3D Res (2015) 6:26
DOI 10.1007/s13319-015-0055-6

3DR EXPRESS

A Novel 2D Image Compression Algorithm Based on Two Levels DWT and DCT Transforms with Enhanced Minimize-Matrix-Size Algorithm for High Resolution Structured Light 3D Surface Reconstruction

M. M. Siddeq · M. A. Rodrigues

Received: 9 April 2015 / Revised: 23 May 2015 / Accepted: 25 May 2015 / Published online: 7 July 2015

Abstract Image compression techniques are widely used on 2D image 2D video 3D images and 3D video. There are many types of compression techniques and among the most popular are JPEG and JPEG2000. In this research, we introduce a new compression method based on applying a two level discrete cosine transform (DCT) and a two level discrete wavelet transform (DWT) in connection with novel compression steps for high-resolution images. The proposed image compression algorithm consists of four steps. (1) Transform an image by a two level DWT followed by a DCT to produce two matrices: DC- and AC-Matrix, or low and high frequency matrix, respectively, (2) apply a second level DCT on the DC-Matrix to generate two arrays, namely nonzero-array and zero-array, (3) apply the Minimize-Matrix-Size algorithm to the AC-Matrix and to the other high-frequencies generated by the second level DWT, (4) apply arithmetic coding to the output of previous steps. A novel decompression algorithm, Fast-Match-Search algorithm (FMS), is used to reconstruct all high-frequency matrices. The FMS-algorithm computes all compressed data probabilities by using a table of data, and then using a binary search algorithm for finding decompressed data inside the table. Thereafter, all decoded DC-values with the decoded AC-coefficients are combined in one matrix followed by inverse two levels DCT with two levels DWT. The technique is tested by compression and reconstruction of 3D surface patches. Additionally, this technique is compared with JPEG and JPEG2000 algorithm through 2D and 3D root-mean-square-error following reconstruction. The results demonstrate that the proposed compression method has better visual properties than JPEG and JPEG2000 and is able to more accurately reconstruct surface patches in 3D.

Keywords DWT · DCT · Minimize-Matrix-Size algorithm · FMS algorithm · 3D reconstruction

M. M. Siddeq (✉) · M. A. Rodrigues
GMPR-Geometric Modelling and Pattern Recognition Research Group, Sheffield Hallam University, Sheffield, UK
e-mail: mamadmmx76@yahoo.com

M. A. Rodrigues
e-mail: M.Rodrigues@shu.ac.uk

1 Introduction

Compression methods are being rapidly developed for large data files such as images, where data compression in multimedia applications has lately become more important. While it is true that the price of storage has steadily decreased, the amount of generated image and video data has increased exponentially. It is more evident on large data repositories such as YouTube and cloud storage offered by a number of suppliers. With the increasing growth of network traffic and storage requirements, more efficient methods are needed for compressing image and video data,

while retaining high quality with significant reduction in storage size. The discrete cosine transform (DCT) has been extensively used [3, 25] in image compression. The image is divided into segments and each segment is then subject of the transform, creating a series of frequency components that correspond with detail levels of the image. Several forms of coding are applied in order to store only coefficients that are found as significant. Such a way is used in the popular JPEG file format, and most video compression methods and multi-media applications are generally based on it [5, 11].

A step beyond JPEG is JPEG2000 that is based on the discrete wavelet transform (DWT) which is one of the mathematical tools for hierarchically decomposing functions. Image compression using wavelet transforms is a powerful method that is preferred by scientists to get the compressed images at higher compression ratios with higher PSNR values [17, 18]. Its superiority in achieving high compression ratio, error resilience, and other features promotes it to become the tomorrow's compression standard and led to the JPEG2000 ISO. As referred to the JPEG abbreviation, which stands for Joint Photographic Expert Group, JPEG2000 codec is more efficient than its predecessor JPEG and overcomes its drawbacks [6]. It also offers higher flexibility compared to even many other codec such as region of interest, high dynamic range of intensity values, multi component, lossy and lossless compression, efficient computation, and compression rate control. The robustness of JPEG2000 stems from its utilization of DWT for encoding image data. DWT exhibits high effectiveness in image compression due to its support to multi-resolution representation in both spatial and frequency domains. In addition, DWT supports progressive image transmission and region of interest coding [1, 7].

Furthermore, a requirement is introduced concerning the compression of 3D data. We demonstrated that while geometry and connectivity of a 3D mesh can be tackled by a number of techniques such as high degree polynomial interpolation [16] or partial differential equations [13, 15], the issue of efficient compression of 2D images both for 3D reconstruction and texture mapping for structured light 3D applications has not been addressed. Moreover, in many applications, it is necessary to transmit 3D models over the Internet to share CAD/CAM models with e-commerce customers, to update content for entertainment applications, or to support collaborative design, analysis, and display of engineering, medical, and scientific datasets. Bandwidth imposes hard limits on the amount of data transmission and, together with storage costs, limit the complexity of the 3D models that can be transmitted over the Internet and other networked environments. It is envisaged that surface patches can be compressed as a 2D image together with 3D calibration parameters, transmitted over a network and

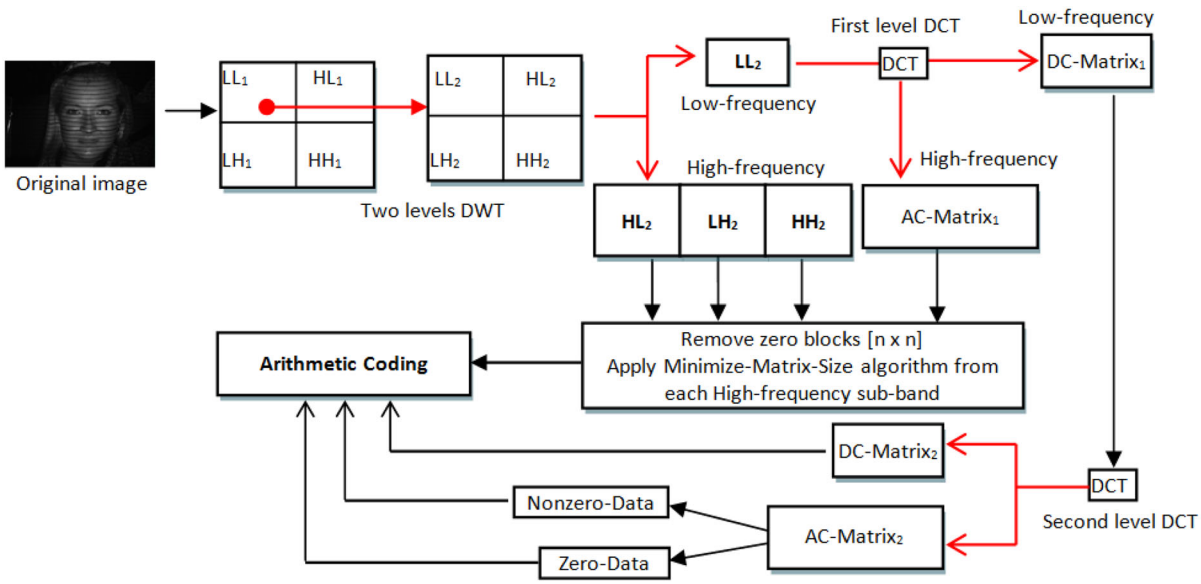


Fig. 1 Layout proposed two levels DWT–DCT compression technique

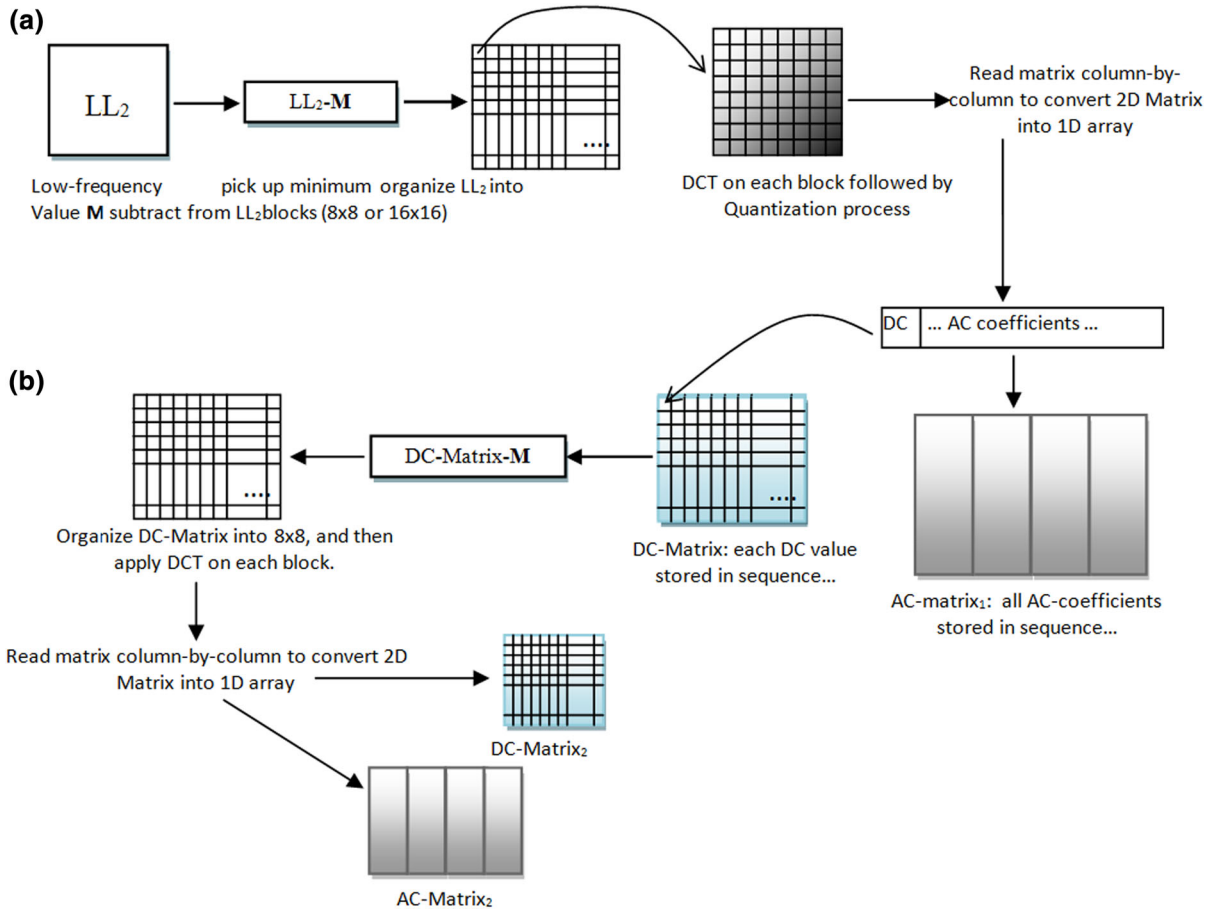


Fig. 2 Two levels DCT applied to LL_2. **a** First level DCT applied to LL_2. **b** Second level DCT applied to LL_2

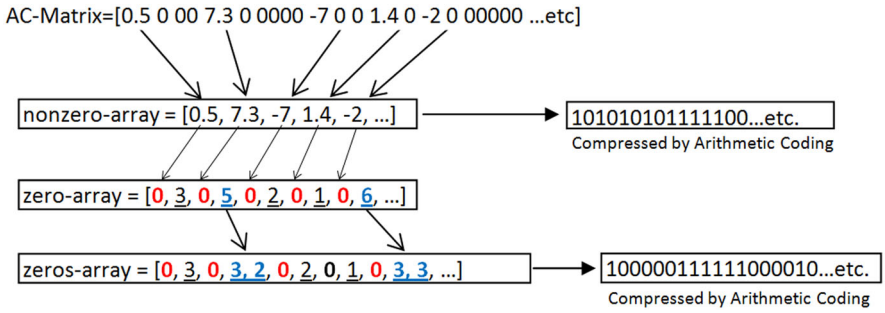


Fig. 3 Separate AC-Matrix into zero-array and nonzero-array

remotely reconstructed (geometry, connectivity and texture map) at the receiving end with the same resolution as the original data [12, 22]. Siddeq and Rodrigues [22] they proposed 2D image compression used in 3D application based on DWT and DCT. DWT linked with DCT for produce series of high-frequency matrices, the same transformation applied again on the low-frequency matrix to produce another series of high-frequencies data array. Finally the series of data are coded by Minimize-Matrix-Size algorithm (MMS), the coded data then decoded by Limited-Sequential Search algorithm (LSS) for reconstruct 2D images. The advantage for this method can get high resolution image at higher compression ratio up to 98 %. However, the complexity for this algorithm made execution time for compression and decompression

within few minutes. For this research we describe a new method for image compression based on two separate transformations; two levels DWT and the low-frequency matrix addressed to two levels DCT, leading to an increased number of high-frequency matrices, which are then shrunk using the Enhanced MMS (EMMS) algorithm. This paper demonstrates that our compression algorithm achieved efficient image compression ratio up to 99.5 % and superior accurate 3D reconstruction compared with standard JPEG and JPEG2000.

2 The Proposed 2D Image Compression Algorithm

This section presents a novel lossy compression algorithm implemented via DWT and DCT. The algorithm starts with a two level DWT. While all high frequencies (HL_1, LH_1, HH_1) of the first level are discarded, all sub-bands of the second level are further encoded. We then apply DCT to the low-frequency sub-band (LL_2) of the second level; the main reason for using DCT is to split into another low frequency and high-frequency matrices (DC- and AC-$Matrix_1$).- The EMMS algorithm is then applied to compress AC-$Matrix_1$ and high frequency matrices (HL_2, LH_2, HH_2). The DC-$Matrix_1$ is subject to a second DCT whose AC-$Matrix_2$ is quantized then subject to arithmetic coding together with DC-$Matrix_2$ and the output of EMMS algorithm as depicted in Fig. 1.

2.1 Two Level Discrete Wavelet Transform (DWT)

The implementation of the wavelet compression scheme is very similar to that of sub-band coding scheme: the signal is decomposed using filter banks [4, 24]. The output of the filter banks is down-sampled, quantized, and encoded. The decoder decodes the coded representation, up-samples and recomposes the signal. Wavelet transform divides the information of an image into an approximation (i.e., LL) and detail sub-band. The approximation sub-band shows the general trend of pixel values and the other three detail sub-bands show the vertical, horizontal and diagonal details in the images. If these details are very small

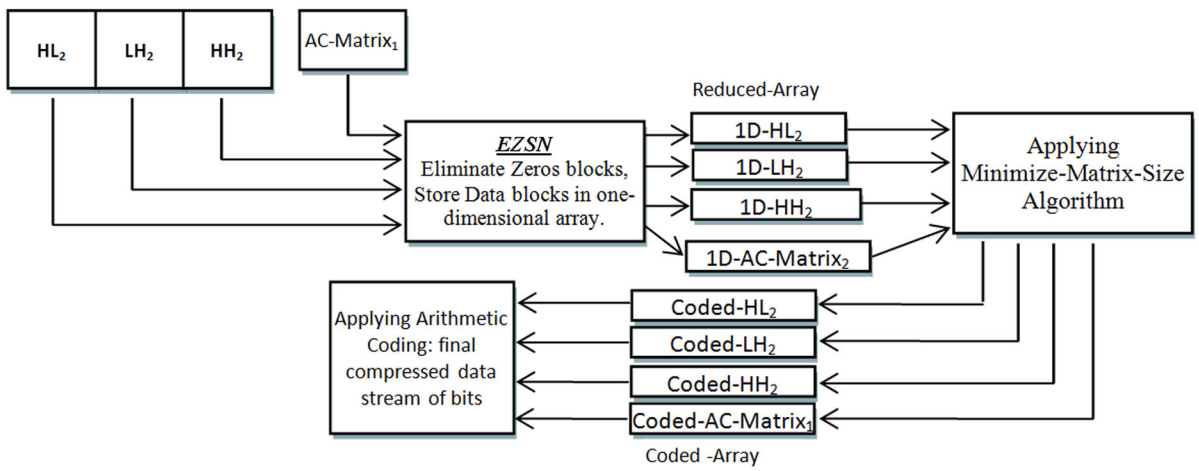


Fig. 4 Layout of the Minimize-Matrix-Size algorithm

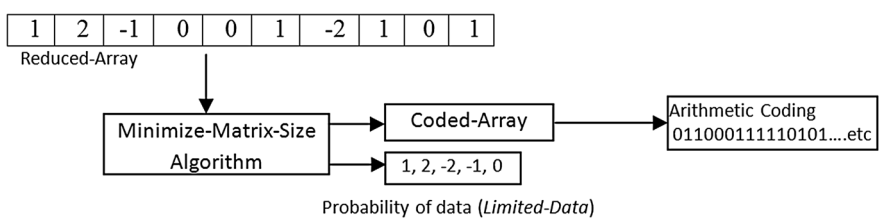


Fig. 5 Illustration of probability data (Limited-Data) for the Reduced-Array

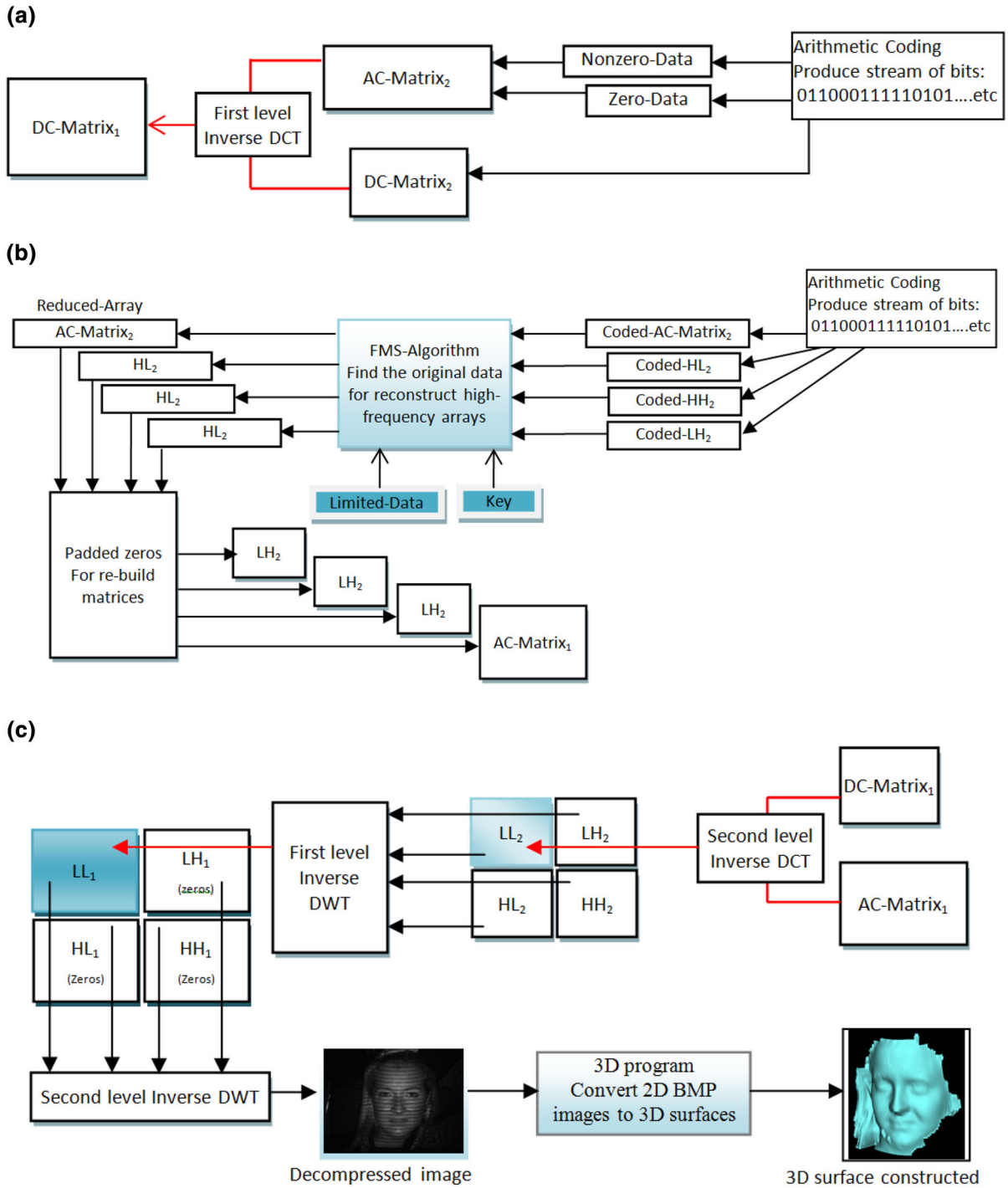


Fig. 6 Layout of the proposed decompression algorithm. **a** First level inverse DCT. **b** FMS-Algorithm applied to reconstruct high-frequency matrices. **c** Two levels inverse DWT with two levels inverse DCT applied to decompress 2D image

then they can be set to zero without significantly changing the image details [20], for this reason the high frequency sub-bands are compressed into fewer bytes. DWT uses filters for decomposing an image; these filters assist to increase the number of zeros in high frequency sub-bands. One common filter used in

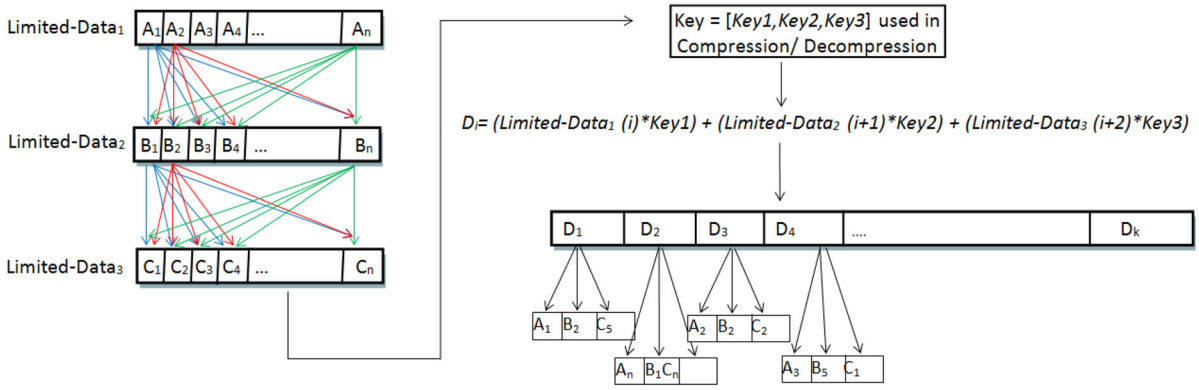


Fig. 7 First stage FMS-algorithm for reconstructing high frequency data from Limited-Data. (*A*–*C*) Uncompressed data which are determined by the unique combination of keys

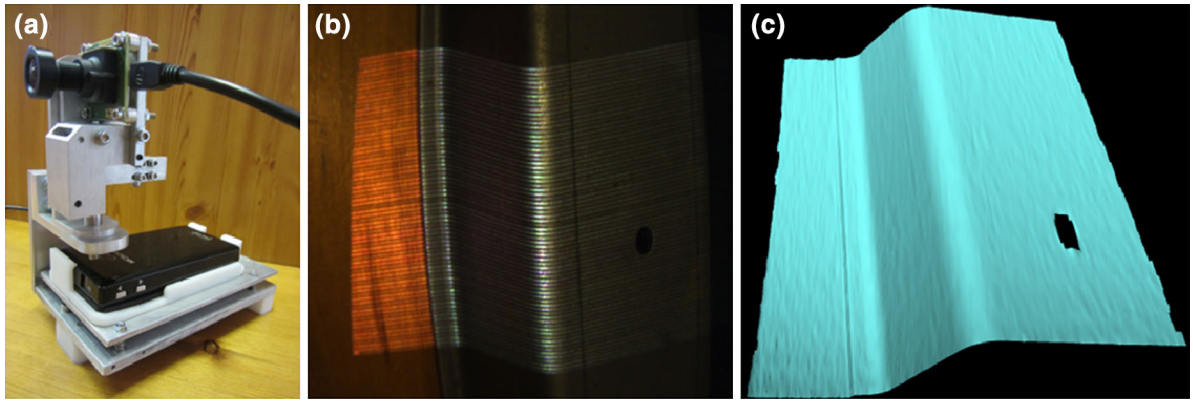


Fig. 8 **a** The 3D scanner developed by the GMPR Group, **b** 2D BMP picture captured by the camera, and **c** 2D image converted into a 3D surface patch

decomposition and composition is called *Daubechies filter*. This filter stores much information about the original image in the "LL" sub-band, while other high-frequency domains contain less significant details, and this is one important property of the Daubechies filter for achieving a high compression ratio. A two-dimensional DWT is implemented on an image twice, first applied on each row, and then applied on each column [11, 18]:

In the proposed research the high frequency sub-bands at first level are set to zero (i.e., discard HL, LH and HH). These sub-bands do not affect image details. Additionally, only a small number of non-zero values are present in these sub-bands. In contrast, high-frequency sub-bands in the second level (HL_2, LH_2 and HH_2) cannot be discarded, as this would significantly affect the image. For this reason, high-frequencies values in this region are quantized. The quantization process in this research depends on the maximum value in each sub-band, as shown by the following equation [23]:

$$H_2 = \frac{H_2}{H_2m}, \tag{1}$$

$$H_2m = \max(H_2) \times Ratio, \tag{2}$$

where "H_2" represents each high-frequency sub-band at second level in DWT (i.e., HL_2, LH_2 and HH_2). While "H_2m" represents maximum value in a sub-band, and the "Ratio" value is used as a control for amount of maximum value, which is used to control image quality. For example, if the maximum value in a sub-band is 60 and $Ratio = 0.1$, the quantization value is $H_2m = 6$, this means all values in a sub-band are divided by 6.

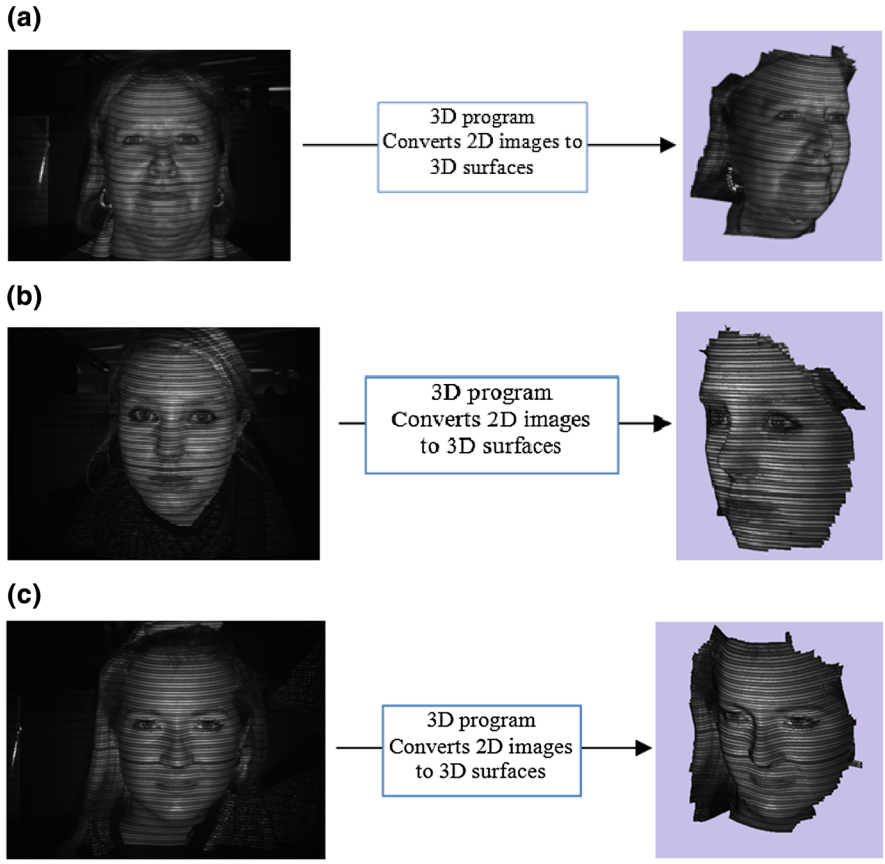


Fig. 9 Original 2D images (*left*) with 3D surface reconstruction (*right*). **a** Original 2D Face1 dimensions 1392 × 1040, Image size: 1.38 Mbytes, converted to 3D surface. **b** Original 2D Face2 dimensions 1392 × 1040, Image size: 1.38 Mbytes, converted to 3D surface. **c** Original 2D Face3 dimensions 1392 × 1040, Image size: 1.38 Mbytes, converted to 3D surface

Each sub-band has different priority for keeping image details. The higher priorities are: LH_2, HL_2 and then HH_2. Most of information about image details are in HL_2 and LH_2. If most of nonzero data in these sub-bands are retained, the image quality will be high even if some information is lost in HH_2. For this reason the *Ratio* value for HL_2, LH_2 and HH_2 is defined in the range $= \{0.1, \ldots, 0.5\}$.

2.2 Two Level Discrete Cosine Transform (DCT)

In this section we describe the two levels DCT applied to low-frequency sub-band "LL_2" (see Fig. 1). A quantization is first applied to LL_2 as follows. All values in LL_2 are subtracted by the minimum of LL_2 and then divided by 2 (i.e., a constant even number). Thereafter, a two-dimensional DCT is applied to produce de-correlated coefficients. Each variable size block (e.g., 8 × 8) in the frequency-domain consists of: the DC-value at the first location, while all the other coefficients are called AC coefficients. The main reason for using DCT to split the final sub-band (LL_2) into two different matrices is to produce a low-frequency matrix (DC-Matrix) and a high-frequency matrix (AC-Matrix). The following two steps are used in the two levels DCT implementation:

Organize LL_2 into 8 × 8 non-overlapping blocks (other sizes can also be used such as 16 × 16), then apply DCT to each block followed by quantization. The following equations represent the DCT and its inverse [2, 18, 25]:

The quantization table is a matrix of the same block size that can be represents as follows:

$$Q(i, j) = Block + (i + j), \tag{3}$$

$$Q(i, j) = Q(i, j) * Scale, \tag{4}$$

where $i, j = 1, 2, \ldots, Block$, $Scale = 1, 2, 3, \ldots, Block$.

After applying the two-dimensional DCT on each 8 × 8 or 16 × 16 block, each block is quantized by

Table 1 Compressed size for 2D image Face1

Applied single level DWT			
Single level DWT High-frequencies (HL, LH and HH)	DCT parameters		Compressed sizes (Kbytes)
	Block sizes	Scales	
Discarded	8 × 8	0.5	51.4
Discarded	8 × 8	1	29.33
Discarded	8 × 8	2	15.6
Discarded	8 × 8	3	10.58
Discarded	8 × 8	4	8
Discarded	8 × 8	5	6.37
Discarded	16 × 16	0.5	28.52
Discarded	16 × 16	1	14.74
Discarded	16 × 16	2	7.38

Applied two levels DWT					
Two levels DWT high-frequencies Ratio value (Eq. (5))			DCT parameters		Compressed sizes (Kbytes)
LH_2	HL_2	HH_2	Block sizes	Scales	
0.3	0.3	0.3	8 × 8	0.5	29.93
0.3	0.3	0.3	8 × 8	1	17.55
0.3	0.3	0.3	8 × 8	2	9.7
0.3	0.3	0.3	8 × 8	3	6.74
0.3	0.3	0.3	16 × 16	0.5	21
0.3	0.3	0.3	16 × 16	1	10.54
0.3	0.3	0.3	16 × 16	2	5.19

the "Q" using dot-division-matrix, which truncates the results. This process removes insignificant coefficients and increases the number of the zeros in the each block. However, in the above Eq. (4), the factor "Scale" is used to increase/decrease the values of "Q". Thus, image details are reduced in case of *Scale* >1. There is no limited range for this factor, because this depends on the DCT coefficients.

Convert each block to 1D array, and then transfer the first value to a new matrix called DC-Matrix. While the rest of data AC coefficients are saved into a new matrix called AC-Matrix. Similarity, the DC-Matrix is transformed again by DCT into another two different matrices: DCT-$Matrix_2$ and AC-$Matrix_2$. Figure 2 illustrates the details of the two levels DCT applied to the sub-band LL_2.

The resulting DC-$Matrix_2$ size is very small and can be represented in a few bytes. On the other hand, the AC-$Matrix_2$ contains lots of zeros with only a few nonzero data. All zero scan be erased and just nonzero data are retained. This process involves separating zeros from nonzero data, as shown in Fig. 3. The zero-array can be computed easily by calculating the number of zeros between two nonzero data. For example, assume the following AC-Matrix = [0.5, 0, 0, 0, 7.3, 0, 0, 0, 0, 0, −7], the zero-array will be [*0*, **3**, *0*, **5**, *0*] where the zeros in italics refer to nonzero data existing at these positions in the original AC-Matrix and the numbers in bold refer to the number of zeros between two consecutive non-zero data. In order to increase the compression ratio, the number "5" in the zero-array can be broken up into "3" and "2" zeros to increase the probability (i.e., re-occurrence) of redundant data. Thus, the new equivalent zero-array would become [*0*, **3**, *0*, **3**, **2**, *0*].

After this procedure, the nonzero-array contains positive and negative data (e.g., nonzero-array = [0.5, 7.3, −7]), and each data size probably

Table 2 Compressed size for 2D image Face2

Applied single level DWT			
Single level DWT High-frequencies (HL, LH and HH)	DCT parameters		Compressed sizes (Kbytes)
	Block sizes	Scales	
Discarded	8 × 8	0.5	47
Discarded	8 × 8	1	26.85
Discarded	8 × 8	2	14.42
Discarded	8 × 8	3	9.91
Discarded	8 × 8	4	7.4
Discarded	16 × 16	0.5	25.33
Discarded	16 × 16	1	13.3
Discarded	16 × 16	2	6.77

Applied two levels DWT					
Two levels DWT High-frequencies Ratio value (Eq. (5))			DCT parameters		Compressed sizes (Kbytes)
LH_2	HL_2	HH_2	Block sizes	Scales	
0.3	0.3	0.3	8 × 8	0.5	35
0.3	0.3	0.3	8 × 8	1	19.34
0.3	0.3	0.3	8 × 8	2	9.9
0.3	0.3	0.3	8 × 8	3	6.41
0.3	0.3	0.3	8 × 8	4	4.66
0.3	0.3	0.3	16 × 16	0.5	26.3
0.3	0.3	0.3	16 × 16	1	12.96
0.3	0.3	0.3	16 × 16	2	5.76

reaches to 32-bits and these data can be compressed by a coding method. In that case the index size (i.e., the header compressed file) is almost 50 % of compressed data size. The index data are used in the decompression stage, therefore, the index data could be broken up into parts for more efficient compression. Each 32-bit data are partitioned into small parts (4-bit), and this process increases the probability of redundant data. Finally these streams of data are subject to arithmetic coding.

2.3 The Minimize-Matrix-Size Algorithm

In this section we introduce a new algorithm to squeeze high-frequency sub-bands into an array, then minimizing that array by using the MMS algorithm, and then compressing the newly minimised array by arithmetic coding. Figure 4 illustrates the steps in this novel high-frequency compression technique.

Each high-frequency sub-band contains lots of zeros with a few nonzero data. We propose a technique to eliminate block of zeros, and store blocks of nonzero data in an array. This technique is useful for squeezing all high-frequency sub-bands, this process is labelled *eliminate zeros and store nonzero data* (*EZSN*) in Fig. 4, applied to each high frequency independently. The EZSN algorithm starts to partition the high-frequency sub-bands into non-overlapped blocks [K × K], and then search for nonzero blocks (i.e., search for at least one nonzero inside a block). If a block contains any nonzero data, this block will be stored in the array called *Reduced-Array*, with the position of that block. Otherwise, the block will be ignored, and the algorithm continues to search for other nonzero blocks. The EZSN algorithm is illustrated in *List-1* below.

```
List-1:EZSN Algorithm
K=8;  %% block size = [KxK]
I=1; LOC=1
while (I< column size for high-frequency sub-band)
J=1;
while (J< row size for high-frequency sub-band)
Block[1..K*K]= Read_Block_from_Matrix(I,J) ;  %%read 8x8 block from high-frequency sub-band
if(Check_Block(Block) = = 'nonzero')  %%check the "Block" content, has it a nonzero?
POSITION [LOC] =I; POSITION [LOC+1] =J; %% Save original location for block contains
                    %%nonzero data in high-frequency sub-band
LOC=LOC+2;
Forn=1: Block_Size* Block_Size
Reduced_Array[P]= Block[n]; %%save nonzero data in new array
++P;
Endfor
Endif
J=J+K;
Endwhile% inner loop
   I=I+K;
Endwhile
```

After each sub-band is squeezed into an array, thereafter, the MMS algorithm is applied to each Reduced Array independently. This method reduces the array size by 1/3, the calculation depends on key values and coefficients of the reduced array, and the result is stored in a new array called Coded-Array. The following equation represents the MMS algorithm [21–23]:

$$\begin{aligned} \textit{Coded Array}\ (P) = {} & \textit{Key}(1) * \textit{RA}(L) + \textit{Key}(2) \\ & * \textit{RA}(L+1) + \textit{Key}(3) \\ & * \textit{RA}(L+2). \end{aligned} \tag{5}$$

Note that "RA" represents Reduced-Array (HL_2, LH_2, HH_2 and AC-$Matrix_1$).

$L = 1, 2, 3, \ldots, N-3$, "N" is the size of Reduced-Array

$P = 1, 2, 3, \ldots, N/3$.

The key values in the above Eq. (5) are generated by a key generator algorithm. Initially compute the maximum value in reduced sub-band, and three keys are generated according to the following steps:

```
M=MAX_VALUE +(MAX_VALUE/2);          %% maximum value selected and divided by "2"
Key[1]=0.1;            %% First Key value <=1
Key[2]=[0.1× M ] ×Factor;             %% Factor=1,2,3,…etc
Key[3]=[ KEY(1)× M+KEY(2)× M ] × Factor;
KEY=[Key[1], Key[2], Key[3]]; %% Final Key values for compression and decompression
```

Each reduced sub-band (see Fig. 4) has its own key value. This depends on the maximum value in each sub-band. The idea for the key is similar to weights used in perceptron neural network: $P = AW_1 + BW_2 + CW_3$, where W_i are the weight values generated randomly and "A", "B" and "C" are data. The output of this summation is "P" and there is only one possible combination for the data values given W_i (more details in Sect. 3). Using a key generator the MMS algorithm illustrated in List-2 [21–23].

```
List -2 : Minimize-Matrix-Size Algorithm
Let Pos=1
MAX_Value=Find_Max_value( Reduced-Arra);
W=Key-Generator(0.1, MAX_Value, 2);
I=1;
while (I< size of Reduced-Array)
    ForK=0 to 2
Coded-Array[Pos]= Coded-Array [Pos] + ( Reduced-Array[I+K] × W[K] );
Endif
Pos++;
I=I+3;
```

Before applying the MMS algorithm, our compression algorithm computes the probability of the Reduced-Array (i.e., compute the probability for each HL_2, LH_2, HH_2 and AC-Matrix$_1$). These probabilities are called *Limited-Data*, which is used later in the decompression stage. The Limited-Data are stored as additional data in the header of the compressed file. Figure 5 illustrates the probability of data for a matrix.

The final step of the compression algorithm is arithmetic coding, which is one of the important methods used in data compression; its takes a stream of data and convert it to a single floating point value. These values lie in the range less than one and greater than zero that, when decoded, return the exact stream of data. The arithmetic coding needs to compute the probability of all data and assign a range for each data (low and high) in order to generate streams of bits [18].

3 The Decompression Algorithm: Fast Matching Search Algorithm (FMS)

The proposed decompression algorithm is the inverse of compression and consists of three stages:

(1) First level inverse DCT to reconstruct the DC-Matrix$_1$;
(2) Apply the FMS-algorithm to decode each sub-band independently (i.e., HL_2, LH_2, HH_2, AC-Matrix$_2$);
(3) Apply the second level inverse DCT with two levels inverse DWT to reconstruct the 2D image.

Once the 2D image is reconstructed, we apply structured light 3D reconstruction algorithms to obtain an approximation of the original 3D surface, from which errors can be computed for the entire surface. Figure 6 shows the layout of the main steps in the proposed decompression algorithm.

The *FMS algorithm* has been designed to compute the original high frequency data. The compressed data contains information about the compression keys and probability data (Limited-Data) followed by streams of compressed high frequency data. Therefore, the FMS algorithm picks up each compressed high frequency data and reads information (key values and Limited-Data) from which the original high frequency data are recovered. The FMS-algorithm is illustrated through the following steps A and B:

(A) Initially, Limited-Data are copied (in memory) three times into separated arrays. This is because expanding the compressed data with the three keys resembles an interconnected data, similar to a network as shown in Fig. 7.
(B) Pick up a data item from "D", the compressed array (i.e., Coded-HL_2 or Coded-LH_2 or Coded-HH_2 or Coded-AC-Matrix$_1$) and search for the combination of (A, B, C) with respective keys that satisfy *D*. Return the triplet decompressed values (A, B, C).

Since the three arrays of Limited-Data contain the same values, that is A1 = B1 = C1, A2 = B2 = C2, and so on the searching algorithm computes all possible combinations of A with Key1, B with Key2 and C with Key3 that yield a result D. As a means of an example consider that Limited-Data$_1$ = [A1 A2 A3], Limited-Data$_2$ = [B1 B2 B3] and Limited-Data$_3$ = [C1 C2 C3]. Then, according to Eq. (5) these represent RA(L), RA(L + 1) and RA(L + 2), respectively, the equation is executed 27 times ($3^3 = 27$) testing all indices and keys. One of these combinations will match the data in (D) (i.e., the original high frequency Coded-LH_2 or Coded-HL_2 or Coded-HH_2 or Coded-AC-Matrix$_1$) as described in Fig. 6b. The match indicates that the unique combination of A, B, C are the original data we are after.

The searching algorithm used in our decompression method is called *binary search algorithm*, the algorithm finds the original data (A, B, C) for any input from array "D". For binary search, the array should be arranged in ascending order. In each step, the algorithm compares the input value with the middle of element of the array "D". If the value matches, then a matching element has been found and its position is returned [9]. Otherwise, if the search is less than the middle element of "D", then the algorithm repeats its action on the sub-array to the left of the middle element or, if the value is greater, on the sub-array to the right. There is no probability for "Not Matched", because the FMS-algorithm computed all compression data possibilities previously.

After the Reduced-Arrays (LH_2, LH_2, HH_2 and AC-Matrix$_1$) are recovered, their full corresponding high frequency matrices are re-build by placing nonzero-data in the exact locations according to EZSN algorithm (see List-1). Then the sub-band LL_2 is reconstructed by combining the DC-Matrix$_1$ and AC-Matrix$_1$ followed by the inverse DCT. Finally, a two level inverse DWT is applied to recover the original 2D BMP image (see Fig. 6c). Once the 2D image is decompressed, its 3D geometric surface is reconstructed such that error analysis can be performed in this dimension.

4 Experimental Results

The results described below use MATLAB-R2013a for performing 2D image compression and decompression, and 3D surface reconstruction was performed with our own software developed within the GMPR Group [12, 13, 15] running on an AMD quad-core microprocessor. The justification for introducing 3D reconstruction is that we can make use of a new set of metrics in terms of error measurements and perceived quality of the 3D visualization to assess the quality of the compression/decompression algorithms. The principle of operation of 3D surface scanning is to project patterns of light onto the target surface whose image is recorded by a camera. The shape of the captured pattern is combined with the spatial relationship between the light source and the camera, to determine the 3D position of the surface along the pattern. The main advantages of the method are speed and accuracy; a surface can be scanned from a single 2D image and processed into 3D surface in a few milliseconds [14]. Figure 8 shows 2D BMP image captured by scanner then converted to 3D surface by our 3D software.

The results in this research are divided into two parts: first, we apply the proposed image compression and decompression methods to 2D grey scale images of human faces. The original and decompressed images are used to generate 3D surface models from which direct comparisons are made in terms of perceived quality of the mesh and objective error measures such as root-mean-square-error (RMSE). Second, we repeat all procedures for 2D compression 2D decompression 3D reconstruction but this time with colour images of objects other than faces. Additionally, the computed 2D and 3D RMSE are used directly for comparison with JPEG and JPEG2000 techniques.

4.1 Compression, Decompression and 3D Reconstruction from Grey Scale Images

As described above the proposed image compression started with DWT. The level of DWT decomposition affects the image quality also the compression ratio, so we divided the results into two parts to show the effects of each independently: single level DWT and two levels DWT. Figure 9 shows the original 2D human faces tested by the proposed algorithm. Tables 1, 2 and 3 show the compressed size by using our algorithm with a single level and two levels DWT for Face1, Face2 and Face3, respectively also Fig. 10, illustrates scale parameter effects on compression data size.

The proposed decompression algorithm (see Sect. 3) applied to the compressed image data recovers the 2D images which are then used by the 3D reconstruction to generate the respective 3D surface. The following figures, Figures 11, 12 and 13 show high-quality, median-quality and low-quality compressed images for; Face1, Face2 and Face3. Also, Tables 4, 5 and 6 show the time execution for the FMS-algorithm.

It is shown through the pictures and tables above that the proposed compression algorithm is successfully applied to grey scale images. In particular, Tables 1, 2 and 3 show a compression of more than 99 % of the original image size compressed and the reconstructed 3D surfaces still preserve most of their quality. Some images are compressed by DCT with block size of 16×16 are also shown capable of generating high quality 3D surface. Also there is not much difference between block sizes of 8×8 and 16×16 for high quality reconstruction images with "Scale = 0.5".

Table 3 Compressed size for 2D image Face2

Applied single level DWT					
Single level DWT High-frequencies (HL, LH and HH)			DCT parameters		Compressed sizes (Kbytes)
			Block sizes	Scales	
Discarded			8 × 8	0.5	49.53
Discarded			8 × 8	1	27.5
Discarded			8 × 8	2	14.31
Discarded			8 × 8	3	9.45
Discarded			8 × 8	4	7.13
Discarded			16 × 16	0.5	26.83
Discarded			16 × 16	1	13.53
Discarded			16 × 16	2	6.52
Applied two levels DWT					
Two levels DWT High-frequencies Ratio value Eq. (5)			DCT parameters		Compressed sizes (Kbytes)
LH_2	HL_2	HH_2	Block sizes	Scales	
0.3	0.3	0.3	8 × 8	0.5	35.78
0.3	0.3	0.3	8 × 8	1	19.6
0.3	0.3	0.3	8 × 8	2	9.63
0.3	0.3	0.3	8 × 8	3	6.2
0.3	0.3	0.3	16 × 16	0.5	25.78
0.3	0.3	0.3	16 × 16	1	12.4
0.3	0.3	0.3	16 × 16	1.7	6.45

4.2 Compression, Decompression and 3D Reconstruction from Colour Images

Colour images contain red, green and blue layers. In JPEG and JPEG2000 colour layers are transformed to "YCbCr" layers before compression. This is because most of information about images available in layer "Y" while other layers "CrCb" contain less information [7, 8]. The proposed image compression tested with YCbCr layers, and then applied on true colour layers (red, green and blue). Figure 14 shows the original colour images tested by our approach.

First the Wall, Room and Corner images as depicted in Fig. 14 were transformed to YCbCr before applying our proposed image compression—both using single level and two level DWT decomposition. Second, our approach was applied on the same colour images but this time using true colour layers. Tables 7, 8 and 9 show the compressed size for the colour images by the proposed compression algorithm: also Fig. 15 shows scale parameter effects on compression data size. Figures 16, 17 and 18 show decompressed colour images (Wall, Room and Corner, respectively) as 3D surface. Additionally, Tables 10, 11 and 12 illustrate the execution time for the FMS-algorithm at single level DWT for the colour images. Similarly, Tables 13, 14 and 15 show the FMS-algorithm time execution for the same colour images by using two levels DWT.

It can be seen from the above Figs. 16, 17 and 18 and Tables 10, 11 and 12 that a single level DWT is applied successfully to the colour images using both YCbCr and RGB layers. Also the two levels DWT (Tables 13, 14 and 15) gives good performance, however, two levels did not perform well on YCbCr layer at higher compression ratios. Both colour images Wall and Room contain green stripe lines; this renders RGB layers more appropriate to be used with the proposed approach. On the other hand, for the image Corner, the YCbCr layer proved more appropriate.

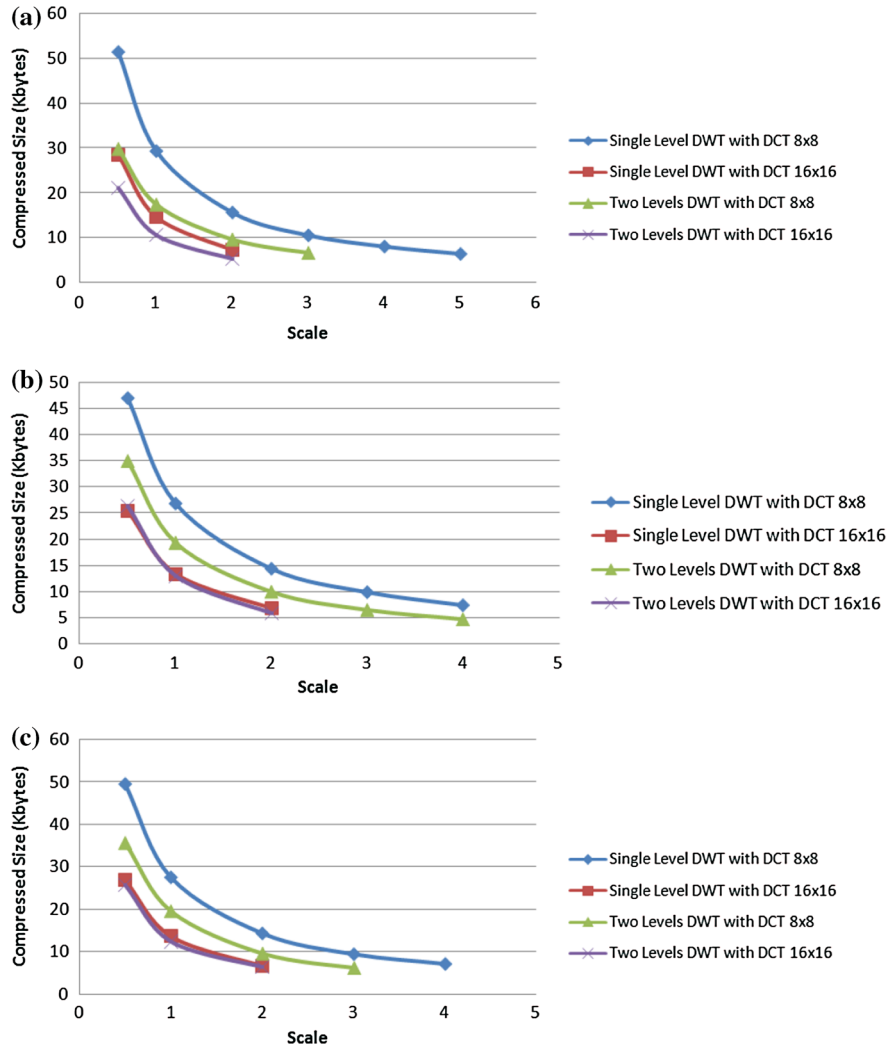


Fig. 10 Illustrates Tables 1, 2 and 3 as chart according to "Scale" and "Compressed Size". **a** This chart illustrates Table 1 for FACE1 image by using different scale value in our approach, as shown in this figure the results by single level DWT and DCT (block size 16 x 16) approximately close to the results obtained by two levels DWT and DCT (block size 8 x 8). **b** This chart illustrates Table 2 for FACE2 image by using different scale value in our approach, as shown in this figure the results by single level DWT and DCT (block size 16 x 16) very close to the results obtained by two levels DWT and DCT (block size 16 x 16). **c** This chart illustrates Table 3 for FACE3 image by using different scale value in our approach, as shown in this figure the results by single level DWT and DCT (block size 16 x 16) approximately equivalent to the results obtained by two levels DWT and DCT (block size 16 x 16)

4.3 Comparison with JPEG2000 and JPEG Compression Techniques

JPEG and JPEG2000 are both used widely in digital image and video compression, also for streaming data over the Internet. In this research we use these two methods for 3D image compression to show quality and compression ratio for comparison with our proposed image compression. The JPEG technique is based on the 2D DCT applied on the partitioned image into 8 × 8 blocks, and then each block encoded by RLE and Huffman encoding [2, 10]. The JPEG2000 is based on the multi-level DWT applied on complete image and then each sub-band quantized and coded by arithmetic encoding [1, 6]. Most image compression applications depend on JPEG and JPEG2000 and allow the user to specify a quality parameter to choose suitable compression ratio. If the image quality is increased the compression ratio is decreased and vice versa [18]. The comparison of these methods with our proposed one is based on for similar compression ratios; we test the image quality by using RMSE. The RMSE is a very popular quality measure, and can be calculated very easily between the decompressed and the original images [18, 19]. We also show visualization 3D surfaces for decompressed 2D images by JPEG and JPEG2000 as a means of comparison. Figures 19–24 shows the 3D reconstructed images respectively by JPEG and JPEG2000.

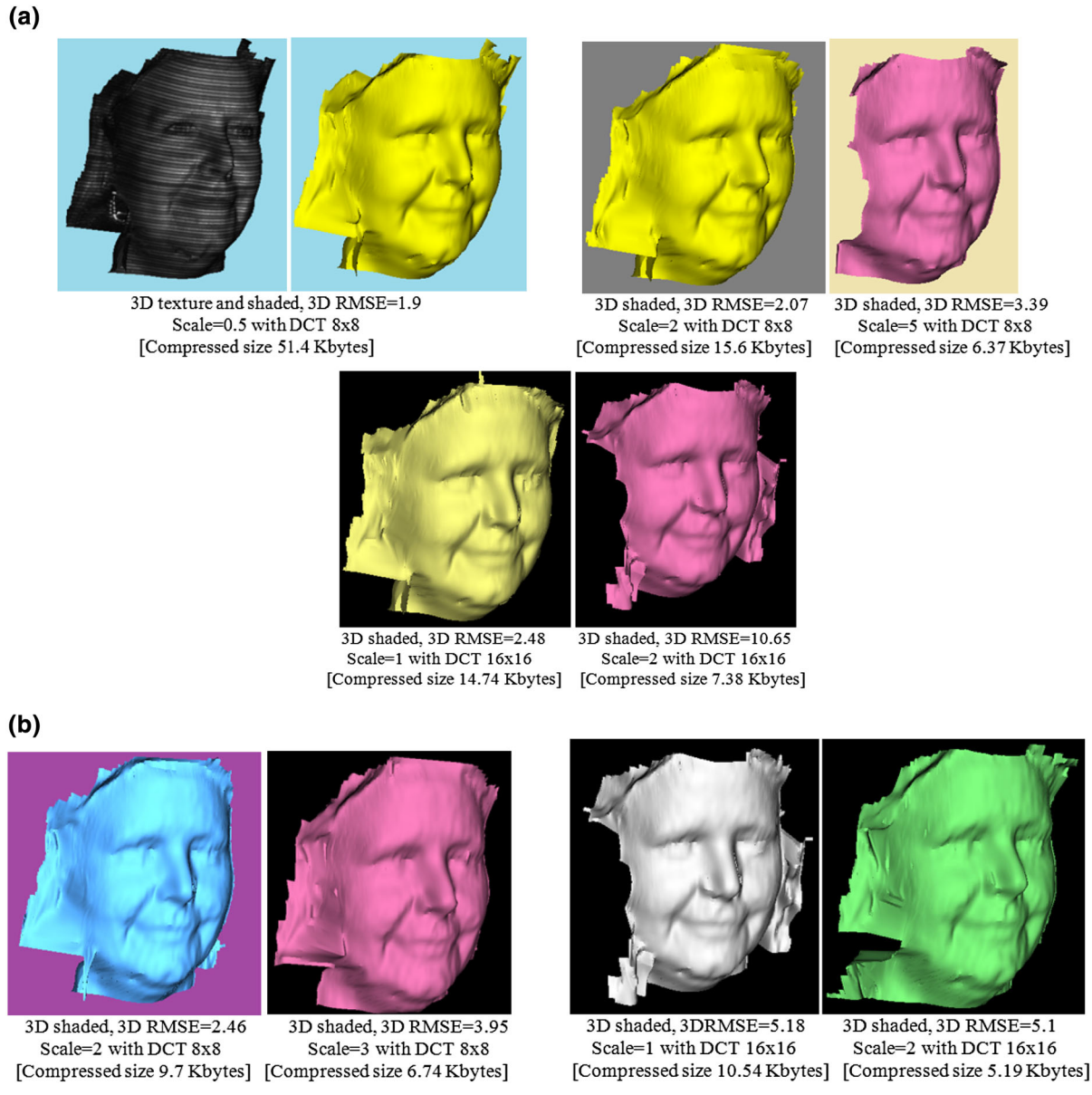


Fig. 11 Decompressed 2D image Face1 by our proposed decompression method, and then converted to a 3D surface. **a** Decompressed 2D BMP images at Single level DWT converted to 3D surface; 3D surface with scale = 0.5 represents high quality image comparable to the original image, and 3D surface with scale = 2 represents median quality image approximately high quality image. Also 3D surface with scale = 5 is low quality image some parts of surface failing to reconstruct. Additionally, using a block size of 16 × 16 DCT further degrades the 3D surface. **b** Decompressed 2D BMP images at two levels DWT converted to 3D surface; 3D surface with scale = 2, 3 and DCT 8 × 8 represent low quality image surface with degradation. Additionally, using a block size of 16 × 16 DCT further degrades the 3D surface

In Tables 16, 17, 18, 19, 20 and 21 "Not Applicable" means the relevant algorithm cannot compress to the required size successfully. Also, the symbol "≈" refers to JPEG2000 and JPEG that can approximately compressed at the required size (i.e., added to compressed size ± 1 Kbyte).

5 Conclusion

This research has presented and demonstrated a novel method for image compression and compared the quality of compression through 3D reconstruction, 3D RMSE and the perceived quality of the 3D

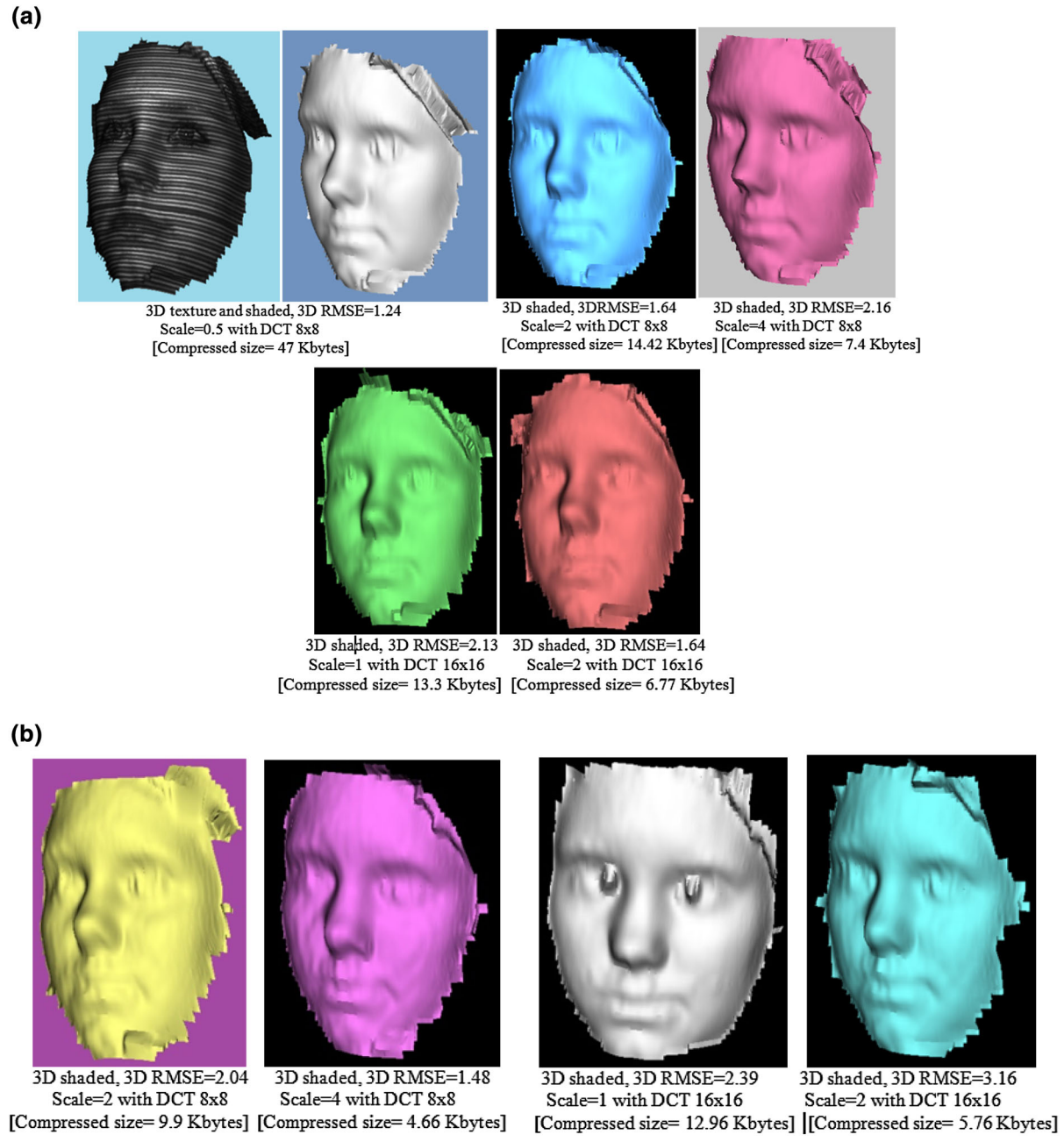


Fig. 12 Decompressed 2D image of Face2 image by our proposed decompression method, and then converted to 3D surface. **a** Decompressed 2D BMP images at Single level DWT converted to 3D surface; 3D surface with scale = 0.5 represents high quality image similar to the original image, and 3D surface with scale = 2 represents a median quality image. Also 3D surface with scale = 4 is low quality with surfaces lightly degraded. Additionally, a block size of 16 × 16 DCT used in our approach degrades some parts of 3D surface. **b** Decompressed 2D BMP images at two levels DWT converted to 3D surface; 3D surface with scale = 2, 4 with DCT 8 × 8 represent low quality image surface with degradation. Similarly, a block size of 16 × 16 DCT used in our approach degrades the 3D surface

visualisation. The method is based on a two levels DWT transform and two levels DCT transform in connection with the proposed MMS algorithm. The results showed that our approach introduced better image quality at higher compression ratios than JPEG and JPEG2000 being capable of accurate 3D

Table 4 FMS time execution for decompressed image Face1

Applied single level DWT with two levels DCT							
DCT		Decompressed RMSE		FMS-algorithm Time execution (s)			
Blocks	Scales	2D RMSE	3D RMSE	AC-Matrix$_1$	LH	HL	HH
8 × 8	0.5	4.42	1.9	4.25	Discarded	Discarded	Discarded
8 × 8	1	4.83	2.31	1.31	Discarded	Discarded	Discarded
8 × 8	2	5.58	2.07	0.23	Discarded	Discarded	Discarded
8 × 8	3	6.21	2.4	0.1	Discarded	Discarded	Discarded
8 × 8	4	6.87	3.34	0.046	Discarded	Discarded	Discarded
8 × 8	5	7.41	3.39	0.046	Discarded	Discarded	Discarded
16 × 16	0.5	6.0	3.96	2.24	Discarded	Discarded	Discarded
16 × 16	1	6.63	2.48	0.23	Discarded	Discarded	Discarded

Applied two levels DWT with two levels DCT										
DCT		Scale used in DCT and second level DWT			Decompressed RMSE		FMS-algorithm Time execution (s)			
Blocks	Scales	LH_2	HL_2	HH_2	2D RMSE	3D RMSE	AC-Matrix$_1$	LH_2	HL_2	HH_2
8 × 8	0.5	0.3	0.3	0.3	6.79	2.19	9.1	0.031	≈0	≈0
8 × 8	1	0.3	0.3	0.3	7.5	4.77	1.75	≈0	0.031	≈0
8 × 8	2	0.3	0.3	0.3	8.1	2.46	0.24	≈0	≈0	≈0
8 × 8	3	0.3	0.3	0.3	8.46	3.95	0.1	≈0	≈0	≈0
16 × 16	0.5	0.3	0.3	0.3	8.1	3.69	2.82	0.031	≈0	≈0
16 × 16	1	0.3	0.3	0.3	8.89	5.18	0.65	≈0	≈0	≈0
16 × 16	2	0.3	0.3	0.3	9.62	5.1	0.18	≈0	≈0	≈0

reconstructing at higher compression ratios. On the other hand, it is more complex than JPEG2000 and JPEG. The most important aspects of the method and their role in providing high quality image with high compression ratios are discussed as follows:

(1) In a two levels DCT, the first level separates the DC-values and AC-values into different matrices; the second level DCT is then applied to the DC-values and this generates two new matrices. The size of the two new matrices are only a few bytes long (because they contain a large number of zeros), this process increases the compression ratio.
(2) Since most of the high-frequency matrices contain lot of zeros as above, in this research we used the EZSN algorithm, to eliminate zeros and keep non-zero data. This process keeps significant information while reducing matrix sizes up to 50 % or more.
(3) The MMS algorithm is used to replace each three coefficients from the high-frequencies matrices by a single floating-point value. This process converts each high-frequency matrix into a one-dimensional array, leading to increased compression ratios while keeping the quality of the high-frequency coefficients.
(4) The FMS-algorithm represents the core of our search algorithm for finding the exact original data from a one-dimensional array (i.e., Reduced-Array) converting to a matrix, and depends on the organized key-values and Limited-Data. According to time execution tables, the FMS-algorithm finds values in a few microseconds, for some high-frequencies needs just few nanoseconds at higher compression ratios.
(5) The key-values and Limited-Data are used in coding and decoding an image, without these information images cannot be reconstructed.
(6) Our proposed image compression algorithm was tested on true colour images (i.e., red, green and blue), obtained higher compression ratios

Table 5 FMS time execution for decompressed image Face2

Applied single level DWT with two levels DCT							
DCT		Decompressed RMSE		FMS-algorithm Time execution (s)			
Blocks	Scales	2D RMSE	3D RMSE	AC-Matrix$_1$	LH	HL	HH
8 × 8	0.5	4.88	1.24	7.06	Discarded	Discarded	Discarded
8 × 8	1	5.22	2.19	1.09	Discarded	Discarded	Discarded
8 × 8	2	5.86	1.64	0.208	Discarded	Discarded	Discarded
8 × 8	3	6.46	2.55	0.109	Discarded	Discarded	Discarded
8 × 8	4	6.96	2.16	0.046	Discarded	Discarded	Discarded
16 × 16	0.5	5.57	1.7	3.4	Discarded	Discarded	Discarded
16 × 16	1	6.09	2.13	0.56	Discarded	Discarded	Discarded
16 × 16	2	6.94	1.63	0.093	Discarded	Discarded	Discarded

Applied two levels DWT with two levels DCT										
DCT		Scale used in DCT and second level DWT			Decompressed RMSE		FMS-algorithm Time execution (s)			
Blocks	Scales	LH_2	HL_2	HH_2	2D RMSE	3D RMSE	AC-Matrix$_1$	LH_2	HL_2	HH_2
8 × 8	0.5	0.3	0.3	0.3	5.16	1.33	13.15	0.046	0.046	0.031
8 × 8	1	0.3	0.3	0.3	5.76	2.07	2.1	0.031	≈0	≈0
8 × 8	2	0.3	0.3	0.3	7.04	2.04	0.35	0.062	≈0	≈0
8 × 8	3	0.3	0.3	0.3	7.91	1.39	0.109	≈0	≈0	≈0
8 × 8	4	0.3	0.3	0.3	8.43	1.48	0.062	≈0	≈0	0.031
16 × 16	0.5	0.3	0.3	0.3	5.73	2.08	2.96	0.093	0.031	0.031
16 × 16	1	0.3	0.3	0.3	6.44	2.39	0.59	0.031	0.031	≈0
16 × 16	2	0.3	0.3	0.3	7.85	3.16	0.124	0.015	≈0	≈0

and high image quality for images containing green striped lines. Additionally, our approach has been tested on YCbCr layers with good quality at higher compression ratios.

(7) Our approach gives better visual image quality compared to JPEG and JPEG2000. This is because our approach removes most of the block artefacts caused by the 8 × 8 two-dimensional DCT. Also our approach uses a single level DWT or two levels DWT rather than multi-level DWT as in JPEG2000; for this reason blurring typical of JPEG2000 is removed in our approach. JPEG and JPEG2000 failed to reconstruct a surface in 3D when compressed to higher ratios while it is demonstrated that our approach can successfully reconstruct the surface and thus, is superior to both on this aspect.

However, there are a larger number of steps in the proposed compression and decompression algorithm than in JPEG and JPEG2000. Also the complexity of FMS-algorithm leads to increased execution time for decompression, because the FMS-algorithm is based on a binary search method.

Future work includes search optimization of the FMS algorithm, as per current implementation this is a constraining step for real-time applications such as video data streaming over the Internet.

Table 6 FMS time execution for decompressed image Face3

Applied single level DWT with two levels DCT							
DCT		Decompressed RMSE		FMS-algorithm Time execution (s)			
Blocks	Scales	2D RMSE	3D RMSE	AC-Matrix$_1$	LH	HL	HH
8 × 8	0.5	4.29	1.26	5.91	Discarded	Discarded	Discarded
8 × 8	1	4.7	1.32	1.1	Discarded	Discarded	Discarded
8 × 8	2	5.43	0.83	0.171	Discarded	Discarded	Discarded
8 × 8	3	6.09	1.68	0.093	Discarded	Discarded	Discarded
8 × 8	4	6.67	1.67	0.046	Discarded	Discarded	Discarded
16 × 16	0.5	1.94	1.29	3.12	Discarded	Discarded	Discarded
16 × 16	1	1.52	1.5	0.59	Discarded	Discarded	Discarded
16 × 16	2	6.65	4.02	0.1	Discarded	Discarded	Discarded

Applied two levels DWT with two levels DCT										
DCT		Scale used in DCT and second level DWT			Decompressed RMSE		FMS-algorithm Time execution (s)			
Blocks	Scales	LH_2	HL_2	HH_2	2D RMSE	3D RMSE	AC-Matrix$_1$	LH_2	HL_2	HH_2
8 × 8	0.5	0.3	0.3	0.3	4.81	1.13	12.00	0.078	0.046	≈0
8 × 8	1	0.3	0.3	0.3	5.53	1.83	1.96	0.015	≈0	≈0
8 × 8	2	0.3	0.3	0.3	6.7	1.48	0.28	0.031	≈0	≈0
8 × 8	3	0.3	0.3	0.3	7.47	1.63	0.078	≈0	0.031	≈0
16 × 16	0.5	0.3	0.3	0.3	5.58	2.08	4.69	0.062	0.015	0.031
16 × 16	1	0.3	0.3	0.3	6.42	1.96	0.98	0.046	≈0	≈0
16 × 16	1.7	0.3	0.3	0.3	7.38	2.61	0.171	≈0	≈0	≈0

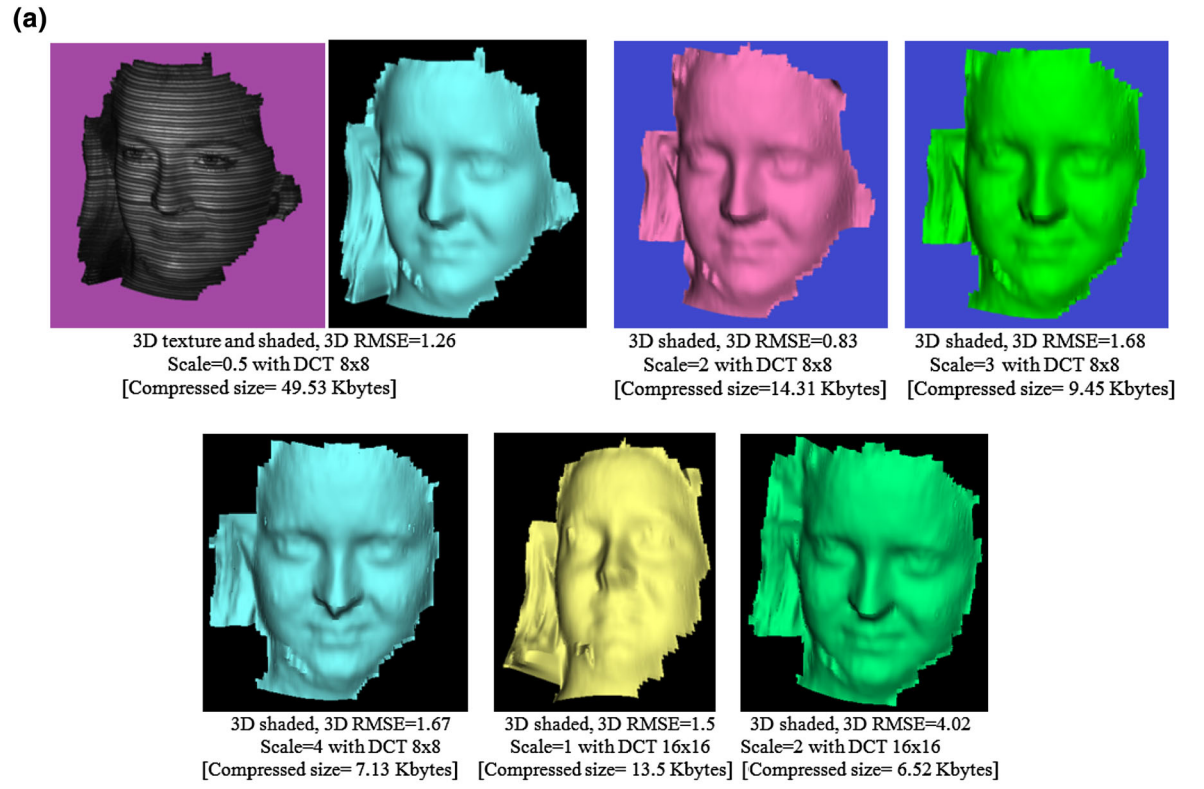


Fig. 13 Decompressed 2D of Face3 image by our proposed decompression method, and then converted to a 3D surface. **a** Decompressed 2D BMP images at Single level DWT converted to 3D surface; 3D surface with scale = 0.5 represent high quality image similar to the original image, and 3D surface with scale = 2 represents median quality image, 3D surface with scale = 3 and 4 are low quality image with slightly degraded surface, while using 16 × 16 block size does not seem to degrade the 3D surface. **b** Decompressed 2D BMP images at two levels DWT converted to 3D surface; 3D surface with scale = 3, 4 and DCT 8x8 represent low quality image degraded surface. However, the block size of 16 × 16 DCT used in our approach has a better quality 3D surface for higher compression ratios

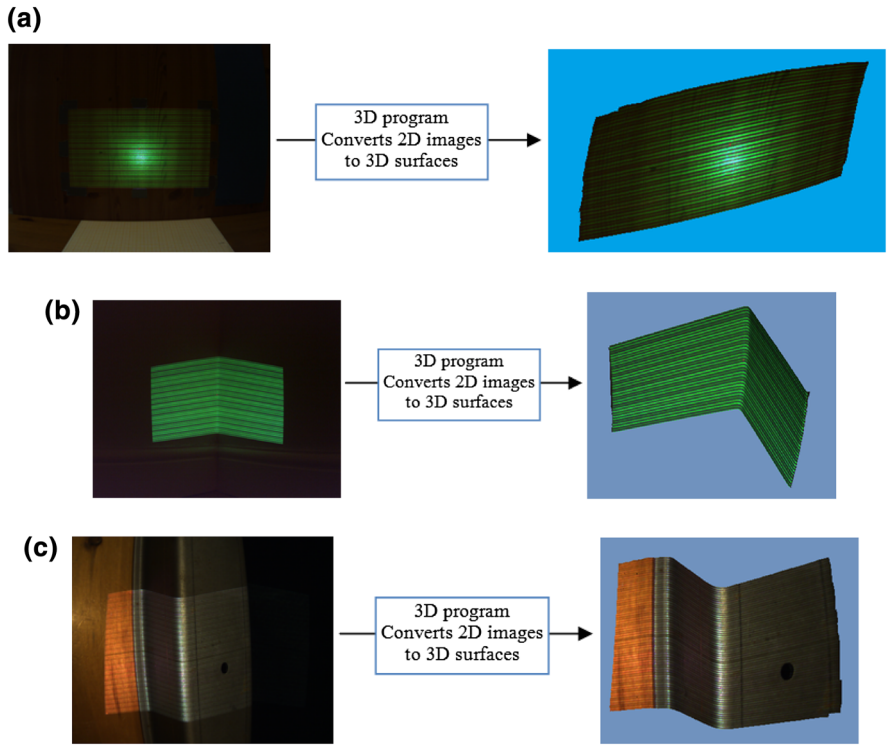


Fig. 14 Original 2D images with 3D surface conversion. **a** Original 2D "Wall" dimensions 1280 × 1024, Image size: 3.75 Mbytes, converted to 3D surface. **b** Original 2D "Room" dimensions 1280 × 1024, Image size: 3.75 Mbytes, converted to 3D surface. **c** Original 2D "Corner" dimensions 1280 × 1024, Image size: 3.75 Mbytes, converted to 3D surface

Table 7 Compressed sizes for 2D colour image Wall

Applied single level DWT with two levels DCT					
Single level DWT High-frequencies (HL, LH and HH) For all colour layer	DCT parameters				Compressed sizes (Kbytes)
	Block sizes	Scales			
		Y	Cb	Cr	
Ignored	8 × 8	0.5	1	1	26.3
Ignored	8 × 8	1	2	2	14.23
Ignored	8 × 8	2	4	4	7.53
Ignored	16 × 16	0.5	1	1	12.3
Ignored	16 × 16	1	2	2	6.55

Table 7 continued

Single level DWT high-frequencies			Block sizes	Red	Green	Blue	Compressed sizes (Kbytes)
Ignored			8 × 8	1	1	1	24.42
Ignored			8 × 8	3	3	3	8.47
Ignored			8 × 8	5	5	5	5.46
Ignored			16 × 16	1	1	1	11.47
Ignored			16 × 16	2	2	2	5.9
Ignored			16 × 16	2.5	2.5	2.5	4.85
Applied two levels DWT with two levels DCT							
Two levels DWT high-frequencies			DCT parameters				Compressed sizes (Kbytes)
Ratio value (Eq. (5)) for all colour layers			Block sizes	Scales			
LH_2	HL_2	HH_2		Y	Cb	Cr	
0.3	0.3	0.3	8 × 8	0.5	1	1	28.1
0.3	0.3	0.3	8 × 8	1	2	2	11.8
0.3	0.3	0.3	16 × 16	0.5	1	1	21.8
0.3	0.3	0.3	16 × 16	1	2	2	8.29
Two levels DWT high-frequencies			Block sizes	Red	Green	Blue	Compressed sizes (Kbytes)
0.3	0.3	0.3	8 × 8	1	1	1	19.32
0.3	0.3	0.3	8 × 8	2	2	2	8.4
0.3	0.3	0.3	16 × 16	1	1	1	12.89
0.3	0.3	0.3	16 × 16	2	2	2	5

Table 8 Compressed size for 2D colour image Room

Applied single level DWT with two levels DCT					
Single level DWT High-frequencies (HL, LH and HH) For all colour layer	DCT parameters				Compressed sizes (Kbytes)
	Block sizes	Scales			
		Y	Cb	Cr	
Ignored	8 × 8	0.5	1	1	58.21
Ignored	8 × 8	1	2	2	34
Ignored	8 × 8	2	4	4	18.92

Table 8 continued

Applied single level DWT with two levels DCT					
Single level DWT High-frequencies (HL, LH and HH) For all colour layer	DCT parameters				Compressed sizes (Kbytes)
	Block sizes	Scales			
		Y	Cb	Cr	
Ignored	16 × 16	0.5	1	1	29.86
Ignored	16 × 16	1	2	2	16.19
Ignored	16 × 16	2	4	4	8.07

Single level DWT high-frequencies	Block sizes	Red	Green	Blue	Compressed sizes (Kbytes)
Ignored	8 × 8	1	1	1	53.25
Ignored	8 × 8	3	3	3	17.66
Ignored	8 × 8	5	5	5	10.73
Ignored	8 × 8	9	9	9	6.28
Ignored	16 × 16	1	1	1	23.83
Ignored	16 × 16	3	3	3	7.91
Ignored	16 × 16	5	5	5	4.65
Ignored	16 × 16	7	7	7	3.38

Applied two levels DWT with two levels DCT							
Two levels DWT high-frequencies Ratio value (Eq. (5)) for all colour layers			DCT parameters				Compressed sizes (Kbytes)
LH_2	HL_2	HH_2	Block sizes	Scales			
				Y	Cb	Cr	
0.3	0.3	0.3	8 × 8	0.5	1	1	47.25
0.3	0.3	0.3	8 × 8	1	2	2	23.96
0.3	0.3	0.3	8 × 8	2	4	4	11.85
0.3	0.3	0.3	16 × 16	0.5	1	1	35.94
0.3	0.3	0.3	16 × 16	1	2	2	17.62
0.3	0.3	0.3	16 × 16	2	4	4	7.94

Two levels DWT high-frequencies			Block sizes	Red	Green	Blue	Compressed sizes (Kbytes)
0.3	0.3	0.3	8 × 8	1	1	1	49
0.3	0.3	0.3	8 × 8	3	3	3	10.42
0.3	0.3	0.3	8 × 8	5	5	5	5.12
0.3	0.3	0.3	16 × 16	1	1	1	36.34
0.3	0.3	0.3	16 × 16	3	3	3	6.61
0.3	0.3	0.3	16 × 16	5	5	5	2.93

Table 9 Compressed size for 2D colour image Corner

Applied single level DWT with two levels DCT						
Single level DWT High-frequencies (HL, LH and HH) For all colour layer			DCT parameters			Compressed sizes (Kbytes)
			Block sizes	Scales		
				Y	Cb	Cr
Ignored			8 × 8	0.5	1	1	31.59
Ignored			8 × 8	1	2	2	15.85
Ignored			16 × 16	0.5	1	1	15.39
Ignored			16 × 16	0.8	6	6	7.8
Single level DWT high-frequencies			Block sizes	Red	Green	Blue	Compressed sizes (Kbytes)
Ignored			8 × 8	1	1	1	44.5
Ignored			8 × 8	2	2	2	22
Ignored			16 × 16	1	1	1	21.16
Ignored			16 × 16	2	2	2	10
Applied two levels DWT with two levels DCT						
Two levels DWT high-frequencies Ratio value (Eq. (5)) for all colour layers			DCT parameters			Compressed sizes (Kbytes)
LH_2	HL_2	HH_2	Block sizes	Scales		
				Y	Cb	Cr
0.3	0.3	0.3	8 × 8	0.5	1	1	31.72
0.3	0.3	0.3	8 × 8	1	2	2	13.11
0.3	0.3	0.3	16 × 16	0.5	1	1	23.14

Table 10 Estimated execution time for FMS-algorithm at single level DWT for Wall image

Compressed sizes (Kbytes)	Block sizes	Time execution (s)		
		Y AC-Matrix$_1$	Cb AC-Matrix$_1$	Cr AC-Matrix$_1$
26.3	8 × 8	1.77	0.093	0.187
14.23	8 × 8	0.202	0.031	0.078
7.53	8 × 8	0.062	≈0	0.031
12.3	16 × 16	0.54	0.015	0.031
6.55	16 × 16	0.109	≈0	0.031
Compressed size (Kbytes)	Block sizes	Red AC-Matrix$_1$	Green AC-Matrix$_1$	Blue AC-Matrix$_1$
24.42	8 × 8	0.12	1.27	0.12
8.47	8 × 8	≈0	0.078	0.031
5.46	8 × 8	≈0	0.046	≈0
11.47	16 × 16	0.078	0.46	0.093
5.9	16 × 16	0.015	0.078	0.031
4.85	16 × 16	0.031	0.031	≈0

Table 11 Estimated execution time for FMS-algorithm at single level DWT for Room image

Compressed sizes (Kbytes)	Block sizes	Time execution (s)		
		Y AC-Matrix	Cb AC-Matrix	Cr AC-Matrix
58.21	8 × 8	1.2	0.093	0.17
34	8 × 8	0.17	0.046	0.046
18.92	8 × 8	0.046	0.046	0.031
29.86	16 × 16	0.96	0.062	0.093
16.19	16 × 16	0.156	0.046	0.046
8.07	16 × 16	0.046	0.031	≈0
Compressed sizes (Kbytes)	Block sizes	Red AC-Matrix	Green AC-Matrix	Blue AC-Matrix
53.25	8 × 8	0.062	1.4	0.21
17.66	8 × 8	≈0	0.124	0.015
10.73	8 × 8	≈0	0.062	≈0
6.28	8 × 8	≈0	0.031	≈0
23.83	16 × 16	0.046	0.68	0.078
7.91	16 × 16	≈0	0.062	0.031
4.65	16 × 16	≈0	0.046	≈0
3.38	16 × 16	≈0	0.031	≈0

Table 12 Estimated execution time for FMS-algorithm at single level DWT for Corner image

Compressed sizes (Kbytes)	Block sizes	Time execution (s)		
		Y AC-Matrix	Cb AC-Matrix	Cr AC-Matrix
31.59	8 × 8	1.2	0.031	0.015
15.85	8 × 8	0.3	0.031	0.031
15.39	16 × 16	1.07	0.031	0.046
7.8	16 × 16	0.5	≈0	≈0
Compressed sizes (Kbytes)	Block sizes	Red AC-Matrix	Green AC-Matrix	Blue AC-Matrix
44.5	8 × 8	0.34	0.48	0.34
22	8 × 8	0.06	0.09	0.09
21.16	16 × 16	0.46	0.48	0.37
10	16 × 16	0.1	0.14	0.1

Table 13 Estimated execution time for FMS-algorithm at two levels DWT for Wall image

Compressed sizes (Kbytes)	Block sizes	Y				Cb				Cr			
		AC-Matrix$_1$	HL_2	LH_2	HH_2	AC-Matrix$_1$	LH_2	HL_2	HH_2	AC-Matrix$_1$	HL_2	LH_2	HH_2
28.1	8 × 8	3.1	0.062	0.062	0.031	0.062	0.03	≈0	≈0	0.24	0.031	≈0	≈0
11.8	8 × 8	0.6	0.015	≈0	≈0	0.046	0.031	≈0	≈0	0.046	0.031	≈0	≈0
21.8	16 × 16	1.5	0.031	0.031	≈0	0.046	0.03	0.031	≈0	0.093	0.031	≈0	≈0
8.29	16 × 16	0.026	0.015	≈0	0.031	0.031	0.03	≈0	≈0	0.062	0.015	≈0	≈0

Table 13 continued

Compressed sizes (Kbytes)	Block sizes	Red				Green				Blue			
		AC-Matrix$_1$	HL_2	LH_2	HH_2	AC-Matrix$_1$	HL_2	LH_2	HH_2	AC-Matrix$_1$	HL_2	LH_2	HH_2
19.32	8 × 8	0.23	0	≈0	≈0	1.5	0.031	0.031	≈0	0.26	≈0	≈0	≈0
8.4	8 × 8	0.062	0.031	≈0	≈0	0.24	≈0	≈0	≈0	0.078	≈0	0.031	≈0
12.89	16 × 16	0.21	0.078	≈0	0.031	0.081	≈0	≈0	≈0	0.17	≈0	0	≈0
5	16 × 16	0.062	≈0	≈0	≈0	0.093	0.031	0.031	≈0	0.062	≈0	≈0	≈0

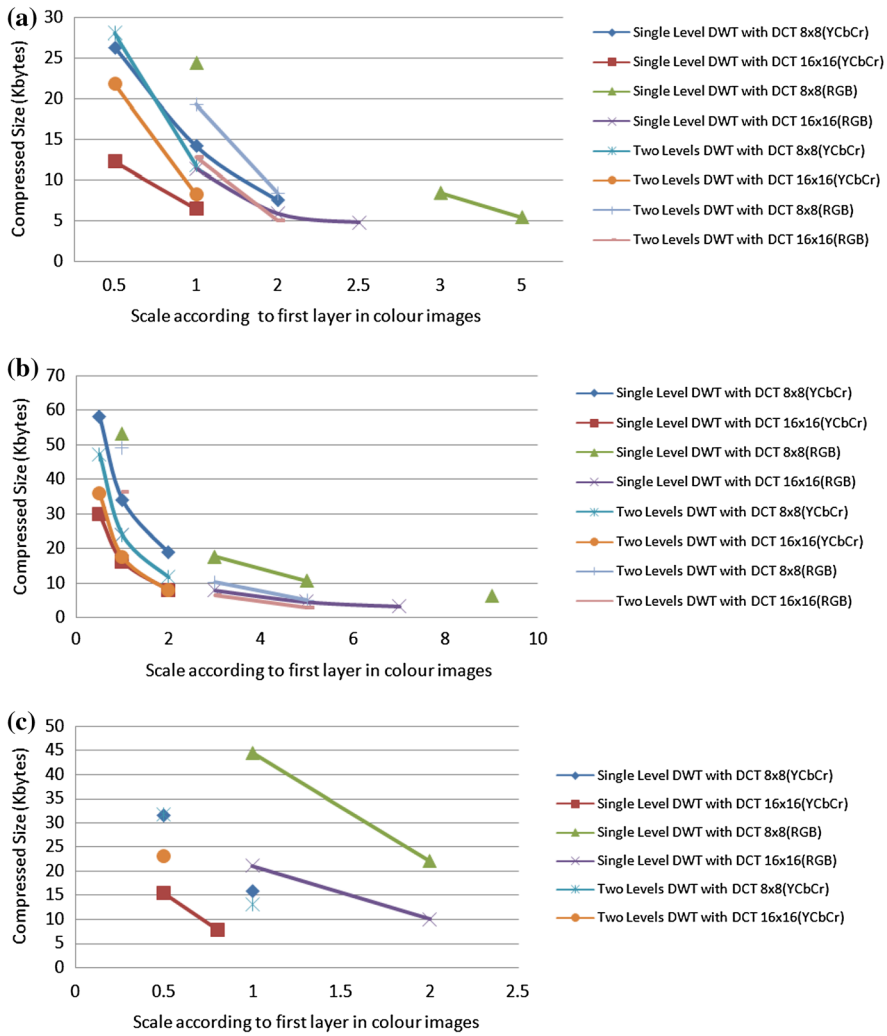


Fig. 15 Illustrates Tables 7, 8 and 9 as chart according to "Scale" and "Compressed Size". **a** Chart from Table 7 for Wall image by using different scale value in our approach, as shown in this figure the results obtained by our approach applied on RGB layers has better compression ratio than using YcbCr layers. **b** Chart from Table 8 for Room image by using different scale value in our approach, as shown in this figure the results obtained by our approach applied on RGB layers has better compression ratio than using YcbCr layers. **c** Chart from Table 9 for Corner image by using different scale value in our approach, as shown in this figure the results obtained by single level DWT with DCT (16 x 16) applied on YcbCr layers has better compression ratio than other options illustrated in this chart

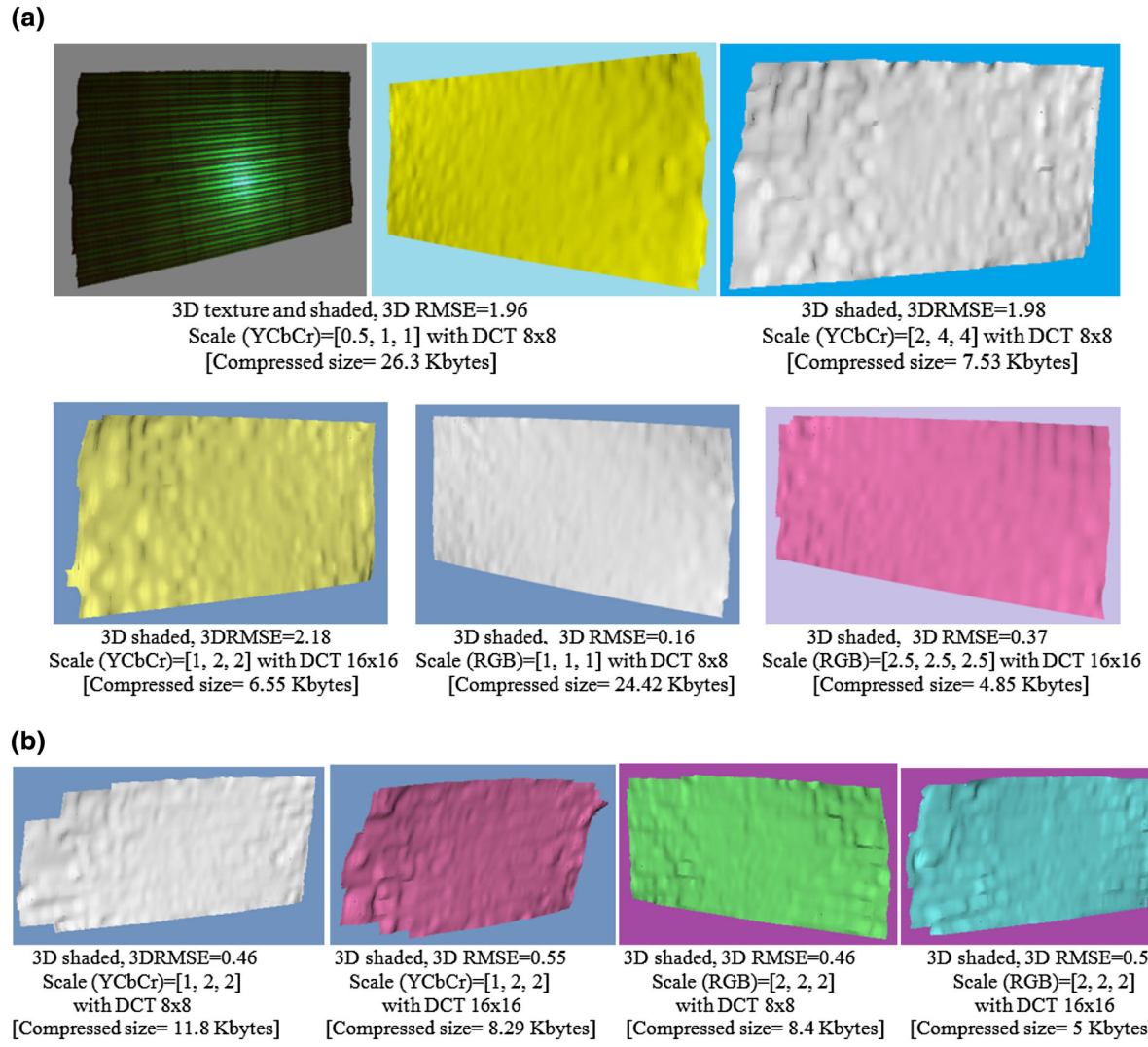
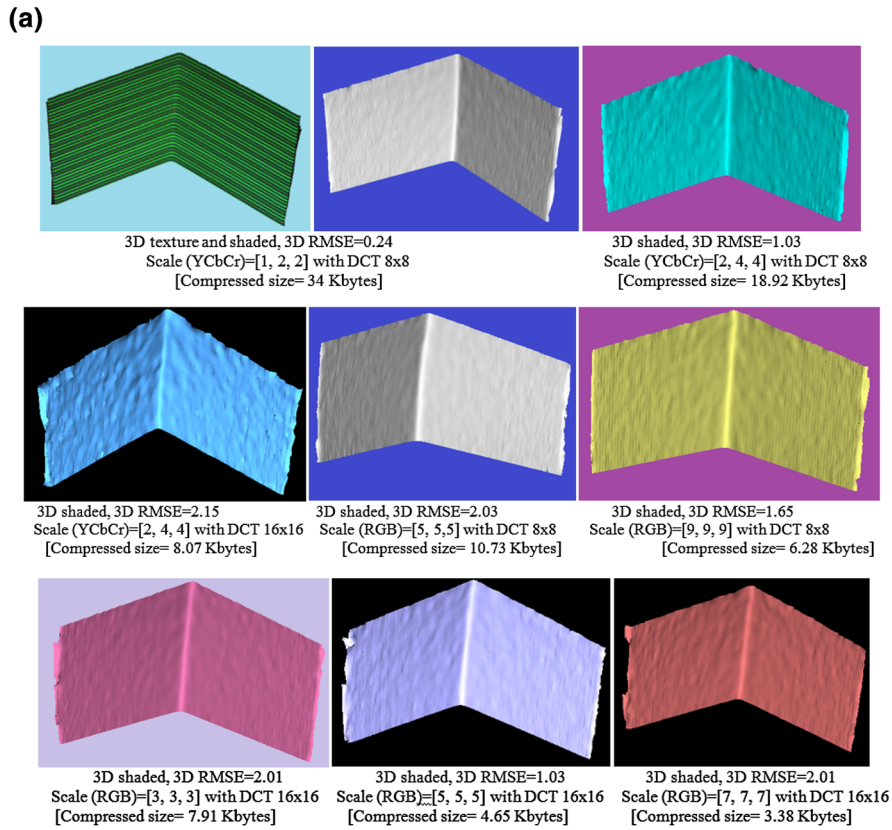
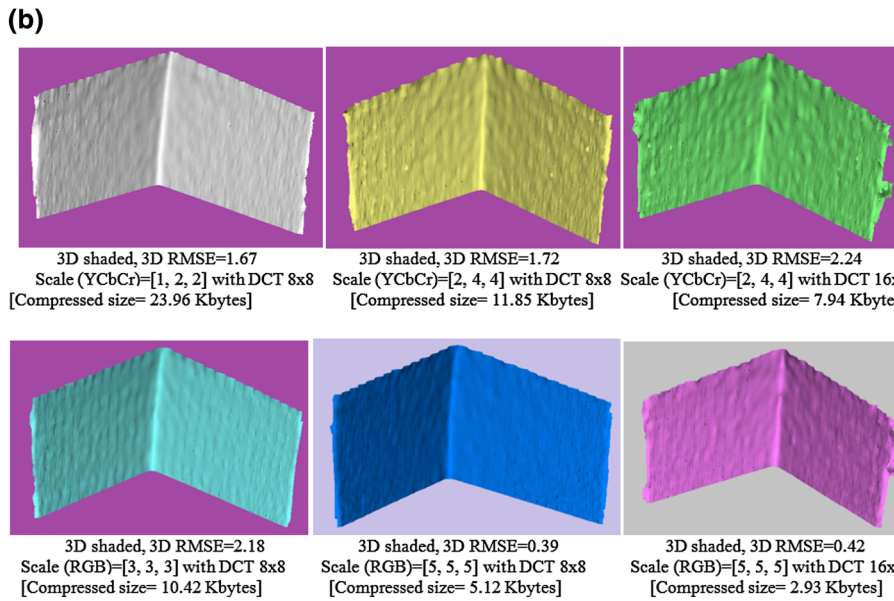


Fig. 16 Decompressed colour Wall image by our proposed decompression approach, and then converted to 3D surface. **a** Decompressed 2D BMP images at single level DWT converted to 3D surface; decompressed 3D surface by using RGB layer has better quality than YCbCr layer. **b** Decompressed 2D BMP images at two levels DWT converted to 3D surface, also decompressed 3D surface using RGB layer has better quality than YCbCr layer at higher compression ratio


(a)
3D texture and shaded, 3D RMSE=0.24
Scale (YCbCr)=[1, 2, 2] with DCT 8x8
[Compressed size= 34 Kbytes]
3D shaded, 3D RMSE=1.03
Scale (YCbCr)=[2, 4, 4] with DCT 8x8
[Compressed size= 18.92 Kbytes]
3D shaded, 3D RMSE=2.15
Scale (YCbCr)=[2, 4, 4] with DCT 16x16
[Compressed size= 8.07 Kbytes]
3D shaded, 3D RMSE=2.03
Scale (RGB)=[5, 5,5] with DCT 8x8
[Compressed size= 10.73 Kbytes]
3D shaded, 3D RMSE=1.65
Scale (RGB)=[9, 9, 9] with DCT 8x8
[Compressed size= 6.28 Kbytes]
3D shaded, 3D RMSE=2.01
Scale (RGB)=[3, 3, 3] with DCT 16x16
[Compressed size= 7.91 Kbytes]
3D shaded, 3D RMSE=1.03
Scale (RGB)=[5, 5, 5] with DCT 16x16
[Compressed size= 4.65 Kbytes]
3D shaded, 3D RMSE=2.01
Scale (RGB)=[7, 7, 7] with DCT 16x16
[Compressed size= 3.38 Kbytes]


(b)
3D shaded, 3D RMSE=1.67
Scale (YCbCr)=[1, 2, 2] with DCT 8x8
[Compressed size= 23.96 Kbytes]
3D shaded, 3D RMSE=1.72
Scale (YCbCr)=[2, 4, 4] with DCT 8x8
[Compressed size= 11.85 Kbytes]
3D shaded, 3D RMSE=2.24
Scale (YCbCr)=[2, 4, 4] with DCT 16x16
[Compressed size= 7.94 Kbytes]
3D shaded, 3D RMSE=2.18
Scale (RGB)=[3, 3, 3] with DCT 8x8
[Compressed size= 10.42 Kbytes]
3D shaded, 3D RMSE=0.39
Scale (RGB)=[5, 5, 5] with DCT 8x8
[Compressed size= 5.12 Kbytes]
3D shaded, 3D RMSE=0.42
Scale (RGB)=[5, 5, 5] with DCT 16x16
[Compressed size= 2.93 Kbytes]

◀**Fig. 17** Decompressed colour *Room* image by our proposed decompression method, and then converted to 3D surface. **a** Decompressed 2D BMP images at Single level DWT results converted to 3D surface; decompressed 3D surface by using RGB layer has better quality than YCbCr layer at higher compression ratio using both block sizes of 8 × 8 or 16 × 16 by DCT. **b** Decompressed 2D BMP images at two levels DWT converted to 3D surface; decompressed 3D surface by using RGB layer has better quality than YCbCr layer at higher compression ratio using block sizes of 8 × 8, also by using block 16 × 16 the surface is still approximately non-degraded

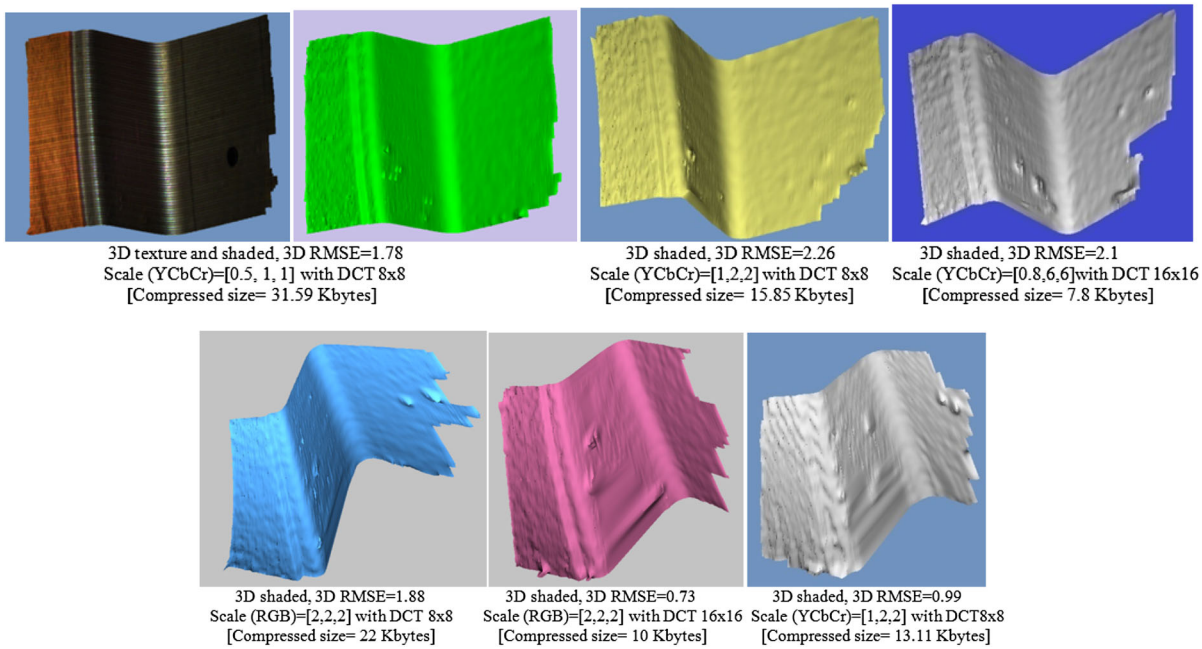


Fig. 18 Decompressed colour "Corner" image by our proposed decompression method, and then converted to 3D surface. The decompressed 3D surface at single level DWT using YCbCr has better quality at higher compression ratio than using RGB layers, also at two levels DWT degradation appears and some parts from surface fail to reconstruct

Table 14 Estimated execution time for FMS-algorithm at two levels DWT for Room image

Compressed sizes (Kbytes)	Block sizes	Y				Cb				Cr			
		AC-Matrix$_1$	HL_2	LH_2	HH_2	AC-Matrix$_1$	HL_2	LH_2	HH_2	AC-Matrix$_1$	HL_2	LH_2	LH_2
47.25	8 × 8	0.15	0.046	≈0	≈0	0.1	0.031	0.031	≈0	2.19	0.078	0.062	0.062
23.96	8 × 8	0.046	0.015	≈0	≈0	0.015	0.031	≈0	≈0	0.43	0.078	0.031	0.031
11.85	8 × 8	≈0	≈0	≈0	≈0	≈0	0.031	≈0	≈0	0.078	0.015	≈0	≈0
35.94	16 × 16	0.124	0.031	≈0	≈0	0.078	0.015	0.031	≈0	1.21	0.062	0.015	0.015
17.62	16 × 16	0.015	0.031	≈0	≈0	0.031	0.031	≈0	≈0	0.28	0.031	0.031	0.031
7.94	16 × 16	0.015	0.031	≈0	≈0	0.031	0.031	0.031	≈0	0.093	0.031	≈0	≈0

Table 14 continued

Compressed sizes (Kbytes)	Block sizes	Red				Green				Blue			
		AC-Matrix$_1$	HL_2	LH_2	LH_2	AC-Matrix$_1$	HL_2	LH_2	HH_2	AC-Matrix$_1$	HL_2	LH_2	LH_2
49	8 × 8	0.35	0.046	0.031	0.031	1.76	0.015	≈0	≈0	0.062	≈0	≈0	≈0
10.42	8 × 8	0.046	≈0	≈0	≈0	0.124	≈0	≈0	≈0	≈0	≈0	≈0	≈0
5.12	8 × 8	0.031	≈0	≈0	≈0	0.062	≈0	≈0	≈0	≈0	≈0	≈0	≈0
36.34	16 × 16	0.17	0.046	0.062	0.062	1.27	0.031	≈0	≈0	0.015	0.031	≈0	≈0
6.61	16 × 16	≈0	≈0	≈0	≈0	0.1	≈0	≈0	≈0	≈0	≈0	≈0	≈0
2.93	16 × 16	0.031	≈0	≈0	≈0	0.062	≈0	≈0	≈0	≈0	0.031	≈0	≈0

Table 15 Estimated execution time for FMS-algorithm at two levels DWT for *Corner* image

Compressed sizes (Kbytes)	Block sizes	Y				Cb				Cr			
		AC-Matrix$_1$	LH_2	HL_2	HH_2	AC-Matrix$_1$	LH_2	HL_2	HH_2	AC-Matrix$_1$	LH_2	HL_2	HH_2
31.72	8 × 8	4	0.016	0.031	≈0	0.093	≈0	0.015	≈0	0.1	0.031	0.031	≈0
13.11	8 × 8	0.85	≈0	0.031	≈0	0.031	≈0	≈0	≈0	0.046	≈0	0.031	≈0
23.14	16 × 16	0.2.49	0.015	0.016	≈0	0.062	0.031	≈0	≈0	0.1	0.031	0.031	≈0

Table 16 Comparison JPEG2000 and JPEG with our approach for Face1 image

Proposed algorithm			JPEG2000		JPEG	
Compressed sizes (Kbytes)	2D RMSE	3D RMSE	2D RMSE	3D RMSE	2D RMSE	3D RMSE
≈29.33	4.83	2.31	3.48	3.31	12	2.73
≈10.58	6.21	2.4	5.33	3.49	Not Applicable	Not Applicable
≈6.37	7.41	3.39	6.32	2.91	Not Applicable	Not Applicable

Table 17 Comparison JPEG2000 and JPEG with our approach for Face2 image

Our Proposed algorithm			JPEG2000		JPEG	
Compressed sizes (Kbytes)	2D RMSE	3D RMSE	2D RMSE	3D RMSE	2D RMSE	3D RMSE
≈26.85	5.22	2.19	4.1	1.51	8.57	1.67
≈14.42	5.86	1.64	5.3	2.68	Not Applicable	Not Applicable
≈7.4	6.96	2.16	6.61	2.57	Not Applicable	Not Applicable
≈4.66	8.43	1.48	7.61	2.62	Not Applicable	Not Applicable

Table 18 Comparison JPEG2000 and JPEG with our approach for Face3 image

Proposed algorithm			JPEG2000		JPEG	
Compressed sizes (Kbytes)	2D RMSE	3D RMSE	2D RMSE	3D RMSE	2D RMSE	3D RMSE
≈27.5	4.7	1.32	3.83	1.25	7.64	1.78
≈14.31	5.43	0.83	5.05	1.82	Not Applicable	Not Applicable
≈6.52	6.65	4.02	6.54	1.85	Not Applicable	Not Applicable

Table 19 Comparison JPEG2000 and JPEG with our approach for Wall image

Proposed algorithm			JPEG2000		JPEG	
Compressed sizes (Kbytes)	2D RMSE	3D RMSE	2D RMSE	3D RMSE	2D RMSE	3D RMSE
≈26.3	4.37	1.96	2.63	0.31	9.66	0.66
≈11.47	4.36	0.2	3.47	0.49	Not Applicable	Not Applicable
≈4.85	4.85	0.37	4.79	1.1	Not Applicable	Not Applicable

Table 20 Comparison JPEG2000 and JPEG with our approach for Room image

Proposed algorithm			JPEG2000		JPEG	
Compressed sizes (Kbytes)	2D RMSE	3D RMSE	2D RMSE	3D RMSE	2D RMSE	3D RMSE
≈17.66	6.36	0.12	7.59	0.23	20	113.59 (*not matched*)
≈6.28	9.0	1.65	11.33	1.35	Not Applicable	Not Applicable
≈3.38	9.22	2.21	13.59	94.67 (*not matched*)	Not Applicable	Not Applicable
≈2.93	11.26	0.42	15.06	Not Applicable	Not Applicable	Not Applicable

Table 21 Comparison JPEG2000 and JPEG with our approach for Corner image

Proposed algorithm			JPEG2000		JPEG	
Compressed sizes (Kbytes)	2D RMSE	3D RMSE	2D RMSE	3D RMSE	2D RMSE	3D RMSE
≈31.59	3.58	1.78	3.12	1.92	5.86	16.46
≈15.39	4.59	0.42	3.86	74.64 (*not matched*)	Not Applicable	Not Applicable
≈7.8	6.5	2.1	4.64	69.55 (*not matched*)	Not Applicable	Not Applicable

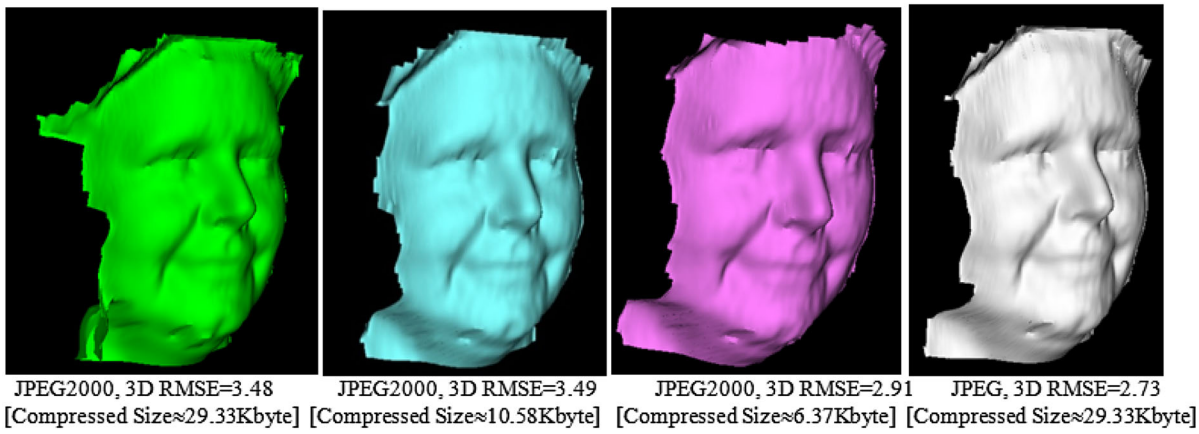


Fig. 19 Decompressed 2D Face1 image by using JPEG2000 and JPEG algorithm, JPEG algorithm can't compress the 2D Face1 image under 29 Kbytes

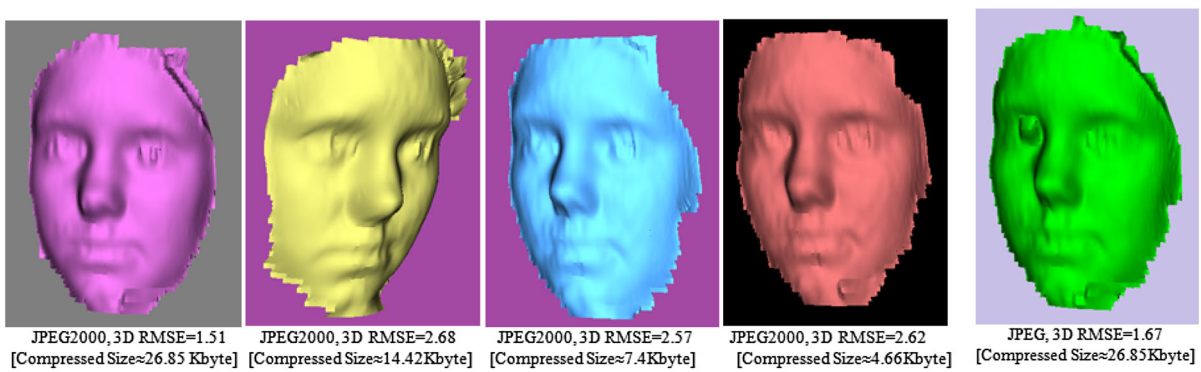


Fig. 20 Decompressed 2D Face2 image by using JPEG2000 and JPEG algorithm degradation appeared by JPEG on the surface at 26 Kbytes

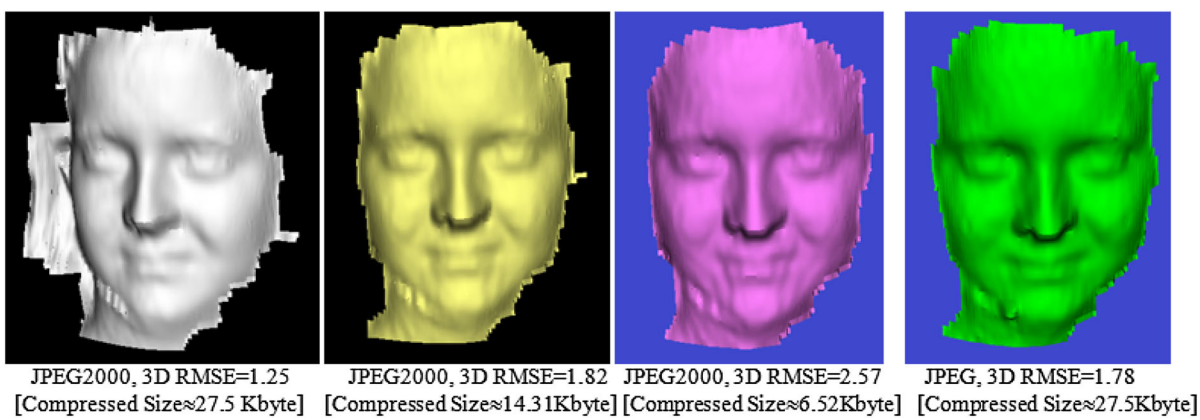


Fig. 21 Decompressed 2D Face3 image by using JPEG2000 and JPEG algorithm, parts of images failed to reconstruct in 3D by JPEG2000 algorithm at 14 Kbytes, degradation appears on the surface by JPEG2000 under 7 Kbytes, also JPEG algorithm degrades the surface at compressed size 27 Kbytes (JPEG fails to compress under 27 Kbytes)

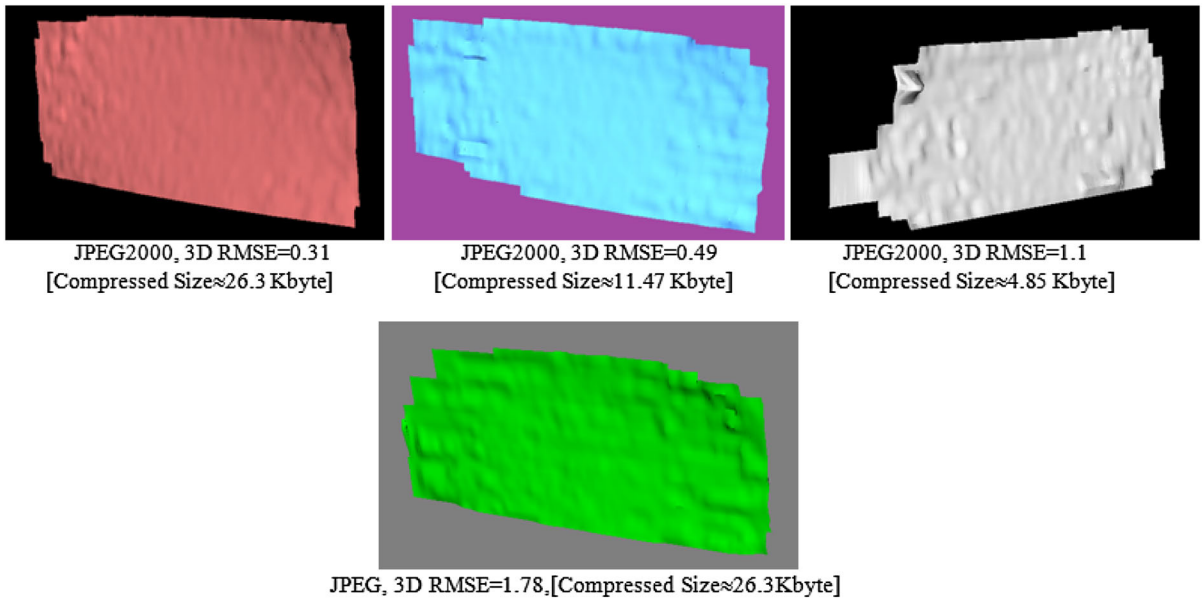


Fig. 22 Decompressed Wall image by using JPEG2000 and JPEG algorithm, degradation appears on surface by JPEG2000 fewer than 12 Kbytes, also JPEG algorithm degrades the surface at compressed size 27 Kbytes, JPEG fails to compress under 27 Kbytes

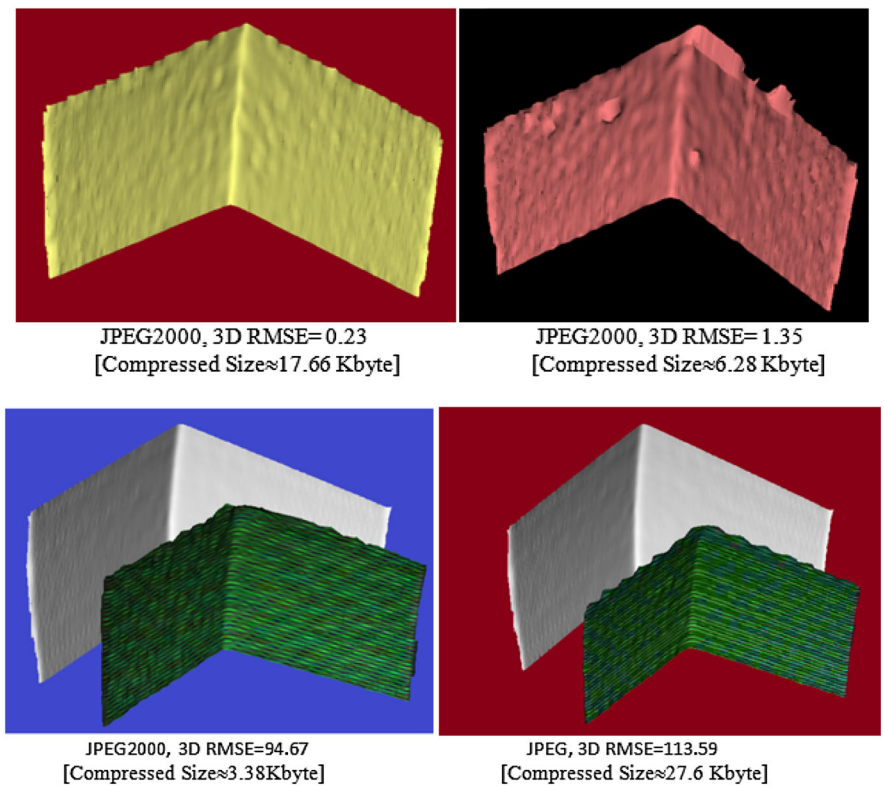


Fig. 23 Decompressed Room image by using JPEG2000 and JPEG algorithm, degradation appears on surface by JPEG2000 fewer than 6 Kbytes, also JPEG2000 cannot reconstruct 3D surface matches with original surface (*grey colour*) at 4 Kbytes, similarly, JPEG algorithm fails to reconstruct 3D surface matching with original surface at 27 Kbytes

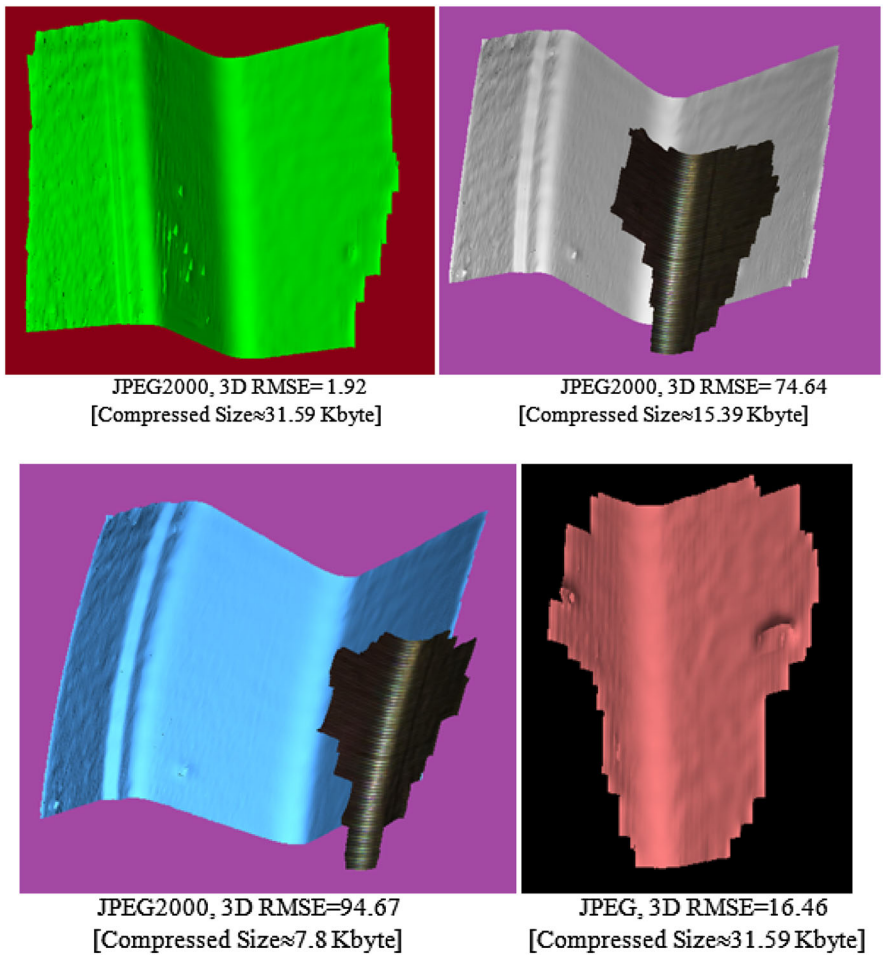


Fig. 24 Decompressed Corner image by using JPEG2000 and JPEG algorithm, *top-left* surface decompressed successfully by JPEG2000, in *top-right* decompressed surface by JPEG2000 not matched with original surface (*grey surface*). Under 15 Kbytes, similarly, JPEG algorithm fails to compress successfully fewer than 31 Kbytes

References

1. Acharya, T., & Tsai, P. S. (2005). *JPEG2000 standard for image compression: Concepts, algorithms and VLSI architectures*. New York: Wiley.
2. Ahmed, N., Natarajan, T., & Rao, K. R. (1974). Discrete cosine transforms. *IEEE Transactions on Computer, C-23*, 90–93.
3. Al-Haj, A. (2007). Combined DWT–DCT digital image watermarking. *Science Publications, Journal of Computer Science, 3*(9), 740–746.
4. Chen, P., & Chang, J.-Y. (2013). An adaptive quantization scheme for 2-D DWT coefficients. *International Journal of Applied Science and Engineering, 11*(1), 85–100.
5. Christopoulos, C., Askelof, J., & Larsson, M. (2000). Efficient methods for encoding regions of interest in the upcoming JPEG 2000 still image coding standard. *IEEE Signal Processing Letters, 7*(9), 247–249.
6. Esakkirajan, S., Veerakumar, T., Senthil Murugan, V., & Navaneethan, P. (2008). Image compression using multiwavelet and multi-stage vector quantization. *WASET International Journal of Signal Processing, 4*(4), 246–253.
7. Gonzalez, R. C., & Woods, R. E. (2001). *Digital image processing*. Boston: Addison Wesley Publishing Company.
8. Horiuchi, T., & Tominaga, S. (2008). Color image coding by colorization approach. *EURASIP Journal on Image and Video Processing*. doi:10.1155/2008/158273.
9. Knuth, D. (1997). *Sorting and searching*: Section 6.2.1: Searching an ordered table. In *The art of computer programming* (3rd Ed., Vol. 3, pp. 409–426). Reading, MA: Addison-Wesley. ISBN 0-201-89685-0.

10. Liu, K.-C. (2012). Prediction error preprocessing for perceptual color image compression. *EURASIP Journal on Image and Video Processing*. doi:10.1186/1687-5281-2012-3.
11. Richardson, I. E. G. (2002). *Video codec design*. New York: Wiley.
12. Rodrigues, M., Kormann, M., Schuhler, C., & Tomek, P. (2013). Structured light techniques for 3D surface reconstruction in robotic tasks. In J. Kacprzyk (Ed.), *Advances in intelligent systems and computing* (pp. 805–814). Heidelberg: Springer.
13. Rodrigues, M., Kormann, M., Schuhler, C., & Tomek, P. (2013b). Robot trajectory planning using OLP and structured light 3D machine vision. In *Lecture notes in computer science Part II. LCNS* (Vol. 8034(8034), pp. 244–253). Heidelberg: Springer.
14. Rodrigues, M., Kormann, M., Schuhler, C., & Tomek, P. (2013d). An intelligent real time 3D vision system for robotic welding tasks. In *Mechatronics and its applications.* IEEE Xplore (pp. 1–6).
15. Rodrigues, M., Osman, A., & Robinson, A. (2013). Partial differential equations for 3D data compression and reconstruction. *Journal of Advances in Dynamical Systems and Applications, 12*(3), 371–378.
16. Rodrigues, M., Robinson, A., & Osman, A. (2010). Efficient 3D data compression through parameterization of free-form surface patches. In *Proceedings of the 2010 international conference on signal process and multimedia applications* (*SIGMAP*, pp. 130–135). Athena: IEEE.
17. Sadashivappa, G., & Ananda Babu, K. V. S. (2002). Performance analysis of image coding using wavelets. *International Journal of Computer Science and Network Security, 8*(10), 144–151.
18. Sayood, K. (2006). *Introduction to data compression* (3rd Ed.). San Francisco: Morgan Kaufman Publishers.
19. Schaefer, G. (2014). Soft computing-based colour quantization. *EURASIP Journal on Image and Video Processing*. doi:10.1186/1687-5281-2014-8.
20. Siddeq, M. M. (2012). Using Sequential Search Algorithm with Single level Discrete Wavelet Transform for Image Compression (SSA-W). *Journal of Advances in Information Technology, 3*(4), 236–249.
21. Siddeq, M. M., & Al-Khafaji, G. (2013). Applied Minimize-Matrix-Size Algorithm on the Transformed images by DCT and DWT used for image Compression. *International Journal of Computer Applications, 70*(15), 34–40.
22. Siddeq, M. M., & Rodrigues, M. (2014a). A new 2D image compression technique for 3D surface reconstruction. In *18th International conference on circuits, systems, communications and computers*, Santorin Island, Greece (pp. 379–386).
23. Siddeq, M. M., & Rodrigues, M. A. (2014b). A Novel Image Compression Algorithm for high resolution 3D Reconstruction. *3D Research, Springer*. doi:10.1007/s13319-014-0007-6.
24. Stamm, M. C., & Ray Liu, K. J. (2010). Wavelet-based image compression anti-forensics. In *Proceedings of 2010 IEEE 17th international conference on image processing*, Hong Kong, 26–29 September 2010 (pp. 1737–1740).
25. Suzuki, T., & Ikehara, M. (2013). Integer fast lapped transforms based on direct-lifting of DCTs for lossy-to-lossless image coding. *EURASIP Journal on Image and Video Processing*. doi:10.1186/1687-5281-2013-65.