Transforming Pixels: Crafting a 3D Integer Discrete Cosine Transform for Advanced Image Compression

R. Rajprabu[1], T. Prathiba[2], Deepa Priya V[3], Arthy Rajkumar[4], Rajkannan. C[5], P. Ramalakshmi[6]
Assistant Professor, Department of Electronics and Communication Engineering, Kamaraj College of Engineering and Technology (An Autonomous Insitution), K.Vellakulam, Tamilnadu, India [1, 2, 6]
Assistant Professor, Department of Information Technology, Kamaraj College of Engineering and Technology (An Autonomous Insitution), K.Vellakulam, Tamilnadu, India[3, 4, 5]

***Abstract*—We propose an innovative technique for image compression based on the 3-dimensional Integer Discrete Cosine Transform (3D-Integer DCT), which will serve as an alternative to the existing DCT-based compression technique. If an image is encoded as cubes [row × column × temporal length] instead of blocks [row × column], higher compression can be achieved. Here, the number of blocks is represented as the temporal length. To construct cubes, we use highly correlated blocks, and the correlation level is determined using the mean absolute difference (MAD). The suggested 3D-Integer DCT-based coder can achieve a higher compression ratio while maintaining the required image quality. It also needs fewer coefficients to encode an image than the usual Joint Photographic Expert Group (JPEG) coder. Adopting integer DCT further reduces the computational complexity of the proposed algorithm, given the abundance of methods available in the literature to determine equivalent integers for DCT. We choose an optimum integer group that minimizes mean squared error (MSE) and improves coding efficiency for computing 3D-Integer DCT. We also conducted a detailed analysis to examine the impact of implementing integer DCT in image compression. When we look at peak signal-to-noise noise ratio (PSNR), bits per pixel, and structural similarity index (SSIM), we see that the proposed algorithm does a better job than the standard real-value DCT-based compression algorithm like JPEG.**

Keywords*—Discrete cosine transform; 3D integer DCT; Image compression; JPEG algorithm

I. Introduction

The image compression algorithm finds its place almost everywhere where storage, retrieval, and image file transfer are required. People widely use the standards developed by the Joint Photographic Experts Group (JPEG) [1] to compress images [2] and [3]. In the earlier release of JPEG, they adopted DCT to achieve energy compaction [4]. Later, they started to adopt the discrete wavelet transform (DWT) [5] because of its higher compression efficiency compared to DCT. While DWT outperforms DCT in hardware implementation, JPEG prefers DCT. This is because DCT specifies faster computation structures [6] to [12].

Almost all video compression standards adopt DCT for the same reason [13] and [14]. If we replace the real values in the basic functions with their equivalent integer values, we can further improve the computational efficiency of DCT. In study [15] and [16], we state a few approximation methods to find the equivalent integer values, preserving the properties of the basis function. Therefore, integer DCT greatly improves the computational efficiency for image compression. A standard DCT-based image compression technique computes the image block by block, with block sizes ranging from 8 x 8 to 32 x 32. We propose a new method based on 3D-IDCT, which promises a higher compression ratio than the current DCT-based compression technique. The proposed algorithm computes DCT using integers, thereby reducing computational complexity during implementation.

II. Related Works

Multi-carrier communication systems use the Discrete Cosine Transform Matrices [21], which contain the submatrix generated by the highest spark with mathematical concepts. Researchers have used the reconstruction of compressed functions to address compression-based sensing issues. This technique will solve the channel estimation-related issues, and it will be applied in both noise-based environments. The innovative image watermarking technique according to the 2-dimensional discrete cosine transform [22] has been implemented to recognize the copyright safety of the images. It has been implemented into the particular image blocks with a fixed coefficient to produce the watermark position by embedding and extracting functions within the frequency coefficients. The iterative sampling technique with the discrete cosine transform [23] has been constructed to minimize the dimensionality issue and also minimize the computational complexity. When applied to the amplitude-related angle, the Bayesian technique uses a set of coefficients with basic functions to quantify the trade-off within posterior uncertainty components. The Differential Evolution Markov Chain technique regenerates a similar level of coefficients with a reduced number of parameters.

The Quantum Discrete Cosine Transform model [24] demonstrates the capability of representing signals and images with a reduced number of coefficients. By developing the quantum compression methodology, we have reduced the computational complexity to allow for real-time applications. The complex, unstructured issue has been reduced to the identification of significant coefficients in an effective manner. The ant colony optimization algorithm utilizes the 2D-DCT [25] technique to minimize Gaussian noise and discover the useful frequency coefficient. The hybrid technique [26] has been constructed to implement the digital watermarking that is applied to images. The technique not only achieves robustness but also eliminates noise during the compression process. Robotic applications implement the multi-variant adaptive regression technique [27] to construct a group of videos and images and identify the image quality during compression. The machine learning technique eliminates image distortion by evaluating the image quality. The digital image forgery has been identified using the cellular automata technique [28] to identify the feature vectors with the nearest neighbor identification technique by discovering the duplicated regions in the image. For the prevention of misinterpretation of the image content, the discrete cosine transform has been implemented for feature extraction in every block.

The steganography technique [29] has been utilized to implement the protection while converting the JPEG image into a similar lossy channel that has the capability of anti-compression to perform the extraction more accurately. When producing the code, the compressing channels have a high detection conflict, which shows the relationship within the minimal distortion technique through the coefficient values. The quantum cosine transforms [30] has been implemented to obtain the highest efficiency while computing the encryption and compression of quantum images. The 5-dimensional hyper-chaotic system is used to compress the input image by providing the Zigzag coding technique with the highest amount of key space and providing enhanced security. The dynamic behavior technique enables security in a hyper-chaotic system. The asymmetric multi-image encryption method with the conditional decomposition technique [31] has been utilized to provide synthesized spectral image classification from the original image. The transformation has been done using DCT within the spatial region to complete the pixel-scrambling process. The multi-valued Fourier transformation was used for phase-only masks.

Multi-focus images have been identified using the spatial frequency technique [32], which combines with the discrete cosine transform method to identify the fusion values from the original images. The mean value of every original image has been computed using the Min-Max normalization and DCT coefficients. The principal component analysis has been computed to provide a better output compared with the other methods. The 16-point discrete Cosine Transform framework [33] has been utilized to provide VLSI hardware applications for processing video and image-based systems. The detection of digital image forgery has been implemented using the discrete cosine transform [34] technique by applying the concept of image splicing and including the regions of the images. The technique employs the dimensional-based decomposition process and the enhanced transformation process to identify the forgery regions. Every block computes the coefficient values, and the SVD algorithm extracts the features. The measurement of roughness has been used to identify the skewness with the feature vector; the feature reduction is used to construct the kernel-based principal component analysis. The hybrid methodology [35] has been utilized to employ the smoothing of the histogram peaks with an adaptive geometric filter used to measure the enhancement.

To improve the visual quality of decompressed images, a frequency-domain filter [36] is used to eliminate the blocking artifacts adaptively.

III. Proposed Techniques

There are four modes of operation in JPEG to compress an image, namely sequential, progressive, hierarchical, and lossless coding. The lossless mode does not use the DCT for energy compaction; rather, it uses predictive coding. The remaining three modes fall under the lossy compression technique. While the sequential mode encodes and decodes the image block by block in a raster scan order, the progressive and hierarchical modes incrementally improve the quality of the compressed image. Fig. 1 displays the block diagram of the base-line JPEG encoder and decoder. The encoding starts with level shifting, 2D-DCT, quantization, zigzag scanning, and finally entropy coding. The decoder side reverses the same process.

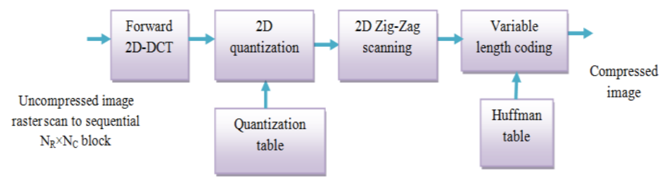

Fig. 1. Block diagram of JPEG encoder.

A. Discrete Cosine Transform

DCT is commonly used in digital signal processing applications; in particular, it finds its space in image and video compression algorithms. There are several forms of DCT transformation, and they are implemented as type I, type II, type III, and type IV. DCT-II is mostly used for compression algorithms. The Eq. (1) and Eq. (2) define the 2-D DCT and inverse DCT within signal f of length Nr × Nc as,

$$F(R,C) = \sqrt{\frac{4}{N_r.N_c}}\ f_r\ f_c\ \sum_{r=0}^{N_r}\sum_{c=0}^{N_c} f(r,c).\ \frac{\pi\cos(2r+1)R}{2\,N_r}\ \frac{\pi\,\mathrm{Cos}(2c+1)C}{2\,N_c} \tag{1}$$

$$f(r,c) = \sqrt{\frac{4}{N_r.N_c}}\ f_r\ f_c \sum_{R=0}^{N_r}\sum_{C=0}^{N_c} F(R,C)\ \frac{\pi\cos(2r+1)R}{2\,N_r}.\ \frac{\pi\cos(2c+1)C}{2\,N_c} \tag{2}$$

where,

$$f[r,c] = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } r=0, c=0 \\ 1, & \text{others} \end{cases}$$

In spite of calculating the DCT for a three-dimensional block of size N×N×N, the 2D-DCT is extended to one more dimension to get the 3D-DCT because it possesses separability and orthogonality. Eq. (3) and Eq. (4) provide the equation for calculating the 3D-DCT.

$$F(R,C,D) = \sqrt{\frac{8}{N_r.N_c.N_d}}\ f_r\ f_c\ f_d \sum_{r=0}^{N_r}\sum_{c=0}^{N_c}\sum_{d=0}^{N_d} f(r,c,d)\,.\ \frac{\cos(2r+1)R\pi}{2\,N_r}\frac{\cos(2c+1)C\pi}{2\,N_c}.\frac{\cos(2d+1)D\pi}{2\,N_d} \tag{3}$$

$$\text{Where } f_r,\ f_c, f_d = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } d, c, r = 0 \\ 1, & \text{others} \end{cases}$$

where, F(R,C,D) denotes the frequency domain intensity value and f(r,c,d) demonstrates the time domain intensity value. The inverse 3D-DCT value is computed in Eq. (4)

$$f(r,c,d) = \sqrt{\frac{8}{N_r . N_c . N_d}}\ f_r\ f_c\ f_d \sum_{R=0}^{N_r}\sum_{C=0}^{N_c}\sum_{D=0}^{N_d} F(R,C,D)$$
$$\frac{\cos(2r+1)R\pi}{2\,N_r}\ \frac{\cos(2c+1)C\pi}{2\,N_c}\ \frac{\cos(2d+1)D\pi}{2\,N_d} \tag{4}$$

3D-DCT based encoder for image compression

The proposed technique for image compression follows the principles of 3D-DCT. Fig. 2 displays the encoder block diagram. The spatial domain represents images as rows and columns of pixels. In order to apply 3D-DCT to the images, highly correlated blocks of size N×N are used to construct the cube. We use Eq. (5), mean absolute difference (MAD), to determine the level of correlation between the blocks. We construct cubes by finding the MAD between the seed block, which is the first block in the cube, and the remaining blocks.

$$\frac{1}{N_r \text{ x } N_C} \sum_{r=1}^{N_r}\sum_{c=1}^{N_c} \left| f(r,c)_1 - f(r,c)_8 \right| \tag{5}$$

where, N_r and N_C denote the total number of pixels in rows and columns,

After building the cube, use DCT in both the time and space domains to get 3D-DCT. This should come after 3D-Quantization, 3D-zigzag scanning, and finally coding with inconsistent extent coding.

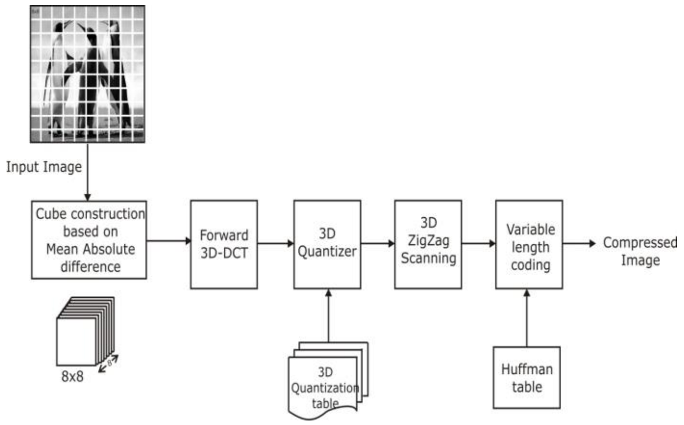

Fig. 2. Encoder block diagram of 3D-integer DCT based encoder.

B. 3D Quantization

3D quantization plays a significant role in image compression. This stage is where the actual compression occurs. DCT, which solely compacts energy and is reversible, renders the quantization process non-reversible due to the truncation or rounding off of the coefficients. Unlike 3D-DCT-based compression techniques, is not applicable. Since DC and AC coefficient ranges are different, in the case of 3D-DCT, the DC coefficient ranges between 300 and 4000, whereas the AC coefficient ranges between ±1000 [17]. Fig. 3 illustrates that the major axis accumulates more than 80% of the cube's total energy.

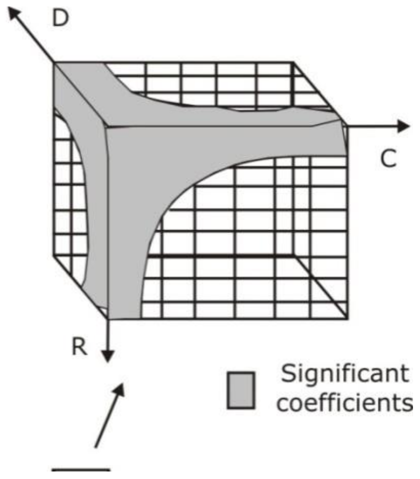

Fig. 3. Distribution of significant coefficients of image cube.

Using the goodness of fit test to analyze the allocation of 3D-DCT coefficients [18], we found that the Gaussian distribution identifies DC coefficients and the Gamma distribution computes AC coefficients. The quantization process assigns due importance to significant coefficients. The research in [11] conducted a detailed analysis, selecting the DC coefficients in the range of 8 to 16, and the AC coefficients in the range of 45 to 250.

C. 3D Zigzag Scanning

Each stage of the encoding process can achieve a significant amount of compression. After performing 3D-Quantization, the majority of AC coefficients become zero. We can achieve higher compression by effectively ordering the coefficients. We order the coefficients concentrated around the major axis first, followed by the remaining 3D-DCT coefficients. The vital thought is that significant coefficients (coefficients around the major axis) are framed according to the summation of the indices given as ($r + c + d \leqslant k$). The smaller the sum, the lower the frequency. The insignificant coefficients are then ordered based on the sum of their indices, given as ($r + c + d \geqslant k$). Fig. 4 shows the 3D-zigzag ordering of 3D-DCT coefficients, where r, c, and d are the integers with the values of 1 to 8 or 16, k = 3, 4,... (r + c + d).

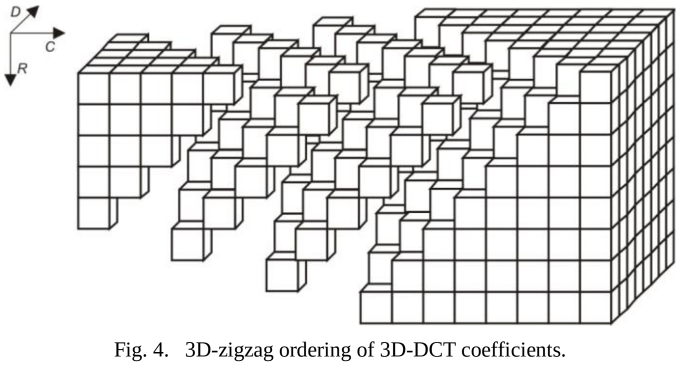

Fig. 4. 3D-zigzag ordering of 3D-DCT coefficients.

D. Integer Discerte Cosine Transform

The implementation of DCT has undergone numerous enhancements, all aimed at reducing complexity by reducing the number of multiplications and additions. When computing DCT with real values, floating-point multiplication and additions become inevitable. The computational complexity and resource utilization rise in the case of floating-point manipulation, computational complexity and resource utilization increase. Integer DCT is regarded as an alternative that will reduce the complexity of the existing DCT-based computational structures. The literature states two different methods to determine the equivalent integer set for the corresponding real value transform: the C-matrix transform [15], an indirect method of computing integer values, and the direct method [16].

E. Effectiveness of Approximated Integer DCT

We evaluate the approximated integer DCT values, determined by the C-matrix method and the direct method, based on the mean square error and transform or coding efficiency. The Markov procedure computes the mean square error and transform efficiency of the 3D-integer DCT based on the mean and unit conflict. The Markov procedure identifies the inter-element correlation coefficients between 0 and 1. The inter-element correlation will be uniform in both the spatial and temporal domains. The direct method [10, 9, 6, 2, 3, 1, 1] generates an optimized integer set with maximum transform efficiency and a relatively low mean squared error, as determined in [19]. Generally, people do not prefer higher integer values, despite their lower mean squared error compared to the optimized integer set. As the number of bits used to represent the integer value improves, the bit length of the multiplier and adder increases, resulting in higher consumption of hardware resources. Generally, we express the computational complexity in terms of multiplications and additions. To compute 3D-Integer DCT, the optimized integer set needs 48 multiplications and 78 additions.

We calculate the positions of these pixel values based on their edges. We divide the areas into different colors based on the edges, and it's important to steer clear of pixel values that directly align with the edges. We emphasize that we should extract values from both sides in the order they occur. We also extract the pixel values located on the image's boundary, which yields favorable outcomes for reconstructing lost pixels during decoding. These techniques yield a one-dimensional signal holding the required pixel values, with which it is possible to build a plan p with all its pixels, which are neighbors to the edges and also the boundary pixels of the image. We visit these pixels row-to-row until we reach the end, and then apply the following algorithm to each pixel y:

F. Algorithm – Interger DCT

```
Begin Procedure
  For a pixel y in plan p it is placed into the line $L_1$
    If $L_1$ is not found empty, then
      Obtain the pixel y from $L_1$
      Remove the value from $L_1$
    End if
    If y is present inside $L_1$, then proceed with Step 2
      Place y on $L_2$ and take it out from p
      Append the pixel value of y to one dimensional signal
      Set its last to y
    End if
    If $L_2$ is not found empty, then
      Obtain the pixel y from $L_2$ and take it out from $L_2$
        For each and every adjacent pixel $Y_a$
          Assumed to be in Pa adjacency positions of y
          Compute the value of p
          Compute the distance within Pa and last of y
        End For
    End if
    If the values are found to be $>sc_i$, then
      Place Pa into line $L_1$
    Else
      Place Pa into line $L_2$ and take it out from p
    End if
      Append the pixel value of pa to one dimensional signal
      and set to y last to y
  End For
End Procedure
```

While we have identified the pixels in line L_1, we have not yet added the value for the one-dimensional signal. On the other hand, the pixels in line L_2 already have their values added to the one-dimensional signal, but the adjacent areas remain unidentified. We guarantee that we won't overlook the row-to-row traversal across all pixels. Once we eliminate pixels that are part of a one-dimensional signal, we ensure termination. This technique aims to gather the pixel values at the edges directly, but it also focuses on s_d since pixel values can also be found at a reasonable distance from the edges. We can view this as an advantage, as edges that are sufficiently close to each other have the potential to contribute to the pixel values.

We must reduce the gathered pixel values by sub-dividing the data and applying small input values to large sets. This leads to a decrease in pixel values near the edges, scattering the values there to lower the resolution in the region. One-dimensional signals allow for identical sub-sectioning. A parameter s_d is used for sectioning $s_d \in \{1,...,255\}$ where s_d is utilized over a multiple channel and it stores each and every value obtained. We previously discussed the marginal change of pixel values over the edges, which also impacts the one-dimensional signal. Linear polynomials can reconstruct the lost pixels, but they don't work between pixels with different edges. As a result, the values of pixels can differ considerably. It is essential to subdivide the pixel values of these different edges alone. It is to be noted that the method for collecting the pixel values has been studied. Imagining that the already-gathered pixels belong to the same section is crucial when using repetitive search. For each and every edge section, it is possible to obtain a different one-dimensional signal. This technique does not require any additional information about the image to be stored.

By flattening the obtained original signal, the sectioning hypothesis can analyze the quality of the created signal for improvement. The proposed technique enables the flattening of an individual one-dimensional signal using filters with a standard deviation of 1, assuming that the pixels have a size of 1 x 1. This technique is expected to eliminate minor gaps in order to improve the compression rate. The sectioned pixels also include some adjacent information. The next stage of data reduction involves applying algebraic functions to the pixel values. Initially, the image will contain 256 distinct pixel values, one for each channel, from which the reduction in areas occurs to L distinct pixel values. The technique, known as tread of a stairway quantization, is an identical quantization technique that allows the construction of both small and large values for the original image.

Let $i_f \in \{0,\ldots,255\}$ is the value of a pixel for a one dimensional signal and let x = 255/((L-1)). The value after quantization in Eq. (6) is:

$$i_g = \left\lfloor \frac{i_f}{x} + \frac{1}{2} \right\rfloor \quad (6)$$

Here $i_g \in \{0,\cdots,L-1\}$, for constructing the image again it is necessary to calculate in Eq. (7).

$$i_f \sim X\, i_g \quad (7)$$

The processing divides the image into L intervals of size x, but does not include the initial and last intervals, which have a size of x/2. After constructing the images again, we set the pixel values of the initial one to 0 and the pixels of the last one to 255. The technique sets the other pixel values to their middle values. In the case of color images, the technique permits storing the quantization of each channel individually. The main focus is to adjust the size to match the possible values of pixels in the one-dimensional signal. Rotate these steps repeatedly until the borders reach a point where they no longer undergo transformation. The points for constructing the images are noted, and good reconstructions of the images are obtained. We need to add these points to reconstruct the pixel values during the decoding process.

G. Image Quality Measurement

Objective measures are generally used to assess the effectiveness of compression algorithms. The luminance component (Y) has been considered for analysis. We provide the PSNR and MSE measurement formulas in Eq. (8) and Eq. (9).

$$PSNR = 10 \log \left[\frac{255^2}{MSE} \right] db \quad (8)$$

$$MSE = \frac{1}{N_r N_c} \sum_{r=0}^{N_r} \sum_{c=0}^{N_c} \left[f(r,c) - \bar{f}(r,c) \right]^2 \quad (9)$$

The variables $f(r,c)$ and $\bar{f}(r,c)$ represent the original frame and the recreated image, respectively, while N_r and N_c signify the image size. Finding the Mean Squared Error (MSE) alone won't accurately reflect the quality of the image, as images with similar MSE tend to have different overall image quality. The structural similarity index is a measure of the perceived image quality between an original and reconstructed image. The study in [20] asserts that, in comparison to PSNR, SSIM provides a significantly superior measure of image quality. In an image, the dependency between the pixels carries information regarding the structure of the object. Therefore, we can calculate the SSIM using Eq. (10) by determining the mean, variance, and covariance of the original and reconstructed images. The SSIM value ranges from 0 to 1. The higher the similarity, the greater the value.

$$\text{SSIM}(m,n) = \frac{(2\mu_m\mu_n + Co_1)(2\sigma_{mn} + Co_2)}{(\mu_m^2 + \mu_n^2 + Co_1)(\sigma_m^2 + \sigma_n^2 + Co_2)} \quad (10)$$

where, μ_m is the mean value of the original sequence and μ_n is the mean value of the recreated sequence. σ_m^2 is the variance of the original sequence, and σ_n^2 is the variance of the reconstructed sequence; it is given in Eq. (11) to Eq. (15).

$$\mu_m = \bar{m} = \frac{1}{N} \sum_{i=1}^{N} m_i \quad (11)$$

$$\mu_n = \bar{n} = \frac{1}{N} \sum_{i=1}^{N} n_i \quad (12)$$

$$\sigma_m^2 = \frac{1}{N-1} \sum_{i=1}^{N} (m_i - \bar{m})^2 \quad (13)$$

$$\sigma_n^2 = \frac{1}{N-1} \sum_{i=1}^{N} (n_i - \bar{n})^2 \quad (14)$$

$$\sigma_{mn} = \frac{1}{N-1} \sum_{i=1}^{N} (m_i - \bar{m})(n_i - \bar{n}) \quad (15)$$

and Co_1 and Co_2 are arbitrary constants given in Eq. (16) and Eq. (17).

$$Co_1 = (Ko_1 Le)^2 \quad (16)$$

$$Co_2 = (Ko_2 Le)^2 \quad (17)$$

where, Le = 255 denotes the dynamic assortment of the signals, N represents the window dimension, and Ko_1=0.01, Ko_2=0.03. Typically, an [8×8] size is chosen for measuring the SSIM. Since it is merely a measure of similarity, one can choose any size.

IV. Experimental Results

Before stating the efficiency of the proposed technique, it is necessary to analyze the impact of using integer DCT for image compression, taking into account various sample images. We conducted the entire simulation using the luminance component and a 4:2:0 sampling format. We chose PSNR, bits per pixel, and SSIM as measures to verify the quality of the compressed image.

In order to analyze the effect of adopting integer DCT to compress image MSE, transform efficiency is considered [19]. It is determined that for the correlation coefficient of $\rho = 0.95$, the transform efficiency of the real value DCT is 93.99, and for the integer set [10, 9, 6, 2, 3, 1, 1], the transform efficiency is 94 with a mean square error of 0.0002. The quality of the compressed image reflects the minor deviations in transform efficiency and MSE. We determine this by comparing the compressed image of real and integer DCT with reference to PSNR, bits per pixel, and SSIM, as presented in Table I.

Tables I and II clearly show that the PSNR degradation between real and integer DCT is not significant. We found the degradation in PSNR of integer DCT to be between 0.01db and 0.11db when compared to real-valued DCT. Table I represents the minor increase in bits per pixel, which corresponds to the PSNR degradation. In addition, the majority of PSNR and bits per pixel values maintain the SSIM of integer DCT at the same level, and in some cases, there is an improvement in the range between 0.0001 and 0.0013. Since there has been no significant change in PSNR, bits per pixel, or SSIM between real and integer DCT, we can collectively state the compression ratio, which falls between 83:1 and 6:1.

The proposed 3D-DCT-based real and integer image coders compress the sample images. Table II compares the results against the standard JPEG coder (real value DCT) and integer value DCT, using a similar quality metric for measuring image quality. When comparing the transform efficiency of 3D-integer DCT to the 3D-DCT algorithm, there will be a slight degradation in the PSNR value. The cause of the degradation is that the transform efficiency of 3D-DCT [19] was 74.88%, whereas in the case of 3D-integer DCT, the transform efficiency was 74.81%. Table II values revealed the degradation in PSNR for 3D-integer DCT. Fig. 5 illustrates how the proposed method generates the SSIM value for various image types and compares it with related methods.

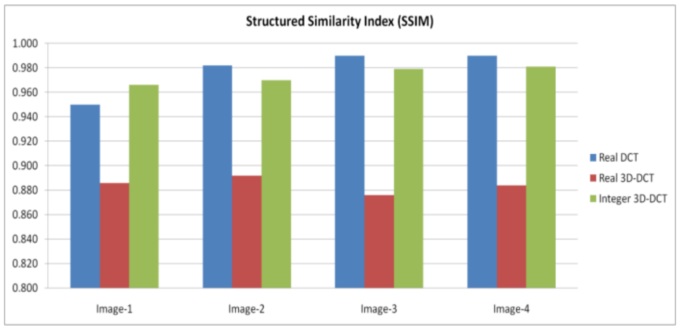

Fig. 5. Comparison of Structural Similarity Index with existing methods.

TABLE I. COMPARISON OF JPEG CODER CONSTRUCTED USING REAL VALUE AND INTEGER DCT

Sample images	JPEG coder constructed using real value DCT			JPEG coder constructed using integer DCT		
	PSNR [db]	Bits per pixel.	SSIM	PSNR [db]	Bits per pixel.	SSIM
Water	27.56	0.1429	0.5753	27.55	0.1424	0.5740
	29.83	0.2040	0.7022	29.82	0.2039	0.7021
	31.53	0.2985	0.7916	31.52	0.2982	0.7913
	33.67	0.4909	0.8772	33.65	0.4902	0.8764
	37	0.9658	0.9403	36.96	0.9672	0.9395
Lighthouse	27.39	0.1488	0.6176	27.39	0.1492	0.6178
	29.94	0.2148	0.7228	29.93	0.2147	0.7223
	31.73	0.3147	0.8110	31.70	0.3136	0.8102
	33.13	0.4577	0.8662	33.10	0.4570	0.8651
	36.86	0.9345	0.9337	36.83	0.9366	0.9334

TABLE II. COMPARISON OF PSNR, BIT RATE AND SSIM VALUES OF REAL AND INTEGER 3D-DCT WITH JPEG

Sample image	Real DCT [JPEG]			Real 3D-DCT			Integer 3D-DCT		
	PSNR [db]	Bits per pixel	SSIM	PSNR [db]	Bits per pixel	SSIM	PSNR [db]	Bits per pixel	SSIM
Lena	26.43	0.1950	0.6665	26.78	0.1066	0.6692	26.78	0.1066	0.6692
	29.62	0.3043	0.8004	28.80	0.2834	0.7643	28.81	0.2897	0.7648
	31.84	0.4305	0.8687	33.47	0.5219	0.8990	33.45	0.5219	0.8986
	33.57	0.5893	0.9089	37.18	0.5312	0.9372	37.13	0.5310	0.9368
	37.79	1.1073	0.9586	39.14	0.5321	0.9463	39.12	0.5318	0.9464
Pepper	26.78	0.1988	0.6949	26.74	0.1217	0.6513	26.73	0.1214	0.6534
	29.69	0.2856	0.8025	29.07	0.2512	0.7582	29.04	0.2532	0.7573
	31.85	0.3894	0.8601	34.27	0.5760	0.8988	34.25	0.5770	0.8978
	34.50	0.5894	0.9170	37.80	0.5833	0.9358	37.76	0.5842	0.9355
	38.43	1.1152	0.9594	38.83	0.5835	0.9407	38.82	0.5844	0.9406
Mandril	23.73	0.1906	0.4943	23.95	0.1052	0.5201	23.95	0.1053	0.5221
	25.97	0.4164	0.7044	25.32	0.4160	0.6340	25.32	0.4101	0.6345
	27.77	0.7125	0.8067	28.71	0.6059	0.8275	28.70	0.6064	0.8270
	29.71	1.1541	0.8781	32.41	0.6170	0.9183	32.38	0.6173	0.9177
	32.46	1.7863	0.9311	37.06	0.6190	0.9631	37.05	0.6195	0.9406

Table II shows that the 3D-DCT-based algorithm significantly improves the PSNR at higher [db] values in all the sample images, based on the bit rate, or bits per pixel. Consider the sample image of "Lena" For a given bit rate of 0.53, the proposed algorithm's PSNR improved by more than 5 db. Similarly, for the given bit rate of 0.58 in the "Pepper" image and 0.61 in the "Mandril" image, the proposed algorithm improved PSNR by 4db and 10db, respectively. The proposed 3D-Integer DCT-based compression algorithm has a compression ratio ranging from 110:1 to 20:1, and it outperforms the JPEG coder. Fig. 6 illustrates the computational time required to produce the compression for both the proposed method and its related methods.

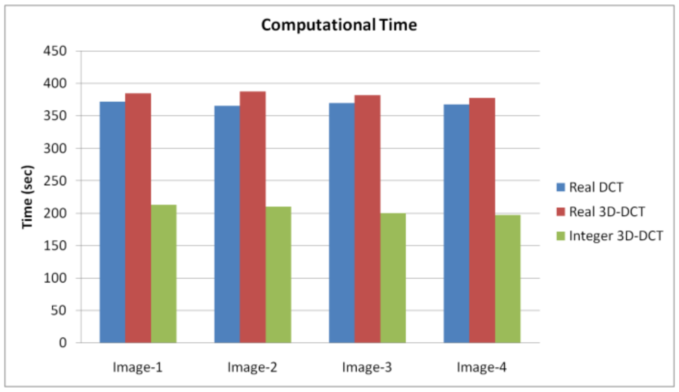

Fig. 6. Comparison of computational time with existing methods.

The primary reason for the proposed algorithm's improvement in PSNR was a reduction in the number of DC coefficients. A normal JPEG encoder encodes images block by block, ranging in dimensions from 8x8 to 32x32. For an image of dimension 512×512, if it is encoded with a block of dimension 8×8, then there are 4096 DC coefficients. The differential encoding method further codes these coefficients. In the proposed 3D-DCT-based algorithm, images are encoded as cubes of dimension 8×8×8 instead of blocks. For the same image size, the 3D-DCT-based compression algorithm requires only 512 DC coefficients. The differential encoding of DC coefficients achieves further rate reduction. The proposed algorithm, as shown in Fig. 7, achieves a greater rate reduction.

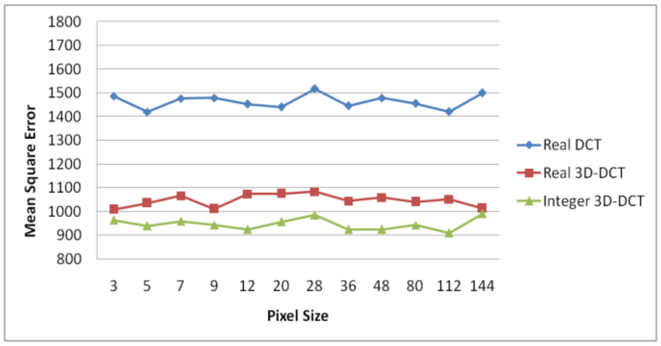

Fig. 7. Mean square error.

The proposed 3D-Integer DCT based algorithm outperforms the standard JPEG based image coder in terms of PSNR and bit rate, however the difference in SSIM of JPEG and the proposed algorithm is very minimum. It is found to be in the range between 0.001 and 0.02. The result holds true not only for the sample images considered for analysis, for any arbitrary image similar improvement can be achieved if it is compressed with proposed 3D-Integer DCT algorithm.

V. Conclusion

This paper proposes an innovative method for image compression based on 3D-Integer DCT. Instead of encoding an image as blocks, the proposed algorithm encodes an image as cubes rather than blocks. The construction of cubes involves the use of highly correlated blocks, with the mean absolute difference determining the correlation between them. The proposed algorithm achieves a higher compression ratio between 120:1 and 20:1, significantly reducing the number of DC coefficients compared to the standard JPEG algorithm, which ranges between 83:1 and 6:1. We observed a difference between real and integer value DCT in terms of PSNR, bit rate, and SSIM, as the coding efficiency and MSE of the approximated integer DCT were very close to the original value DCT. It holds true for higher-order real and integer DCTs. Experimental results reveal that at higher bit rates, the proposed algorithm outperforms the standard JPEG algorithm with a significant PSNR value and comparable SSIM value. We found the maximum PSNR improvement to be between 4 db and 10 db. There is a greater possibility of implementing the proposed algorithm in hardware due to its reduced computational complexity compared to implementing integer DCT. The proposed algorithm is suitable for applications that require a high compression ratio without compromising image quality. The future scope of the work can be used to extract features while using machine learning algorithms.

References

[1] G. K. Wallace, “The JPEG still picture compression standard”, IEEE Transactions on Consumer Electronics Year: 1992, Volume: 38, Issue: 1 Pages: xviii - xxxiv.

[2] Jian-Jiun Ding; Ying-Wun Huang; Pao-Yen Lin; Soo-Chang Pei; Hsin-Hui Chen; Yu-Hsiang Wang, “Two-Dimensional Orthogonal DCT Expansion in Trapezoid and Triangular Blocks and Modified JPEG Image Compression”, IEEE Transactions on Image Processing (Volume: 22, Issue: 9, Sept. 2013, Page(s): 3664 – 3675.

[3] Chang Sun; En-Hui Yang, “An Efficient DCT-Based Image Compression System Based on Laplacian Transparent Composite Model” IEEE Transactions on Image Processing Year: 2015, Volume: 24, Issue: 3 Pages: 886 – 900.

[4] Ahmed N., Natarajan T., Rao K. R., Discrete Cosine Transform, IEEE T. Comput., C-23 (1974), No. 1, 90-93.

[5] Skodras; C. Christopoulos; T. Ebrahimi, “The JPEG 2000 still image compression standard” IEEE Signal Processing Magazine Year: 2001, Volume: 18, Issue: 5 Pages: 36 – 58.

[6] CW. Kok, “Fast algorithm for computing discrete cosine transform,” IEEE Trans. Signal Processing, vol. 45, pp. 757–760. Mar. 1997.

[7] G. Plonka and M. Tasche, “‘Fast and numerically stable algorithms for discrete cosine transforms,” Journal on Linear Algebra and its Applications, vol. 394, pp. 309–345. Jan. 2005.

[8] S.C. Chan and K.L. Ho, “A new two-dimensional fast cosine transform algorithm,” IEEE Trans. Signal Processing, vol. 39, pp. 481–485. Feb. 1991.

[9] H.R. Wu, and F.J. Paoloni, “A two-dimensional fast cosine transform algorithm based on Hou’s approach,” IEEE Trans. Signal Processing, vol. 39, pp. 544–546, Feb. 1991.

[10] E. Feig and S. Winograd, “Fast algorithms for the discrete cosine transform,” IEEE Trans. Signal Processing, vol. 40, pp. 2174–2193, Sep. 1992.

[11] I. Martisius, D. Birvinskas, V. Jusas and Z. Tamosevicius, “A 2-D DCT Hardware Codec based on Loeffler Algorithm,” ELEKTRONIKA IR ELEKTROTECHNIKA, vol. 113, pp. 47-50, Mar. 2011.

[12] A. Edirisuriya, A. Madanayake, R.J. Cintra, V.S. Dimitrov and N. Rajapaksha, “A Single-Channel Architecture for Algebraic Integer-Based 8X8 2-D DCT Computation,” IEEE Trans. Circuits and systems on video technology, vol. 23, pp. 2083-2089, June. 2013.

[13] Tien-Ying, K & Chen-Hung, C 2006, 'Fast variable block size motion estimation for H.264 using likelihood and correlation of motion field', IEEE Transactions on Circuits and Systems on Video Technology, vol. 16, no. 10, pp. 1185-1195.

[14] Panusopone, K, Xue, F & Limin, W 2007, 'An efficient implementation of motion estimation with weight prediction for ITU-T H.264 MPEG-4 AVC', IEEE Transactions on Consumer Electronics, vol. 53, no. 3, pp. 974-978.

[15] H.S. Kwak, R. Srinivasan and K.R. Rao, “C-matrix transform', IEEE Trans. Acoustics, Speech, and Signal Processing, vol. ASSP-31, pp. 1304–1307, Jan. 2003.

[16] S.C. Pei and J.J. Ding, “The integer transforms analogous to discrete trigonometric transforms,” IEEE Trans. Signal Processing, vol. 48, pp. 3345–3364, Dec. 2000.

[17] Chan R. K. W., Lee M. C., 3D-DCT Quantization as a Compression Technique for Video Sequences, in Proceedings of the IEEE International Conference on Virtual Systems and Multimedia, Geneva, Switzerland (1997), 188-196.

[18] Bhaskaranand M., Gibson J. D., Distribution of 3D-DCT Coefficients for Video, in Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing, Taipei, Taiwan (2009), 793-796.

[19] Augustin Jacob, Senthilkumar Natarajan “FPGA implementation of optimal 3D-integer DCT structure for video compression” The scientific world journal, September 2015.

[20] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, & Eero P. Simoncelli 2004, ‘Image quality assessment: From Error visibility to structural similarity’, IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612.

[21] María Elena Domínguez-Jiménez, Full spark of even discrete cosine transforms, Signal Processing, vol. 176, 2020, Article 107632.

[22] Zihan Yuan, Decheng Liu, Xueting Zhang, Qingtang Su, New image blind watermarking method based on two-dimensional discrete cosine transform Optik, vol. 204, 2020, Article 164152.

[23] Mattia Aleardi, Discrete cosine transform for parameter space reduction in linear and non-linear AVA inversions, Journal of Applied Geophysics, 2020, Article 104106.

[24] Chao-Yang Pang, Ri-Gui Zhou, Ben-Qiong Hu, WenWen Hu, Ahmed El-Rafei, Signal and image compression using quantum discrete cosine transform, Information Sciences, vol. 473, 2019, pp. 121-141.

[25] Aref Miri, Saeed Sharifian, Shima Rashidi, Madjid Ghods, Medical image denoising based on 2D discrete cosine transform via ant colony optimization, Optik, vol. 156, 2018, pp. 938-948.

[26] Reem A. Alotaibi, Lamiaa A. Elrefaei, Text-image watermarking based on integer wavelet transform (IWT) and discrete cosine transform (DCT), Applied Computing and Informatics, vol. 15, no. 2, 2019, pp. 191-202.

[27] Kanjar De, V. Masilamani, No-reference Image Sharpness Measure using Discrete Cosine Transform Statistics and Multivariate Adaptive Regression Splines for Robotic Applications, Procedia Computer Science, vol. 133, 2018, pp. 268-275.

[28] Gulnawaz Gani, Fasel Qadir, A robust copy-move forgery detection technique based on discrete cosine transform and cellular automata, Journal of Information Security and Applications, vol. 54, 2020, Article 102510.

[29] Bao, Z., Guo, Y., Li, X. et al. A robust image steganography based on the concatenated error correction encoder and discrete cosine transform coefficients. J Ambient Intell Human Comput 11, 1889–1901 (2020).

[30] Xiao-Zhen Li, Wei-Wei Chen & Yun-Qian Wang, Quantum Image Compression-Encryption Scheme Based on Quantum Discrete Cosine Transform, International Journal of Theoretical Physics, vol. 57, 2018, pp. 2904–2919.

[31] Guanghui Ren, Jianan Han, Jiahui Fu & Mingguang Shan, Asymmetric multiple-image interference cryptosystem using discrete cosine transform and conditional decomposition, Optical Review volume 27, 2020, pp. 1–8.

[32] Vakaimalar E, Mala K & Suresh Babu R, Multifocus image fusion scheme based on discrete cosine transform and spatial frequency, Multimedia Tools and Applications, vol. 78, 2019, pp.17573–17587.

[33] M. Thiruveni & D. Shanthi, Efficient VLSI Architecture for 16-Point Discrete Cosine Transform, Proceedings of the National Academy of Sciences, India Section A: Physical Sciences, vol. 90, 2020, pp. 27–37.

[34] Zahra Moghaddasi, Hamid A. Jalab & Rafidah Md. Noor, Image splicing forgery detection based on low-dimensional singular value decomposition of discrete cosine transform coefficients, Neural Computing and Applications, vol. 31, 2019, pp. 7867–7877.

[35] Shubhi kansal & Rajiv Kumar Tripathi, Adaptive Geometric Filtering Based on Average Brightness of the Image and Discrete Cosine Transform Coefficient Adjustment for Gray and Color Image Enhancement, Arabian Journal for Science and Engineering, vol. 45, 2020, pp. 1655–1668.

[36] Xue, J.; Yin, L.; Lan, Z.; Long, M.; Li, G.; Wang, Z.; Xie, X. 3D DCT Based Image Compression Method for the Medical Endoscopic Application. Sensors 2021, 21, 1817. https://doi.org/10.3390/s21051817.