REMOVING IMAGE COMPRESSION ARTIFACTS

Muhammad Asjad (4830568)
Department of Computing and Information Technology
University of Wollongong

ABSTRACT

DCT compression methods can introduce visible artifacts at block boundaries of images. In this project we take a look at an algorithm proposed by Luo and Ward for the removal of edges of the blocks in regions that are affected by blockiness effects. The algorithm uses an adaptive approach where it considers the local information of the image before proceeding with the modification of pixel values based on weighted average of neighbouring blocks. Experiments show that the algorithm can significantly reduce the visible blockiness artifacts from the given images.

1. INTRODUCTION

Data compression is the process of removing the redundant data present in a given image while trying to preserve as much information present in the original image as possible[3]. The common types of data redundancies can be characterized into three categories. 1) Coding Redundancy 2) Spatial Redundancy 3) Irrelevant Information. Several methods are available compress the given image for transmission and storage. However these methods come with their own set of challenges. For example block-based DCT compression methods can introduce certain types of visible artifacts at block boundaries[4]. This digression is because of coarse quantization of coefficients. If we do not take into account the existing correlations among adjacent blocks then these types of "blocking artifact" can occur. Several filtering techniques exist to reduce the blockiness by reducing the high frequency components present near the block boundaries. In this project we will take a look at the technique proposed by Luo and Ward for removing the blocking effects in smooth regions of the image.

2. ALGORITHM

The algorithm proposed by Luo and Ward is based on the idea that some of the DCT coefficients of a step function(representing a sharp edge) have nonzero values[4]. First step in this approach is to detect the strong texture and edge regions based on frequency co-efficients. We iterate through the image and extract two neighboring 8x8 blocks in each step. If they have similar frequency properties and the boundary 8x8 pixel contains no high frequencies then portion of image is considered to be smooth. DCT filtering has shown to be effective in this case as it ensures pixel value continuity. Our core algorithm consists of following steps:

i. Extracting adjacent blocks

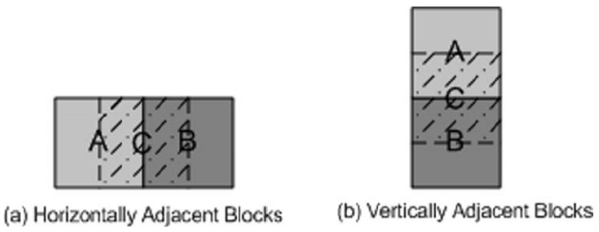

Figure 1. Blockiness Diagram [4]

We pick two blocks A and B from the given image as show in figure 1. Block C will contain boundary pixels as it will contains right and left half of Block A and Block B respectively. In the first phase we pick horizontally adjacent blocks and iterate column wise through the image, while in the second phase we pick vertically adjacent blocks while iterating row wise through the image.

ii. Calculating DCT transform

The Next step will be to calculate the DCT matrix of all the three blocks. DCT has the ability to concentrate information and is used in image compression applications. The definition of the two-dimensional DCT for an input image A and output image B is:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & 1 \le p \le M-1 \end{cases}$$

and

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & 1 \le q \le N-1 \end{cases}$$

iii. Threshold Comparisons:

To adapt the filtering to the local information content of the image we apply the following conditions. They will ensure that modifications are done to relatively smooth regions.

a. $|F_a(0,0) - F_b(0,0)| < T_1$

b. $|F_a(0,1) - F_b(0,1)| < T_2$.

c. $|F_c(3,3)| < T_3$.

where values for T_1, T_2, T_3 are predefined and are found through experimentation[4]. If the above three conditions are met we can proceed with blockiness reduction in DCT domain as explained in next step 4.

iv. Computing Modified Block C

It is observed that Block C will have non-zero values at row 0 and column= 0,1,3,5,7 of the image. Hence we will modify five coefficients values of block C, but we have to modify them in a way that original information (e.g sharp edges) in the image should not be lost during the process. Modified block C can be computed using weighted average of blocks A, B, and C which makes filtering more adaptive to image contents [4]. Following equations are used for the computing the modified block C which takes into account information present in neighboring blocks B and C[4].

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

where

$$v = 0,1$$

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

where

$$v = 3,5,7$$

$$F_{Mc}(u,v) = F_c(u,v) \quad \text{for all} \quad v \neq 0,1,3,5,7$$

v. Replacing input image with inverse DCT of Modified C block

In this step we compute the inverse DCT of modified block C and replace the original block C pixel values with the newly computed values. The new modified pixel values of block C can be represented by the following equation[4]:

$$C_{Mc}(k,l) = c(k,l) + \xi(l)$$

where

$$\begin{aligned}\xi(l) &= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \left(C_u C_v \left(F_{Mc}(u,v) - F_c(u,v)\right)\right. \\ &\qquad \left. \times W_{DCT}(u,k) W_{DCT}(v,l)\right) \\ &= C_0 C_0 \left[F_{Mc}(0,0) - F_c(0,0)\right] \\ &\quad + \sum_{i=0}^{3} C_0 C_v \left[F_{Mc}(0,v) - F_c(0,v)\right] \\ &\quad \times W_{DCT}(v,l)\end{aligned}$$

where $v = 2i + 1$.

Steps i to v will remove discontinuity present at boundary pixels of horizontally adjacent blocks. These step are repeated in Phase II to remove discontinuity from vertically adjacent blocks[4].

3. MATLAB IMPLEMENTATION

In this section we discuss some of the major steps in the matlab implementation of the algorithm mentioned in the pervious section.

Step 1: Extracting 8x8 Blocks A, B and C

We start off by extracting three 8x8 blocks where in the first phase, blocks A and B are horizontally aligned and Block C overlaps on both halves of Block A and B as shown in figure 1. In the 2nd Phase these blocks are vertically aligned and image pixels are iterated in a row wise manner.

```
f_blockA = f(row_pixel:row_pixel + 7,column_pixel:
column_pixel + 7);

f_blockC = f(row_pixel:row_pixel+7,column_pixel + 4 :
column_pixel + 11);

f_blockB = f(row_pixel :row_pixel +7,column_pixel + 8:
column_pixel + 15);
```

Step 2: Computing DCT:

We use the dct2 command to compute the corresponding DCT matrixes for Block A, B and C. The command uses fast fourier transform algorithm and is computationally efficient.

```
DCT_blockA = dct2(f_blockA);
DCT_blockB = dct2(f_blockB);
DCT_blockC = dct2(f_blockC);
```

Step 3: Adaptive Filtering

As discussed in the algorithms section, we need our filter to adapt to the local information of the image so as not modify unintended pixels of the image. We apply three conditions in this step:

Threshold Comparisons:

Condition 1:

This condition will ensure that Block A has a similar horizontal frequency property as block B and will be met if the first row of the DCT matrix off block A and that of block B have close values[4].

```
DCT_BLOCKA(0,0) - DCT_BLOCKB(0,0) < Threshold_1
```

Condition 2:
The boundary between block A and block B belongs to a relatively smooth region.

```
DCT_BLOCKA(0,1) - DCT_BLOCKB(0,1) < Threshold_2
```

Condition 3:

```
DCT_BLOCKC(3,3) < Threshold_3
```

where values of Thresholds in the above conditions are are:

```
Threshold_1 = 350
Threshold_2 = 120
Threshold_3 = 60
```

The above values are predetermined through experimentation by the authors[4].

If the three constraints above are satisfied, we perform the blockiness reduction in the DCT domain by modifying the five relevant coefficients of the DCT_BLOCKC matrix. The new values are calculated by weighted average of Blocks A,B,C (using the equations mentioned in the pervious algorithm section). This makes the filter more adaptive to image contents. We implement the equations using the following code:

```
Modified_BLOCKC(0,Y) = 0.6 * DCT_BLOCKC
+ 0.2* (DCT_BLOCKA + DCT_BLOCKB)
where Y = 0,1
```

```
Modified_BLOCKC(0,Y) = 0.5 * DCT_BLOCKC
+ 0.25* (DCT_BLOCKA + DCT_BLOCKB)
where Y = 3,5,7
```

```
Modified_BLOCKC(X,Y) = BLOCKC(X,Y)

where for all values where Y is not equal to
0,1,3,5,7
```

Step 4: Replacing Block C with inverse DCT of Modified Block C

The last step is to replace the Block C co-ordinate pixel values in the original image by the values of the matrix that holds the inverse DCT of modified block C. In matlab we first compute the inverse DCT of the modified block C using the idct2 command and then replace the pixel values directly using the following lines of code:

```
inv_DCT_blockC = idct2(DCT_blockC);

f(row_pixel2 + 4:row_pixel2 +
11,column_pixel2 : column_pixel2 +
7) = inv_DCT_blockC(:,:);
```

4. EXPERIMENTS AND RESULTS

In matlab we have the 'imwrite' command to write image to graphic file. We used this image to generate input files with varying quality. In matlab we can specify the quality as a scalar value in the *imwrite* command. The range is from [0,100]. 0 is lower quality and higher compression and 100 is higher quality and lower compression.
We applied our algorithm on various input images of varying compression rate and image quality. Figure 1 and 2 show input images compressed using JPEG scheme with a quality factor or 10 and 20 respectively. Figure 3 and 4 show the results obtained after our algorithm was applied. Furthermore PSNR can be used to measure the quality of reconstruction. It is an approximation to how humans perceive reconstruction quality of the image. We used built-in matlab command to calculate PSNR values with original uncompressed image as a reference. Overall it was observed that our algorithm significantly reduced the blocking artifacts from the input images(judged using objective and subjective measures). The algorithm has the characteristic of smoothing out undesired block edges while retaining its sharpness. This is possible because high frequency components of the given input image remain untouched by our algorithm. Lastly it was observed that algorithm is computationally efficient and can produce the output under 2secs. The major time consuming computation in our code is calculating the 2-D DCT of each block.

The following page(Figures 1-4) contains the input output visual comparisons of the images.

Fig. 1. Input Image Compressed using JPEG scheme with a quality factor of 10.

Fig. 3. Output Image obtained after algorithm was applied to input image in figure 1. PNSR = 27.5dB

Fig. 2. Input Image Compressed using JPEG scheme with a quality factor of 10.

Fig. 4. Output Image obtained after algorithm was applied to input image in figure 2. PNSR = 28.9dB

5. FUTURE IMPROVEMENTS

Through our experiments we propose following future improvements in the code

a. **Adaptive Threshold values**
Current Algorithm uses hard-coded threshold values to ensure original contents of the image remain preserved. Our aim here is to only remove sharp edges introduced by the compression algorithm and not remove any actual edges present in the image. The hard-coded values currently being used in the image can work well for some of the images but cannot generalize well. Therefore for our algorithm to work equally well with all sets of input images, we need to come up with a generic approach that uses local information of the image to come up with these values. The values should also be dependent of the bit rate of the given input image as they can result in varying amount of filtering for different bit rates.

b. **Boundary Block value comparisons**:
To save computations, only one pixel value comparison was done of the boundary containing block(condition 3 in step 3 of our algorithm). This step was to ensure that boundary containing block is of low frequency content[4]. Through experimentation we found that more number of comparisons can potentially result in better performance. However the correlation of computational savings vs quality gain is not clear at this point and further experimentation is needed to explore this.

6. CONCLUSION

In this project we applied an adaptive artifact removal technique proposed by Ying and Rabab[4]. From the results it was observed that the algorithm is fairly efficient in removing the blockiness. Since the algorithm is adaptive to local image information, it works well in preserving original image content such sharp edges etc. Future Improvements might include using adaptive threshold values and increased comparisons of values contained in the boundary containing block.

References

[1] Jain, Anil K., *Fundamentals of Digital Image Processing,* Englewood Cliffs, NJ, Prentice Hall, 1989, pp. 150-153.

[2] Pennebaker, William B., and Joan L. Mitchell, *JPEG: Still Image Data Compression Standard,* Van Nostrand Reinhold, 1993.

[3] Rafeal C. Gonazalez, Richard E. Woods, "Digital Image Processing", 3rd Edition.

[4] Ying Luo and Rabab K. Ward, "Removing the Blocking Artifacts of Block-Based DCT Compressed Images", *IEEE transactions on image processing*, vol. 12, no. 7, July 2003.

[5] B.Bing Zeng, "Reduction of blocking effect in DCT-coded images using zero-masking techniques," *Signal Process.,* vol. 79, no. 2, pp. 205–211, Dec. 1999.