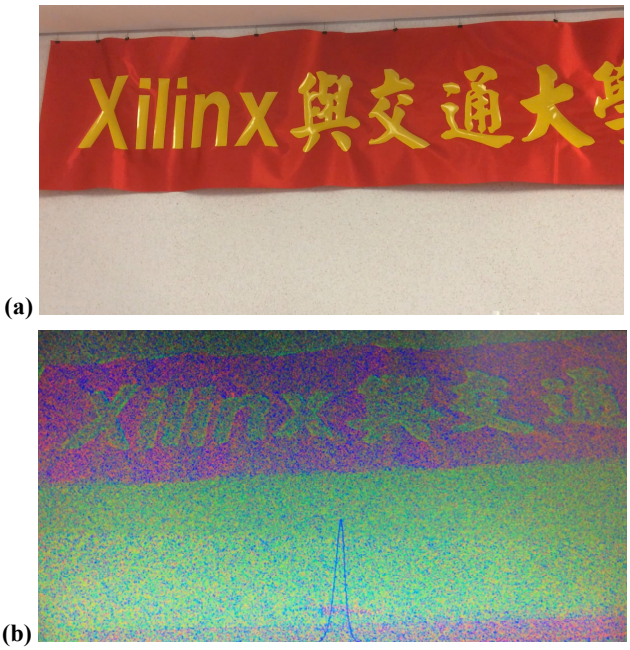


Figure 9.FPGA implementation on a DE2-70 with a D5M camera module for real-time simple predictive-corrective coding.

For demonstration purposes and visible in figure 10, these error corrective values are enhanced and displayed in real-time via the VGA signal onto a connected monitor. These values are also collected and used to calculate and display a histogram. Therefore, it the prediction algorithm is visible and in particular the influences of scenery. For instance, the histogram adapts dynamically and has a wider base on a more complex scene (i.e. harder to do a correct prediction), or has a peak when there is an overflow (i.e. too much lightning can give a big spot with all the same high value). In Table IV an overview of the hardware utilization for the FPGA implementation is summarized.

Table 4. FPGA demonstrator hardware utilization

Characteristic	Used	Total Available	% Utilization
Total logic elements	3,085	68,416	5
Total combinatorial functions	2,355	68,416	3
Dedicated logic regisers	2,021	68,416	3
Total registers	2,065		
Total memory bits	587,256	1,152,000	51
Multipliers	0	300	0



(a)

(b)

Figure 10. Real-time results of the FPGA implementation : (a) the original scene that is captured by the camera, (b) the error corrective values that are enhanced and the historgram of these values.

6. CONCLUSION

In this paper, it is proven that a synergy can be reached by combining a color filter array (e.g. Bayer pattern) with known predictive-corrective coding filters and an entropy encoding for a higher overall lossless compression of images. The four altered techniques (i.e. DPCM, Paeth, GAP and JPEG-LS) have been examined and compared with the use of a high resolution benchmarking image set. It is concluded that the update GAP algorithm has reached the best overall result. With an average compression factor of 5.3:1 it surpasses the same algorithm on the full color images, which only got a compression factor of 2:1. Nevertheless, there is still room for improvement as many different approaches have not been invented (e.g. Bayer pattern specific algorithms).

For our bigger research goal of implementing a compression algorithm in hardware for multi-camera systems, it already shows to apply these techniques before the demosaicing process. Therefore, in future work, the GAP algorithm will be implemented as a SoC architecture to reduce the bandwidth requirement in multi-camera systems, without compromising the latency. Although, this will not give enough room to have a hundred cameras interconnected, it is a big step forwards to reduce the communication bandwidth a fivefold. Other possible improvements (e.g. software-defined networking) benefit from this approach and are researched in our group in parallel.

The research in this paper was partly funded by the bilateral FWO-MOST (Belgian Research Council research cooperation contract G.0524.13.

7. REFERENCES

[1] Wallace, G.K., "The JPEG still picture compression standard," Consumer Electronics, IEEE Transactions on , vol.38, no.1, pp.xviii,xxxiv, Feb 1992

[2] Alan W. Paeth, "Image File Compression Made Easy", in Graphics Gems II (edited by James Arvo), Academic Press, 1995, ISBN 0-12-059756-X, pp. 93-101.

[3] X. Wu, N. Memon, "CALIC - A context based adaptive lossless codec", Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP-96, 7-10 May 1996, Vol. 4, pp. 1890-1893.

[4] Eran A. Edirisinghe, Satish Bedi, Christos Grecos, "Improvements to JPEG-LS via diagonal edge-based prediction.", in Proc. SPIE 4671, Visual Communications and Image Processing 2002, 604 (January 7, 2002);

[5] A. Motten, L. Claesen, Y. Pan, "Trinocular Stereo Vision using a Multi Level Hierarchical Classification Structure", chapter in "VLSI-SoC: From Algorithms to Circuits and System-on-Chip Design", editors: A. Coskun, A. Burg, R. Reis, M. Guthaus, Springer ISBN 978-3-642-45072-3, pp. 45-63.

[6] R. Szeliski, "Computer Vision: Algorithms and Applications", Texts in Computer Science 2011, Springer, ISBN: 978-1-84882-934-3.

[7] Abdulkadir Akin, "Real-Time High-Resolution Multiple-Camera Depth Map Estimation Hardware and Its Applications", Ph.D. Thesis EPFL Lausanne, 2015.