Empirical Study of Discrete Cosine Transform on Image Compression.

O. Shoewu[1*]; Olusegun O. Omitola[2]; and Segun O. Olatinwo[3]

[1]Department of Electronics and Computer Engineering, Lagos State University, Epe Campus, Nigeria.
[2]Department of Computer Engineering, Afe Babalola University, Adeo Ekiti, Ekiti State, Nigeria.
[3]Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria.

E-mail: engrshoewu@yahoo.com*
omitolasegun@yahoo.com
segunolatinwo@yahoo.co,uk

ABSTRACT

Image compression can be described as a way of minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. This paper presents the study of image compression using Discrete Cosine Transform (DCT). This work employs DCT method to compress images in other to have high compression ratio and high quality image. However, this study presents a number of merits, namely: reduction in redundancies of the image data so as to transmit data in an efficient form and ability to attach and transfer images over the web or over the network in less time.

(Keywords**:** discrete cosine transform, TCT, image compression, decompression)

INTRODUCTION

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.
As our use of and reliance on computers continues to grow, so too does our need for efficient ways of storing large amount of data. The issue of efficient reduction in size of data and information has led to the research and development of software tools that attempts to provide a solution to these problems. These systems and tools are based on the compression's technology.

Compression is an act of reducing the size of data. In this project, image compression using Discrete Cosine Transform (DCT) will be our study.

A common characteristic of most images is that they contain some redundant information. Image compression is necessary to reduce the number of bits needed to represent an image by removing the redundancies as much as possible. The methods of image compression are categories into two (2) major classes which are:

1. Lossy Compression.
2. Lossless Compression.

In this study, DCT will be the method to use for the compression of the images which is under the category of Lossy Compression method.

JPEG process is widely used form of Lossy image compression that center's around the DCT that is going to be use in this study. The DCT works by separating images into parts of differing frequencies during a step called QUANTIZATION where part of compression actually occurs, the less important frequencies are discarded, hence the use of the term LOSSY. Then only the most important frequencies are used to retrieve the image in the decompression process. As a result, reconstructed image contain some distortion. The level of distortion is adjustable [1].

The usual steps involved in using DCT as a method of image compression are:

1. The image is broken down into 10 x 10 block of pixel.

2. Working from left to right, top to bottom, the Discrete Cosine Transform is applied to each block.

3. Each block is compressed through Quantization.

4. The array of compressed blocks that constitute the image is stored in a drastically reduced amount of space.

5. The image is re-constructed through the decompression, a process that uses the Inverse Discrete Cosine Transform (IDCT).

METHODS AND MATERIALS

System Requirements Analysis

Image compression is necessary to reduce the number of bits needed to represent an image by removing the redundancies as much as possible. For example, someone with a web page or online catalog that uses dozens or perhaps hundreds of images will more than likely need to do some form of image compression to store those images.

In this section, an attempt was made to show that Discrete Cosine Transform is a very good method of storing those images [19].

The requirements for implementing DCT are: the Discrete Cosine Transform code (program), the Discrete Cosine Transform Equation and the Quantization matrix. These requirements will be discussed in the following sections

Discrete Cosine Transform Code

The Discrete Cosine Transform code can be developed and implemented by using capable languages like Visual Basic, JAVA or C++. Discrete Cosine Transform perform the following operations in building a robust system: creation, that is, breaking down of images into any desire blocks of pixels, storing the array of the compressed blocks that constitute the image in a drastically reduced amount of space and image reconstruction through a process that uses the IDCT.

Discrete Cosine Transform Equation

The second component of the Discrete Cosine Transform requirement is the DCT equation. In order for the program to run, DCT equation must be implemented.

The DCT equation (Eq. 1) computes the i,jthentry of the DCT of an image.

$$\mathrm{D}(i,j) = \frac{1}{\sqrt{2N}}\mathrm{C}(i)\mathrm{C}(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} p(x,y)\cos\left[\frac{(2x+1)i\pi}{2N}\right]\cos\left[\frac{(2y+1)j\pi}{2N}\right] \quad \textbf{1}$$

$$\mathrm{C}(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases} \quad \textbf{2}$$

$p(x,y)$ is the x,yth element of the image represented by the matrix p. N is the size of the block that the DCT is done on. The Equation calculates one entry (i,j^{th}) of the transformed image from the pixel values of the original image matrix. For the study, 10x10 block is what we will be using, N equals 10 and r and y range from 0 to 9.Therefore D (i,j) would be as in Equation (3).

$$D(i,j) = \frac{1}{4}C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7} P(x,y)\cos\left[\frac{(2x+1)i\pi}{20}\right]\cos\left[\frac{(2y+1)j\pi}{20}\right] \quad \textbf{3}$$

Because the DCT uses cosine functions, the resulting matrix depends on the horizontal, diagonal and vertical frequencies. Therefore an image black with a lot of change in frequency has a very random looking resulting matrix, while an image matrix of just one color has a resulting matrix of a large value for the first element and zeroes for the other elements.

THE DCT MATRIX

To get the matrix form of Equation (1) we will use the following equation:

$$T_{i,j} \quad \begin{cases} \frac{1}{N} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & \text{if } i = 0 \end{cases}$$

For a 10x10 block it results in this matrix:

The first row (i=l) of the matrix has all the entries equal to $\frac{1}{\sqrt{10}}$ as expected from Equation (4).

The columns of T form an orthonormal set, so T is an orthogonal matrix. When doing the inverse DCT the orthogonality of T is important, as the inverse of T is T^{t} which is easy to calculate.

DOING THE DCT ON A 10X10 BLOCK

It should be noted that the pixel values of a black-and-white image range from 0 to 1023 in steps of 1, where pure black is represented by 0 and pure white by 1023. Thus it can be seen how a photo, illustration, etc. can be accurately represented by these 1024 shades of gray. Since an image comprises hundreds or even thousands of 10x10 blocks of pixels, the following description of what happens to one 10x10 block is a microcosm of the JPEG process; what is done to one block of image pixels is done to all of them, in the order earlier specified. Starting with a block of image-pixel values. This particular block was chosen from the very upper-left hand corner of an image.

$$Original = \begin{pmatrix} 554 & 523 & 523 & 523 & 509 & 509 & 509 & 531 & 600 & 524 \\ 592 & 580 & 536 & 554 & 507 & 527 & 570 & 521 & 500 & 526 \\ 654 & 598 & 554 & 580 & 514 & 541 & 560 & 511 & 499 & 528 \\ 639 & 580 & 536 & 554 & 514 & 518 & 550 & 541 & 573 & 530 \\ 580 & 554 & 536 & 567 & 521 & 522 & 540 & 515 & 523 & 532 \\ 528 & 536 & 523 & 536 & 519 & 521 & 530 & 521 & 554 & 534 \\ 523 & 505 & 510 & 549 & 513 & 515 & 520 & 574 & 521 & 536 \\ 510 & 512 & 523 & 523 & 501 & 519 & 510 & 551 & 531 & 538 \\ 523 & 591 & 554 & 566 & 504 & 524 & 510 & 541 & 525 & 540 \\ 534 & 521 & 539 & 554 & 521 & 573 & 541 & 518 & 527 & 522 \end{pmatrix}$$

Because the DCT is designed to work on pixel values ranging from –128 to 127, the original block is "leveled off" by subtracting 512 from each entry. This results in the following matrix:

$$M = \begin{pmatrix} 42 & 11 & 11 & 11 & .3 & .-3 & -3 & 19 & 88 & 12 \\ 80 & 68 & 24 & 42 & -5 & .15 & 58 & 9 & -12 & 14 \\ 142 & 86 & 42 & 68 & 2 & 29 & 48 & -1 & -13 & 16 \\ 127 & 68 & 24 & 42 & 5 & 6 & 38 & 29 & 61 & 18 \\ 68 & 42 & 24 & 55 & 9 & 10 & 28 & 3 & 11 & 20 \\ 16 & 24 & 11 & 24 & 7 & 9 & 18 & 9 & 42 & 22 \\ 11 & -7 & -2 & 37 & 1 & 3 & 8 & 62 & 9 & 24 \\ -2 & 0 & 11 & 11 & -11 & 7 & -2 & 39 & 19 & 26 \\ 11 & 79 & 42 & 54 & -8 & 12 & -2 & 29 & 13 & 28 \\ 22 & 9 & 27 & 42 & 9 & 61 & 29 & 6 & 15 & 10 \end{pmatrix}$$

We are now ready to perform the Discrete Cosine transform, which is accomplished by matrix multiplication.

$$D = TMT^{l}$$

In Equation (5) matrix M is first multiplied on the left by the DCT matrix *T* from the previous section; this transforms the rows. The columns are then transformed by multiplying on the right by the transpose of the DCT matrix. This yields the following matrix.

$$D = \begin{pmatrix} 249.9 & 72.7 & 74.8 & 10.3 & -54.5 & 19.3 & 44.9 & 21.2 & -42.7 & -42.1 \\ 31.3 & 66.9 & 57.0 & 54.0 & 19.8 & 16.8 & -2.7 & 16.5 & -26.9 & 36.5 \\ -6.1 & -1.1 & -12.8 & -13.8 & -7.3 & -20.4 & -18.8 & 16.3 & 1.9 & -7.3 \\ -56.0 & -43.1 & -16.6 & -4.3 & -33.8 & 11.4 & -23.3 & -15.5 & 24.9 & 4.1 \\ -26.8 & -48.9 & -23.1 & -20.7 & 14.6 & 16.1 & -30.9 & 25.8 & -20.3 & 14.4 \\ 16.2 & -32.6 & 22.5 & 20.8 & -29.4 & 3.3 & -6.3 & 2.1 & -25.5 & 31.8 \\ -1.3 & -42.1 & 5.1 & 15.9 & 6.8 & 48.5 & -7.1 & 26.8 & 16.8 & 9.1 \\ 19.2 & 10.0 & 27.7 & -11.9 & -10.2 & 7.3 & -27.7 & 3.7 & -12.6 & -2.1 \\ 8.6 & -10.6 & -18.2 & -8.3 & -5.7 & 20.2 & 12.5 & 10.6 & 18.0 & -18.3 \\ 6.9 & 29.3 & -9.9 & -11.8 & 0.8 & -0.4 & 28.8 & -13.1 & 7.7 & -20.4 \end{pmatrix}$$

This block matrix now consists of 100 DCT coefficients, *cy*, where *i* and *j* range from 0 to 9. The top-left coefficient, c_{00}, correlates to the low frequencies of the original image block. As we move away from c_{00} in all directions, the DCT coefficients correlate to higher and higher frequencies of the image block, where c_{99} corresponds to the highest frequency. It is important to note that the human eye is most sensitive to low frequencies, and results from the quantization step will reflect this fact.

QUANTIZATION

Our 10x10 block of DCT coefficients is now ready for compression by quantization. A remarkable and highly useful feature of the JPEG process is that in this step, varying levels of image compression and quality are obtainable through selection of specific quantization matrices. This enable the user to decide on the quality levels ranging from 1 to 100, where 1 gives the poorest image quality and highest compression, while 100 gives the best quality and lowest compression. As a result, the quality/compression ratio can be tailored to suit different needs.

Subjective experiments involving the human visual system have resulted in the JPEG standard quantization matrix. With a quality level 50, this matrix renders both high compression and excellent the compressed image quality.

$$Q_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 & 71 & 81 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 65 & 50 & 45 \\ 14 & 13 & 26 & 24 & 40 & 57 & 69 & 56 & 43 & 30 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 & 64 & 32 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 & 70 & 63 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 & 81 & 76 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 107 & 100 & 86 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 & 105 & 94 \\ 80 & 104 & 108 & 117 & 124 & 131 & 128 & 121 & 100 & 82 \\ 84 & 122 & 131 & 133 & 146 & 137 & 140 & 138 & 95 & 86 \end{pmatrix}$$

If however, another level of quality and compression is desired, scalar multiples of the JPEG standard quantization matrix may be used. For a quality level greater than 50 (less compression, higher image quality), the standard quantization matrix is multiplied by (100-quality level)/50. For a quality less than 50 (more compression, lower image quality), the standard quantization matrix is multiplied by 50/quality level.

The scaled quantization matrix is then rounded and clipped to have positive integer values ranging from 1 to 255.

Quantization is then achieved by dividing each element in the transformed image matrix *D* by the corresponding element in the quantization matrix, and then rounding to the nearest integer value. In this study, quantization matrix level 50 (i.e., Q_{50}) is used.

$$C_{ij} = round\left[\frac{Dij}{Qij}\right]$$

$$C = \begin{pmatrix} 16 & 7 & 8 & 1 & -2 & 1 & 1 & 0 & -1 & -1 \\ 3 & 6 & 4 & 5 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & -3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Note that the coefficient situated near the upper-left corner correspond to the lower frequencies – to which the human eye is most sensitive – of the image block. In addition, the zeros represent the less important, higher frequencies that have been discarded, giving rise to the Lossy part of compression. As mentioned earlier, only the remaining nonzero coefficients will be used to reconstruct the image.

CODING

The quantized matrix *C* is now ready for the final step of compression. Before storage, all coefficients of *C* are converted by an encoder to a stream of binary data (01101011…). In-depth coverage of the coding process is beyond the scope of this project work.

However, we can point out one key aspect that the reader is sure to appreciate. The advantage lies in the consolidation of relatively large runs of zeros, which compress very well.

DECOMPRESSION

Reconstruction of our image begins by decoding the bit stream representing the quantized matrix *C*. each element of *C* is then multiplied by the corresponding element of the quantization matrix originally used.

$$R_{ij} = Q_{ij} \, x \, C_{ij}$$

$$R_{ij} = -70 + \begin{pmatrix} 339 & 210 & 228 & 395 & 715 & 1328 & 1627 & 1721 & 1740 & 1751 \\ 348 & 386 & 456 & 592 & 857 & 1377 & 1432 & 1322 & 1170 & 942 \\ -28 & -30 & -38 & -53 & -91 & -144 & -149 & -118 & -107 & -62 \\ -132 & -115 & -135 & -201 & -282 & -500 & -556 & -572 & -542 & -552 \\ -59 & -64 & -94 & -140 & -253 & -291 & -308 & -285 & -282 & \\ 4 & -1 & -4 & -3 & -2 & -18 & -9 & -4 & 21 & 36 \\ -12 & -12 & -14 & -19 & -26 & -58 & -60 & -65 & -50 & -45 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The IDCT is next to applied to matrix *R,* which is rounded to the nearest integer. Finally, 512 is added to each element of that result, giving us the decompressed JPEG version *N* of our original 10x10 image block M.

$$N = round(T^{l} RT) + 512$$

$$N = \begin{pmatrix} 1749 & -130 & 417 & 478 & 528 & 482 & 508 & 548 & 506 & 503 \\ 2705 & -482 & 338 & 456 & 5948 & 452 & 512 & 574 & 511 & 478 \\ 3176 & -907 & 325 & 439 & 541 & 436 & 525 & 586 & 518 & 460 \\ 2626 & 679 & 415 & 449 & 549 & 455 & 530 & 568 & 513 & 476 \\ 1354 & -91 & 560 & 494 & 546 & 497 & 530 & 501 & 510 & 508 \\ 202 & 455 & 667 & 551 & 530 & 532 & 531 & 498 & 514 & 535 \\ -226 & 392 & 804 & 560 & 530 & 548 & 540 & 479 & 1204 & 535 \\ 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 \\ 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 \\ 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 \end{pmatrix}$$

COMPARISON OF MATRICES

$$Original = \begin{pmatrix} 554 & 523 & 523 & 523 & 509 & 509 & 509 & 531 & 600 & 524 \\ 592 & 580 & 536 & 554 & 507 & 527 & 570 & 521 & 500 & 526 \\ 654 & 598 & 554 & 580 & 514 & 541 & 560 & 511 & 499 & 528 \\ 639 & 580 & 536 & 554 & 514 & 518 & 550 & 541 & 573 & 530 \\ 580 & 554 & 536 & 567 & 521 & 522 & 540 & 515 & 523 & 532 \\ 528 & 536 & 523 & 536 & 519 & 521 & 530 & 521 & 554 & 534 \\ 523 & 505 & 510 & 549 & 513 & 515 & 520 & 574 & 521 & 536 \\ 510 & 512 & 523 & 523 & 501 & 519 & 510 & 551 & 531 & 538 \\ 523 & 591 & 554 & 566 & 504 & 524 & 510 & 541 & 525 & 540 \\ 534 & 521 & 539 & 554 & 521 & 573 & 541 & 518 & 527 & 522 \end{pmatrix}$$

$$De\text{-}compressed = \begin{pmatrix} 1749 & -130 & 417 & 478 & 528 & 482 & 508 & 548 & 506 & 503 \\ 2705 & -482 & 338 & 456 & 5948 & 452 & 512 & 574 & 511 & 478 \\ 3176 & -907 & 325 & 439 & 541 & 436 & 525 & 586 & 518 & 460 \\ 2626 & 679 & 415 & 449 & 549 & 455 & 530 & 568 & 513 & 476 \\ 1354 & -91 & 560 & 494 & 546 & 497 & 530 & 501 & 510 & 508 \\ 202 & 455 & 667 & 551 & 530 & 532 & 531 & 498 & 514 & 535 \\ -226 & 392 & 804 & 560 & 530 & 548 & 540 & 479 & 1204 & 535 \\ 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 \\ 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 \\ 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 & 512 \end{pmatrix}$$

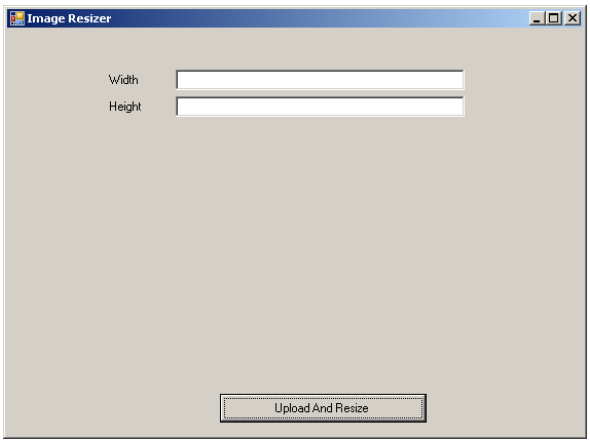
This is not a remarkable result, considering the expectation of the output according to the reviewed DCT with 8x8 matrices with which 70% of the DCT coefficients were discarded prior to image block decompression/reconstruction.

Here, the decompressed matrix has higher co-efficient than the original matrix which in essence means that DCT coefficients were not discarded. Also in the decompressed matrix we have negative figures which is going to give a bad image if used to compressed an image.

SYSTEM IMPLEMENTATION

Program Module

For convenience purpose, debugging and referencing, the program is divided into functions of manageable sizes. Some of the tasks performed are explained below:
Opening Screen: This is used to tell you what the topic is all about and used to display specified test. It then asks you for the password to continue.



- Width and Height: Here the user is expected to put the desired Width and Height of the output. What should be of note here is that even if the values put into this columns are as exact as the original image, due to the compression going on, a noticeable reduction in size is observed.

- Upload and Resize: The user is expected to choose this button to browse through the computer to choose the picture image to work on.

RESULTS

After the implementation of the system program, the developed system was tested and was observed to have a good quality while slightly compressing the image. The size of the output is noticed to be smaller than the input even at same resolution.

231kb compressed to 126kb

SYSTEM PERFORMANCE ANALYSIS

After the program has been successfully implemented, the output of the image is then compared to see if it meets the required constraints such as reduction in redundancies of the image data in order to be able to transmit data in an efficient form, storage need and ability to attach and transfer images over the web or over the network in less time. It was discovered that the developed system is efficient for image compression.

CONCLUSION

The Discrete Cosine Transform (DCT) is widely used in image compression algorithm, due to its energy compaction for correlated image pixels. The basis function for the DCT is the cosine, a real function that is easy to compute. The DCT is a unitary transform, and the sum of the energy in the transform and spatial domains is the same.

In this work, an attempt was made to prove that a DCT of higher matrix is better for compression but has to be run in binaries alone i.e. the matrix have to be 2^2, 2^3 or 2^4, etc.

DCT can offer a new paradigm for image compression. A number of advantages of using DCT have been identified such as ease of transmission of large images over the web, maximum usage of resources and transmission of images over the network in lesser time.

RECOMMENDATIONS

From this study, the following recommendations were made:

- This software can be used by organizations into image compression.
- It can be used for graphic works for good quality and high compression.

Due to its multi-resolution zooming capacities, it is then recommended for the following people:

- People doing a lot of video coding in other to present good images and videos with high graphics (i.e., High Density Television (HDTV)).
- People into high graphics games (e.g., Play Station 3®, XBOX 360®, etc.
- People into production of high resolution wallpapers and screen savers.

REFERENCES

1. Jacquin, A.E. 1992. “Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformation”. *IEEE Transaction on Image Processing*.1(1):18-30.
2. Jain, A.K. 1995. *Fundamentals of Digital Image Processing*. PHI: New Delhi, India.
3. Fisher, Y. 1995. *Fractal Image Compression: Theory and Application*. Springer Verlag: New York, NY.
4. Barnsley, M.F. 1996. “Fractal Image Compression”. *Notices of the AMS*. June:657-662.
5. Netravali, A.N. and B.G. Haskell. 1995. *Digital Pictures: Representation, Compression, and Standards (2nd Ed)*. Plenum Press: New York, NY.
6. Mandelbrot, B.B. 1982. *Fractal Geometry of Nature*. W.H. Freeman and Co.: New York, NY.
7. Mitra, S.K., C.A. Murthy, and M.K. Kundu. 2000. “Partitioned Iterated Function System: A New Tool for Digital Imaging”. *IETE Journal of Research*. 16(5) Sep-Oct:279-298.
8. Conzalez, R. and P. Wintz. 1987. *Digital Image Processing*. Addison-Wesley Publishing Company: New York, NY.
9. Saupe, D. and S. Jacob. 1997 “Variance Based Quad-Trees in Fractal Image Compression”. *Electronic Letters*. 33(1):46-48.
10. Soyjauadha, K.M.S. and I.J. Bacus. 2002. “Fractal Image Compression”. *International Journal of electrical Engineering Education*. Jan.2002.
11. http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html.
12. Barnsley, M.F. and L.P. Hurd. 1993. *Image Compression*. AK Peters, Ltd,: Wellesley, MA.
13. Said, A. and W.A. Pearlman. 1996. “An Image Multi-resolution Representation for Lossless and Lossy Image Compression”. *IEEE Transactions on Image Processing*. 5:1303-1310.

14. Castleman, K. 1996. *Digital Image Processing*. Prentice Hall: New York, NY.

15. Cherkassky, V. and F. Mulier. 1998. *Learning from Data*. John Wiley and Sons: New York, NY.

16. Said, A. and W.A. Pearlman. 1996. “A New Fast and Efficient Image Code Based on Set Partitioning in Hierarchical Trees”. *IEEE Transactions on Circuits and Systems for Video Technology*. 6:243-250.

17. Sikora, T. 2005. “Trends and Perspectives in Image and Video Coding”. *Proceedings of the IEEE*. 93(1):6–17.

18. Cho, N.I. and S.K. Mitra. 2000. “Warped Discrete Cosine Transform and its Application in Image Compression”. *IEEE Transactions on Circuits and Systems for Video Technology*. 10(8):1364–1373.

19. Bruckstein, A.M., M. Elad, and R. Kimmel. 2003. “Down-Scaling for Better Transform Compression”. *IEEE Transactions on Image Processing*. 12(9):1132–1144.

20. Tsaig, Y., M. Elad, P. Milanfar, and G.H. Golub. 2005. “Variable Projection for Near-Optimal Filtering in Low Bit-Rate Block Coders”. *IEEE Transactions on Circuits and Systems for Video Technology*. 15(1):154–160.

21. Gonzales, R. and R. Woods. 1992. *Digital Image Processing*. Addison-Wesley: Reading, MA.

22. Tekalp, A.M. 1995. *Digital Video Processing*. Prentice-Hall: Upper Saddle River, NJ.

SUGGESTED CITATION

Shoewu, O., O.O. Omitola, and S.O. Olatinwo. 2014. “Empirical Study of Discrete Cosine Transform on Image Compression”. *Pacific Journal of Science and Technology*. 15(1):130-140.

Pacific Journal of Science and Technology