Edge-Adaptive Jpeg Image Compression Using MATLAB

DR. THIDA AUNG
Professor and Head, Department of Electronic Engineering, Technological University (Lashio), Lashio, Myanmar

Abstract- ***In this paper, Edge-adaptive JPEG image compression is designed and implemented. The standard JPEG is a very popular image compression method results in a good compression with low computational complexity and memory requirements. JPEG gives users the ability to take an image and compresses it with little or no noticeable quality degradation. Compression is useful method because it helps to reduce the consumption of expensive resources such as storage space or transmission bandwidth. Edge-adaptive JPEG image compression is a modified image compression of standard JPEG. This technique achieves higher visual quality and smaller image data size than standard JPEG at the same bit rate. Edge-adaptive JPEG achieves compression at variable bit rates with minimal loss in visual quality. Visual activity is measured using a Canny edge detector; the image is segmented with quadtree decomposition and apply DCT (discrete cosine transform) to each of the channels. The performance of image compression is simulated and analyzed by using MATLAB.***

Indexed Terms- ***Edge-adaptive JPEG, standard JPEG, image compression, MATLAB***

I. INTRODUCTION

Digital image compression algorithms have become increasingly popular due to the need to achieve cost-effective solutions in transmitting and storing images. The JPEG compression algorithm is at its best on photographs and paintings of realistic scenes with smooth variations of tone and color. For web usage, where the amount of data used for an image is important, JPEG is very popular. JPEG/Exif is also the most common format saved by digital cameras. JPEG may not be as well suited for line drawings and other textual or iconic graphics, where the sharp contrasts between adjacent pixels can cause noticeable artifacts. Such images may be better saved in a lossless graphics format such as TIFF, GIF, PNG, or a raw image format. The edge-adaptive JPEG actually includes a lossless coding mode, but that mode is not supported in most products.

As the typical use of edge-adaptive JPEG is a lossy compression method, which somewhat reduces the image fidelity. It should not be used in scenarios where the exact reproduction of the data is required such as some scientific and medical imaging applications and certain technical image processing work. Edge-adaptive JPEG is also not well suited to files that will undergo multiple edits, as some image quality will usually be lost each time the image is decompressed and recompressed, particularly if the image is cropped or shifted, or if encoding parameters are changed. To avoid this, an image that is being modified or may be modified in the future can be saved in a lossless format, with a copy exported as JPEG for distribution.

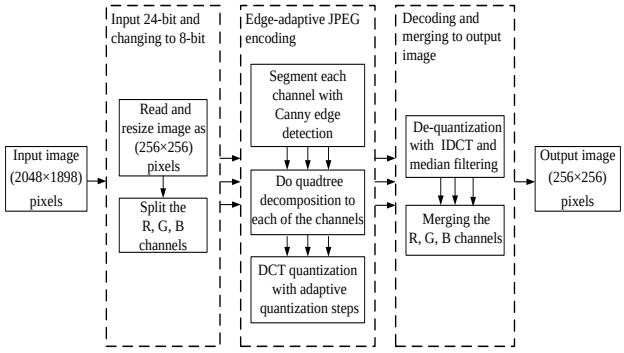


Figure 1. Block Diagram of the System

In this system, resize 24-bit input image, split to the (red, green, blue) channels, Canny edge detection, quadtree decomposition, discrete cosine transform quantization, de-quantization with inverse discrete cosine transform, median filtering and merging to the (red, green, blue) channel images are included.

II. EDGE-ADAPTIVE JPEG IMAGE COMPRESSION

Edge-adaptive JPEG is a modified JPEG algorithm that provides better visual quality than the Q-factor scaling method commonly used with JPEG implementations. The quantization step sizes are adapted to the activity level of the block and the activity selection is based on an edge-driven quadtree decomposition of the image.

Most image software uses the simpler JFIF format when creating a JPEG file, which among other things specifies the encoding method. Here is a brief description of one of the more common methods of encoding when applied to an input that has 24 bits per pixel (eight each of red, green, and blue). This particular option is a lossy data compression method. Figure 2 shows the JPEG encoder and decoder diagram.

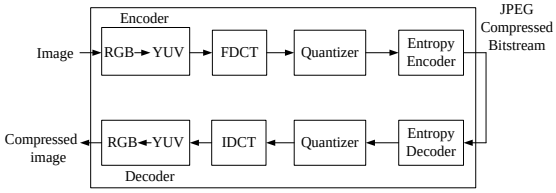


Figure 2. JPEG Encoder and Decoder Diagram

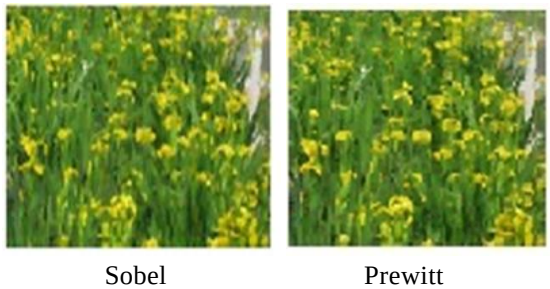
A. Edge Detection Techniques

Edge detection is one of the most commonly used operations in image analysis. Edge detection identifies locations in an image where the intensity changes rapidly, that is, intensity boundaries. Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Image Edge detection significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. Since edge detection is in the forefront of image processing for object detection, it is crucial to have a good understanding of edge detection algorithms. An edge is defined by a discontinuity in gray level values.

An edge is the boundary between an object and the background. The shape of edges in images depends on many parameters: the geometrical and optical properties of the object, the illumination conditions, and the noise level in the images. Edge detection is important data reduction step since it encodes information based on the structure of the image. Using edge detection vital information of the image is preserved while keeping aside less important information that effectively reduces dynamic range of the image and eliminates pixel redundancy. Edge detectors may well be classified into Gradient edge Detector, Laplacian Method and Gaussian edge detector.

B. Comparison of Edge Detection Methods

In the edge-adaptive JPEG image compression algorithm, it is contained various edge detection methods such as Sobel, Prewitt, Robert, Laplacian of Gaussian and Canny edge detection. Among all of these edge detections, Canny edge detector can detect specially in noise conditions and it has low error rate than other edge detectors. Figure 3 shows the edge-adaptive JPEG images with various edge detectors in image compression.



Sobel Prewitt

Robert Laplacian

Canny

Figure 3. Edge-adaptive JPEG Images with Various Edge Detector

C. Edge-adaptive Quantization

Edge-adaptive JPEG is a modified JPEG image compression method. Visual activity is measured using a Canny edge detector and the image is segmented into different activity regions via quadtree decomposition. This decomposition is used to identify the uniform quantizer step sizes used in various regions of the image. Quadpression, is based on an edge-driven quadtree decomposition of the image. The quadpression approach gives consistently higher visual quality at the same bit rates than the Q-factor technique with a very small overhead. For a 512×512 image, the smallest sub-image is 8×8.

D. Quadtree Decomposition

After the image edges are identified by the Canny edge detector, the image is segmented into a quadtree decomposition. A quadtree decomposition decomposes a $2^N \times 2^N$ image into a n-level hierarchy, where all sub-images at level n have a size $2^n \times 2^n$, $n \leq N$. This structure corresponds to a tree, where each $2^n \times 2^n$ sub-image can either be a leaf if it is not further subdivided or can branch into four $2^{n-1} \times 2^{n-1}$ more sub-images, each a child node. To decompose an image or a sub-image at each level of the tree, a binary decision is made to determine whether it needs to be divided into four more sub-images. This procedure is then repeated recursively on each resulting sub-image.

Figure 4. Quadtree Decomposition for the Lena Image competence.

The quadtree decomposition is combined with the results of Canny edge detection to decompose each sub-image using the presence of edges in the sub-image as the binary decision to further subdivide the sub-image or not. If more than t% of the pixels in the sub-image is identified as edges, the sub-image is then split into four more sub-images. Small sub-images correspond to very active areas while large sub-images correspond to less active regions of the image. Very active images will have a quadtree decomposition containing many small blocks, while less active images will contain large blocks corresponding to flatter areas of the image. Figure 4 shows the quadtree segmentation of the Lena image with its edges identified. Each quadtree sub-image is represented by the mean value of the image intensity.

E. Quadpression

With the image segmented into sub-images corresponding to different activity areas, the image is compressed using a modified version of JPEG. For each 8 × 8 DCT block, the step sizes of the quantization matrix are scaled according to the size of the quadtree sub-image that contains the 8 × 8 DCT block. The scaling must also be coded with the image and consist of a maximum of six integer values for the six sub-image sizes corresponding to the six levels of the quadtree decomposition. The DC coefficient is kept intact to avoid large blocking effects and to smooth out the errors across the entire image. This is in contrast to the Q-factor approach, where the entire matrix is scaled.

Scales larger than 8 improve the compression ratios but it leads to much distorted images. So integers scaling ranging from 1 to 8 were adopted. A scaling of 1 for all sub-image sizes corresponds to the JPEG standard with the recommended matrices. Quadpression method can be used to assign the scaling for the different activity sub-images. First, it assigns large scaling to the larger sub-images and subsequently smaller scaling to the smaller sub-images. The same scaling is used for all the sub-images that have the same size. The user can select the scaling that will result in the desired bit rate. Although this method has the disadvantage that the scaling are not optimized for the particular bit rate,

it is a fast alternative when the user is simply interested in improving the image quality and the overall distribution of blockiness that arises when the same matrix is used for all the blocks in the image.

III. THE SYSTEM IMPLEMNETATION

In the edge-adaptive JPEG image compression algorithm, it is divided into four categories;

- Input and Changing to 8-bit
- Edge Adaptive JPEG Encoding
- Edge Adaptive JPEG Decoding
- Merging to the Output Image

Firstly, the original R, G, B color image must input to compress the image with edge-adaptive JPEG image compression. Figure 5 shows the lotus R, G, B original image. This image is 2048×1898 pixels and image format is jpeg. Figure 6 shows the flowchart of the overall system.

Figure 5. Lotus R, G, B Original Image (2048×1898) Pixels

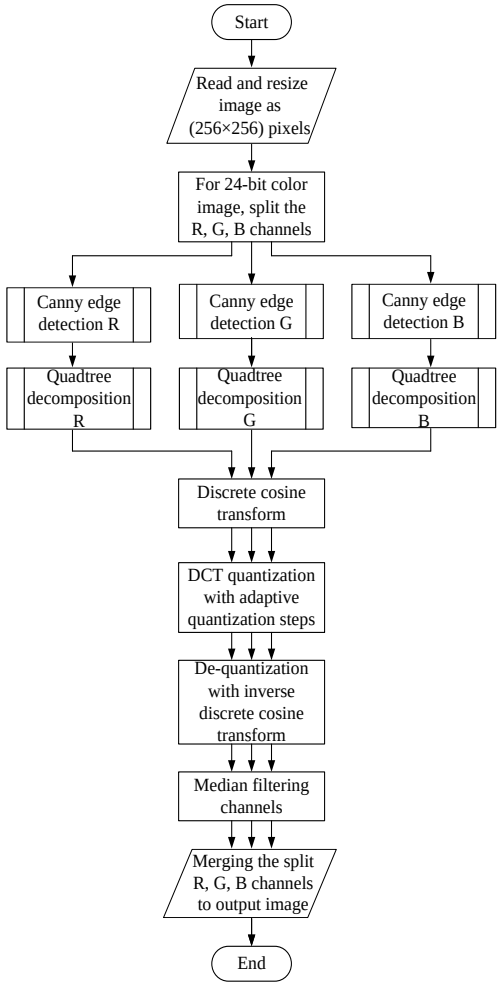


Figure 6. The Flowchart of Overall System

A. Input and Changing to 8-bit

Firstly, the original color input image is read. This image is 2048×1898 pixels and 24-bit color image. Then, resize the input image as 256×256 pixels. Split the red, green, blue channels image from 24-bit to three 8-bit images. This stage reduces the image data size in any image compression algorithm.

B. Edge Adaptive JPEG Encoding

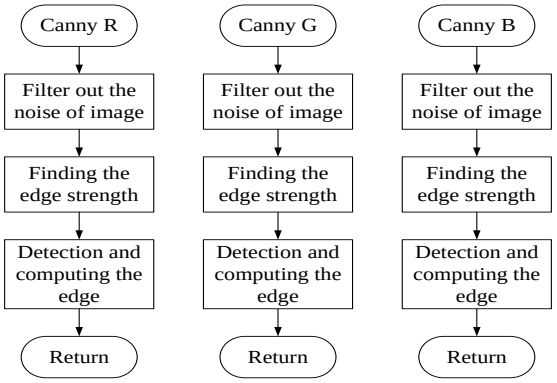
The edge-adaptive JPEG encoding process contains the following steps

- Canny edge detection
- Quadtree decomposition
- Discrete cosine transform
- DCT quantization with adaptive quantization steps

The split images are segmented and detected with Canny edge detection. Quadtree decomposition is

decomposed to each of the split images. The discrete cosine transform and DCT quantization with adaptive quantization steps are applied to each of the channels.

C. Canny Edge Detection



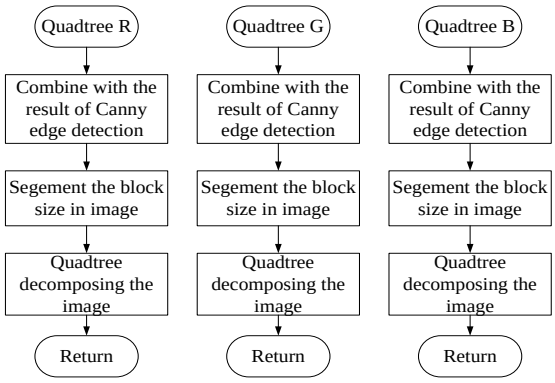
(a) (b) (c)

Figure 7. Flowcharts of (a) Red (b) Green (c) Blue Channels in Canny Edge Detector

In order to implement the canny edge detector algorithm, a series of steps must be followed. The first step is to filter out any noise in the original image before trying to locate and detect any edges. After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. Finally, canny edge detector detects and computes the edges of image. The most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be no responses to non-edges. The theory is explained in the previous chapter. Figure 7 shows the flowcharts of (a) Red (b) Green (c) Blue channels in canny edge detector.

D. Quadtree Decomposition

Quadtree decomposition is an analysis technique that involves subdividing an image into blocks that are more homogeneous than the image itself. This technique reveals information about the structure of the image. It is also useful as the first step in adaptive compression algorithms.



(a) (b) (c)

Figure 8. Flowcharts of (a) Red (b) Green (c) Blue Channels in Quadtree Decomposition

Quadtree decomposition is combined with the result of canny edge detection. It is segmented the block size in image. And then, quadtree decomposing applied the image. The theory is explained in the previous chapter. Figure 8 shows the flowcharts of (a) Red (b) Green (c) Blue channels in quadtree decomposition.

E. Discrete Consine Transform

In the image compression, all pixel values are automatically represented by real numbers. DCT compute two dimensional of image by using 8×8 block for red, green and blue channels. The DCT compression is based on the fact that most natural images have sparse edges. Hence, most blocks contain primarily low frequencies, and can be represented by a small number of coefficients without significant precision loss. Edges are problematic since it is associated with high spatial frequency. Each DCT block selects a different number of coefficients with the largest amplitude. Thus, smooth regions of an image can be represented by a small number of coefficients, whereas edges and high-frequency textures would be represented by large number of coefficients. This will solve the problem of edges, while leaving the algorithm efficient.

F. DCT Quantization with Adaptive Quantization Steps

The quantization step sizes are not homogenously the same. Instead, the quantization step sizes are different for the overall image according to the rate of pixel value changes in each of the quadtree blocks. Choosing the required step size for the specific block

is such that in a certain block. If there is a greatly changes in pixel values, the low quantization step size is chosen. But if there is a little change in pixel values, the larger quantization step size is chosen.

G. Edge Adaptive JPEG Decoding

The edge-adaptive JPEG decoding process contains the following steps:

- De-quantization with IDCT
- Median filtering

De-quantization with inverse discrete cosine transform (IDCT) allows reconstruction of an image frame that had been encoded by the DCT. And hence, it transforms back to the time domain. Median filtering method is used to smooth the red, green, blue channels of image. It needs to be used in order to reduce the amount of noise and texture in the image.

H. Merging to the Output Image

Output image is merged into a 24-bit color image from three 8-bit split channels by using concatenate (cat) function. The cat function is a simple way to build multidimensional arrays; it concatenates a list of arrays along a specified dimension.

I = cat (dim, A1, A2...) Equation 1

Where A1, A2, and so on are the arrays to concatenate and dim is the dimension along which to concatenate the arrays. Output image is (256×256) pixels and 24-bit R, G, B color image.

IV. RESULT ANALYSIS AND DICUSSION

The performance test and results are discussed in detail. MATLAB/GUI tools are supported for the image compression. In the main page of the system, it needs to press the all buttons sequentially. Firstly, user has to choose the image in the choose image button. After choosing the image, it needs to press the load, resize image, split R, G, B channels, show Canny, show quadtree, discrete cosine transform, DCT quantization, inverse discrete cosine transform and edge-adaptive JPEG buttons. The edge-adaptive JPEG output image is merged with 24-bit RGB image after pressing the all buttons. In the choose image button, it is contained Lotus, Lena, Cruise, Lilies and Mountain View images. All this images can be used to start the image compression algorithm with edge-adaptive JPEG.

A. Input and Changing to 8 bits

The original RGB color image must input to start the image compression. The input image is the original Lotus image with 24-bit and also it is 2048×1898 pixels. Figure 9 shows the Lotus image when user chooses the Lotus image in the choose image and then press the load button.

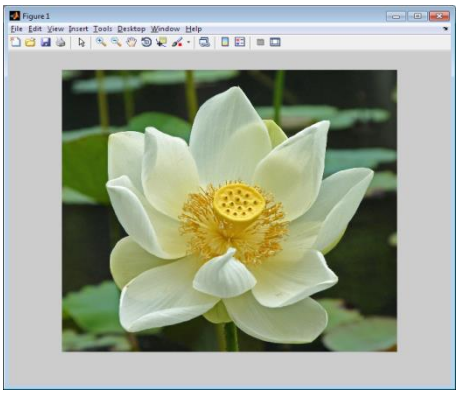


Figure 9. Original Lotus Image (2048×1898) Pixels

Figure 10 shows the resize image of original color image. It is 24-bit RGB and 256×256 pixels color image.

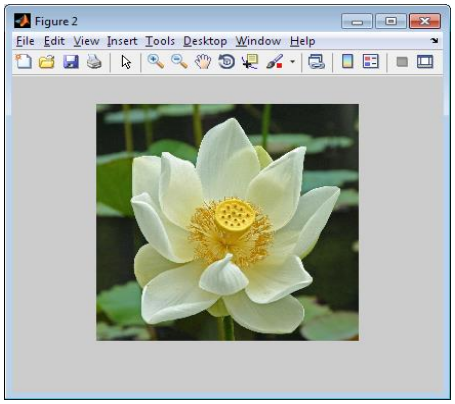


Figure 10. Resize Lotus Image (256×256) Pixels

It is necessary to split three 8-bit images from 24-bit RGB color image. This stage protects from data losses in any RGB color image. Figure 11 shows the three 8-bit split red, green, blue channels image.

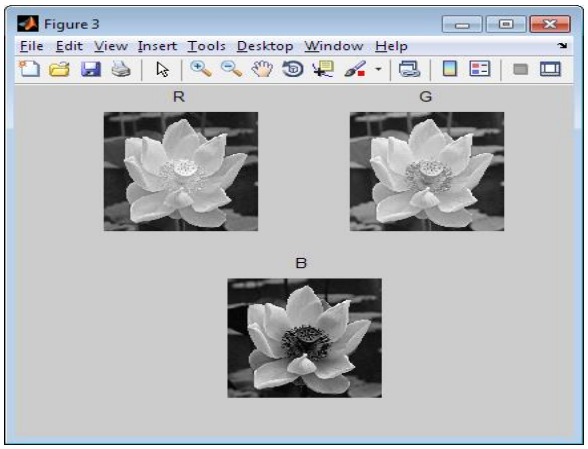


Figure 11. 8-bit Split R, G, B Images

B. Encoding Process in Edge Adaptive JPEG

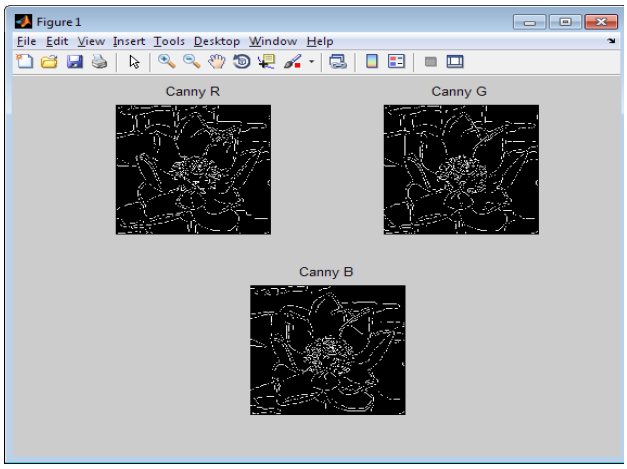


Figure 12. Canny Edge Detection of R, G, B Images

The edge-adaptive JPEG encoding process contains Canny edge detection, quadtree decomposition, discrete cosine transform and DCT quantization with adaptive quantization steps. After splitting the R, G, B channels image, Canny edge detector filters out the noise and detects the edges of image. Figure 12 shows the Canny edge detection of red, green and blue channels image.

Quadtree decomposition decomposes the R, G, B channels image of Canny edge detection. Figure 13 shows the quadtree decomposition of red, green and blue channels image.

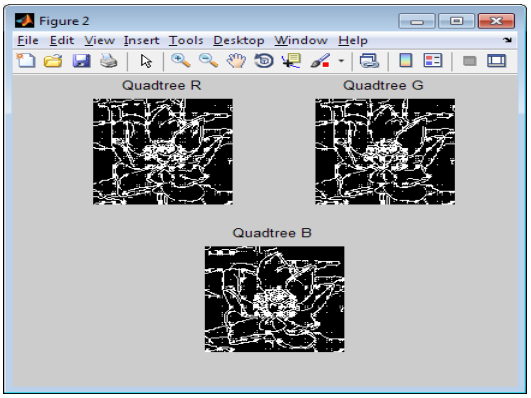


Figure 13. Quadtree Decomposition of R, G, B Images

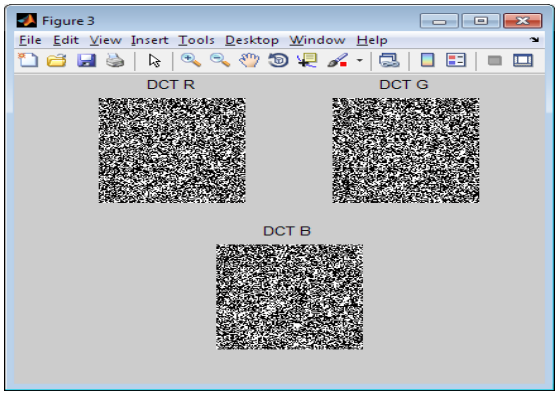


Figure 14. DCT of R, G, B Images

The image is compressed using discrete cosine transform, it is divided into subimages of 8 x 8 pixels to speed up calculations, and then each subimage is transformed and processed separately. Figure 14 shows the discrete cosine transform (DCT) of red, green and blue channels image.

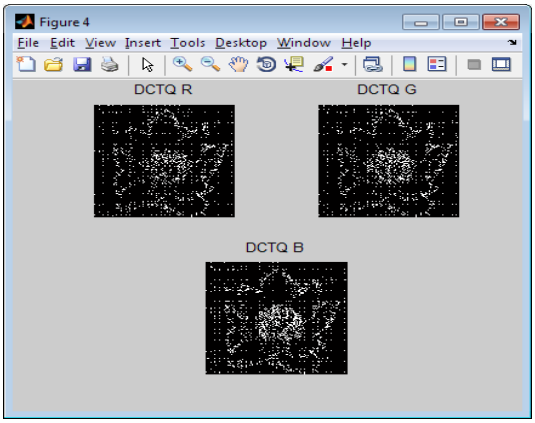


Figure 15. DCT Quantization of R, G, B Images

Quantization steps are quantized separately for the overall image depends on the edges of image. Figure 15 shows the discrete cosine transform (DCT) quantization with adaptive quantization steps of red, green and blue channels image.

C. Decoding Process in Edge Adaptive JPEG

The edge-adaptive JPEG decoding process contains de-quantization with inverse discrete cosine transform (IDCT) and median filtering method. IDCT is the image reconstruction, with each subimage being reconstructed and placed into the appropriate image position. Median filtering method is to filter out the noise and smooth the image. Figure 16 shows the de-quantization with inverse discrete cosine transform of red, green and blue channels image.

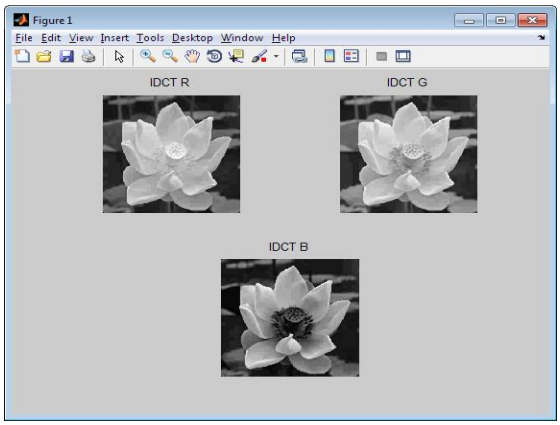


Figure 16. IDCT of R, G, B Images

D. Merging to the Output Image

Output image is merged from three 8-bit split red, green, blue channels image into a 24-bit RGB color image. It is also the output result of edge-adaptive JPEG image compression. Figure 17 shows the edge-adaptive JPEG Lotus image with 24-bit color image.

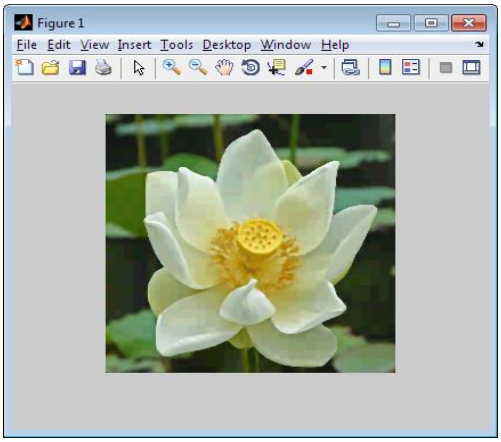


Figure 17. Edge-Adaptive JPEG Lotus Image

E. Comparison of Images in Original Image and Edge-adaptive JPEG Image

Image compression is an application of data compression that encodes the original image with few bits. Edge-adaptive JPEG image compression can give smaller image data size placed on hard disk space than original image. In this section, it is showed images in before compression and after compression.

The edge-adaptive JPEG compression method gives the output images having the smaller image data size while maintaining their visual qualities at high level as normal JPEG.

Table 1. Comparison of Image Compression in JPEG and Edge-Adaptive JPEG

Image Name	Original Image	JPEG	Edge-adaptive JPEG
Lotus 	(2048×1898) Pixels 384 kB	68 kB	56 kB
Lena 	(512×512) Pixels 772 kB	84 kB	60 kB
Lilies 	(600×600) Pixels 1.03 MB	132 kB	92 kB
Mountain View 	(568×568) Pixels 948 kB	96 kB	60 kB

The objective of the image compression is to reduce its data size without much distortion to its original image quality and so it is attained by using edge-adaptive JPEG technique. However, this experiment is not an exception to the image compression theory which states that the more the image is compressed, the more its visual quality is lost. So there are some distortions which are unavoidable factors.

V. CONCLUSION

The best well known image compression methods are predictive coding, orthogonal transform and subband coding. Predictive coding such as DPCM (Differential Pulse Code Modulation) is a lossless coding method, which means that the decoded image and the original image have the same value for every corresponding element. Orthogonal transform such as KLT (Karhunen-Loeve Transform) and DCT (Discrete Cosine Transform) are the two most well-known orthogonal transforms. The DCT based image compression standard such as edge-adaptive JPEG is a lossy coding method that will result in some loss of details and unrecoverable distortion. Subband coding such as DWT (Discrete Wavelet Transform) is also a lossy coding method. The DCT based image compression such as edge-adaptive JPEG performs very well at moderate bit rates than other image compression methods. However, at higher compression ratio, the quality of the image slightly degrades because of the artifacts resulting from the block-based DCT scheme. Since edge detection is the initial step in object recognition, it is necessary to know the differences between edge detection algorithms. Canny edge detection algorithm is computationally more expensive compared to Sobel, Prewitt, Robert and Laplacian operators. However, the Canny edge detection algorithm performs better than all these operators under almost all scenarios. Edge-adaptive JPEG can give smaller image data size than standard JPEG. Finally, it can be concluded that edge-adaptive JPEG is the better method for image compression for the priority of compression ratio while maintaining the visual quality of the original image.

VI. RECOMMENDATION

Further and deeper investigations should be carried out to achieve a better visual quality images. It could be accomplished by extending the current successfully-implemented edge-adaptive JPEG image compression method.

ACKNOWLEDGMENT

The author would like to thank all persons who involved towards the successful completion for this research work.

REFERENCES

[1] Richard H. Wiggins, MD.H. Christian Davidson, MD.H. Ric Harnsberger, MD. Jason R. Lauman, BS. Patricia A. Goede, BS, 2001, vol. 21, “Image File Formats: Past, Present, and Future”.

[2] <http://www.wikipedia.org/wiki/Portable-Network-Graphics>.

[3] Sonal, No date, “A Study of Various Image Compression Techniques”, Dinesh Kumar Department of Computer Science & Engineering Guru Jhambheswar University of Science and Technology, Hisar.

[4] Rafael C. Gonzalez, March 1993, vol.15, “Digital Image Processing”.

[5] <http://www.wikipedia.org/wiki/Image-compression>.

[6] <http://www.wikipedia.org/wiki/Peak-signal-to-noise-ratio>.

[7] <http://www.wikipedia.org/wiki/Mean-squared-error>.

[8] <http://www.wikipedia.org/wiki/JPEG>.

[9] Wei-Yi Wei, No date, “An Introduction to Image Compression”, Graduate Institute of Communication Engineering, National Taiwan University, Taiwan.

[10] Rajwinder Kaur, Monika Verma, Kalpna and Harish Kundra, No date, “Classification of Various Edge Detectors”, Department of Computer Science, RIEIT, Railmajra.

[11] Marcia G. Ramos and Sheila S. Hemami, No date, “Edge-adaptive JPEG image compression”, School of Electrical Engineering, Cornell University, and Ithaca.

[12] Hong Shan Neoh and Asher Hazanchuk, No date, “Adaptive Edge Detection for Real-Time Video Processing using FPGAs”.