DJ Journal of Advances in Electronics and Communication Engineering, Vol. 4(1) 2018, pp. 16-23

RESEARCH ARTICLE

High Performance Adder-Based Stepwise Linear Interpolation

[*]C John Moses[1], D Selvathi[2], G Shaya Edal Queen[3]
[1]Associate Professor, Dept. of ECE, Sreyas Institute of Engineering and Technology, India.
[2]Professor, Dept. of ECE, Mepco Schlenk Engineering College, Sivakasi, India.
[3]Assistant Professor, Department of Electronics and Communication Engineering, James College of Engineering and Technology, Nagercoil- 629 852 India.

Received- 13 September 2017, Revised- 4 December 2017, Accepted- 21 December 2017, Published- 26 December 2017

ABSTRACT

Image interpolation is the process of increasing the number of pixels in an image to maintain the quality while enlarging an image. Quality and complexity vary with different interpolation methods. Filters can be used along with the interpolation techniques such that the quality of the image is increased. The hardware architecture of Multiplier-Based Linear Interpolation (MBLI), Adder-Based Stepwise Linear Interpolation (ABSI) and clamp filter-based ABSI interpolation algorithm are simulated in MATLAB Simulink and Xilinx ISE generator. The clamp filter-based ABSI provides better quality than other related methods.

Keywords: Image interpolation, Filtering, Multiplier based linear interpolation, Clamp filter, Adder based stepwise linear interpolation.

1. INTRODUCTION

Interpolation is the process of introducing new pixels into the original image [1]. It refers to the resizing of digital image. It is also known as up-scaling or resolution enhancement. An effective interpolation method does not include any complex structure or image [2]. Linear interpolation provides better image quality than the nearest neighbor interpolation. As the order goes on increasing the computational complexity also increases. Linear interpolation is frequently used in the reconstruction of images [3-6]. Linear interpolation is used to acquire data at specific points with selective sampling mechanism [7]. Linear interpolation reduces the number of multiplications with a slight increase in the number of additions. Still, since such fixed interpolation kernel based techniques are space-invariant, they do not provide required data of the missed pixels, and thereby result in blurring, ringing and zigzagging mostly around edges or complex regions without considering essential data of high frequency elements signifying edge characteristics and local patterns [8-10].

Different algorithms have been developed to improve the quality of image interpolation such as interpolation by axiomatic approach [11], interpolation by edge localization [12], statistical interpolation [13], interpolation using high quality measure [14] and extended linear interpolation [15]. However, these methods utilize complex algorithms to improve the quality and therefore increase the computational complexity. The proposed method utilizes low complexity ABSI with Clamp Filter (CF) to improve the quality of image interpolation.

The rest of this paper is organized as follows. Section 2 explains the interpolation algorithms MBLI and ABSI. Section 3 explains the proposed CF kernel. Section 4 gives a comparative analysis based on quantitative measures such as Peak Signal to Noise Ratio (PSNR), Structural Similarity

*Corresponding author. Tel.: +917339104796
Email address: erjohnmoses@gmail.com (C.J.Moses)
Double blind peer review under responsibility of DJ Publications
https://dx.doi.org/10.18831/djece.org/2018011003

(SSIM) and VLSI characteristics such as Look Up Tables (LUTs), gates count and memory requirements. Section 5 gives conclusion and scope of the research.

2. MBLI, ABSI AND ABSI WITH CLAMP FILTER

This section describes the linear interpolation formula and computation of address and distance generator required for linear interpolation. Linear interpolation uses first order polynomial equation to introduce new pixels. In ABSI method only simple operators that compare, shift and add are required. By using ABSI, the interpolated pixel values can be calculated using (2.1).

$$y_k = x_1 + (x_2 - x_1) >> (d - d_1)$$
$$y_k = x_2 + (x_1 - x_2) >> (d - d_2) \quad (2.1)$$

MBLI requires complex power consuming operators such as multipliers and dividers. MBLI provide better PSNR with a good image quality but consumes more power and area. By using MBLI, the interpolated pixel value is calculated using (2.2).

$$y_k = x_1 + (x_2 - x_1) \times (d_1/d)$$
$$y_k = x_2 + (x_1 - x_2) \times (d_2/d) \quad (2.2)$$

ABSI with CF is proposed to improve the quality of the image by increasing the PSNR value.

3. CLAMP FILTER-BASED LINEAR INTERPOLATION

The CF is a low pass filter that levels the undesired discontinuous edges and reduces block effect caused by interpolation system [16]. It is characterized by a convolution kernel which is combined with matrix coefficients that indicate the dependency of a filtered pixel over its neighbors. In addition, the array of the kernel is a single positive value which is present at its center and the other surrounded values are ones. The input image is filtered by using CF and the filtered image is downscaled by using the bilinear system. Further the downscaled image is interpolated by using ABSI. The convolution kernel of a 3x3 CF is represented as follows:

3.1. 3x3 kernel clamp filter

The array of a 3x3 kernel for a CF [17] is given by (3.1).

$$\text{kernel}_c = \begin{bmatrix} 1 & 1 & 1 \\ 1 & c & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3.1)$$

where c is a clamp parameter that is set based on the image characteristics. The degree of clamp is adjusted by varying the clamp parameter. The CF gain for a 3x3 kernel can be calculated as shown in equation (3.2).

$$\text{CF} - \text{gain} = C + 8 \quad (3.2)$$

In this algorithm, due to the implementation of the CF by a convolution operation, the filtered values are augmented by filter gain. The convolution result of 3x3 kernel CF is given by (3.3).

$$p'(i,j) = p(i,j) * \begin{bmatrix} 1 & 1 & 1 \\ 1 & c & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3.3)$$

where, $p'(i,j)$ is the filtered pixel value of the original pixel $p(i,j)$. The representation of the original pixel $p(i,j)$ is given by (3.4).

$$p(i,j) = \begin{bmatrix} p(i-1,j-1) & p(i,j-1) & p(i+!,j-1) \\ p(i-1,j) & p(i,j) & p(i+!,j) \\ p(i-1,j+1) & p(i,j+!) & p(i+1,j+!) \end{bmatrix} \quad (3.4)$$

The average intensity of the scaled image is placed near to the original value by dividing the filter results by filter gain. It is given by (3.5).

$$p'(i,j) = \frac{p(i,j) * \begin{bmatrix} 1 & 1 & 1 \\ 1 & c & 1 \\ 1 & 1 & 1 \end{bmatrix}}{c+8} \quad (3.5)$$

For each given pixel P(i, j), the convolution result for a 3x3 kernel CF is determined as per (3.6).

$$p'(i,j) = \frac{\begin{matrix} p(i-1,j-1)+p(i,j-1)+p(i+1,j-1)+p(i-1,j) \\ +C*p(i,j)+p(i+1,j+1)+p(i,j+1)+p(i+!,j+!) \end{matrix}}{C+8} \quad (3.6)$$

where p(i, j) is the convolution result of pixel p(i, j) for a 3x3 kernel CF.

4. EXPERIMENTAL RESULTS

This section presents the performance evaluation and comparison of MBLI and ABSI. The performance of scaling algorithm is evaluated based on two categories such as

quantitative and performance measures. Quantitative measure specifies the quality of the algorithm based on the measurements such as SSIM and PSNR. The performance measure specifies the computational complexity of image interpolation algorithms. Here quantitative and performance measure of the MBLI and the ABSI method are evaluated by using MATLAB simulation and MATLAB Simulink with Xilinx system generator respectively.

4.1. Quantitative measures

Quantitative measure is the figure of merit used for the evaluation of image processing technique. The two different qualitative analysis measurements are used for the selected images to analyze the quality of the scaling algorithm. In order to get these measures, the input image is first down sampled by the factor of 0.5 and up sampled by the factor of 2.

The mean square error (MSE) is used to quantify a noisy approximation of the mxn refined and the original images. MSE is defined in (4.1).

$$\mathrm{MSE} = \frac{1}{\mathrm{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [p(i,j) - p'(i,j)]^2 \quad (4.1)$$

where, M and N are the width and height of the original image respectively. The scaled image quality is denoted in dB with PSNR. [18] PSNR is measured as defined in (4.1),

$$\mathrm{PSNR} = \frac{10\log_{10}\mathrm{MAX}^2}{\mathrm{MSE}} \quad (4.2)$$

The maximum value of each pixel is 255, if the pixels consist of eight bits per sample. Hence, the quality in dB with PSNR is defined in (4.3) as,

$$\mathrm{PSNR} = \frac{10\log_{10}255^2}{\mathrm{MSE}} \quad (4.3)$$

PSNR is commonly used to assess the quality of scaled images. Higher value of PSNR indicates better quality of scaled image. The comparison of the proposed algorithm with others is done by measuring SSIM. SSIM value between two images, x and y can be calculated using (4.4).

$$\mathrm{SSIM} = \frac{(2\mu_x\mu_y + C1)(2\sigma_{xy} + c2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)} \quad (4.4)$$

where μ_x and μ_y refers to the mean of the input and output images that can be defined by,

$$\mu_x = \frac{\mathrm{input(K1,K2)}}{\mathrm{N1 \times N2}} \quad (4.5)$$

$$\mu_y = \frac{output(K1,K2)}{N1 \times N2} \quad (4.6)$$

In equation 4.5, σ_x and σ_y denote the variance of input and output images respectively. σ_{xy} denotes the covariance of the input and output images, which is computed in equation (4.7).

$$\sigma_x = \frac{(\mathrm{input(K1,K2)} - \mu_x)^2}{\mathrm{N1 \times N2}}$$
$$\sigma_y = \frac{(\mathrm{output(K1,K2)} - \mu_y)^2}{\mathrm{N1 \times N2}}$$
$$\sigma_{xy} = \frac{(\mathrm{input(K1,K2)} - \mu_x)(\mathrm{output(K1,K2)} - \mu_y)}{\mathrm{N1 \times N2}} \quad (4.7)$$

where, N1 and N2 are the size of the input and output images, K1 and K2 represent the each pixel value in the input and output images respectively. $\sigma_{xy,}$ σ_x^2 and σ_y^2 denote the mean, variance and covariance of x and y respectively and c1 and c2 are the constants. The qualities of low-complexity scaling algorithms can be evaluated by MATLAB tool that develops PSNR and SSIM values by the original and refined images. For testing the quality of the image scaling algorithms, the Kodak images with the size of 768x512 pixels are collected from the Kodak database. Each test image with the size of 768x512 is scaled down using bilinear system. The downscaled image is refined to the size of 768x512 by MBLI and ABSI, and the scaling algorithm of this work is ABSI with CF. Figure.1 shows the scaled down and refined images by MBLI and ABSI. The experimental results show that the proposed ABSI with CF achieve better quantitative quality than MBLI and ABSI method.

Table A1 lists the PSNR results and the table A2 lists the SSIM results of 24 Kodak images.

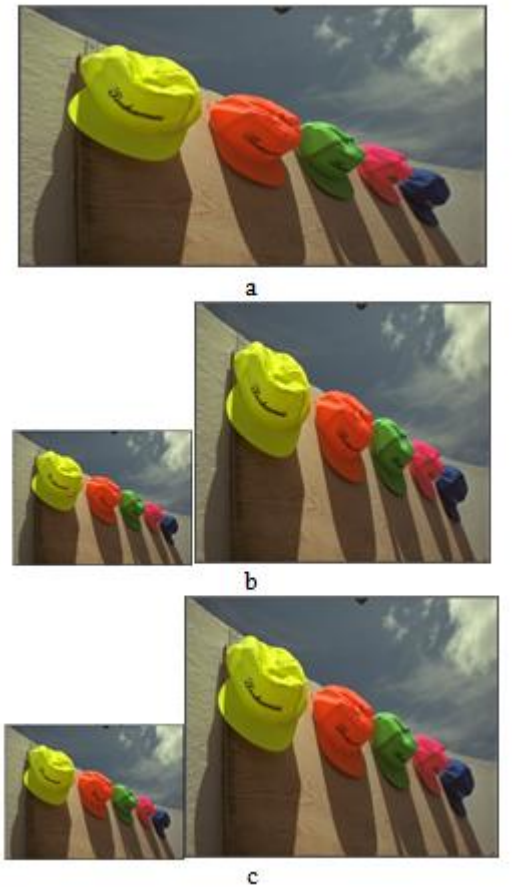

Figure 1.(a) Original tested image (b) down-scaled and up-scaled images using MBLI (c) down-scaled and up-scaled images using ABSI with CF.

Table A1 shows the quantitative analysis measurements such as PSNR and SSIM of various Kodak images for varying scaling ratios using both MBLI and ABSI algorithm. The average PSNR value of the ABSI interpolation algorithm (19.8451dB) is close to the average PSNR value of the MBLI interpolation algorithm (19.9330dB) and its average SSIM value of ABSI interpolation algorithm (0.8113) is also close to MBLI interpolation algorithm (0.8154). In order to increase the quality of the image, ABSI interpolation algorithm is performed with clamp filters using 3x3 Kernel. Figure 2 shows the resulted images of CF based ABSI interpolation algorithm. At first the original image in figure 2(a) is filtered by using CF which is shown in figure 2(b) and filtered original image is downscaled by a factor of 0.5 which is shown in figure 2(c). This downscaled image is again filtered by using CF as in figure 3(d) and it is up-scaled by a factor of 2 that is in figure 3(e) using ABSI interpolation algorithm.

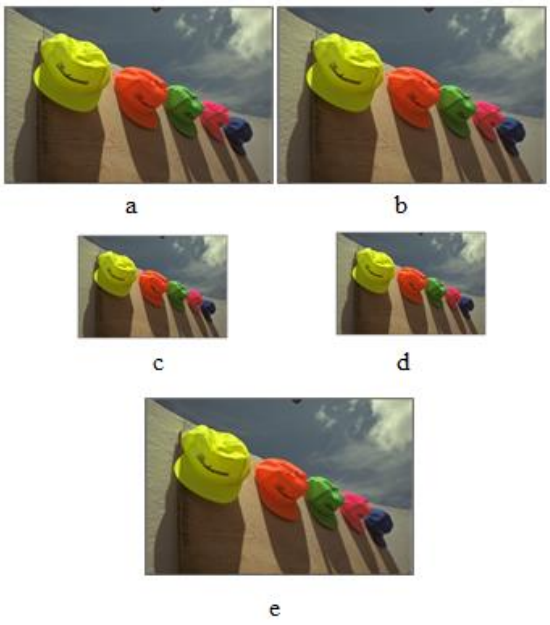

Figure 2.(a) Original image (b) Filtered original image (c) Downscaled image by 0.5 (d) Filtered downscaled image (e) Up-scaled image

Table A2 shows the quantitative measurements of PSNR (dB) and SSIM values obtained on 2x enlarged images of various Kodak images for ABSI interpolation algorithm with clamp filters using 3x3 kernel. As shown in table A2, the average PSNR value of the ABSI interpolation algorithm with CF (20.7022dB) is higher than other filter-based ABSI interpolation algorithm and its average SSIM value of ABSI interpolation algorithm with CF (0.8394) is higher than other filter-based ABSI interpolation algorithm.

4.2. Analysis on area dependent parameters

Image visual quality is not the only issue to be considered when choosing the interpolation approach; the computational complexity should also be taken into account, especially in real time applications. Performance measure calculates the computational and memory requirements of the scaling algorithms. It includes overall device utilization summary, power report, timing report and total memory usage of image scaling circuits.

The architecture performance is simulated, synthesized and implemented by using MATLAB Simulink and Xilinx ISE 14.5. Various device utilization factors are estimated by synthesizing the existing and proposed algorithms. Table A3 shows the performance comparison of different device utilization

factors of MBLI, ABSI and ABSI interpolation with CF. It shows the number of LUTs, slice registers, Input Output Blocks (IOBs) and time and memory requirements used in MBLI, and ABSI with and without filters.

5. CONCLUSION AND RECOMMENDATIONS

In this paper, a filter based linear interpolation technique is presented to improve the quality of interpolated image. A better quality of interpolation is achieved by using clamp filter-based linear interpolation. ABSI with CF has obtained a PSNR value of 20.7022dB and an SSIM value of 0.8394dB. ABSI with CF has obtained a 0.8571dB improvement in PSNR when compared with ABSI without filter. Further, the work is to be improved by using adaptive edge-based linear interpolation. Also the complexity is to be reduced by using low complexity kernel of clamp filters.

REFERENCES

[1] C.John Moses and D.Selvathi, VLSI Architecture of an Area Efficient Image Interpolation, International Journal of Engineering and Technology, Vol. 6, No. 2, 2014, pp. 1121-1130.

[2] Chung-Chi Lin, Chishyan Liaw and Ching-Tsomg Tsai, A Piecewise Linear Convolution Interpolation with Third-order Approximation for Real-time Image Processing, IEEE International Conference on Systems Man and Cybernetics, Turkey, 2010, pp. 3632-3637, http://dx.doi.org/10.1109/ICSMC.2010.5641886.

[3] Shih-Lun Chen, VLSI Implementation of an Adaptive Edge Enhanced Image Scalar for Real Time Multimedia Applications, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 23, No. 9, 2013, pp. 1510-1522, http://dx.doi.org/10.1109/TCSVT.2013.2248492.

[4] Chung-chi Lin, Ming-hwa Sheu, Huann-keng Chiang, Chishyan Liaw and Zeng-chuan Wu, The Efficient VLSI Design of BI-CUBIC Convolution Interpolation for Digital Image Processing, IEEE International Symposium on Circuits and Systems, USA, 2008, pp. 480-483, http://dx.doi.org/10.1109/ISCAS.2008.4541459.

[5] S.R.Juliet Anesha, A Review of Face Recognition Techniques, Journal of Excellence in Computer Science and Engineering, Vol. 2, No. 2, 2016, pp. 10-17, http://dx.doi.org/10.18831/djcse.in/2016021002.

[6] Chung-Hsun Huang and Chao-Yang Chang, An Area and Power Efficient Adder-Based Stepwise Linear Interpolation for Digital Signal Processing, IEEE Transactions on Consumer Electronics, Vol. 62, No. 1, 2016, pp. 69-75, http://dx.doi.org/10.1109/TCE.2016.7448565.

[7] E.Meijering, K.J.Zuiderveld and M.A.Viergever, Image Reconstruction by Convolution with Symmetrical Piecewise n^{th}-order Polynomial Kernels, IEEE Transaction on Image Processing, Vol. 8, No. 2, 1999, pp. 192–201, http://dx.doi.org/10.1109/83.743854.

[8] T.S.Kiran, A Framework in Shadow Detection and Compensation of Images, DJ Journal of Advances in Electronics and Communication Engineering, Vol. 2, No. 3, 2016, pp. 1-9, http://dx.doi.org/10.18831/djece.org/2016031001.

[9] Joohyeok Kim and Jechang Jeong, A New Adaptive Linear Interpolation Algorithm Using Pattern Weight Based on Inverse Gradient, First International Conference on Advances in Multimedia, France, 2009, pp. 58-61, http://dx.doi.org/10.1109/MMEDIA.2009.18.

[10] M.R.Resmi and E.Arun, Foreground Segmentation for Live Videos by Texture Features Journal of Excellence in Computer Science and Engineering, Vol. 2, No. 2, 2016, pp. 1-9, http://dx.doi.org/10.18831/djcse.in/2016021001.

[11] V.Caselles, J.M.Morel and C.Sbert, An Axiomatic Approach to Image Interpolation, IEEE Transactions on Image Processing, Vol. 7, No. 3, 1998,

pp. 376–386, http://dx.doi.org/10.1109/83.661188.

[12] K.Jensen and D.Anastassiou, Subpixel Edge Localization and the Interpolation of Still Images, IEEE Transactions on Image Processing, Vol. 4, No. 3, 1995, pp. 285-295, http://dx.doi.org/10.1109/83.366477.

[13] W.Y.V.Leung, P.J.Bones and R.G.Lane, Statistical Interpolation of Sampled Images, Optical Engineering, Vol. 40, No. 4, 2001, pp. 547–553, http://dx.doi.org/10.1117/1.1353799.

[14] S.Hemaa Abd Alameer, Nabaa Jihad, and Wijdan Abdul Ameer Hussan, Image Interpolation Using High Quality Measure Project Report, College of Science, Ministry of Higher Education and Scientific Research, Baghdad University, 2012.

[15] C.Lin, M.Sheu, H.Chiang, Z.Wu, J.Tu, and C.Chen, A Low-Cost VLSI Design of Extended Linear Interpolation for Real Time Digital Image Processing, International Conference on Embedded Software and Systems, 2008, pp. 196-202, http://dx.doi.org/10.1109/ICESS.2008.85.

[16] S.L.Chen, H.Y.Huang and C.H.Luo, A Low-Cost High-Quality Adaptive Scalar for Real-Time Multimedia Applications, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 21, No. 11, 2011, pp. 1600-1611, http://dx.doi.org/10.1109/TCSVT.2011.2129790.

[17] Shih-Lun Chen, VLSI Implementation of an Adaptive Edge-Enhanced Image Scalar for Real-Time Multimedia Applications, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 23. No. 9, 2013, http://dx.doi.org/10.1109/TCSVT.2013.2248492.

[18] Shih-Lun Chen, VLSI Implementation of a Low cost High quality Image Scaling Processer, IEEE Transactions on Circuit and System II: Express Briefs, Vol. 60, No. 1, 2013, pp. 31-35, http://dx.doi.org/10.1109/TCSII.2012.2234873.

APPENDIX

Table A1.Comparison of PSNR and SSIM value for the 24 Kodak images of MBLI ABSI interpolation algorithms

Images	MBLI algorithm [6]		ABSI algorithm [6]	
	PSNR (dB)	SSIM	PSNR(dB)	SSIM
Kodim01	17.4822	0.5877	17.4509	0.5880
Kodim02	23.8030	0.7502	23.8501	0.7534
Kodim03	24.0986	0.9323	24.1286	0.9324
Kodim04	21.8350	0.8689	21.8772	0.8703
Kodim05	16.6114	0.6731	16.5161	0.6660
Kodim06	19.6749	0.8905	19.0946	0.8760
Kodim07	19.6054	0.7620	19.0689	0.7315
Kodim08	13.6937	0.6136	14.0703	0.6392
Kodim09	19.5752	0.7593	20.0014	0.7790
Kodim10	21.4903	0.8381	21.4677	0.8367
Kodim11	20.0166	0.8141	19.9561	0.8111
Kodim12	21.5069	0.8867	21.4353	0.8848
Kodim13	16.7424	0.7368	16.2692	0.7091
Kodim14	19.1656	0.8258	18.8055	0.8111
Kodim15	19.6690	0.9462	20.2838	0.9532
Kodim16	23.6348	0.9205	23.0242	0.9090
Kodim17	21.1103	0.8911	21.2671	0.8947
Kodim18	19.1370	0.7021	19.0128	0.6939
Kodim19	18.5644	0.7923	18.2538	0.7781
Kodim20	20.0921	0.9585	19.8489	0.9561
Kodim21	19.2153	0.7900	18.6465	0.7624
Kodim22	20.9163	0.8610	21.2625	0.8710
Kodim23	22.4567	0.9537	22.5366	0.9368
Kodim24	18.2970	0.8331	18.1565	0.8279
Average	19.9330	0.8154	19.8451	0.8113

Table A2.Comparison of PSNR and SSIM for 24 Kodak images filter based ABSI algorithm

Images	ABSI with CF (Proposed)		ABSI with sharp filter		ABSI with combined filter	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM	PSNR (dB)	SSIM
Kodim01	18.5142	0.6508	15.0792	0.4483	17.7327	0.6040
Kodim02	24.6224	0.7849	22.0638	0.6701	24.0187	0.7609
Kodim03	24.8757	0.9423	22.2881	0.9008	24.3202	0.9352
Kodim04	22.5970	0.8879	20.4439	0.8279	21.9798	0.8731
Kodim05	17.3290	0.7032	14.7686	0.5733	16.6891	0.6756
Kodim06	20.0592	0.8982	16.6189	0.7991	19.3643	0.8825
Kodim07	20.0660	0.7767	17.4057	0.6446	19.1841	0.7377
Kodim08	14.9503	0.6876	12.4153	0.5388	14.2512	0.6502
Kodim09	21.1906	0.8216	18.3492	0.7039	20.3804	0.7931
Kodim10	22.3577	0.8634	19.7001	0.7709	21.6872	0.8442
Kodim11	20.7989	0.8391	18.0333	0.7336	20.1561	0.8183
Kodim12	22.1927	0.9015	19.7736	0.8394	21.5981	0.8888
Kodim13	17.1803	0.7516	14.1901	0.5968	16.5211	0.7211
Kodim14	19.6972	0.8405	16.9056	0.7345	18.9875	0.8178
Kodim15	20.8486	0.9586	19.1717	0.9402	20.3487	0.9539
Kodim16	23.9110	0.9245	20.5415	0.8496	23.2717	0.9136
Kodim17	22.0243	0.9099	19.7068	0.8557	21.3907	0.8975
Kodim18	19.8855	0.7354	17.1812	0.5946	19.2219	0.7046

Kodim19	19.2668	0.8175	16.2040	0.6813	18.5072	0.7887
Kodim20	20.5918	0.9628	18.2856	0.9382	20.0306	0.9579
Kodim21	19.5364	0.7978	16.6942	0.6716	18.8382	0.7703
Kodim22	22.0536	0.8903	19.5129	0.8178	21.4580	0.8762
Kodim23	23.2552	0.9458	21.0430	0.9131	22.7078	0.9392
Kodim24	19.0493	0.8546	16.4094	0.7595	18.4062	0.8360
Average	20.7022	0.8394	18.0327	0.7418	20.0438	0.81834

Table A3.Comparison of FPGA parameters of MBLI, ABSI and CF based ABSI interpolation algorithm

Parameters	MBLI [6]	ABSI [6]	ABSI with CF (proposed)
Number of LUTs	46	40	242
Number of LUTs & FFpairs	55	47	345
Number of CLB/ Slices	15	14	102
Number of SR	24	24	191
Number of IOBs	33	121	121
Combinational path delay (ns)	0.798	0.798	0.798
CPU time (s)	16.66	11.43	14.07
Elapsed time (s)	17.00	11.00	14.00
Memory usage (MB)	1040	1044	1044
Power (W)	0.267	0.266	0.267