IJCCCE Vol. 16, No.1, 2016

Supporting Zooming-in Process for Image Compression Based on High-Order Weighted 3D Polynomials Fitting

Riyadh Jabbar Soudani

Computer Engineering Department, University of Technology

e-mail: dr_rjs2007@yahoo.com

Received: 5/8/2015

Accepted: 29/2/2016

Abstract – This paper presents a proposed technique to compress images using weighted 3D polynomials fitting technique that fits all pixels as possible in the image. This technique uses high-order weighted 3D polynomials to obtain high quality compressed images especially in the medical images. These types of images seek for high details with an acceptable compression ratio. This procedure of weighted 3D polynomials fitting ensures to preserve the quality of image during the decompression and zooming-in process. After applying scalar quantization and Huffman encoding to the weighted polynomials coefficients for each block of image; mean square error (MSE), peak signal to noise ratio (PSNR), processing time, and compression ratio (CR) are evaluated for different degree of weighted polynomials and for different medical image block sizes. Computer results showed that the proposed technique gives an acceptable image quality under zooming-in process compared with standard surface fitting that uses non-weighted polynomials but at the expanse of compression ratio.

.

Keywords: – Digital image processing, Image compression, Polynomial fitting,Image zooming.

1. Introduction

Compression of visual information became very important for efficient use of media storage and for fast data transfer. Grayscale images require 8-bit for each pixel, while color images require 24-bit (three channels). Furthermore, the size of uncompressed image requires high bandwidth with low transfer rate. That is why the compression process plays an important role in dealing with multi media. The techniques of image compression are either lossless or lossy process.

In lossless compression algorithms, the original image is recovered from the compressed image without losses any information. They use statistical methods to minimize the redundancy [1]. Most of applications that need accurate requirements such as medical imaging use lossless compression techniques. These techniques include run length encoding [2], Huffman encoding [3], Lempel–Ziv–Welch (LZW) coding [4], and area coding [5]. Lossy compression algorithms give higher compression ratios than lossless algorithms. They are widely used in most applications when the quality of images is not the mean issue. These techniques include transformation coding (such as discrete Fourier transform (DFT), discrete cosine transform (DCT), and discrete wavelet transform (DWT)) [6], vector quantization, fractal coding, block truncation coding (BTC), and subband coding [7].

Other compression techniques are Interpolation [8] and surface fitting. Surface fitting method uses single non-weighted polynomial that fits the values of blocks pixels and transmits only the coefficients of the polynomial for each block [9], [10].

The compression process in general attempts to decrease the size of uncompressed image while maintaining the quality of reconstructed image.

Many parameters are used to test the performance of image compression process such as mean squared error (MSE), peak signal to noise ratio (PSNR), and compression ratio (CR). For an image of size *mxn*, the MSE is the cumulative squared error between uncompressed (original) image $f(i,j)$ and the reconstructed image $g(i,j)$ where $i = 1,2 \dots m$ and $j = 1,2 \dots n$ and defined by [11]:

$$MSE = \frac{1}{mxn}\sum_{i=1}^{m}\sum_{j=1}^{n}|f(i,j) - g(i,j)|^2 \qquad (1)$$

PSNR is also used as a measure of quality of reconstructed image and defined as:

$$PSNR\ (dB) = 10\ log_{10}\left[\frac{[2^b-1]^2}{MSE}\right] = 20\ log_{10}\left[\frac{2^b-1}{\sqrt{MSE}}\right] \qquad (2)$$

Where *b* is pixel depth in bits. Low MSE and high PSNR means better compression scheme.

Additional parameter is compression ratio, which is defined as:

$$compression\ ratio(CR) = \frac{uncompressed\ image}{compressed\ image} \qquad (3)$$

In this paper, a proposed technique that uses weighted 3D polynomial fitting technique to compress a color medical image to support zooming-in process. Instead of using low order non-weighted polynomial, which is used in the traditional image surface fitting, high order weighted polynomials will be used to maintain zooming-in details and preserve the quality of medical image during the decompression process.

2. Non-weighted 3D polynomial fitting

In dealing with image processing, it is considered that image information in 2D matrix form, and be constructed of *m*rows

and *n* columns. Each element of this matrix represents the pixel value. In this technique, the image is divided to non-overlapping square blocks, and the following non-weighted polynomial applies to each block [12].

$$Poly_k(x,y) = p_0 + p_1x + p_2y + \cdots + p_{(2k-1)}x^k + p_{2k}y^k \quad (4)$$

Where $p_0, p_1, \ldots, p_{2k}$ are coefficients of non-weighted polynomial and k is the order of polynomial. There is no combination terms of x and y in (4), that is why it called non-weighted polynomial. The coefficients are calculated such that the MSE is reduced for each block. A simplified application of first order polynomial fitting was used in [9].

After extracting the coefficients, the quantization and coding process (such as Huffman coding) are applied to minimize the bits-representation of coefficients. High-order polynomial means low MSE and gives good image quality. However, in the same time gives low compression ratio due to increasing in the number of coefficients.

This paper will test the following non-weighted polynomials for comparison with the proposed technique.

$$Poly1(x,y) = p_0 + p_1x + p_2y \quad (5)$$

$$Poly2(x,y) = p_0 + p_1x + p_2y + p_3x^2 + p_4y^2 \quad (6)$$

$$Poly3(x,y) = p_0 + p_1x + p_2y + \cdots + p_5x^3 + p_6y^3 \quad (7)$$

These polynomials will apply separately to the same medical image in Figure1 (high-details image of a marrow cavity where hematopoietic stem cells are found inside human body, size 256x512) for different block sizes (4x4, 8x8, and 16x16) as in the following steps:

a) Splitting the medical color image into three channels (red, green, and blue).
b) For each channel, the matrix is divided into 4x4 blocks.
c) For each block, first-order polynomial (5) is applied and the three coefficients are calculated according to the least squares fitting [12].
d) These coefficients (except p_0) are quantized using uniform quantizer which has 2^5levels (which means each coefficient will take 5 bits). The step size (Δ) between any two-quantization levels is evaluated according to the following equation [13]:

$$\Delta = \frac{|Max\ Coefficient - Min\ Coefficient|}{2^5} \quad (8)$$

e) For additional minimization of the bit-representation of coefficients, Huffman coding [13] is applied to encode the coefficients values according to the probability of occurrence of the coefficient.

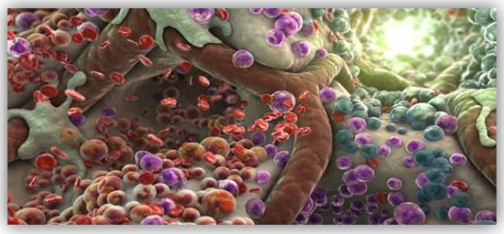


Figure1 Original medical image(high details stem cells).

The same steps are applied for each block of sizes 8x8 and 16x16, and for second-order and third-order polynomials (6) and (7). For decompression process, each block is constructed (16 points for case 4x4-block) from its own coefficients and the final image is reconstructed from all blocks. The compression time is evaluated for each case and the parameters (1-3) are calculated using Matlab (R2014a) program (on Intel i7-core 3.6 GHz, Windows7 64 bits OS with 32 GB RAM).

For 4x4-block size, Figure2 shows model calculations of coefficients and the resulted fitting pixels for randomly selected block.

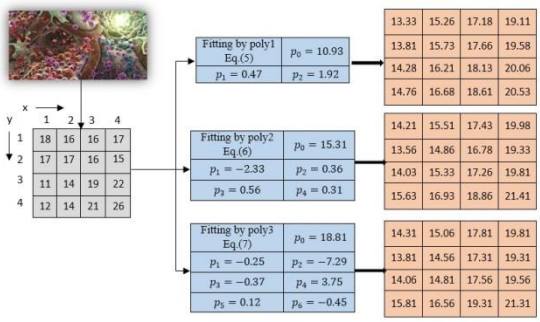


Figure 2Non-weighted Polynomial fitting and pixels values extraction for 4x4-block size.

The following sample calculations show how to preserve the upper-left pixel in the three blocks (Figure2). The original pixel has the value 18 with indices (x=1 and y=1). Applying the non-weighted polynomials (5-7) yields:

- for poly1; $10.93 + 0.47(1) + 1.92(1) = 13.3$,
- for poly2; $15.31 - 2.33(1) + 0.36(1) + 0.56(1)^2 + 0.31(1)^2 = 14.2$,
- for poly3; $18.81 - 0.25(1) - 7.29(1) - 0.37(1)^2 + 3.75(1)^2 + 0.12(1)^3 - 0.45(1)^3 = 14.3$.

The model of three non-weighted Polynomials and the related Matlab functions can be shown in Appendix A. The results in Table1 show the applications of these three types of non-weighted polynomials fitting for different block sizes.

It seems when the order of non-weighted polynomial is increased, the quality of image improved slightly but at the expanse of compression ratio. Zooming-in details in this traditional fitting (non-weighted polynomials) will be corrupted due to missing the weighted coefficients that enhance the fitting process.

Table 1 non-weighted polynomials fitting for high details image

Non-weighted Polynomial	Block Size	MSE	PSNR (dB)	CR	Compression Time per Blk (sec)	Reconstructed Image
Poly1 3 coefficients	4x4	151.25	26.33	7.11	0.033	
	8x8	401.54	22.09	28.44	0.042	
	16x16	770.34	19.26	113.77	0.093	
Poly2 5 coefficients	4x4	98.62	28.19	4.57	0.036	
	8x8	307.99	23.24	18.28	0.049	
	16x16	637.28	20.08	73.14	0.112	
Poly3 7 coefficients	4x4	85.73	28.79	3.36	0.047	
	8x8	272.37	23.77	13.47	0.058	
	16x16	578.74	20.50	53.89	0.122	

Figure 3 shows this drawback for the best case (7 coefficients polynomial with 4x4-block size) and for two zooming-in factors 3x and 6x.

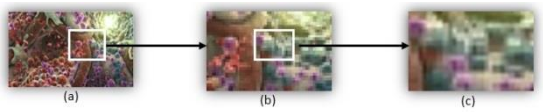


Figure 3Corrupted details during zooming-in process: (a) Reconstructed image, (b) Zooming-in 3x, (c) Zooming-in 6x

Figure 4 shows the behavior of non-weighted polynomials for randomly selected 4x4-block of the original medical image.

18	16	16	17
17	17	16	15
11	14	19	22
12	14	21	26

(a)

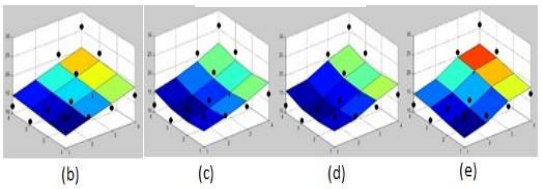


Figure 4Surface fitting of non-weighted polynomials: (a) randomly selected 4x4-block, (b-e) fitting byPoly1~Poly4.

Black points in the Figure 4 represent the original pixels (without fitting).Clearly, the fitting process does not cover all original pixels even if high-order non-weighted polynomials are used.

Table 2Samples of coefficients values for non-weighted polynomials fitting

Polynomial	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
Poly1	10.93	0.47	1.92						
Poly2	15.31	-2.33	0.36	0.56	0.31				
Poly3	18.81	-0.25	-7.29	-0.37	3.75	0.12	-0.45		
Poly4	12.53	3.12	-7.51	-2.18	-8.77	0.62	2.50	-0.06	-0.25

Table 2 shows the values of coefficients for Poly1 to Poly4 for the same block values in Figure4. For high-order polynomials, the values of high-order coefficients have less affected and hence the fitting process does not sound appropriate to preserve the correct pixels values as shown obviously in Figure 4 where surfaces (d) and (e) have not been improved. As a result, most of image details will be lost during the decompression process and under zooming-in process.

3. Weighted 3D Polynomial Fitting (the Proposed Technique)

The fitting process using traditional polynomials (non-weighted) seems does not preserve the correct pixels values after decompression process even though high-order degree is used. Hence, the image details will be lost or degraded under zooming-in manner. To obtain smooth fitting process and cover most of pixels in correct 3D polynomial, the combination of x and y terms must be involved in fitting polynomial. These terms will give high non-linearity behavior that increases the turning points in the polynomial. The 3D weighted polynomial that will be used in proposed technique is defined as:

$$Poly_{kk}(x,y) = p_{00} + p_{10}x + p_{01}y + \cdots + p_{k0}x^k + p_{(k-1)1}x^{k-1}y + \cdots + p_{1(k-1)}xy^{k-1} + p_{0k}y^k \quad (9)$$

Where $k > 1$ and p_{ij} is the coefficient of $x^i y^j$ term. The coefficients p_{ij} when $1 \le i,j < k$ are called weighted coefficients and the related $x^i y^j$ called weighted terms. These weighted terms will play a dominant factor in fitting improvement process due to smooth non-linearity behavior in the 3D polynomial.

Three models of 3D weighted polynomials will be used in the fitting process as follows:

$$Poly_{22}(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 \quad (10)$$

$$Poly_{32}(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 + p_{03}y^3 \quad (11)$$

$$Poly_{44}(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + \cdots + p_{31}x^3y + p_{22}x^2y^2 + p_{13}xy^3 + p_{04}y^4 \quad (12)$$

The flow chart of compression process for the proposed technique can be shown in Figure 5.

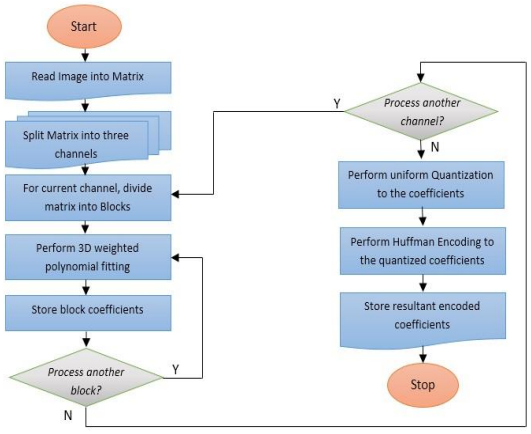


Figure 5 Compression process for the proposed technique

The effect of choice of quantization levels will control the image quality and CR. More quantization levels means better quality but low CR, while less quantization levels means high CR but at

the expanse of image quality.

In Figure 2, it is clear that the first coefficient (p_0) in all polynomials has high priority than the rest of coefficients (this is also true for weighted polynomial coefficients). Therefore, the coefficient $p_0$will stay as 8 bits, while others coefficients (which have smaller values) will be rounded and quantized by using 5-bits.

To increase CR further, Huffman encoding algorithm will be applied to the quantized coefficients. This process will not disturb the image quality; it just minimizes the bit representation of the quantized coefficients. The Huffman encoding is an optimum coding in the sense that no other uniquely decodable set of code words has a smaller average code-word length for a given source. The Huffman encoding algorithm [13] works as follows:

a) The quantized coefficients are set in the descending order.
b) The quantized coefficients of least probabilities are regarded as being combined into a new symbol with probability equal to the sum of the two original probabilities. The probability of the new symbol is placed in the list in accordance with its value.
c) The procedure is repeated until the final list of symbols. Symbol of only two for which a '0' and '1' are assigned.
d) The code for each symbol is found by working backward and tracing the sequence of 0s and 1s assigned to that symbol as well as its successor.

Figure 6 shows the flow chart of the decompression process for proposed technique.

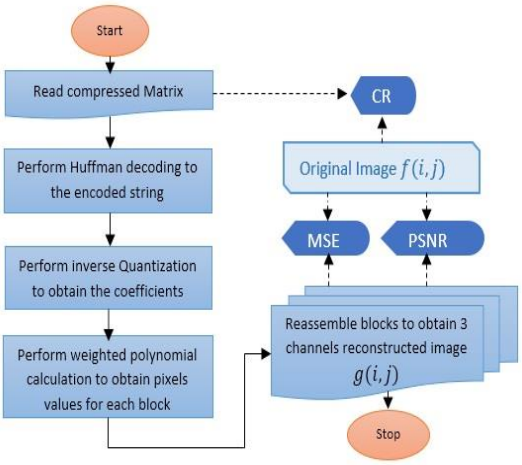


Figure 6Decompression process for the proposed technique.

4. Results

Matlab (R2014a) implementations have been carried out for the medical color image and for different block sizes. The results are compared with non-weighted polynomial fitting technique based on MSE, PSNR, CR, and zooming-in process. The model of weighted 3D polynomials fitting (10-12) will use the following Matlab function:

$$poly3Dfit = fit([x,y],z,'polykk') \quad (13)$$

Where x and y are 2D indices of the block, z is the pixels values of the block, and k is the degree of the weighted polynomial that will fit the block pixels.

Table 3 shows the results of the proposed technique (weighted 3D polynomials fitting) for the following parameters:

- Medical color Image 512x256 (uncompressed size=512 * 256*3*8=3145728 bits).
- Non-overlapped 4x4, 8x8, and 16x16 block sizes.
- 5-bit uniform quantization for weighted coefficients p_{ij}has been performed using Matlab function *imquantize()*, while coefficient p_{00}

in proposed technique will stay as 8 bits.

- The Huffman encoding and decoding have been performed using the functions *mat2huff ( )* and *huff2mat ( )* which have been described in [11].

Table 3 Results of weighted 3D polynomials fitting (proposed technique)

weighted 3D Polynomial	Block Size	Comp-ressed size (bits)	MSE	PSNR (dB)	CR	Compression Time per Blk (sec)	Reconstructed Image
Poly22 (6 coefficients)	4x4	712704	61.55	30.23	4.41	0.015	
	8x8	178176	250.48	24.14	17.65	0.029	
	16x16	44544	564.38	20.61	70.62	0.099	
Poly33 (10 coefficients)	4x4	1204224	21.23	34.86	2.61	0.016	
	8x8	301056	163.25	26.00	10.44	0.031	
	16x16	75264	428.89	21.80	41.79	0.088	
Poly44 (15 coefficients)	8x8	454656	108.80	27.76	6.91	0.030	
	16x16	113664	334.04	22.89	27.67	0.089	

The calculation of processing time for a block compression has been performed by using the MATLAB commands (tic-toc). As shown in Table 3, for high quality image (PSNR=34.86 dB), Poly33 (4x4 block size) is the best choice, but for good CR (10.44) Poly33 (8x8 block size) is an alternative good choice. For Poly44, the case of 4x4 block size is not practical to use due to CR will be around unity (16 pixels with 15 coefficients).

Figure 7 shows the behavior of weighted 3D polynomials fitting (10-12) for randomly selected 8x8-block size of the original image where black points denote the original pixels of the block.

Clearly, the fitting process of Poly33 cover most of original pixels due to using the weighted polynomials that generate more turning points in fitting process and hence is better than fitting process of non-weighted polynomialsas in Figure4.

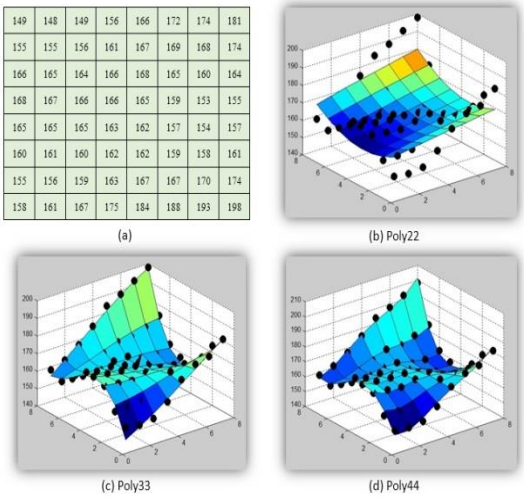


Figure 7Surface fitting of weighted 3D polynomials: (a) randomly selected 8x8-block, (b-d) fitting byPoly22~Poly44

Additionally, preserving image details with this proposed technique gives the process of zooming-in a clear details and smooth depth (comparing with poor zooming-in for non-weighted polynomials fitting in Figure3) as shown in Figure.8 for the case Poly33 fitting and 4x4-block size and for two zooming-in factors 3x and 6x.

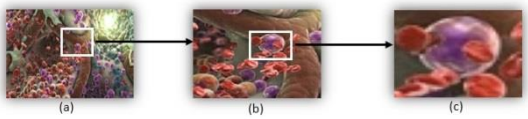
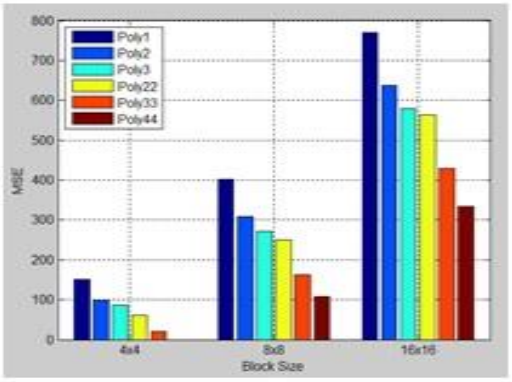
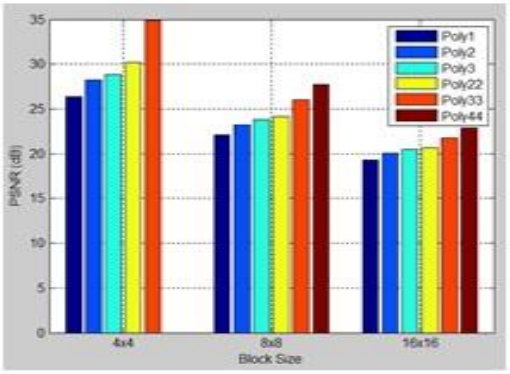


Figure 8Clear details (stem cells) during zooming-in process by proposed technique: (a) reconstructed image, (b) zooming-in 3x, (c) zooming-in 6x.

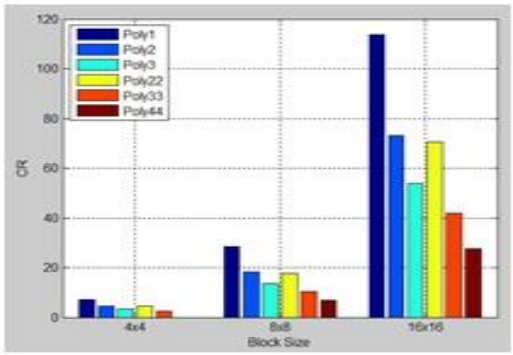
Figure 9 shows the image parameters comparison between non-weighted 3D polynomial fitting (Poly1, Poly2, and Poly3) and the proposed technique (weighted 3D polynomial fitting Poly22, Poly33, and Poly44) based on MSE, PSNR, and CR.



(a)



(b)



(c)

Figure 9Image parameters comparison between non-weighted polynomial fitting and proposed technique based on: (a) mean squared error, (b) peak signal to noise ratio, (c) compression ratio.

Another useful comparison is for zooming-in process between non-weighted 3D polynomial fitting and the proposed technique to show how the quality could be preserved under two passes of zooming-in process. Table4 shows this comparison for 3x (300%) and 6x (600%) zooming-in factors.

Table 4Image qualitycomparison under 3x and 6x zooming-in factors.

Fitting Method	Image quality, PSNR (dB)		
	Reconstructed Image	3x zooming-in	6x zooming-in
Non-weighted 3D polynomial *best case: Poly3, 4x4 block size*	28.79 Figure 3 (a)	17.27 Figure 3 (b)	10.08 Figure 3 (c)
Weighted 3D polynomial *best case: Poly33, 4x4 block size* (proposed technique)	34.86 Figure 8 (a)	31.83 Figure 8 (b)	24.16 Figure 8 (c)

5. Conclusion

In this paper, a weighted 3D polynomial fitting is proposed for image compression that support zooming-in process. Integrating with scalar quantizer and Huffman encoding offered a smooth fitting and clear depth details due to using weighted 3D polynomial which increases the turning points during the block fitting. This enhancement in depth of details can be seen visibly when comparing Figure 3 and Figure 8 under two passes of zooming-in depth. Additionally, Matlab results showed that the proposed technique is better than non- weighted 3D polynomial fitting for preserving the image details as shown in Figure 9 when comparing MSE and PSNR. For the best case (Poly33 and 8x8 block size), MSE=163.25 and PSNR= 26 dB while for traditional fitting (Poly3 and 8x8 block size), MSE=272.37 and PSNR=23.77 dB. In addition, the case of Poly33 with 4x4 block size, has also provided a superior degree of details (PSNR= 34.86 dB) but at expanse of compression ratio. In all cases, the proposed technique (weighted 3D polynomial fitting) maintains high details under compression and zooming-in process.

Appendix

The model of three non-weighted Polynomials and the related Matlab functions can be shown in Table A1.

Table A1 Models of Matlab functions

Non-Weighted Polynomial Type	MATLAB Function
$Poly1(x,y) = p_0 + p_1x + p_2y$	*P=fittype (@ (a, b, c, x, y) a+b*x+c*y, 'independent', {'x', 'y'} ,'dependent', 'z')* *poly3Dfit = fit([x,y], z, P,'StartPoint',[1,1,1])*
$Poly2(x,y) = p_0 + p_1x + p_2y + p_3x^2 + p_4y^2$	*P=fittype (@ (a, b, c, d, e, x, y) a+b*x+c*y+d*x.^2 +e*y.^2, 'independent',{'x', 'y'}, 'dependent', 'z')* *poly3Dfit = fit([x,y], z, P,'StartPoint',[1,1,1,1,1])*
$Poly3(x,y) = p_0 + p_1x + p_2y + p_3x^2 + p_4y^2 + p_5x^2 + p_6y^2$	*P=fittype (@ (a, b, c, d, e, f, g, x, y) a+b*x+c*y+d*x.^2 +e*y.^2+f*x^3 +g*y.^3, 'independent', {'x', 'y'} ,'dependent', 'z')* *poly3Dfit = fit([x,y], z, P,'StartPoint',[1,1,1,1,1,1,1])*

References

[1] M. Yang, N.Bourbakis, "An Overview of Lossless Digital Image Compression Techniques" Circuits & Systems, 2005 48th Midwest Symposium, Vol. 2, IEEE, pp. 1099-1102, Aug 2005.W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] D.Chakraborty, S. Banerjee, "Efficient Lossless Colour Image Compression Using Run Length Encoding and Special Character Replacement", International Journal on Computer Science and Engineering, Vol. 3 No. 7, pp. 2719-2725, July 2011.

[3] M. Sharma, "Compression Using Huffman Coding", IJCSNS International Journal of Computer Science and Network Security, Vol. 10, No.5, pp. 133-141, May 2010.

[4] S. kaur, V. Sulochana, "Design and Implementation of LZW Data Compression Algorithm", International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.4, pp.71-81, July 2012.

[5] D. Raj, "Modeling of Image Compression and Decompression using Huffman Code Technique", International Journal of Computer Science and Mobile Computing, Vol.2, Issue. 12, pp. 440-447, Dec. 2013.

[6] H. B. Kekre, T.Sarode, P.Natu, "Image Compression Using Real Fourier Transform, Its Wavelet Transform And Hybrid Wavelet With DCT", International Journal of Advanced Computer Sciences and Applications, Vol. 4, Issue. 5, pp. 41-47, Science and Information Society (SAI), Iraqi Virtual Science Library, 2013.

[7] A. Singh, M.Gahlawat, "Image Compression and its Various Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 6, pp.650-654, June 2013.

[8] X. Zhang, X. Wu, "Image Interpolation by Adaptive 2-D Autoregressive Modeling and Soft-Decision Estimation", IEEE transactions on image processing, vol. 17, Issue. 6, pp. 887-896, June 2008.

[9] A. M. Butt, R. A.Sattar, "On Image Compression Using Curve Fitting", Master Thesis, Computer Science, School of Computing, Blekinge Institute of Technology, Thesis No. MCS-2010-35,Sweden 2010.

[10] Y.S. Chen, H.T. Yen, W.H. Hsu, "Compression of Color Image via the Technique of Surface Fitting", Graphical Models and Image Processing, Vol. 56, Issue 3, pp. 272–279, 1994.

[11] R.C. Gonzalez, "Digital image processing using Matlab", Pearson Prentice Hill, 2004, ch.8.

[12] J.H.Mathews, K.D.Fink, "Numerical methods using Matlab", fourth edition, Pearson prentice hall, 2004, ch.4.

[13] J.S.Chitode," Information Coding Techniques", 1st edition, 2008, ch.4.