January 2015

Removal Of Blocking Artifacts From JPEG-Compressed Images Using An Adaptive Filtering Algorithm

Riddhiman Goswami

REMOVAL OF BLOCKING ARTIFACTS FROM JPEG-COMPRESSED IMAGES USING AN ADAPTIVE FILTERING ALGORITHM

by

Riddhiman Goswami

Master of Science, University of North Dakota, 2015

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

In partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota

May

2015

This thesis, submitted by Riddhiman Goswami in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

_____

Dr. Ronald Marsh

_____

Dr. Travis Desell

_____

Dr. Eunjin Kim

This thesis is being submitted by the appointed advisory committee as having met all the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

_____

Wayne Swisher

Dean of the School of Graduate Studies

_____

Date

PERMISSION

Title	Removal of blocking artifacts from JPEG-compressed images using an adaptive filtering algorithm
Department	Computer Science
Degree	Master of Computer Science

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in his absence, by the chairperson of the department or the dean of the graduate school. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Signature _____Riddhiman Goswami_____

Date _____5/5/2015_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to the members of my advisory committee for their guidance and support during my time in the master's program at the University of North Dakota. I am immensely grateful to my advisor and Chairperson Dr. Ronald Marsh, and my committee members Dr. Travis Desell and Dr. Eunjin Kim. Without the guidance and the time they have generously provided, this thesis would not have been possible.

In particular, I wish to thank Dr. Ronald Marsh for agreeing to be my advisor, for introducing me to this fascinating topic, for all the invaluable lessons he taught me about Image Processing and Computer Graphics, and for all the useful comments and remarks he has contributed over the last couple of years. I appreciate the guidance and encouragement he has given me throughout my academic years at UND, and especially throughout the thesis process.

In addition, I am grateful to Dr. Travis Desell and Dr. Eunjin Kim for agreeing to serve as a members of my committee and for the support and assistance they have provided throughout my academic program at UND.

I would particularly like to thank my family, friends and my fellow students at UND who have supported me throughout this entire process, both by keeping me harmonious and helping me put the pieces together.

ABSTRACT

The aim of this research was to develop an algorithm that will produce a considerable improvement in the quality of JPEG images, by removing blocking and ringing artifacts, irrespective of the level of compression present in the image. We review multiple published related works, and finally present a computationally efficient algorithm for reducing the blocky and Gibbs oscillation artifacts commonly present in JPEG compressed images. The algorithm alpha-blends a smoothed version of the image with the original image; however, the blending is controlled by a limit factor that considers the amount of compression present and any local edge information derived from the application of a Prewitt filter. In addition, the actual value of the blending coefficient (α) is derived from the local Mean Structural Similarity Index Measure (MSSIM) which is also adjusted by a factor that also considers the amount of compression present. We also present our results as well as the results for a variety of other papers whose authors used other post compression filtering methods.

CHAPTER I

INTRODUCTION

Image compression is the minimization of the size of a graphics file, by reducing the number of bytes in the image while keeping the quality and visual integrity of the image within acceptable standards. The primary advantages of image processing are transmission and storage.

With the rise of the popularity of the Internet as the principal medium of communication, the demand for faster transmission of data has never been more paramount. High-speed transfer of data in graphical format, via remote video communication, data from GPS and geo-stationary satellites, or instant image sharing hosts over the Internet, all benefit from reduction of the size of the graphic files via image compression, as smaller file sizes will require less bandwidth to be transferred over a network. The other benefit from reducing the size of the image files is the amount of data that can be stored in a storage device. The price of storage devices has been greatly reduced and the technology behind them has vastly improved over the last decade, but the compression of image files in order to increase the amount of data storage is still highly desirable.

Image compression can be broadly classified into lossless and lossy compression. As their names suggest, in lossless compression all the data from the original image is preserved after it has been compressed. Some popular methods of lossless encoding

include run-length encoding, which takes identical and consecutive data elements and replaces them with a single value of that element and a count of the number of bits replaced, and DEFLATE [31], a compression technique that employs a combination of the LZ77 [32, 33] algorithm and Huffman coding. These methods can be used for BMP files and PNG files respectively.

In lossy compression, a significant amount of data from the original image file is lost, which may sometimes result in the degradation in the quality of image. One of the most popular lossy compression technique is the standard set by the Joint Photographic Experts Group, or as commonly referred to by its acronym, JPEG [34].

JPEG is an extremely popular algorithm for lossy image compression, which sees extensive usage in digital images. The algorithm allows the users some freedom regarding the amount of compression to be used, which usually is a trade-off between the file size and the quality of the image. Performing optimally, the JPEG algorithm can produce compression ratios up to 10:1 with very little loss of quality in the compressed image.

The standard for the JPEG algorithm was created by the Joint Photographic Experts Group, a committee created by two working groups, ISO (International Organization for Standardization) and ITU-T (International Telecommunication Union). The actual standard for JPEG algorithm is formally known as ISO/IEC IS 10918-1 | ITU-T Recommendation T.87 [34].

The JPEG compression algorithm, commonly used for digital images and video, works best when applied to photographs and paintings of realistic scenes that contain smooth

variations of tone and color. JPEG is not well suited for line drawings and other textual or iconic graphics, where the sharp contrasts between adjacent pixels can cause noticeable artifacts. Due to the lossy nature of the algorithm, which leads to loss of image data and creation of artifacts in the compressed image, JPEG is not suited for applications like scientific visualization or medical applications, which require exact reproduction of the original image data. The image in Figure 1 is an example of how the loss of data from the JPEG compression can lead to production of artifacts and distortions. The left part of the image has a higher amount of compression than the right part, as a result the left side shows a significant amount of loss of detail compared to the rest of the image.

Figure 1. Loss of image data in JPEG compression; Source: Wikimedia commons

JPEG uses a block-based discrete cosine transform (BDCT) coding scheme where an image is first divided into 8 x 8 non-overlapping blocks. Each block is then transformed using the discrete cosine transform (DCT), followed by quantization and variable length coding. Discrete cosine transform is a Fourier-like transform in which the basis function consists only of cosines (but no sines). Just like the discrete Fourier transform, the DCT

also operates on discrete, finite sequences. It can be shown, that the use of just the cosines in expressing a finite set of data points leads to efficient compression by reducing the size of the basis of the transform space. The DCT finds multiple applications in lossy compression because using this method the majority of the signal information is found within a range of low-frequency components, which are not lost.

JPEG encoding provides users with some control over the compression process, allowing for tradeoffs between image quality and file size (compression ratio). Higher compression ratios may result in undesirable visual artifacts in the decoded image such as blockiness (artificial discontinuities along the block boundaries) and Gibbs oscillations (ringing artifacts near strong edges).

Modern displays commonly use an 8 bit value to encode each color band (R, G, and B) producing a 24-bit pixel encoding. As JPEG compresses each color band of an image independently, color representations with high correlation between bands (like RGB) are not a good match for JPEG. Conversely, color representations with low correlation between bands (like Y'CbCr) are.

The JPEG encoding process consists of several steps:

1. Color Space Transformation - Convert the image color space from RGB to Y′CbCr. This is done to allow for greater compression as the brightness information (Y'), which is more important to the perceptual quality of the image, is confined to a single channel.

2. Downsample - Reduce the resolution of the two chroma bands (Cb & Cr) by a factor of 2.

3. Block Splitting - Split each band into blocks of 8×8 pixels,

4. Discrete Cosine Transform – Each block is converted to a frequency-domain representation, using a normalized, two-dimensional type-II discrete cosine transform (DCT):

$$G_{u,v} = \sum_{x=0}^{7} \sum_{y=0}^{7} \propto (u) \propto (v) g_{x,y} \cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y + \frac{1}{2}\right)v\right]$$

5. Quantization – As human vision is more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components.

6. Entropy Coding - The resulting data for all 8×8 blocks is further compressed with a variant of Huffman encoding (lossless).

A substantial body of literature addresses the question of reducing visual artifacts in the decoded image. Post processing approaches are common. One such approach is to use adaptive spatial filtering [13] or an adaptive fuzzy post-filtering approach [6]. These techniques commonly involve classification and low-pass filtering. They typically classify each block has having strong edges or weak edges and then apply a set of predefined spatial filters. The effectiveness of these methods is highly dependent on block characteristics and on the specific filter design.

The aim of this research was to develop an algorithm that will produce a considerable improvement in the quality of JPEG images, irrespective of the level of compression

present in the image. With that in mind, for this research, an adaptive low-pass filtering approach was chosen, as low pass filtering can potentially be applied to any image, irrespective of its compression level. An adaptive computationally efficient algorithm for reducing the artifacts, while preserving the level of quality in the images, is presented. The algorithm alpha-blends a smoothed version of the image with the original image; however, the blending is controlled by a limit factor that considers the amount of compression present and any local edge information derived from the application of a Prewitt filter. In addition, the actual value of the blending coefficient (α) is derived from the local Mean Structural Similarity Index Measure (MSSIM) which is also adjusted by a factor that also considers the amount of compression present.

CHAPTER II

RELATED WORK AND RESEARCH ON BLOCKING AND RINGING ARTIFACTS

When a digital image is compressed using a block transform algorithm, like the JPEG algorithm, the core principle of the compression algorithm leads to the creation of artifacts in the compressed image. In JPEG, the Discrete Cosine Transform is applied to 8x8 block of pixels, followed by a quantization of the transform coefficients of each block. The greater the compression, the coefficients are quantized more coarsely. Since the quantization is applied to each 8x8 block separately, the DCT coefficients of each block will be quantized differently. This leads to discontinuities in color and luminosity between the neighboring blocks. This phenomenon is especially perceptible in areas of the image where there is a lack of complex detail that can camouflage the artifact. These artifacts are commonly referred to as Blocking artifacts, as demonstrated in the Figure 2. The left and right part of Figure 2 are the uncompressed and highly compressed versions of the same image respectively, the compressed version displaying blocking artifacts.

Figure 2. Example of blocking artifacts; Source: Wikimedia commons

Another type of artifact caused by the quantization of the coefficients is Ringing artifact. Ringing artifacts are a class of artifacts that occur specifically due to low-pass filter processing. To understand how they occur, a short account of the Gibb's phenomenon is in order.

In 1898 Albert Michelson devised a method to compute the Fourier series decomposition of a periodic signal and then resynthesize those components to reproduce the original signal. He found that resynthesizing a square wave always gave rise to oscillations at jump discontinuities. In fact his results were consistent to that of several other contemporary mathematicians. However, it was later shown by J. Willard Gibbs in 1899 that contrary to popular belief these oscillations were not actually a manifestation of mechanical flaws of the device used. The overshoot of the discontinuity is of a mathematical origin; no matter how many higher harmonics are considered, the oscillations never die down, they just approach a finite limit. This came to be known as the Gibb's phenomenon.

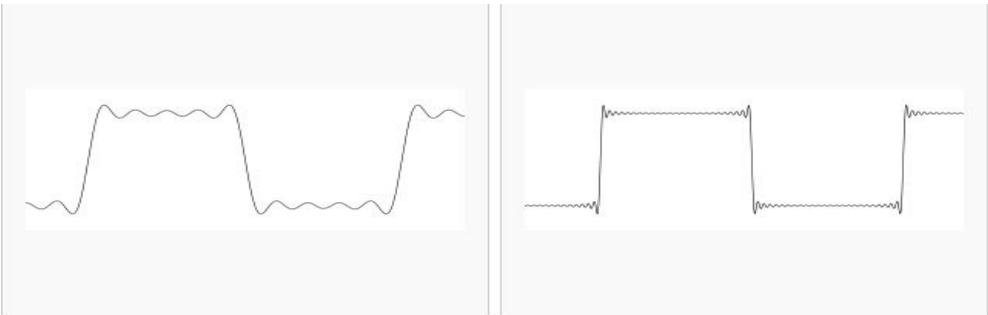


Figure 3. Gibb’s oscillation; Source: Wikimedia commons

So, when a signal is made to go through a low-pass filter, the response overshoots above and then undershoots below the steady-state value thus causing the formation of a ring, as

shown in the following Figure 4. In JPEG compression such artifacts may arise due to two reasons.

1. After performing DCT on the 8x8 blocks, due to filtering, the higher frequency components are lost. Which cause the response to oscillate between an overshoot and an undershoot causing "ringing".
2. Splitting the image in 8x8 blocks always run the risk that the number of blocks may not be an integer or an integer multiple of 8. The edges present in the image might be encoded poorly, spanning across multiple uneven blocks, leading to formation of rings.

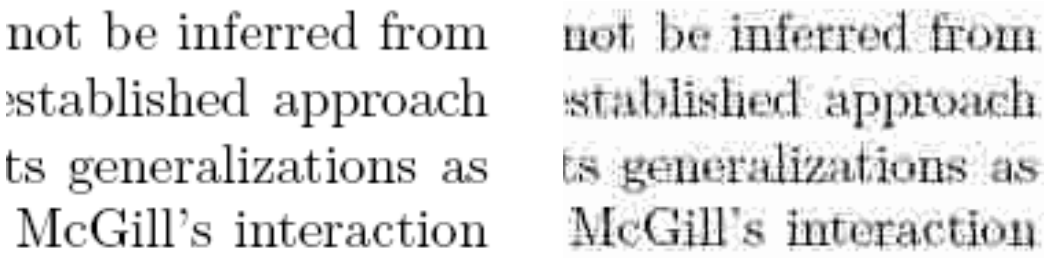


Figure 4. Ringing artifacts; Source: Wikimedia commons

Low-pass filtering and frequency domain techniques

Lee et al [22] proposed a post processing algorithm that reduces blocking artifacts, staircase noise and corner outliers. Staircase noise are created when a transform block in the image contains an image edge. This causes the edge to degrade and results into formation of step-like artifacts along the image edge, as shown in Figure 5(a). Corner outliers are formed when, after the quantization of the DCT coefficients, the corner-most pixel in a transform block has far greater or smaller value than its neighboring blocks.

Figure 5(b) highlights a corner outlier pixel, which is easily distinguishable in the Figure 5(a) as its pixel value is far greater than its neighboring pixels.

Their proposed algorithm is a four step process. First, they use an edge detection operator to create the absolute gradient magnitude of the image. They use the Sobel operator for this purpose. This is followed by generating an edge map by thresholding the gradient image obtained from the Sobel filter. Further, using this edge map, the compressed image is classified into an edge area or monotone area. Similarly, a local edge map is also generated for each 8x8 transform block. A global and local threshold value are calculated.

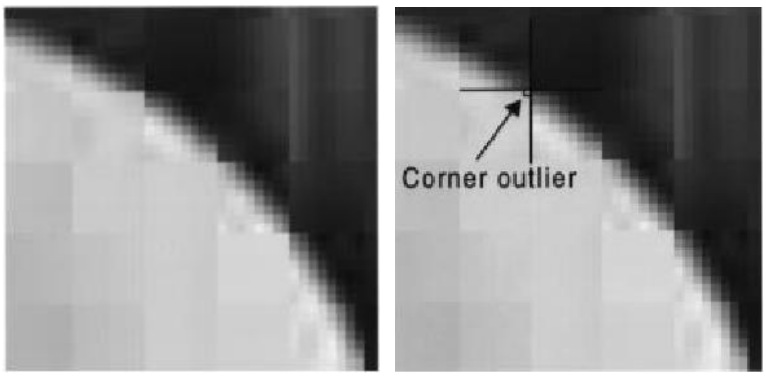


Figure 5 (a). Staircase Noise, (b). Corner outlier; Source: Lee et al [22]

Their proposed algorithm is a four step process. First, they use an edge detection operator to create the absolute gradient magnitude of the image. They use the Sobel operator for this purpose. This is followed by generating an edge map by thresholding the gradient image obtained from the Sobel filter. Further, using this edge map, the compressed image is classified into an edge area or monotone area. Similarly, a local edge map is also generated for each 8x8 transform block. A global and local threshold value are calculated.

If the nth block does not contain much variation in it, the ratio $\frac{\sigma_n}{m_n}$, where σ_n is the standard deviation and m_n is the mean of the nth block of the gradient image generated by the Sobel filter, is closer to zero, which results in the local threshold value T_n to be closer to T_g, the global threshold value. On the other hand, if the block contains sufficient variation, the ratio $\frac{\sigma_n}{m_n}$ in increased and generates the locall threshold value, that is much smalller than its global counterpart. A local edge map is generated using this local threshold value within 6x6 pixels of the nth block, excluding the edge pixels, so that the artifacts near the edge pixels, if any, are not detected as image edge details.

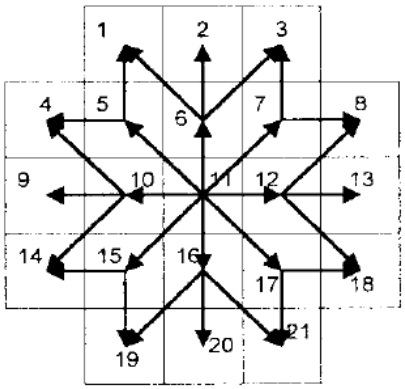
Second, a smoothing filter is introduced to the JPEG compressed that removes both block artifacts in the areas with less variance and staircase noise along the edges of the image. The algorithm applies a 1-D directional smoothing filter along the edges on all points on the edge map. The direction of all the edges on the edge map is calculated as:

$$\theta_e(x, y) = Q[\theta(x, y)] - 90^o$$

Where, $\theta_e(x, y)$ is the direction of the edge at location (x, y), Q is the quantization factor and $\theta(x, y)$is the direction of the gradient vector calculated by the application of the Sobel filter. The purpose of the edge-directional smoothing filter is to reduce the staircase noise generated along the edges in the image.

Third, an adaptive 2-D lowpass filter is applied to the image. This step removes the blocking artifacts that are produced within the areas of the image that does not contain

much variance.



0	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	1	1	0

0	1	1	1	0
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
0	1	1	1	0

Figure 6 (a). Application of filtering, (b). kernel 1, (c). kernel 2; Source: Lee et al [22]

Two separate kernels are used for filtering, as shown in Figure 6(b) and 6(c) respectively. Figure 6(a) represents the manner in which the adaptive filtering, as described below:

If the center pixel of the 5x5 block, that is being filtered, contains an image edge, no filtering is applied. Whether or not the center pixel is an edge can be determined from the global edge map or local edge map of the block.

- If no edge pixel is contained within the block, center or otherwise, then average mean filtering is applied to the block by convoluting it with kernel a.
- If the block does contain an edge pixel, not in the central point but the neighboring pixel around the center, a weighted average filtering is applied to the block by setting the pixels containing the edges and their neighboring pixels to 0, followed by convoluting the remaining pixels of the block with the weights from the kernel b and calculating the average of the block.

Finally, a 2x2 block window is used to detect and remove any corner outliers in the image. The 2x2 window compares the difference between the corner pixel, its neighbors and a threshold value, which is 20% of the global threshold previously determined, and determines if the corner pixel is greater or smaller than its neighbors by a large margin. Upon detection of a corner outlier, the pixel and its immediate neighbors are re-calculated as α and α_1:

$$\propto = Round[3a + b + c + d/6]$$

$$\propto_1 = Round\left[\frac{2a_1 + \propto}{3}\right]$$

$$\propto_2 = Round\left[\frac{2a_2 + \propto}{3}\right]$$

Where, a, b, c and d are the corner outlier and the corner pixels of the neighboring blocks, and a_1 and a_2 are the pixels next to corner outlier within the block.

Singh et al [13] proposes a novel procedure that models a blocking artifact between two neighboring block of pixels. Using the model they detect the presence of any blocking artifact and remove them by filtering adaptively, based on the Human Visual System.

The Human Visual System model is based on the biological and psychological processes of human perception of images, colors and luminosity. In image and video processing, it is often used to take advantage of the capabilities and limits of human vision to optimize existing technology to deliver optimum results as the human eye can perceive without loss of quality. The HVS model states that the human eye can perceive changes in luminosity better than change in color, and cannot perceive high frequency details of an

image, thus allowing the high frequency components to be quantized without noticing the loss of image data.

For two adjacent blocks, b1 and b2, after a DCT quantization, an artifact may be created between the pair, due to each block being quantized separately. This artifact can be modeled by a 2-d step function and can be simulated as a new block created from the existing blocks b1 and b2.

The step function is defined as:

$$s(i,j) = \begin{cases} -\frac{1}{8}, \forall\ i\epsilon[0,7],\ \ j\epsilon[0,7] \\ \frac{1}{8}, \forall\ i\epsilon[0,7],\ \ j\epsilon[0,7] \end{cases}$$

And the new block b is derived as:

$$b(i,j) = \ \beta s(i,j) + \mu + r(i,j),\ \ \forall\ i,j\ \epsilon[0,7]$$

Where, β is the amplitude of s, the 2-d step function, μ is the mean of the block b, and r is the residual block.

Upon application of DCT upon the block b, the above mentioned parameters are computed as:

$$\mu = \frac{B(0,0)}{8},\ \ \beta = \sum_{j=0}^{7} v_j B(0,j)$$

Applying DCT on the residual block r, the transformed block R is derived as follows:

$$R = B, \quad R(0,0) = 0, \quad R(0,i) = R(0,i) - \beta v_i, \forall i \in [0,7]$$

Applying the parameters calculated above into the HVS model:

$$\eta = \frac{|\beta|}{\left(1 + A_{total}^{h}\right)(1 + (\mu/\mu_0)^{\gamma})}$$

Where, A_{total}^{h} is the horizontal activity in the block b. μ_0 is set to 150 and γ is set to 2.

If the visibility η of a DCT block is less than a threshold value, τ, then no filtering is applied to that block. If the visibility is greater or equal to the threshold value, then different filtering techniques are applied to that block, based on whether it falls in a smooth or non-smooth region.

A block, or a region of blocks, is classified as smooth or non-smooth based on its frequency properties. If the two blocks b1 and b2 have similar frequency and the block b between them, that is comprised of the edge between the b1 and b2, does not contain high frequency components, then b1 and b2 are classified as smooth regions, and vice versa.

If a block is classified as smooth, blocking artifacts are reduced by modifying the first row of the DCT coefficient matrix of block b, using the weighted average of adjacent blocks b1, b2 and b as:

$$M_B(0,j) = \alpha_0 B(0,j) + \beta_0[B_1(0,j) + B_2(0,j)], \ \ \forall j \in [0,1]$$

$$M_B(0,j) = \alpha_1 B(0,j) + \beta_1[B_1(0,j) + B_2(0,j)], \ \ \forall j = 2n+1, n \in [1,3]$$

$$M_B(0,j) = B(0,j), \quad \forall j = 2n, \ \ n \in [1,3]$$

Where $\alpha_0 + 2\beta_0 = 1$ and $\alpha_1 + 2\beta_1 = 1$, and the values of the constants $\alpha_0 = 0.6$, $\beta_0 = 0.2$, $\alpha_1 = 0.5$, $\beta_1 = 0.25$ are chosen by trial and error, varied from 0.0 to 1.0.

If a block is classified as non-smooth, then the previous technique is not applied, as it may lead to increase in artifacts present in the compressed image. Instead, a smoothing Sigma filter is used. The Sigma filter, based on the sigma probability of the Gaussian distribution, smooths the image noise by averaging only those neighborhood pixels which have the intensities within a fixed sigma range of the center pixel, preserving the image edges and finer details. A 5x5 window is chosen for the filter, as it is found to return best results by experimentation.

Singh et all [27] further improves the above work by classifying the blocks as smooth, non-smooth and intermediate, and using an adaptive pixel replacement algorithm. This approach improves the result of their previous attempt at preserving the detail of the image with minimum loss of image data. This method also reduces the complexity and computational overload by a considerable extent.

Filtering is applied to the compressed image based on the classification of the area of the image, and its frequency properties, the filter is being applied to. The smooth regions contain low frequency components, the non-smooth areas contain high frequency, and the regions classified as intermediate contain mid-range frequency components. As before, to determine whether an area is smooth, non-smooth or intermediate- from two adjacent 8x8 blocks, b1 and b2, 4 pixels are taken from either side of the block boundary, thus forming a block b comprising of half of each of the two adjacent blocks. The variation in pixels within b is calculated as:

$$A(p) = \sum_{k=1}^{7} \emptyset(p_k - p_{k+1})$$

Where, $A(p)$ is the block boundary activity and p are the pixel intensities of the block.

The number of pixels to be modified by filtering depend on the type of region being filtered. Six pixels are modified while filtering a smooth region, as highlighted in Figure 7, four for non-smooth regions and only two for the regions classified as intermediate. If the block boundary activity of two adjacent blocks is less than a threshold value, i.e., $(p) < T_1$, then block b, the intermediate block that comprises of half of either of the adjacent blocks, is classified as smooth. An offset value at the block boundary is calculated as:

$$Offset = |(M + L) - (N + O)|$$

Where M, L, N, and O are pixels across the boundary between the two adjacent blocks.

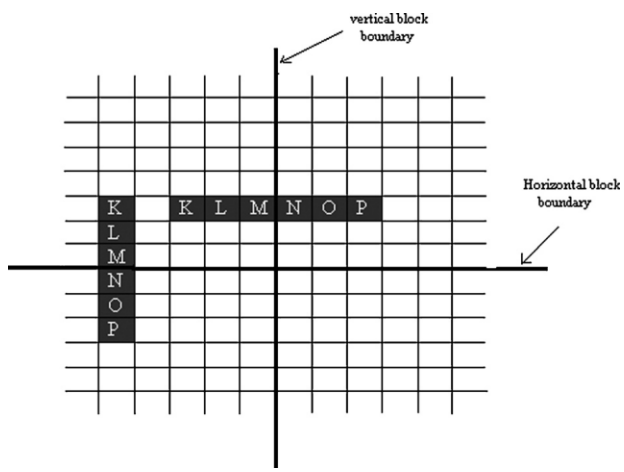


Figure 7. Pixels adjacent to the block boundary; Source: Singh et all [27]

A second condition is used to confirm the smooth nature of the region. For the region to be smooth, $|(M + L) - (N + O)| < 2Q$, where Q is the quality parameter of the JPEG algorithm. If the parameters hold true, then filtering is applied as:

$$If\ (M+L) > (N+O)$$

$$k = K - \frac{offset}{8};\ \ l = L - \frac{Offset}{6};\ \ m = M - \frac{Offset}{4}$$

$$Else\ if (M+L) < (N+O)$$

$$k = K + \frac{offset}{8};\ \ l = L + \frac{Offset}{6};\ \ m = M + \frac{Offset}{4}$$

Similarly, if $A(p) > T_2$, i.e., the block boundary activity is greater than the threshold value T_2, the region can be classified as non-smooth. The offset is calculated the same way as before.

$$Offset = |(M+L) - (N+O)|$$

The second condition confirming the non-smooth nature of the region is $|(M+L) - (N+O)| < Q$, where Q is the quality parameter of the JPEG algorithm. If the parameters hold true, then filtering is applied as:

$$If\ (M+L) > (N+O)$$

$$l = L - \frac{Offset}{6};\ \ m = M - \frac{Offset}{4}$$

$$Else\ if (M+L) < (N+O)$$

$$l = L + \frac{Offset}{6};\ \ m = M + \frac{Offset}{4}$$

Finally, the region is classified as intermediate region if any one of the following conditions are fulfilled:

$$T_1 < A(p) < T_2$$

$$A(p) < T_1 \ \ \&\& \ \ |(M + L) - (N + O)| > \ 2Q$$

$$A(p) < T_2 \ \ \&\& \ \ |(M + L) - (N + O)| > \ Q$$

If the blocks are classified as an intermediate region, then a 3x3 smoothing low pass filter is applied to the pixels M and N on either side of the block boundary.

Saig et al [1] propose the use of optimal interpolation and decimation filters in block coders. Like most algorithms that depend on a block-based approach, typical block coders are of high speed and low-complexity, and perform reasonably well, but suffer from the creation of artifacts in the decoded image while processing images of low bit-rate.

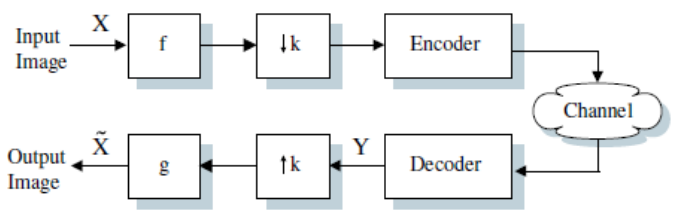


Figure 8. Flowchart of a block coder

A typical block coder takes an input image X, as depicted in Figure 8, and down-samples X by a factor of k after convolving it with a filter f. The image is then encoded using the respective encoding algorithm. While decoding the image, X is up-samples it by a factor of k again and convolves the result with a filter g, producing the result. The proposed algorithm works in two parts: initially, determining an optimal framework for the

interpolation filter g, and finally, determining an optimal framework for the decimation filter (f) with respect to the previously determined optimal interpolation filter (g).

Application of the optimal interpolation (g) and decimation (f) filters in the block coder algorithm returns reasonable improvement in quality over the original decoded image.

Kieu et al [2] proposes a technique to reduce the blocking artifacts, created in the low activity regions of the image, using a ternary polynomial surface to model the pixel intensities of the image and recover the image details lost during quantization by compensating the DCT coefficients lost during quantization.

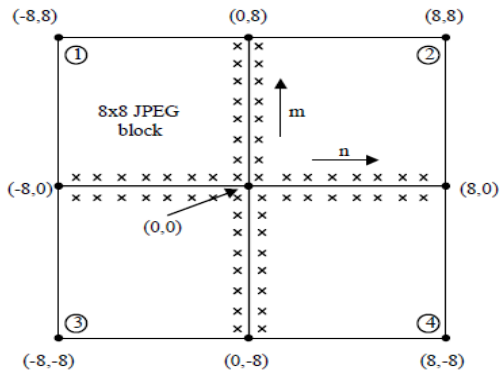


Figure 9. Macro block

In Figure 9, a 2x2 macro block of the JPEG image, the highlighted boundaries are the targets areas where blocking artifacts will be removed from. The ternary surface modeling the image intensity for the 2x2 macro block is calculated, and linear programming techniques are applied to minimize the difference between the pixel values across the block boundary of the macro block, while compensating for the quantization error introduced in the images after the JPEG compression.

Abboud [7] present a simple adaptive low-pass filtering approach that reduces the blocking artifacts present in the image without degrading the quality of the image. This approach exploits a property of HVS, such that the human vision is more sensitive to blocking artifacts present in smoother areas than areas that have a lot of activity. The algorithm classifies regions of the image into highly smooth, relatively smooth, and highly detailed, and then applies strong filtering and weak filtering respectively.

To classify the image into different type of regions, the following function is used:

$$count = \Phi(v0 - v1) + \Phi(v1 - v2) + \Phi(v2 - v3) + \Phi(v4 - v5) + \Phi(v5 - v6) + \Phi(v6 - v7)$$

Where v0 to v7 are adjacent pixel values along the edges in 8x8 block, as shown in Figure 10. If the count equals 6, the area is classified as very smooth. If the value of count falls between 1 to 5, the area is classified as relatively smooth. If the count is 0, then the area is classified as highly detailed.

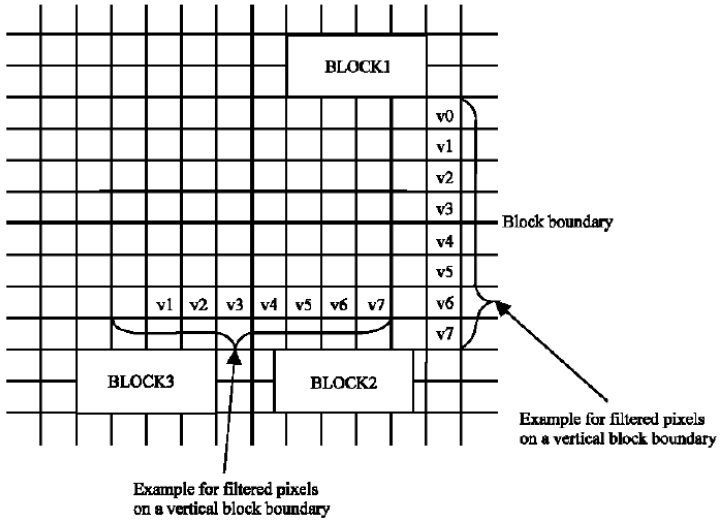


Figure 10. Adjacent blocks of pixels

For blocks of the image classified as highly detailed, a low amount of filtering is applied by using a factor a = 0.5. For relatively smooth blocks, a moderate amount of filtering is applied using a = 0.4. Finally, for blocks that are very smooth, strong filtering is applied using a = 0.3. Filtering is applied horizontally and vertically to the vertical and horizontal block boundaries respectively.

Post-processing is applied in two ways. The first algorithm applies the filtering along the vertical boundaries of the blocks, followed by the horizontal boundaries.

After count has been calculated, if it equals 6, the above equations are calculated for v3 and v4 using h(n) for a=0.3, v2 and v5 using a=0.4, and v1 and v6 using a=0.5. If count falls between 1 and 5, indicating the region to be relatively smooth, v3 and v4 are calculated using a=0.4, and v2 and v5 are calculated using a=0.5. If the count is 0, then the region is complex in nature, and only v3 and v4 are calculated using a=0.5.

The second algorithm follows all the steps of the first one, differing only when dealing with the pixels that are filtered both vertically and horizontally. For these pixels, the horizontal and vertical filtering is applied independent of the other, and the mean of the two values is chosen.

Liaw et al [12] proposes an approach in image deblocking and restoration based on Classified Vector Quantization. In CVQ, a codebook, consisting of codewords generated from a set of training vectors, is used for both the encoding and decoding process.

Before the application of CQV, a deblocking algorithm is applied to the image. The algorithm works in two steps: classification and deblocking. During the classification phase, the image is downsampled into 8x8 blocks. Each block is classified into smooth

and non-smooth based on the block's DCT coefficients. Following the classification, deblocking is applied to the block $B_{m,n}$ and its neighbors. If $B_{m,n}$ and its neighboring block $B_{m+1,n}$ are smooth, then a 1-D filter {1, 1, 2, 4, 2, 1, 1} is applied to one half of the block $B_{m,n}$ and the adjacent half of the neighboring block $B_{m+1,n}$. If the blocks are non-smooth, then a 1-D filter {1, 4, 1} is applied instead. This process is repeated for all the neighboring blocks of $B_{m,n}$.

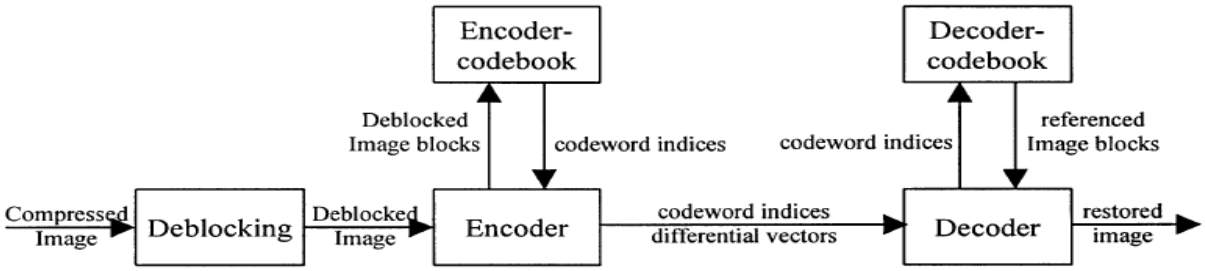


Figure 11. Image restoration using CQV

Traditionally, Vector Quantization uses a single codebook for both encoding and decoding, and this codebook is generated by the application of Generalized Lloyd algorithm. The authors modify this method to suit their task better, and device two separate codebooks, for encoding and decoding respectively, as shown in Figure 11. The two codebooks are not completely independent of either.

Once both codebooks have been derived, restoration of the image is applied via further classification of the image. The images are broken into 4x4 blocks, and the mean of the pixel intensities of the blocks is calculated and is used to classify the blocks into different sub-categories: uniform, texture and edge. If the block has been classified as non-uniform, further classification is required to determine whether it falls under an edge

class or a texture class. If the block belongs to the edge class, the edge orientation is determined as well.

Finally, using the information gathered so far, the process of restoration of the image, which consists of encoding and decoding, begins. For the encoding part, the mean of a block of the image is determined and using the information the block is classified and sub-classified into its respective category. If it belongs to a non-edge class, then its respective codeword is determined from the corresponding class of codewords in the codebook. If the block does belong to the edge-class, then the edge direction is calculated and the corresponding code word is retrieved from the code book based on class, type and direction. If a suitable codeword is found, it is subtracted from the input block to calculate the differential vector, and the codeword index and the differential vector are recorded. If no suitable codeword is found, then the block is not changed. This process is repeated till all the blocks have been encoded.

The decoding part of the image restoration works similarly. The mean values of the blocks are obtained from the indices that had been recorded. Using the mean value the class of the block is determined. If the block is of non-edge type, then its respective codeword is determined from the corresponding class of codewords in the codebook, determined uniquely from the class and block information. If the block does belong to the edge-class, then the edge orientation is calculated and the corresponding code word is retrieved from the code book based on class, type and orientation. This process is repeated till all the blocks have been decoded. By the combination of encoding and decoding the blocks of the compressed image, the image quality can be restored.

Chou et al [9] present a simple post-processing algorithm that attempts to estimate the quantization error present in the JPEG compressed images. They model each DC coefficient of all the blocks of the image as Gaussian random variables, and each AC coefficient as zero-mean Laplacian random variables. They apply a probabilistic framework to estimate the mean squared error of each DCT coefficient, and an estimate of quantization error for each n x n block, taking advantage of the fact that DCT being a unitary transform, the mean squared quantization error in DCT domain is equal to the mean squared error in spatial domain. They proceed to determine row and column vectors of discontinuities of the pixel intensities across the image, and using a threshold value determined from the mean squared errors previously, attempt to identify the discontinuities caused by quantization error. Upon identification of an anomaly in the pixel intensities, a new pixel intensity is calculated for the relevant pixels by using a proportionality constant determined from the threshold value and the mean of the image.

The algorithm is low on complexity and returns impressive results, upon application of the test image Lena, as shown in Figure 12. The compressed image is shown on the left and the filtered result is displayed on the right.

Figure 12. Lena, before and after

Triantafyllidis et al [10] proposes a very similar algorithm, which attempts to determine the quantization error by reconstructing the quantization coefficients. They go further, classifying the regions of the image into high and low detail, so an adaptive filtering can be applied.

Projection Onto Convex Sets

Another popular deblocking method uses the concept of projection onto convex sets. A convex set is defined as one in which a every point on a line segment drawn through two elements of the set, lies within the set, as shown in Figure 13(a). Figure 13(b) shows a non-convex set, that doesn't follow this condition.

POCS is an algorithm which ensures that given a set of closed convex sets, if we project a point onto one of them, then after a number of iterations it will converge into a point on the intersection of the closed convex sets.

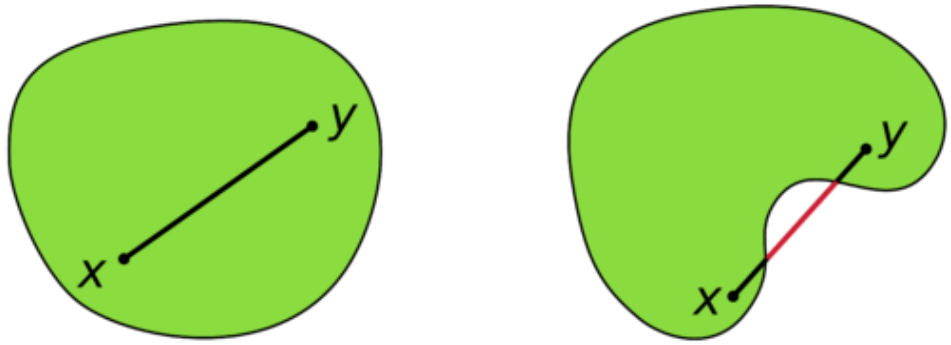


Figure 13. (a) Convex set, (b) Non-Convex set

Mathematically it means, if we define a point by X in Euclidean space and C1,C2,C3....,Cn as closed convex sets such that C1 U C2 U C3 U C4 U.....Cn is non

empty, then application of a projection operator which projects a point on to Ci cause X to converge on to a point X* E Ui=1 to m Ci.

Several deblocking methods using POCS can be found in literature. In all such methods, whether the pixel intensity will be adjusted to the global or regional threshold methods, is a binary decision. And different constraints are imposed on certain critical parameters that helps in making this decision. These constraints are described by closed convex sets. So if one projects the data pertaining to an image onto one of these constraint sets, POCS ensures that after a certain number of iterations it gets projected on to the intersection of all these constraint sets thereby ensuring optimum result. The trick then lies is designing the constraint sets.

Ju Jia Zou et al [4] constraint are used; two locally adaptive smoothness constraints, a locally adaptive quantization constraint and an intensity constraint. The deblocking effect is noticeable. Additionally, it preserves image details while reducing artifacts. Peak signal to noise ratio (PSNR) is used to quantify the quality of the reconstructed image. The results of this paper demonstrate the superiority of this method over standard JPEG-decoder method.

Gan et al [6],[18] propose a novel smoothness constraint set using the concept of Wiener filtering. They use a least mean-square formulation of the noise statistics obtained from the Wiener filter. This gives an advantage over adaptive Wiener filtering which leads to loss of information. Limiting output of low-pass filter may salvage some details but this method has been proven to be non-convergent unless the filter is ideal. This approach

therefore utilizes the effectiveness of Wiener filtering while eliminating the artifacts by use of POCS. They define two constraint sets as follows:

$$C_w = \{X: (X - \bar{X})^t M (X - \bar{X}) \leq E_w\}$$

$$C_r = \{X: (X - Y)^t R (X - Y) \leq E_r\}$$

Where X is the DCT of the image x, X' the prior mean of X and Y the quantized DCT coefficients, M and R are two matrices constructed using the mean square errors for noise σ_n and mean square errors of the DCT coefficients σ_x . E_a helps in setting a threshold ensuring our image lies within the space which can be projected onto the chosen convex sets.

This algorithm is computationally less taxing as it operates only in the transform domain, i.e., BDCT is not applied in each iteration of POCS.

In **[11], Weerasinghe et al.** use a new family of convex smoothness constraint sets. These sets preserves edge details and smoothes different regions of the image differently.

A fuzzy c-means clustering algorithm is used to segment the image into homogenous regions while leaving out the ambiguous pixels. These pixels are estimated during POCS iteration by imposing the constraints on them.

The smoothness constraints are then imposed on these regions thereby constructing the convex sets. Then the deblocking is achieved by application of the POCS algorithm. The advantage of this approach is that it does not oversmooth the edges as they form the boundaries of the homogenous regions which remain unaffected by the constraints. On the negative side, this approach is limited only to images that have homogenous regions.

The results obtained are displayed in Figure 14, 14(a) showing the original image, 14(b) the compressed version and finally 14(c) the result after processing the compressed image.

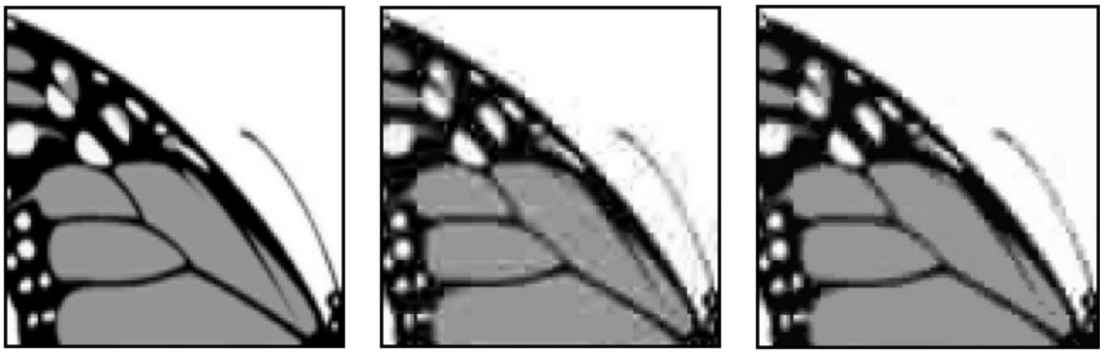


Figure 14. (a) Original, (b) After DCT transform, (c) Result after deblocking

Alter et al [3] propose a POCS method based on Total Variation minimization. The TV method yields good results in smoothing artifacts without smoothing the edges present in the images, which lead to loss of details in the image. TV method is most effective against Gibbs phenomenon, and in order to increase the efficacy of the method, the algorithm attempts to reconstruct the DCT coefficients, that have been lost due to quantization, which removes blocking artifacts present in the image.

To preserve the textures present inside the blocks, the Total Variance regularization needs to be stronger near the edges of the blocks. To maintain this a positive weight α is introduced, such that the value of α is greater in smoother areas and lower near the edges and textures. Using this weight, the DCT coefficients of the original image are modeled as a convex set U, and the deblocked image U^* is obtained by optimizing a derived convex function $J_\alpha(U^{*)}$. By the application of POCS, the image data lost during quantization is minimized in the reconstruction of the deblocked image.

Other popular methods

Apart from the methods described above, a large number of other algorithms are commonly used. A network designed and trained to create a distortion recovery image that gets added to the decompressed image reducing artifacts has also been proposed [32]. Several proposals have been made that make use of wavelet transforms [8, 28]. Hyuk and Kim [28] use the wavelet transform domain to decompose the image into subbands. They then reduce the blocky noise by a linear minimum mean square error filter, which fully exploits the characteristics of the signal and noise components in each subband. After the blocky noise is reduced, the granular noise can further be decreased by exploiting its nonstructuredness. Finally, methods using different variations of vector quantization have been made [26].

CHAPTER III

THE ALPHABLEND ALGORITHM

There are multiple methods that can be applied to improve the quality of an image compressed with the JPEG algorithm. An adaptive low pass filtering was chosen for this study, over multiple other processing techniques, for a number of reasons. Low pass filtering is a post processing technique that can be potentially applied to any image, irrespective of the compression level or other relevant meta-data. Methods like MAPS rely on prior knowledge of the image, which makes the method of limited effectiveness if the original image is absent. Methods like POCS achieve great results in restoration of images, but at a very high cost of computational complexity and processor overhead. Our approach doesn't need the original image data to be present, it can be applied to images of multiple formats of JPEG algorithm, and it can be also applied to other image formats as well, with little modification to the algorithm. The post-processing algorithms are simple to implement, and they do not require a large computational overhead.

Two different approaches have been made in an attempt to remove blocking artifacts from the compressed JPEG images. The first method uses the Peak Signal-to-Noise Ratio (PSNR) [13] values of the images with an adaptive limit to alter the pixel values of the compressed image. This approach, though it returned improved results, was deemed unsuitable as it required the original uncompressed image in order to determine the limit that it uses to calculate the values for pixel replacement. For assessment of image quality,

PSNR and Mean Structural Similarity Index Measure (MSSIM) [13] are used. PSNR is the ratio between the maximum possible power of a signal and the maximum power of an intermingled noise that corrupts the original signal. PSNR is most commonly used in image processing as a measure of quality for reconstructed images from lossy algorithms. The original signal is the original image data before compression, and the noise being the artifacts caused by the loss of data due to compression. For an mxn image I:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

MSE is the Mean Squared Error, an error estimate between the original image I and the final processed image K. From this equation, PSNR can be calculated as:

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

Where, MAX is the maximum pixel value of the original image I.

MSSIM, or Mean Structural Similarity Index Measure [13], is the method of measuring the amount of similarity between two given images. MSSIM, as a full reference metric, references the uncompressed data of the original image to determine the quality of a compressed image. For an image SSIM is calculated over a nxn window x of the original image and another window y of the same dimensions of the processed image.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Where, μ_x and μ_y are the average of the pixel values of x and y respectively, σ_x^2 and σ_y^2 are the variance of x and y respectively, and σ_{xy} is the covariance of x and y. c_1 and c_2 are two constants.

Using the equation above, MSSIM of an image can be derived by calculating the mean of the SSIM values of all the nxn windows of the image.

The second attempt, as stated previously, is an adaptive low-pass filter. The filter is designed to return optimal results over a large spectrum of compression. It does not require any prior knowledge of the image to be filtered. The algorithm is based on the concept of alpha blending used in image processing. Alpha blending is the method that combines an image with a background in a way such that the two images blend in partially or perfectly.

Alphablend filter

The Alphablend filter's goal is to reduce noise and artifacts in the compressed image, while ensuring that the details of the image are not lost during filtering. The algorithm uses an adaptive limit value, which is calculated from the compression ratio of the image being filtered, to determine whether or not a pixel needs to be altered. An edge map of the image is derived using the Prewitt filter, and the pixel values of the edge map are compared with the limit. If the pixel value of the edge map is greater than the limit, which it usually is for an edge within the image, then no filtering is applied to that pixel, in order to retain the edge details present in the image. If the pixel value of the image is lower than the limit, then the value is altered using the following Alpha blending equation as used in computer graphics.

$$M[i][j] = ROUND\big((1.0-\propto) * M[i][j] + \propto * LowPass[i][j]\big)$$

Where, LowPass is the image M after the application of a low pass filter, and the alpha value is calculated as:

$$\propto = CLAMP\left(0.0, 1.0, \frac{MSSIM(i,j,M)}{C}\right)$$

Where, the alpha value (∝), ranged between 0 and 1, is calculated from the MSSIM of the block containing the pixel (i, j) in image M. C is constant determined from the compression ratio of the image.

The Proposed Algorithm

The original algorithm for the alphablend filter can be summarized by the flowchart in Figure 15.

A low-pass filter (One_Pass_Filter()) is applied to the image and the result is stored separately. This step is followed by the creation of the edge map (Two_Pass_Filter()) using the Prewitt operator. The Prewitt operator is one of the most popular edge detection operator. It uses a vertical and horizontal kernel, which are convolved with an image to calculate approximate derivatives of the image pixels in both vertical and horizontal directions.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * A \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * A$$

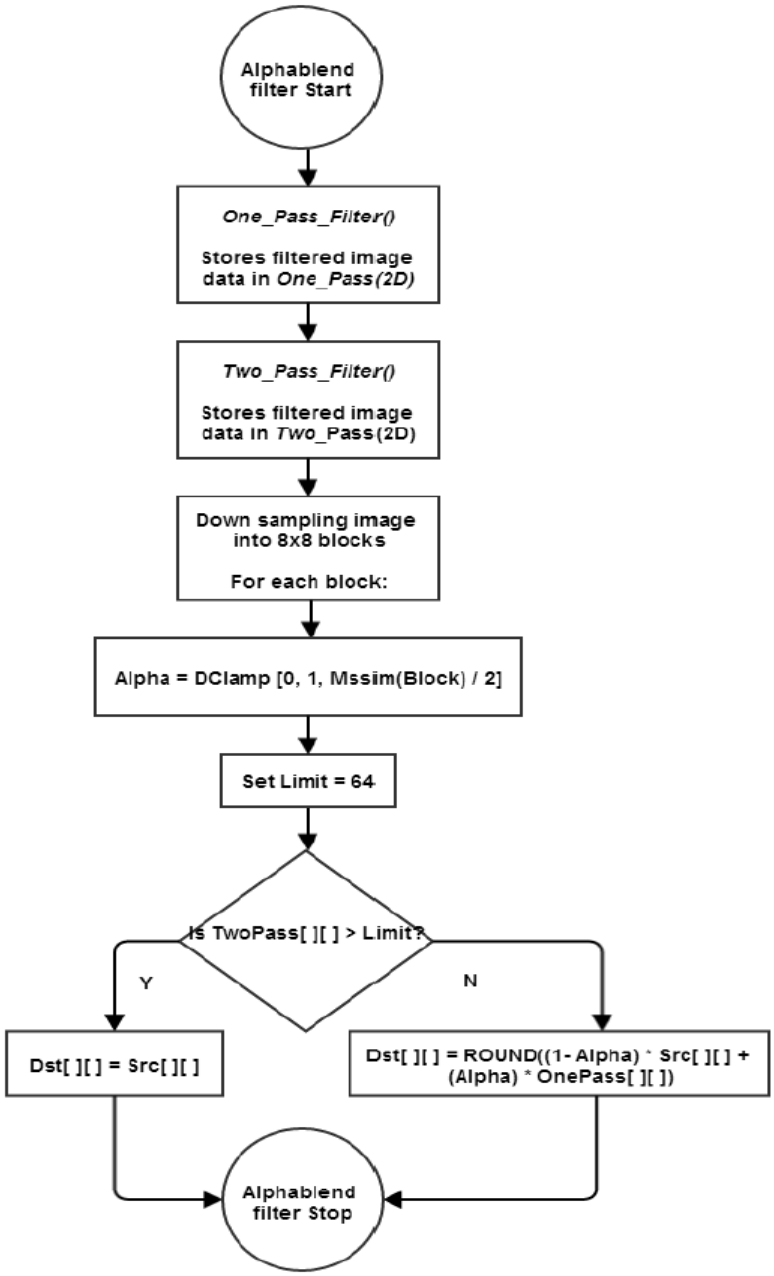


Figure 15. Flowchart of the original Alphablend algorithm

Where, G_x and G_y are the horizontal and vertical gradient approximations of every point on image A, and * is the convolution operation between A and the horizontal and vertical kernels of the Prewitt operator.

From the horizontal and vertical gradient approximations, for each point or pixel of A, the magnitude of the gradient can be derived as:

$$Two\_Pass[x][y] = \sqrt{G_x^2 + G_y^2}$$

Figure. 16 (a) Lena original, (b) Image gradient map generated by applying the Prewitt filter

After the edge map has been created, the original image is downsampled in to 12x12 blocks of pixels centered on each 8x8 pixel block. The alpha value is calculated from the MSSIM for each block using the formula:

$$\propto = CLAMP\left(0.0, 1.0, \frac{MSSIM(x, y, M)}{C}\right)$$

A constant value limit is set to 64, and the denominator C is set to 2. For each 12x12 block, the corresponding pixels on the edge map is compared with the limit. If the pixel is an image edge, then no processing is applied to the image. However, if the pixel value falls under the limit, it is altered as follows:

$$M[i][j] = ROUND\big((1.0-\propto) * M[i][j] + \propto * LowPass[i][j]\big)$$

This process is repeated iteratively by shifting the 12x12 block by one pixel horizontally at every iteration until it reaches the extreme edge of the image, then moving it back to its original position and shifting it one pixel vertically followed by the horizontal shift again. This process is repeated until every pixel in the image has been processed.

In Table 1, the results obtained by application of the algorithm on the test images are displayed. The *paper original* and *paper result* are an amalgamation of results obtained from various published works, the columns contain the PSNR values of the images before and after processing respectively. The test images are generated with their bits per pixel values are as close to that of the images used in the published results. The *percentage increase* columns show the increase in PSNR values, after processing, compared to the original PSNR values of the test images.

Table 1. Initial results obtained from using the original algorithm

paper	compression	paper	paper	%		compression	original	alphablend	%
no	bpp	original	result	increase		bpp	psnr	psnr	increase
4	0.158	26.61	27.93	4.96		0.153	28.77	29.02	0.87
9	0.15	26.44	27.5	4.01		0.153	28.77	29.02	0.87
25	0.15	26.64	27.77	4.24		0.153	28.77	29.02	0.87
27	0.169	28.45	29.15	2.46		0.17	29.34	29.6	0.89
28	0.16	29.48	30.52	3.53		0.17	29.34	29.6	0.89
4	0.188	28.46	29.64	4.15		0.187	29.82	30.11	0.97
18	0.189	30.08	31.08	3.32		0.187	29.82	30.11	0.97
27	0.187	28.98	29.63	2.24		0.187	29.82	30.11	0.97
27	0.209	29.62	30.21	1.99		0.204	30.28	30.58	0.99
30	0.2	28.9	29.44	1.87		0.204	30.28	30.58	0.99
15	0.217	29.47	30.27	2.71		0.22	30.63	30.96	1.08
2	0.24	30.63	30.58	-0.16		0.247	31.3	31.64	1.09
4	0.24	30.7	31.71	3.29		0.247	31.3	31.64	1.09
9	0.24	29.58	30.37	2.67		0.247	31.3	31.64	1.09
25	0.24	29.77	30.56	2.65		0.247	31.3	31.64	1.09
32	0.25	28.85	29.66	2.81		0.247	31.3	31.64	1.09
4	0.318	32.32	33.09	2.38		0.32	32.65	32.98	1.01
30	0.3	31.68	32.19	1.61		0.32	32.65	32.98	1.01
9	0.43	32.36	32.81	1.39		0.44	34.13	34.41	0.82
32	0.5	31.67	32.38	2.24		0.509	34.75	35.02	0.78
4	0.626	35.79	36.28	1.37		0.628	35.66	35.84	0.5
4	0.997	37.87	38.27	1.06		0.98	37.62	37.52	-0.27

The results above clearly show the improvement in the quality of the image, but upon comparison with the results from published works on JPEG image enhancement, the initial results fall short.

The initial algorithm used the image data from the original uncompressed image, along with the compressed image, in calculating the MSSIM, which it later uses to calculate the alpha value used for filtering. Since one of the goals of the algorithm was to be independent of original image, further changes were required. Moreover, the algorithm used a fixed limit in its calculation and the other goal for it was to be adaptive to the amount of compression present in the image.

The results obtained from the initial algorithm are plotted as a graph in Figure. 17. The higher the PSNR ratio, better the quality of the image after post-processing. The results were mostly inferior to the ones obtained from other published results.

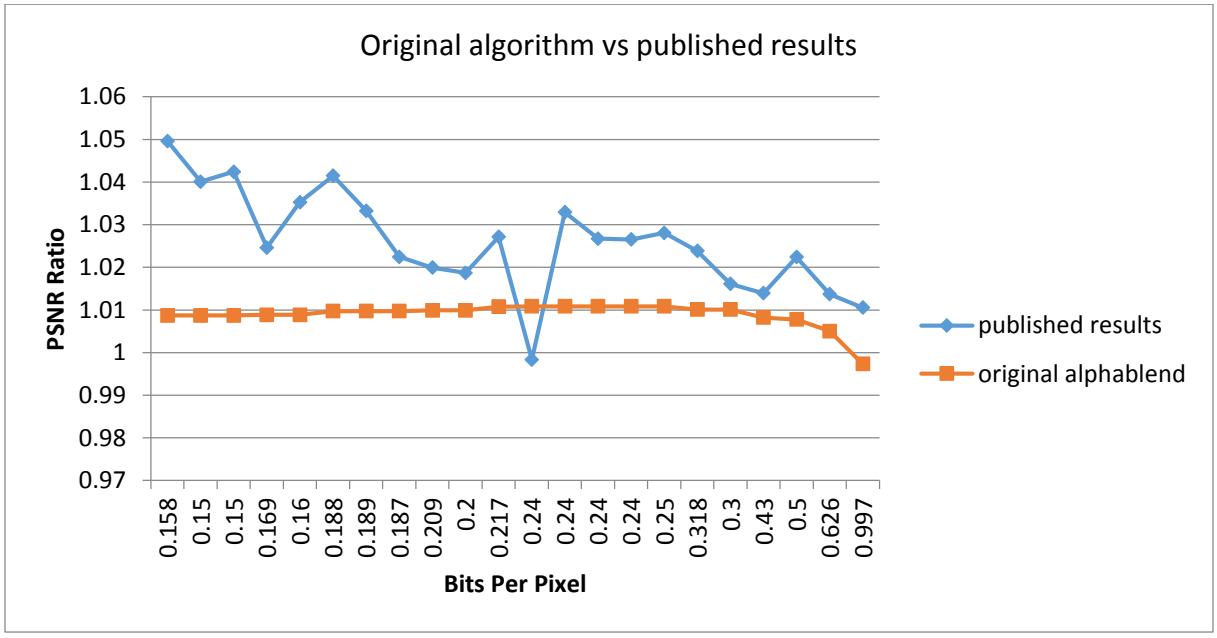


Figure 17. Initial results compared with the results obtained from various published works

The first change made to the algorithm was in the calculation of MSSIM. Initially, MSSIM was calculated for the compressed image with respect to the original image. The code was modified to now calculate the MSSIM value between the compressed image and the low pass filtered version of the image.

As it can be seen in Table 2, the change in the algorithm returned improved results in images with high compression, increasing from 0.87% to 2.92%. On the other hand, in case of images of low compression, the returned results displays a massive degradation of image quality, up to a decrease of almost 10% PSNR value. This showed that the algorithm worked better on some compression levels than others, and a second change was required, as previously stated, to make the algorithm adaptive to a broad range of compression level present in the images.

Table 2. Results after changing the MSSIM code to be independent of the original image

paper no	compression bpp	paper original	paper result	% increase		compression bpp	original psnr	alphablend psnr	% increase
4	0.158	26.61	27.93	4.96		0.153	28.77	29.61	2.92
9	0.15	26.44	27.5	4.01		0.153	28.77	29.61	2.92
25	0.15	26.64	27.77	4.24		0.153	28.77	29.61	2.92
27	0.169	28.45	29.15	2.46		0.17	29.34	30.1	2.59
28	0.16	29.48	30.52	3.53		0.17	29.34	30.1	2.59
4	0.188	28.46	29.64	4.15		0.187	29.82	30.48	2.21
18	0.189	30.08	31.08	3.32		0.187	29.82	30.48	2.21
27	0.187	28.98	29.63	2.24		0.187	29.82	30.48	2.21
27	0.209	29.62	30.21	1.99		0.204	30.28	30.82	1.78
30	0.2	28.9	29.44	1.87		0.204	30.28	30.82	1.78
15	0.217	29.47	30.27	2.71		0.22	30.63	31.09	1.5
2	0.24	30.63	30.58	-0.16		0.247	31.3	31.54	0.77
4	0.24	30.7	31.71	3.29		0.247	31.3	31.54	0.77
9	0.24	29.58	30.37	2.67		0.247	31.3	31.54	0.77
25	0.24	29.77	30.56	2.65		0.247	31.3	31.54	0.77
32	0.25	28.85	29.66	2.81		0.247	31.3	31.54	0.77
4	0.318	32.32	33.09	2.38		0.32	32.65	32.33	-0.98
30	0.3	31.68	32.19	1.61		0.32	32.65	32.33	-0.98
9	0.43	32.36	32.81	1.39		0.44	34.13	32.93	-3.52
32	0.5	31.67	32.38	2.24		0.509	34.75	33.12	-4.69
4	0.626	35.79	36.28	1.37		0.628	35.66	33.22	-6.84
4	0.997	37.87	38.27	1.06		0.98	37.62	33.57	-10.77

The compression in an image is represented by bits per pixel (bpp). The bpp value of an image decreases as the amount of compression applied to the image is increased. The bits per pixel value of a image is calculated as:

$$bits = image_{size} * 8$$

$$pixels = image_{height} * image_{width}$$

$$image_{bpp} = bits/pixels$$

Since a grayscale image has 8 bits per pixel, the number of bits present in the image can be calculated by multiplying the image size in bytes by 8. The total number of pixels present in the image can be easily determined by multiplying the width and height of the image in number of pixels. This, compression ratio of an image can be determined from the number of bits present and the total number of pixels.

Once the compression ratio of the image is determined, a set of test images (Lena and Peppers) are created. The test images have an even range of compression, ranging from 5% to 95%. Then, upon each image from the set of test images, a brute force technique is applied over the value of the constant C in the function for calculating the alpha value that had been set to 2.

$$\propto = CLAMP\left(0.0, 1.0, \frac{MSSIM(x, y, M)}{C}\right)$$

The value of C is ranged between 0 and 10, incremented by 0.1. The PSNR values of all the results for each value of limit, from1 to 255, is recorded and the best result is selected for each image.

The purpose of this is to determine which value of the constant C returns the best results for that level of compression, so a function for the constant can be created over the bits per pixel values of the image. The data received from the brute force method is applied to a curve fitting tool in Matlab and the following function is determined.

$$C = 0.3727 * bpp^3 - 2.1423 * bpp^2 + 5.2906 * bpp + 0.055$$

Table 3 shows the results obtained by the brute force iteration, the optimum value of C' that returns the best PSNR values for each version of Lena. A second value for the constant, shown in the column *C,* is obtained by using the above equation to determine the optimal value of C, with respect to the compression present in the image.

A second brute force application is set up, with respect to the new adaptive C value. This time, the value of the limit, which had been fixed at 64, is ranged from 1 to 255, and is applied to every image in the test set, the denominator C in the function being dynamically calculated from the compression ratio of the image. The PSNR values of all the results for each value of limit, from1 to 255, is recorded and the best result is selected for each image. The best results for every level of compression is selected and the corresponding limit value is fit into the curve fitting tool, with respect to the compression level, to generate the following function for the limit:

$$limit = 34.12 * bpp^{-0.8432} + 42.11$$

Table 3. Results after the brute force attempt to calculate the optimal value of the constant C

Lena versions	BPP	Original Psnr	Alphablend Psnr	C'		Alphablend Psnr	C
5	0.117	27.21	27.61	0.8		27.61	0.6
10	0.204	30.28	30.71	1.1		30.71	1.1
15	0.273	31.81	32.22	1.5		32.21	1.3
20	0.334	32.82	33.21	1.7		33.21	1.6
25	0.39	33.56	33.91	1.8		33.91	1.8
30	0.441	34.13	34.45	1.9		34.43	2
35	0.494	34.61	34.91	1.9		34.9	2.2
40	0.536	34.98	35.26	2.6		35.25	2.3
45	0.585	35.35	35.61	3		35.61	2.5
50	0.628	35.66	35.91	3.1		35.91	2.6
55	0.671	35.96	36.21	3.2		36.2	2.8
60	0.726	36.31	36.55	3.2		36.55	2.9
65	0.798	36.71	36.93	3.6		36.93	3.1
70	0.887	37.17	37.38	3.6		37.37	3.3
75	0.988	37.68	37.87	3.7		37.87	3.6
80	1.15	38.38	38.55	3.8		38.55	3.9
85	1.371	39.29	39.44	5.5		39.42	4.2
90	1.789	40.69	40.81	5.7		40.8	4.8
95	2.742	43.69	43.79	7.5		43.76	6.1

Table 4, as the previous table, shows the results obtained by the second brute force iteration. The optimum values of the limit that return the best PSNR values, for each version of Lena, are shown in the column $Limit'$. A second value of the limit is obtained by using the above equation to determine the values of limit dynamically, instead of a brute force approach, with respect to the compression present in the image. These values are shown in the column labeled $Limit$.

Furthermore, a third brute force application is set up, in a similar manner, to re-calculate the denominator C of the alpha function, now taking the adaptive limit into consideration. C is once again ranged between 0 to 10, incremented by 0.1, and applied to every image

Table 4. Results after the second brute force attempt to calculate the optimal limit with respect to the optimal value of the constant C

Lena versions	BPP	Original Psnr	Alphablend Psnr	C	Limit'		Alphablend Psnr	C	Limit
5	0.117	27.21	28.18	0.6	255		28.18	0.6	250.8
10	0.204	30.28	30.96	1.1	196		30.95	1.1	172.2
15	0.273	31.81	32.37	1.3	179		32.35	1.3	144
20	0.334	32.82	33.29	1.6	132		33.28	1.6	128.1
25	0.39	33.56	33.98	1.8	145		33.98	1.8	117.6
30	0.441	34.13	34.48	2	156		34.47	2	110.1
35	0.494	34.61	34.92	2.2	119		34.91	2.2	104
40	0.536	34.98	35.27	2.3	94		35.26	2.3	99.8
45	0.585	35.35	35.62	2.5	90		35.62	2.5	95.7
50	0.628	35.66	35.92	2.6	82		35.92	2.6	92.6
55	0.671	35.96	36.21	2.8	85		36.21	2.8	89.9
60	0.726	36.31	36.55	2.9	75		36.55	2.9	86.8
65	0.798	36.71	36.93	3.1	86		36.93	3.1	83.4
70	0.887	37.17	37.38	3.3	73		37.38	3.3	79.9
75	0.988	37.68	37.87	3.6	72		37.87	3.6	76.6
80	1.15	38.38	38.55	3.9	69		38.55	3.9	72.4
85	1.371	39.29	39.42	4.2	51		39.41	4.2	68.3
90	1.789	40.69	40.81	4.8	40		40.8	4.8	63
95	2.742	43.69	43.81	6.1	33		43.78	6.1	56.7

in the test set. The resulting data is fitted into the curve fitting tool in Matlab and the following function is generated:

$$C = 5.243 * e\left(-\left(\frac{(bpp - 2.414)}{1.224}\right)^2\right) + 3.374 * e\left(-\left(\frac{(bpp - 1.057)}{0.8201}\right)^2\right)$$

Final Results

The following table lists the final results of the algorithm:

Table 5. Final results of the alphablend algorithm, with dynamically calculated limit & constant C

paper no	compression bpp	paper original	paper result	% increase		compression bpp	original psnr	alphablend psnr	% increase
4	0.158	26.61	27.93	4.96		0.153	28.77	29.64	3.02
9	0.15	26.44	27.5	4.01		0.153	28.77	29.64	3.02
25	0.15	26.64	27.77	4.24		0.153	28.77	29.64	3.02
27	0.169	28.45	29.15	2.46		0.17	29.34	30.17	2.83
28	0.16	29.48	30.52	3.53		0.17	29.34	30.17	2.83
4	0.188	28.46	29.64	4.15		0.187	29.82	30.61	2.65
18	0.189	30.08	31.08	3.32		0.187	29.82	30.61	2.65
27	0.187	28.98	29.63	2.24		0.187	29.82	30.61	2.65
27	0.209	29.62	30.21	1.99		0.204	30.28	31	2.38
30	0.2	28.9	29.44	1.87		0.204	30.28	31	2.38
15	0.217	29.47	30.27	2.71		0.22	30.63	31.34	2.32
2	0.24	30.63	30.58	-0.16		0.247	31.3	31.95	2.08
4	0.24	30.7	31.71	3.29		0.247	31.3	31.95	2.08
9	0.24	29.58	30.37	2.67		0.247	31.3	31.95	2.08
25	0.24	29.77	30.56	2.65		0.247	31.3	31.95	2.08
32	0.25	28.85	29.66	2.81		0.247	31.3	31.95	2.08
4	0.318	32.32	33.09	2.38		0.32	32.65	33.17	1.59
30	0.3	31.68	32.19	1.61		0.32	32.65	33.17	1.59
9	0.43	32.36	32.81	1.39		0.44	34.13	34.49	1.05
32	0.5	31.67	32.38	2.24		0.509	34.75	35.07	0.92
4	0.626	35.79	36.28	1.37		0.628	35.66	35.94	0.79
4	0.997	37.87	38.27	1.06		0.98	37.62	37.8	0.48

The results obtained from the final algorithm are plotted as a graph in Figure. 18. The higher the PSNR ratio, better the quality of the image after post-processing. The results show significant improvement over the ones obtained from the initial algorithm, and are better than multiple published results, for images with the same bits-per-pixel value.

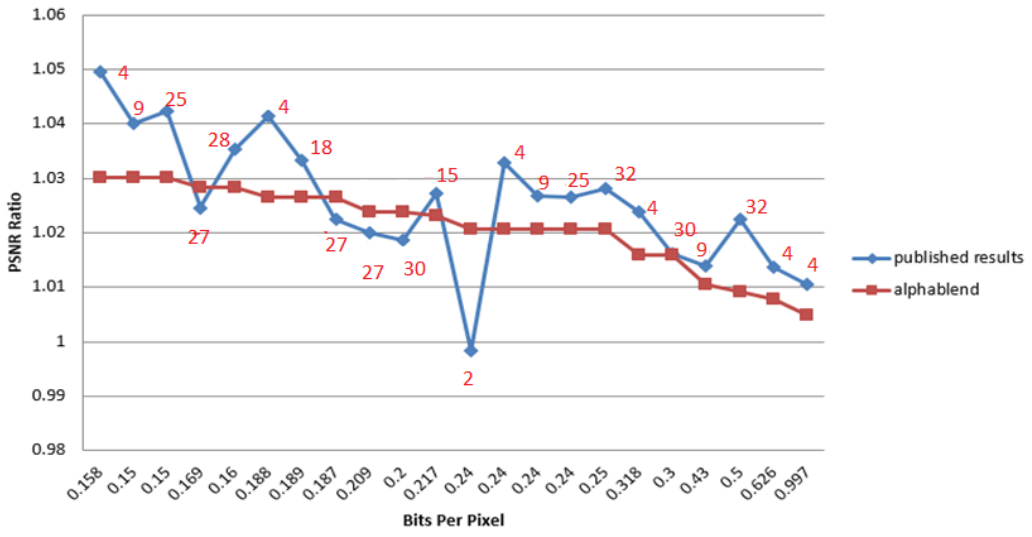


Figure 18. Final results

Figure 19 provides example images for low compression (0.323 bpp), medium compression (0.204 bpp), and high compression (0.153 bpp) images. The top row are the original image sections (Lena), while the bottom row are the results for each image after application of the Alphablend filter.

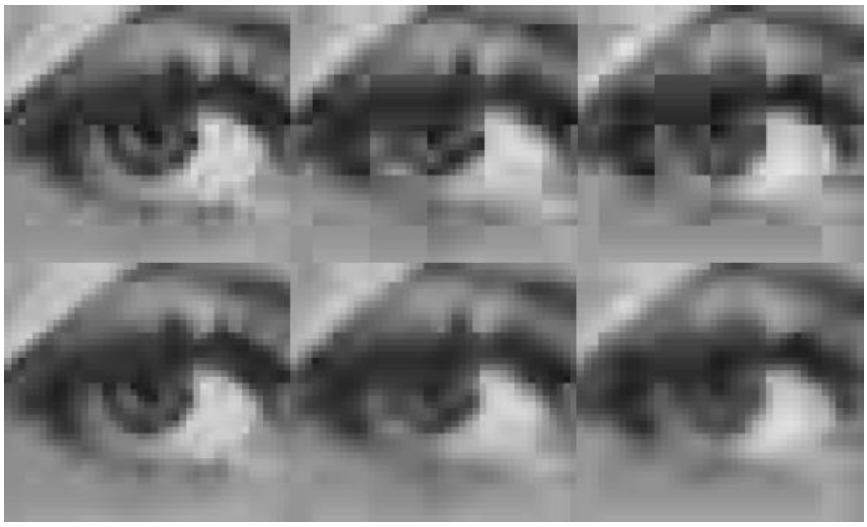


Figure 19. Zoomed-in sample from Lena, before and after processing (a) Low compression (0.323 bpp), (b) Medium compression (0.204 bpp), (c) High Compression (0.153 bpp)

Figure 20. (a) Lena with low compression (0.323 bpp) (b) Filtered image

Figure 21 (a) Lena with medium compression (0.204 bpp) (b) Filtered image

Figure 22 (a) Lena with high compression (0.153 bpp) (b) Filtered image

CHAPTER IV

CONCLUSION

Due to high demand for faster transmission of data, image processing plays a very important role in applications ranging from video communication, data from GPS satellites, etc. to data storage devices. Image compression techniques fall between two categories –lossless and lossy. The former category suffers no loss of information after the application of the compression algorithm, while algorithms falling in the latter category suffer from a significant loss of information from the original image after compression. One of the most popular lossy compression algorithms is JPEG.

An extremely popular and widely used algorithm, JPEG can produce compression ratios up to 10:1 with very little deterioration of image quality. JPEG uses a block-based discrete cosine transform (BDCT) coding scheme, dividing the image into 8x8 blocks and transforming each block using DCT, followed by a quantization of the transform coefficients of each block. Each block is quantized separately, so the DCT coefficients of each block are quantized differently. This leads to the creation of blocking artifacts from the discontinuities in color and luminosity between adjacent blocks. Also, due to the loss of the high-frequency components during the quantization, ringing artifacts are created along the edges present in the original image.

The aim of this research was to develop an algorithm that will produce a considerable improvement in the quality of JPEG images, irrespective of the level of compression present in the image. With that in mind, an adaptive low-pass filtering approach was chosen. Low pass filtering can potentially be applied to any image, irrespective of its compression level. While methods like POCS based algorithms are highly successful in restoring the image quality, they come with a high computational complexity and processor overhead, unlike this method.

The results show a clear improvement in the overall quality of the images, and the image samples after post-processing clearly demonstrate that the blocking and ringing artifacts are smoothed out to a certain degree. The algorithm returns better results than those obtained from multiple published works, such as Kieu, V.T. and Nguyen [2], Singh et al [27] and Li et al [30].

The advantage of the Alphablend algorithm is that it can be applied to any image with any level of compression. It has very low computational complexity, and the algorithm is easy to implement. Although some of the published works have returned better results than the Alphablend algorithm, few of these works have been applied over images with such a broad spectrum of compression, whereas the Alphablend algorithm returns improved results consistently. At present, the algorithm can be applied to grayscale images. With minor changes to the algorithm, it can be tailored to work on RGB color images as well. As RGB images have 24-bits (8-bits for each band) instead of 8-bits in grayscale images, the bits per pixels for the images will have to be calculated differently. Also, the additional color pixels will have to be taken into account while calculating the MSSIM for the images. Furthermore, there is scope of greatly increasing the

effectiveness of the algorithm. The precision of the algorithm can be further improved by applying it over a broader spectrum of test images, the data from which can be used to generate much more accurate values for the constants used in the Alphablend functions.

REFERENCES

1. Y. Tsaig, M. Elad, G.H. Golub, and P. Milanfar, "**Optimal framework for low bit-rate block coders**", in Proc. International Conference on Image Processing (ICIP) (2), 2003, pp.219-222.

2. Kieu, V.T. and Nguyen, D.T. "**Surface fitting approach for reducing blocking artifacts in low bitrate DCT decoded images**," in Proc. International Conference on Image Processing (ICIP) (3), 2001, pp150-153.

3. Francois Alter, Sylvain Durand, and Jacques Froment, "**Deblocking DCT-based compressed images with weighted total variation**", in Proc. IEEE International Conference on Acoustics, Speech , and Signal Processing (ICASSP) (3) 2004, pp 221-224.

4. Zou, Ju Jia, "**Reducing artifacts in BDCT-coded images by adaptive pixel-adjustment**," in Proc. 17th International conference on Pattern Recognition (ICPR) (1), 2004, pp 508-511

5. Yao Nie, Hao-Song Kong, Vetro, A., Huifang Sun, and Barner, K.E., "**Fast adaptive fuzzy post-filtering for coding artifacts removal in interlaced video**," in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (2), 2005, pp 993-996.

6. Xiangchao Gan, Alan Wee-Chung Liew, and Hong Yan, "**A smoothness constraint set based on local statistics of BDCT coeeficients for image postprocessing**," Image and Vision Computing, (23) 2005, pp 721-737.

7. I. Abboud, "**Deblocking in BDCT image and video coding using a simple and effective method**," Information technology Journal 5 (3), 2006, pp 422-426. ISSN 1812-5638.

8. Tai-Chiu Hsung and Daniel Pak-Kong Lun, "**Application of singularity detection for the deblocking of JPEG decoded images**," IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 45, No. 5, 1998, pp 640-644.

9. Jim Chou, Matthew Crouse, and Kannan Ramchandran, "**A simple algorithm for removing blocking artifacts in block-transform coded images**," IEEE Signal Processing Letters, Vol. 5, No. 2, 1998 pp 33-35.

10. G.A. Triantafyllidis, M. Varnuska, D. Sampson, D. Tzovaras, and M.G. Strintzis, "**An effiecient algorithm for the enhancement of JPEG-coded images**," Computers & Graphics 27, 2003, pp 529-534.

11. Chaminda Weerasinghe, Alan Wee-Chung Liew, and Hong Yan, "**Artifact reduction in compressed images based on region homogeneity constraints using the projection onto convex sets algorithm**," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 10, 2002 pp 891-897.

12. Yi-Ching Liaw, Winston Lo, and Jim Z.C. Lai, "**Image restoration of compressed images using classified vector quantization**," Pattern Recognition (35), 2002, pp 329-340.

13. Sukhwinder Singh, Vinod Kumar, and H.K Verma, "**Reduction of blocking artifacts in JPEG compressed images**," Digital Signal Processing 17, 2007, pp 225-243.

14. Bahadir K. Gunturk, Yucel Altunbasak, and Russell M. Mersereau, "**Multiframe blocking-artifact reduction for transform-coded video,**" IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 4, 2002, pp 276-282.

15. Shizhong Liu and Alan C. Bovik, "**Efficient DCT-domain blind measurement and reduction of blocking artifacts**," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 12, 2002, pp 1139-1149.

16. Camelia Popa, Aural Vlaicu, Mihaela Gordon, and Bogdan Orza, "**Fuzzy contrast enhancement for images in the compressed domain**," in Proc. International Multiconference on Computer Science and Information Technology, 2007, ISSN 1896-7094, pp 161-170.

17. Jinshan Tang, Jeonghoon Kim, and Eli Peli, "**Image enhancement in the JPEG domain for people with vision impairment**," IEEE Transactions on Biomedical Engineering, Vol. 51, No. 11 2004, pp 2013-2023.

18. Xiangchao Gan, Alan Wee-Chung Liew, and Hong Yan, "**Blocking artifact reduction in compressed images based on edge-adaptive quadrangle meshes**," Journal of Visual Communication and Image Representation, 14, 2003, pp 492–507.

19. Zhou Wang, Bovik, A.C., and Evan, B.L., "**Blind measurement of blocking artifacts in images**," in Proc. International Conference on Image Processing, Vol. 3, 2000, pp 981-984.

20. Kirenko, I.O., Muijs, R., and Shao, L., "**Coding artifact reduction using non-reference block grid visibility measure**," in Proc. IEEE International Conference on Multimedia and Expo, 2006, ISBN 1-4244-0366-7, pp 469-472.

21. Ling Shao and Kirenko, I., "**Coding artifact reduction based on local entropy analysis**," IEEE Transactions on Consumer Electronics, Vol. 53, Issue 2, ISSN**:** 0098-3063, 2007, pp 691-696.

22. 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, Vol. 7, No. 2, 1998, pp 229-234.

23. R. Samaduni, A. Sundararajan, and A. Said, "**Deringing and deblocking DCT compression artifacts with efficient shifted transforms**," Proc. IEEE Int. Conf. on Image Processing, (ICIP '04), Singapore, vol. 3, pp.1799-1802, Oct. 2004.

24. Amjed S. Al-Fahoum and Ali M. Reza, "**Combined edge crispiness and statistical differencing for deblocking JPEG compressed images**," IEEE Transactions on Image Processing, Vol. 10, No. 9, 2001, pp 1288-1298.

25. Kiryung Lee, Dong Sik Kim, and Taejeong Kim, "**Regression-Based Prediction for Blocking Artifact Reduction in JPEG-Compressed Images**," IEEE Transactions on Image Processing, Vol. 14, No. 1, 2005, pp 36-48.

26. Jim Z.C. Lai, Yi-Ching Liaw, and Winston Lo, "**Artifact reduction of JPEG coded images using mean-removed classified vector quantization,**" Signal Processing, 82, 2002, pp 1375-1388.

27. Jagroop Singh, Sukhwinder Singh, Dilbag Singh, and Moin Uddin, "**A signal adaptive filter for blocking effect reduction of JPEG compressed images**," International Journal of Electronic and Communications 65, 2011, pp 827-839.

28. Hyuk Choi and Taejeong Kim, "**Blocking-Artifact Reduction in Block-Coded Images Using Wavelet-Based Subband Decomposition**," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 5, 2000, pp 801-805.

29. George A. Triantafyllidis, Dimitrios Tzovaras, and Michael Gerassimos Strintzis,"**Blocking Artifact Detection and Reduction in Compressed Data**," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 10, 2002, pp 877-890.

30. Zhen Li and Edward J. Delp, "**Block Artifact Reduction Using a Transform-Domain Markov Random Field Model**," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 15, No. 12, 2005, pp 1583-1593.

31. Du˜ng T. Võ, Truong Q. Nguyen, Sehoon Yea, and Anthony Vetro, "**Adaptive Fuzzy Filtering for Artifact Reduction in Compressed Images and Videos**," IEEE Transactions on Image Processing, Vol. 18, No. 6, 2009, pp 1166-1178.

32. S. Chen, Z. He, and P.M. Grant, "**Artificial neural network visual model for image quality enhancement**," Neurocomputing 30, pp 339-346, 2000.

33. DEFLATE Compressed data format specification, http://www.ietf.org/rfc/rfc1951.txt

34. An explanation of Dflate algorithm, http://www.zlib.net/feldspar.html

35. LZ&& Compression algorithm, http://msdn.microsoft.com/en-us/library/ee916854.aspx

36. JPEG, http://www.jpeg.org/jpeg/index.html