# Blocking Artifacts Reduction in JPEG Images

*Thesis submitted in partial fulfillment of the requirements of the award of degree of*

**Master of Technology**

*in*

**Computer Science and Applications**

*Submitted By*


**Puneet Malhotra**

(Roll No. 601003021)


*Under the supervision of*


**Mr. Singara Singh**

Assistant Professor



School of Mathematics and Computer Applications

Thapar University

Patiala – 147004


June 2012

# CERTIFICATE

I hereby certify that the work which is being presented in thesis entitled "**Blocking Artifacts Reduction in JPEG Images**", in the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Applications (*CSA*) submitted in School of Mathematics and Computer Applications (*SMCA*), Thapar University Patiala is an authentic record of my own work carried out under the supervision of Mr. Singara Singh and refers other researcher's work which are dually listed in reference section.

The material presented in this thesis has not been submitted for the award of any other degree of this or any other university.

**(Puneet Malhotra)**
Roll No. 601003021
M.Tech (CSA)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

02.07.2012
**(Mr. Singara Singh)**
Assistant Professor
*SMCA*

**Countersigned by**

**(Dr. S.S. Bhatia)**
Head
*SMCA*
Thapar University
Patiala

**(Dr. S. K. Mohapatra)**
Dean (Academic Affairs)
Thapar University
Patiala

# ACKNOWLEDGEMENT

I wish to express my sincere thanks and deep sense of gratitude to my teacher and guide Mr. Singara Singh, Assistant Professor, School of Mathematics and Computer Applications (*SMCA*), Thapar University, Patiala, Punjab, for his constant inspiration, scholarly guidance and helpful suggestion throughout the course of my thesis work.

I am very thankful to Ms. Maninder Kaur and all other faculty members of *SMCA* for their intellectual support throughout the course. I am also thankful to all staff members of *SMCA* for their kind cooperation and sincere help. My special thanks are due to family members and friends who constantly encouraged me to complete my thesis work.

**Puneet Malhotra**
Roll No. 601003021
M.Tech (CSA)

# LIST OF ABBREVIATIONS

| | |
|---|---|
| HDTV | High Definition Television |
| ISDN | Integrated Services Digital Network |
| JPEG | Joint Photographic Expert Group |
| MPEG | Motion Picture Expert Group |
| SNR | Signal to Noise Ratio |
| ISO | International Standard Organization |
| ITU | International TV System Committee |
| ANSI | American National Standard Institute |
| DPCM | Differential Pulse Code Modulation |
| PSNR | Peak Signal Noise Ratio |
| DCT | Discrete Cosine Transform |
| IDCT | Inverse Discrete Cosine Transform |
| MSE | Means Square Error |
| GUI | Graphic User Interface |
| DFT | Discrete Fourier Transform |
| IEC | International Electro technical Commission |
| CCITT | Consultative Committee for International Telephone And Telegraph |
| POCS | Projection onto Convex Set |
| VQ | Vector Quantization |
| DFOVS | Deblocking Frames Of Variable Sizes |
| RLE | Run Length Encoding |
| EOB | End Of Block |
| APDCT | All Phase Discrete Cosine Transform |
| CR | Compression Ratio |
| DAC | Divide And Conquer |
| BER | Bit Rate |

| | |
|---|---|
| bpp | Bits per pixel |
| 1-D | One Dimension |
| 2-D | Two Dimension |
| FDCT | Forward Discrete Cosine Transform |
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| ICM | Iterative Conditional Mode |
| MRF | Markov Random Field |
| HVS | Human Vision System |
| HMW | Huber-Markov Random Field Model |
| WABG | Wight Adaption By Grading |
| MSDS | Mean Squared Difference of Slope |
| QP | Quadratic Programming |
| DMSD | Difference in Mean Square Difference |
| Et al. | And others |
| BDCT | Block-based Discrete Cosine Transform |

# LIST OF TABLES

# LIST OF FIGURES

# TABLE OF CONTENTS

# ABSTRACT

With the advent of new technologies, it becomes indispensable to keep the existing technology updated and upgraded with new technologies because to replace the existing technology with new technology takes a lot of time and up gradation in the existing system. In the era of multimedia many researchers in different industries devote their time to improve the quality of the multimedia technologies and present the improved and highly sophisticated methods. Today lot of communication on internet and satellites take place in the form of images and videos. Daily millions of images of our earth, the moon, stars, planets and our ecological system are sent at every moment to the metrological centers and space research centers so that they can study and analysis this cosmos. In order to keep these communications and transmission alacritous the images need to be compressed by a sophisticated method. But to improve something could disrupt the other thing, hence on decompression, the recovered images carry some quality disorders like blocking artifacts, ringing effects and blurring which causes loss of information or we can say loss of quality.

The main purpose of this thesis is to reduce the blocking artifacts in *JPEG* image up to a very extent keeping in mind that less computation efforts would be made. It has been observed that post processing of artifacts removal gives better image quality. The post processing is performed either in spatial domain or frequency domain. The proposed method is hybrid model of both spatial domain and frequency domain. In the proposed method, the post-processing is performed in two phases. In phase one, pre-filtering is implemented in spatial domain while in phase two, the selective modifications on *DCT*-coefficients are performed in frequency domain. The proposed method is tested on different gray-scaled standard images at different bit rates and qualitative measures of these images are analyzed. The collected numerical results are then plotted on graphs to show the effectiveness of the proposed approach.

# CHAPTER 1
# INTRODUCTION

## 1.1 Data Compression

Data compression is the process of making a file of any kind of data smaller by minimizing the amount of data present so that this minimized form of compressed data can be sent over digital communication or can be stored in digital memory. Data Compression is process of representing information in a compact form. We create these compact representations by identifying and removing redundancies that exist in the data. Data can be character in a text file, numbers that are samples of speech or image waveforms or sequences of numbers that are generated by other processes. The Data compression is hugely used in digital communication like Mobile Communication, image, audios and videos on website, digital TV etc. whenever we make a long-distance call and we are using compression, we use modem or fax machine and we take benefit of compression. If we use a satellite TV service, we are using compression. With the continuing growth of modem communications technology, demand for image transmission and storage is increasing rapidly. Advances in computer technology for mass storage and digital processing have paved the way for implementing advanced data compression techniques to improve the efficiency of transmission and storage of images.
Our main concern will be on image compression, data compression algorithms are used in *JPEG*, *MPEG* standards to reduce the number of bits required to represent an image or a video sequence.

### Why we need Data Compression?

The reason we need data compression is that more and more of the information can be exchanged or transmitted and received in a limited resources like bandwidth and hardware required in transmitting the data.

## 1.2 Compression Types

There are two types of compressions:

1. Lossless Compression
2. Lossy Compression

### 1.2.1 Lossless Compression

Lossless compression techniques, as their name implies, involves no loss of data. If data have been losslessly compressed, the original data can be recovered exactly from the compressed data. Lossless compression is generally used for applications that cannot tolerate any difference between the original and compressed data. If data of any kind are to be processed or enhanced later to yield more information, it is important that the integrity be preserved.

### 1.2.2 Lossy Compression

Lossy compression techniques involve some loss of information, and data that have been compressed using lossy techniques generally cannot be recovered or reconstructed exactly. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratio than is possible with lossless compression.

In many lossy compression applications we are required to represent each source output using one of small number of codeword. The number of possible distinct source output values is generally much larger than the number of code words available to represent them. The process of representing a large set of values with a much smaller set is called quantization**.** This is one of the simplest and most general ideas in lossy compression. Quantization is a very simple process. However the design of the quantizer has a significant impact on the amount of compression obtained and loss incurred in a lossy compression scheme.

The process of converting, or digitizing, the almost infinitely variable amplitude of an analog waveform to one of a finite series of discrete levels is called Quantization.

In the context of image processing, reducing the number of bits in an image is called quantization. Quantization can be understood by following example-

Say we have an image of size 100×100 with 8-bits per pixel, which means there would be $2^8$=256 color values or we can say that there would be single value of color for every pixel. So this image needs 100×100×8=8000 bits or 1000 bytes for the storage on the disk. Now we bring the concept of quantization which says that we can take the sample of values which will fall in a fixed interval. We use three bits per pixel to store the image using quantization. For this we will divide the range of the numbers from 0 to 255 into $2^3$=8 intervals. Which are- 0-31, 32-63, 64-95, 96-127, 128-159, 160-191, 192-223, 234-255. And every interval can be assigned a binary code, which are-

$$2^3 = \{000,001,010,011,100,101,111\}$$

So any value which comes between 0-31 will be encoded as 000 and between 32-63 will be encoded as 001 and so on. Thus we need only 100×100×3=3000 bits or 3750 bytes to store an image on the disk. Later this quantized image is reconstructed by the process of dequantization.

There are two types of quantization.

i. **Scalar Quantization** encodes each term of the source sequence separately. Converting a real number to binary strings requires a mapping from *R* to discrete alphabet, called Scalar Quantization. In Scalar Quantization inputs and outputs are scalars.

ii. **Vector Quantization** segments source sequence into *n*-blocks which are quantized together. Converting a real *n*-tuple or vector of dimension *n* to binary string requires mapping $R^n$ to a discrete alphabet called Vector Quantization. In Vector quantization inputs and outputs are vectors.

### 1.3 Measures of Performance

A compression algorithm can be evaluated in a number of different ways. We could measure the relative complexity of the algorithm, the memory required to implement the algorithm, how fast the algorithm performs on a given machine, the amount of compression, the quality of the reconstructed image or data etc.

#### 1.3.1 Distortion

In lossy compression, the reconstruction differs from the original data. Therefore, in order to determine the efficiency of c compression algorithm, we need to have a parameter to quantify the difference, so the difference between the original and the reconstruction is often called the distortion.

For two given monochrome images of size $M \times N$, $I_1$ be the original image and $I_2$ be the decompressed image the objective measure can be defined as follows.

#### 1.3.2 Mean Square Error

The current image processing systems are designed to minimize the Mean Square Error *(MSE)* between the original image and reconstructed image, where *MSE* is defined as

$$\boldsymbol{MSE = \sigma_e^2 = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I_1(i,j) - I_2(i,j))^2} \quad \ldots (1.1)$$

Now for the performance and quality purpose we find out the Peak Signal to Noise Ratio (*PSNR*), using above (1.1), PSNR is used to evaluate the quality of the image. The *PSNR* Value of decompressed image against its original image can be computed as follows.

### 1.3.3 Peak-Signal-to-Noise-Ratio

The PSNR can be defined as follow

Based on *MSE,* the *PSNR* in decibel (*dB)* has the following expression.

$$\boldsymbol{PSNR = 10.log_{10}\left(\frac{I_{max}^2}{\sigma_e^2}\right) = 20.log_{10}\left(\frac{I_{max}}{\sigma_e}\right)} \quad \ldots (1.2)$$

Where $I_{max}$ is the maximum value of image which typically assumes the value of 255. The *PSNR* and *MSE* are the most commonly used measures of reconstruction quality in image compression.

### 1.3.4 Compression Ratio

A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression to the number of bits required to represent the data after compression, this ratio is called compression ratio. The Compression Ratio *(CR)* is another frequently used measure of how effectively an image has been compressed. For example, if the original image needs $s_1$ bits to be represented and the image after compression need $s_2$ bits to be represented then the compression ratio is given by

$$\text{compression ratio} = \frac{s_1}{s_2} \quad \ldots (1.3)$$

Assume that we have an uncompressed image of size 512×512 and each pixel is given 8-bits then total number of bits required to represent this image are

$$512 \times 512 \times 8 = 2097152\ bits$$

So

$$s_1 = 2097152$$

And after compression if we get 107633 bits to represent the compressed image then

$$s_2 = 107633\ bits$$

Therefore from (1.3) compression ratio becomes

$$\text{compression ratio} = \frac{2097152}{107633} = 19.4843 \quad \ldots (1.4)$$

### 1.3.5 Bits per pixel

This is another way of reporting compression performance is to provide the average number of bits required to represent a single sample. This is also referred to as bit rate. The compression ratio is closely related to the concept of coding rate or bit rate. The coding rate of still image compression method may be expressed in terms of bits-per-pixel, thus a lower coding rate indicates a higher level of compression. Since an original image is typically represented by eight bits, the number of bits-per-pixel is computed as

$$bpp = \frac{\mathbf{8}}{\boldsymbol{compression\ ratio}} \quad \ldots (1.5)$$

Therefore from (1.4) and (1.5) bpp becomes

$$bpp = \frac{8}{19.4843} = 0.4106$$

# CHAPTER 2

# JPEG BASELINE SYSTEM

## 2.1 Overview

Digital images contain large amount of information that need evolving effective techniques for storing and transmitting the ever increasing volumes of data. Image compression addresses the problem by reducing the amount of data required to represent a digital image. Discrete Cosine Transformation (*DCT*), quantization and entropy encoding are the steps in the compression of the *JPEG* image format. In this proposed work, the performance analysis is carried out. Code is written for the Joint Photographic Expert Group (*JPEG*), the factors *CR*, *SNR*, *PSNR* and *MSE* are studied considering different compression ratio for different images. It is found that performance will not remain same for different images even though compression factor is same.

The joint in *JPEG* refers to collaboration between *CCITT* and *ISO*. *JPEG* convenes officially as the *ISO* committee designated *JTCl/SC2/WGlO*, but operates in close informal collaboration with *CCIlT SGVIII*. *JPEG* will be both an *ISO* Standard and a *CCITT* Recommendation. Text of both will be identical.

Image compression is the technology of image data rate reduction to save storage space and reduce transmission rate requirements. Image compression offers the solution for diverse imaging applications requiring a vast amount of data to represent digital images [17].

The rapid growth of digital imaging applications, including desktop publishing, multimedia, teleconferencing, and High-Definition Television (*HDTV*) has increased the need for effective and standardized image compression techniques. Image compression is a technique used to reduce the storage and transmission costs. The existing techniques used for compressing image files are broadly classified into two categories, namely lossless and lossy compression techniques. In lossy compression techniques, the original digital image is usually transformed through an invertible linear transform into another domain, where it is highly de-correlated by the transform. This de-correlation concentrates the important image information into a more compact form. The transformed coefficients are then quantized yielding bit-streams containing long stretches of zeros. Such bi-streams can be coded efficiently to remove the redundancy and store it into a compressed file. The decompression reverses this process to produce the recovered image. The two-dimensional (*2-D*) *DCT* is an invertible linear transform and is widely

used in many practical image compression systems, because of its compression performance and computational efficiency.

*JPEG* typically achieves 10:1 compression with little perceptible loss in image quality. *JPEG* compression is used in a number of image file formats. It is the most common image format used by digital cameras and other photographic image capture device. In computing, *JPEG* is commonly used method of lossy compression for photographic images. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality [16].

The *JPEG* defines three different coding systems:

1. A lossy baseline coding system, which is based on the *DCT* and is adequate for most compression applications.
2. An extended coding system, for greater compression, higher precision, or progressive reconstruction applications;
3. A lossless independent coding system for reversible compression.

To be *JPEG* compatible, a product or system must include support for the baseline system. In the baseline system, often called the sequential baseline system, the input and output data precision is limited to 8-bits, whereas the quantized *DCT* values are restricted 11-bits. *DCT*-based encoding algorithms are always lossy by nature. *DCT* algorithms are capable of achieving a high degree of compression with only minimal loss of data. The compression itself is performed in three sequential steps: *DCT* computation, quantization, and variable-length code assignment.

The image is first subdivided into pixel blocks of size 8×8, which are processed left to right, top to bottom. As each 8×8 block or sub-image is encountered, its 64 pixels are level shifted by subtracting the factor $2^{n-1}$, where $2^n$ the maximum number of gray levels and n is is the number of bits assigned to each pixel. The *2-D DCT* of the block is then computed and further quantized and reordered, using the zigzag pattern to form a one-dimensional(*1-D*) sequence of quantized coefficients [2] then further encode the resulting coefficients(image data) using a Huffman variable word-length coding to remove redundancies in the image data.

The *DCT* works by separating images into parts of differing frequencies. During quantization, where parts of compression actually occur, the less important frequencies are discarded, hence the use of the term lossy. The remaining most important frequencies are used to retrieve the image in the decompression process. As a result, reconstructed image is an image with distortion [16].

Since the *1-D* reordered array generated under the zigzag pattern is qualitatively arranged according to increasing spatial frequency, the *JPEG* coding procedure is designed to take advantage of the long runs of zeros that normally result from the reordering [2].

The *JPEG* method is used for both color and gray images. A gray image is made up of pixels each of which holds a single number corresponding to the gray levels of the image at a particular location. The gray levels span the full range from black to white in a series of very fine steps, normally 256 different gray shades. Assuming 256 gray levels, each black and white pixel can be stored in a single byte (8-bits) of memory [16].

### 2.2 Processing Steps for *DCT*-Based Coding

Fig. 2.1 shows the key processing steps which are the heart of the *DCT*-based modes of operation. These figures illustrate the special case of single-component (grayscale) image compression. The reader can grasp the essentials of *DCT*-based compression by thinking of it as essentially compression of a stream of 8×8 blocks of grayscale image samples. For *DCT* sequential-mode codecs, which include the Baseline sequential codec, the simplified figure indicate how single-component compression works in a fairly complete way. Each 8×8 block is input, makes its way through each processing step, and yields output in compressed form into the data stream. For *DCT* progressive-mode codecs, an image buffer exists prior to the entropy coding step, so that an image can be stored and then parceled out in multiple scans with successively improving quality [17].

#### 2.2.1Forward Discrete Cosine Transform (*FDCT*) and Inverse Discrete Cosine Transform (*IDCT*)

At the input to the encoder , source image samples are grouped into 8x8 blocks, shifted from unsigned integers with range [0,$2^p$-1] to signed integers with range [-$2^{p-1}$,$2^{p-1}-1$] where *p* is the number of bits assigned to each pixel, and input to *FDCT*. At the output from the decoder, the *IDCT* outputs 8×8 sample blocks to form the reconstructed image. The following equations are the idealized mathematical definition of the 8×8 *FDCT* and 8×8 *IDCT*.

$$\boldsymbol{F(u,v) = \frac{1}{4} C(u) C(v) [\sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) * cos \frac{(2x+1)u\pi}{16} cos \frac{(2y+1)v\pi}{16}]} \quad \text{... (2.1)}$$

$$\boldsymbol{f(x,y) = \frac{1}{4} [\sum_{u=0}^{7} \sum_{v=0}^{7} C(u) C(v) F(u,v) * cos \frac{(2x+1)u\pi}{16} cos \frac{(2y+1)v\pi}{16}]} \quad \text{... (2.2)}$$

where: $\boldsymbol{C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} \; for \; u,v = 0 \; ; \\ 1 \; otherwise. \end{cases}}$

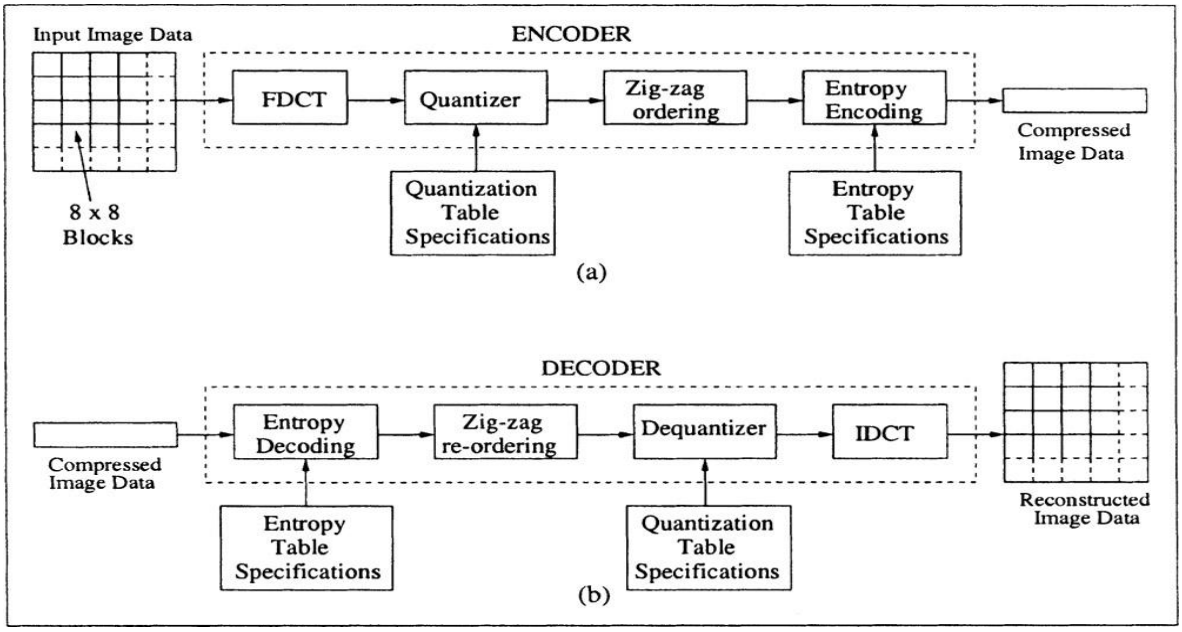


**Fig.2.1: *JPEG* compression and decompression steps, (a) Compression, (b) Decompression.**

The *DCT* is related to the Discrete Fourier Transform (*DFT*). Each 8×8 block of source image samples is effectively a 64-point discrete signal which is a function of the two spatial dimensions *x* and *y*. The *DCT* takes such a signal as its input and decomposes it into 64 orthogonal basis signals. Each contains one of the 64 unique *2-D* spatial frequencies which comprise the input signal's spectrum. The output of the *DCT* is the set of 64 basis-signal amplitudes or *DCT* coefficients whose values are uniquely determined by the particular 64-point input signal.

The *DCT* coefficient values can thus be regarded as the relative amount of the *2-D* spatial frequencies contained in the 64-point input signal. The coefficient with zero frequency in both dimensions is called the *DC* coefficient and the remaining 63 coefficients are called the *AC* coefficients. Because sample values typically vary slowly from point to point across an image, the *FDCT* processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. For a typical 8×8 sample block from a typical source image, most of the spatial frequencies have zero or near-zero amplitude and need not be encoded.

At the decoder the *IDCT* reverses the processing steps. It takes the 64 *DCT* coefficients (which at the point have been quantized) and reconstructs a 64-point output image signal by summing the basis signals. Mathematically, the *DCT* is one to one mapping for 64-

point vectors between the image and the frequency domains. If the *FDCT* and *IDCT* could be computed with perfect accuracy and if the *DCT* coefficient were not quantized, the original 64-point signal could be exactly recovered. In principle, the *DCT* introduces no loss to the source image samples; it merely transforms them to a domain in which they can be more efficiently encoded [17].

### 2.2.2 Quantization

After outputs from the *FDCT*, each of the 64 *DCT* coefficients is uniformly quantized in conjunction with a 64-element quantization Table as in Fig. 2.2 which must be specified by the application as an input to the encoder. Each element can be any integer value from 1 to 255, which specifies the step size of the quantizer for its corresponding *DCT* coefficient. The purpose of quantization is to achieve further compression by representing *DCT* coefficients with no greater precision than is necessary to achieve the desired image quality. When the aim is to compress the image as much as possible without visible artifacts, each step size ideally should be chosen as the perceptual threshold or "just noticeable difference" for the visual contribution of its corresponding cosine basis function.

In general, higher spatial frequencies are less visible to the human eye than low frequencies. Therefore, the quantization factors are usually chosen to be larger for the higher frequencies. The following quantization matrix is widely used for monochrome images and for the luminance component of a color image. It is given in the *JPEG* standards documents. Stated another way, the goal of this processing step is to discard information which is not visually significant. Quantization is many-to-one mapping, and therefore is fundamentally lossy. It is the principal source of lossiness in *DCT*-based encoders [17]. A typical normalization or quantization matrix is given by

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

**Fig. 2.2: A typical normalization matrix.**

Quantization is defined as division of each *DCT* coefficient by its corresponding quantizer step size followed by rounding to the nearest integer.

$$\boldsymbol{F^{Q}(u,v) = Round\left(\frac{F(u,v)}{Q(u,v)}\right)} \quad \ldots (2.3)$$

This output value is normalized by the quantizer step size. Dequantization is the inverse function, which in this case means simply that the normalization is removed by multiplying by the step size, which returns the result to a representation appropriate for input to the *IDCT*.

$$\boldsymbol{F^{Q'}(u,v) = F^{Q}(u,v) * Q(u,v)} \quad \ldots (2.4)$$

### 2.2.3 *DC* Coding and Zigzag Sequence

After quantization the *DC* coefficient is treated separately from the 63 *AC* coefficients. The *DC* coefficient is a measure of the average value of the 64 image samples. Because there is usually strong correlation between the *DC* coefficients of adjacent 8×8 blocks, the quantized *DC* coefficient is encoded as the difference from the *DC* term of the previous block in the encoding order. As shown in Fig. 2.3. The special treatment is worthwhile, as *DC* coefficients frequently contain a significant fraction of the total image energy. Finally, all of the quantized coefficients are ordered into zigzag sequence, also shown in Fig. 2.4, where $AC_{01}$ can be regarded as the *DCT* coefficient of zeroth row and first column of the image block similarly $AC_{77}$ can be regarded as the *DCT* coefficient of 7$^{th}$ row and 7$^{th}$ column of the block and so on. This ordering helps to facilitate entropy coding by placing low-frequency coefficients (which are more likely to be nonzero) before high-frequency coefficients [17].

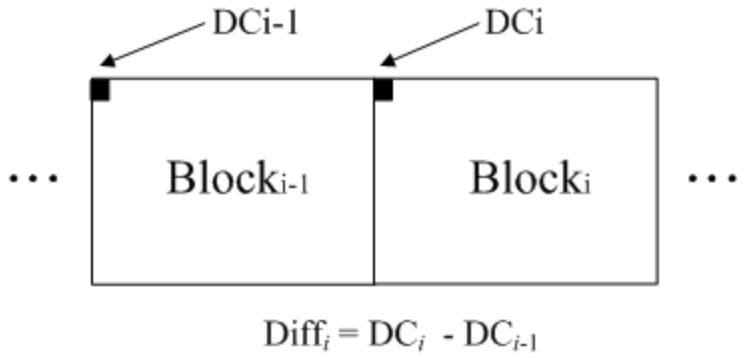


**Fig. 2.3: Differential *DC* encoding.**

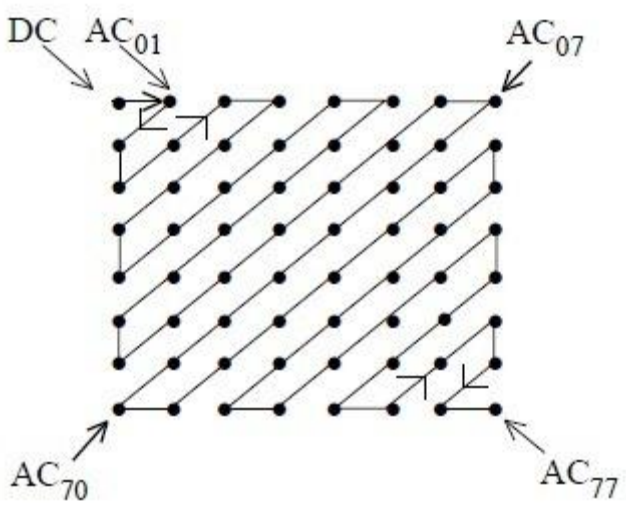


**Fig. 2.4: Zigzag sequence.**

### 2.2.4 Entropy Coding

The final *DCT*-based encoder processing step is entropy coding. Entropy coding is a special form of lossless data compression. This step achieves additional compression losslessly by encoding the quantized *DCT* coefficients more compactly based on their statistical characteristics. The *JPEG* proposal specifies two entropy encoding methods – Huffman Coding [18] and arithmetic coding. The baseline sequential codes use Huffman coding, but codecs with both methods are specified for all modes of operation.

It involves arranging the image components in a zigzag order employing Run Length Encoding (*RLE*) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left. The *JPEG* standard also allows, but does not require, the use of arithmetic coding, which is mathematically superior to Huffman coding. However, this feature is rarely used as it is covered by patents and because it is much slower to encode and decode compared to Huffman coding. Arithmetic coding typically makes files about 5% smaller.

In Entropy coding while encoding an 8×8 block the *DC* coefficient is encoded according to the range in which it falls (see Table-2.2), and then its corresponding *DC* category is checked. And then default *DC* table is referred (see Table-2.1) to know the corresponding base code and length. Length tells us about the total length of the final encoded code. Length is always the summation of base code length and category which the *DC* belongs to. If the base code length is less than the total length then the remaining bits must be generated from the Least Significant Bits (*LSBs*) of the difference value. Difference value means the difference between the *DC* value of current 8×8 block and that of previous 8×8

block. For a general *DC* difference category (say, category *K*), an additional *K* bits are needed and computed as either *K LSBs* of the positive difference or the *K LSBs* of the negative difference minus 1.

The nonzero *AC* coefficients of the reordered array are coded similarly from the Table-2.1 and Table 8.19 in [2] the principal difference is that each default *AC* Huffman code word depends on the number of zero-valued coefficients preceding the nonzero coefficient to be coded, as well as the magnitude category of the nonzero coefficient.(see the Table 8.19 in [2]), here run means the number of zero valued coefficients preceding the current nonzero *AC* coefficient, category denotes the category the *AC* coefficient belongs to and length is the total length of the final coded word.

Here length is always summation of length of base code and category. Whenever the length of base code is less than the total length then the remaining bits are generated by the same process used to arrive at the *LSBs* of the *DC* difference code. There is a code 1010 which tells us the ending of the 8×8 block, that's why it is called End Of Block (*EOB)*. And there is a code under the run=*F* and category=0, it tells us that there have been 15 zero-valued *AC* coefficients preceded by the non-zero *AC* coefficient.

**Table 2.1: *JPEG* default *DC* code (luminance).**

| Category | Base Code | Length |
|---|---|---|
| 0 | 00 | 2 |
| 1 | 010 | 4 |
| 2 | 011 | 5 |
| 3 | 100 | 6 |
| 4 | 101 | 7 |
| 5 | 110 | 8 |
| 6 | 1110 | 10 |
| 7 | 11110 | 12 |
| 8 | 111110 | 14 |
| 9 | 1111110 | 16 |
| A | 11111110 | 18 |
| B | 111111110 | 20 |

**Table 2.2: *JPEG* coefficient coding categories.**

| Range | DC Difference Category | AC Category |
|---|---|---|
| 0 | 0 | N/A |
| -1,1 | 1 | 1 |
| -3,-2,2,3 | 2 | 2 |
| -7,…,-4,4,…7 | 3 | 3 |
| -15,…,-8,8,…,15 | 4 | 4 |
| -31,…,-16,16,…31 | 5 | 5 |
| -63,…,-32,32,…63 | 6 | 6 |
| -127,…,-64,64,…,127 | 7 | 7 |
| -255,…,-128,128,…255 | 8 | 8 |
| -511,…,-256,256,…,511 | 9 | 9 |
| -1023,…,-512,512,…,1023 | A | A |
| -2047,…,-1024,1024,…,2047 | B | B |
| -4095,…,-2048,2048,…,4095 | C | C |
| -8191,…,-4096,4096,…,8191 | D | D |
| -16383,…,-8192,8192,…,16383 | E | E |
| -32767,…,-16384,16384,…,32767 | F | N/A |

Consider compression and reconstruction of the following 8×8 sub-image with the *JPEG* baseline standard.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 98 | 113 | 114 | 117 | 125 | 116 | 127 | 88 |
| 98 | 100 | 100 | 74 | 73 | 80 | 80 | 54 |
| 72 | 60 | 59 | 56 | 57 | 50 | 49 | 53 |
| 57 | 50 | 52 | 58 | 47 | 51 | 47 | 47 |
| 52 | 48 | 47 | 47 | 48 | 44 | 47 | 50 |
| 51 | 61 | 67 | 47 | 60 | 39 | 40 | 87 |
| 56 | 92 | 90 | 57 | 93 | 47 | 42 | 112 |
| 69 | 93 | 121 | 67 | 84 | 84 | 91 | 72 |

**Fig. 2.5: 8×8 original sub-image.**

The original image consists of 256 or $2^8$ possible gray scale levels, so the coding process begins by level shifting the pixels of the original subimage by $-2^7$ or -128 gray levels. The resulting shifted array is

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -30 | -15 | -14 | -11 | -3 | -12 | -1 | -40 |
| -30 | -28 | -28 | -54 | -55 | -48 | -48 | -74 |
| -56 | -68 | -69 | -72 | -71 | -78 | -79 | -75 |
| -71 | -78 | -76 | -70 | -81 | -77 | -81 | -81 |
| -76 | -80 | -81 | -81 | -80 | -84 | -81 | -78 |
| -77 | -67 | -61 | -81 | -68 | -89 | -88 | -41 |
| -72 | -36 | -38 | -71 | -35 | -81 | -86 | -16 |
| -59 | -35 | -7 | -61 | -44 | -44 | -37 | -56 |

**Fig. 2.6: 8 × 8 128 level shifted sub-image.**

Now this 128 level shifted image is when transformed in accordance with the *FDCT* of (2.1) for *N=8*, becomes

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -458 | -24 | 0 | -23 | -9 | -22 | 4 | 22 |
| 49 | 15 | -12 | 39 | -8 | 36 | -14 | -14 |
| 152 | 3 | -18 | -4 | -31 | -7 | -5 | 18 |
| 26 | -18 | 0 | -9 | 15 | -14 | -7 | -1 |
| 27 | -13 | -21 | 18 | -20 | 27 | -8 | -2 |
| 13 | -15 | -5 | -13 | 23 | -19 | -4 | -3 |
| -2 | -12 | -3 | 3 | 1 | 10 | 0 | 1 |
| -3 | -5 | 0 | 2 | 10 | -8 | -5 | -2 |

**Fig.2.7: 8×8 *FDCT* transformed sub-image.**

If the *JPEG* recommended normalization array of Fig. 2.2 is used to quantize the transformed array, the scaled and truncated [that is, normalized in accordance with (2.3)] coefficients are in Fig. 2.8

where, for instance, the *DC* coefficient is computed as

$$F^q(0,0) = Round[\frac{F(0,0)}{Q(0,0)}]$$

$$= Round\left[-\frac{458}{16}\right] = -29$$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -29 | 2 | 0 | -1 | 0 | -1 | 0 | 0 |
| 4 | 1 | -1 | 2 | 0 | 1 | 0 | 0 |
| 11 | 0 | -1 | 0 | -1 | 0 | 0 | 0 |
| 2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | -1 | -1 | 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 |

**Fig. 2.8: 8×8 Quantized sub-image.**

Here it should be noticed that the transformation and normalization process produces a large number of zero-valued coefficients. When the coefficients are reordered in accordance with the zigzag ordering pattern of Fig. 2.4, the resulting *1-D* coefficients sequence is

[-29 2 4 11 1 0 -1 -1 0 2 1 -1 -1 2 0 -1 0 0 0 -1 1 0 0 -1 0 -1 1 *EOB* ]

Where the *EOB* symbol denotes the end-of-block condition. A special *EOB* Huffman code word is provided to indicate that the remainders of the default *JPEG* code for the reordered sequences are zeros [2].

The construction of the default *JPEG* code for the reordered coefficient sequence begins with the computation of the difference between the current *DC* coefficient and that of the previously encoded sub-image [2]. The sub-image that we are taking throughout this section for this let us assume that the *DC* coefficient of the transformed and quantized sub-image to its immediate left was -17, the resulting *DPCM* difference is [-29-(-17)] or -12,which lies in *DC* difference category 4 of Table-2.2. In accordance with the default Huffman difference code of the Table-2.1, the proper base code for a category is 101 (a 3-bit code), while the total length of a completely encoded category 4 coefficient is 7 bits. The remaining 4 bits must be generated from the *LSBs* of the difference value. For a general *DC* difference category (say, category *K*), additional *K* bits are needed and computed as either the *K LSBs* of the positive difference or the *K LSBs* of the negative difference minus 1. For a difference of -12, the appropriate *LSBs* are 0011, and the complete *DPCM* coded *DC* code word is 1010011.

The nonzero *AC* coefficients of the reordered array are coded similarly from Table-2.2 and Table-8.19 in [2]. The principal difference is that each default *AC* Huffman code word depends on the zero-valued coefficients preceding the nonzero coefficient to be coded, as well as the magnitude category of the nonzero coefficient. Thus the first

nonzero *AC* coefficient of the reordered array is coded as 0110. The first 2 bits of this code indicate that the coefficient was in magnitude category 2 and proceeded by no zero-valued coefficients; the last 2 bits are generated by the same process used to arrive at the *LSBs* of the *DC* difference code. Continuing in this manner, the completely coded (reordered) array is

1010011(*DC*) 0110 100100 10111011 001 11000 000 11100110 001 000 000 0110 11000 1110100 001 110110 11000 001(*ACs*) 1010(*EOB*).

Where the spaces have been inserted solely for readability. The total number of bits in the completely coded reordered array is 90. The resulting compression ratio is 512/90, or about 5.68:1.

To decompress a *JPEG* compressed sub-image, the decoder must first recreate the normalized transform coefficients that led to the compressed bit stream. In order to recreate the normalized transform coefficients our tool will read every bit of the stream one by one and send to the function and these function will check with table values if any bit sequence matches with the table values and if match is found then our tool's function will compute the other parameters for that coefficient like Run (for *AC*)/Category, length.

Here the regenerated array of quantized coefficients is

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -29 | 2 | 0 | -1 | 0 | -1 | 0 | 0 |
| 4 | 1 | -1 | 2 | 0 | 1 | 0 | 0 |
| 11 | 0 | -1 | 0 | -1 | 0 | 0 | 0 |
| 2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | -1 | -1 | 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 |

**Fig. 2.9: Regenerated array of quantized coefficients.**

After de-normalization in accordance with (4), the array becomes, shown in Fig. 2.10.

Where, for example, the *DC* coefficient is computed according to (4)

i.e.

$$\boldsymbol{F^{Q'}(u,v) = F^{Q}(u,v)Q(u,v) = (-29) \times (16) = -464}$$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -464 | 22 | 0 | -16 | 0 | -40 | 0 | 0 |
| 48 | 12 | -14 | 38 | 0 | 58 | 0 | 0 |
| 154 | 0 | -16 | 0 | -40 | 0 | 0 | 0 |
| 28 | -17 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | -22 | -37 | 0 | 0 | 0 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Fig. 2.10: De-normalized matrix.**

Now the completely reconstructed sub-image is obtained by taking the *IDCT* of the de-normalized array in accordance with (2) to obtain

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -26 | -23 | -9 | -7 | -12 | 1 | -6 | -44 |
| -31 | -41 | -44 | -49 | -56 | -48 | -48 | -67 |
| -46 | -60 | -70 | -71 | -75 | -80 | -80 | -78 |
| -75 | -80 | -82 | -71 | -66 | -85 | -93 | -75 |
| -91 | -83 | -88 | -81 | -69 | -90 | -99 | -67 |
| -73 | -52 | -72 | -86 | -70 | -84 | -88 | -46 |
| -62 | -21 | -43 | -71 | -53 | -61 | -71 | -32 |
| -80 | -18 | -28 | -56 | -35 | -44 | -65 | -37 |

**Fig. 2.11: *IDCT* matrix of de-normalized array.**

Now level shifting each inverse transformed pixel by +128 to yield.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 102 | 105 | 119 | 121 | 116 | 129 | 122 | 84 |
| 97 | 87 | 84 | 79 | 72 | 80 | 80 | 61 |
| 82 | 68 | 58 | 57 | 53 | 48 | 48 | 50 |
| 53 | 48 | 46 | 57 | 62 | 43 | 35 | 53 |
| 37 | 45 | 40 | 47 | 59 | 38 | 29 | 61 |
| 55 | 76 | 56 | 42 | 58 | 44 | 40 | 82 |
| 66 | 107 | 85 | 57 | 75 | 67 | 57 | 96 |
| 48 | 110 | 100 | 72 | 93 | 84 | 63 | 91 |

**Fig. 2.12: Decompressed image pixel values**.

In our experiment results, we will measure the quality of images in terms of *PSNR* and *bpp*. Here we will see various images at various *bpp* with their corresponding *PSNR* values. Here each image is of size 512×512.

**Table 2.3: *PSNR* of *JPEG* encoded images.**

| **Image** | **Quantization Factor (*q*)** | **bit rate(*bpp*)** | ***PSNR* (*dB*)** |
|---|---|---|---|
| Lena | 1 | 0.639034 | 35.78721 |
| | 2 | 0.410587 | 33.6988 |
| | 3 | 0.319443 | 32.3304 |
| | 4 | 0.269257 | 31.25481 |
| | 5 | 0.235943 | 30.40820 |
| | 6 | 0.213051 | 29.62699 |
| | 7 | 0.195747 | 28.98775 |
| | 8 | 0.182648 | 28.44898 |
| Cameraman | 1 | 0.571552 | 38.83284 |
| | 2 | 0.379787 | 35.66638 |
| | 3 | 0.300735 | 33.82659 |
| | 4 | 0.257023 | 32.39632 |
| | 5 | 0.226219 | 31.22943 |
| | 6 | 0.205257 | 30.32567 |
| | 7 | 0.188751 | 29.54375 |
| | 8 | 0.175797 | 28.88331 |
| Elaine | 1 | 0.671173 | 33.07912 |
| | 2 | 0.384293 | 31.99949 |
| | 3 | 0.28698 | 31.23214 |
| | 4 | 0.239525 | 30.55003 |
| | 5 | 0.208233 | 29.93641 |
| | 6 | 0.187626 | 29.37748 |
| | 7 | 0.171974 | 28.80774 |
| | 8 | 0.162182 | 28.32341 |

**Fig. 2.13: Lena original image.**

**Fig. 2.14: Lena at 0.639 bpp.**

**Fig. 2.15: Lena at 0.235 bpp.**

**Fig. 2.16: Lena at 0.195 bpp.**

**Fig. 2.17: Lena at 0.182 bpp.**

The image Lena of size 512×512 grayscale shown in Fig. 2.13 compressed at different bit rates is shown in Fig. 2.14, Fig. 2.15, Fig. 2.16 and Fig. 2.17.

**Summary**

In this chapter we have discussed the fundamental steps of *JPEG* compression and decompression, each step has precisely been elaborated and for further knowledge the references have been marked. As *JPEG* consists of *DCT*, Quantization, Zigzag sequence and then entropy encoding same steps are processed in reverse order in decompression. Further with the compression and decompression steps their quality measures like *PSNR* value and bit rate have been discussed.

# CHAPTER 3
# LITERATURE SURVEY

The work that has been done in reduction of blocking effects and artifacts in *DCT*-based images using various techniques is described further in this section focusing mainly on the optimized techniques to remove or minimize the blocking effect.

When a high compression ratio is required, the coded image suffers from blocking effects. The blocking artifacts in smooth regions are removed by modifying a few *DCT* coefficients appropriately. Several techniques have been suggested to minimize the blocking effects.

Two approaches are generally adopted. First approach is preprocessing and second approach is post processing. In the first approach, the reduction of blocking artifacts is performed before encoding, but the methods based on this approach do not conform to the existing standards such as *JPEG* and *MPEG.* While in the second approach, the reconstructed image is post processed for improving. Post processing of the decoded image may be carried out in spatial domain or in frequency domain [5].

## 3.1 Analysis in Spatial Domain

Reeve *et al.* [12] proposed a symmetric, *2-D* 3×3 Gaussian spatial filtering method for the pixels along the block boundaries. However, it causes blurring of the image due to its low-pass nature. Ramamurthi *et al.* [13] proposed nonlinear space-variant filter which adapts to the varying shape of the local signal spectrum, and reduces only the locally out-of-band noise. The algorithm implements a *2-D* filter in the areas away from edges, and for near edges, *1-D* filter aligned parallel to edge so as to minimize the blocking artifacts. Meier et al. [14] presented a region-based method for enhancement of images degraded by blocking effects. In this method, the degraded image is segmented by a region growing algorithm, and each region is the filtered using a low-pass filter. It preserves the edges, as filtering is not applied to region boundaries.

Lee *et al.* [15] proposed a post processing algorithm to reduce the blocking artifacts in *JPEG* compressed images after classifying them into edge area and monotone area according to the edge map which is obtained after thresholding the gradient absolute image. The signal adaptive filtering consists of a *1-D* directional smoothing filtering for edge area and *2-D* adaptive average filtering for monotone area. A corner outlier detection/replacement scheme is also given to remove the corner outlier.

Chou *et al.* [22] remove blockiness by performing a simple nonlinear smoothing of pixels. Authors first formed the maximum likelihood estimation of quantization noise to differentiate between artificial and actual edges.

The theory of Projection Onto Convex Sets (*POCS*) [3, 6,7,23, and 24] was used to derive some iterative algorithms. In these methods, initially closed convex constraint sets are defined which correspond to all of the available data on the original image. Iterative computations of alternating projections onto these convex sets recover the original image from the compressed image. However, these methods usually have high computational complexity, and thus are difficult to adapt to real-time image processing applications.

In [3] new directional smoothness constraint sets were defined based on line processes modeling of the image edge structure. The definition of these smoothness sets also took into account the fact that the visibility of compression artifacts in an image is spatially varying. To overcome the numerical difficulty in computing the projections onto these sets, a Divide-And-Conquer (*DAC*) strategy was introduced such that their projections were easier to compute. In [3] authors discussed about the ringing effect in addition to blocking effects in compressed images, Which is caused by the loss of high-frequency coefficients and appears as ringing patterns around sharp edges. The ringing effect is particularly prominent in images that are of high contrast and contain sharp edges and thin structures, such as document images. In this Authors defined smoothness constraint sets on ringing artifacts, in compressed image the location of ringing artifacts are not known priori, unlike the blocking artifacts which are known to take place at block boundaries, so the line process concept was used to describe the edge structure in an image. Using line process concept authors derived some directional smoothness constraints sets which were difficult to be included in a recovery algorithm. So authors applied a *DAC* strategy to these directions smoothing sets and derived new smoothness constraints, hence their projection was easier to be computed. The algorithm proposed in [3] was easy to implement and converged very rapidly, but demanded a lot of computation to define smoothing constraint sets.

Luo *et al*. [26] proposed a two step approach for blocking artifacts reduction based on Maximum A Posteriori (*MAP)*, designed by a Huber-Markov random field model (*HMW*). First, a *DC* calibration is performed in a block by block fashion based on gradient continuity constraints over the block boundaries. Then, a modified Huber-Markov random field model is employed in order to differentiate the pixels on the block

boundary from those inside the block. Finally a local optimization technique, Iterative Conditional Mode (*ICM*) is applied to employ smoothing algorithms.

Meier *et al.* [27] proposed a method in which firstly the image is segmented by a Markov Random Fields (*MRF*) segmentation algorithm and then each region enhanced separately using an *MRF* model. This not only preserves edges but also allows texture areas to be treated differently. Coudoux *et al*. [28] proposed a method based on a nonlinear, space variant filtering. A visibility parameter is computed for each artifact using several characteristics of the Human Vision System (*HVS*). Then this information is used to steer the selection of an adaptive nonlinear, space variant smoothing operation at block boundaries.

Averbuch *et al.* [29] proposed a new class of algorithm for de-blocking or removing blocking effects from *JPEG* compressed images and video sequences. Authors used the Wight Adaption By Grading *(WABG)* approach to eliminate the ghosting effect. The ghosting effect is characterized by blurred false edges which are adjacent to real edges. Applying *WABG* based algorithm still exhibits minor blocking artifacts in monotone areas. Authors classified the blocks in two categories and processed separately, *i.e.* uniform blocks and non-uniform blocks. The uniform blocks were processed in three iterations by the *WABG* scheme and the non-uniform blocks were processed by De-blocking Frames Of Variable Sizes *(DFOVS).*

Zon *et al*. [30] proposed a method based on *POCS* that offers a necessary and suitable way to adjust pixels intensity. In the proposed method firstly the adjustment of a gray scale value is determined by local properties of the pixel, secondly three locally adaptive constraints are introduced to improve de-blocking results. Thirdly, the *HVS* modeling and relaxed projections are incorporated to make pixels adjust appropriately.

Park *et al.* [31] proposed the algorithm which can be implemented in both pixel and *DCT* domain with higher accuracy. The proposed method measures the blocking artifacts by using the original pixel difference in block boundary without using the original image.

### 3.2 Analysis in Frequency Domain

Minami *et al*. [32] presented a new approach for reducing the blocking effect in frequency domain by using the correlation between the intensity values of boundary pixels of two neighboring blocks. A new index to measure the blocking effects namely the Mean Squared Difference of Slope (*MSDS*) was introduced. It was observed under mild assumption that the quantization of the *DCT* coefficients of two neighboring blocks

increases the expected value of the *MSDS* between the slope across two adjacent blocks, and the average between the boundary slopes of each of the two blocks. This technique removes the blocking effects by minimizing the *MSDS*, while imposing linear constraints corresponding to quantization bounds. To minimize the *MSDS*, authors formulated a Quadratic Programming (*QP*) problem and solved it using the gradient projection method. Lakhani *et al.* [33] also reduced blocking effects using *MSDS*. Here authors analyzed the problem of minimization of the *MSDS* and presented a very different formulation and solution. First authors obtained a closed-form representation of the optimization function and then showed that this function is convex. Next authors derived equations to compute the exact, global minimum. These equations were used to predict the four lowest *DCT* Coefficients.

Traintafyllidis et al. [34] proposed another enhanced method of minimizing *MSDS*, which involved all the neighboring blocks i.e. horizontal and vertical blocks including the diagonally located neighboring blocks. This technique was based on reducing the blocking artifacts in the smooth regions of the image. The correlation between the intensity values of the boundary pixels of two neighboring blocks in the *DCT* domain was used to distinguish between smooth and non smooth regions.

Zeng [8] proposed a simple *DCT* domain method for blocking artifact reduction by applying a zero masking to the DCT coefficients of some shifted image blocks. Author first recognized that visible boundaries between two adjacent blocks in the coded image are primarily oriented along the horizontal and vertical directions. Then author constituted a new data block that includes the original visible boundary in the middle position. By doing *DCT* on this newly formed block, author found that there always exist *AC* components of significant energy at four positions in the first row: (0, *j*) for *j*= 1,3,5,7 in both horizontal and vertical directions. So instead of dropping these entire five components author proposed to drop three components which were 3, 5 and 7 and set them zero, which resulted to less visible boundaries. The algorithm gave optimum result on repeating this process for a number of iteration which made this algorithm computationally heavy. However, the loss of edge information caused by the zero-masking scheme was observable.

Liu *et al*. [35] proposed a *DCT* domain method for blind measurement of blocking artifacts, by modeling the artifacts as *2-D* step functions in shifted blocks [20]. A fast *DCT* domain algorithm extracts all the parameters needed to detect the presence of, and estimate the amplitude of blocking artifacts, by making use of *HVS* properties. Using the

estimate of blockiness, an efficient *DCT*-domain method was developed, which adaptively reduces detected blocking artifacts. The proposed technique had low computational cost.

Luo *et al.* [9] adopted [8] concept of zero masking and analyzed this idea further and gave more computationally efficient process to bring more smoothness in low-bit and high-bit rate compressed images in less time while preserving the sharp edges and suppressing the blocking effects. Authors explored the idea of correlation between the intensity values of the boundary gray scale values of two neighboring blocks in the *DCT* domain. The continuity of the pixels across the boundary is recovered from the continuity of the pixels values of the two neighboring blocks. This idea was used to distinguish between smooth and non-smooth regions. Authors observed that modifying or zeroing the 3, 5 and 7 indexed *DCT*-Coefficients of first row of all the blocks could bring some artifacts in the image, so to minimize the computation complexity of algorithm, in order to modify the *DCT*-Coefficients, authors imposed some constraint before modifying the *DCT*-coefficients which affected the smoothing of the image.

Wang *et al*. [36] proposed a method for blind measuring the degree of blocking artifacts and post-processing technology for removing these discontinuities was proposed. Authors utilized Walsh Transform to form a *DC* Image for obtaining the edge distribution in the original image by Walsh transform and local threshold technology, the precision of edge detection by Sobel operator was improved, so blurring in objective edge was reduced. In the process of de-blocking, compensatory coefficients matrices and adaptive smooth method were introduced to improve post-processed image performance.

Mikhael *et al*. [37] proposed a novel multiple transform domain split Vector Quantization (*VQ*) technique for Image compression, by using the proposed technique, a lower data rates was achieved for the same *PSNR*. This was accomplished at the expense of increased computational complexity at the encoder. The proposed method can control the compression ratio at certain critical regions of the image so that the target recognition performance can be preserved.

Park et al. [38] proposed a blind but accurate measurement algorithm for blocking artifacts. The proposed algorithm can be implemented in both pixel and *DCT* domains without using the original image. The proposed method can be used to improve the performance of existing algorithms reducing the blocking artifacts.

Tai et al. [39] proposed a method which focused on how to efficiently detect the blocks which caused ringing artifacts, without a lot of computational complexity. These ringing

blocks would be further filtered using the range filter and the texture region would be able to preserve as well as the smooth region. The proposed algorithm applied the local available information to achieve much computational saving with the better visual quality.

Nguyen *et al.* [40] proposed an efficient method to reduce blocking, ringing, and color bleeding artifacts. In proposed method for deblocking a conditionally low pass filter was applied for local shifted blocks. To avoid blurry effect, a novel fuzzy filter was proposed for deringing. Filter coefficients of this filter were weighted locally to reduce the painting-like effect of fuzzy filters.

Chacko et al. [41] proposed a method for reducing the blocking artifacts and increasing the visually quality of the image, in order to do this, authors applied the Gaussian filter to the compressed image. In order to reproduce the high frequency coefficients a differential image is obtained. From the differential image the artifacts contents are removed and smoothed edges are retained. The edge detected image is added with the filtered image to improve the sharpness of the image. Since *PSNR* alone does not indicate the quality of the image. So a new measuring parameter called *DMSD* was also used to measure the quality of the image.

# CHAPTER 4

# PROBLEM DISCUSSION AND PROPOSED APPROACH

## 4.1 Introduction

Transform-based data compression is by far the most popular choice in both image and video coding applications and it has been a very popular technique for digital image compression. Due to its near-optimal energy compaction property and the availability of fast algorithms and hardware implementations, the block-based discrete cosine transform (*BDCT*) is the dominant one among various transforms. Therefore, the *BDCT* is used in most of current image and video compression standards, such as *JPEG* and *MPEG*. The basic approach is to divide the whole image into fixed size blocks (normally 8×8 or 16×16). Then the image data is de-correlated via an orthogonal transform. In *JPEG*, the *DCT* is used. After the image data has been transformed into frequency domain, the *DCT* coefficients are quantized and the *2-D* data is converted to *1-D* using the zigzag structure. The non-zero amplitudes and run-length of zeros is then coded and saved or further transmitted.

The *BDCT* based coding can successfully compress images by a factor of around ten, with nearly no perceptible artifacts. However, at low bit rates, a major problem associated with the *BDCT* compression is that the decoded images manifest visually objectionable artifacts. These are due to the quantization step of the *JPEG* algorithm. One of the well-known artifacts in low-bit-rate transform-coded images is the blocking effect, which is noticeable in the form of undesired visible block boundaries.

## 4.2 Problem statement

Here our sole purpose would be to reduce these blocking artifacts as far as possible retaining the image quality as maximum as possible. To serve our purpose we may have to compromise with the complexity of the algorithm in terms of memory and time. Here we will take two papers into consideration which are basis for this thesis.

In [8] *Zeng* provided a solution for the *DCT* transformed blocking artifacts, although before this solution several techniques were made available to minimize the blocking effects. The theory *POCS* has been used to derive some iterative algorithms [6, 7], but they require a large number of iterations to reach the convergence. Method using interleaved image block before the encoding were also suggested [10, 11], but they are not in conformity with the coding standards.

In this author provided a *DCT* transform domain solution to the problem of blocking effect reduction. Author first recognized that visible boundaries between two adjacent blocks in the coded image are primarily oriented along the horizontal and vertical directions. Then author constituted a new data block that includes the original visible boundary in the middle position and model it as a *2-D* step function contaminated by noise with zero mean and a small variance. By doing *DCT* on such blocks, they found that there always exist *AC* components of significant energy in a few fixed positions. Based on this observation, they proposed to zero out some of these *AC* components and demonstrated that doing so can make blocking artifacts much less visible.

Author observed all the blocks which suffer the blocking effect either horizontally or vertically and found that each of these blocks and its neighboring block together constitute approximately a horizontal or vertically *2-D* step function.

Say if there are two horizontally neighboring blocks, $[a]_{8\times8}$ and $[b]_{8\times8}$ and let's take the right half from $[a]_{8\times8}$ and the left half from $[b]_{8\times8}$ so as to constitute a new $8 \times 8$ image block $[x]_{8\times8}$ that shows a horizontal step behavior- original visible boundary remains in the middle place:

$$[x]_{8\times8} = \left[[a]_{8\times(5:8)} \quad [b]_{8\times(1:4)}\right] \qquad \ldots (4.1)$$

By doing *DCT* on this newly formed block, author found that there always exist *AC* components of significant energy at four positions in the first row: (0,*j*) for *j*= 1,3,5,7. In particular it was observed that the *AC* energy is 100% packed within these four components for the noise-free case (i.e. all other *AC* components are zero). So instead of dropping all these five coefficients author proposed to drop three components which were: $X_{0,3}, X_{0,5}$ and $X_{0,7}$ by setting them zeros which caused to less visible boundaries. Same phenomena could be applied in vertical direction.

For more smoothness this process could be repeated for several times, i.e. to run the horizontal and vertical processing in an interleaved manner for several times. The reason behind this repeating idea was that new boundaries, though less visible, might have been created after running the processing each time. The experimental results showed that such repeated processing would saturate after a sufficient number of passes.

Author applied the algorithm to a number of *JPEG*-coded images at low bit rates, for *N*=1, i.e. no. of iteration, to *N*=5. It was seen that they did not share any noticeable difference. By comparing them with the original image, they found that the *PSNR* had in fact decreased slightly but as compared with the *JPEG*-coded image, the processed image

showed a much less visible blocking effect. It was also noted that, while blocking effect has been effectively suppressed, the sharpness along edges were basically preserved, because they only modified a minimal number of the *DCT* coefficients. So it can be said that assessing visually and by using *PSNR* does not always give consistent results. This phenomenon is actually well known in low bit rate image/video compression.

Later Luo *et al.* [9] adopted [8] concept of zero masking and analyzed this idea further and gave more computationally efficient process to bring more smoothness in low-bit and high-bit rate compressed images in less time while preserving the sharp edges and suppressing the blocking effects. However, the method author proposed was totally different from that of [8]. In [8], good results were obtained at very low bit-rates, i.e., for images with high blockiness effects, otherwise artifact will appear. Authors' method is capable of bringing the smoothing in the case when the blockiness is not so strong. Authors explored the idea of correlation between the intensity values of the boundary gray scale values of two neighboring blocks in the *DCT* domain. The continuity of the pixels across the boundary is recovered from the continuity of the pixels values of the two neighboring blocks. Authors observed that for the strong texture and edge regions, the nature of the frequencies of the two neighboring blocks differ from each other. Thus this kind of variation introduces artifacts in these non-smooth regions. Therefore authors firstly detected these regions based on the frequency coefficients. If the two $8 \times 8$ neighboring blocks of a boundary have similar frequency properties and the $8 \times 8$ pixel area around the boundary does not have high frequencies then the latter area is declared to be of smooth nature.

For example if we assume two horizontal $8 \times 8$ adjacent blocks *A* and *B*, and let the right half of block *A* and left half of block *B* form a block denoted as *C*. Let $F_a(u,v), F_b(u,v)\ and\ F_c(u,v)$ be the *DCT* coefficients ob block *A*, *B* and *C*.

From mathematical equations and calculation authors deduced some results regarding frequency coefficients which should be modified in order to have less blockiness in non-smooth regions. Authors found that $F_c(u,v)$ has non-zero values only for *v*=0, 1,3,5,7 and *u*=0, which implied that the noise energy of the horizontal boundary discontinuity of block *C* was represented only in some coefficients of the first row of the *DCT* Coefficient array. So in order to reduce the blocking effect between two horizontal adjacent $(8 \times 8)$ blocks, $F_c(u,v)$ for *v*=0, 1, 3, 5, 7 and *u*=0 should be modified.

Before modifying these indexed values *DCT* coefficients authors mentioned that as discussed in [8] the dropping or zeroing the odd numbered *DCT* coefficients without taking into consideration the information about the nature of the image in that neighborhood might result in artifacts. That's why authors emphasized that the conditions that should be met before modifying the blockiness appearance of the relatively smooth regions are:

i. Block *A* has as similar horizontal frequency property as block *B*.
ii. The boundary between block *A* and block *B* belongs to a relatively smooth region.

To save on the number of computations and to meet first condition authors imposed two constraints to modify the coefficients, which are-

a. $|F_a(0,0) - F_b(0,0)| < T1$ …(4.2)

b. $|F_a(0,1) - F_b(0,1)| < T2$ …(4.3)

To address condition (ii) above authors observed that if there was a strong edge between block *A* and block *B*, this edge appeared in block *C*. Thus for condition (ii) to be satisfied, it also has to make sure that block *C* is of low frequency content. The presence of texture or strong diagonal edges would result in relatively high values of the high order *DCT* Coefficients. Form the experiment authors found that to save on number of computation steps it was enough to meet following constraint:

c. $|F_c(3,3)| < T3$ …(4.4)

Here *T1, T2* and *T3* were predetermined i.e. fixed thresholds. From the experiment authors found that *T1=350, T2=120 and T3=60* gave the best results. The selections of these thresholds were based on the observation of the blockiness and experiments. The choices of the thresholds were image and compression ratio dependent.

If the three constraint above are satisfied, the blockiness reduction in the *DCT* domain by modifying the five relevant coefficients of $F_c(u, v)$. The first row of the *DCT* coefficient matrix of block *C* is modified by the weighted average of blocks *A*, *B* and *C*.

The advantage of using weighted average of adjacent block coefficients to modify the *AC* coefficients of block *C*, instead of simply reducing or zeroing these coefficients, is that it is more adaptive to image contents.

Let's assume that $F_{Mc}(0, v)$ be the modified *DCT* coefficient of block *C*. The modification is formulated as follows:

$$F_{Mc}(0, v) = \alpha_0 F_c(0, v) + \beta_0 [F_a(0, v) + F_b(0, v)],$$

where

$$v = 0,1 \quad \ldots (4.5)$$

$$F_{Mc}(0, v) = \alpha_1 F_c(0, v) + \beta_1 [F_a(0, v) + F_b(0, v)]$$

where

$$v = 3,5,7 \quad \ldots (4.6)$$

$$F_{Mc}(u, v) = F_c(u, v)$$

$$\text{For all } v \neq 0,1,3,5,7 \quad \ldots (4.7)$$

where experimental value of $\alpha_0 = 0.6,\ \beta_0 = 0.2\ and\ \alpha_1 = 0.5,\ \beta_1 = 0.25$ .

Then the pixels of block $C(k, l)$, are replaced by $C_M(k, l)$, the inverse *DCT* of $F_{Mc}(u, v)$.

### 4.3 Proposed Approach

As we know that when image is compressed at low bit rates, the visually defecting blocking artifacts generates. The algorithm and computation parameters given by Luo *et al.* [9] are quite efficient in terms of minimizing artifacts and maximizing the visible quality of a decompressed image at low bit rate or very low bit rate.

As stated in [9] that this algorithm is a post processing algorithm so we implemented this algorithm to some standard image which were made to be decompressed at very low bit rates and it was found that however the visible quality of image was getting richer but the *PSNR* value was slightly decreasing. As the image quality is measured in *PSNR (dB)* against corresponding bit rate, so we tried to fill this gap between decompressed image without the algorithm and image recovered after post processing suggested in [9] algorithm.

In order to increase the *PSNR* value while maintaining or increasing the visibility quality that was obtained after post processing the algorithm we used the All Phase Discrete Cosine Transform ( *APDCT* ) [19] anti-aliasing and low pass filter before applying [9] algorithm. Since this is a spatial low pass filter so it allowed all the low frequency gray scale values restricting high frequency values, in contrast it suppressed all the high frequency pixels values. The pre-filtering caused to pass only the low frequency gray scale values and finally when these gray scale values were inputted to this algorithm then this pre-filtering reduced the computation complexity and more than this we achieved better performance values of *PSNR*, in fact the *PSNR* value achieved in proposed algorithm was greater than both the *PSNR* values of decompressed image without [9] algorithm and with [9] algorithm respectively at low bit rates.

### 4.3.1 Design of the Anti-Aliasing Filter Based on *APDCT*

The basic idea of all phase filtering [20] was derived from the superimposing digital filter. This idea was proposed by Hou in [21], which was based on the Walsh, Fourier and *IDCT*.

Here take 7×7 filter for example, and lets have the detailed explanation of the *APDCT* filter.

Let $[z]_{2N-1}$ be the *1-D* unit impulse response vector of all phase sequency filtering, and $F_N$ be the *N-D* response vector, the design of *1-D APDCT* filter with the length of *2N-1* is composed of (4.8) and (4.9).

$$[z_{1/2}]_N = [\, z(0), z(1), \dots, z(N-1)]^T = KF_N \quad \dots (4.8)$$

$$z(n) = z(-n), n = 0,1, \dots, N-1. \quad \dots (4.9)$$

Where the elements of matrix *K* are

$$K(m,n) = \begin{cases} \dfrac{N-m}{N^2}, & m = 0,1, \dots, N-1, n = 0, \\ \dfrac{1}{N^2}\left[(N-m)\cos\dfrac{mn\pi}{n} - \csc\dfrac{n\pi}{N}\sin\dfrac{mn\pi}{N}\right], & m = 0,1,..,N-1, n = 1,2, \dots, N-1 \end{cases} \quad \dots (4.10)$$

Similarly, the method applied to design *1-D* filter can be used to design *2-D* filter. Suppose the unit impulse frequency response matrix of *2-D* filter be$[z]_{(2N-1)\times(2N-1)}$, and $F_{N\times N}$ be the ideal low-pass filter matrix of *N*×*N*, the design of *2-D APDCT* filter with size *(2N-1)* × *(2N-1)* is composed of

$$[z_{1/4}]_{N\times N} = KF_{N\times N}V^T, \quad \dots (4.11)$$

$$z(m,n) = z(-m,n) = z(m,-n) = z(-m,-n), m,n = 0,1, \dots, N-1, \quad \dots (4.12)$$

where the matrix *K* is defined as (4.10). For example, when *N*=4,

$$F_{4\times4} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$ One fourth of the *APDCT* filter is

$$[z_{1]/4}]_{4\times4=} \begin{pmatrix} 0.2500 & 0.1288 & 0.0183 & -0.0221 \\ 0.1288 & 0.0663 & 0.0094 & -0.0114 \\ 0.0183 & 0.0094 & 0.0013 & -0.0016 \\ -0.0021 & -0.0114 & -0.0016 & 0.0020 \end{pmatrix}$$

According to (4.11), the zero-phase symmetric property of all phase filters, the 7×7 *APDCT* filter matrix becomes

$$[z]_{7\times7} = \begin{pmatrix} 0.0020 & -0.0016 & -0.0114 & -0.0221 & -0.0114 & -0.0016 & 0.0020 \\ -0.0016 & 0.0013 & 0.0094 & 0.0183 & 0.0094 & 0.0013 & -0.0016 \\ -0.0114 & 0.0094 & 0.0663 & 0.1288 & 0.0663 & 0.0094 & -0.0114 \\ 0.0221 & 0.0183 & 0.1288 & 0.2500 & 0.1288 & 0.0183 & -0.0221 \\ -0.0114 & 0.0094 & 0.0663 & 0.1288 & 0.0663 & 0.0094 & -0.0114 \\ -0.0016 & 0.0013 & 0.0094 & 0.0183 & 0.0094 & 0.0013 & -0.0016 \\ 0.0020 & -0.0016 & -0.0114 & -0.0221 & -0.0114 & -0.0016 & 0.0020 \end{pmatrix}$$

So we will be using this *APDCT* anti-aliasing and low-pass filter in proposed algorithm. Now we come to our experimental results and values which will verify our proposed algorithm.

### 4.4 Experimental Results

In order to verify our proposed approach, we applied our modified algorithm to several standard images of size 512×512. The quality measure values that we got applying three decompression methods on images at different bit rates is shown in Table 4.1. in Table 4.1 we have taken the comparison of three algorithm that are, the basic *JPEG* algorithm, the algorithm in [9] and the proposed one that we have applied the *APDCT* filter with [9] algorithm. Here size of each image is 512×512. In our experiment the images are compressed and decompressed at different bit-rates and their *PSNR* values are noted down.

**Table 4.1: *PSNR* comparison of proposed approach with existing approaches.**

| Image | Bit Rate | *PSNR* (*dB*) | | |
|---|---|---|---|---|
| | ***bpp*** | **Basic *JPEG*** | **[9]** | **Proposed** |
| Lena | 0.639034 | 35.78721 | 34.97826 | 34.24775 |
| | 0.410587 | 33.6988 | 33.26925 | 33.35673 |
| | 0.319443 | 32.3304 | 32.11007 | 32.55866 |
| | 0.269257 | 31.25481 | 31.19589 | 31.82552 |
| | 0.235943 | 30.4082 | 30.4133 | 31.12371 |
| | 0.213051 | 29.62699 | 29.73108 | 30.4976 |
| | 0.195747 | 28.98775 | 29.16224 | 29.94137 |
| | 0.182648 | 28.44898 | 28.64119 | 29.46692 |

| | | | | |
|---|---|---|---|---|
| Cameraman | 0.571552 | 38.83284 | 36.25304 | 35.33551 |
| | 0.379787 | 35.66638 | 34.34603 | 34.36283 |
| | 0.300735 | 33.82659 | 33.06805 | 33.46834 |
| | 0.257023 | 32.39632 | 32.01204 | 32.60517 |
| | 0.226219 | 31.22943 | 30.9428 | 31.70709 |
| | 0.205257 | 30.32567 | 30.27836 | 31.04602 |
| | 0.188751 | 29.54375 | 29.55129 | 30.36234 |
| | 0.175797 | 28.88331 | 28.99021 | 29.76818 |
| Mandrill | 1.180088 | 34.16665 | 30.98416 | 29.48894 |
| | 0.795166 | 30.98449 | 29.20819 | 28.80957 |
| | 0.610184 | 29.13966 | 27.97505 | 28.08642 |
| | 0.500237 | 27.81529 | 27.02789 | 27.45208 |
| | 0.421543 | 26.78348 | 26.16129 | 26.76762 |
| | 0.365406 | 25.96289 | 25.58685 | 26.26335 |
| | 0.320431 | 25.27269 | 24.96995 | 25.69842 |
| | 0.286297 | 24.68321 | 24.47418 | 25.2202 |
| Jet plane | 0.665134 | 36.55907 | 34.23614 | 33.19898 |
| | 0.432323 | 33.99064 | 32.6677 | 32.43801 |
| | 0.335670 | 32.34289 | 31.40995 | 31.65639 |
| | 0.281834 | 31.14013 | 30.58345 | 31.05554 |
| | 0.246586 | 30.19021 | 29.7732 | 30.42923 |
| | 0.223194 | 29.33056 | 29.1481 | 29.77503 |
| | 0.204773 | 28.59363 | 28.44535 | 29.13383 |
| | 0.189880 | 28.00089 | 27.93186 | 28.64912 |
| Peppers | 0.653950 | 33.91719 | 33.18783 | 32.78939 |
| | 0.392578 | 32.61636 | 32.12728 | 32.15995 |
| | 0.300785 | 31.68093 | 31.34932 | 31.59835 |
| | 0.255356 | 30.84374 | 30.70691 | 31.10314 |
| | 0.224854 | 30.12432 | 30.04433 | 30.55853 |
| | 0.203770 | 29.48971 | 29.54848 | 30.10345 |
| | 0.187798 | 28.84937 | 28.97532 | 29.57284 |
| | 0.176300 | 28.36218 | 28.58437 | 29.20233 |
| Elaine | 0.671173 | 33.07912 | 32.60898 | 32.58693 |
| | 0.384293 | 31.99949 | 31.73737 | 31.961 |
| | 0.28698 | 31.23214 | 31.10286 | 31.43356 |
| | 0.239525 | 30.55003 | 30.57728 | 30.99304 |
| | 0.208233 | 29.93641 | 30.0271 | 30.5008 |
| | 0.187626 | 29.37748 | 29.61188 | 30.11287 |
| | 0.171974 | 28.80774 | 29.10256 | 29.63163 |
| | 0.162182 | 28.32341 | 28.69048 | 29.26507 |
| Lake | 0.88945 | 33.1434 | 31.74044 | 30.7069 |
| | 0.571976 | 31.18101 | 30.26883 | 30.06659 |
| | 0.438515 | 29.91629 | 29.2744 | 29.44897 |
| | 0.364315 | 28.934 | 28.50052 | 28.90765 |
| | 0.315239 | 28.09009 | 27.69764 | 28.25672 |
| | 0.278904 | 27.40882 | 27.22412 | 27.82702 |
| | 0.251835 | 26.83097 | 26.71193 | 27.39553 |
| | 0.231514 | 26.24603 | 26.17192 | 26.88896 |

| | | | | |
|---|---|---|---|---|
| Living Room | 0.884785 | 33.26954 | 31.25176 | 29.59643 |
| | 0.570076 | 31.07984 | 29.819 | 29.10033 |
| | 0.430714 | 29.75167 | 28.85659 | 28.6124 |
| | 0.352772 | 28.79112 | 28.17388 | 28.18471 |
| | 0.301723 | 27.99376 | 27.49501 | 27.70673 |
| | 0.267136 | 27.33035 | 27.03733 | 27.34926 |
| | 0.239834 | 26.71845 | 26.4987 | 26.89575 |
| | 0.218018 | 26.22592 | 26.1095 | 26.52554 |
| Woman Blonde | 0.769913 | 32.55606 | 31.91957 | 30.39008 |
| | 0.470566 | 30.84536 | 30.47209 | 29.85575 |
| | 0.356831 | 29.77999 | 29.54729 | 29.36356 |
| | 0.293198 | 28.9702 | 28.86704 | 28.89949 |
| | 0.252457 | 28.34965 | 28.28504 | 28.46446 |
| | 0.226387 | 27.73888 | 27.8216 | 28.11321 |
| | 0.204685 | 27.22817 | 27.34533 | 27.68801 |
| | 0.189438 | 26.79755 | 26.95415 | 27.3645 |
| Walk Bridge | 1.215088 | 30.00168 | 28.38046 | 27.0365 |
| | 0.788788 | 28.07752 | 27.0136 | 26.55067 |
| | 0.590652 | 26.96947 | 26.1723 | 26.09332 |
| | 0.477921 | 26.14048 | 25.55358 | 25.69811 |
| | 0.39679 | 25.46897 | 24.97744 | 25.24125 |
| | 0.341919 | 24.91666 | 24.56979 | 24.87601 |
| | 0.299316 | 24.44742 | 24.14733 | 24.52371 |
| | 0.268978 | 24.02076 | 23.83138 | 24.22611 |

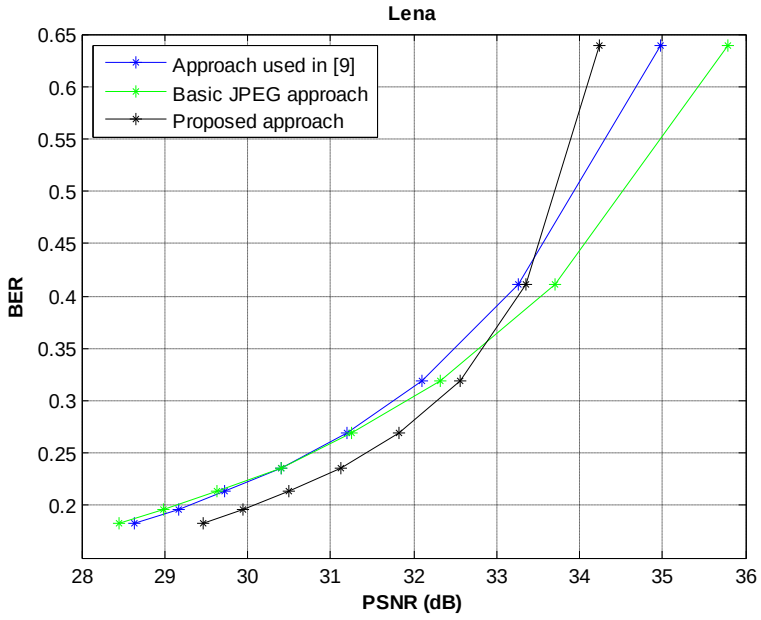


**Fig. 4.1: *PSNR* comparison of Lena image.**

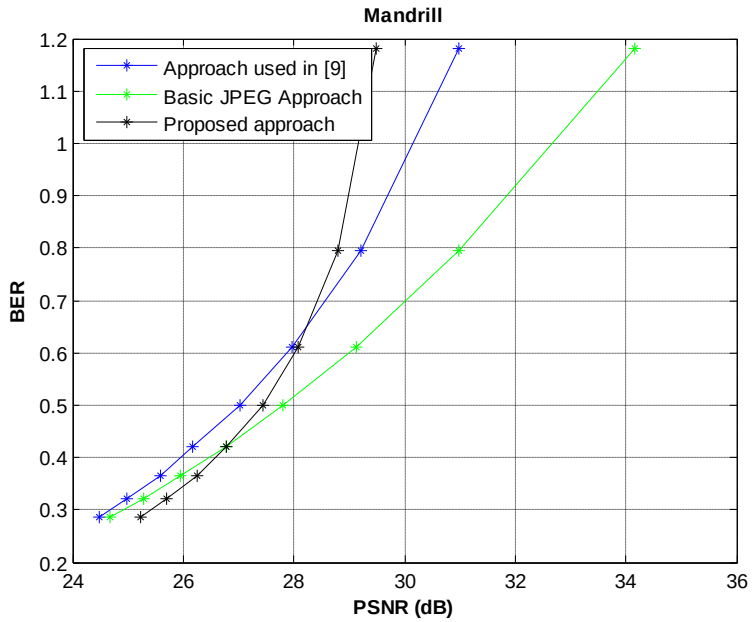


**Fig. 4.2: *PSNR* comparison of Mandrill image.**

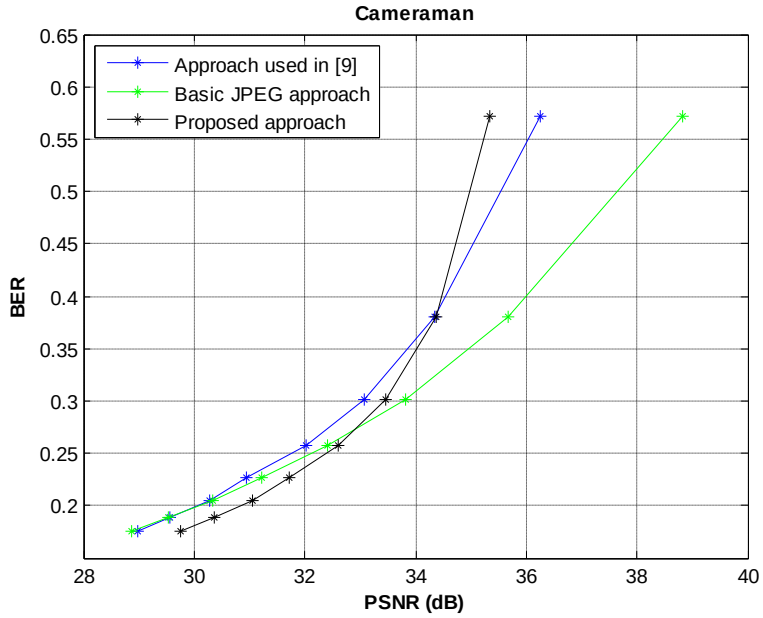


**Fig. 4.3: *PSNR* comparison of Cameraman image.**

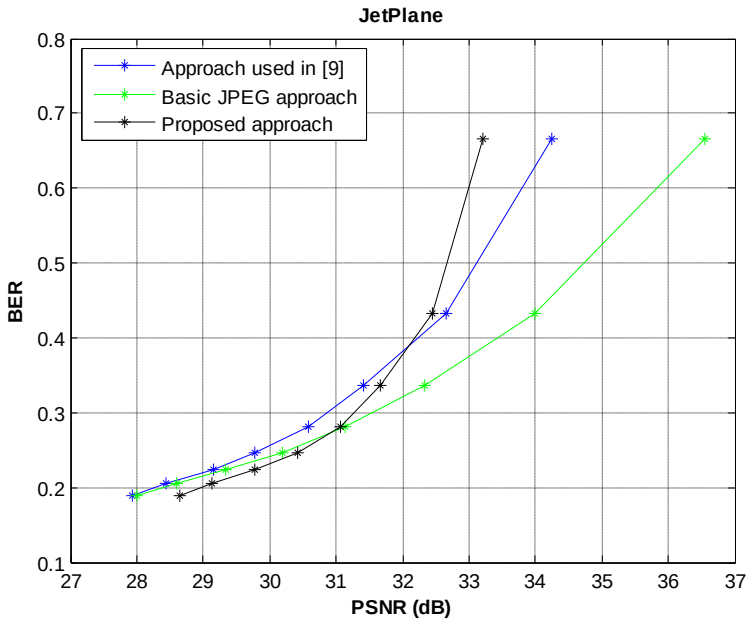


**Fig. 4.4: *PSNR* comparison of Jet Plane image.**

From the Fig. 4.1, Fig. 4.2, Fig. 4.3, Fig. 4.4 and Fig. 4.5, graphically it can easily be noted that the *PSNR* values of our algorithm increases with the bit rate decreases so our algorithm gives slightly better results as the bit rate decreases.

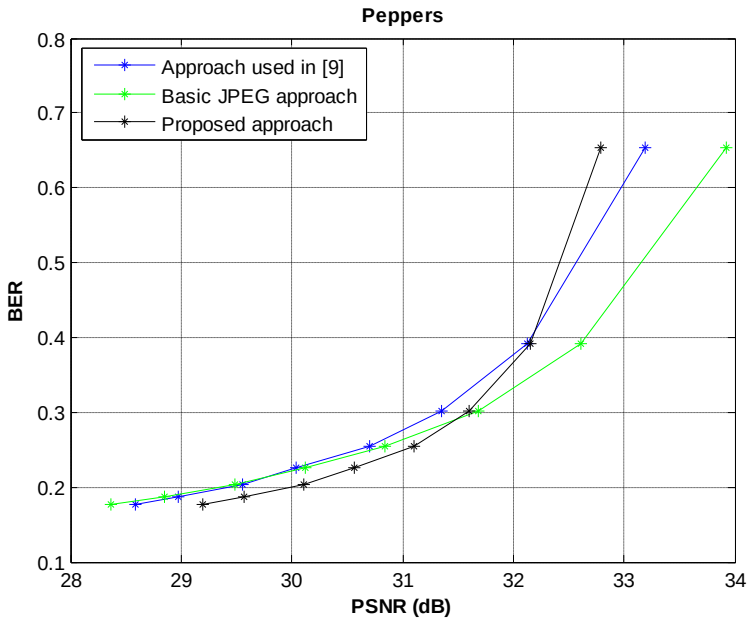


**Fig. 4.5: *PSNR* comparison of Peppers image.**

**Fig. 4.6: Reconstructed image by *JPEG* at 0.213 bpp.**

**Fig. 4.7: Reconstructed image by approach used in [9] at 0.213 bpp.**

**Fig. 4.8: Reconstructed image by proposed approach at 0.213 bpp.**

**Fig. 4.9: Reconstructed image by *JPEG* at 0.182 bpp.**

**Fig. 4.10: Reconstructed image by approach used in [9] at 0.182 bpp.**

**Fig. 4.11: Reconstructed image by proposed approach at 0.182 bpp.**

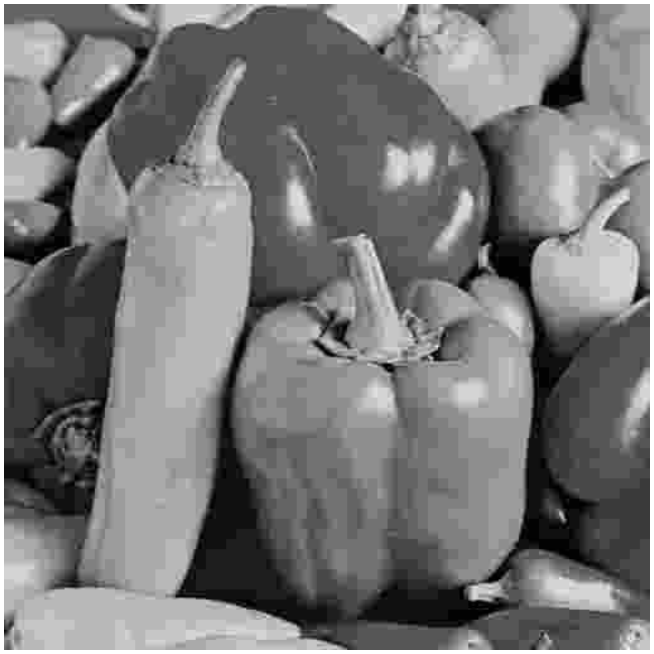

**Fig. 4.12: Reconstructed image by *JPEG* at 0.187 bpp.**

**Fig. 4.13: Reconstructed image by approach used in [9] at 0.187 bpp.**

**Fig. 4.14: Reconstructed image by proposed approach at 0.187 bpp.**

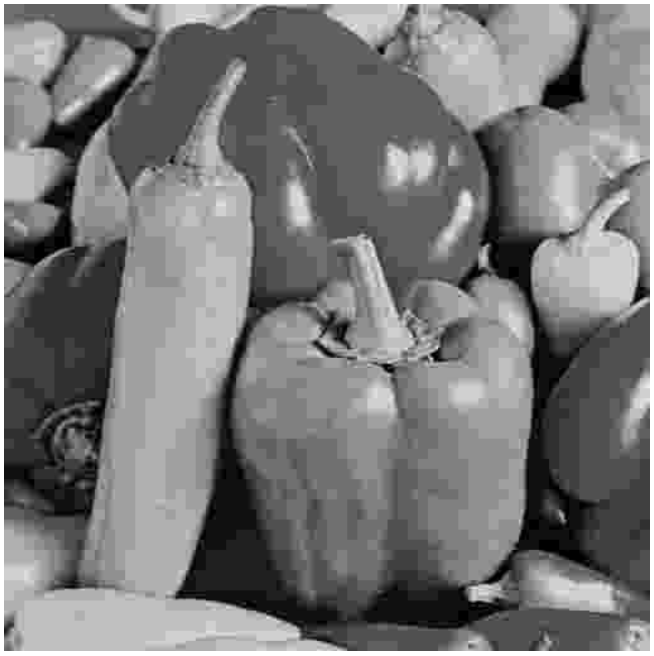

**Fig. 4.15: Reconstructed image by *JPEG* at 0.1763 bpp.**

**Fig. 4.16: Reconstructed image by approach used in [9] at 0.1763 bpp.**

**Fig. 4.17: Reconstructed image by proposed approach at 0.1763 bpp.**

From Fig. 4.6 to Fig. 4.17, it can easily be observed that proposed approach gives more smoothness than that of existing approaches.

**Summary**

In this chapter, we presented the brief overview of the [9] algorithm that has been popular to reduce the blocking effect in images compressed at very low-bit rates and discussed their ideas and gaps and deficiencies in the algorithms in terms of computation efforts and quality measures parameters. Then we presented our proposed algorithm and presented the experimental results after applying our proposed algorithm on the number of standard images available in literature. The *PSNR* values of proposed approach are compared with existing approaches, by compressing different images at different bit rates.

# CHAPTER 5
# CONCLUSION AND FUTURE SCOPE

As *JPEG* images suffers from the blocking artifacts when they are compressed at very low-bit rates, many researchers have suggested many approaches to reduce the artifacts like blocking effects, blurring, ringing effects. In this thesis, we presented an approach which is fusion of spatial domain and frequency domain post-processing approach in which low-pass, narrow band, anti-aliasing *APDCT* filter is used in spatial domain to suppress the high frequency components. It passes the low frequency *DCT* coefficients. Then selective *DCT* coefficients are updated in frequency domain to reduce the blocking artifacts in *JPEG* compressed images. But here we didn't consider the optimal design of better low-pass filters used after the decompression of image. Even though using our approach, we attained better experimental values and better smoothness after compressing the images at very low bit rates.

But apart from blocking, ringing is another annoying artifact that appears frequently in *JPEG* compressed images. The ringing effects come in the images when the high frequency components are suppressed from the images. The future work will be to detect the ringing effects on some sharp-edged images and reduce the ringing effects from the *JPEG* compressed images.

## References


[1] K. Sayood , "Introduction to Data Compression", 3rd Edition, Elsevier, 2006.

[2] R. C. Gonzalez, R. E. Woods and S. L. Eddins, "Digital Image Processing Using MATLAB", 5th Edition, Pearson, 2009.

[3] Y. Yang and N. P. Galatsanos, "Removal of Compression Artifacts Using Projections Onto Convex Sets and Line Process Modeling", IEEE Transaction on Image Processing, Vol. 6, No. 10, pp.1345-1357, 1997.

[4] A. Gresho, "Quantization", IEEE Communication Magazine, Vol. 15, No. 5, pp.16-29, 1977.

[5] S. Singh, V. Kumar and H. K. Verma, "Reduction of Blocking Artifacts in JPEG Compressed Image", Digital Signal Processing, Vol. 17, No.1, pp. 225-243, 2007.

[6] Y. Yang, N.P. Galatsanos and A.K. Katsaggelos, "Regularised Reconstruction to Reduce Blocking Artifacts of Block Discrete Cosine Transform Compressed Image", IEEE Transactions on Circuit and Systems for Video Technology, Vol. 3, No. 6, pp. 421-432, 1993.

[7] A. Zakhor, "Iterative Procedures for Reduction of Blocking Effects in Transform Image Coding", IEEE Transactions on Circuit and System for Video Technology, Vol. 2,No. 1, pp. 91-95, 1992.

[8] B. Zeng, "Reduction of Blocking Effect in DCT-coded Images Using Zero-Masking Techniques", Signal Processing, Vol. 79, No. 2, pp. 205-211, 1999.

[9] Y. Luo and R. K. Ward, "Removing the Blocking Artifacts of Block-Based DCT Compressed Images", IEEE Transactions on Image Processing, Vol. 12, No.7, pp. 838-842, 2003.

[10] G. Aharoni, A. Averbuch, R. Coifman and M. Israeli, "Local Cosine Transform- A Method for the Reduction of the Blocking Effect in JPEG", Journal of Mathematical Imaging and Vision, Vol. 3,No. 1, pp. 7-38, 1993.

[11] H.S Malvar and D.H. Staelin, "The LOT: Transform Coding Without Blocking Effects", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol.37, No. 4, pp. 553-559, 1989.

[12] H. Reeve and J. Lim, "Reduction of Blocking Effect in Image Coding", IEEE International Conference on ICASSP '83, Acoustics, Speech and Signal Processing, Vol. 8, pp.1212–1215, Apr. 1983.

[13] B. Ramamurthi and A. Gersho, "Nonlinear Space-Variant Post Processing of Block

Coded Images", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 34, No. 5, pp. 1258–1268, 1986.

[14] T. Meier, K.N. Ngan and G. Crebbin, "A Region-Based Algorithm for Enhancement of Images Degraded by Blocking Effects", Proceedings of 1996 IEEE TENCON. Digital Signal Processing Applications, Vol. 1, pp. 405–408, Nov. 1996.

[15] Y.L. Lee, H.C. Kim and H.W. Park, "Blocking Effect Reduction of JPEG Images by Signal Adaptive Filtering", IEEE Transactions on Image Processing, Vo. 7, No. 2, pp. 229–234, 1998.

[16] S.V Viraktamath and G.V. Attimarad, "Performance Analysis of JPEG Algorithm", Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011), pp. 629-633, July 2011.

[17] G.K. Wallace, "The JPEG Still Picture Compression Standard", IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, pp. 18-34, 1992.

[18] D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proceedings of IRE, Vol. 40, No. 9, pp. 1098-1101, 1952.

[19] C.-You, Z.-X. Hou, K. He and A.-P. Yang, "JPEG-Based Image Coding Algorithm at Low Bit Rates with Down-Sampling and Interpolation", 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08, pp. 1-5, Oct. 2008.

[20] Z. X. Hou and X. Yang, "The All Phase DFT Filter", Proceedings of the 10th IEEE Conference on Digital Signal Processing Workshop, 2002 and the 2nd IEEE signal Processing Education Workshop, pp. 221-226, Oct. 2002.

[21] Z. X. Hou, "The Construction of the Three Kinds of 2-Dimensional Superimpose Digital Filter," Journal of Tianjin University, Vol. 18, No. 1, pp. 29-41, 1985.

[22] J. Chou, M. Crouse and K. Ramchadran, "A Simple Algorithm for Removing Blocking Artifacts in Block-Transforms Coded Images", IEEE Signal Processing Letters, Vol. 5, No. 2, pp. 33–35, 1998.

[23] H. Paek, R.-C. Kim and S.-U. Lee, "On the POCS-Based Post-Processing Technique to Reduce the Blocking Artifacts in Transform Coded Images", IEEE Transactions on Circuits System for Video Technology, Vol. 8, No.3, pp. 358–367, 1998.

[24] H. Paek, R.-C. Kim and S.-U. Lee, "A DCT-Based Spatially Adaptive Post Processing Technique to Reduce the Blocking Artifacts in Transform Coded Images", IEEE Transactions on Circuits System for Video Technology, Vol. 10, No.

1, pp. 36–41, 2000.

[25] J. Luo, C.W. Chen, K.J. Parker and T.S. Huang, "Artifact Reduction in Low Bit Rate DCT-Based Image Compression", IEEE Transactions on Image Processing, Vol. 5, No. 9, pp. 1363–1368, 1996.

[26] J. Luo, C.W. Chen, K.J. Parker and T.S. Huang, "Artifact Reduction in Low Bit Rate DCT-Based Image Compression", IEEE Transactions on Image Processing, Vol. 5, No. 9, pp. 1363–1368, 1996.

[27] T. Meier, K. N. Ngan and G. Crebbin, "Reduction of Blocking Artifacts in Image and Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 3, pp. 490-500, 1999.

[28] F.X.Coudoux, M.Gzalet and P.Corlary, "Reduction of Blocking Effecting DCT-Coded Images Based on a Visual Perception Criterion", Signal Processing: Image Communication, Vol. 11, pp.179-186, 1998.

[29] A. Z. Averbuch, A. Schclar and D. L. Donoho, "Deblocking of Block-Transform Compressed Images Using Weighted Sums of Symmetrically Aligned Pixels", IEEE Transactions on Image Processing, Vol. 14, No. 2, pp. 200-212, 2005.

[30] J. J. Zon and H. Yan, "A Deblocking Method for BDCT Compressed Images Based on Adaptive Projections", IEEE Transactions on Circuits Systems and Video Technology, Vol. 15, No. 3, pp. 430-434, 2005.

[31] C.-S. park, J. Hyungkim and S.-Jeako, "Fast Blind Measurement of Blocking Artifacts in Both Pixel and DCT domain", Journal of Mathematical Imaging and Vision, Vol. 28, No. 3, pp. 279-284, 2007.

[32] S. Minami and A. Zakhor, "An Optimization Approach for Removing Blocking Effects in Transform Coding" , IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 2, pp. 74–82, 1995.

[33] G. Lakhani and N. Zhong, "Derivation of Prediction Equations for Blocking Effect Reduction", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 3, pp. 415–418, 1999.

[34] G.A.Triantafyllids, D.Tzovaras and M.G.Strintzis, "Blocking Artifact Detection and Reduction in Compressed Data", IEEE Transactions on Circuits Systems and Video Technology, Vol. 12, No. 10, pp. 877-890, 2002.

[35] S.Liu and A.C Bovik, "Efficient DCT-Domain Blind Measurement and Reduction of Blocking Artifacts", IEEE Transactions on Circuits Systems and Video Technology,

Vol.12, No. 12, pp. 1139-1149, 2002.

[36] C. Wang, W-J. Zhang and X.-Z. Fang, "Adaptive Reduction of Blocking Artifacts in DCT Domain for Highly Compressed Images", IEEE Transactions on Consumer Electronics, Vol. 50, No. 2, pp. 647-654, 2004.

[37] W. B. Mikhael and P. Ragothaman, "An Efficient Image Representation Technique Using Vector Quantization in Multiple Transform Domains", Circuits, Systems and Signal Processing, Vol. 24, No.1, pp. 19-33, 2005.

[38] C.-S. Park, J.-H. Kim and S.-J. Ko**, "**Fast Blind Measurement of Blocking Artifacts in Both Pixel and DCT Domains", Journal of Mathematical Imaging Vision, Vol. 28, No. 3, pp. 279–284, 2007.

[39] S.-C. Tai, B.-J. Chen and M. Choi, "An Efficient Method for the Detection of Ringing Artifacts and De-Ringing in JPEG Image", Picture Coding Symposium, (PCS) 2010, pp. 578-581, Dec. 2010.

[40] T. Q. Nguyen and D˜. T. Vo, "Localized Filtering For Artifacts Removal in Compressed Images", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011, pp. 1269-1272, May 2011.

[41] S. Chacko, L. T. Joseph and J. Jayakumar, "Artifacts Removal and Edge Detection of Digitally Compressed Images", International Journal of Scientific and Engineering Research, Vol. 3, No. 4, pp. 1-6, 2012.