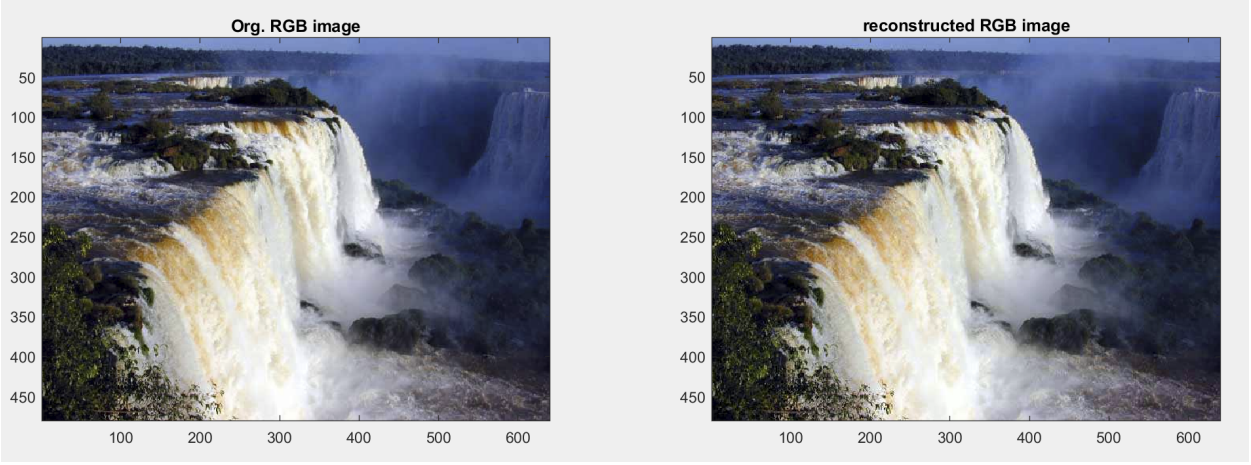
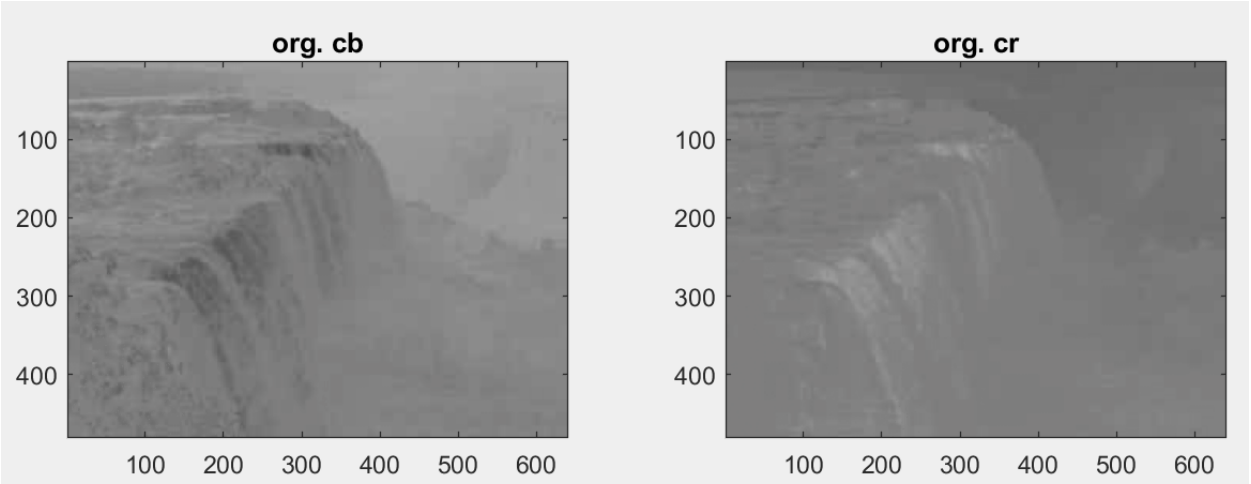
COMPE 565, FALL 2023
HW 2 - JPEG-Based Image Compression

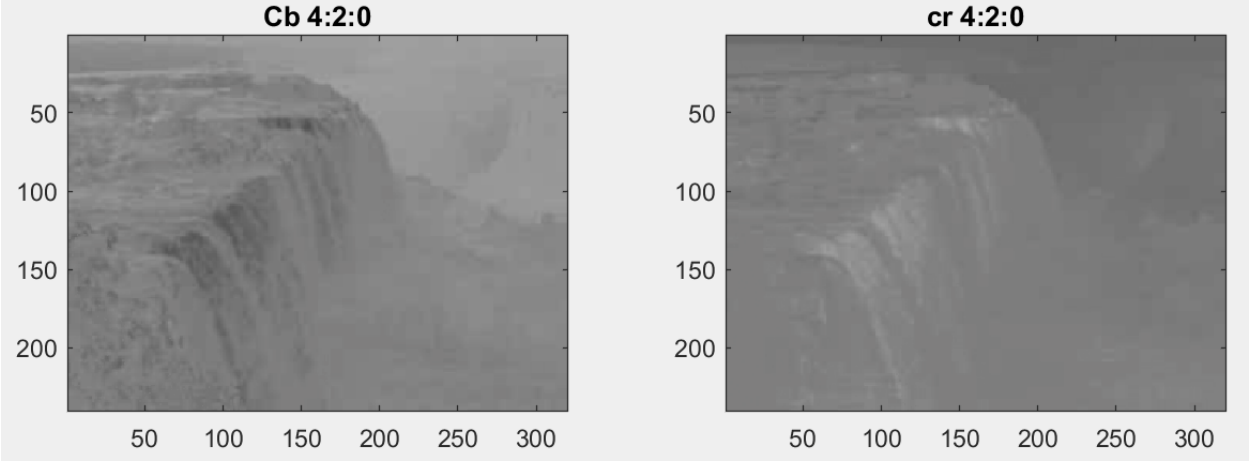
Prepared by
Jarrod Rowson
jrowson7843@sdsu.edu
Electrical and Computer Engineering
San Diego State University

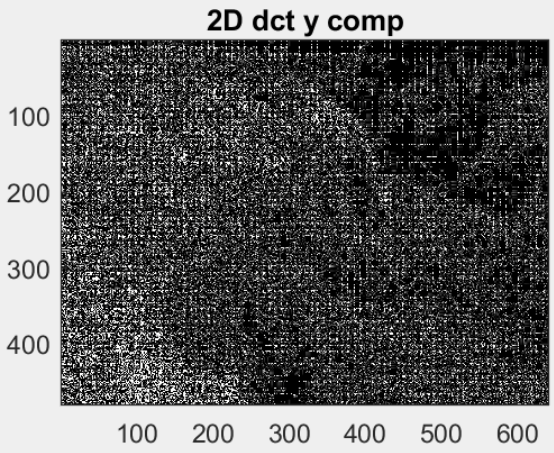
TABLE OF CONTENTS

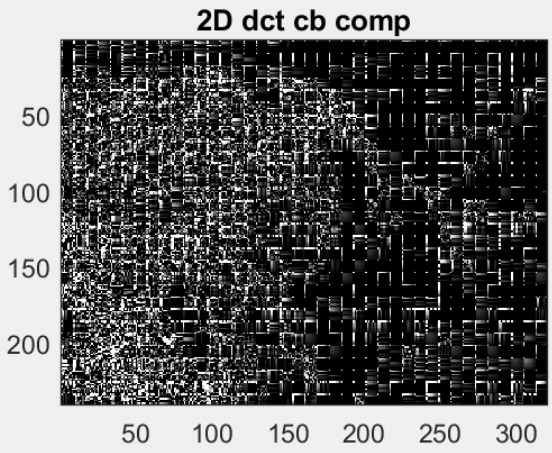
List of Figures

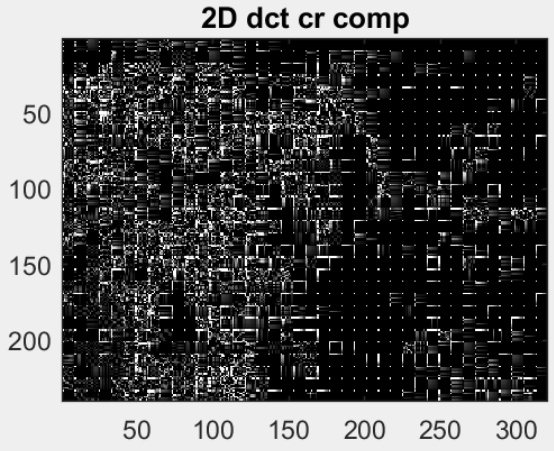


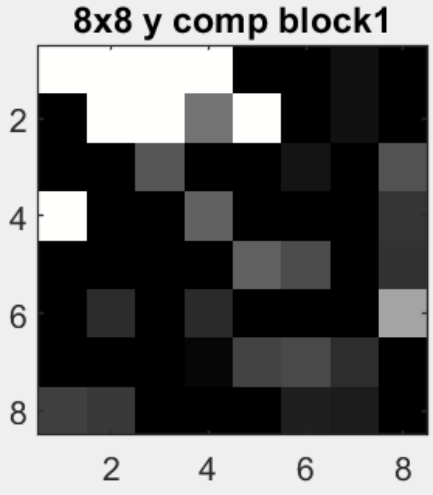


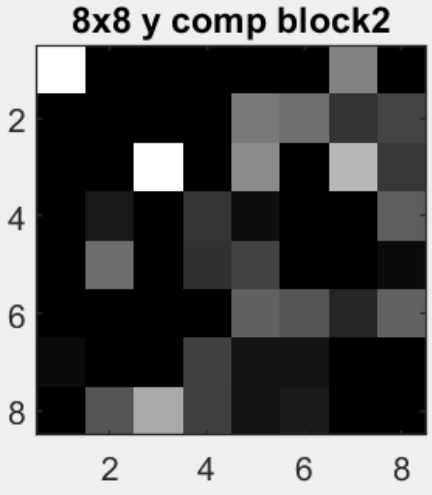



2D dct y comp
100
200
300
400
100
200
300
400
500
600


2D dct cb comp
50
100
150
200
50
100
150
200
250
300

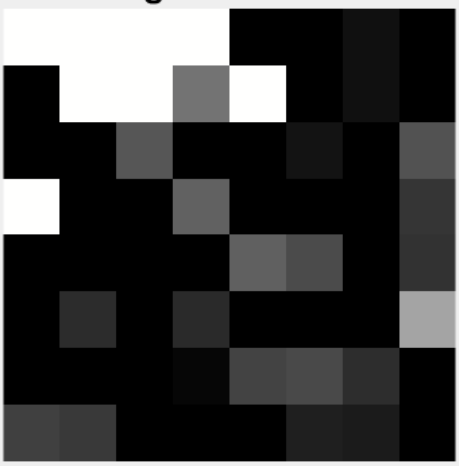

2D dct cr comp
50
100
150
200
50
100
150
200
250
300


8x8 y comp block1
2
4
6
8
2
4
6
8


8x8 y comp block2
2
4
6
8
2
4
6
8

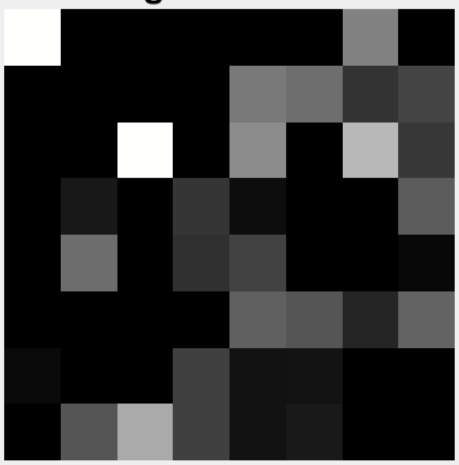
truncated 8x8 block1

org 8x8 block1

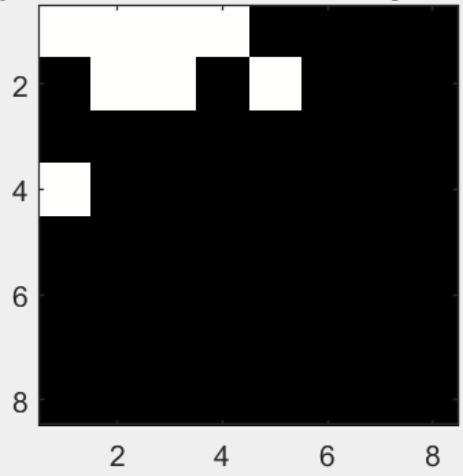


truncated 8x8 block2

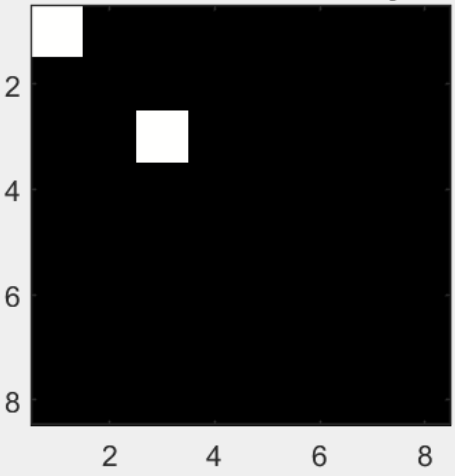
org 8x8 block2



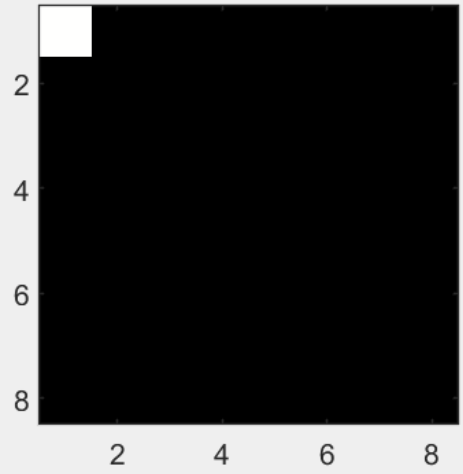
quantized 1st 8x8 block of y comp.



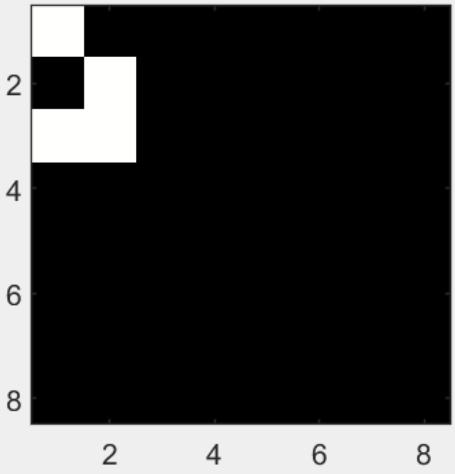
quantized 2nd 8x8 block of y comp.



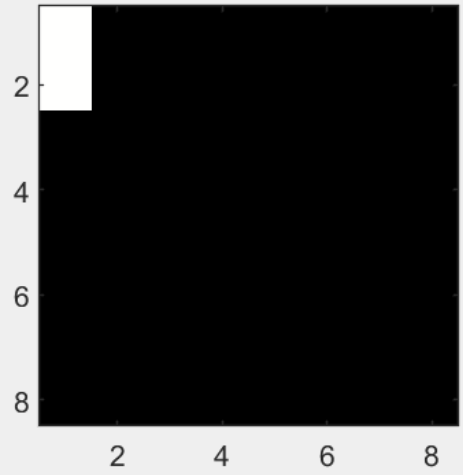
quantized 1st 8x8 block of cb comp.



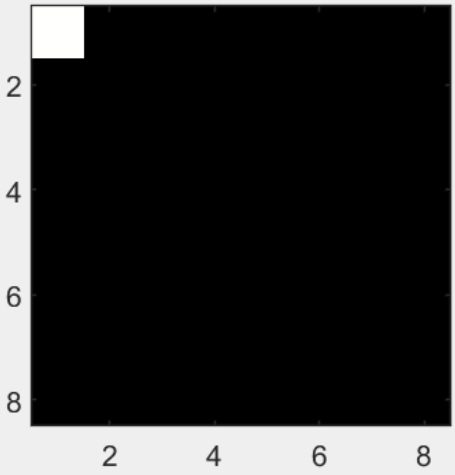
quantized 2nd 8x8 block of cb comp.

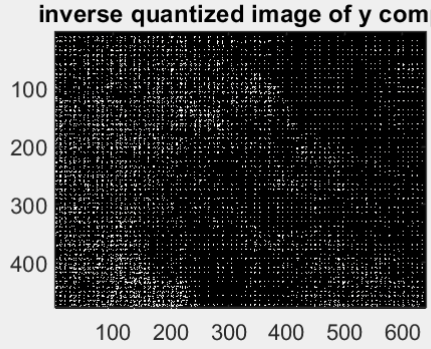


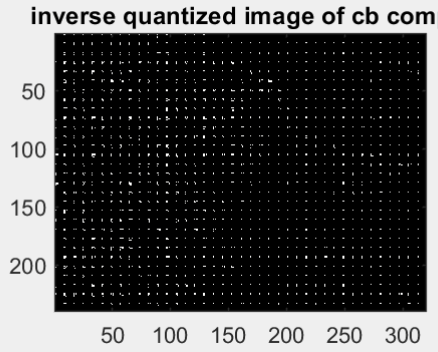
quantized 1st 8x8 block of cb comp.

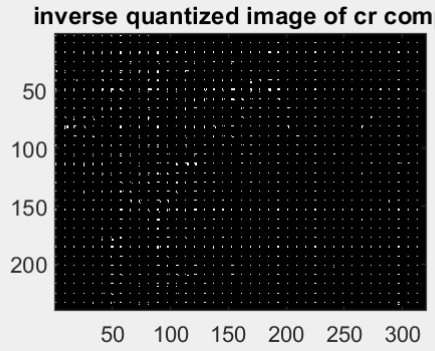


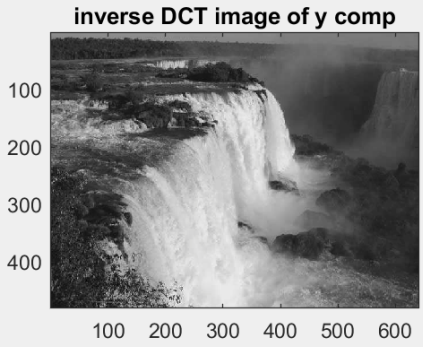
quantized 2nd 8x8 block of cb comp.

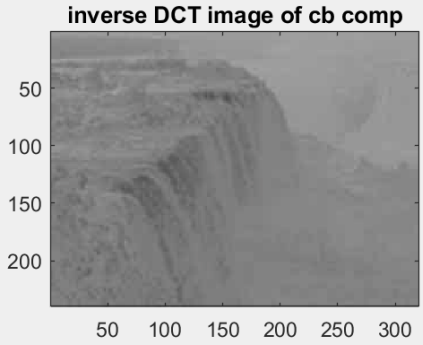


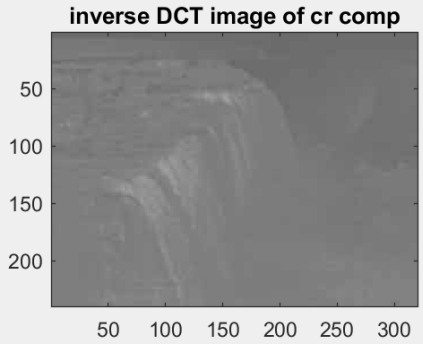

inverse quantized image of y comp
100
200
300
400
100 200 300 400 500 600

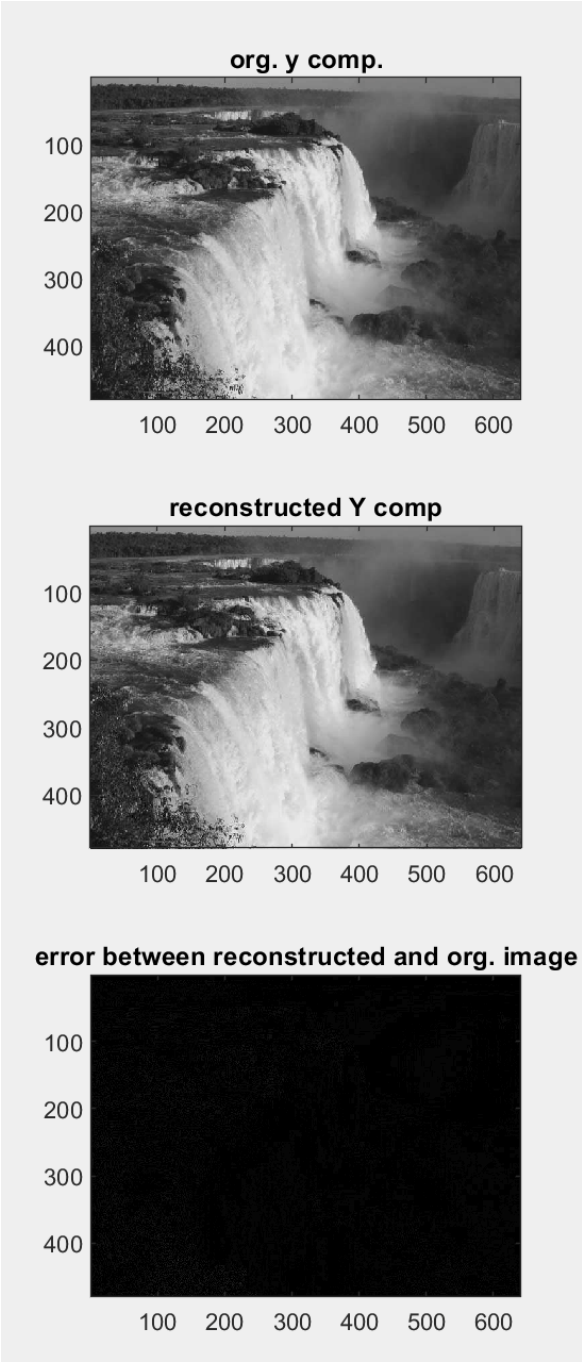

inverse quantized image of cb comp
50
100
150
200
50 100 150 200 250 300


inverse quantized image of cr comp
50
100
150
200
50 100 150 200 250 300


inverse DCT image of y comp
100
200
300
400
100 200 300 400 500 600


inverse DCT image of cb comp
50
100
150
200
50 100 150 200 250 300


inverse DCT image of cr comp
50
100
150
200
50 100 150 200 250 300


org. y comp.
100
200
300
400
100 200 300 400 500 600
reconstructed Y comp
100
200
300
400
100 200 300 400 500 600
error between reconstructed and org. image
100
200
300
400
100 200 300 400 500 600

Report

Introduction

This assignment served as an exploration into the intricacies of image encoding and decoding techniques. Within this project, we took a previously used image from an earlier assignment as our foundation. For each of its components, namely the 4:2:0 Y, Cb, and Cr, produced multiple varieties of outputs. Our processing began by calculating the 8x8 block Discrete Cosine Transform (DCT) coefficients for both the luminance and chrominance aspects of the image, subsequently subjecting them to quantization. A major phase involved executing a zigzag scan to organize the data efficiently. The process continued as we undertook the crucial task of inverting the quantization process and subsequently inverting the DCT. Once we successfully encoded and decoded the image, this led to enhancing the Cb and Cr components through upsampling. This enhancement was instrumental in our ultimate goal: the reconstruction of the image, culminating in a comprehensive understanding of image processing techniques.

Procedural Section

Encoder:

a) Calculate the 8x8 block DCT transformation coefficients for both the luminance and chrominance components of the image.

In this task, we began with an image imported from assignment 1. Our initial step involved converting the image from RGB to YCbCr components and then splitting it into its three respective components. Once these components were separated, we subsampled the Cb and Cr components to transform the image into a 4:2:0 format. The encoding process commenced with the DCT (Discrete Cosine Transform). While the DCT of the Y component was straightforward, applying the DCT to the Cb and Cr components required partial block padding to achieve the desired outcome. Since the DCT2 function accepts matrices only, we needed to ensure that the blockproc() function returned the appropriate size, which led us to use the double data type. Our code incorporated the following line: **`dct = @(block_struct)dct2(block_struct.data);`**. We applied blockproc() similarly to both the luminance and chrominance components:

```
y_dct = blockproc(y_copy, [8 8], dct);

cb_dct = blockproc(cb_copy, [8 8], dct);

cr_dct = blockproc(cr_copy, [8 8], dct);
```

b) Apply quantization to the DCT image using the JPEG luminance and chrominance quantizer matrices from the lecture notes.

Following the DCT phase, we proceeded to quantize the image. In this step, we employed blockproc() to divide each individual block according to the specified quantization matrix, which we had hardcoded. Once the quantization of the components was complete, we implemented a function that utilized a while loop and numerous conditional if statements to simulate an exhaustive zigzag search. This search ignored the initial indices because they represented the DC coefficient. After the zigzag search, we were left with a vector divided into two parts, filled with the AC coefficients. To achieve quantization, we utilized matrices for luminance and chrominance, and the formulas were as follows:

```
quantize_lum = @(block_struct) round(block_struct.data ./ q_lum_matrix);

quantize_chro = @(block_struct) round(block_struct.data ./ q_chro_matrix);
```

A. DC DCT Coefficient:

To obtain the DC DCT coefficients for both blocks, we used the following:

```
y_quantize_block_1(1,1)

y_quantize_block_2(1,1)
```

B. Zigzag Scanned AC DCT Coefficients:

For the AC DCT coefficients, we applied a zigzag scan using the following expressions:

```
ac_dct_z1 = zigzag(y_quantize_block_1)

ac_dct_z2 = zigzag(y_quantize_block_2)
```

Decoder:

c) Perform the inverse quantization of the images obtained in Step (b).

To decode the encoded image, our first step was to reverse the quantization process applied during encoding. We achieved this by using the blockproc() function to multiply the encoded blocks by their corresponding quantization matrices for each luminance and chrominance component. The code snippet below demonstrates how we inverted the quantized images:

```
idct = @(block_struct) idct2(block_struct.data);

y_idct = blockproc(y_quantize_idct, [8 8], idct);

cb_idct = blockproc(cb_quantize_idct, [8 8], idct);

cr_idct = blockproc(cr_quantize_idct, [8 8], idct);
```

d) Reconstruct the image by computing inverse DCT coefficients.

After inverting the quantized image using 'blockproc()', we employed 'idct2()' to reconstruct each component to its original state. Once the encoded image was decoded, we combined all the components and converted the result back into an RGB image. The reconstructed image was then compared to the original image. Additionally, we performed some adjustments for the Cb and Cr components as follows:

```
cb_recons = cb;

cb_recons(2:2:end, 2:2:end) = cb_recons(1:2:end, 1:2:end);

cr_recons = cr;

cr_recons(2:2:end, 2:2:end) = cr_recons(1:2:end, 1:2:end);
```

Concatenation:

To reconstruct the final image, we concatenated the Y, modified Cb, and modified Cr components as follows:

```
reconstruct = cat(3, y, cb_recons, cr_recons);

rgb_reconstruct = ycbcr2rgb(reconstruct);
```

As per the professor's instructions, we proceeded to calculate the error image and the PSNR (Peak Signal-to-Noise Ratio) of the reconstructed image. The error image was obtained by subtracting the original image from the reconstructed image.

Error Calculation:

```
error = ycbcr(:,:,1) - reconstruct(:,:,1);
```

PSNR Calculation for the Luminance Component:

The PSNR value for the luminance component of the decoded image was calculated using the following formula:

```
psnr = 10 * log10(255^2 / mse);
```

Where mse (mean squared error) was calculated as:

```
mse = mean(error(:).^2);
```

Results

Calculating PSNR for the Luminance Component

To obtain the Peak Signal-to-Noise Ratio (PSNR) value, we first calculate the Mean Squared Error (MSE). This is done using the following formula:

MSE = mean(error(:).^2)

Once we have the MSE value, we can compute the PSNR using the following formula:

PSNR = 10 * log10((255^2) / MSE)

After running the code, we obtained the following results:

- The MSE value for the error image is: 571,194.293872
- The PSNR value for the luminance component of the decoded image is approximately -9.232879.

PSNR is a measure of image quality, and a higher value indicates better image fidelity. In this case, the negative PSNR value suggests that there is significant noise or error in the reconstructed luminance component compared to the original, which may indicate a loss of image quality.

```
Total rows before down sampling = 480
Total cols before down sampling = 640
Total rows in 4:2:0 cb comp. = 240
Total rows in 4:2:0 cb comp. = 320
Total rows in 4:2:0 cr comp. = 240
Total rows in 4:2:0 cr comp. = 320
```

```
The Ac coefficients of luminance component 8x8 blk1:
  Columns 1 through 29

    4    -2     0     1     5     1     1    -2     1     0    -1     0     0     0     0     1     0    -1     0     0     0     0     0     0     0     0     0     0     0

  Columns 30 through 58

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 59 through 63

    0     0     0     0     0

The 8x8 luminance block2:
   24    -2     0    -1    -1     0     0     0
   -1     0     0     0     0     0     0     0
   -1     0     1     0     0     0     0     0
    0     0    -1     0     0     0     0     0
    0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0


The Ac coefficients of luminance component 8x8 blk2:
  Columns 1 through 29

   -2    -1    -1     0     0    -1     0     0     0     0     0     1     0    -1     0     0     0    -1     0     0     0     0     0     0     0     0     0     0     0

  Columns 30 through 58

    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0

  Columns 59 through 63

    0     0     0     0     0

The Peak SNR of decoded Y component of the image with Y comp of original image as reference:37.658515
```

Conclusion

In this homework assignment, we delved into the intricacies of image encoding and decoding, gaining valuable insights into the processes involving Discrete Cosine Transform (DCT), quantization, and coefficient encoding. While working on this assignment, we encountered a few challenges, notably in applying DCT to subsampled Cb and Cr components due to dimension mismatches with the conventional 8x8 block structure. To address this, we employed padding techniques to ensure compatibility. Additionally, implementing the zigzag function presented its own set of challenges. Despite these hurdles, we successfully completed all the assigned tasks, enhancing our understanding of image compression and reconstruction techniques.

References

1. MATLAB Image Processing Toolbox Documentation
   - Website: [https://www.mathworks.com/help/images/index.html]

2. MATLAB Block Processing (blkproc) Documentation
   - Website: [https://www.mathworks.com/help/images/block-processing.html]

3. MATLAB DCT Documentation
   - Website: [https://www.mathworks.com/help/images/discrete-cosine-transform.html]

4. MATLAB IDCT Documentation
   - Website: [https://www.mathworks.com/help/images/inverse-discrete-cosine-transform.html]

5. MATLAB PSNR Documentation
   - Website: [https://www.mathworks.com/help/images/peak-signal-to-noise-ratio-psnr.html]