Audio & Image Compression

Bashar Beshoti (207370248)

April 20, 2024

Course Information

– **Course Title:** Audio & Image Compression

– **Course Code:** 203.3880

– **Assignment :** Final assignment - Image

– **Due Date :** 04.04.2024

1 Block Truncation Coding (BTC)

$$\boxed{\hat{x} = \frac{1}{n^2}\sum_i x_i, \sigma = \sqrt{(x^2 - \hat{x}^2)}}$$

With this method, the average and standard deviation are found for each block: Transfer (to the receiver) for each pixel '1' or '0' respectively, '1' if it is higher than the average or '0' if it is lower than average.
ONLY THE '0' AND '1' MATRIX IS TRANSFERRED TO THE RECEIVER.
In reconstruction (at the receiver), in order to preserve the above two statistical moments, each pixel is "reconstructed" according to the formula:

$$x^- = \hat{x} - \sigma \cdot \sqrt{\frac{n_{above}}{n_{under}}}, x^+ = \hat{x} + \sigma \cdot \sqrt{\frac{n_{above}}{n_{under}}}$$

That is, pixels for which '0' was transferred get an average value minus the standard deviation multiplied by the constant in the root, and those who received a value of '1' receive the average value plus the standard deviation multiplied by a constant inside the root (note that these are inverse constants!).

- n_{above} - The number of pixels that received a value of '1'
- n_{under} - The number of pixels that received a value of '0'

1.1 A block of pixels (0-255) with a size of 4x4 should be produced where the average of the block will be two digits the last of your ID number and the standard deviation will be the mean divided by the third digit from the end.

If a special problem arises - for example if the dividing digit is 0 can be used in the next book after it, etc. It is of course possible to use the assistance of various AI systems

Answer : My ID is 207370248. Then, the average is 48 and the standard deviation is 24. and now I'm going to create a matrix 4x4 fulfill the requirements above:
MODIFIED

$$\text{Block} = \begin{pmatrix} 41 & 68 & 28 & 38 \\ 22 & 39 & 58 & 40 \\ 26 & 127 & 43 & 45 \\ 56 & 47 & 60 & 30 \end{pmatrix}$$

$$\sigma = \sqrt{(x^2 - \hat{x}^2)} = 23.92$$

$$\hat{x} = \frac{1}{n^2}\sum_i x_i = 48$$

1.2 Show the generated block, and perform the process of finding a '0' or '1' representation for each pixel

Answer : *MODIFIED* Upon definition written in question's context. We can define the following function.

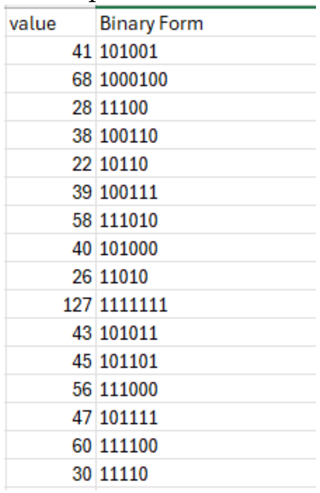
$$f_{(x,y)} = \begin{cases} 1 & x \in [0,3], y \in [0,3], Block(x,y) <= Average \\ 0 & x \in [0,3], y \in [0,3], Block(x,y) > Average \end{cases}$$

Therefore,

$$\text{Binary Block} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

But if I'm mistakenly misunderstood the question and required the binary format of each value. Here it is :

Figure 1: Table of pixels from decimal to binary



value	Binary Form
41	101001
68	1000100
28	11100
38	100110
22	10110
39	100111
58	111010
40	101000
26	11010
127	1111111
43	101011
45	101101
56	111000
47	101111
60	111100
30	11110

$$\text{Binary Format Block} = \begin{pmatrix} 101001 & 1000100 & 11100 & 100110 \\ 10110 & 100111 & 111010 & 101000 \\ 11010 & 1111111 & 101011 & 101101 \\ 111000 & 101111 & 111100 & 11110 \end{pmatrix}_2$$

1.3 Restore the block based on the process shown above (finding x^-, x^+) and perform Quantization by Round to the nearest whole number.

Answer : *MODIFIED* - still barely effect Based on the Binary Block Matrix, the number of 0's are 11 and number of 1's are 5. So, let's calculate both of x^+, x^- from the equations mentioned above.

$$x^{-} = \hat{x} - \sigma \cdot \sqrt{\frac{n_{above}}{n_{under}}} = 48 + 23.92 \cdot \sqrt{\frac{5}{9}} \approx 66$$

$$x^{-} = \hat{x} - \sigma \cdot \sqrt{\frac{n_{above}}{n_{under}}} = 48 - 23.92 \cdot \sqrt{\frac{5}{9}} \approx 30$$

$$\text{Quantized Block} = \begin{pmatrix} 30 & 66 & 30 & 30 \\ 30 & 30 & 66 & 30 \\ 30 & 66 & 30 & 30 \\ 66 & 30 & 66 & 30 \end{pmatrix}$$

1.4 The statistical moments for the created block must be calculated - were they really saved?

Answer : *MODIFIED*

if i understand correctly from the response in questions forum i asked, statistical moments is average. so to reshape the question be like: Calculate the average for the created Block which is :

$$average' = \frac{1}{n^2} \sum_{0 \leq i,j \leq} Quanatized - Block(i,j) = \frac{1}{4^2}(11 \cdot 30 + 5 \cdot 66) = 41.25$$

$$\sigma' = \sqrt{(x^2 - \hat{x}^2)} = 16.686$$

1.5 Compute the MSE between the original block and the reconstructed block.

Answer : *MODIFIED*

$$MSE = \frac{1}{n^2} \sum (P_{source} - P(Quantized))^2 = 316.125$$

1.6 Calculate the compression ratio between the original block and the compressed block.

Answer :

$$Compression - Ratio = \frac{Original - Data - Size}{Compressed - Data - Size}$$

Since each pixel value from Original Block is a value in range $[0, 255]$. then it is represented by 8 bits for each value. the total size of original data block is block size times pixel size.

$$Original - Data - Size = (Block - Size) \times (Pixel - Size) = 16 * 8bits = 144bits$$

$$Compressed - Data - Size = (Block - Size) \times (Pixel - Size) = 16 * 6bits = 96bits$$

$$Compression - Ratio = \frac{Original - Data - Size}{Compressed - Data - Size} = \frac{144}{96} = \frac{4}{3}$$

	ORIGINAL-COMPRESSED
SNR	9.59
PSNR	17.077

Table 1: SNR,PSNR Calculation

1.7 Infer conclusions quality of compression and recovery for the original block.

Answer :

SNR,PSNR Calculation: We can infer that :

1. Compression Ratio: The compression ratio gives an idea of how much the data has been compressed. A higher compression ratio means more compression, but it may also lead to more loss of information. After achieving a compression ratio of $\frac{4}{3} : 1$, meaning the compressed block requires $\frac{4}{3}$ times less data compared to the original block.
2. Mean Squared Error (MSE): The MSE between the original block and the reconstructed block gives a measure of the error introduced by the compression and reconstruction process. The MSE in this context equals to 534.125. A lower MSE means less error and better quality of reconstruction.
3. Signal-to-Noise Ratio (SNR) and Peak Signal-to-Noise Ratio (PSNR): These metrics give a measure of the quality of the reconstructed block compared to the original block. A higher SNR or PSNR means better quality of reconstruction. Based on Table above, we can induce that there the signal power is bigger than the noise especially since both PSNR & SNR are positive values.

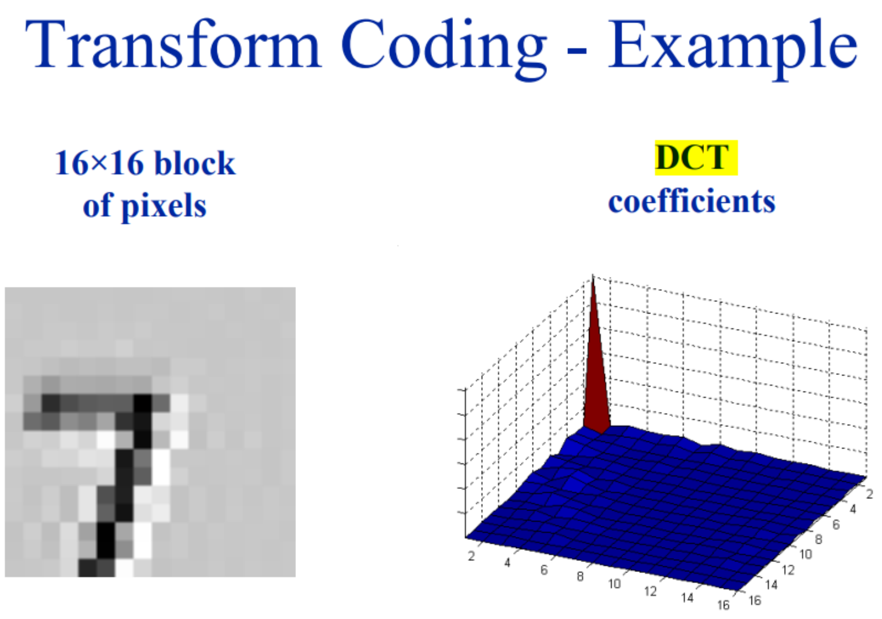
1.8 Bonus 5 points:

Claim: There is a certain mathematical relationship between this method and JPEG's method - do you think this claim is true correctly? You have to reason!

Answer : Yes, the claim that there is a certain mathematical relationship between Block Truncation Coding (BTC) and JPEG's method is **correct**.

Quantization Both BTC and JPEG rely on a fundamental mathematical concept called *Quantization* to achieve image compression. This process aims to reduce the number of bits needed to represent the image data by discarding or scaling down certain information.

BTC performs a simpler form of quantization during image restoration. It uses the average ($\hat{x}$) and standard deviation (σ) of each block to restore pixel values. JPEG employs a more sophisticated form of quantization based on the Discrete Cosine Transform (DCT) as we learned in lectures. This transform converts the image data from the spatial domain (pixel intensity values) to the frequency domain.



Transform Coding - Example

16×16 block of pixels

DCT coefficients

Differences:

– Time Complexity: BTC's quantization is simpler and less computationally expensive compared to JPEG's DCT-based approach.

– Concept : BTC quantizes based on average, standard deviation, and category proportions. JPEG targets specific frequencies based on human visual perception for better compression efficiency.

Similarities:

– Reducing the number of bits needed to represent the image data.

– Introducing controlled information loss for compression.

2 Code

Implementation:

```
import numpy as np

original_block = np.array([[40, 72, 24, 36],
                           [16, 37, 60, 38],
                           [22, 144, 42, 44],
                           [58, 47, 62, 26]])

signal_power = np.var(original_block)

reconstructed_block = np.ones(original_block.shape) * 37
reconstructed_block[original_block > 48] = 59  # Replace elements
   less than or equal to average with 59

MSE = np.sum(original_block - reconstructed_block) ** 2

noise_power = np.mean((original_block - reconstructed_block) ** 2) #
   => 534.125
# noise_power = np.var(original_block - reconstructed_block) =>
   517.109375

snr = 10 * np.log10(signal_power / noise_power)

max_pixel_value = 255  # Assuming 8-bit grayscale image
psnr = 10 * np.log10(max_pixel_value**2 / noise_power)

print('Original matrix:')
print(original_block)

print('reconstructed_block matrix:')
print(reconstructed_block)

print('diff mat')
print(np.abs(original_block - reconstructed_block))


print('MSE IS = ' + str(MSE))


print('noise_power IS = ' + str(noise_power))

print('snr IS = ' + str(snr))

print('psnr IS = ' + str(psnr))
```

Output

```
Original matrix:
[[ 40  72  24  36]
 [ 16  37  60  38]
 [ 22 144  42  44]
 [ 58  47  62  26]]
reconstructed_block matrix:
[[37. 59. 37. 37.]
 [37. 37. 59. 37.]
 [37. 59. 37. 37.]
 [59. 37. 59. 37.]]
diff mat
[[ 3. 13. 13.  1.]
 [21.  0.  1.  1.]
 [15. 85.  5.  7.]
 [ 1. 10.  3. 11.]]
MSE IS = 4356.0
noise_power IS = 534.125
snr IS = 1.977336545354818
psnr IS = 20.854374550426577

=== Code Execution Successful ===
```