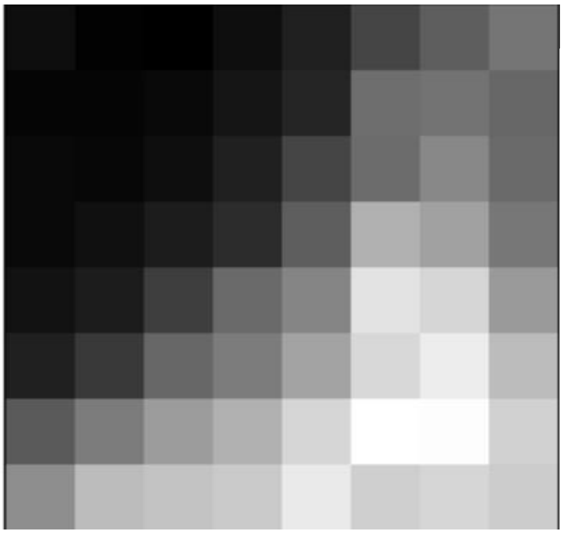
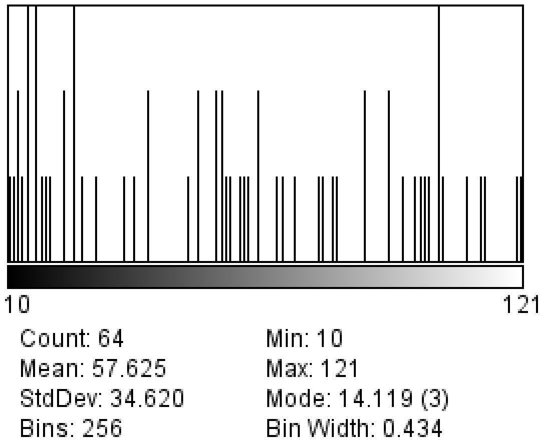
Session 4

9. Lossy JPEG Image Approximation

1. 8×8 pixels image containing standard JPEG quantization weights Q at 50% quality was created as shown in figures 1.





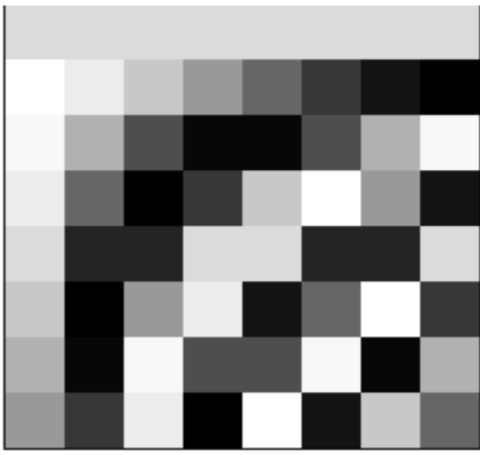
8*8_Q at 50% quality

Histogram of 8*8_Q at 50% quality

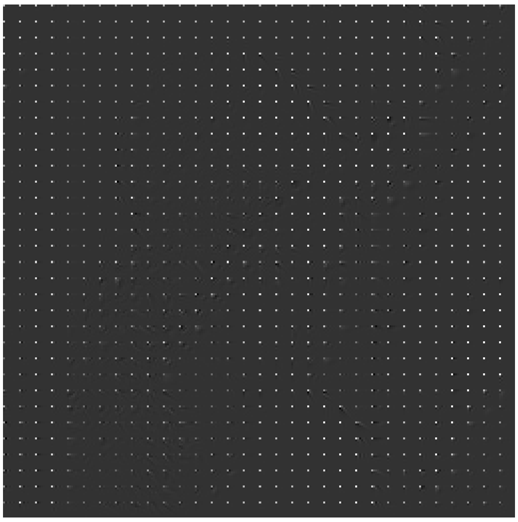
Figures1

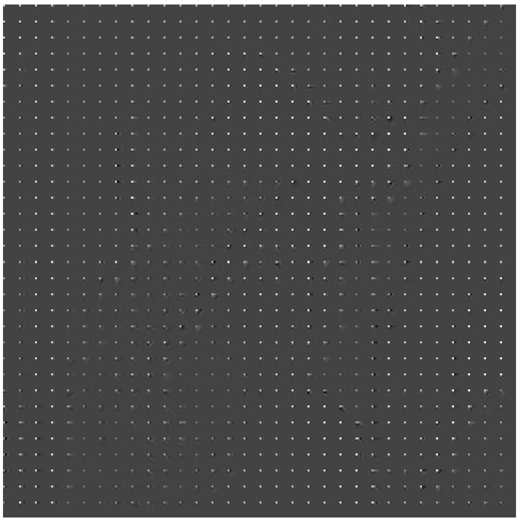
The common image trend decides the values in the q-matrix. The DC component of an image and low AC components dominates an image representation. The top left corner of the DCT has majority of the image strength, the q-matrix is made smaller at the top left corner to preserve this part of the image. The values in the bottom right of the q matrix are made high because the top left portion already characterizes well the image. High frequency signals are attenuated.

2. An approximate function that apply for each 8 × 8 pixels blocks: DCT, Q, IQ, IDCT was written.



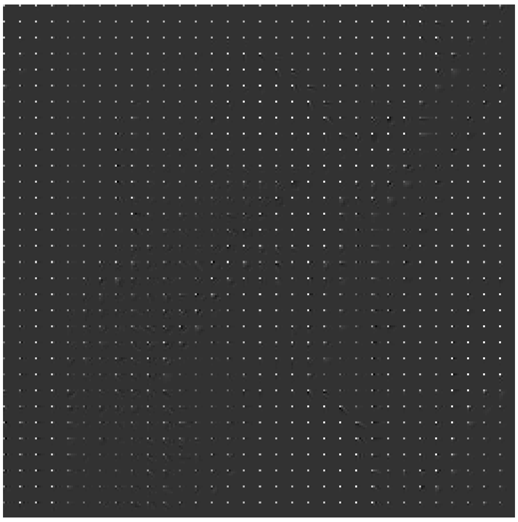
DCT_ matrix (8*8)





DCT

DCT_Q



DCT_Q_IQ

DCT_Q_IQ_IDCT

Figures 2

3. A clip function for exporting images to 8 bpp integer grayscale values was written limiting integer values in the valid range [0..255].

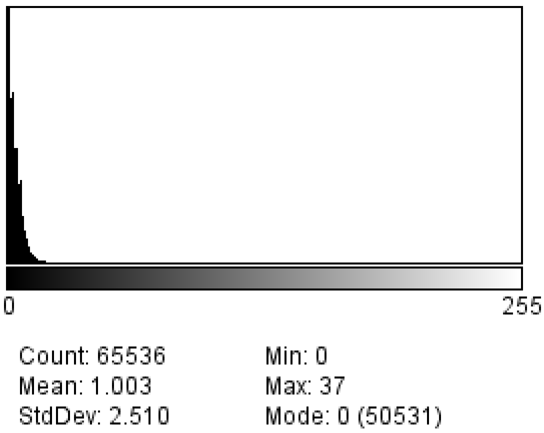
4. Create a difference image of your 8 bpp result with a baseline JPEG file at 50% quality

– Set the JPEG quality in the “Edit > Options > Input/Output...” menu in ImageJ

– Is the difference image zero everywhere, as expected? If not, can you explain why?

Clipped image

50% quality



Difference Image

Histogram of difference Image

Figures 3

The difference image is mostly “0” everywhere. This is because of the slight difference in the implementation of the clipping function of the Image J when compared to that of C++.

Packing DCT coefficients

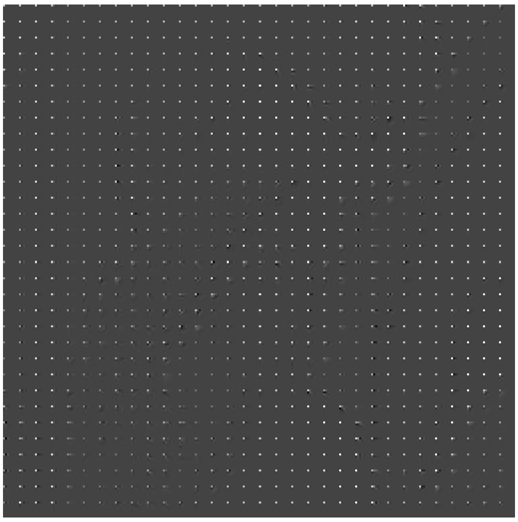
1. Write two encode and decode functions for splitting the approximate method into:

1. Encoding an original image into 8 × 8 pixels blocks of quantized DCT coefficients

2. Decoding the image approximation from 8 × 8 pixels blocks of quantized DCT coefficients.

An encode function was written to encode an original image into 8*8 pixels blocks of quantized DCT coefficients.

A decode function was written to decode the image approximation from 8 × 8 pixels blocks of quantized DCT coefficients.



Encoded Image

Decoded Image

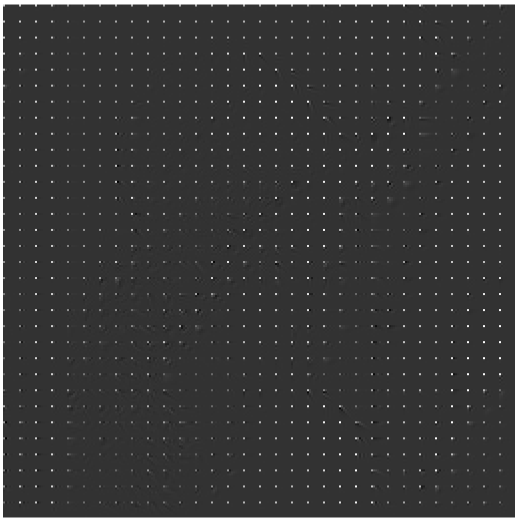
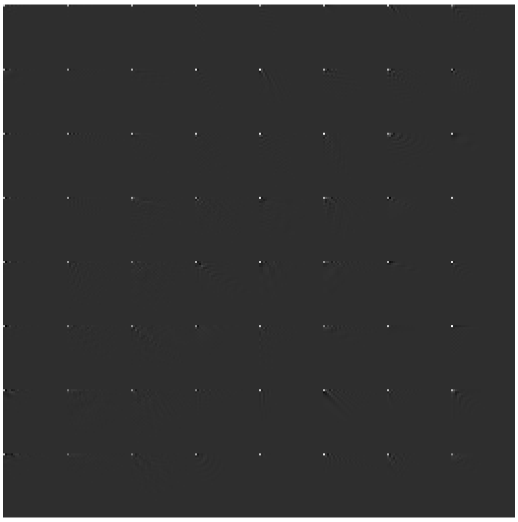
Figures 4

2. Compare two ways to layout the DCT coefficients (for a 256 × 256 pixels image):

1. 256 × 256 pixels layout with a 32 × 32 grid of 8 × 8 contiguous DCT coefficients

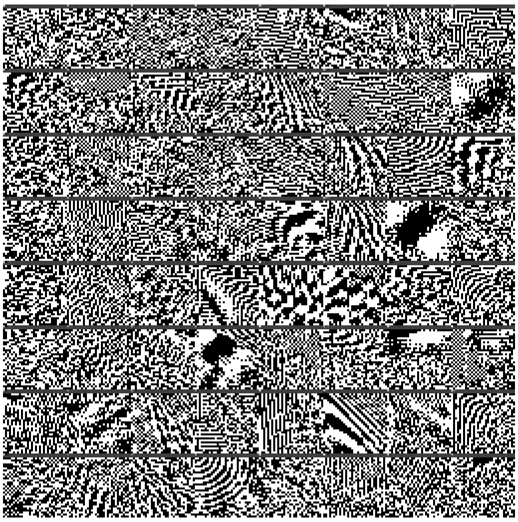
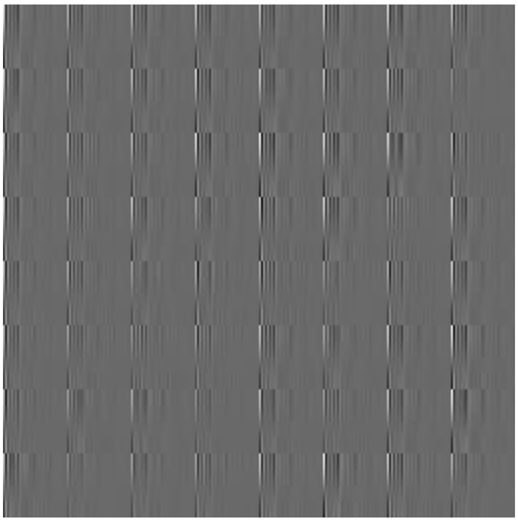
2. 256 × 256 pixels layout with a 8 × 8 grid of 32 × 32 interleaved DCT coefficients

3. Compare side-by-side the two options above. What is your favorite layout? Why?

32 × 32 grid of 8 × 8 contiguous DCT coefficients   8 × 8 grid of 32 × 32 interleaved DCT coefficients

Figures 5

Encoded Interleaved   Decoded Interleaved

Figures 6

I prefer 32 × 32 grid of 8 × 8 contiguous DCT coefficients because the encoded and decoded image produced a meaningful result.

I also tried encoding and decoding the interleaved pixels and the result is as shown in figures 6. The reason why this happened was because I used a Quantization matrix of 8*8 (64 elements) but (32*32 elements are required). If I used a quantization matrix of 32*32, I am hopeful I would get a better Encoded and Decoded image when compared to the one in Figures 6.

Q matrix (32*32): Only 64 elements are present

Figures 7