3D Res (2017) 8:5
DOI 10.1007/s13319-017-0116-0

3DR EXPRESS

# DCT and DST Based Image Compression for 3D Reconstruction

**Mohammed M. Siddeq · Marcos A. Rodrigues**

Received: 19 December 2016 / Revised: 27 January 2017 / Accepted: 31 January 2017 / Published online: 4 February 2017

**Abstract** This paper introduces a new method for 2D image compression whose quality is demonstrated through accurate 3D reconstruction using structured light techniques and 3D reconstruction from multiple viewpoints. The method is based on two discrete transforms: (1) A one-dimensional Discrete Cosine Transform (DCT) is applied to each row of the image. (2) The output from the previous step is transformed again by a one-dimensional Discrete Sine Transform (DST), which is applied to each column of data generating new sets of high-frequency components followed by quantization of the higher frequencies. The output is then divided into two parts where the low-frequency components are compressed by arithmetic coding and the high frequency ones by an efficient minimization encoding algorithm. At decompression stage, a binary search algorithm is used to recover the original high frequency components. The technique is demonstrated by compressing 2D images up to 99% compression ratio. The decompressed images, which include images with structured light patterns for 3D reconstruction and from multiple viewpoints, are of high perceptual quality yielding accurate 3D reconstruction. Perceptual assessment and objective quality of compression are compared with JPEG and JPEG2000 through 2D and 3D RMSE. Results show that the proposed compression method is superior to both JPEG and JPEG2000 concerning 3D reconstruction, and with equivalent perceptual quality to JPEG2000.

**Keywords** DCT · DST · High frequency minimization · Binary search algorithm

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

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

## 1 Introduction

Transform coding is at the heart of the majority 2D image/video coding systems and standards. Spatial image data (image samples or motion-compensated residual samples) are transformed into a different representation, the transform domain. There are good reasons for transforming image data in this way. Spatial image data is inherently 'difficult' to compress; neighbouring samples are highly correlated and the energy tends to be evenly distributed across an image, making it difficult to discard data or reduce the precision of data without adversely affecting image quality [1, 2].With a suitable choice of transform, the data becomes 'easier' to compress in the transform domain. There are several desirable properties of a transform for compression. It should compact the energy in the image, i.e., concentrate the energy into a small number of significant values; it should decorrelate the data so that discarding ‘insignificant’ data—normally high frequency data—has a minimal effect on image quality; and it should be suitable for practical implementation in software and hardware [3, 4].

The two most widely used image compression transforms are the discrete cosine transform (DCT) and the discrete wavelet transform (DWT) [3–5]. The DCT is usually applied to small, regular blocks of image samples (e.g. $8 \times 8$ squares) and the DWT is usually applied to larger image sections or to complete images. Many alternatives have been proposed, for example 3D transforms (dealing with spatial and temporal correlation), variable block size transforms, fractal transforms, and Gabor analysis. The DCT has proved particularly useful and it is at the core of most current generation of image and video coding standards, including JPEG, H.261, H.263, H.263+, MPEG-l, MPEG-2 and MPEG-4 [6, 7].

To demonstrate the effectiveness of our approach, we focus on compressing 2D image data appropriate for 3D reconstruction. This includes 3D reconstruction from structured light images, and 3D reconstruction from multiple viewpoint images. Previously, we have demonstrated that while geometry and connectivity of a 3D mesh can be tackled by several techniques such as high degree polynomial interpolation [8] or partial differential equations [9, 10], the issue of efficient compression of 2D images both for 3D reconstruction and texture mapping has not yet been addressed in a satisfactory manner. Moreover, in most applications that share common data, it is necessary to transmit 3D models over the Internet. For example, to share CAD/CAM assets, e-commerce applications, update content for entertainment applications, or to support collaborative design, analysis, display of engineering, medical and scientific datasets. Bandwidth imposes hard limits on the amount of data transmission and, together with storage costs calls for more efficient 3D data compression for exchange over the Internet and other networked environments. Using structured light techniques for 3D reconstruction, surface patches can be compressed as a 2D image together with 3D calibration parameters, transmitted over a network and remotely reconstructed (geometry, connectivity and texture map) at the receiving end with the same resolution as the original data [11, 12].

Related to the techniques proposed in this paper, our previous work on data compression is summarized as follows: (1) Focused on compressing structured light images for 3D reconstruction, Siddeq and Rodrigues [11] proposed a method in 2014 where a single level DWT is followed by a DCT on the LL sub-band yielding the DC component and the AC-matrix. A second DWT is applied to the DC components whose second level LL2 sub-band is transformed again by DCT. A matrix minimization algorithm was applied to the AC-matrix and other sub-bands. Compression ratios of up to 98% were achieved. (2) Siddeq and Rodrigues in same year [13], proposed technique where a DWT was applied to variant arrangements of data blocks followed by arithmetic coding. The novel aspect of that paper is at decompression stage, where a Parallel Sequential Search Algorithm was proposed and demonstrated. Compression ratios of up to 98.8% were achieved. (3) In Siddeq and Rodrigues [14] a two-level DWT was applied followed by a DCT to generate a DC-component array and an MA-Matrix (Multi-Array Matrix). The MA-Matrix was then partitioned into blocks and a minimization algorithm coded each block followed by the removal of zero valued coefficients and arithmetic coding. At decompression stage, a new algorithm called Fast-Match-Search decompression was used to reconstruct the high-frequency matrices by computing data probabilities through a binary search algorithm in association with a look up table. A comparative analysis of various combinations of DWT and DCT block sizes was performed, with compression ratios up to 99.5%.

In this paper, we introduce a new method based on DCT and DST for compressing 2D images with structured light patterns for 3D surface reconstruction (i.e., the 2D images have embedded stripe patterns, and the detection and processing of such patterns are used to generate a 3D surface). Additionally, the method is applied to a series of 2D images (with no structured light patterns) and used to convert from multiple 2D images to a 3D surface [15]. Following the discrete transformations, a high-frequency minimization method is used to convert each three adjacent coefficients to a single integer, reducing the data size to a third of its original size. The final step is to apply arithmetic coding to the reduced data. The main steps in the compression algorithm are depicted in Fig. 1.

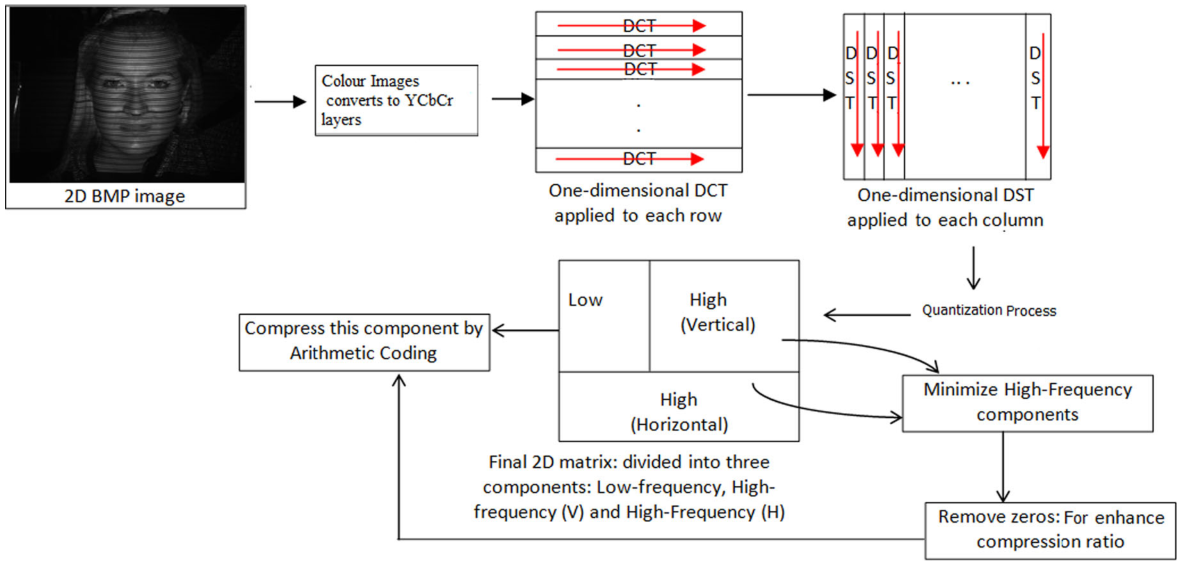

2D BMP image
Colour Images converts to YCbCr layers
DCT
One-dimensional DCT applied to each row
DST
One-dimensional DST applied to each column
Quantization Process
Low
High (Vertical)
High (Horizontal)
Final 2D matrix: divided into three components: Low-frequency, High-frequency (V) and High-Frequency (H)
Compress this component by Arithmetic Coding
Minimize High-Frequency components
Remove zeros: For enhance compression ratio

**Fig. 1** The main steps of the proposed compression algorithm

## 2 Using One-Dimensional Discrete Cosine Transform (DCT)

The one-dimensional DCT is used to transform each row from an image (spatial domain) to obtain transform image called "$T_{dct}$", a shown in the following [3–5]:

$$T_{dct}(i) = \frac{\sqrt{2}}{n} C(i) \sum_{t=0}^{n-1} I(t) \cos\left[\frac{(2t+1)i\pi}{2n}\right] \quad (1)$$

$$I(t) = \frac{\sqrt{2}}{n} \sum_{j=0}^{n-1} C(j)\, T_{dct}(j) \cos\left[\frac{(2t+1)j\pi}{2n}\right] \quad (2)$$

where

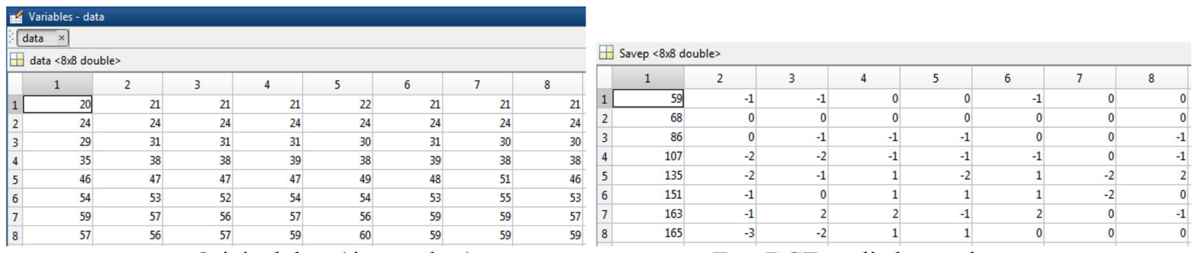
$$C(i) = \begin{cases} = 2^{-1/2}, & if\ i = 0 \\ = 1, & if\ i > 0 \end{cases}$$

where $i = 0, 1, 2, 3, \ldots, n-1$ represents images row size from "$\boldsymbol{I}$", and the output is a set of DCT coefficients "$\boldsymbol{T_{dct}}$". The first coefficient is called DC coefficient, and the rest are referred to as the AC coefficients. Notice that the coefficients are real numbers, and they are rounded off to integers. The important feature of the DCT is that it is useful in image compression [16]. It takes correlated input data and concentrates its energy in just the first few transform coefficients. If the input data consists of correlated quantities, then most of the "$n$" transform coefficients produced by the DCT are zeros or small numbers [17], and only a few are large (normally the first data). The early coefficients contain the most important (low-frequency) image information and the later coefficients contain the less-important (high-frequency) image information [6, 18]. This feature allows good compression performance as a proportion of the less important coefficient scan be discarded without much degradation of image quality. Figure 2 shows the DCT applied to each row of a small image size $8 \times 8$ without using scalar quantization.

## 3 One Dimensional Discrete Sine Transform (DST)

Our research has indicated that a one dimensional DCT works together with a one-dimensional DST yielding large amounts of high-frequency components. These high frequency components are useful to obtain high compression ratios comparable to the JPEG technique. In this research, we will apply one dimensional DST to each column of the transformed matrix "$\mathbf{T_{dct}}$" from previous section. The DST definition is represented as follows [17, 19]:

$$T_{dst}(k) = \sum_{i=1}^{n} T_{dct}(i)\ \mathrm{Sin}\left(\pi \frac{k*i}{n+1}\right) \quad (3)$$



Variables - data

data <8x8 double>

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 20 | 21 | 21 | 21 | 22 | 21 | 21 | 21 |
| 2 | 24 | 24 | 24 | 24 | 24 | 24 | 24 | 24 |
| 3 | 29 | 31 | 31 | 31 | 30 | 31 | 30 | 30 |
| 4 | 35 | 38 | 38 | 39 | 38 | 39 | 38 | 38 |
| 5 | 46 | 47 | 47 | 47 | 49 | 48 | 51 | 46 |
| 6 | 54 | 53 | 52 | 54 | 54 | 53 | 55 | 53 |
| 7 | 59 | 57 | 56 | 57 | 56 | 59 | 59 | 57 |
| 8 | 57 | 56 | 57 | 59 | 60 | 59 | 59 | 59 |

Original data ( image data)

Savep <8x8 double>

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 59 | -1 | -1 | 0 | 0 | -1 | 0 | 0 |
| 2 | 68 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 86 | 0 | -1 | -1 | -1 | 0 | 0 | -1 |
| 4 | 107 | -2 | -2 | -1 | -1 | -1 | 0 | -1 |
| 5 | 135 | -2 | -1 | 1 | -2 | 1 | -2 | 2 |
| 6 | 151 | -1 | 0 | 1 | 1 | 1 | -2 | 0 |
| 7 | 163 | -1 | 2 | 2 | -1 | 2 | 0 | -1 |
| 8 | 165 | -3 | -2 | 1 | 1 | 0 | 0 | 0 |

$\boldsymbol{T_{dct}}$ : DCT applied to each row

(Coefficients are rounded off to integers)

**Fig. 2** *Left* Original block of data, *right* $\boldsymbol{T_{dct}}$ produced by applying one-dimensional DCT to each row independently

$K = 1,\, 2, \ldots N$

$$T_{dct}(i) = \frac{2}{N+1} \sum_{k=1}^{n} T_{dst}(k)\ \mathrm{Sin}\left(\pi \frac{k * i}{n+1}\right) \qquad (4)$$

Equation (3) is used to transform "*n*" values of "$\boldsymbol{T_{dct}}$" matrix into "*n*" coefficients. These are the low and high frequency coefficients containing important and less important image information. The one-dimensional DST is applied to each column of "$\boldsymbol{T_{dct}}$" to produce a new transformed matrix "$\boldsymbol{T_{dst}}$". The DST is equivalent to the imaginary part of the Discrete Fourier Transformation (DFT) and the results of the DST are real numbers [20, 21]. The main advantage of using the DST for image compression in this context is that the DST preservers the image quality encoded by the low frequency components of "$\boldsymbol{T_{dct}}$" and increases the number of zeros, which can be discarded without loss of quality.

After the DST, we apply a quantization of the high frequency components of the transformed matrix "$\boldsymbol{T_{dst}}$". In this way, the quantization means losing only insignificant information from the matrix. Each coefficient in the matrix is divided by the corresponding number from a "*Quantization table*" and the result is rounded off to the nearest integer. The following equation is proposed as a quantization table.

$$Q(i,j) = (i + j) * F \qquad (5)$$

*where: $F > 0$ and i, $j = 1,2,3,\ldots,$ n $\times$ m (image dimensions).*

In Eq. (5) "*F*" is a real number greater than zero. This value affects image quality as for "$F > 1$" image quality is decreased. There is no limit for *F*, however, from our experiments we suggest *F* from 0.1 to 10. Figure 3 shows the DST applied to each column and quantized by Eq. (5).

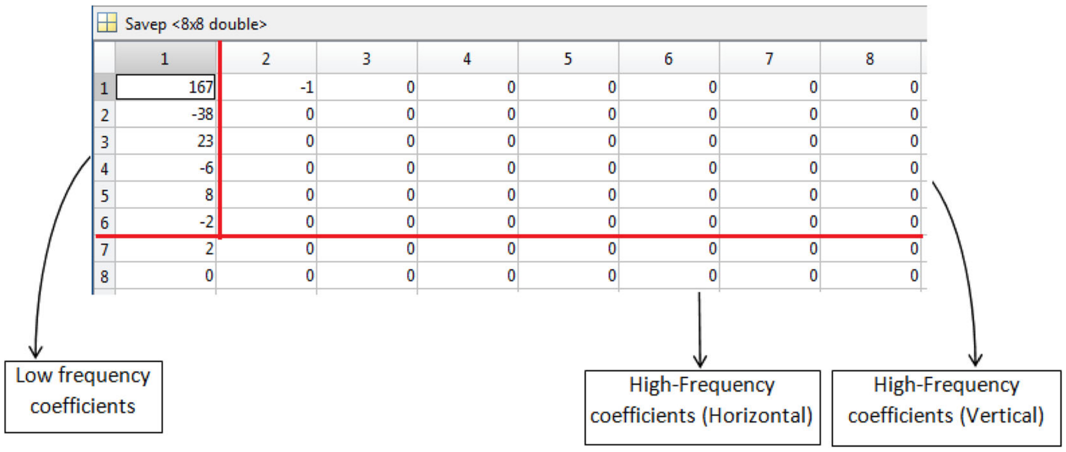
In the above example, low and high frequency components are determined by the user. The low-frequency ones are not compressed any further, we just represent them in fewer bytes by arithmetic coding. Meanwhile, the high-frequency components either horizontal or vertical are compressed by the High-Frequency Minimization algorithm described in the next section.

## 4 High Frequency Minimization Algorithm

In this section, we describe an algorithm to convert the high-frequency coefficients (i.e. from previous section results passed to Minimization algorithm) into a compressed array called *Minimized-Array* through a matrix minimization method involving eliminating zeros and triplet encoding whose output is then subjected to arithmetic coding. Normally, the high frequency components contain large numbers of zeroes with a few nonzero data. The technique eliminates zeroes and enhances compression ratio [11, 13, 14, 16, 18].

The high-frequency minimization algorithm is applied further reducing the size of high-frequency sub-matrix by 2/3. This process hinges on defining three key values and multiplying these by three adjacent entries in $\boldsymbol{H}$ (the matrix of high frequency coefficients) which are then summed over producing single integer values as shown in Fig. 4 [13, 14].

Thus, each set of the three entries from $\boldsymbol{H}$ are converted into a single value which are then stored into a new coded array (*Minimized-Array*). Assuming that $N$ is the length of $\boldsymbol{H}$, $i = 1, 2, \ldots, N-3$, and $j$ is the


Savep <8x8 double>

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 167 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | -38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | -6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Low frequency coefficients
High-Frequency coefficients (Horizontal)
High-Frequency coefficients (Vertical)

**Fig. 3** One-dimensional DST applied to each Column independently of $\boldsymbol{T_{dct}}$ followed by quantization with F = 2 (See Eq. (5))

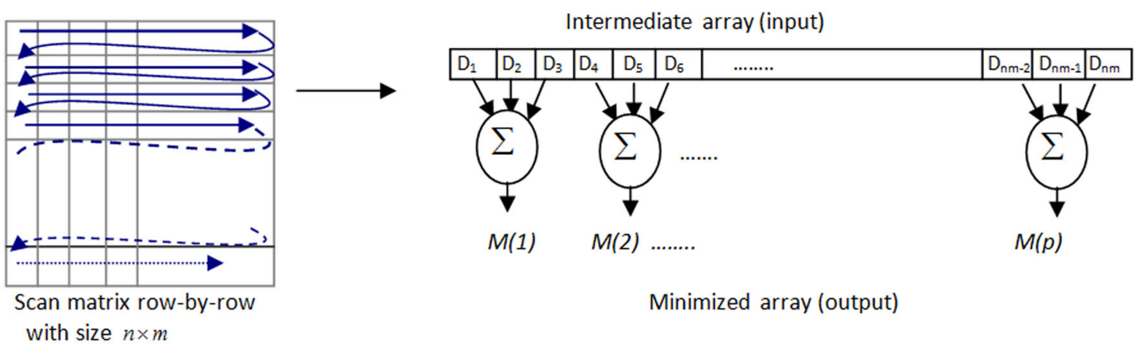

Intermediate array (input)
$D_1$ $D_2$ $D_3$ $D_4$ $D_5$ $D_6$ ........ $D_{nm-2}$ $D_{nm-1}$ $D_{nm}$
$\Sigma$ $\Sigma$ ....... $\Sigma$
M(1) M(2) ........ M(p)
Minimized array (output)
Scan matrix row-by-row with size $n \times m$

**Fig. 4** High-Frequency Minimization Algorithm used to compress coefficients ($D_1$, $D_2$..., $D_{nm}$) from matrix $\boldsymbol{H}$ (i.e. matrix $\boldsymbol{H}$ scanned row-by-row for compression)

index of new coded array, the following transformations define the high frequency encoding [11, 13, 14, 18]:

$$\boldsymbol{Minimized\_Array}_j = K_1\boldsymbol{H}_i + K_2\boldsymbol{H}_{(i+1)} + K_3\boldsymbol{H}_{(i+2)} \tag{4}$$

The key values $K_1$, $K_2$, $K_3$ are generated by a key generator algorithm as in [14, 18] described through Eqs. 5, 6, 7 and 8 below. Because the keys are data-dependent of max($\boldsymbol{H}$), each matrix will have their unique set of keys if their max($\boldsymbol{H}$) are distinct.

$$M = 1.5\max(H) \tag{5}$$

$$K_1 = 1 \tag{6}$$

$$K_2 = K_1 + M + Factor \tag{7}$$

$$K_3 = Factor * M(K_1 + K_2) \tag{8}$$

where $Factor \geq 1$ and $K_1 \geq 1$ are integer values. The quantity *Factor* is a scaling factor to enlarge the degree of separation between the 3 generated keys. The keys themselves are the weights of each triplet summation in the minimized-array. The original values of each triplet can later be recovered by estimating the $\boldsymbol{H}$ values (See Sect. 5) for that ***Minimized-Array***. Following the models above, the Minimized-Array for the example in Fig. 3 can be illustrated in the following Table 1.

Our compression method creates a new array of header data $\boldsymbol{H}$, which is used later by the decompression algorithm to estimate the original data values. This information is kept in the header of the compressed file as a string. Figure 5 below illustrates the concept through a numerical example.

Per above example in Fig. 3, the Limited-Data can be estimated from high-frequency sub-matrices

**Table 1** From the example of Fig. 3: each high-frequency sub-matrix is compressed independently

| High-frequency sub-matrix | Compressed size | Comments |
|---|---|---|
| Minimized-array$_{(\text{Vertical})}$ = {−1, 0, 0, 0, … 0} | Compressed size 16 | 48 (original size)/3 = 16 data |
| Minimized-array$_{(\text{Horizontal})}$ = {2, 0, 0, 0, 0, 0} | Compressed size 6 | 16 (original size)/3 = 5.3 (last zero is alone) |

Assume M = 2 (Maximum value in high-frequency sub-matrix: Horizontal), the Keys values will be: $K_1 = 1$, $K_2 = 5$, $K_3 = 18$ for both high-frequencies components: Horizontal and Vertical

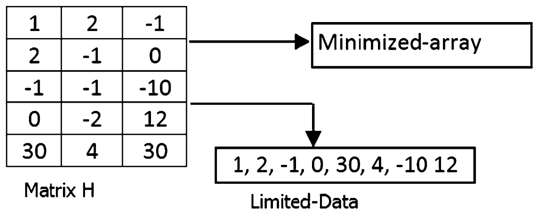


**Fig. 5** Limited-Data appearing in ***H*** are kept in the header file for recovery

(Horizontal and Vertical). Limited-Data$_{(\text{Vertical})}$ = {−1, 0} and Limited-Data$_{(\text{Horizontal})}$ = {2, 0}.

The encoded triplets in the *Minimized-Array* may contain large number of zeros which can be further encoded through a process proposed by [14]. For example, assume the following encoded *Minimized-Array* = {125, 0, 0, 0, 73, 0, 0, 0, 0, 0, −17}.The zero array will be {0, 3, 0, 5, 0} where the zeros in red refer to nonzero data existing at these positions and the numbers in black refer to the number of zeros between two consecutive non-zero data. To increase the compression ratio, the number 5 can be broken up into 3 and 2 to increase data redundancy. Thus, the equivalent zero array would be {0, 3, 0, 3, 2, 0} and the nonzero array would be {125, 73, −17}. According to this method, the *Minimized-Array* both Horizontal and Vertical can be illustrated in Table 2.

The final step of compression is arithmetic coding which computes the probability of all data and assigns a range to each data (low and high) to generate streams of compressed bits [5].The arithmetic coding applied here takes a stream of data and converts into a single floating point value. The output is in the range between zero and one that, when decoded, returns the exact original stream of data.

## 5 The Fast-Matching Search Decompression Algorithm

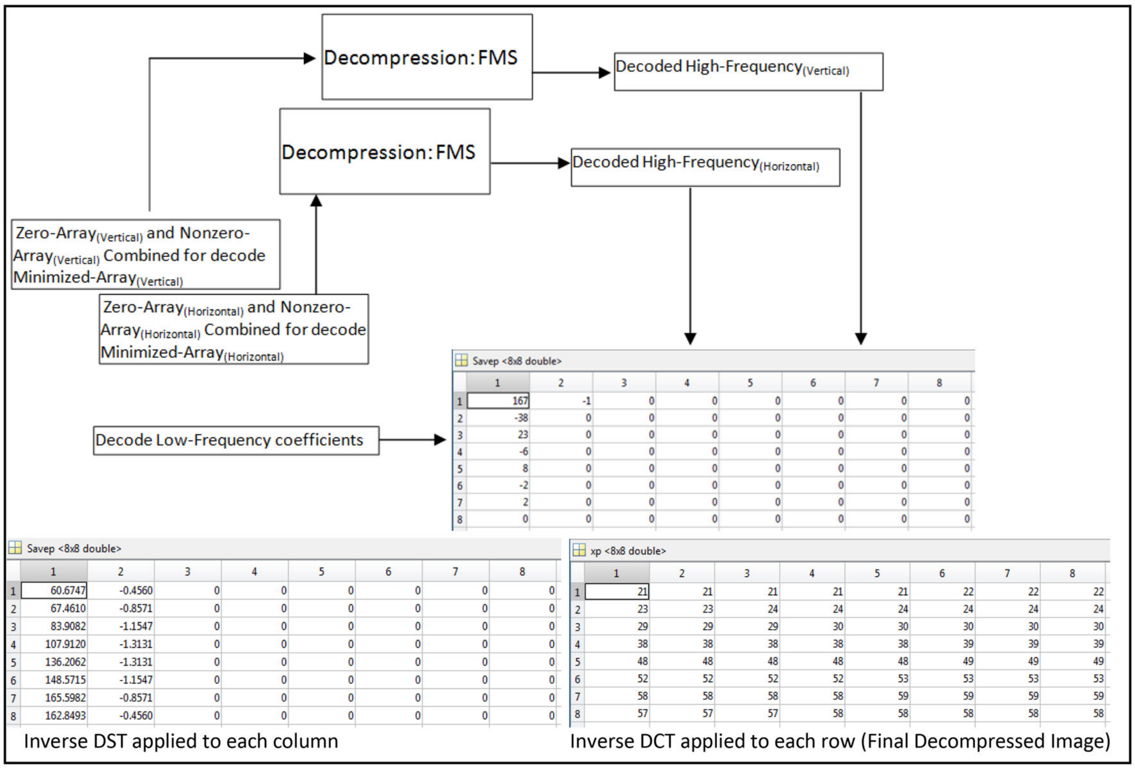
The decompression algorithm is the inverse of compression. First, decode the Minimized-Array for both horizontal and vertical components by combining the zero-array with the non-zero-array. Second, decode high-frequencies from the Minimized-Array using the fast matching search (FMS) algorithm [14]. Third, inverse the DST and DCT to reconstruct the original 2D image. The images are then assessed on their perceptual quality and on their ability to reconstruct the 3D structures compared with the original images. Figure 6 illustrates the decompression method.

The *Fast Matching Search Algorithm (FMS)* has been designed to recover the original high frequency data. The compressed data contains information about the compression keys ($K_1$, $K_2$ and $K_3$) and Limited-Data followed by streams of compressed high frequency data. Therefore, the FMS algorithm picks up each compressed high frequency data and decodes it using the key values and compares whether the result is expressed in the Limited-Data. Given 3 possible

**Table 2** Each *Minimized-Array* is coded to zero-array and nonzero-array

| High-frequency sub-matrix | Zero-array | Nonzero-array |
|---|---|---|
| Minimized-array$_{(\text{Vertical})}$ = {−1,0, 0, 0 … 0} | Zero$_{(\text{Vertical})}$ = {**0**, 5, 5, 5} | Nonzero-array$_{(\text{Vertical})}$ = {−1} |
| Minimized-array$_{(\text{Horizontal})}$ = {2, 0, 0, 0, 0, 0} | Zero$_{(\text{Horizontal})}$ = {**0**, 5} | Nonzero-Array$_{(\text{Horizontal})}$ = {2} |

The “0” refers to the nonzero data in Nonzero-Arrays


Decompression: FMS
Decoded High-Frequency(Vertical)
Decompression: FMS
Decoded High-Frequency(Horizontal)
Zero-Array(Vertical) and Nonzero-Array(Vertical) Combined for decode Minimized-Array(Vertical)
Zero-Array(Horizontal) and Nonzero-Array(Horizontal) Combined for decode Minimized-Array(Horizontal)
Decode Low-Frequency coefficients
Inverse DST applied to each column
Inverse DCT applied to each row (Final Decompressed Image)

Savep <8x8 double>

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 167 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | -38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | -6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | -2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Savep <8x8 double>

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 60.6747 | -0.4560 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 67.4610 | -0.8571 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 83.9082 | -1.1547 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 107.9120 | -1.3131 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 136.2062 | -1.3131 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 148.5715 | -1.1547 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 165.5982 | -0.8571 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 162.8493 | -0.4560 | 0 | 0 | 0 | 0 | 0 | 0 |

xp <8x8 double>

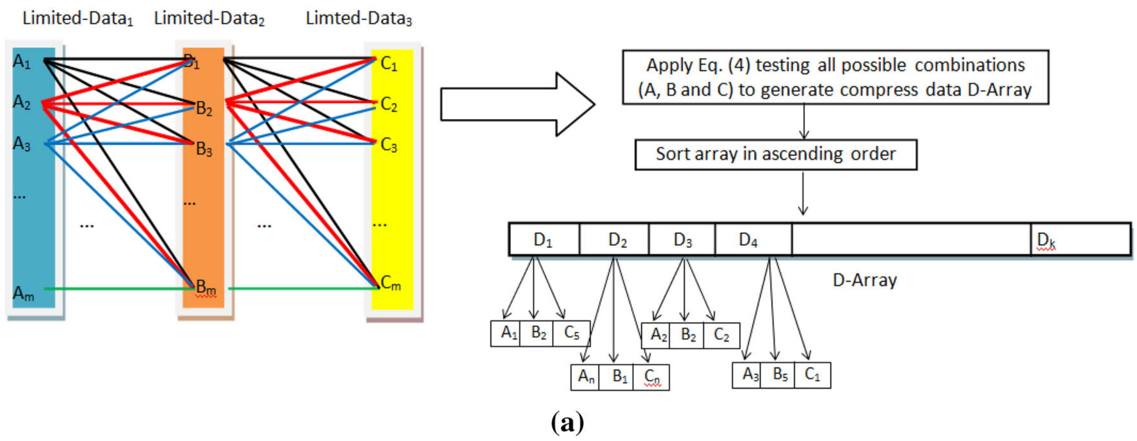
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 21 | 21 | 21 | 21 | 21 | 22 | 22 | 22 |
| 2 | 23 | 23 | 24 | 24 | 24 | 24 | 24 | 24 |
| 3 | 29 | 29 | 29 | 30 | 30 | 30 | 30 | 30 |
| 4 | 38 | 38 | 38 | 38 | 38 | 39 | 39 | 39 |
| 5 | 48 | 48 | 48 | 48 | 48 | 49 | 49 | 49 |
| 6 | 52 | 52 | 52 | 52 | 53 | 53 | 53 | 53 |
| 7 | 58 | 58 | 58 | 58 | 59 | 59 | 59 | 59 |
| 8 | 57 | 57 | 57 | 58 | 58 | 58 | 58 | 58 |

**Fig. 6** The steps in the decompression algorithm

values from Limited Data, there is only one possible correct result for each key combination, so the data is uniquely decoded. We illustrate the FMS-Algorithm through the following steps A and B [14]:

(A) Initially, the Limited-Data is copied into three separated arrays given that we used three keys for compression. The algorithm picks three items of data (one from each Limited-Data) and apply these to Eq. (4) using the three compression keys. The method resembles an interconnected array ***D***, where each value is combined with each other value, similar to a network as shown in Fig. 7a.
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 $K_1$, B with $K_2$ and C with $K_3$ that yield an intermediate array 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. (4) these represent H(L), H(L + 1) and H(L + 2) respectively. The equation is executed 27 times ($3\text{-}^{3} = 27$) testing all possibilities. One of these combinations in array D will match the value in the compressed data. The match indicates that the unique combination of A, B and C are the original data. If we apply this to our example, Limited-$Data_{(vertical)}$ = {−1, 0} (See Table 1) number of possibilities for A = [−1, 0], B = [−1, 0] and C = [−1, 0] as shown in Table 3.

(B) The searching algorithm used in the decompression method is called Binary Search Algorithm. It finds the original data (A, B, C) for any input from compressed data file "Minimized-Array". For binary search, the D-Array should be arranged in ascending order.
The decompression algorithm compares a value from the Minimized-Array 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 (i.e. the relevant A, B and C are the decompressed data we are after) [22].


Limited-Data1
Limited-Data2
Limted-Data3
A1
A2
A3
Am
B1
B2
B3
Bm
C1
C2
C3
Cm
Apply Eq. (4) testing all possible combinations (A, B and C) to generate compress data D-Array
Sort array in ascending order
D1
D2
D3
D4
Dk
D-Array
A1 B2 C5
An B1 Cn
A2 B2 C2
A3 B5 C1
(a)

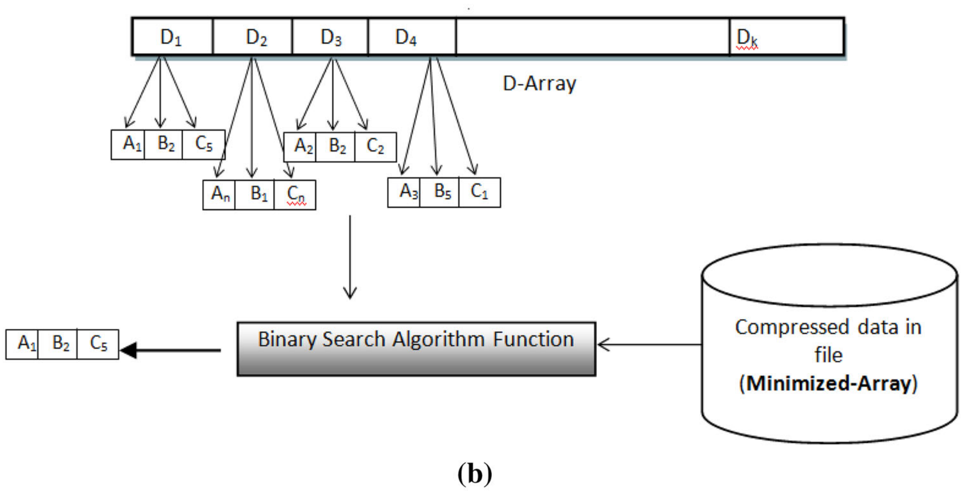

D1
D2
D3
D4
Dk
D-Array
A1 B2 C5
An B1 Cn
A2 B2 C2
A3 B5 C1
Binary Search Algorithm Function
Compressed data in file (Minimized-Array)
A1 B2 C5
(b)

**Fig. 7** **a**, **b** FMS-Algorithm for reconstructing high frequency data from Limited-Data. A, B and C are the original data which are determined by the unique combination of keys. **a** Estimate all possible compressed data saved in D-array (i.e. each possible compressed data connected with their relevant original data). **b** Decompression by using Binary Searching algorithm

**Table 3** All possible data computed according to Eq. (4) to generate the D-Array ($K_1 = 1$, $K_2 = 3$ and $K_3 = 4$)

| Limited-data$_1$ (A) | Limited-data$_2$ (B) | Limited-data$_3$ (C) | D-array |
|---|---|---|---|
| −1 | −1 | −1 | 8 |
| −1 | −1 | 0 | −4 |
| −1 | 0 | −1 | −5 |
| −1 | 0 | 0 | −1 |
| 0 | −1 | −1 | −7 |
| 0 | −1 | 0 | −3 |
| 0 | 0 | −1 | −4 |
| 0 | 0 | 0 | 0 |

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 of “Not Matched”, because the FMS-Algorithm computes all compression data possibilities as shown in Fig. 7b.

Once the horizontal and vertical high frequency components are recovered by the FMS-Algorithm, they are combined to regenerate the 2D matrix (See Fig. 6). Then each data from the matrix is multiplied by each data in $\boldsymbol{Q}$ (Eq. 5) followed by the inverse DST (Eq. 4) applied to each column. Finally, we multiply each data by $\boldsymbol{F}$ followed by the inverse DCT (Eq. 2) applied to each row to recover the original 2D image as shown in Fig. 6. If we compare the results in Fig. 6 with the original $8 \times 8$ matrix of Fig. 2, we find that there is not much difference, and these differences do not affect image quality. For this reason, our technique is very attractive for image compression.

## 6 Experimental Results

The experimental results described here were implemented in MATLAB R2013a and Visual C++ 2008 running on an AMD Quad-Core microprocessor. We describe the results in two parts: first, we apply the method to general 2D images of different sizes and assess their perceived visual quality and RMSE. Additionally, we compare our compression method with JPEG and JPEG2000 through the visualization of 2D images, 3D surface reconstruction from multiple views and RMSE error measures.

Second, we apply the compression and decompression algorithms to 2D images that contain structured light patterns allowing 3D surface data to be generated from those patterns. The rationale is that a high-quality image compression is required otherwise the resulting 3D structure from the decompressed image will contain apparent dissimilarities when compared to the 3D structure obtained from the original (uncompressed) data. We report on these differences in 3D through visualization and standard measures of RMSE-root mean square error.

### 6.1 Results for 2D Images

In this Section, we apply the algorithms to generic 2D images, that is, images that do not contain structured light patterns as described in the previous section. In this case, the quality of the compression is performed by perceptual assessment and by the RMSE measure. We use images with varying sizes from 2.25 to 9 MB. Also, we present a comparison with JPEG and JPEG2000 highlighting the differences in compressed image sizes and the perceived quality of the compression.

Figure 8a gives an indication of compression ratios achieved with our approach while in (b) is shown details with comparative analysis with JPEG2000 and JPEG. First, the decoded 'baby' image by JPEG2000 contains some blurring at places, while the same image decoded by our approach and JPEG are of higher quality. Second, the decoded 'eyes' image by JPEG algorithm had some block artefacts resulting in a lower quality compression. Also, the same image decoded by our approach and JPEG2000 at equivalent compression ratios, has excellent image quality. Finally, the decoded 'girl' image by JPEG2000 is slightly degraded, while our approach and JPEG show good image quality.

Additionally, we applied our compression techniques to a series of 2D images and used **Autodesk 123DCatch** software to generate a 3D model from multiple images. The objective is to perform a direct comparison between our approach and both JPEG and JPEG2000 on the ability to perform 3D reconstruction from multiple views. Images are uploaded to the Autodesk server for processing which normally takes a few minutes. The 123D Catch software uses photogrammetric techniques to measure distances between objects producing a 3D model (i.e. image processing is performed by stitching a plain seam with correct sides together). The application may ask the user to select common points on the seam that could not be determined automatically [15, 23]. Compression sizes and RMSE for all images used are depicted in Table 4.

Figure 9 shows two series of 2D images for objects "APPLE", and "FACE" (all images are available from 123D Catch website). We start by compressing each series of images whose compressed sizes and 2D RMSE measures are shown in Table 4. A direct comparison of compression with JPEG and JPEG2000 is presented in Table 5. It is clearly shown that our approach and JPEG2000 can reach an equivalent compression ratio, while the JPEG technique does not. It is important to stress that both our technique and JPEG depend on DCT. The main difference is that our approach is based on DCT with DST and the coefficients are compressed by the frequency

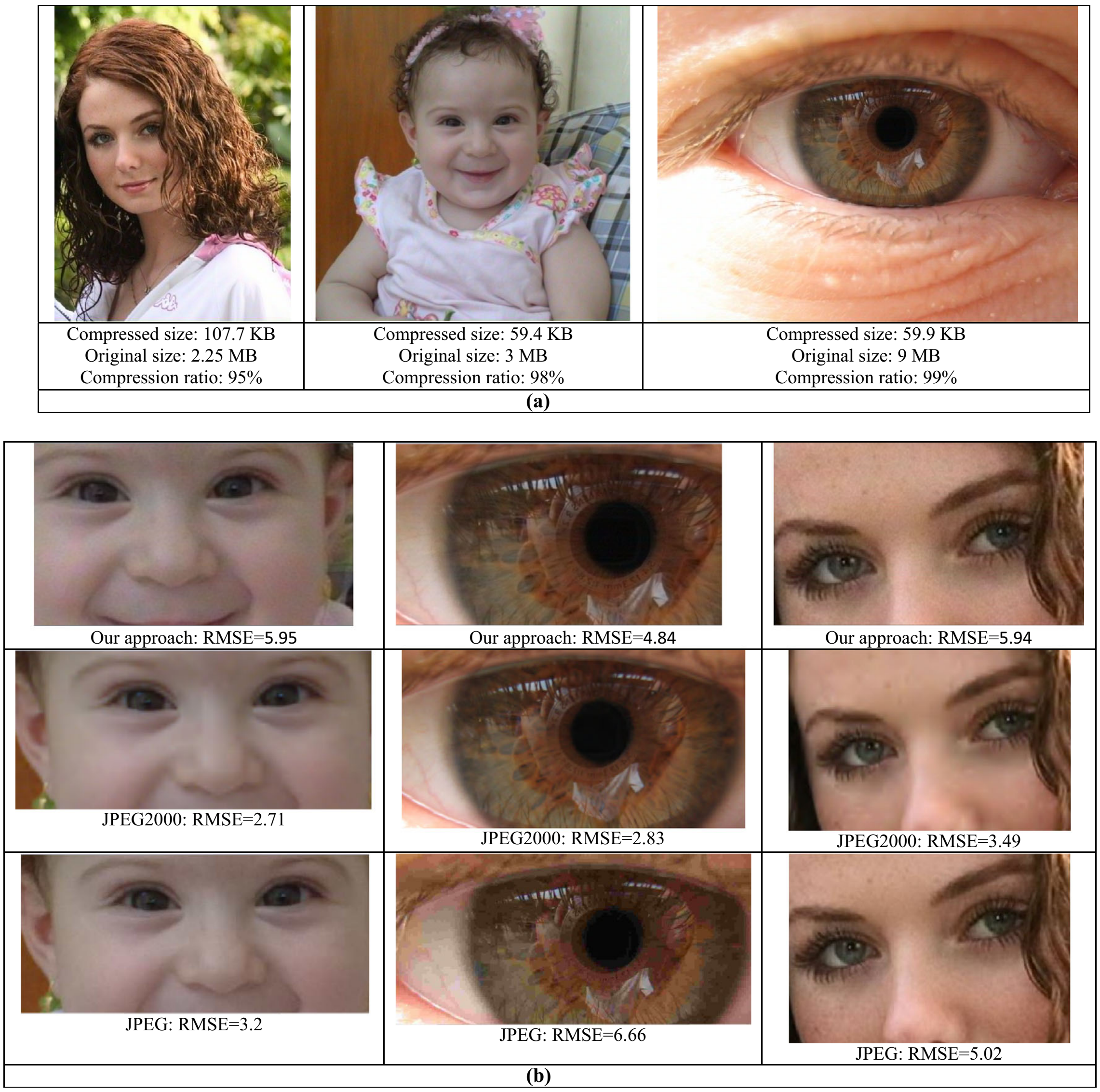


**Fig. 8** Compressed images by JPEG and JPEG2000 at equivalent compressed file sizes as with our approach. **a** Compressed and decompressed 2D images by our approach. **b** Details of compression/decompression by our approach, JPEG2000 and JPEG respectively

minimization algorithm, which renders our technique far superior to JPEG as shown in the comparative analysis of Fig. 10.

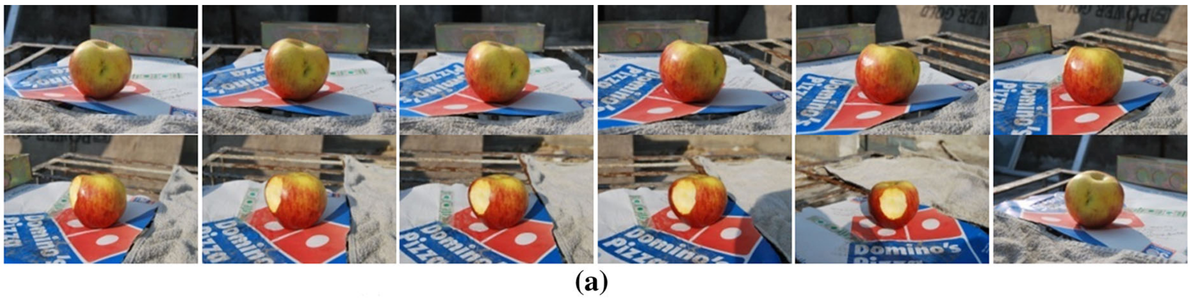
In our method, DCT with DST are applied on an image as one block. The used low frequency block size for colour was 150 $\times$ 150, the scalar quantization for DCT was 1, 5 and 5 for each layer (Y, Cb and Cr) respectively. Furthermore, the quantization matrix used after DST performs an aggressive quantization, this means that approximately 50% of the coefficients are zeros (i.e. the left bottom of the image matrix contains lot of zeros after the quantization process—see Eq. 5).

## 6.2 Results for Structured Light Images and 3D Surfaces

3D surface reconstruction was performed with our own software developed within the GMPR group [9, 10, 12]. The justification for introducing 3D

**Table 4** Compressed sizes and 2D RMSE measures

| Image name | Number of images | Original image size (MB) | Quantization parameters used in DST | | | Compressed image size (MB) | Average compressed size of each image (MB) | Average 2D RMSE |
|---|---|---|---|---|---|---|---|---|
| | | | Y | Cb | Cr | | | |
| Baby | 1 | 3 | 0.5 | 5 | 5 | 0.0594 | 0.0594 | 5.95 |
| Eyes | 1 | 9 | 0.5 | 5 | 5 | 0.0599 | 0.0599 | 4.84 |
| Girl | 1 | 2.25 | 0.5 | 5 | 5 | 0.1077 | 0.1077 | 5.94 |
| Apple | 48 | 336 | 2 | 5 | 5 | 1.94 | 0.0414 | 8.33 |
| Face | 28 | 200.7 | 1 | 5 | 5 | 1.72 | 0.0629 | 5.68 |



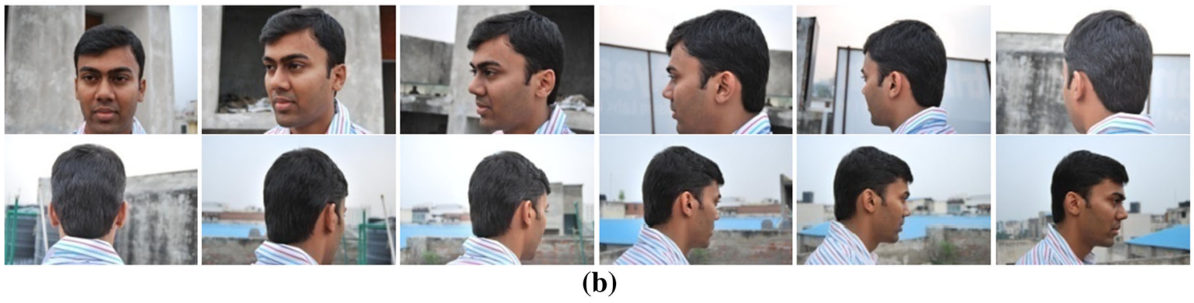
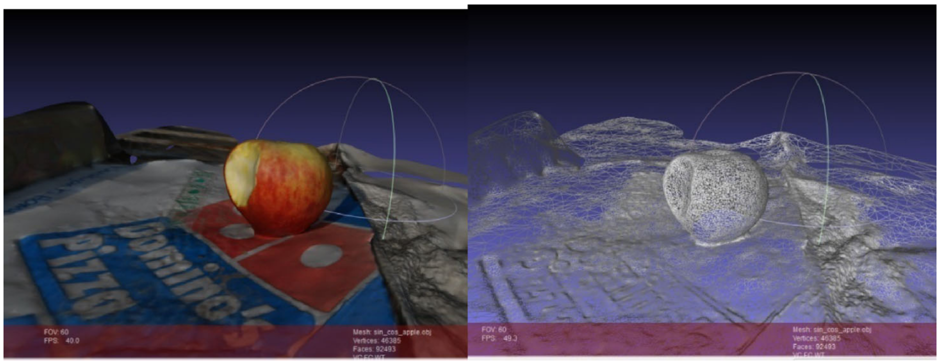


**Fig. 9** **a**, **b** show series of 2D images used to generate 3D models by 123D Catch. **a** *Apple images* Number of image 48 images. **b** *Face images* Number of images 28 images

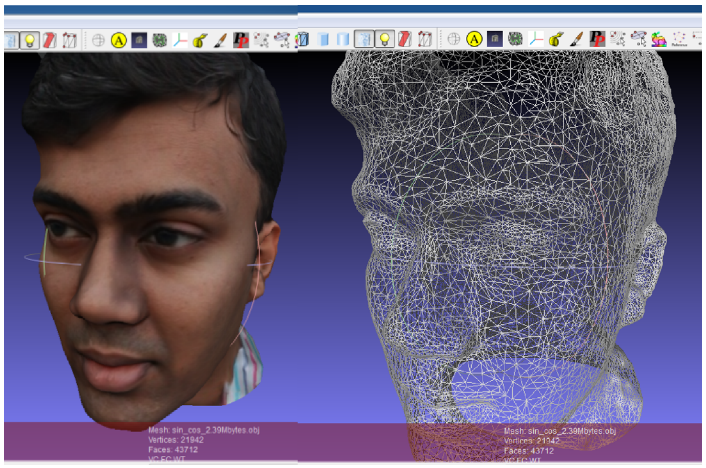
**Table 5** Comparison of 3D reconstruction for images compressed to the same size. Note that JPEG failed to reconstruct the 3D structure as the images were too deteriorated

| Multiple 2D images | Original size (MB) | Compressed size (MB) | 2D RMSE | | |
|---|---|---|---|---|---|
| | | | Our approach | JPEG2000 | JPEG |
| APPLE | 336 | 1.94 | 9.5 | 6.58 | Fail |
| FACE | 200.7 | 1.72 | 5.1 | 3.39 | Fail |

reconstruction is that we can make use of a new set of metrics in terms of error measures and perceived quality of the 3D visualization to assess the quality of the compression/decompression algorithms. The principle of operation of GMPR 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



(a)



(b)

**Fig. 10** **a**, **b** Successful 3D reconstruction following compression by our approach. Images were compressed to the same size by our approach, JPEG and JPEG2000. **a** 3D model for series of APPLE images decompressed by our approach (48 images, average 2D RMSE = 8.33, total compressed size = 1.94 MB). The compression ratio for the 3D mesh is 99.4% for connectivity and vertices. **b** 3D model for series of FACE images decompressed by our approach (28 images, average 2D RMSE = 5.68, total compressed size = 1.72 MB). The compression ratio for the 3D mesh is 99.1% for connectivity and vertices

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 [24]. The scanner is depicted in Fig. 11.

Figure 12 shows several test images used to generate 3D surfaces both in grayscale and colour. The top row shows two grayscale face images, FACE1 and FACE2 with size 1.37 MB and dimensions $1392 \times 1040$ pixels. The bottom row shows colour images CORNER and METAL with size 3.75 MB and dimension $1280 \times 1024$ pixels. We use the RMSE measure to compute the differences between decompressed images and original ones. The RMSE however, cannot give an absolute indication of which is the 'best' reconstructed image or 3D surface, as errors may be concentrated in a region that may or may not be relevant to the perception of quality. To get a better assessment of quality, we analyse 3D surface images at various compression ratios.

Table 6 shows the compressed size for our approach using two different values of quantization. First, the quantization scalar for FACE1 and FACE2 is 1. This means that after DCT each coefficient is divided by 1, this means rounding off each floating-point value to integer. Similarly, after DST the quantization equation is applied with $F$ (Eq. 5).

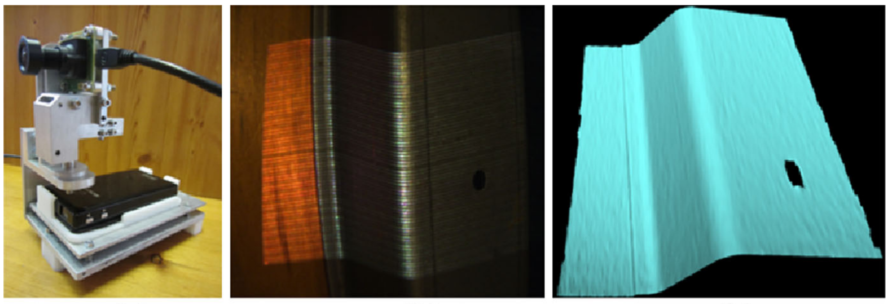


**Fig. 11** **a** depicts the GMPR scanner together with an image captured by the camera (**b**) which is then converted into a 3D surface and visualized (**c**). Note that only the portions of the image that contain patterns (*stripes*) can be converted into 3D; other parts of the image are ignored by the 3D reconstruction algorithms

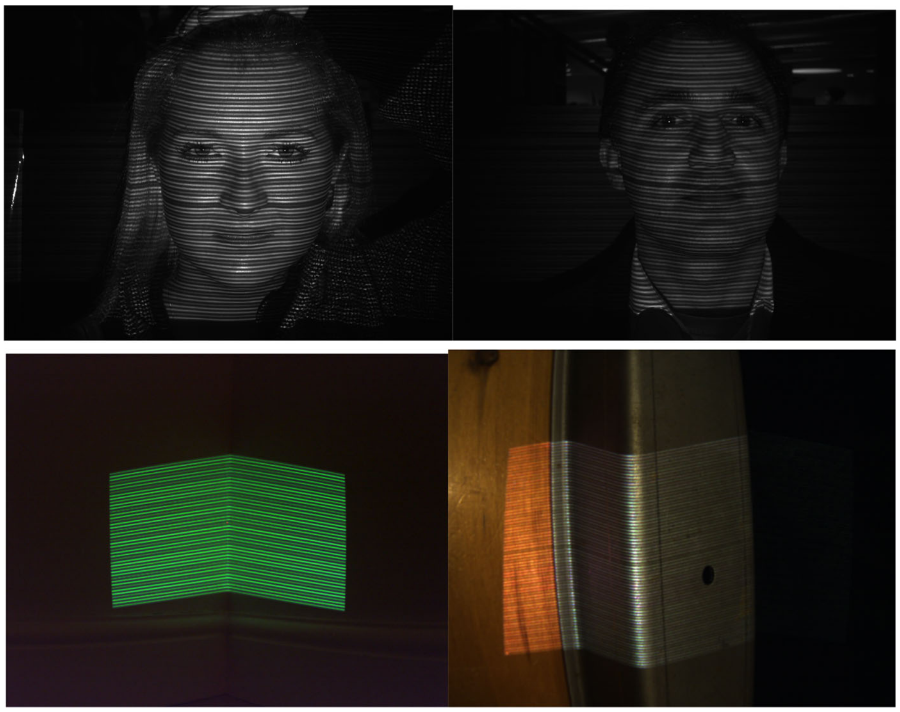


**Fig. 12** Structured light images used to generate 3D surfaces. *Top row* grayscale images FACE1 and FACE2, and colour images CORNER and METAL respectively. Images were compressed to the same size by our approach, JPEG and JPEG2000

The colour images are defined by using colour transformation [5, 25] into YCbCr format. We then apply the proposed approach to each layer independently. For this reason, after DCT the quantization scalar for colour images is $\{1, 5, 5\}$ for each layer of **Y, Cb** and **Cr** respectively.

Figure 13 shows the visualization of the decompressed 2D images using different values for quantization. These decompressed images are converted to 3D surfaces. FACE1 on top of Fig. 13 from left to right are higher quality surface per 3D RMSE. In fact, some parts of 3D surface have disappeared at higher compression ratio. But in FACE2 in the middle, the 3D reconstructed image at higher compression ratio is approximately the same as for low compression ratio. This means that 3D reconstruction depends on the structured light's quality in an image. Figure 13 (bottom) shows zoomed-in regions for the two images, the structure light patterns are clearly present at 99% compression ratio.

**Table 6** Structured light images compressed by our approach

| Image name | Original image size (MB) | Original image size | | Compressed size (KB) | 2D RMSE | 3D RMSE |
|---|---|---|---|---|---|---|
| | | DCT | DST | | | |
| FACE1 | 1.37 | 1 | 2 | 18.75 | 4.82 | 1.51 |
| | | 1 | 6 | 11.7 | 6.22 | 1.54 |
| FACE2 | 1.37 | 1 | 2 | 15.6 | 1.89 | 2.25 |
| | | 1 | 6 | 7.8 | 2.56 | 2.67 |
| CORNER | 3.75 | {1, 5, 5} | {2, 2, 2} | 21.2 | 5.56 | 1.36 |
| | | {1, 5, 5} | {2, 3, 3} | 14.7 | 7.0 | 0.5 |
| METAL | 3.75 | {1, 5, 5} | {1, 5, 5} | 27.5 | 5.25 | 1.87 |
| | | {1, 5, 5} | {2, 5, 5} | 12.1 | 5.62 | 1.98 |

Figure 14 shows 3D reconstructed surfaces for CORNER and METAL images respectively. On top, the quality of CORNER 3D surface at 99% compression ratio. But the 3D surface (top right) has some artefacts; this type of artefacts not present in the original and decompressed 2D image at lower compression ratio. Artefacts appear when the structure light patterns are not clearly defined in the image, or are degraded after compression and decompression. In Fig. 14 middle, the decompressed METAL image is converted to a 3D surface. The reconstructed 3D surface of middle right is degraded for all cases in which compression ratios exceed 99%.To analyse 2D colour image compression, we zoomed-in the decompressed 2D images. It is shown that the structure lights are clearly visible at higher compression ratios of 99%.

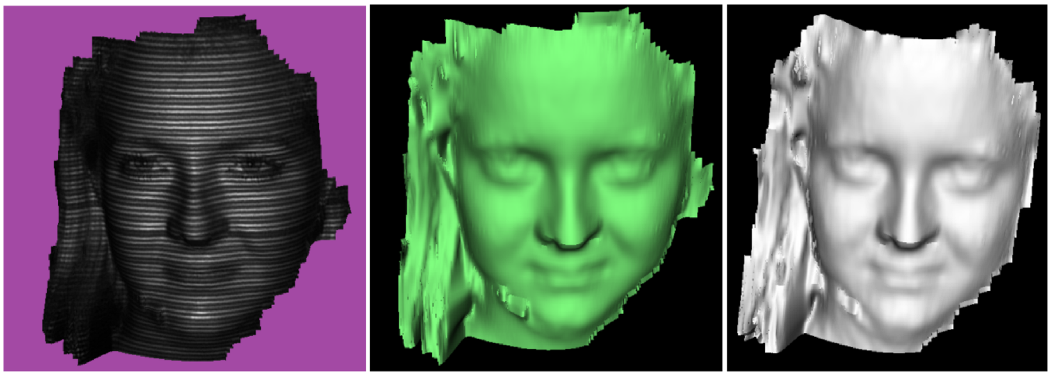
For a comparative analysis, we compressed and decompressed the 2D images by JPEG2000 and JPEG, then converted to a 3D surface. Table 7 and Fig. 15 describe the compressed and decompressed results respectively for JPEG2000. The comparison is based on applying the same compression ratios between JPEG2000 and our approach and show the visualization for the two methods. While the JPEG algorithm simply failed to compress the images at the required ratio. Also, the important point to note is that JPEG2000 [26] cannot decompress some 2D images to equivalent quality for 3D reconstruction or if it does, the 3D surface contains some degradation. Additionally, Fig. 16 shows the compressed 2D images by JPEG2000 with zoomed in image details.

## 7 Conclusions

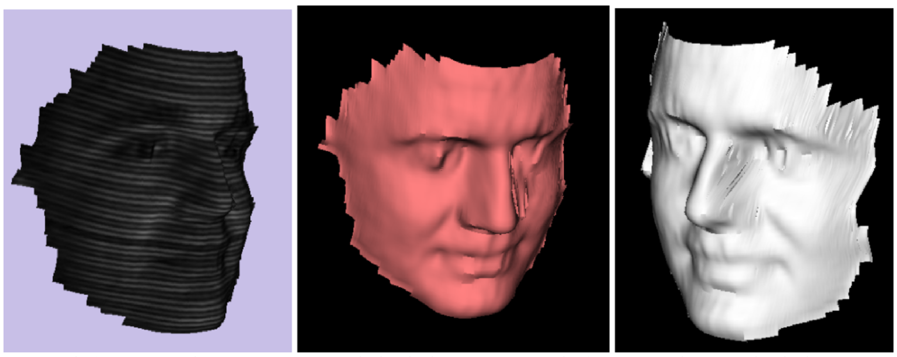
This paper has presented and demonstrated a new method for image compression and illustrated the quality of compression through 2D and 3D reconstruction, 2D and 3D RMSE. Our compression algorithm is based on DCT applied to each row of an image, then followed by DST which is applied to each column of the matrix. After the transformation stage, the minimization of high frequency algorithm is used to reduce the number of high-frequency coefficients. The compression stage is then completed with arithmetic coding. In the decoding stage, the Fast-Matching-Search algorithm based on binary search is used to recover the original data. The results show that our approach introduces better image quality at higher compression ratios than JPEG and JPEG2000 as it can more accurately reconstruct 3D surfaces than both techniques. A slight disadvantage of the proposed method is that it is more complex than both JPEG2000 and JPEG. This is because our approach uses two types of transforms, and that neither JPEG nor JPEG2000 rely on a search method.

The most important aspects of the method and their role in providing high quality image with high compression ratios are identified as follows:

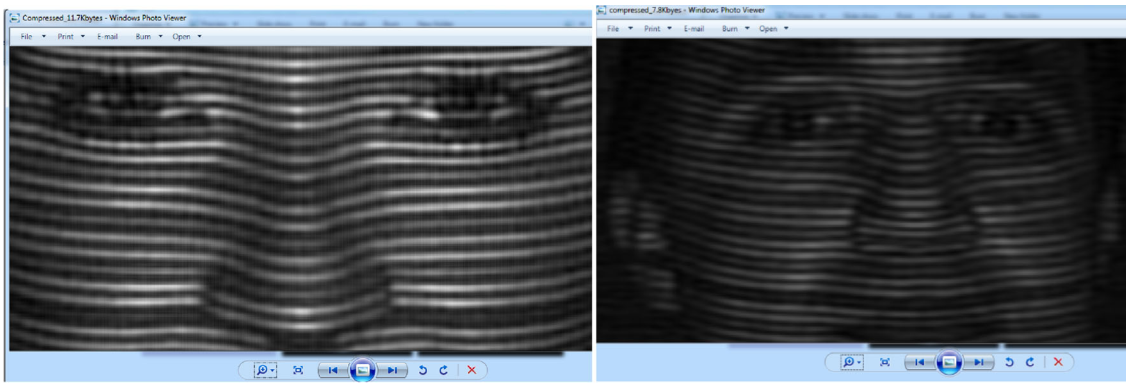
1. The one-dimensional DCT can be applied to an image row (i.e. larger block sizes $\geq 8$). Equally, the one-dimensional DST can be applied to each column of the output from DCT.
2. The user can ignore the scalar quantization to remove higher frequency coefficients (i.e.



FACE1: Compressed size 18.75 Kbytes (texture and shaded) Compressed Size=11.7Kbytes (shaded)
3D reconstructed FACE1 from decompressed image by our approach



FACE2: Compressed size 15.6 Kbytes (texture and shaded) Compressed Size=7.8 Kbytes (shaded)
3D reconstructed FACE2 from decompressed image by our approach


Compressed_11.7Kbytes - Windows Photo Viewer
File Print E-mail Burn Open
compressed_7.8Kbytes - Windows Photo Viewer
File Print E-mail Burn Open

2D decompressed images zoomed-in, to show the details: FACE1 and FACE2 at higher compression ratio

**Fig. 13** *Top* FACE1 shows decompressed 3D surface with texture and shaded at compressed size 18.7 and 11.7 KB. Middle: FACE2 shows decompressed 3D surface with texture and shaded at compressed size 15.6 and 7.8 KB. *Bottom* details of 2D images FACE1 and FACE2 respectively at the higher compression ratio

keeping more coefficients increases image quality).

3. The two-dimensional quantization (cf. Eq. 5) provides a more aggressive quantization removing most of matrix contents as about 50% of the matrix entries are zero. Applying this over the DST can keep image quality at higher compression ratios.

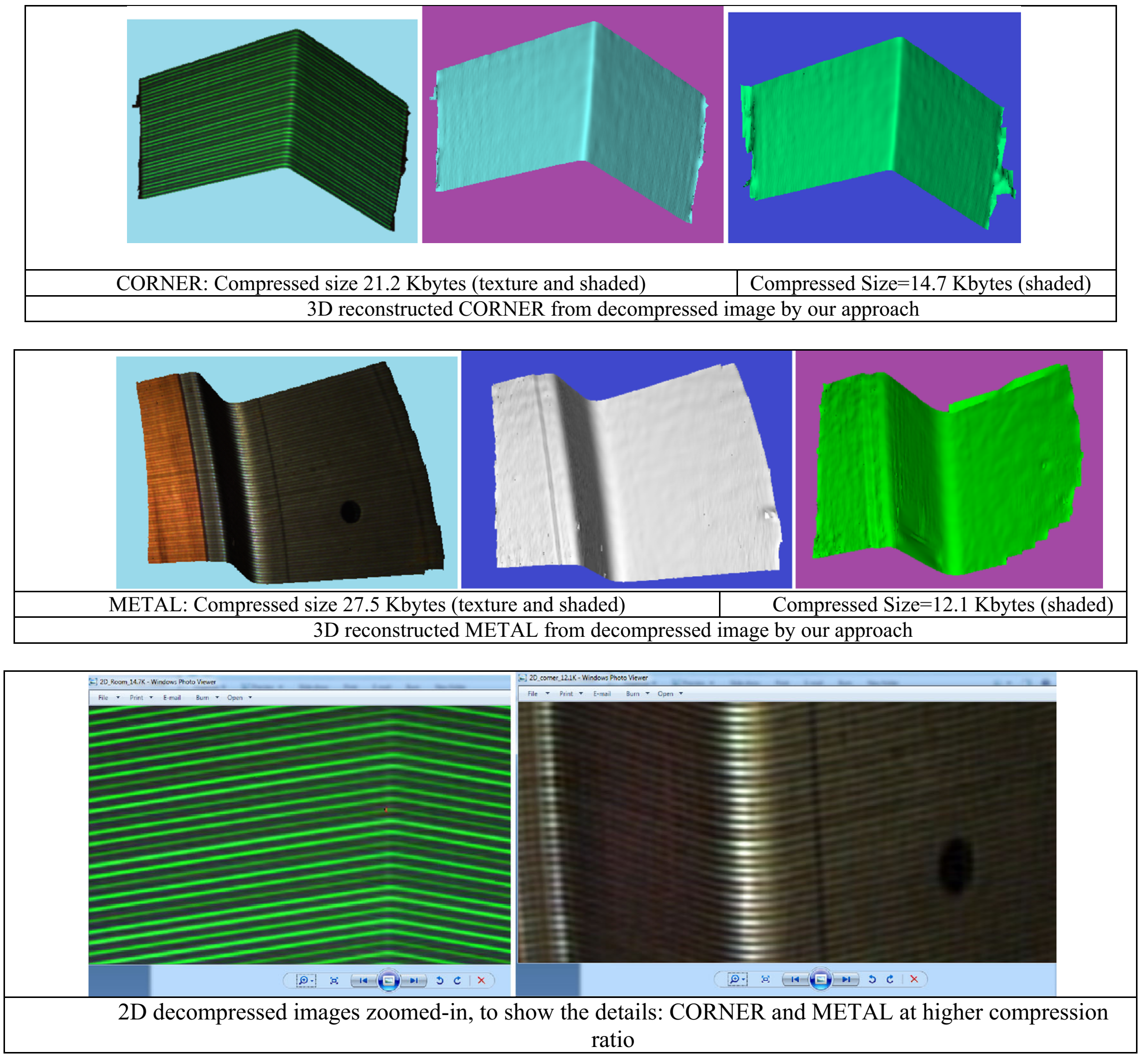

CORNER: Compressed size 21.2 Kbytes (texture and shaded)
Compressed Size=14.7 Kbytes (shaded)
3D reconstructed CORNER from decompressed image by our approach
METAL: Compressed size 27.5 Kbytes (texture and shaded)
Compressed Size=12.1 Kbytes (shaded)
3D reconstructed METAL from decompressed image by our approach
2D decompressed images zoomed-in, to show the details: CORNER and METAL at higher compression ratio

**Fig. 14** *Top* shows decompressed 3D surface of CORNER with texture and shaded at compressed sizes 21.2 and 14.7 KB. *Middle* shows decompressed 3D surface of METAL with *texture and shaded* at compressed sizes 27.5 and 12.1 KB. *Bottom* zoomed-in details for 2D images CORNER and METAL respectively at higher compression ratio

**Table 7** Compression and decompression for 3D reconstruction by JPEG2000 and JPEG at higher compression ratios. All images were compressed to the same size by the techniques

| Image name | Compressed size (KB) | JPEG2000 | | JPEG | |
|---|---|---|---|---|---|
| | | 2D RMSE | 3D RMSE | 2D RMSE | 3D RMSE |
| FACE1 | 11.7 | 6.3 | 1.8 | FAIL | FAIL |
| FACE2 | 7.8 | 3.2 | 2.66 | FAIL | FAIL |
| CORNER | 14.7 | 5.7 | 0.63 | FAIL | FAIL |
| METAL | 13.4 | 4.17 | FAIL | FAIL | FAIL |

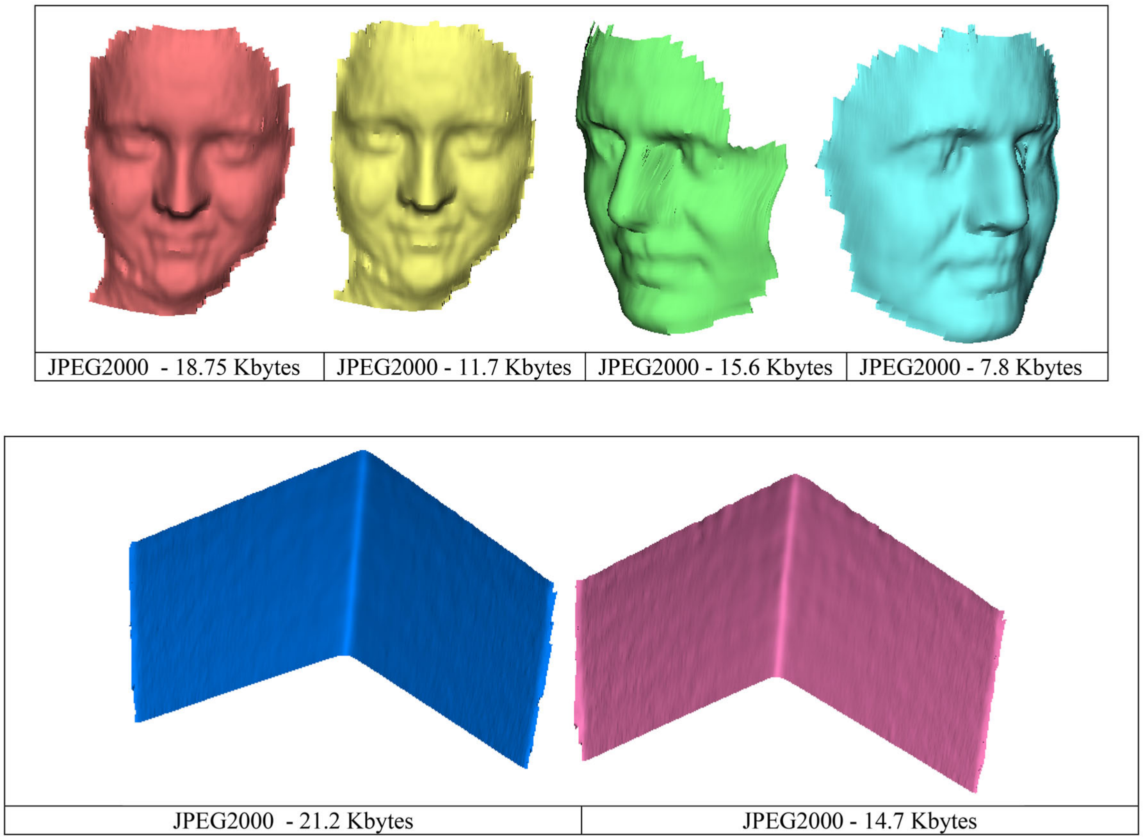


**Fig. 15** *Top* 3D reconstructed surface for FACE1 and FACE2 respectively using JPEG2000. *Bottom* CORNER image successfully 3D reconstructed, while METAL image failed 3D reconstruction

4. The final transformed matrix is divided into: low-frequency sub-matrix, and horizontal and vertical high-frequency matrices.
5. The minimization of high frequency algorithm produces a Minimized-Array used to replace each three values from the high-frequencies sub-bands by a single integer value. This process reduces the coefficients by 2/3 leading to increased compression ratios.
6. Since the Minimized-Array for both vertical and horizontal high-frequencies contains large number of zeros, we applied a new method to eliminate zeros and keep nonzero data. The process keeps significant information while reducing data up to 80%.
7. At decompression stage, the Fast-Matching-Search algorithm is the engine for estimating the original data from the minimized array and depends on the organized key values and the availability of a set of unique data. The efficient C++ implementation allows this algorithm to recover the high-frequency matrices very efficiently.
8. The key values and unique data are used for coding and decoding an image, without this information images cannot be recovered. This is an important point as a compressed image is equivalent to an encrypted image that can only be reconstructed if the keys are available. This has applications to secure transmission and storage of images and video data.
9. Our proposed image compression algorithm was tested on true colour and YCbCr layered images at high compression ratios. Additionally, the approach was tested on images resulting in better 3D reconstruction than JPEG2000 and JPEG.
10. The experiments indicate that the technique can be used for real-time applications such as

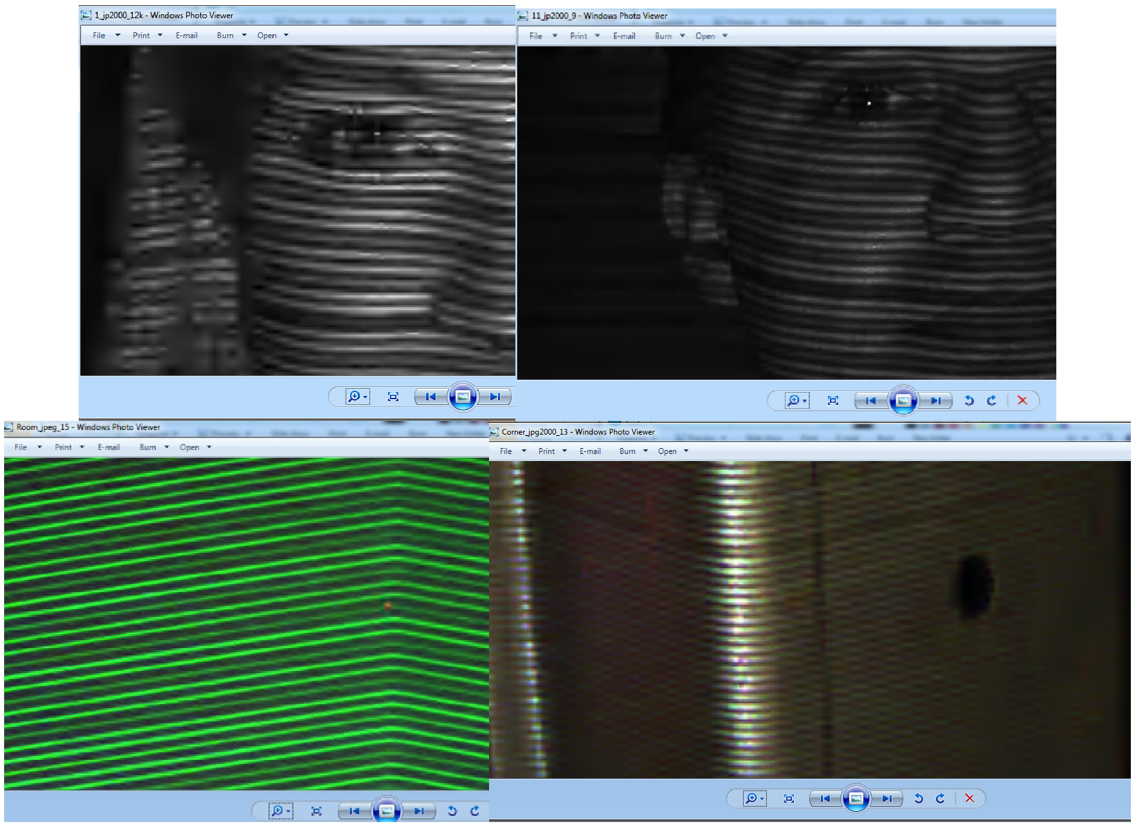

1_jp2000_12k - Windows Photo Viewer
File Print E-mail Burn Open
11_jp2000_9 - Windows Photo Viewer
File Print E-mail Burn Open
Room_jpeg_15 - Windows Photo Viewer
File Print E-mail Burn Open
Corner_jpg2000_13 - Windows Photo Viewer
File Print E-mail Burn Open

**Fig. 16** Details of 2D decompressed images by JPEG2000: *Top* FACE1 on the left is clearly blurred leading to degraded 3D reconstruction. *Bottom* METAL image on the right is blurred rendering it unable to reconstruct a 3D surface

3D data objects and video data streaming over the Internet.

Future work is focused on efficient implementation of the decoding steps and their application to video compression. Research is underway and will be reported soon.

## References

1. Al-Haj, A. (2007). Combined DWT-DCT digital image watermarking, *Science Publications*. *Journal of Computer Science, 3*(9), 740–746.
2. 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.
3. Martucci, S. A. (1994). Symmetric convolution and the discrete sine and cosine transforms. *IEEE Transactions on Signal Processing, SP-42*(5), 1038–1051.
4. Richardson, I. E. G. (2002). *Video codec design*. New York: John Wiley & Sons.
5. Sayood, K. (2000). *Introduction to data compression* (2nd ed.). New York, Los Altos: Academic Press, Morgan Kaufman Publishers.
6. Pennebaker, W. B., & Mitchell, J. L. (1993). *JPEG: Still image data compression standard*. New York: Van Nostrand Reinhold.
7. Kekre, H. B., Sarode, T., & Natu, P. (2013). Efficient image compression technique using full column and row transforms on colour image. *International Journal of Advances in Engineering and Technology, 6*(1), 88–100.
8. Rodrigues, M., Robinson, A., & Osman, A. (2010). Efficient 3D data compression through parameterization of free-form surface patches, In *Signal Process and Multimedia Applications (SIGMAP), Proceedings of the 2010 International Conference on* IEEE, (pp. 130–135).
9. Rodrigues, M., Osman, A., & Robinson, A. (2013). Partial differential equations for 3D data compression and reconstruction. *Journal Advances in Dynamical Systems and Applications, 12*(3), 371–378.
10. Rodrigues, M., Kormann, M., Schuhler, C., & Tomek, P. (2013b). Robot trajectory planning using OLP and structured light 3D machine vision. *Lecture notes in Computer*

*Science Part II. LCNS, 8034 (8034)*. (pp. 244–253), Heidelberg: Springer.

11. Siddeq, M. M., & Rodrigues, M.A. (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).
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). Springer: Heidelberg.
13. Siddeq, M. M., & Rodrigues, M. A. (2014). *A novel image compression algorithm for high resolution 3D reconstruction, 3D research*. Berlin: Springer. doi:10.1007/s13319-014-0007-6.
14. Siddeq, M. M., & Rodrigues, M. (2015). 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. *3D Research, 6*(3), 26. doi:10.1007/s13319-015-0055-6.
15. D Catch. http://www.123dapp.com/howto/catch. Accessed May 2016.
16. 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).
17. Pelaes, E. G., & Lano, Y. (1998). Image coding using discrete sine transform with axis rotation. *IEEE Transactions on Consumer Electronics, 44*(4), 1284–1290.
18. Siddeq, M. M., & Rodrigues, M. (2015a). Applied sequential-search algorithm for compression-encryption of high-resolution structured light 3D data. In: K. Blashki, & Y. Xiao (Eds.), *MCCSIS: Multi-conference on computer science and information systems 2015*. IADIS Press, (pp. 195–202).
19. Discrete Sine Transform. (2016). https://en.wikipedia.org/wiki/Discrete_sine_transform. Accessed Nov 2016.
20. Dhamija, S., & Jain, P. (2011). Comparative Analysis for Discrete Sine Transform as a suitable method for noise estimation. *IJCSI International Journal of Computer Science Issues, 8*(5, 3).
21. Sasikumar, M., & Moni, R. S (2014). Use of Discrete Sine Transform for a novel image denoising technique. *International Journal of Image Processing (IJIP), 8*(4).
22. Knuth, D. (1997). *Sorting and searching*: Section 6.2.1: Searching an ordered table, the art of computer programming 3 (3rd ed.), Vol 1, Fundemental Algorithms. Publisher: Addison-Wesley. pp. 409–426. ISBN 0-201-89685-0.
23. Autodesk 123D. https://en.wikipedia.org/wiki/Autodesk_123D.Accessed May 2016.
24. 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).
25. Gonzalez, R. C., & Woods, R. E. (2001). *Digital image processing*. Reading: Addison Wesley Publishing Company.
26. Acharya, T., & Tsai, P. S. (2005). *JPEG2000 standard for image compression: Concepts, algorithms and VLSI architectures*. New York: John Wiley & Sons.