

## Session 5

### Delta encoding of DC coefficients

1. Create a 32 x 32 pixels image from the quantized DC terms of each 8 x 8 pixels block  
Evaluate qualitatively and quantitatively the difference with the downsized (average) image.

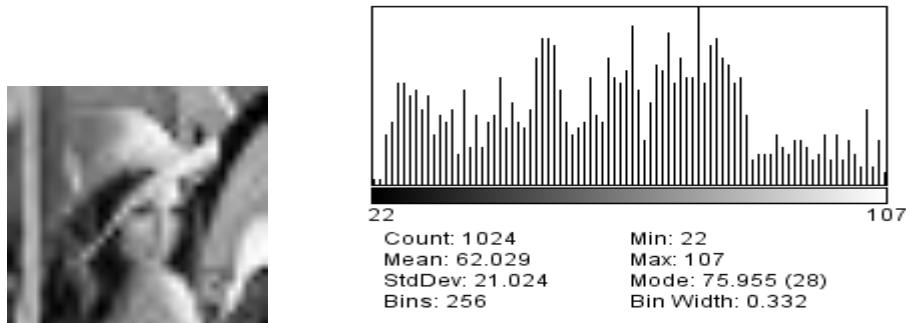


Figure1: 32 x 32 pixels image and histogram from the quantized DC terms of each 8 x 8 pixels block

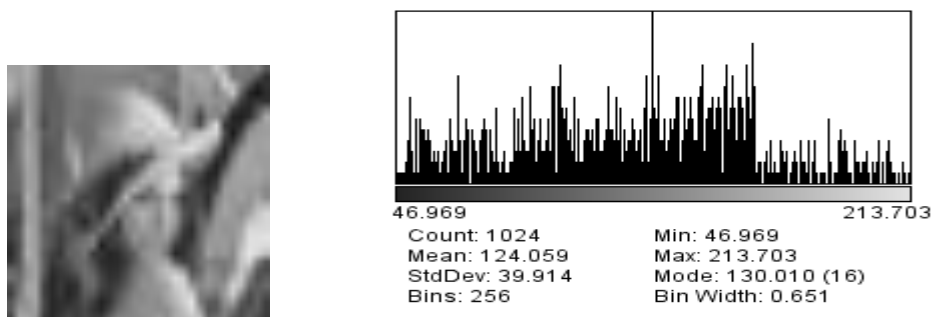


Figure2: Downsized (Average) Image

The downsized image has higher mean, min and max values. This means that its pixel values are higher than the pixel values of the image created from the quantized DC terms of each 8\*8 pixels block.

2. Create a text file containing successive differences between quantized DC terms of each block  
Note that DC terms average each 8x8 pixels block and are therefore spatially correlated.

Text file containing successive differences between quantized DC terms of each block was created.

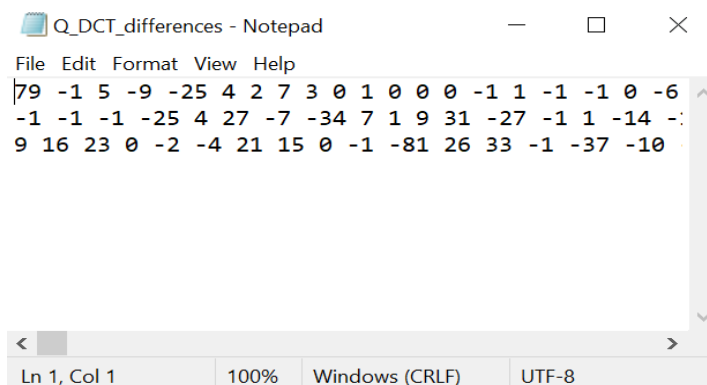


Figure 3: Text file of successive DCT differences

3. Read the delta encoded DC terms from the text file and reconstruct the low-definition image.

The delta encoded DC terms were read from the text file and low-definition image was reconstructed as shown in figure 4.



Figure4: Reconstruction of low definition image.

The Delta encoding of the DC coefficients help to achieve a compression ratio of 1.4. The reconstructed image is the same as the image in figure 1

#### Run-length encoding (RLE) of AC coefficients

1. Imagine a method for ordering and coding groups of quantized AC terms with run lengths  
Each run length may represent a contiguous sequence of either zeros or arbitrary values

Run-length encoding helps to achieve lossless data compression. Runs of consecutive identical data values are stored as a single data value and count.

Run-length encoding can be achieved with different methods such as:

- Encoding along the horizontal axis
- Encoding along the vertical axis
- Encoding 4 by 4-pixel tiles
- Zig zag encoding

Zig zag encoding was implemented because it has the benefit of hitting long runs of zeros.

2. Update your encode function for printing a sequence of run lengths from coefficients.

The encode function was updated to print sequence of run lengths from coefficients.

An image of 256\*256 pixels was divided into blocks of 8\*8-pixel values. The first element of each 8\*8-pixel block is the DC term. The actual AC terms:  $(256*256) - (32*32) = 32*32*63 = 64512$ .

For each block of 8\*8 pixels, zigzag RL ordering was performed, then the 1<sup>st</sup> term (DC value) was discarded leaving only the 63 AC terms for each 8\*8-pixel block. RL encoding was performed on the AC terms.

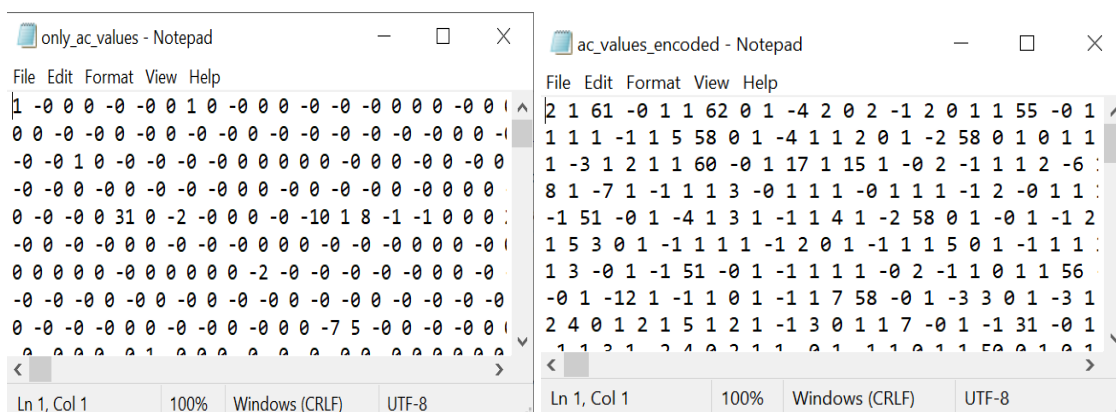


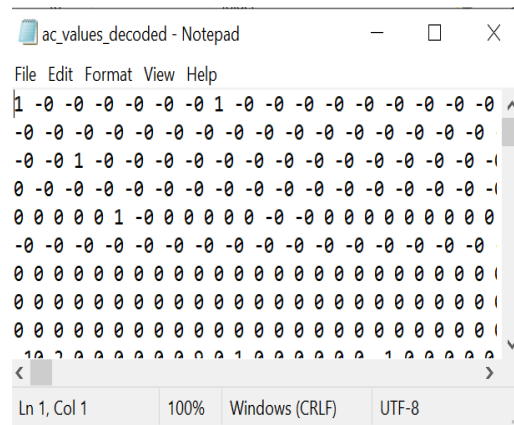
Figure 5: AC terms

Encoded AC Terms

3. Update your decode function for reconstructing AC coefficients from a sequence of run lengths.

The decode function was updated to reconstruct AC coefficients from a sequence of run lengths.

The encoded AC terms were passed as parameter to the decoder. The AC values were decoded. For each block of 8\*8 pixels, the first term (DC values) was assigned 0 to achieve accurate inverse Run Length ordering. This DC values were removed after inverse Run Length ordering. The result of the inverse Run Length ordering is the same as the AC values. This encoding of AC terms using Run Length encoding help to achieve a compression ratio of about 3.5.



```
ac_values_decoded - Notepad
File Edit Format View Help
1 -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 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 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 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 0 0 0 0 0 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
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Figure 6: Decoded AC values