CompE565, Fall 2021

Homework 2: JPEG based Image Compression

Prepared by

Jacob Bananal (820416396)

E-mail: *jbananal2509@sdsu.edu*

Computer Engineering

San Diego State University

Table of Contents

List of figures

Introduction

For homework assignment 2 on JPEG based image compression, I got a better understanding on how to implement the concepts of encoding and decoding an image using Discrete Cosine Transformation and Quantization. I was able to compute the DCT blocks of the Y, Cr, and the Cb components in MATLAB and get an idea on what these blocks look like when displayed. I have also created a function that zig zag scans the AC component of the DCT block which will be at the end of the code. After the encoding part, the decoding part was where I implemented the inverse DCT transformation and was able to recreate the image. Towards the end of the homework assignment, I was able to see the error between the original y component and the reconstructed component.

Procedural Section

(1) Encoder

(a) Compute the 8x8 block DCT transform coefficients of the luminance and chrominance components of the image.

Before finding the 8x8 block DCT transform, I first displayed the YCbCr and subsampled the Cb and Cr bands using 4:2:0 from the original image we used in the previous homework assignment. Afterwards, I used the block processing function that will perform the DCT transform of the 8x8 blocks which I was required to display the first two blocks of the 6th row of the luminance component.

(b) Quantize the DCT image by using the JPEG luminance and chrominance quantizer matrix from the lecture notes.

There were two parts of this step that I needed to perform in order to quantize the DCT image which was first finding the DC DCT coefficients and then the AC DCT coefficients. I used the given luminance and chrominance matrices that were from the lecture powerpoint and the video. I used a similar DCT transform processing like I did in the previous step, except I quantized by performing the given equations and the matrices I mentioned above in order to find the DC DCT coefficients which allowed me to display the required first two blocks of the 6th row. For the AC DCT coefficients, I had to write a separate function called zig zag which will move right of the first & last row, move down on the first & last column, and finally move diagonally up & down on the first row and last column. My zig zag function is at the end of the code so I will be calling it which will then zig zag through the matrices and display the required AC DCT coefficients.

(2) Decoder

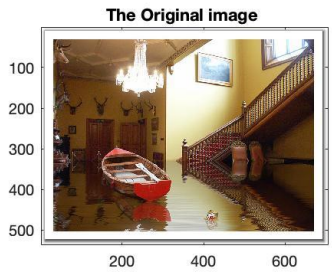
(a) Compute the inverse Quantized images obtained in Step (b)

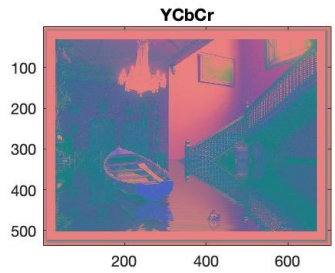
For the inverse quantized images, I used the same block processing in the previous steps but instead, I multiplied the luminance matrix with the quantized matrix. Then, I did the same for the Cb and Cr components. After multiplying, that is where I applied the block processing method and then displayed the required first two blocks of the 6th row.

(b) Reconstruct the image by computing Inverse DCT coefficients

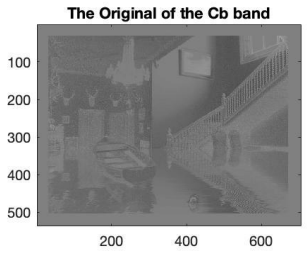
For this step, I applied the inverse DCT to the y, Cr, and Cb components separately using the same block processing method. After applying, I then moved on to using the linear interpolation which I used from homework 1. With the linear interpolation, I was able to use this method to reconstruct the images I wanted. In order to display the error, I simply subtract the original y band from the reconstructed y band which I will display the output below in the results section. Towards the end, I was required to calculate the PSNR for the luminance component of the decoded image. I did this by using MATLAB command “psnr()” which will then calculate the reconstructed YCbCr and the original YCbCr.

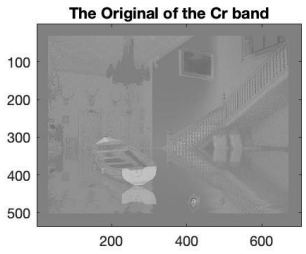
Results:

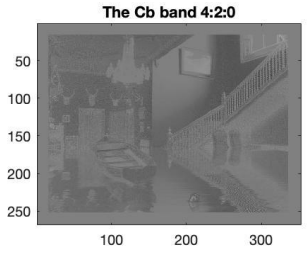


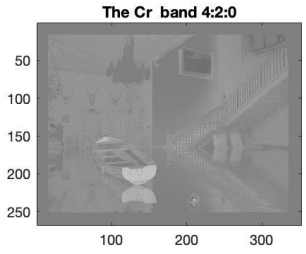


The first image above shows the original image along with the YCbCr image that I converted using the method I used in the previous homework.

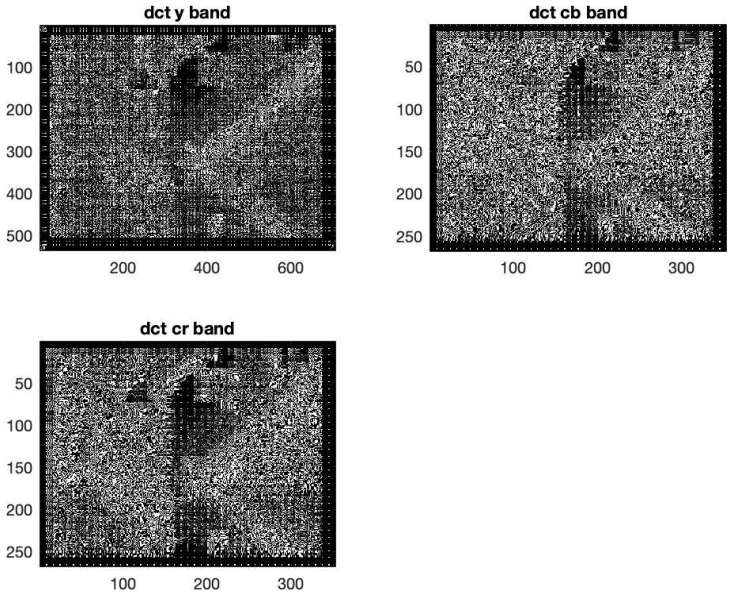




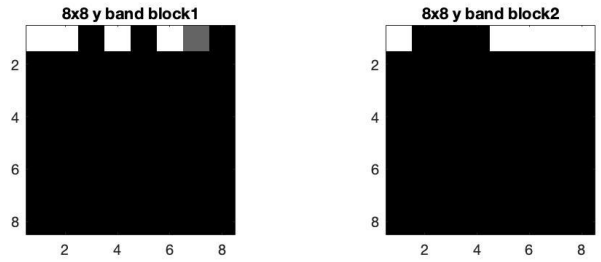




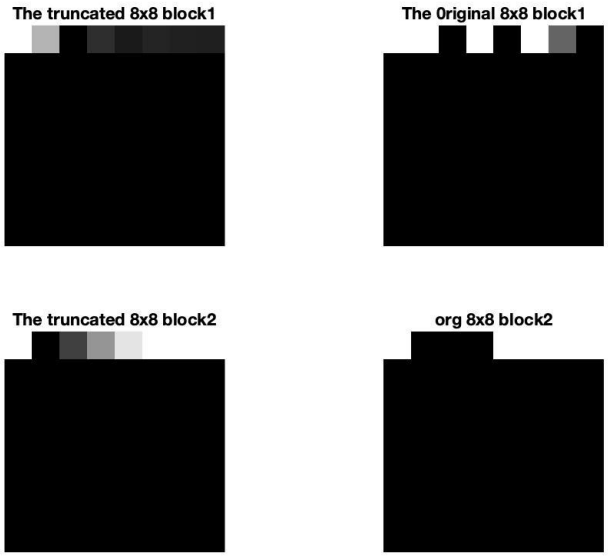
The second image shows the original image of the Cb and Cr band then using the 4:2:0 method, I also displayed the Cb and Cr band 4:2:0.



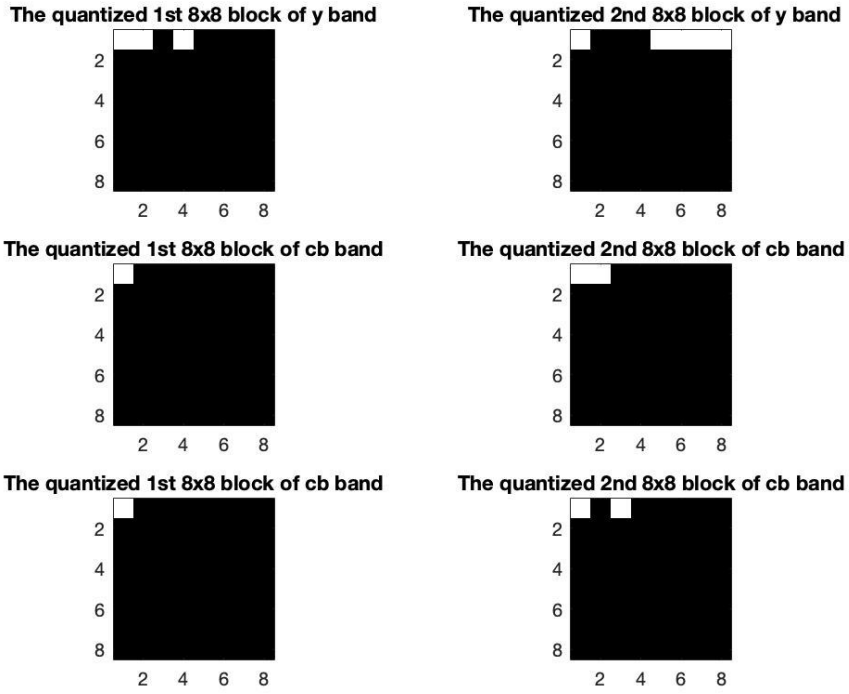
For the third figure, this displays the DCT transform of the y, Cb, and Cr components using the DCT transform method.



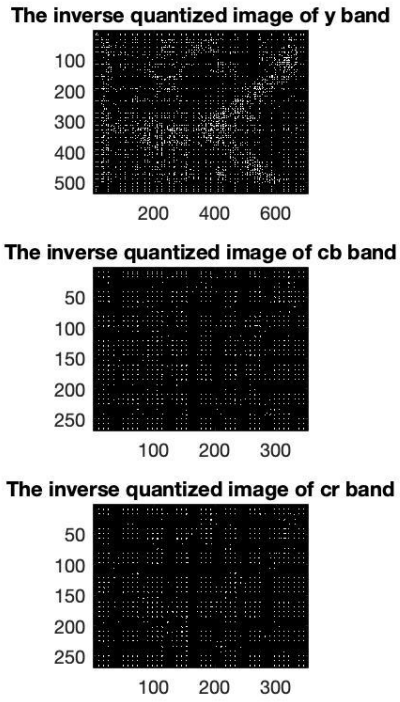
For the fourth figure, this shows the 8x8 y band of the first two blocks from the 6th row.



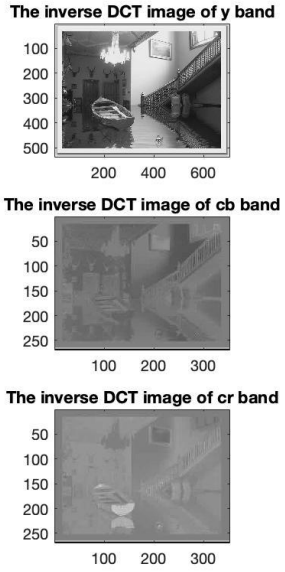
In the fifth figure, using the truncated function I created towards the end of the code, this will display the truncated of the 8x8 block 1 and 2 from the 6th row.



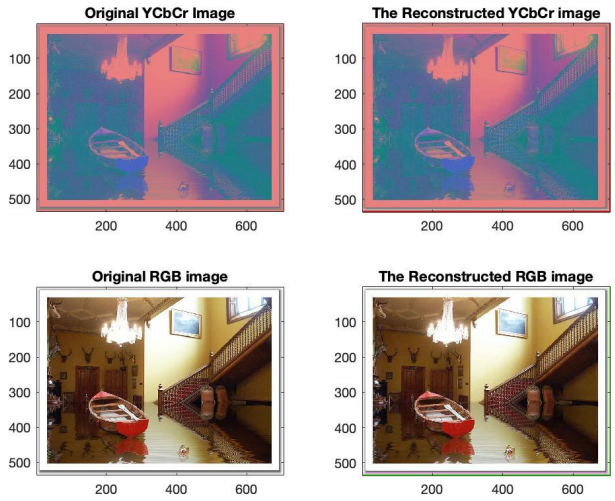
In figure six, I was required to quantize the first two blocks of the y, Cr, and Cb components as you can see above.



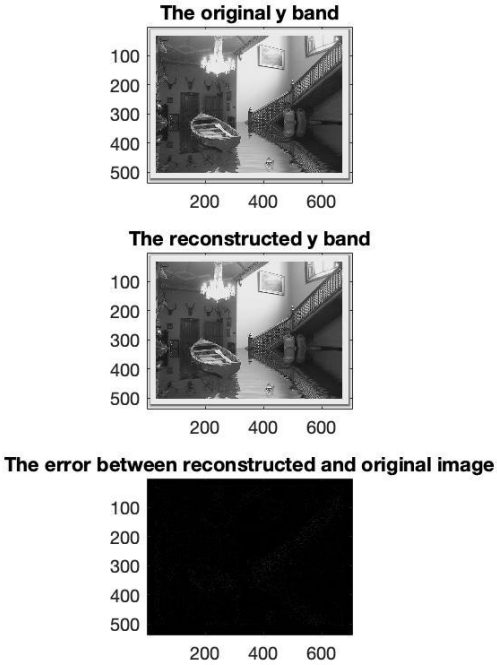
For the seventh figure, I inverse quantize the images of the y, Cb, and Cr components.



In the eighth figure, this shows the inverse of the DCT images for the y, Cb, and Cr components.



In the ninth figure, this displays the original of the YCbCr and RGB images along with the reconstructed images of both of them.



Finally, for the last figure, this shows the original image of the y component and the reconstructed component. By subtracting the original y component from the reconstructed y component, this simply gives me the error of both of them as you can see above.

PSNR result:

The Peak SNR of decoded Y band: 33.519912

I got this from using the MATLAB command psnr in order to get the required peak SNR of the y component.

Conclusion

In conclusion, I've gained a better knowledge on encoding and decoding an image that I was given. For the encoding part, I thought it was straight to the point because all we needed to do was use the same block processing method but we needed to do some stuff beforehand like finding the YCbCr and using 4:2:0. The only issue I had and took me a while to figure out was finding the first two blocks of the 6th row. I thought that took me the longest to figure out but I was able to accomplish it. Also, with the zig zag function, I found it interesting just the way I learned about it and the way it works. When it came to the decoding part, it was simply just using some of my MATLAB code from my previous homework and translating it over to the decoding part which helped me a lot. I wasn't sure whether or not my PSNR number was right but hopefully I was able to get the right answer for it.

References

[1] MathWorks. (2021). “psnr” (R2021b).

https://www.mathworks.com/help/images/ref/psnr.html

[2] MathWorks. (2021). “Discrete Cosine Transform” (R2021b).

https://www.mathworks.com/help/images/discrete-cosine-transform.html