Image Compression and Blurring Using Different Techniques

Manan Shah (B18CSE030)
Nishant Jain (B18CSE067)
Ishit Patel (B18EE021)
Manvendra Shah (B18EE026)
Manav Garg (B18BB016)

Abstract

The compression and blurring techniques used in the project are essential and efficient for professional use, though there are many other advancements that can be used in these algorithms to make the compression more efficient and useful. Given the time and resources, that is possible to achieve. Among the three techniques used that is DCT, DWT and Chroma Subsampling (bandwidth reduction), We observe that DWT is the best technique as it does lossless compression also compared to only lossy compression by DCT. In chroma subsampling (lossy compression) we are taking advantage of the human visual system's lower acuity for colour differences than for luminance.[1] In blurring, if we use the same kernel to blur a given image, and perform convolution in the spatial domain and multiply in the frequency domain we get the same result (blurred image). Blurring in frequency domain takes less execution time.

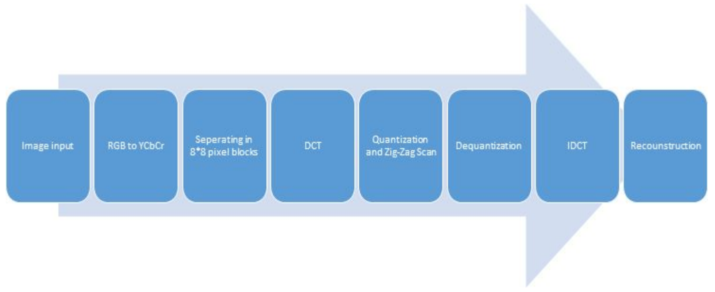
1. Introduction

Image compression is lowering the size of a graphical file without much degradation of the quality of the image to a level that is unacceptable or which shows a major difference to human eyes. The reduction of storing needs allows a number of images per fixed storage space and also reduces the time required to be transferred over the internet or downloaded from online sources. And Blurring techniques are used before edge detection and many such tasks.

2. Methodology

Image Compression using DCT

Image Input – We take a .jpg image as an input using imread() function. The input image must be a 512*512 image. Convert the RGB parameters of the image to YCbCr using conversion formulas.



Partition the image: We partition each of the three components (Y, Cb, Cr) of the image into 8×8 pixel blocks.[3]

Apply the DCT- Every M matrix is transformed using the two-dimensional DCT into the coefficient space. The two-dimensional DCT is given by:

$$D(i,j) = \left(\tfrac{1}{4}\right)C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} p(x,y)\left(cos\left(\frac{(2x+1)i\pi}{16}\right)cos\left(\frac{(2y+1)j\pi}{16}\right)\right)$$

where C(i) = 1√2 if i = 0 and 1 otherwise.[2] This can be expressed as a matrix multiplication using a matrix

$$T(i,j) = \frac{1}{\sqrt{8}} \quad for\ i = 0$$

$$T(i,j) = (\tfrac{1}{2})cos\frac{(2j+1)i\pi}{16} \quad for\ i > 0$$

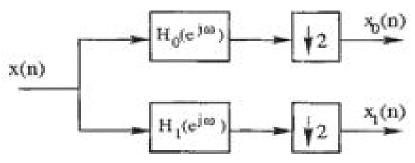
The corresponding DCT of M, which we denote D then amounts to D = TMT'.[2] The entries of D give the component-wise intensity change frequency of the image. The coefficients in the top left of D are the low-frequency intensity changes and the coefficients in the bottom right are the high-frequency intensity changes.[2]

Quantization - Quantization is the step where the actual compression takes place.[2] This process involves element by element division of the D matrices by a matrix known as the quantization matrix Q and then rounding the entries.[2] There is a standard JPEG quantization matrix.

Decompression - Firstly we multiply by the quantisation matrix on an element by element basis to obtain reconstructed DCT coefficient matrix R and then perform the inverse DCT on R to obtain the reconstructed image RI.[2] The inverse DCT can be calculated using the T matrix RI = T 'RT.[2]

Multilevel Decomposition of Image

Input: We start by taking an input image of 512x512. This image is taken from inbuilt function pywt.data.camera(), which is a standard image for image processing.



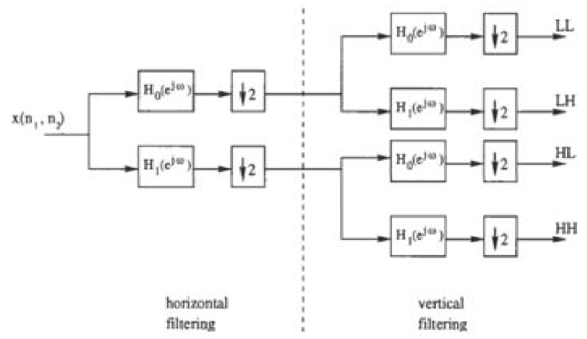
The input signal x(n) is processed through two paths.[3] In the upper path, x(n) is passed through a lowpass filter followed by downsampling by a factor of 2. In the lower path, x(n) is passed through a high-pass filter followed by downsampling by a factor of 2.[3] The kernel for low pass filter is the three-tap FIR filter h_0 (n) = (1/4,1/2, 1/4) and we have trivially obtained a high pass filter kernel as h_1(n) = (-0.04,0.08,-0.04). The outputs y_0(n) and y_1(n) after applying respective filters(convolution) are:

Low Pass: $y_0(n) = x(n) * h_0(n)$ **High Pass:** $y_1(n) = x(n) * h_1(n)$

Downsampled signal are $x_0(n) = y_0(2n)$ and $x_1(n) = y_0(2n)$

The same process for image is carried first on rows for horizontal filtering and then on columns for vertical filtering which create 4 sub-images LL (low-low), LH (low-high), HL(high-low) and HH (high-high) sub-bands.[3]

First-Level Decomposition() : **Second-Level Decomposition(first level decomposition on LL):**



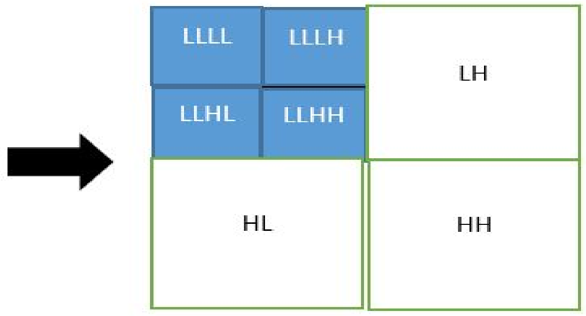


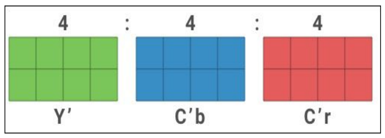
Fig 1.

Image Compression using chroma subsampling

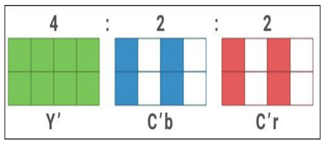
Image Input – We take a .jpg image as an input using imread() function. The input image must be a 512*512 image. Convert the RGB parameters of the image to YCbCr using conversion formulas.

Partition of Image: Now since the image has three components (luminance, the colour red and colour blue), compression can be applied to any of the three components.

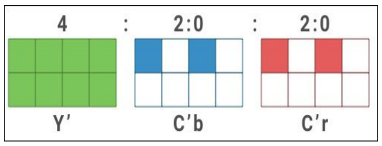
A signal with chroma 4:4:4 means that the original image and transports both luminance and color data entirely.[5] Here a:b:c means we take 2*a matrix from our original matrix where 'b' means the number of distinct pixel values in the top row and 'c' means the number of distinct pixels in the bottom row. In a four by two array of pixels, 4:2:2 has only two distinct values in the top row and bottom row. For achieving this we just copy the values at even column index and paste and odd column index in one horizontal row (vertically it would remain the same). Now 4:2:0 means '0' means that there is no distinct value in the bottom row. So, in this case, we just copy the value from the above row and paste in the bottom row.[4][5]



4:4:4



4:2:2



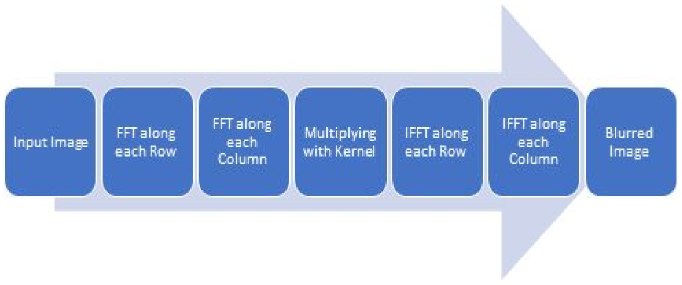
4:2:0

Blurring with Convolution in Spatial Domain :

Take the input image and a kernel of 5*5 having each value 1/25. The sum of the matrix must be 1 to get a blurred image.

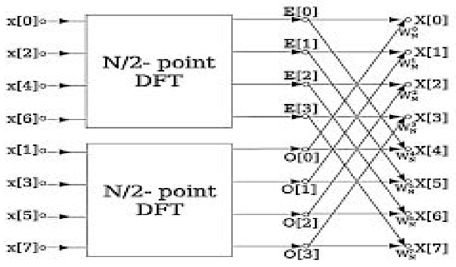
Blurring with Multiplication in Frequency Domain:

The image is first converted into Frequency Domain by the following steps :



$$\sum_{n=0}^{N-1} a_n e^{-2\pi i n k/N} = \sum_{n=0}^{N/2-1} a_{2n} e^{-2\pi i (2n) k/N}$$
$$+ \sum_{n=0}^{N/2-1} a_{2n+1} e^{-2\pi i (2n+1) k/N}$$
$$= \sum_{n=0}^{N/2-1} a_n^{\text{even}} e^{-2\pi i n k/(N/2)}$$
$$+ e^{-2\pi i k/N} \sum_{n=0}^{N/2-1} a_n^{\text{odd}} e^{-2\pi i n k/(N/2)},$$

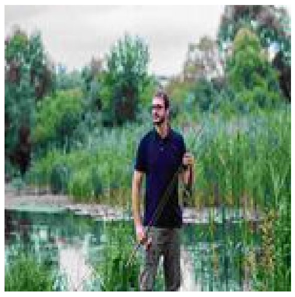
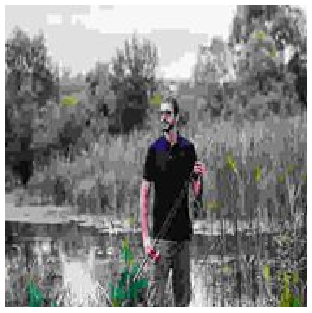
Flowchart of Blurring in Frequency Domain of Image
Recursive formula of function calling in FFT



Recursive call is made on even and odd index of the present array, then the FFT of the odd index and even index are obtained and finally are merged to get FFT of the present array as shown in the figure. Then it is multiplied with low pass kernel and IFFT on it to get the output image.[6]

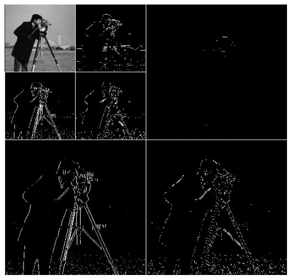
3. Results and Observations

DCT :

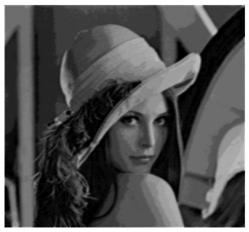
a) b)  c) 

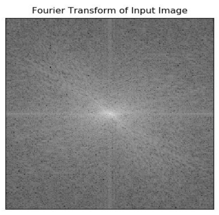
a)original image b) compressed with q(1,10,10) c) compressed with q(10,20,20)

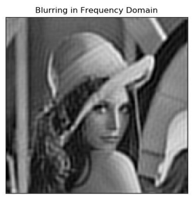
Multilevel decomposition of Image: Original vs Decomposed into different sub-bands (Refer Fig. 1)



<u>Blurring :</u>







Input Image Blurring(Spatial Domain) FFT of Image Blurring(Frequency Domain)

Execution Time in Spatial Domain: 22.90096259117126 sec

Execution Time in Frequency Domain : 8.395942211151123 sec(FFT) + 7.731654167175293 sec(IFFT)

4. Conclusions and Limitations

DWT is the best technique amongst those used in the project as it can do lossless compression also. The transforms can be used as preprocessed data for other tasks. But for fine analysis, it becomes computationally difficult. DCT is lossy and thus creates low efficiency but can be made better by using entropy coding and Huffman tables and the transforms can be reconstructed in DWT as a future scope. Blurring in Frequency Domain takes less execution time then Blurring in Spatial Domain. Here FFT deals with complex values and also there is some approximation because of values of the order 10^{-8}. Chroma subsampling creates losses in originality in images with high colour variations.

References

[1] S. Winkler, C. J. van den Branden Lambrecht, and M. Kunt (2001). "Vision and Video: Models and Applications". In Christian J. van den Branden Lambrecht (ed.). *Vision models and applications to image and video processing*. Springer.

[2] School of Mathematics and Statistics, Sydney, research paper http://www.maths.usyd.edu.au/u/olver/teaching/Computation/ExampleProject.pdf

[3] *Handbook of Image and Video Processing*, Academic Press, San Diego, USA, 2000, pp. 289-296.

[4] M. J. McNamara, *Chroma Subsampling Explained: Giving a Little to Save a Lot:* https://www.projectorcentral.com/chroma-subsampling-explained.htm (Accessed on 21/11/19)

[5] A. Babcock, Chroma Subsampling 4:4:4 vs 4:2:2 vs 4:2:0, https://www.rtings.com/tv/learn/chroma-subsampling.

[6] J. Vanderplas, *Understanding the FFT Algorithm*, https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/ (Accessed on 20/11/19)

[7] S. W. Smith, *Fourier Image Analysis*, http://www.dspguide.com/ch24/5.(Accessed on 18/11/19)