Wireless Pers Commun (2017) 97:4773–4787
DOI 10.1007/s11277-017-4750-x

# Image Compression Using Two Dimensional DCT and Least Squares Interpolation

Sameh A. Eisa[1] · Waleed Lotfy[2]

Published online: 5 August 2017

**Abstract** This paper introduces a new image compression method utilizing a combination of discrete cosine transform and least squares interpolation method. Presented is a discussion of the mathematical background, outline of the approach, complexity computations, pseudocode, and an explanation of how to implement the algorithm for applications that require the coded bits to be binary streams. We then provide the results, including comparisons to many recently published works. The results indicate positive progress and effectiveness of the new approach in terms of comparability to other works and applicability in real time applications.

**Keywords** Image compression · Signal processing · LSM interpolation · DCT

## 1 Introduction

Technological and research advances demand ongoing developments in image processing. Over the past two decades, advances in signal processing and communications have allowed for rapid and wide growth within the field of image processing. Most of the applications stemming from image processing research have triggered researchers to explore, invent, develop, and apply mathematical techniques to audio, image, and video processing.

In particular, our focus is upon interpolation techniques. Within the literature there are numerous papers such as [1, 6, 8, 9, 15–17, 19] that utilize interpolation methods. In [1] Edge Assisted Wavelet Based Interpolation (EAWI) was used for preserved image compression. In [6] an adaptive interpolation algorithm for fast video encoding in H.264 was

✉ Sameh A. Eisa
sameh.eisa@student.nmt.edu; sameheisa235@hotmail.com

Waleed Lotfy
waleed.lutfi90@gmail.com

1 Department of Mathematics, New Mexico Tech, Socorro, NM, USA

2 Alexandria, Egypt

introduced. Papers [8, 9] contain Lagrange Interpolation as a method of error correction coding for images and frame interpolation method respectively. In [15] the author proposed another image compression method that uses interpolation. An algorithm based on interpolation for curve length calculation was proposed within [16] for use with images. In [19] Cubic Spline Interpolation was used as a Coding/Decoding (CODEC) technique for images, while in [17], audio CODECs were introduced by the author based on Least Squares Interpolation Method (LSM), which is extended within this paper as an image compression method. Other trials have been found in the literature that utilize mathematical techniques directly or indirectly to approach new paths for image compression. Some of these trials involve Wavelet methods such as [2, 3, 10, 12] or others such as [4, 5, 7, 11, 13, 14, 18, 20, 21].

This paper introduces an image compression method that depends on two dimensional Discrete Cosine Transform (DCT) and one dimensional LSM. While the researcher in [17] was able to demonstrate that LSM was sufficient to approach audio CODECs that resulted in competitive compression ratios, this paper extends the LSM by merging it with DCT in order to have image compression ratios that are of a high quality. The results of this paper are compared to the results published in [1, 2, 12, 13, 15, 19, 20]. When compared to the published works of the last decade, the results show a promising amount of progress. These comparisons are discussed within the paper, especially the comparison with interpolation-based techniques as laid out in the results section.

## 2 The Methodology

In this section, both the mathematical background and the outline to utilize it in our application for image compression are explained.

### 2.1 The Mathematical Background

Two main methods are used in our approach. The first one is DCT in two dimensional space. The second one is LSM in one dimensional space.

For a given image of dimension $N \times M$ pixels, DCT can be applied to map the image (square ones) into the frequency domain. The mapping uses the formula (see [22] for more details about the formula and implementation),

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{1}$$

where,

$$\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{N}} & \text{for } u = 0 \\ \sqrt{\dfrac{2}{N}} & \text{for } u \neq 0 \end{cases}$$

The function $f(x,y)$ stands for the pixel given integer values. After applying DCT, the function $C(u, v)$ stands for the corresponding mapped values of the pixels. The pair $(x, y)$ locates any pixel value in the time domain, whereas the pair $(u, v)$ is to locate the mapped value of the pixel in the frequency domain. The inverse transform is given by the formula (see [23] for more details about the formula and implementation),

$$f(x,y) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{2}$$

Applying DCT to image processing has been the basis of many standard techniques in the literature, especially for compression methods. Many of the references we cite used DCT. Additionally, many other image CODECs methods use it, such as the JPEG based algorithms. After applying DCT, we can remove blocks of the resultant matrix that represent unnecessary frequencies. In our approach, we keep only a small block of the resultant matrix in the upper left corner that represents the most important frequencies of the image. This block is some ratio of the whole matrix size. Determining the size of the block is a part of the algorithm we use and is a factor in determining the targeted compression ratio.

For a given row of data $y$, LSM can be applied to generate a polynomial of degree $n$ that fits (approximate) the row data with least squares error, such that,

$$y_i \cong a_0 + a_1 x_i + a_2 x_i^2 \ldots a_n x_i^n$$

We note that $y$ is a vector (row data of pixels' values of length $N_s$). We set the values of $x$ to be any chosen row data such as 1, 2, 3… $N_s$. We then find $a_0, a_1, a_2 \ldots a_n$ by solving a system of linear equations. Dimension of $y$ is greater than or equals $n + 1$. The system represented in a matrix form is as follows,

$$Y = XA \tag{3}$$

$$A_{(n+1)\times 1} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \quad Y_{(n+1)\times 1} = \begin{bmatrix} \sum_{i=1}^{N_s} y_i \\ \sum_{i=1}^{N_s} x_i y_i \\ \sum_{i=1}^{N_s} x_i^2 y_i \\ \vdots \\ \sum_{i=1}^{N_s} x_i^n y_i \end{bmatrix}$$

$$\text{and} \quad X_{(n+1)\times(n+1)} = \begin{bmatrix} N_s & \sum_{i=1}^{N_s} x_i & \sum_{i=1}^{N_s} x_i^2 & \cdots & \sum_{i=1}^{N_s} x_i^n \\ \sum_{i=1}^{N_s} x_i & \sum_{i=1}^{N_s} x_i^2 & \sum_{i=1}^{N_s} x_i^3 & \cdots & \sum_{i=1}^{N_s} x_i^{n+1} \\ \sum_{i=1}^{N_s} x_i^2 & \sum_{i=1}^{N_s} x_i^3 & \sum_{i=1}^{N_s} x_i^4 & \cdots & \sum_{i=1}^{N_s} x_i^{n+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N_s} x_i^n & \sum_{i=1}^{N_s} x_i^{n+1} & \sum_{i=1}^{N_s} x_i^{n+2} & \cdots & \sum_{i=1}^{N_s} x_i^{2n} \end{bmatrix} \tag{4}$$

then,

$$A = X^{-1}Y \tag{5}$$

In order to judge the modeling, we determine the value $r$ (coefficient of determination) which clarifies the fitting of the proposed polynomial corresponding to the row data values (pixels' values),

$$r = \sqrt{\frac{\sum_{i=1}^{N_s} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N_s} (y_i - \bar{y})^2}} \tag{6}$$

where $\hat{y}_i$ is the theoretical $y$-value corresponding to $x_i$ (calculated through the polynomial) and $\bar{y}$ is the mean value of all experimental $y$-values. Whenever $r\%$ tends to 100, then no errors are present. This value helps us to judge whether or not the proposed polynomial represents the pixel data efficiently.

## 2.2 Outline of the Approach

Although we are considering gray scale images for this approach, it can also be extended to colored images. Figure 1 shows the nine images we are testing in this paper. Each image has been numbered to identify it throughout the paper. In this paper, Peak Signal to Noise Ratio (PSNR) formula is given by,

$$PSNR(\text{db}) = 10\log_{10}\left[\frac{255^2}{MSE}\right] \tag{7}$$

where,

$$MSE = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (x_i - y_i)^2$$

and it is used to indicate quality of the results as mentioned in the cited papers. In our approach, there are two sources of compression. One source comes from LSM $(CR_{LSM}(bpp))$ and the second from DCT $(CR_{DCT}(bpp))$. So the final compression ratio for $N \times M$ image is given by,

$$CR(bpp) = \frac{number\ of\ codded\ bits}{N \times M} = CR_{LSM} \times CR_{DCT}$$

In the next section, a pseudocode is given to outline the implementation of the approach. Our algorithm applies DCT to a given image of size $N \times M$. A block of $N_1 \times M_1$ in the

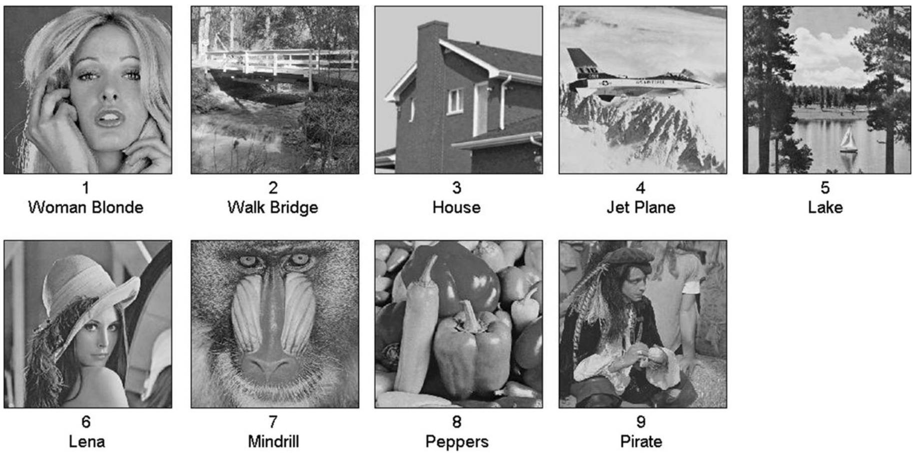

**Fig. 1** Images to be used in this paper with names and given indices

upper left corner of the resultant matrix is cut, such that $N_1 < N$ and $M_1 < M$. We then apply inverse DCT to the $N_1 \times M_1$ block, resulting in another $N_1 \times M_1$ block in the time domain. For a specified $N_s$ (number of samples) in every row in this block LSM is applied. The coefficients of the polynomials are quantized to be saved/transmitted as binary streams. The size of this matrix is $N_2 \times M_2$. In order to have a $CR_{LSM}$ we need $n + 1 < N_s$.

We can notice that $CR_{DCT} = \frac{N_1 \times M_1}{N \times M}$, $CR_{LSM} = \frac{N_2 \times M_2}{N_1 \times M_1}$, and then the final $CR = CR_{LSM} \times CR_{DCT} = \frac{N_2 \times M_2}{N \times M}$. For the given number of bits that represent every pixel ($bpp$), we compute $CR(bpp) = CR \times bpp$. For the tested images in this paper $bpp = 8$.

In order to recover the image, the coding process is reversed. We use the quantized coefficients of the polynomials to produce the rows of $N_1 \times M_1$. Then DCT is applied to this matrix. We add zeros to the $N_1 \times M_1$ matrix in the frequency domain, so the size becomes $N \times M$. The image is then recovered by applying inverse DCT to this Matrix.

## 3 Discussion and Explanation of the Algorithm

### 3.1 Pseudocode

```
Compression Coding:
    Input: Image of size $N \times M$, $CR_{DCT}$,$CR_{LSM}$, $n$, bpp.
    Step 1: Read image $N \times M$ as an array of pixels (integer values)
    Step 2: Apply 2-dim DCT on the image using Eq (1)
    Step 3: Discard insignificant frequencies and use $CR_{DCT}$ to overwrite the original image size by $N_1 \times M_1$.
    Step 4: Apply inverse 2-dim DCT on image Eq (2).
    Step 5: Use $CR_{LSM}$ and $n$ to calculate $N_s$.
    Step 6: For every row in the image and for every $N_s$ in every row, apply $n^{th}$ order LSM.
    Step 7: Save the coefficients of LSM in $N_2 \times M_2$ array such that $\{a_0\}$ and $\{a_1\}$ are in different columns.
    Step 8: Convert floating point values to integers and overwrite $\{a_0\}$ and $\{a_1\}$.
    Step 9: Shift the most frequent values of first coefficients $\{a_0\}$ and second coefficients $\{a_1\}$ to zeros respectively and save the offset.
    Step 10: Limit first coefficients to bpp-1 bits, second coefficients to bpp + 1 bits, and save the coefficients
    Output: Array $N_2 \times M_2$ of quantized LSM coefficients ready to be represented by binary streams.
    Decompression Coding:
    Input: Array of size $N_2 \times M_2$, offset, the sizes $N_1 \times M_1$ and $N \times M$.
    Step 1: Read the compressed coefficients $N_2 \times M_2$.
    Step 2: Evaluate the polynomial values from the coefficients to recover $N_1 \times M_1$.
    Step 3: Apply 2-dim DCT on $N_1 \times M_1$ image.
    Step 4: Append zeros to the image in frequency domain to extend it up to $N \times M$.
    Step 5: Apply inverse 2-dim DCT on the image.
    Step 6: Apply Gaussian filter to enhance image quality (optional).
    Output: Recovered original image $N \times M$ with the required compression ($CR_{DCT} \times CR_{LSM}$).
```

## 3.2 LSM Needs DCT for Competitive Compression Ratio (CR)

In Eisa [17], LSM was enough to achieve good quality compressed audio. Although applying LSM for images is an original idea and gives high quality results, the CRs are not competitive enough when compared to the most recent published works. Figure 2 shows a processed image using LSM without DCT. The accuracy and the quality of the result are significantly different when compared with the results of applying both LSM and DCT (Fig. 3). Also, to strengthen this argument, we applied LSM with and without DCT to all of our images with different CRs. The result is summarized in Table 1. We notice that there is no single case in which LSM alone performs better, while the converse holds true.

To try if there is a way to improve LSM significantly by itself, we have performed different LSM trials without DCT, such as, two dimensional (surface fitting) LSM. The results are close. They are slightly better for the two dimensional LSM, but with more complexity. For example, Lena was processed under both LSM with CR = 2.67 bpp, the first degree one dimensional PSNR is approximately 31, while the first degree two dimensional is approximately 32, with no observable quality difference. Also, we consider studying LSM with the same CR with different polynomial degrees and the results are suggesting that either there are no differences in the quality or there is even less quality for higher degrees since more polynomial coefficients are distorted by quantization. Table 2 shows a comparison between LSM trials with different degrees. As a consequence of this, we consider utilizing one dimensional LSM with a fixed polynomial degree n = 1 for less complications and easier calculations.

## 3.3 Production of the Binary Streams

The final matrix that is stored or transmitted should encompass an integer value range similar to the pixels in order to allow a transfer to binary streams. The CR will be computed by the new bpp value. Optimally the minimum integer of the quantized

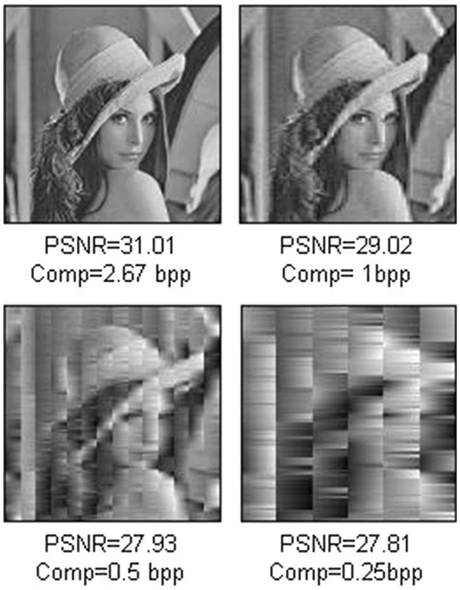

**Fig. 2** Lena (Image 6) PSNR under LSM first degree for CR 2.67, 1, 0.5, 0.25 bpp

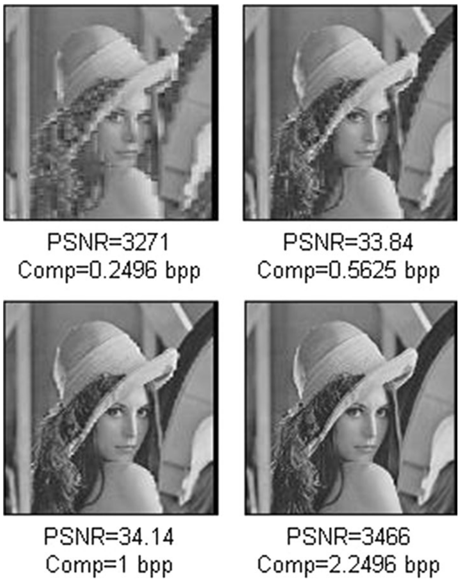

**Fig. 3** Lena (Image 6) with different CRs (bpp) and their PSNR (processed with DCT and LSM)

**Table 1** Experimentation of the 9 images used in this paper with LSM alone and with DCT for different CRs

| | PSNR with only LSM | | | PSNR with LSM and DCT | | |
|---|---|---|---|---|---|---|
| | bpp = 1 | bpp = 0.5 | bpp = 0.25 | bpp = 1 | bpp = 0.5 | bpp = 0.25 |
| Image 1 | 30.95 | 29.14 | 27.96 | 33.13 | 32.65 | 32.68 |
| Image 2 | 29.47 | 28.36 | 27.29 | 30.03 | 28.97 | 27.44 |
| Image 3 | 31.15 | 29.31 | 26.71 | 36.70 | 35.54 | 35.95 |
| Image 4 | 31.10 | 28.69 | 27.48 | 34.47 | 33.63 | 33.41 |
| Image 5 | 30.23 | 28.53 | 27.24 | 33.27 | 33.05 | 33.41 |
| Image 6 | 31.10 | 29.26 | 28.33 | 34.14 | 33.48 | 32.71 |
| Image 7 | 29.28 | 28.35 | 27.41 | 31.00 | 30.18 | 29.20 |
| Image 8 | 31.28 | 29.31 | 27.87 | 33.75 | 31.30 | 28.26 |
| Image 9 | 30.39 | 28.62 | 27.45 | 30.96 | 27.64 | 31.29 |

**Table 2** PSNR results for the nine images after applying LSM with different degrees and CR = 4 bpp

| Image index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| n = 1 PSNR | 32.2 | 30.66 | 36.26 | 34.01 | 32.24 | 32.92 | 30.41 | 32.77 | 32.1 |
| n = 2 PSNR | 30.97 | 29.97 | 33.89 | 32.28 | 31.1 | 31.27 | 29.73 | 31.19 | 30.88 |
| n = 3 PSNR | 28.86 | 28.44 | 31.41 | 29.65 | 28.93 | 29.14 | 28.3 | 28.93 | 28.81 |
| n = 4 PSNR | 28 | 27.77 | 29.68 | 28.29 | 28.06 | 28.02 | 27.69 | 28.01 | 27.92 |

polynomial coefficients $z_{min}$ and the maximum integer of the quantized polynomial coefficients $z_{max}$ are necessary to satisfy $|z_{max} - z_{min}| \leq 2^{z_0}$, where $z_0$ represents the required bpp to transfer the identified range of integers to binary streams. According to our trials, it is

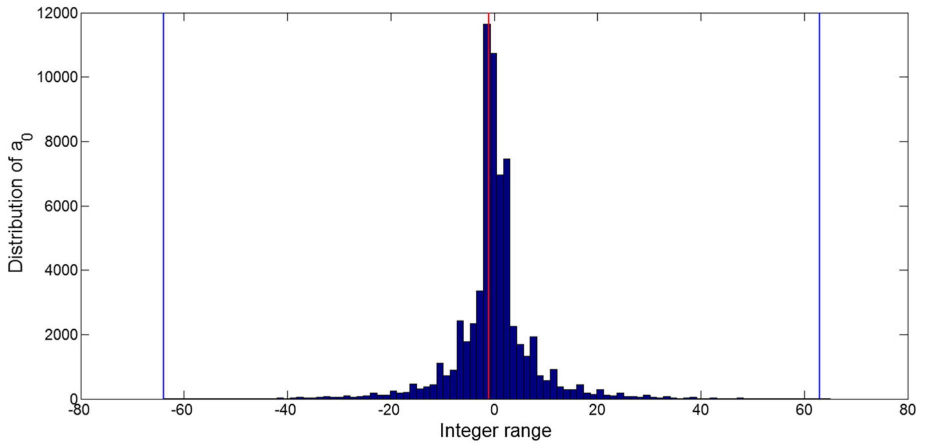

**Fig. 4** Distribution of $\{a_0\}$ around the most frequent integer (the vertical line in the middle) is represented with 128 integer range (between the extreme two vertical lines)

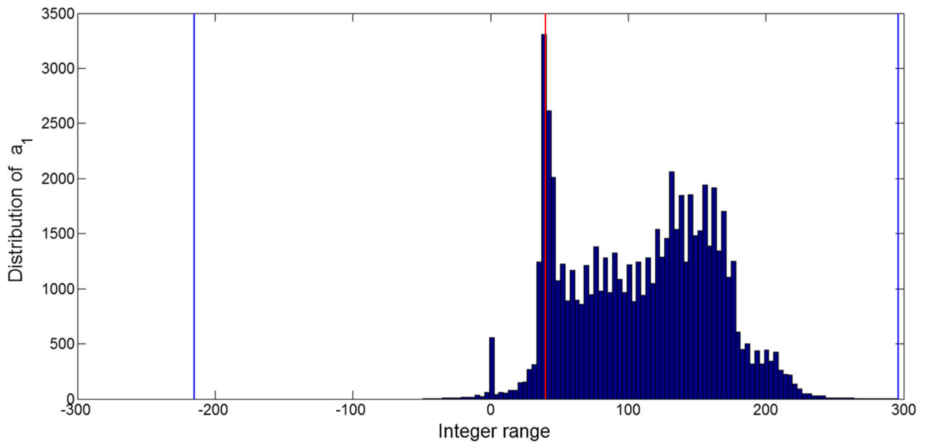

**Fig. 5** Distribution of $\{a_1\}$ around the most frequent integer (the vertical line in the middle) is represented with 512 integer range (between the two extreme vertical lines)

challenging to achieve consistent quantized values of the first degree polynomial coefficients and their integer value range. We shifted the most frequent integer to zero and distributed the other integers around zero. We noticed that the majority of integers of $\{a_0\}$ and $\{a_1\}$ in the distributions are different, such that $\{a_0\}$ need less bpp than $\{a_1\}$ to be represented. For more clarification, let $z_{image}$ be bpp for the tested image. In the matrix of polynomial coefficients, the columns of coefficients $\{a_0\}$ can be represented by $(z_{image} - 1)$ bpp. The other columns that represent the first degree terms $\{a_1\}$ can be represented by $(z_{image} + 1)$ bpp. As an example, in this paper's trials, $z_{image} = 8$ bpp, $\{a_0\}$ can be represented by 7 bpp and $\{a_1\}$ can be represented by 9 bpp. Figures 4 and 5 show the distributions of $\{a_0\}$ and $\{a_1\}$ for the image Lena.

## 3.4 Complexity Computations

Both the coding and decoding algorithms, each containing two parts, are discussed within this section. Inside the coding and the decoding parts, we have the two dimensional DCT, inverse DCT algorithms, and the sequence of one dimensional LSM implementations. For simplicity, and since all tested images in this paper are squares, we suppose that $N = M$, $N_1 = M_1$, $N_2 = M_2$. By looking at Eqs. (1, 2), and as found within the literature, the implementation of these equations use four nested 'for' loops, so it has complexity of $O(N^4)$ for $N \times N$ matrix. It is also known in the literature of DCT based algorithms, such as JPEG, that one dimensional DCT is of $O(N^2)$ can be used by running it $2N$ times to have the two dimensional DCT, therefore the complexity can be reduced to $O(N^3)$. Eventually, $O(N^2)$ one dimensional DCT can be replaced with one factored similarly to a Fast Fourier Transform which would have $O(NlogN)$, so the whole complexity of two dimensional DCT can be reduced to $O(N^2 logN)$. The inverse algorithm has the same complexity. For LSM, it is applied $\left(\frac{N_1}{N_s}\right)$ times in every row in the $N_1 \times N_1$ matrix. The complexity of one LSM implementation is multiplied by $\left(\frac{N_1}{N_s}\right) \times (N_1)$ in order to have the whole complexity for the sequence of LSM implementations that cover the whole $N_1 \times N_1$ matrix. By looking at Eqs. (3, 4) and from any advanced numerical linear algebra reference, we know that for one LSM implementation, we have dominant complexity from solving the system, as seen in Eq. (5), and it is $O\left((n+1)^2 N_s\right)$. Therefore the whole complexity of the sequence of LSM implementations is $O\left(N_1^2 (n+1)^2\right)$. The total complexity of the coding part of the algorithm is as follows:

- Two dimensional DCT for $N \times N$ matrix, $O(N^2 logN)$.
- Inverse two dimensional DCT for $N_1 \times N_1$ matrix, $O\left(N_1^2 logN_1\right)$.
- LSM for $N_1 \times N_1$ matrix, $O\left(N_1^2 (n+1)^2\right)$.

The total complexity of the decoding (recovery) part of the algorithm is as follows:

- Operations by recovering the fitted values from the LSM coefficients, $O\left(N_1^2\right)$.
- Two dimensional DCT for $N_1 \times N_1$ matrix, $O\left(N_1^2 logN_1\right)$.
- Inverse two dimensional DCT for $N \times N$ matrix, $O(N^2 logN)$.
- Applying filter (optional).

Since the complexity under $O$ operator is more precise, as $N$ approaches infinite, we notice that for $512 \times 512$ images, like the images we are testing, the complexity of LSM is more in the compression section compared to the decompression section. This is because 512 is not large enough to have similar/close processing time for implementing LSM (in the compression section) and only substituting back the values (in the decompression section). Additionally, there is a vast difference between $O\left(N_1^2 (n+1)^2\right)$ and $O\left(N_1^2\right)$ in the compression coding and the decompression coding respectively. For both of these reasons, the decompression of this algorithm is significantly faster when compared to the compression. From our trials, as shown in Fig. 6, the processing time for a fixed regular processor (laptop) and by Matlab suggests real time applications for the approach proposed in this paper.

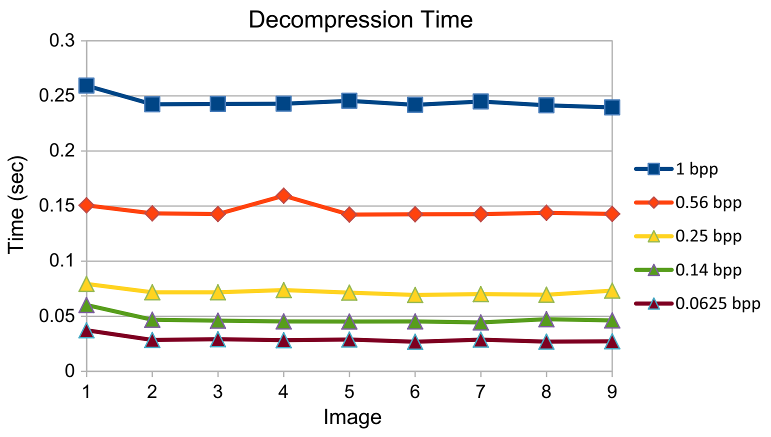

**Fig. 6** Decompression time for the whole image in seconds for the nine images with different CRs (bpp)

## 4 Results

Figures 3, 7, 8 and 9 depict graphical and visual results for our algorithm and approach. Figures 3, 7 and 8 show Lena, Mandrill, and House as fully recovered images with different CRs. Figure 9 shows a summary of different trials utilizing the nine images and showing their corresponding PSNRs. When these trials are compared to the most current literature, we find competitive PSNRs for high CRs. By reviewing Fig. 9, it is noted that our most successful PSNR can be seen in House (Image 3, Fig. 8), showing consistent quality across the different CRs tested.

By looking at Fig. 9, we notice that for all images the PSNR decreases with a decrease in bpp. However, it is not the case for Image 7 and partially Image 8. We think the reason is the non consistency associated with quantizing the LSM polynomial coefficients (explained Sect. 3.3). We may argue that LSM gives better accuracy, and therefore less error for larger $CR_{LSM}$. This is true for the non-quantized coefficients of LSM; however, the quantization process in our case is a probabilistic process, which has no guarantee that LSM will have a consistency in its error. This is because our quantization process is based on a search process for the better quantized combination that allows for generating binary streams. And it is not a deterministic-based process.

Table 3 illustrates a comparison between our method and methods available via recent published results. Although Lena (Image 6, Fig. 3) is not our best PSNR compared to our other tested images as shown in Fig. 9, Lena is still predominately higher in PSNR when compared to the majority of published works. Comparing the interpolation-based methods with our results indicated a significant progress towards PSNR.

Figure 10 presents a print for Image 7 for different CRs and their corresponding PSNRs by different algorithms. As we noticed, many of the papers cited in Table 3 have used JPEG, JPEG 2000, and SPHIT either directly or indirectly such as [12, 19, 20]. This is why we preferred to have Fig. 10 providing the print for our proposed method, JPEG, JPEG 2000, and SPHIT. We observe that in Fig. 10, we see the proposed method in general doing better than JPEG and JPEG 2000, but this is not the case for SPHIT. This observation is interesting as we can notice that the proposed method is having higher PSNRs compared with methods that used SPHIT directly or indirectly, as shown in Table 3 (explained by

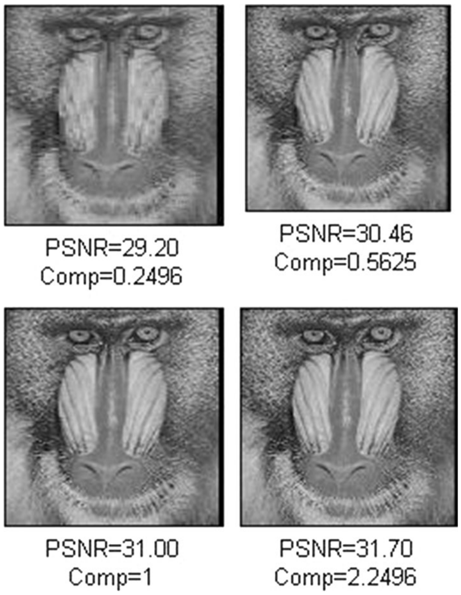

**Fig. 7** Mandrill (Image 7) with different CRs (bpp) and their PSNR

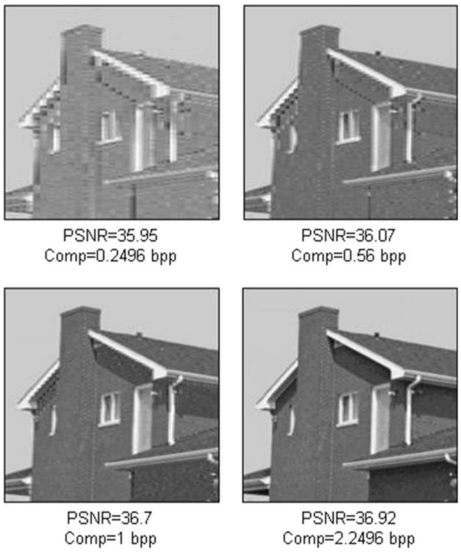

**Fig. 8** House (Image 3) with different CRs (bpp) and their PSNR

detail in the previous paragraph). The case of Image 7 (Mindrill) is showing that SPHIT in some cases can be consistently better than the proposed method. We can expect that to happen in image compression in general, but the case of Image 7 can trigger more research to find if there are features or segmentations in some images that cause the proposed method to lose part of its efficiency, more specifically to SHPIT and methods depend on it.

The research presented in paper [1] reports that the EAWI method shows better performance when compared to the Bi-Cubic and Spline methods. The method being presented here outperformed the EAWI and, as such, the Bi-Cubic and Spline methods. Also, some of the comparisons in Table 3 are CRs of lossless methods (not quantized data),

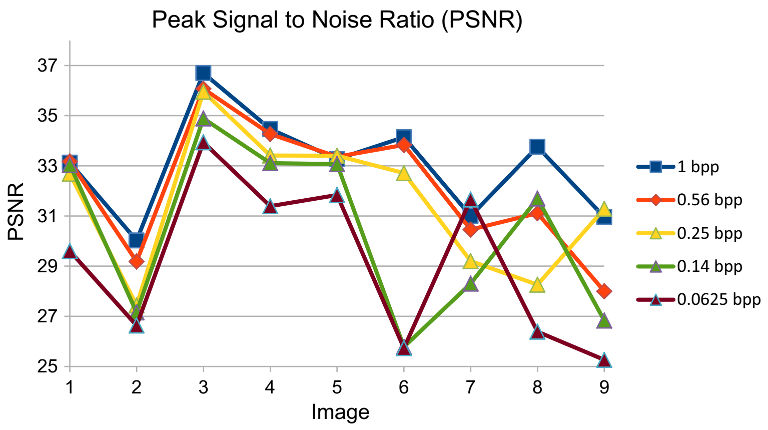

**Fig. 9** PSNR for the nine images with different CRs (bpp)

**Table 3** Comparisons of Lena, pepper, and house with many available published results

| Image | bpp/compression | Method/PSNR | This work's bpp/PSNR |
|---|---|---|---|
| Lena/image 6 | 4.3 (bpp) | EAWI [1]/33.534 (db) | 4.3 (bpp)/35.27 (db) |
| | 0.25 (bpp) | Method in [2]/33.2(db) | 0.2469 (bpp)/32.71 (db) |
| | | Standard JPEG [19]/30.5 (db) | |
| | | Optimized JPEG [19]/31.05 (db) | |
| | | Modified JPEG [19]/31.78 (db) | |
| | | Optimized JPEG-OPTQ [19]/32.6 (db) | |
| | 0.2 (bpp) | B-Spline [12]/29.45 (db) | 0.2 (bpp)/32.37 (db) |
| | | SPHIT [12]/27.43 (db) | |
| | Compression 30% or higher of the original size | Method in [15]/32.1 (db) or Less | 1 (bpp) = 12.5% of original size/34.14 (db) |
| | Compression 25% of original size | Cubic B-spline [13]/29.06 | |
| | | Optimized cubic B-spline [13]/30.17 | |
| | | Optimized B-spline [13]/30.8 | |
| Pepper/image 8 | 4.3 (bpp) | EAWI [1]/30.452 | 4.3 (bpp)/34.06 (db) |
| House/image 3 | 0.2 (bpp) | Method in [20]/26.47 | 0.2 (bpp)/35.62 (db) |
| | | JPEG/29.35 | |

which caused a significant rise in the PSNRs. Our method is able to provide quantized data that is ready to be transmitted/stored as binary streams while still showing better results overall.

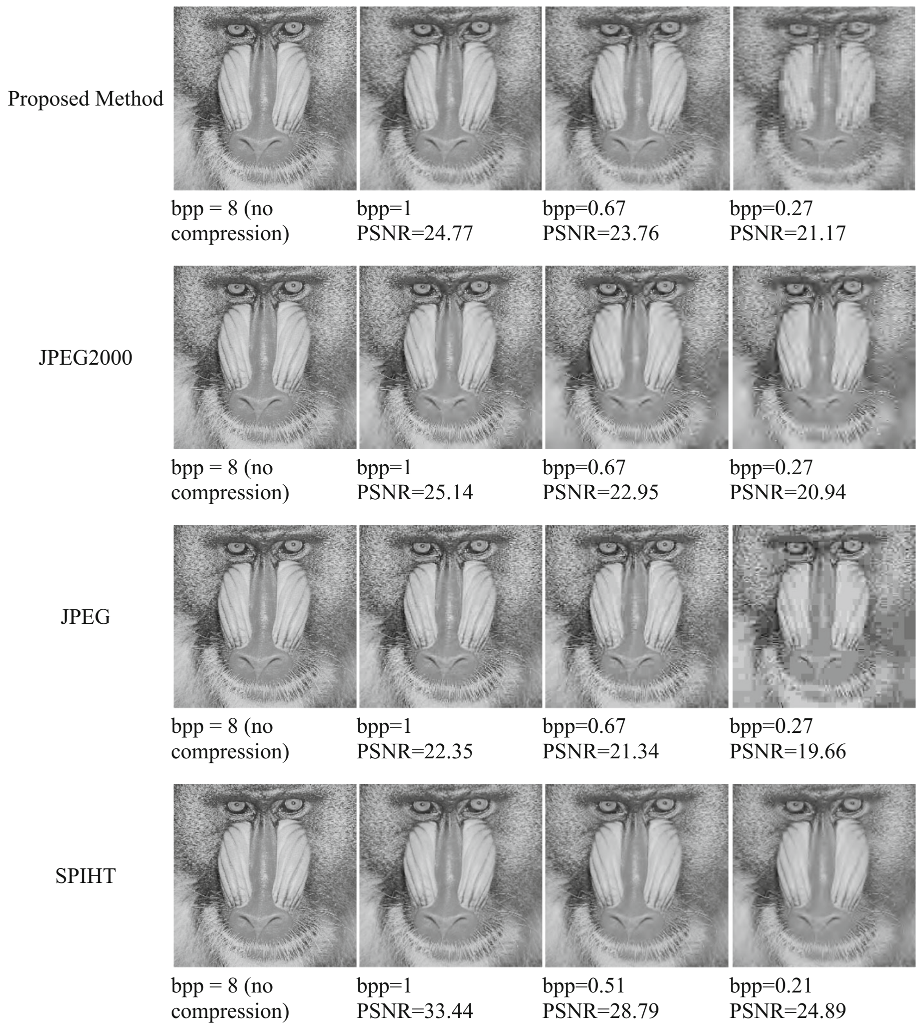

**Fig. 10** Image 7 printed and processed with the proposed method, JPEG, JPEH 2000, and SPIHT

## 5 Conclusion and Future Work

In this paper, we introduced a new approach to image compression. The approach uses two dimensional DCT and one dimensional LSM. Our proposed algorithm is capable of coding the image so it can be stored/transmitted as binary streams. Pseudocode and a full explanation of the steps for the algorithm were provided within this paper. The complexity computations along with the results from testing the decompression of images suggest the possibility that this method could be used in real time applications. The comparison showed significant progress, especially when compared to the research on interpolation-based methods. The significant improvement LSM has shown when merged with DCT will open many doors into future research opportunities and real time application. With this approach, we can see the possibility of extending techniques that are DCT/interpolation-based. We anticipate future works that will utilize the results as a base to expand upon.

**Acknowledgements** The authors of this research would like to show their gratitude to Skye Eisa, Dual M.A., LPCC for help in presenting the research findings through suggestions and editing of this text.

## References

1. Joseph, B. A., Ramachandran, B. (2012). Enhanced quality preserved image compression technique using edge assisted wavelet based interpolation. In P. S. Thilagam et al. (Eds.), *ADCONS 2011: Advanced Computing, Networking and Security* (vol. 7135, pp. 146–153). Berlin: Springer-Verlag.
2. Ouafi, A., Ahmed, A. T., Baarir, Z., & Zitouni, A. (2008). A modified embedded zerotree wavelet (MEZW) algorithm for image compression. *Journal of Mathematical Imaging and Vision, 30,* 298–307.
3. Averbuch, A. Z., & Zheludev, V. A. (2004). A new family of spline-based biorthogonal wavelet transforms and their application to image compression. *IEEE Transaction on Image Processing, 13*(7), 993–1007.
4. Giurcaneanu, C. D., & Tabus, I. (2002). Optimal coding of quantized Laplacian sources for predictive image compression. *Journal of Mathematical Imaging and Vision, 16,* 251–268.
5. Franssens, G., De Maziere, M., Fonteyn, D., & Fussen, D. (1998) Image compression based on a multipoint Taylor series representation. In *International symposium on computer graphics, image processing, and vision (SIBGRAPI '98).*
6. Bailo, G., Bariani, M., Chiappori, A., & Stagnaro, R. (2006). Adaptive interpolation algorithm for fast and efficient video encoding in H.26. In *14th European signal processing conference (EUSIPCO 2006), Florence.*
7. Lütai, G., & Feng, L. (2001). A method with scattered data spline and wavelets for image compression. In Y. Y. Tang, et al. (Eds.), *Wavelet Analysis and Its Applications* (vol. 2251, pp. 49–53). Berlin: Springer-Verlag.
8. Ajorloo, H., Manzuri-Shalmani, M. T., & Lakdashti, A. (2007). A Lagrange interpolation based error correction coding for the images. In *Proceedings of the 5th international symposium on image and signal processing and analysis.*
9. Oh, H., Lee, J., Min, C., & Jeong, J. (2010). Frame interpolation method based on adaptive threshold and adjacent pixels. In *2010 Sixth international conference on intelligent information hiding and multimedia signal processing.*
10. Rajakumar, K., & Arivoli, T. (2015). Lossy image compression using multiwavelet transform for wireless transmission. *Wireless Personal Communications, 87,* 315–333.
11. Das, M., Nethercott, J., & Rahrig, F. W. (1996). An efficient lossless image compression scheme for hierarchical, block-by-block transmission. In *IEEE 39th Midwest symposium on circuits and systems* (vol. 2).
12. Fahmy, M. F., Fahmy, G., & Fahmy, O. F. (2011). B-spline wavelets for signal denoising and image compression. *Signal, Image and Video Processing, 5,* 141–153.
13. Fahmy, M. F., & Fahmy, G. (2012). C12. Image compression using exponential b-spline functions. In *29th national radio science conference (NRSC 2012).*
14. Franti, P., Ageenko, E. I., & Kolesnikov, A. (1999). Vectorising and feature-based filtering for line-drawing image compression. *Pattern Analysis and Applications, 2,* 285–291.
15. Prabhakaran, P. J., & Poonacha, P. G. (2015). A new decimation and interpolation algorithm and an efficient lossless compression technique for images. In *2015 twenty first national conference on communications (NCC).*
16. Eisa, S. A. N. (2014). Numerical curve length calculation using polynomial interpolation. *Journal of Mathematical Imaging and Vision, 49,* 377–383.
17. Eisa, S. A. N. (2014). Three audio CODECs using the LSM interpolation and comparison with PCM. *British Journal of Mathematics & Computer Science, 4*(10), 1365–1380.
18. Yin, S., & Balchen, J. G. (1997). Image compression and transmission through a low-rate ultrasonic link in subsea telerobotic applications. *Journal of Mathematical Imaging and Vision, 7,* 41–53.
19. Lin, T. C., Truong, T. K., Chen, S. H., Lin, C. C., & Chen, P. D. (2006). DCT-based image CODEC embedded cubic spline interpolation with optimal quantization. In *Proceedings of the eighth IEEE international symposium on multimedia (ISM'06).*

20. Bastani, V., Helfroush, M. S., & Kasiri, K. (2010). Image compression based on spatial redundancy removal and image inpainting. *Journal of Zhejiang University Science C (Comput & Electron), 11*(2), 92–100.
21. Chen, X., & Cheng, A. M. K. (1997). An imprecise algorithm for real-time compressed image and video transmission. In *Sixth international conference on computer communications and networks.*
22. Algorithm and formula of two dimensional DCT in Mathworks for Matlab implementation: https://www.mathworks.com/help/images/ref/dct2.html.
23. Algorithm and formula of two dimensional Inverse DCT in Mathworks for Matlab implementation: https://www.mathworks.com/help/images/ref/idct2.html.

**Sameh A. Eisa** received his Bachelor degree in Communications and Electronic Engineering, 2010, Alexandria University, Egypt. Finished his Ph.D in May, 2017 in Applied and Industrial Mathematics, New Mexico Tech, USA. Have several publications in applied mathematics, mathematical modeling, control theroy and systems, and mathematical imaging and signal processing. The current areas of research are differential equations, dynamical systems, mathematical modeling, applied mathematics, industrial mathematics, and mathematical imaging and signal processing.

**Waleed Lotfy** received his B.Sc. in 2013 from Alexandria University in Computer and Communication Engineering, Egypt. He works now as a High Performance Computing Systems Engineer at Brightskies Technologies in Egypt. His interests lie mainly in the fields of High Performance Computing and Big Data.