A Novel Method for Image and Video Compression based on Two-Level DCT with Hexadata Coding

RODRIGUES, Marcos <http://orcid.org/0000-0002-6083-1303> and SIDDEQ, Mohammed

A Novel Method for Image and Video Compression based on Two-Level DCT with Hexadata Coding

Mohammed. M. Siddeq[1] and Marcos. A. Rodrigues[2]
[1]NTU-Northern Technical University, Computer Engineering Dept., Kirkuk- IRAQ
[2]GMPR-Geometric Modelling and Pattern Recognition Research Group,
Sheffield Hallam University, Sheffield, UK
mamadmmx76@gmail.com, M.Rodrigues@shu.ac.uk

Abstract
In this paper a novel method for 2D image compression is proposed and demonstrated through high quality reconstruction with compression ratios up to 99%. The proposed novel algorithm is based on a two-level Discrete Cosine Transform (DCT) followed by Hexadata coding and arithmetic coding at compression stage. The novel method consists of four main steps:1) A two-level DCT is applied to an image to reinforce the low frequency coefficients and increase the number of high frequency coefficients to facilitate the compression process; 2) The Hexadata coding algorithm is applied to each high frequency matrix separately through five different keys to reduce each matrix to 1/6 of their original size; 3) Build a probability table of original high-frequency data required in the decoding step; and 4) Apply arithmetic coding to compress each of the outputs of steps (2) and (3). At decompression stage, arithmetic decoding and a Fast Matching Search Algorithm (FMS-Algorithm) decodes the high frequency coefficients of step (2) using the probability table of step (3). Finally, two level inverse DCT is applied to decode the high frequency coefficients to reconstruct the image. The technique is demonstrated on still images including video streaming from YouTube. The results show that the proposed method yields high compression ratios up to 99% with better perceptual quality of reconstructed images as compared with the popular JPEG method.

Keywords: Image Compression, DCT, Hexadata Coding (HD-Coding).

1. Introduction
JPEG [1], [2] is the most widely used technique for image compression. It consists of simple steps and efficient hardware and software implementations of JPEG are widely available. Although JPEG was first developed for 2D image compression, it is also commonly used for encoding document-images (i.e. documents consisting of multiple images). Unfortunately, document-images encoded by the JPEG algorithm show blocks of artefacts at higher compression ratios [3,4]. Such artefacts significantly reduce the visual quality and clarity of the text, graphics and images when decompressed. Recently, several new coding methods have been developed for image compression [5]. However, the encoding processes of these methods are also ultimately more complex than the JPEG algorithm. For this reason, the JPEG algorithm remains the preferred coding method for image, document-image and video compression, and especially firmware systems [6,7].

The contribution of this research is to introduce and demonstrate a new method for image compression based on two-level DCT with Hexadata coding and arithmetic coding. The novelty of the proposed Hexadata compression algorithm is to reduce coefficients in high-frequency matrices. A common knowledge in data compression is that unrepeated coefficients or unrepeated sequences make it more difficult to compress data and, for this reason, the novel proposed method helps out by squeezing six data items into a single value. This new idea is successfully demonstrated on different types of high-quality images.

The advantages of the method are the elimination of block artefacts, high compression ratios and high perceptual quality of the decoded images. This is contrasted with the JPEG technique whose main problem is very low visual

quality at higher compression ratios due to block artefacts and reduced colour level. Experiments described in Section 6 show the superiority of the proposed method as compared to JPEG.

The steps in the proposed compression algorithm are illustrated in pseudocode as follows:

Hexadata coding algorithm in pseudocode:

```
//Read image from disk:
img = read_image(path);
//Break image into non-overlapping blocks:
xBlock = divide_image_into_blocks(img);
//First level DCT:
bDCT = apply_DCT_to_each_block(xBlock);
bDCT_Q=Quantization_Process(Factor, bDCT);
//DC and AC coefficients are kept into separate matrices:
[DC1_Matrix, AC1_Matrix] = extract_DC_AC_coefficients(bDCT_Q);
//Divide DC1 matrix into non-overlapping blocks:
xBlock2 = divide_DC_matrix_into_blocks(DC1_Matrix);
// Second level DCT:
bDCT2 = apply_DCT_to_each_row(xBlock2);
bDCT2_Q=uniform_Quantization_Process(K, bDCT2);
 // For each row save DC values as array and AC as concatenated matrix:
[DC2_Array, AC2_Matrix] = extract_DC_AC_coefficients(bDCT2_Q);
// Apply Hexadata and arithmetic coding algorithms:
[AC_Encoding1] = HexaDataCoding(AC1_Matrix);
  [Compression_Level1] = ArithmeticCoding(Encoding1);
[AC_Encoding2] = HexaDataCoding(AC2_Matrix);
  [Compression_Level2] = ArithmeticCoding(Encoding2);

[Compression_Level3] = ArithmeticCoding(DC2_Array);
SaveCompressedData(Compression_Level1,Compression_Level2,Compression_Level3);
```

The rest of the paper is organized as follows. Section 2 introduces the Hexadata (HD) compression method, Section 3 elaborates on the two-level DCT, Section 4 describes the decompression steps, Section 5 presents experimental results, Section 6 shows a comparative analysis with JPEG mainly from a visual perceptual quality assessment, and finally Section 7 presents conclusions from the current study.

2. The Hexadata Compression Method

The Hexadata compression as proposed in this paper is applied to high frequency coefficients obtained from the DCT. The proposed method is based on converting 6 data items to a single value using 5 different keys. Figure 1 represents the data flow.

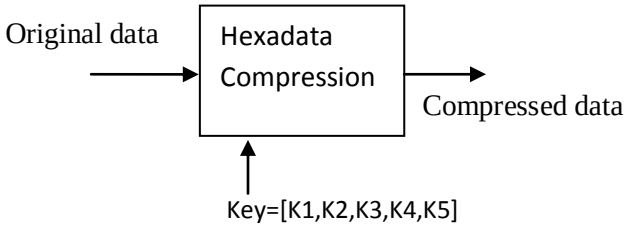


Figure 1: The Hexadata coding algorithm converts a set of original data to a stream of compressed data. The compression keys K_1 … K_5 are randomly generated.

The following steps illustrate the Hexadata compression method:

Step 1: The method reduces each group of six data items to a single value; this single value can be seen as a compressed-encrypted data as without knowing the compression keys the original data cannot be recovered. The process is thus, based on a set of five different keys applied to the data hierarchically. The following equation represents the first stage of Hexadata compression:

$$E(i) = K1 \times D(n) + K2 \times D(n+1) + K3 \times D(n+2) \quad (1)$$

Where *E(i)* is the encoded output from the stream of original data *D(n)*. The keys K1, K2, K3 are randomly generated and *i* is the index of encoded data items.

Step 2: The output from step 1 is converted to another stream of encoded data by multiplying each two values by their respective second level keys as shown in Figure 2 through the following equation:

$$X(j) = K4 \times E(i) + K5 \times E(i+1) \quad (2)$$

Where *X(j)* is the final compressed output from previous encoded data *E(i)* and K4, K5 are the randomly generated second level compression keys. Now the final encoded data are stored with index *j*.

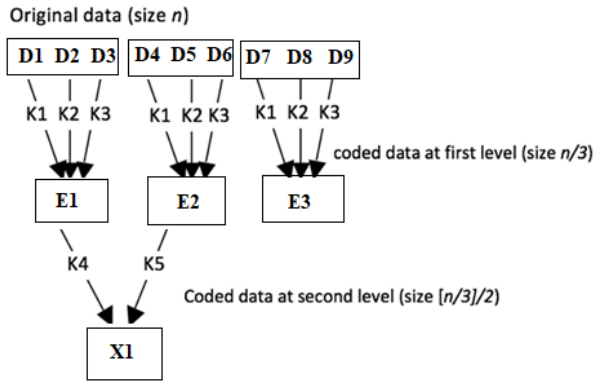


Figure 2: The Hexadata compression algorithm: compression keys are hierarchically applied to the data.

The reason we refer to the proposed algorithm Hexadata (HD) compression is because the length (number of items) of an array is minimized to 1/6 of their original length as each six items of data are converted to a single value. The key values K1, K2, K3, K4 and K5 are generated by a key generator algorithm according to the following steps [11,12]:

```
M  = (max(data))/2; // max value of array to be compressed
K1 = 1;             // set first 0 < key <=1
K2 = K1+M+F;        // Where F >=1 is an integer scaling factor.
K3 = F*M*(K1+K2);   // Where * is the multiplication operator.
K4 = rand;          // second level keys randomly generated by
K5 = rand;          // random function range=[0 - 1].
```

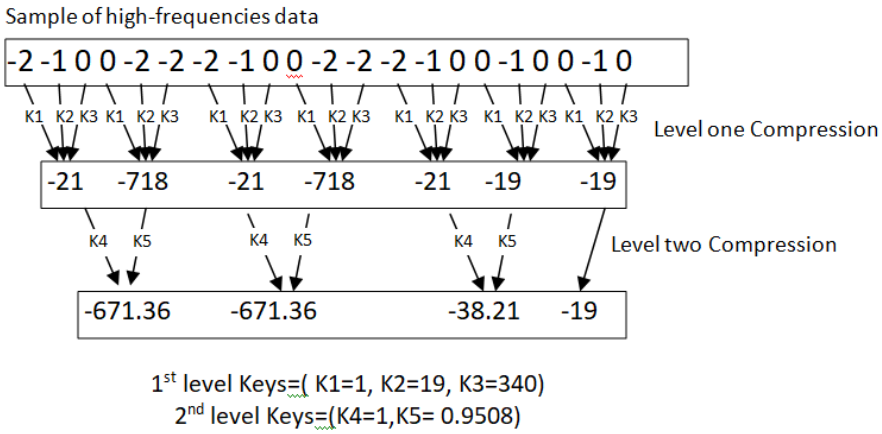


Figure 3: Illustration of the proposed Hexadata compression applied to a sample of high-frequency data.

3. The Two-level DCT Method

Siddeq and Rodrigues used two different types of discrete transforms (DWT and DCT) since 2014 reported in various publications [8-12]. The rationale behind using different transformations is to try and increase the number of high-frequency coefficients as these can help increasing compression ratios without loss of quality. In this research we apply the DCT [1,2] twice as it is a very efficient transformation capable of producing the sought high-frequency coefficients. The reason behind applying DCT twice is similar to DWT combined with DCT, as mentioned above in previous research. In practical terms, we show in this paper that applying DCT twice increases the number of high-frequency coefficients with reduced number of low-frequency coefficients, which is one of the advantages of the DCT we want to fully exploit.

Lossy image compression methods make use of quantization that can be of many types such as scalar, uniform and dot-division matrix [5]. Our method is based on dot-division matrix, with a quantization matrix n × n containing data generated through Equation 3. The proposed compression algorithm begins by dividing an image into non-overlapping n × n blocks (n ≥ 8) and then apply the DCT to produce de-correlated coefficients [3,4] followed by the proposed quantization equation:

$$Q_{(i,j)} = \begin{cases} 1, & i \text{ and } j = 1 \\ L, & i \text{ and } j \neq 1 \end{cases} \quad (3)$$

Where $i,j=1,2,\ldots,n$ and the quantization factor is an integer $L \geq 1$. Each of n×n blocks are quantized by Eq. (3) using dot division matrix followed by truncation of the result. The main reason to use this type of quantization is to remove insignificant coefficients and increases the number of zeros in each block. The parameter L main role is to increase or decrease the quality of an image. Thus, image details are reduced as the value of L increases. The range of L depends on the maximum absolute value of DCT coefficients; thus, there is not a fixed limit for the factor L.

After the transformation, each block in the frequency domain consists of a DC-component at location (0,0) representing the average value of the samples in the block while all other coefficients are called the AC coefficients. Each DC-coefficient is saved into a one-dimensional array called the DC_array, whose size is thus, the number of DC-coefficients in the image. The array is then divided into m sub-arrays ($m \geq 8$) and each sub-array is transformed by a one-dimensional DCT followed by a thresholding by the linear equation q(i)=K to remove insignificant coefficients. We recommend using K ≥ 2 so not to lose fine details in the image. The reason to use a double DCT in the sequence as described here is to increase the number of high-frequency components and reduce the DC-components to a minimum. After this stage, all DC components are compressed by arithmetic coding.

Please note that only the AC coefficients are encoded by the Hexadata algorithm and then Arithmetic Coding, which is a lossless transformation, plays an important role in helping to increase compression ratios significantly [15]. All Hexadata coded data are then subject to arithmetic coding. Figure 3 illustrates the proposed method.

As part of Hexa-data coding, a probability table is built representing a set of unique input data. This table is required at decompression stage. Figure 4 illustrates the probability data for some encoded data. We stress that sorting ascending is a very important step in the proposed method to speed up decompression, as a binary search method is used to determine the original data items composed of a single compressed value. For the same reason of speeding up decompression, the relevant 6 data items are saved in the probability table side-by-side with the compressed data.

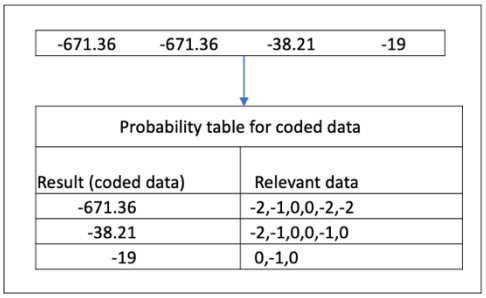


Figure 4: Illustration of probability table for compressed data. The input data is defined from Figure 3.

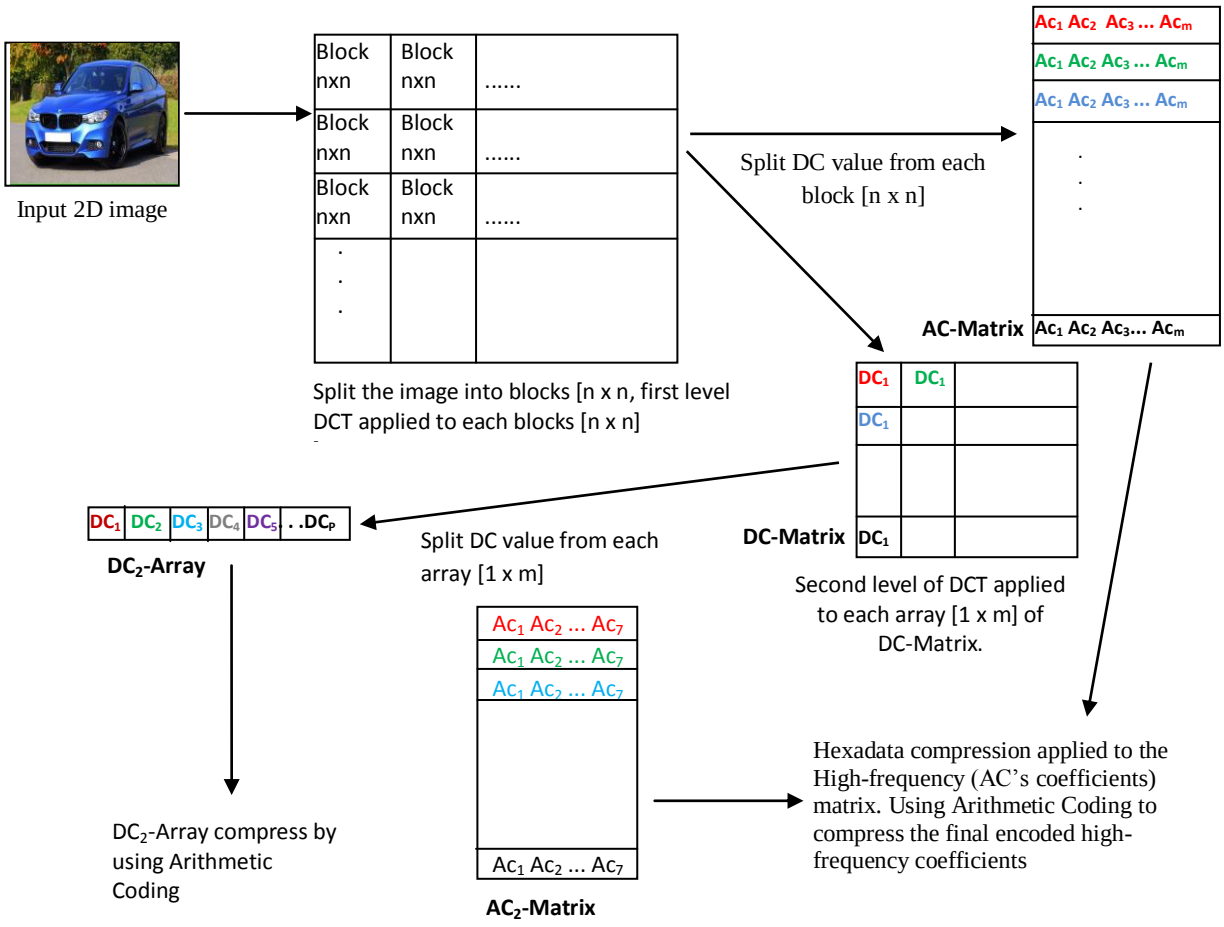


Figure 5: Proposed data transformations by two-level DCT and Hexadata Coding algorithm

4. The Decompression Algorithm

To decode the data, we reverse the compression steps. Decompression begins with arithmetic decoding then matching each compressed data with the outputs in the probability table. Thereafter, if the match is successful, the result will be the relevant 6 data items representing the decoded data. The method comprises a binary search within the probability table. Figure 6 illustrates the decoding steps. A Fast Matching Search algorithm based on binary search compares, at every iteration, the estimated values (i.e. the possible decoded values) of the data items with the items at the middle of the probability table. The probability table is split into 2, then each half (left or right) split into 2 again and so on. If the result matches then the estimated data items are the relevant 6 data representing the decompressed data. Otherwise, if the estimated value is less than the one obtained from the probability table, then the algorithm start searching again on the left sub-array or on the right sub-array if the value is greater [13]. There is no possibility of "Not Matched", because all the probabilities have been saved in the probability table at compression stage. This decoding method runs much faster than our previous algorithm proposed in the Patent WO 2016/135510A1 [14].

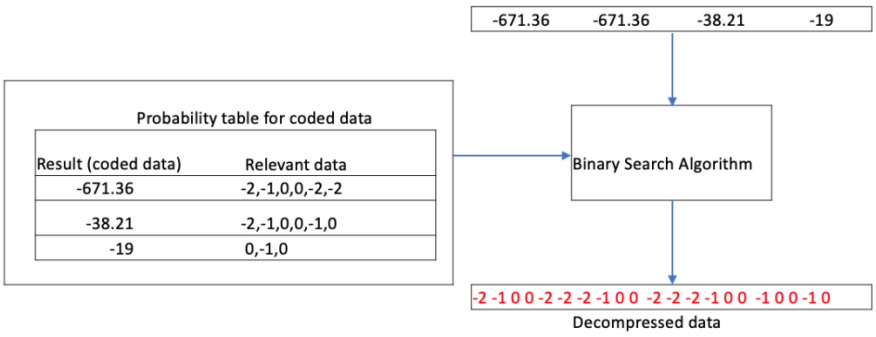


Figure 6: Reversing the Hexadata algorithm at decompression.

The inverse Hexadata algorithm is based on *binary search algorithm*, where matches in the probability table represent the decoded data. All high-frequency coefficients are decompressed by the inverse Hexadata algorithm and combined with the decompressed DC values (i.e. DC values decompressed by arithmetic decoding). The inverse DCT is then applied to the combined data obtaining the DC components. Similarly, the decoded DC components are combined with the decompressed high-frequency values followed by an inverse two dimensional DCT to obtain the original decompressed image. The decompression algorithm is illustrated in Figure 7.

5. Experimental Results

The proposed algorithm was implemented in MATLAB R2014a running on an Intel Core i7-3740QM microprocessor (8-CPUs) with Video card: GTX 960 NVIDIA. Before compression, we transform the RGB colour image to YCbCr format, which is used by most image compression algorithms for efficiency reasons [1,2]. Also, most image information is present in the first layer "Y" called brightness, while other components of "CbCr" contain less information about the image [3] and can be compressed more aggressively.

In this section we describe results in two parts:

1) We apply the method to general 2D images of different sizes and evaluate the quality of the images and their RMSE and SSIM [19].
2) We apply the proposed compression and decompression technique to a stream of video images.

Table 1 shows the first part of results by applying the compression method to four selected images shown in Figures 8, 9, 10 and 11.

Table 1: Compressed and Decompressed images by our approach.

Image Name		Original Image Size (MB)	1st Level DCT block size	2nd Level DCT data size	Quantization Factor			CR:1	Compressed Size (KB)	RMSE	SSIM
					Y	Cb	Cr				
Girl	High Quality	19.7	32 x 32	8	5	20	20	57	353	1.43	0.97
	Medium Quality	19.7	32 x 32	8	10	40	40	109	184	1.59	0.98
	Low Quality	19.7	32 x 32	8	30	70	70	248	81.1	2.04	0.82
Backyard	High Quality	48.2	32 x 32	8	9	10	10	18	2.59	3.2	0.91
	Medium Quality	48.2	64 x 64	8	20	20	20	99	495	7.3	0.899
	Low Quality	48.2	64 x 64	8	20	20	20	178	276	8.2	0.892
BMW	High Quality	38.7	32 x 32	8	20	50	50	46	854	4.8	0.893
	Medium Quality	38.7	32 x 32	8	50	150	150	78	504	7.4	0.886
	Low Quality	38.7	64 x 64	8	10	20	20	149	265	9.2	0.878
Big Ben	High Quality	28.5	32 x 32	8	50	50	50	41	706	6.9	0.95
	Medium Quality	28.5	32 x 32	8	100	100	100	75	389	9.5	0.881
	Low Quality	28.5	64 x 64	8	150	200	200	116	251	12.1	0.81

Note that the total compressed image sizes in Table 1 include the probability table which is saved in the header file (see Figure 4). The information in the probability table is needed at decompression stage and it can be used as part of an encryption key for security applications as, without this information, the image is un-recoverable. CR in Table 1 refer to Compression Ratio [2]

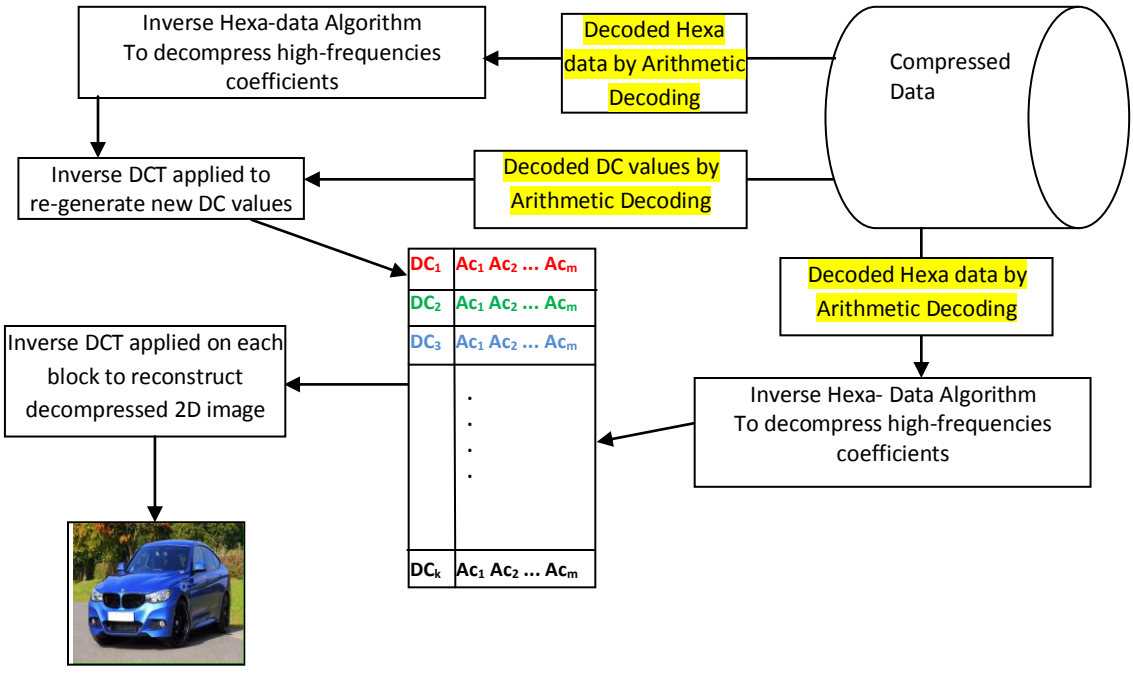
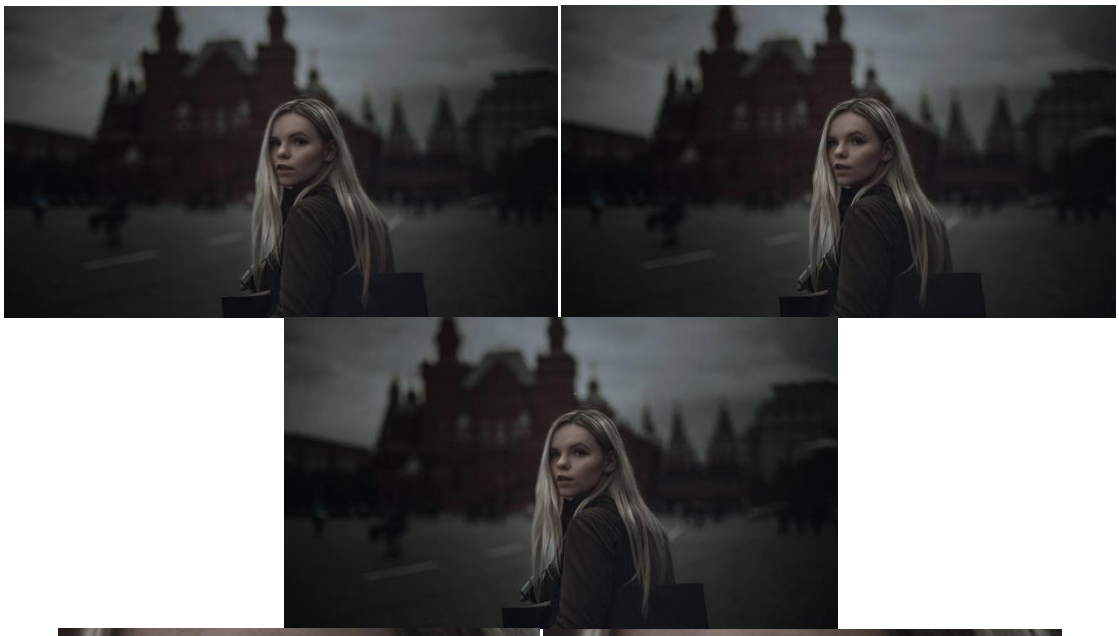
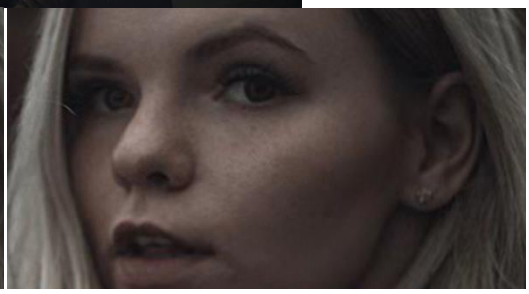


Figure 7: Steps in the proposed decompression algorithm



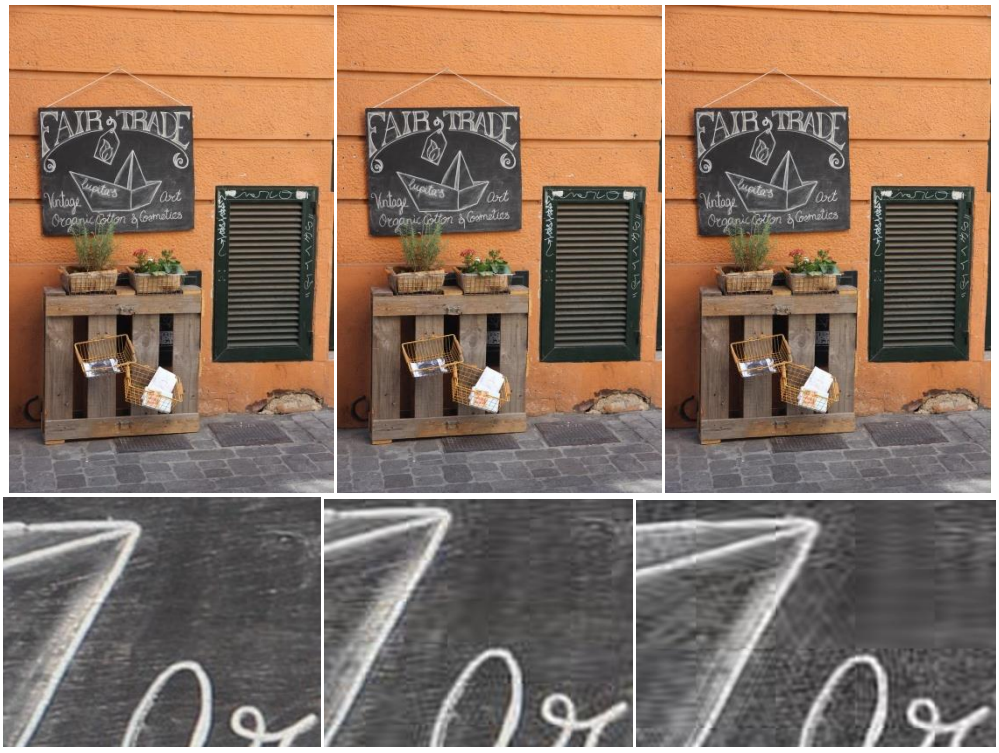
Compression ratio= 0.0175, RMSE=1.43



Compression radio=0.0091, RMSE=1.59

Compression ratio= 0.004, RMSE=2.04

Figure 8: Decompressed image by our approach: (left-top) represents high quality decompressed image with RMSE=1.43 by keeping most of details of the image, (Right-top) medium image quality RMSE=1.59 by removing some high-frequencies coefficients, which is not much different from previous decompressed image, (Middle-down) decompressed image and RMSE=2.04 which is a lower quality image as quantization removes some details from the low and high-frequency coefficients. Degradation (block artefacts) show in the zoomed in image.



RMSE=3.2
Compression Ratio=0.053

RMSE=7.3
Compression Ratio=0.01

RMSE=8.2
Compression Ratio=0.0055

Zoomed-in part of the decompressed Backyard images

Figure 9: Decompressed image by our approach. Left column: high quality decompressed image with RMSE=3.1 by keeping low-frequency details and most of high-frequency details. Middle column: medium image quality with RMSE=7.3 part of high-frequencies coefficients removed by quantization process, which is not much different from the original image. Right column: decompressed image with RMSE=8.2 represents a lower quality image as quantization removes some details of the low and most of high-frequency coefficients. Moreover, the DCT block size varies according to image quality; this is to demonstrate the ability of our proposed algorithm of using different block sizes.

Compression Ratio=0.02, RMSE=4.8

Compression Ratio=0.012, RMSE=7.4

Compression Ratio=0.0066, RMSE=9.2

Zoomed-in part of the decompressed BMW images

Figure 10: Our approach compresses a high-resolution 2D image to over 99%. Top row: represents decompressed 2D image with RMSE=4.8 still a high-resolution image. Middle row: decompressed 2D image with RMSE=7.4, in this stage most of high-frequencies are removed and some low-frequency data are changed. Although the RMSE is larger than previous value, image details are still preserved. Bottom row: decompressed image with RMSE=9.2 is a lower quality reconstruction.

Compression Ratio=0.024, RMSE=6.9

Compression Ratio=0.013, RMSE=9.5

Compression Ratio=0.008, RMSE=12.1

Figure 11: Decompressed image by our approach. Top row: high quality decompressed image with RMSE=6.9 by keeping low-frequency details and most of high-frequency details with quantization factor (*L*=50). Middle row: medium image quality with some degradation with RMSE=9.5and high-frequency coefficients partially removed by quantization factor (L=100). Bottom row: decompressed image with RMSE=12.1is a lower quality image as some details of the low and most of high-frequency coefficients are removed by quantization. This level of compression (over 99%) shows block artefacts.

Having demonstrated the algorithm for still images, we turn our attention to video compression. Table 2 illustrates results of the proposed method. The quality of video is largely dependent on the quantization factor used and the DCT block size. In general, it can be stated that the larger the quantization factor the lower the quality of the image, and the smaller the DCT block the higher the quality of the image. Two video sequence samples are shown in Figures 12 and 13.

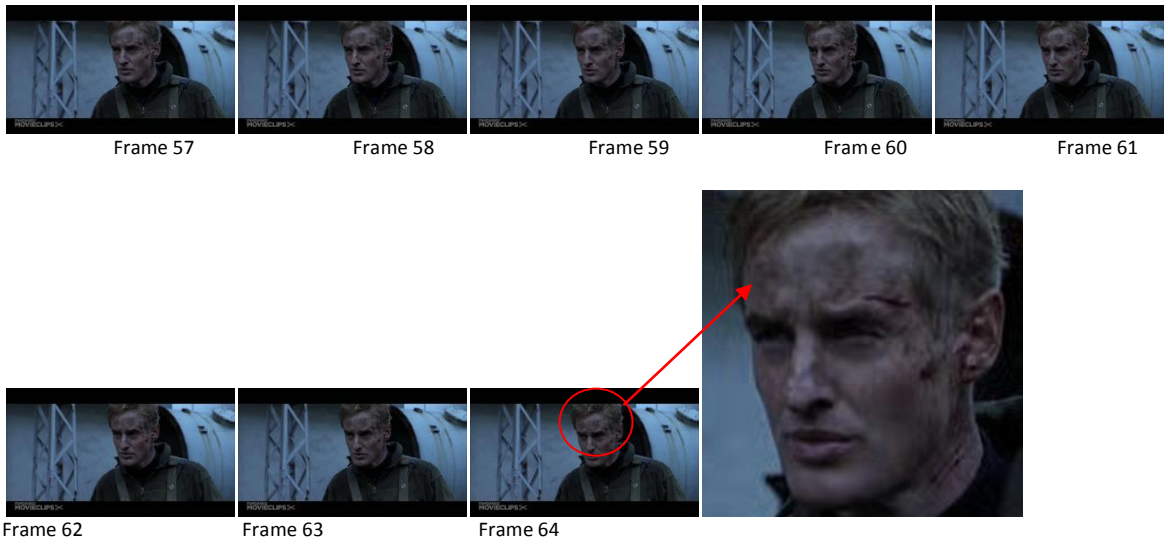
Table 2: Video compression and decompression by our approach applied to YCbCr layers to each image independently.

Video Name	Original video Size (MB)	Number of frames	Each image size (MB)	1st Level DCT block size	2nd Level DCT data size	Quantization Factor			CR:1	Compressed Video Size (MB)	Average RMSE	Average SSIM
						Y	Cb	Cr				
Video_1	379.52	64	5.93	32x32	16	**15**	**35**	**35**	391	0.97	1.7	0.972
	379.52	64	5.93	64x64	16	**35**	**45**	**45**	387	0.98	2.29	0.951
	379.52	64	5.93	64x64	16	**50**	**50**	**50**	412	0.92	2.67	0.92
Video_2	336.6	128	2.63	32x32	16	**10**	**25**	**25**	369	0.91	2.7	0.93
	336.6	128	2.63	16x16	8	**50**	**100**	**100**	374	0.899	5.2	0.88
	336.6	128	2.63	16x16	16	**100**	**150**	**150**	391	0.892	7.4	0.789

Note that on video compression quoted in Table 2 each frame is compressed independently and each frame has its local information (keys, probability table). Therefore, each compressed frame includes its local information independently. The reason behind using different block sizes of 16x6 and 8x8 is to show the ability of our proposed method to compress images at higher compression ratios using different block sizes. Changing block sizes can reduce the DC-components and increase the AC-coefficients as shown by our experiments. Comparing with the JPEG algorithm, the fact that JPEG is fixed on one level DCT and on block size of 8x8 is a handicap leading to lower image quality at high compression ratios.

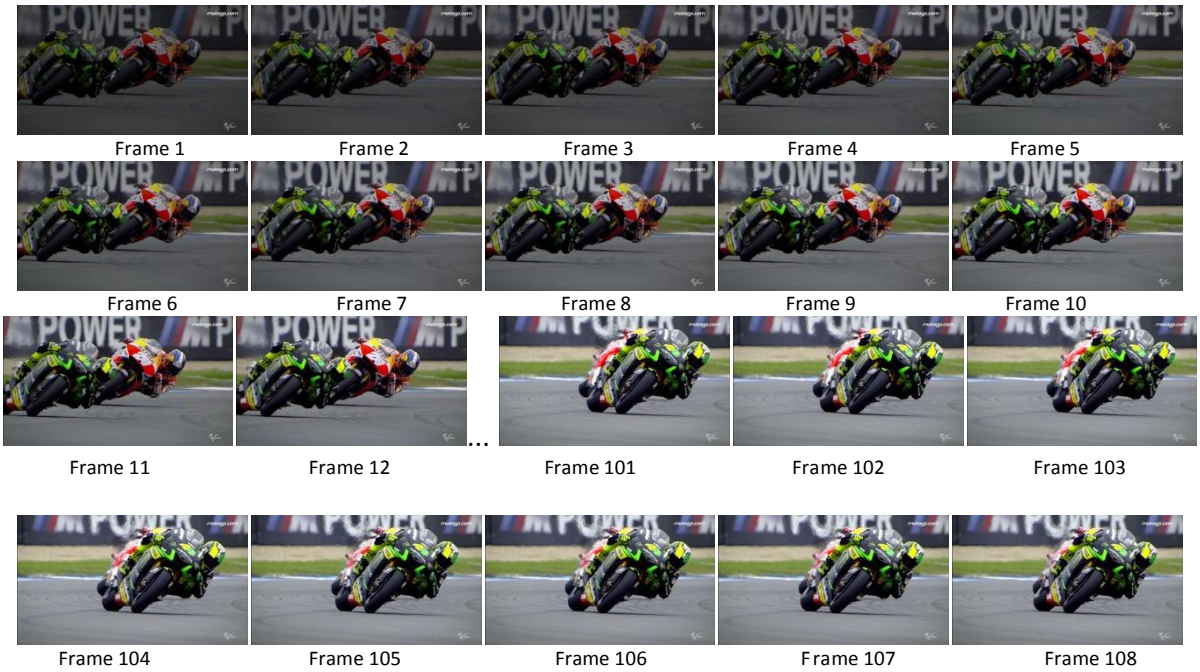
(a). Decompressed **Video_1** consist of 64 frames, average RMSE =1.7

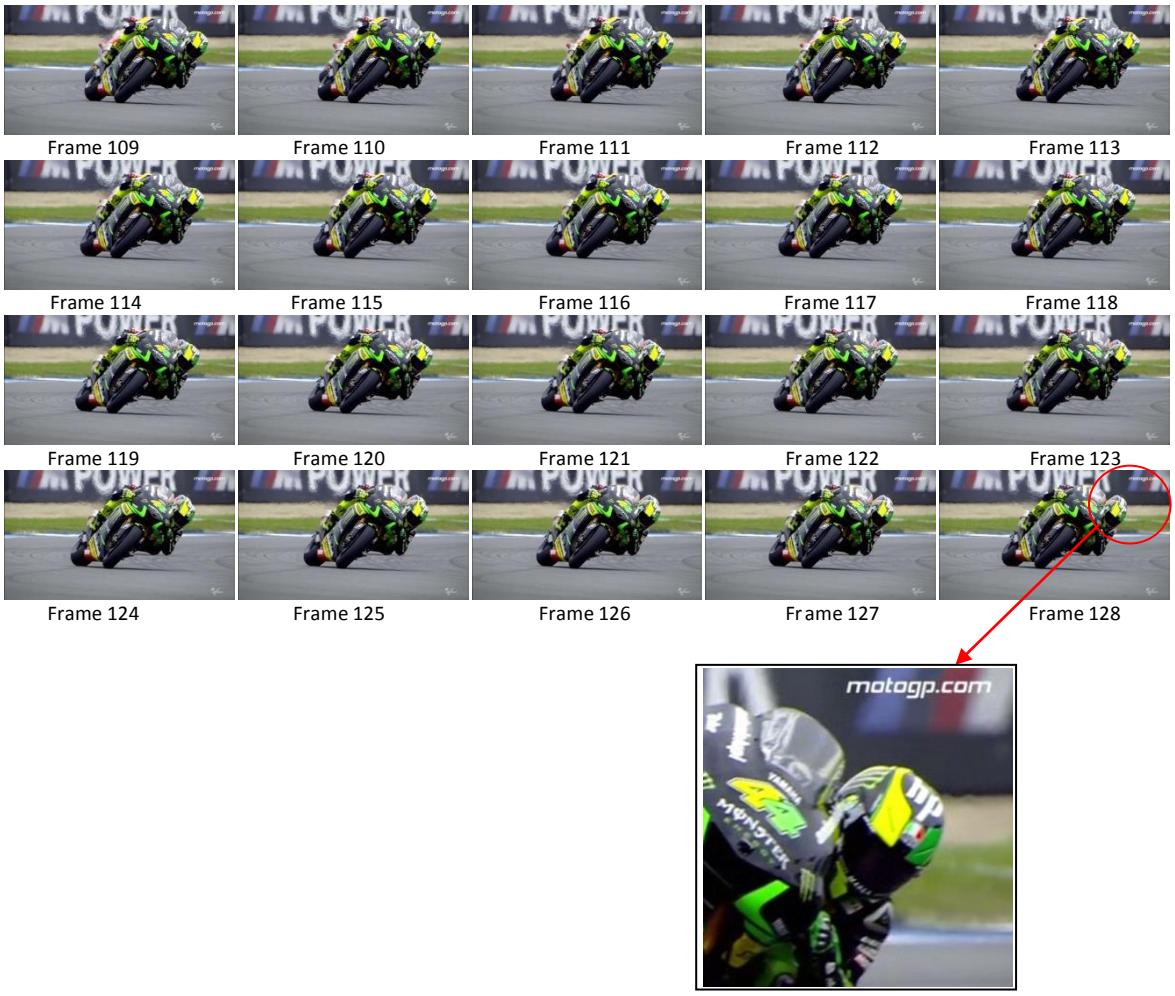
(b). Shows last 10 frames from the decompressed **Video_1**, average RMSE=2.29



(c). Shows last 10 frames from the decompressed **Video_1**, average RMSE= 2.67

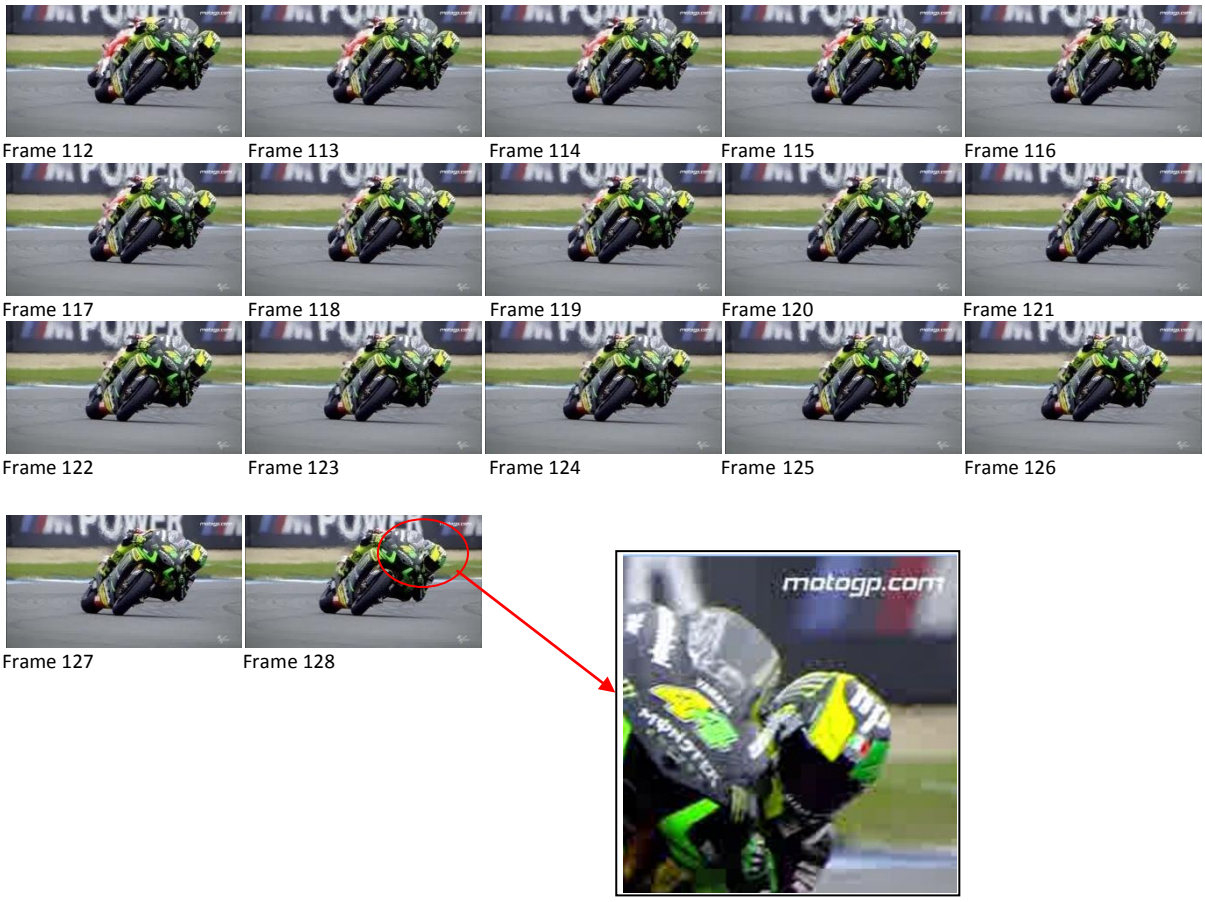
Figure 12: Decompressed video "**Behind Enemy Lines Movie CLIP (2001) HD**". **(a)** Shows decompressed video with average RMSE for 64 frames at 1.7, and compression ratio of 0.989. **(b)** Shows just the last 10 frames from the decompressed video for comparison with previous frames in (a). Some degradation is observed in the decompressed and the average RMSE for 64 frames is 2.29 yielding compression ratio of 0.993. **(c)** RMSE =2.67 for all frames. The last 10 frames show degradation in the decompressed video. The degradation is negligible in comparison with previous decompressed frames in (b). The compression ratio is 0.995.





(a). Decompressed **Video_2**consist of 128 frames, average RMSE =2.7

(b). Shows just the last 18 frames of 128 decompressed **Video_2** frames, average RMSE =5.2



(c). Shows just the last 18 frames of 128 decompressed **Video_2** frames, average RMSE =7.4

Figure 13: Decompressed video "**MotoGP™_Indianapolis_2014_Best_slow_motion**". **(a)** Shows the decompressed video for 128 frames with average RMSE=2.7, and video compression ratio of 0.942. **(b)** Shows just the last 18 frames from the decompressed video for comparison with previous decompressed frames in (a), some

degradation appeared in the decompressed video. The average RMSE for 128 frames is 5.2 and compression ratio of 0.982. **(c)** Shows the last 18 decompressed frames which shows some degradation with RMSE of 7.4and compression ratio of 0.989 (i.e. the degradation appears only after zooming-in).

6. Comparison with the JPEG Technique

The JPEG technique applied to the images and videos used in Section 5 yields good visual quality with lower compression complexity. However, the JPEG technique is unable to produce good visual quality at higher compression ratios, as the quality of the image is significantly degraded [15, 16]. Meanwhile the colour levels are reduced; in other words, real greyscales are lost yielding unreal 2D images. Figures 14, 15 and 16 show a comparison between JPEG and the proposed HD coding compression technique. The comparison is based on visual properties at higher compression ratios.

JPEG, RMSE=10.2, compressed size=112KB

HD-Coding, RMSE=2.04, compressed size=80.1KB

JPEG, RMSE=16.5, compressed size=295KB

HD-Coding, RMSE=8.2, compressed size=276KB

JPEG, RMSE=11.8, compressed size=303KB

HD-Coding, RMSE=9.2, compressed size=265KB

JPEG, RMSE=15, compressed size=261KB

HD-Coding, RMSE=12.1, compressed size=251KB

Left column: compressed and decompressed images by **JPEG** technique compared with our proposed **HD coding** (right column) at higher compression ratios. The **JPEG** technique could not reach the expected compressed size, also degradation clearly appears in the images. The **JPEG** compression reduces the colour levels in the image for higher compression ratios, and this affects image quality.

Figure 14: Visual quality comparison between **JPEG** and our proposed **HD coding** technique.

(a) As we can see samples of "Video_1" compressed by **JPEG** technique (i.e. each image compressed independently, total compressed video size= 2.31 Mbytes, Average RMSE = 11.5).

(b) Samples of "Video_1" compressed by our proposed technique **HD-Coding** (i.e. each image compressed independently, total compressed video size= 1.99 Mbytes), Average RMSE = 7.4. (See Table 2)

Figure 15: (a) shows the decompressed Video_1 by JPEG at higher compression ratio, the stream of the images is completely degraded, especially the details on the face. (b) Decompressed same video at higher compression ratio, we can clearly recognize the face of the actor and the expression on his face.

(a) As we can see samples of "Video_2" compressed by **JPEG** technique (i.e. each image compressed independently, total compressed video size= 4.0 Mbytes, average RMSE = 6.8).

(b) Samples of "Video_2" compressed by our proposed **HD coding** technique (i.e. each image compressed independently, total compressed video size= 4.05 MB), average RMSE = 7.4 (see Table 2).

Figure 16 (a) shows the decompressed Video_2 by JPEG at higher compression ratio, the stream of the images is slightly degraded. (b) Decompressed same video at higher compression ratio, we cannot identify degradations, but the RMSE is slightly higher than JPEG's RMSE, which means mathematically (i.e. according to RMSE rules) JPEG is better than our approach in this case.

Additionally, we applied our proposed algorithm to three greyscale images which are very popular in digital image compression [2,3] providing thus, a direct comparison for common images used in the literature. Table 3 and Figure 17 shows results of our proposed algorithm for greyscale images while Figure 18 shows results for JPEG compression.

Table 3: Compressed greyscale images at higher compression ratios by our proposed method

Image Name	Original Image Size	1st Level DCT block size	2nd Level DCT data size	Quantization Factor	CR:1	Compressed Size	RMSE	SSIM
Lena	1.0 MB	16x16	8	70	32	31.6 KB	4.79	0.776
Woman	256 KB	8x8	8	50	9	26.1 KB	14.01	0.678
Apples	1.37 KB	32x32	8	70	55	25.2 KB	4.29	0.808

The comparison between JPEG and our proposed method in Figures 15 and 16 are based on the RMSE and higher compression ratios. Our proposed algorithm shows superior performance to JPEG compressing Lena and Apples images while for the Woman image JPEG is the winner. This means our proposed algorithm works very well for larger images or more complex images (>=1 MB) as the proportion of high frequency components increases.

Lena Compressed size=31.6 KB, Women Compressed size=26 .1 KB, Apples Compressed size=25.2 KB
(Compressed images by our proposed method)

Figure 17: Our method: (Left): Lena's image compressed by our method with RMSE=4.79, (Middle) Woman's image with RMSE=14.01, (Right) Apple's image with RMSE=4.29.

Lena Compressed size=32.1 KB, Women Compressed size=24 .4 KB, Apples Compressed size=29 KB
(Compressed images by JPEG technique)

Figure 18: JPEG method: (Left) Lena's image compressed by JPEG with RMSE=5.9, (Middle) Woman's image with RMSE=9.3, (Right) Apple's image with RMSE=6.4.

A further comparison with previous work of Siddeq and Rodrigues which was based on two discrete transformations DWT and DCT is made here. Note that a main disadvantage of previous work is the complexity of the compression algorithms [9,10,11]. Table 4 shows the compression ratio for the Minimize-Matrix-Size algorithm (previous work [10,11,14]) compared with our proposed approach described in this paper. It is important to stress the novelties of the proposed approach which are the reduced number of steps at compression and decompression stages, resulting in faster reconstruction from compressed data with higher compression ratios. Additionally, the compression is intrinsically more secure with five different keys which presents an advantage to image compression of security sensitive images and videos.

Table 4: Comparison Matrix Minimisation algorithm with our proposed Algorithm

Image Name	Original Image Size	Matrix Minimisation algorithm (previous work [10,11,14])				Our proposed Algorithm			
		CR:1	Compressed Size	RMSE	SSIM	CR:1	Compressed Size	RMSE	SSIM
Lena (Greyscale)	1.0 MB	10	99 KB	3.8	0.781	32	31.6 KB	4.79	0.776
Woman (Greyscale)	256 KB	3	67 KB	4.7	0.729	9	26.1 KB	14.01	0.678
Apples (Greyscale)	1.37 KB	25	56 KB	2.3	0.854	55	25.2 KB	4.29	0.808
Girl (Colour)	19.7 MB	34	582 KB	1.7	0.91	248	81.1 KB	2.04	0.82
Back yard (Colour)	48.2 MB	25	1.9 MB	4.8	0.902	178	276 KB	8.2	0.892
BMW (Colour)	38.7 MB	32	1.2 MB	5.9	0.891	149	265 KB	9.2	0.878
Big Ben (Colour)	28.5 MB	32	906 KB	7.9	0.85	116	251 KB	12.1	0.81
Video_1 (Colour)	379.52 MB	35	10.8 MB	3.9 (Average RMSE)	0.831	190	1.99 Mbytes (Compressed Video Size)	2.67 (Average RMSE)	0.92
Video_2 (Colour)	336.6 MB	28	12 MB	6.9 (Average RMSE)	0.792	83	4.04 Mbytes (Compressed Video Size)	7.4 (Average RMSE)	0.789

To summarize the comparative analysis, results show that our proposed compression algorithm is capable of compressing images at higher compression ratios over 99% with no substantial degradation. In other words, it can compress to higher compression ratios than the JPEG technique and our previous work. However, the length of the compressed information placed at the header file required at decompression stage (e.g. the probability table which can be quite large) negatively affect file sizes.

The main contributions of the proposed method are highlighted as follows.

A- Faster than and with higher compression ratios than our previous work [10,11,14]. The main reason is that search runs sequentially in previous work with decoding time from seconds to minutes depending on the size of image. In the work presented here, decoding times are less than a second to a few seconds depending on the size of the image.

B- Images are compressed using five different keys while in our previous work we used three different keys. This means that the proposed method has higher security credentials than previous work as keys can be used as password protecting the file.

7. Conclusion

This paper presented a new method for image and video compression and demonstrated the quality of compression through RMSE and visual quality of reconstructed images. Similar to JPEG technique which is based on the DCT, our proposed HD coding method is based on a two-level DCT, and is significantly different from JPEG in the way the transformations are applied. It embodies a number of additional steps at compression stage where the most important and significant ones are the compression of high frequency data by the Hexadata algorithm, leading to increased compression ratios, and the coding of the data by using five different keys which are generated by a key generator.

At decompression stage, a binary fast matching search algorithm is used to recover the high-frequency matrix, using the same five symmetric keys (i.e. the same keys used in the compression steps). Another feature of the algorithm is its reduced complexity making it much faster than any of our previous work. The results demonstrate that the

approach yields high image quality at high compression ratios compared with JPEG technique and our previous work. Furthermore, it is demonstrated that it is able to reconstruct video frames at high compression ratios.

On the down side, the complexity of HD coding method with multiple steps is greater than that of existing codecs such as JPEG due to the coding of each six items of data which also increases the execution time for large images. We are working on increasing the compression ratio of header file information and on increasing performance through concurrent programming techniques and results will be reported in the near future.

As future work, we intend to address the issue of security by protecting the generated keys (and perhaps some further information from the header such as probability table) by encrypting those with standard algorithms such as AES which are proven methods. This would make the proposed method a per-file compression technique with partial encryption as, without the encrypted information, data cannot be recovered.

References

[1] **I.E. G. Richardson (2002),** *Video Codec Design*, John Wiley &Sons.

[2] **K. Sayood (2000)**, *Introduction to Data Compression*, 2nd edition, Academic Press, Morgan Kaufman Publishers.

[3] **R. C. Gonzalez** and **R. E. Woods (2001)**, *Digital Image Processing*, Addison Wesley publishing company.

[4] **Yaqin Xie, Jiayin Yu,Shiyu Guo,Qun Ding and Erfu Wang (2019)**, Image Encryption Scheme with Compressed Sensing Based on New Three-Dimensional Chaotic System, Entropy 2019, 21(9), 819; https://doi.org/10.3390/e21090819.

[5] **Md. Ahasan Kabir and M. Rubaiyat Hossain Mondal (2018)**, Edge-Based and Prediction-Based Transformations for Lossless Image Compression, Journal Imaging 2018, 4(5), 64; https://doi.org/10.3390/jimaging4050064 - 04 May 2018

[6] **Jose Balsa,Tomás Domínguez-Bolaño,Óscar Fresnedo,José A. García-Naya andLuis Castedo (2019)**, Transmission of Still Images Using Low-Complexity Analog Joint Source-Channel Coding, Sensors 2019, 19(13), 2932; https://doi.org/10.3390/s19132932 - 03 Jul 2019

[7] **Halah Saadoon Shihab, Suhaidi Shafie, Abdul Rahman Ramli and Fauzan Ahmad (2017)**. Enhancement of Satellite Image Compression Using a Hybrid (DWT–DCT) Algorithm. Sens Imaging (2017) 18: 30. https://doi.org/10.1007/s11220-017-0183-6

[8] **M.M. Siddeq** and **G. Al-Khafaji (2013)**, Applied Minimize-Matrix-Size Algorithm on the Transformed images by DCT and DWT used for image Compression, *International Journal of Computer Applications, Vol.70, No. 15.*

[9] **M.M. Siddeq** and **M.A. Rodrigues (2014)** A Novel Image Compression Algorithm for high resolution 3D Reconstruction, *3D Research. Springer Vol. 5 No.2.*DOI 10.1007/s13319-014-0007-6

[10] **M.M. Siddeq** and **Rodrigues, Marcos (2015)**. Applied sequential-search algorithm for compression-encryption of high-resolution structured light 3D data. In: BLASHKI, Katherine and XIAO, Yingcai, (eds.) *MCCSIS: Multiconference on Computer Science and Information Systems 2015*. IADIS Press, 195-202

[11] **M.M. Siddeq** and **Rodrigues Marcos (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), p. 26. DOI 10.1007/s13319-015-0055-6

[12] **Siddeq, Mohammed and Rodrigues, Marcos (2017)**. A Novel High Frequency Encoding Algorithm for Image Compression. EURASIP Journal on Advances in Signal Processing, 26. **DOI:** 10.1186/s13634-017-0461-4.

[13] **Knuth, Donald (1997)**,*Sorting and Searching*: Section 6.2.1: Searching an Ordered Table, The Art of Computer Programming 3 (3rd Ed.), Addison-Wesley. pp. 409–426. ISBN 0-201-89685-0

[14] **Sheffield Hallam University, Mohammed M Siddeq,** and **Marcos A Rodrigues (2016)**. Image Data Compression and Decompression Using Minimize Size Matrix Algorithm. WO 2016/135510 A1.

[15] **ShuyunYuan, Jianbo Hu (2019).** Research on image compression technology based on Huffman coding, Journal of Visual Communication and Image Representation Volume 59, February 2019, Pages 33-38

[16] **Peiya Li, Kwok-Tung Lo (2019).** Joint image encryption and compression schemes based on 16 × 16 DCT, Journal of Visual Communication and Image Representation. Volume 58, January 2019, Pages 12-24

[17] **YouTube (2019), Behind Enemy Lines (3/5) Movie CLIP HD**.https://www.youtube.com/watch?v=XFdEntyO6TY, last access Jan-2019.

[18] **YouTube (2019), MotoGP™ Indianapolis 2014 -Slow motion**.https://www.youtube.com/watch?v=XFdEntyO6TY, last access Feb-2019.

[19] **Wang Zhou, Bovik, Alan C., Sheikh, Hamid R., and Simoncelli, Eero P (2004).** Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, Volume 13, Issue 4, pp. 600–612, April 2004