EN3551 Assignment 2

Application of 2D-DCT for Image Compression

Due on or before: Monday, October 21, 2024

---

1 Introduction and General Guidelines

This programming assignment is concerned with the application of 2D-DCT to compress digital images. Section 2 contains a general overview of the theory associated with this assignment, followed by Section 3 and Section 4, which contains the specific tasks for the assignment.

Submit a report containing your solutions, diagrams, and explanations for each task. Briefly explain/justify how you obtained each of your answers. This will help us determine your understanding of the problem whether or not you got the correct answer. Moreover, in the event of an incorrect answer, we can still try to give you partial credit based on the explanation you provide. You need to append your MATLAB codes as an appendix of this report.

2 Theoretical Background

2.1 Preliminaries

We can write the 1D-DCT (type 2) of a 1D signal $x[n]$ as [1];

$$C(k) = \sqrt{\frac{2}{N}}\alpha(k)\sum_{n=0}^{N-1} x[n]\cos\left(\frac{(2n+1)k\pi}{2N}\right), \tag{1}$$

where

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{2}}; & k = 0 \\ 1; & 1 \leq k \leq N-1. \end{cases} \tag{2}$$

The 2D-DCT of a 2D signal (represented as a 2-D array or a matrix) $x[m,n]$ of size $N \times N$ is defined as;

$$C(i,k) = \frac{2\alpha(i)\alpha(k)}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n]\cos\left(\frac{(2m+1)i\pi}{2N}\right)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad 0 \leq i,k \leq N-1, \tag{3}$$

where $\alpha(l)$, $l = i$ or k,

$$\alpha(l) = \begin{cases} \frac{1}{\sqrt{2}}; & l = 0 \\ 1; & 1 \leq l \leq N-1 \end{cases} \tag{4}$$

Furthermore, the 2-D inverse DCT (IDCT) is given by,

$$x[m,n] = \frac{2}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} \alpha(i)\alpha(k) C(i,k) \cos\left(\frac{(2m+1)i\pi}{2N}\right) \cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad 0 \leq m,n \leq N-1. \tag{5}$$

Note that MATLAB commands `dct2` and `idct2` can be used to compute 2-D DCT and IDCT, respectively.

A frequently useful 2-D DCT employed for image compression deals with 8×8 data blocks, i.e.,

$$C(i,k) = \frac{\alpha(i)\alpha(k)}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} x[m,n] \cos\left(\frac{(2m+1)i\pi}{16}\right) \cos\left(\frac{(2n+1)k\pi}{16}\right) \quad 0 \leq i,k \leq 7. \tag{6}$$

Among the 64 DCT coefficients defined by (6), the coefficient with $(i,k) = (0,0)$ is found to be

$$C(0,0) = \frac{1}{8} \sum_{m=0}^{7} \sum_{n=0}^{7} x[m,n]. \tag{7}$$

This is proportional to the average value of the signal, making it the DC coefficient. The other coefficients, $C(i,k)$ with $(i,k) \neq (0,0)$, are AC coefficients. If one regards the resulting DCT block $C(i,k), 0 \leq i,k, \leq 7$ as an 8×8 matrix and denote it by $\mathbf{C}$, then (6) implies that

$$\mathbf{C} = \frac{1}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} x[m,n] \mathbf{B}_{m,n} \tag{8}$$

Here the $\mathbf{B}_{m,n}$'s (shown in Figure 1) are the 8×8 basis matrices formed only by cosine functions and depend only on indexes m, n. From (8) we see that the 2-D DCT of a data block $x[m,n]$ can be obtained as the weighted sum of basis matrices.

2.2 2D DCT is Energy Preserving and Orthogonal

The energy of a 2-D signal $x[m,n]$ is defined as the sum of the squares of all its components, i.e.,

$$\text{Energy of } \{x[m,n]\} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x^2[m,n].$$

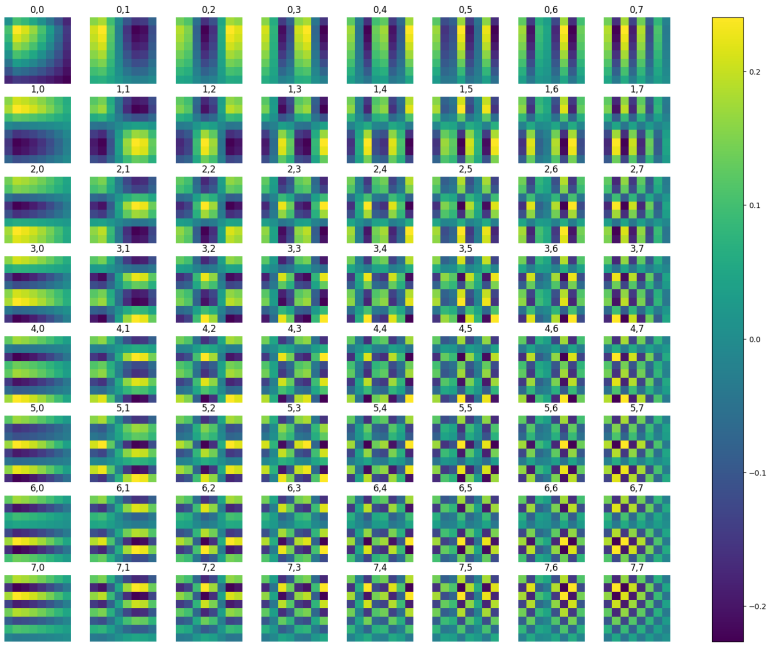


Figure 1: Basis matrices $\mathbf{B}_{m,n}$ for each m, n

Using Eqs. (5) and (6), energy of $x[m,n]$ can be computed as

$$
\begin{aligned}
\text{Energy of } \{x[m,n]\} &= \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x^2[m,n] \\
&= \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n].x[m,n] \\
&= \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n].\frac{2}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1}\alpha(i)\alpha(k)C(i,k)\cos\left(\frac{(2m+1)i\pi}{2N}\right)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \\
&= \sum_{i=0}^{N-1}\sum_{k=0}^{N-1} C(i,k).\frac{2\alpha(i)\alpha(k)}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n]\cos\left(\frac{(2m+1)i\pi}{2N}\right)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \\
&= \sum_{i=0}^{N-1}\sum_{k=0}^{N-1} C(i,k).C(i,k) \\
&= \sum_{i=0}^{N-1}\sum_{k=0}^{N-1} C^2(i,k) \\
&= \text{Energy of } \{C[i,k]\}.
\end{aligned}
$$

This means the 2-D DCT does not change the signal's energy. It also can be shown that the 2-D DCT is an *orthogonal transform* in the sense that the "inner products" of the basis matrices

$\{\mathbf{B_{m,n}}\}$ (see Eq. 8) satisfy,

$$\text{sum}\left(\text{sum}\left(\mathbf{B_{m,n}} \circ \mathbf{B_{p,q}}\right)\right) = \begin{cases} 16 & \text{if } (m,n) = (p,q) \\ 0 & \text{otherwise,} \end{cases}$$

where $\circ$ denotes the pointwise multiplication.

2.3 2D DCT is Separable: A Cascade of Two 1D DCTs

From Eq. (3), it is observed that the quantities involving index i and those involving index k are separate from each other. Taking this as an advantage, we can write Eq. (3) as,

$$C(i,k) = \underbrace{\sqrt{\frac{2}{N}}\alpha(i) \sum_{m=0}^{N-1} \cos\left(\frac{(2m+1)i\pi}{2N}\right) \bullet \underbrace{\left(\sqrt{\frac{2}{N}}\alpha(k) \sum_{n=0}^{N-1} x[m,n] \cos\left(\frac{(2n+1)k\pi}{2N}\right)\right)}_{\text{1-D DCT for } \{x[m,n]\} \text{ with } m \text{ fixed, denote this DCT as } \hat{C}(m,k)}}_{\text{1-D DCT for signal } \hat{C}(m,k) \text{ with index } k \text{ fixed}} . \tag{9}$$

In other words, the 2-D DCT can be formed in two 1-D steps: perform 1-D DCT row-by-row for the input 2-D signal $x[m,n]$, resulting in an intermediate 2-D signal denoted by $\hat{C}(m,k)$; then perform 1-D DCT column-by-column for $\hat{C}(m,k)$ yielding the 2-D DCT $C(i,k)$.

If we regard $C(i,k)$ as an $N \times N$ matrix denoted by $\mathbf{C_N}$, Eq. (9) implies that $\mathbf{C_N} = \mathbf{D_N X D_N^T}$ where $\mathbf{X} = x[m,n]$ and,

$$\mathbf{D_N} = \sqrt{\frac{2}{N}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\left(\frac{\pi}{2N}\right) & \cos\left(\frac{3\pi}{2N}\right) & \cdots & \cos\left(\frac{(2N-1)\pi}{2N}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \cos\left(\frac{(N-1)\pi}{2N}\right) & \cos\left(\frac{3(N-1)\pi}{2N}\right) & \cdots & \cos\left(\frac{(2N-1)(N-1)\pi}{2N}\right) \end{bmatrix}.$$

In particular, for an 8×8 data block $\mathbf{X_{block}}$, the 2-D DCT can be calculated as,

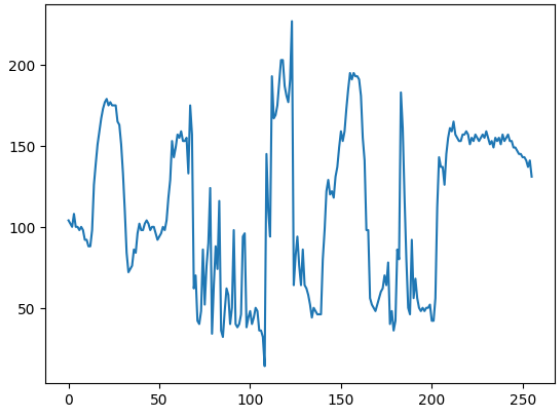
$$\mathbf{C_8} = \mathbf{D_8 X_{block} D_8^T}$$

where

$$\mathbf{D_8} = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.0913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & 0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

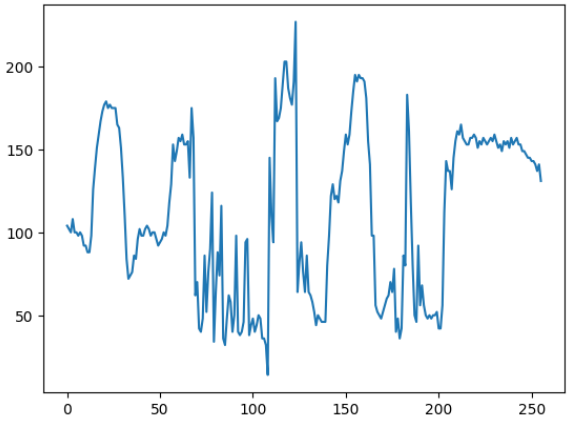
Notice that the matrix $\mathbf{D_N}$ is orthogonal, namely it satisfies $\mathbf{D_N D_N^T} = \mathbf{D_N^T D_N} = \mathbf{I_N}$.

(a) Image Lena

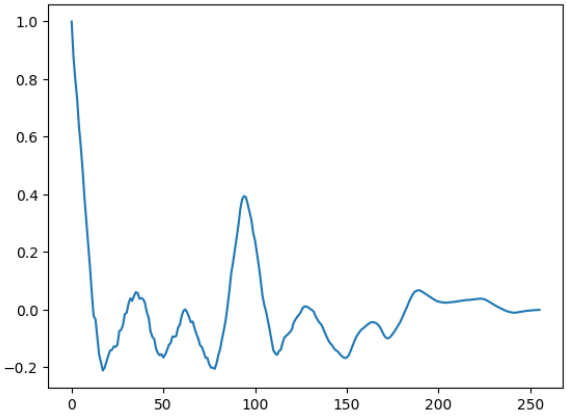


(b) 8-bit gray level of the 128th row of Lena

Figure 2



(a) 8-bit gray level of the 128th row of Lena



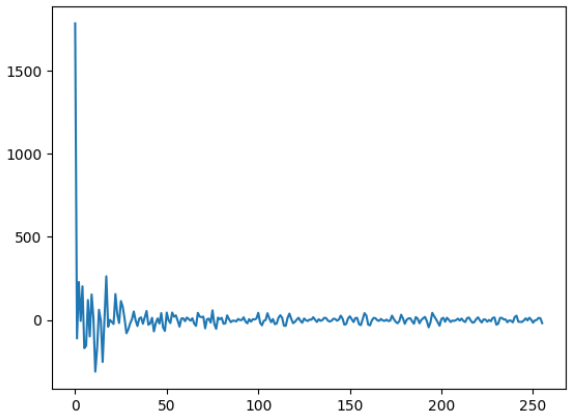
(b) Normalized autocorrelation of the signal

Figure 3

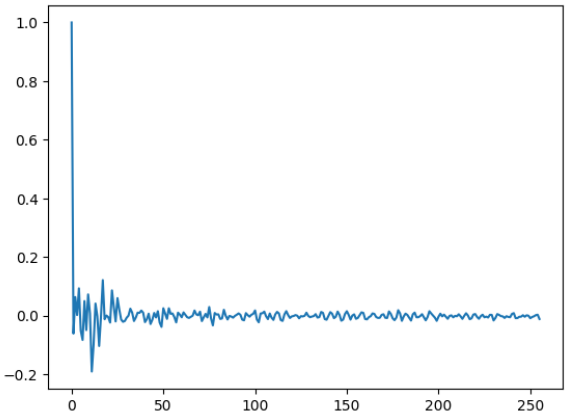
2.4 De-correlation Ability

Image compression is made possible by exploiting the redundancy present within the image, and the compression process fundamentally involves eliminating this redundancy from the image. The presence of correlation in a signal indicates that it can be further compressed. For illustration purposes, we examine the line in the 128th row of the 256×256 grey scale test image Lena (see Figure 2a), which is shown in Figure 2b. The 128th row of the image and its autocorrelation (measure of how the signal is related to its shifted versions) are shown in Figures 3a and 3b, respectively. It is evident from Figure 3b that the signal contains a great deal of redundancy because the correlation between the signal and its neighbouring samples is strong and decays rather slowly with respect to the shift.

Now the 1-D DCT is applied to the above signal. The 1-D DCT $\hat{C}(128, k)$ and its autocorrelation are shown in Figures 4a and 4b, respectively. This clearly demonstrates the *decorrelation*



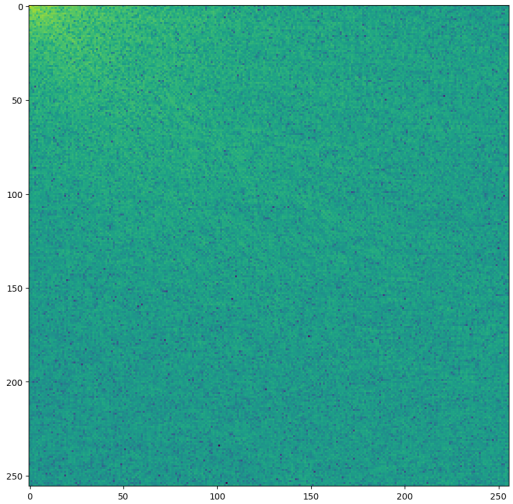
(a) 1-D DCT of the 128th row of Lena



(b) Normalized autocorrelation of DCT sequence

Figure 4

(a) Image of Lena



(b) 2D DCT on Lena

Figure 5

ability of the DCT.

2.5 Energy Compaction Ability

The energy content of the signal, specifically the 128th row of Lena, measures at 2.6909×10^6. Since DCT conserves energy, its application to this signal also results in an energy of 2.6909×10^6. Upon examination of Figure 4a, it becomes evident that the majority of the energy in the DCT sequence is concentrated within a limited number of initial samples. In fact, the energy contained in the first 15 DCT samples totals 2.4786×10^6, accounting for approximately 92.11% of the overall energy. It should be stressed that energy compaction is not just a coincidence that occurs in this particular case, but it is an intrinsic property of the DCT.

Figures 5a and 5b show image Lena and its 2-D DCT, which demonstrates the energy compaction property of the 2-D DCT. The lighter spots in Figure 5b represent samples of large values while the black dots represent samples of small values.

3 Procedure

3.1 2-D DCT - Application to Image Compression

Suppose the image to be compressed has size $M \times N$, where M and N are multiples of 8, and represeted with 8 bits. The image is first divided into 8×8 blocks. The steps described below are applied to each of these blocks. Let $\mathbf{B}$ be a representative 8×8 data block.

Step 1: Level-Off and 2-D DCT

Since DCT gives a lower DC coefficient when pixel values are in the range -128 to 127 rather than in the range 0 to 255, $\mathbf{B}$ is levelled off by subtracting 128 from each entry. For illustration purposes, we take the (16,18)-th block of the 256×256 image Lena as matrix $\mathbf{B}$.

$$\mathbf{B} = \begin{bmatrix} 125 & 134 & 137 & 139 & 138 & 138 & 141 & 142 \\ 113 & 119 & 126 & 134 & 139 & 141 & 144 & 149 \\ 80 & 95 & 103 & 106 & 114 & 127 & 141 & 147 \\ 63 & 65 & 53 & 62 & 75 & 86 & 108 & 130 \\ 93 & 80 & 60 & 33 & 35 & 35 & 52 & 69 \\ 126 & 108 & 88 & 74 & 53 & 45 & 35 & 32 \\ 130 & 116 & 90 & 96 & 62 & 63 & 55 & 49 \\ 115 & 80 & 61 & 65 & 68 & 88 & 68 & 75 \end{bmatrix}$$

The modified data block after subtracting 128 from $\mathbf{B}$ is given by,

$$\tilde{\mathbf{B}} = \begin{bmatrix} -3 & 6 & 9 & 11 & 10 & 10 & 13 & 14 \\ -15 & -9 & -2 & 6 & 11 & 13 & 16 & 21 \\ -48 & -33 & -25 & -22 & -14 & -1 & 13 & 19 \\ -65 & -63 & -75 & -66 & -53 & -42 & -20 & 2 \\ -35 & -48 & -68 & -95 & -93 & -93 & -76 & -59 \\ -2 & -20 & -40 & -54 & -75 & -83 & -93 & -96 \\ 2 & -12 & -38 & -32 & -66 & -65 & -73 & -79 \\ -13 & -48 & -67 & -63 & -60 & -40 & -60 & -53 \end{bmatrix}$$

Applying 2-D DCT to $\tilde{\mathbf{B}}$, we obtain,

$$\mathbf{C} = \mathbf{T_8}\tilde{\mathbf{B}}\mathbf{T_8^T}$$
$$= \begin{bmatrix} -272.3750 & 17.1771 & 46.7784 & 4.2270 & 6.1250 & -0.5799 & 0.2421 & -8.4813 \\ 182.5146 & -109.5361 & -28.0398 & -25.1231 & -8.9567 & -7.0036 & -6.1703 & 10.3445 \\ 117.4897 & 19.1429 & -30.8911 & 15.7065 & 1.7309 & 6.7882 & 5.4116 & -8.5788 \\ -23.4612 & 97.7162 & -0.0872 & -7.0795 & -1.0461 & 2.6980 & -2.9351 & 2.5446 \\ -48.3750 & -35.2958 & 27.0407 & 6.4060 & 4.1250 & -7.3615 & 3.5470 & 4.3781 \\ 15.8386 & -7.1745 & -7.9234 & -6.7518 & -0.5166 & 3.8019 & -8.3849 & -7.4654 \\ 0.4477 & 1.3348 & -5.5884 & 3.4787 & -4.6404 & -0.7361 & 4.6411 & 2.6707 \\ -3.6673 & 9.8948 & 6.2377 & -7.3148 & -7.0342 & 1.0065 & -0.6727 & 2.3138 \end{bmatrix}$$

Alternatively, one may apply MATLAB function `dct2` to $\tilde{\mathbf{B}}$ to obtain $\mathbf{C}$.

Step 2: Quantization

A critical step in image compression is quantization and coding of the DCT coefficients in $\mathbf{C}$. The quantization may be achieved by converting matrix $\mathbf{C} = \{c_{i,j}\}$ to matrix $\mathbf{S} = \{s_{i,j}\}$ determined by,

$$s_{i,j} = \text{round}\left(\frac{c_{i,j}}{q_{i,j}}\right) \tag{10}$$

where $\mathbf{Q} = \{q_{ij}\}$ is a *quantization matrix* that may be selected according to a desired quality level of the compression. In the JPEG standard, subjective experiments involving the human visual system have produced their own quantization matrices. For example, with a quality level 50 (on a 1-to-100 scale) we have,

$$\mathbf{Q_{50}} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantization matrices of other quality levels can be obtained by multiplying the above standard quantization matrix by a scaling factor τ where,

$$\tau = \begin{cases} \dfrac{100 - \text{quality level}}{50}, & \text{quality level} > 50 \\ \dfrac{50}{\text{quality level}}, & \text{quality level} < 50 \end{cases}$$

The scaled quantization matrix is then rounded and clipped to integer values between 0 and 255. With $\mathbf{C}$ obtained in Step 1 and $\mathbf{Q} = \mathbf{Q_{50}}$, the matrix $\mathbf{S}$ in Eq. 10 is computed as

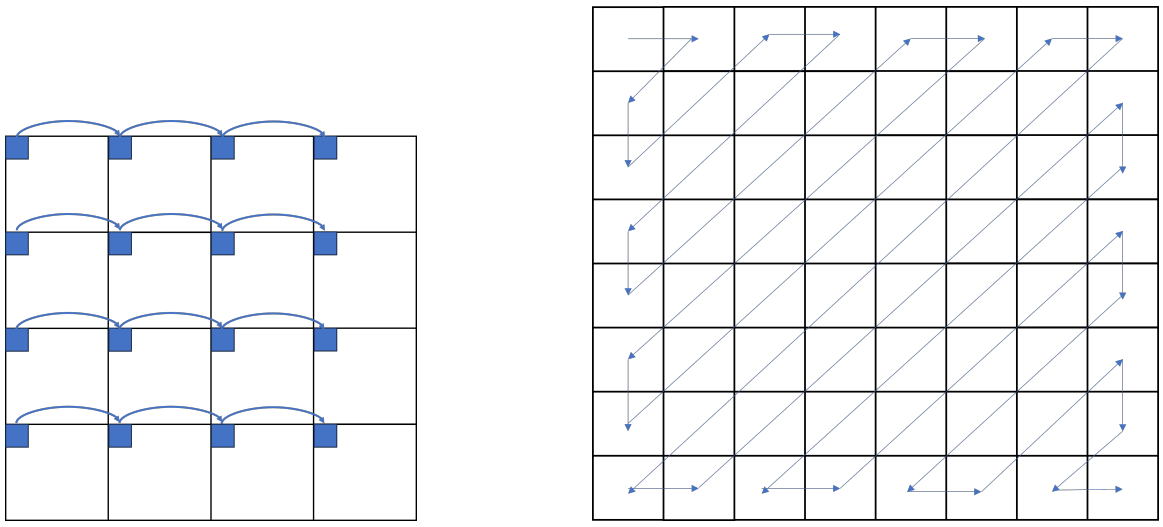
$$\mathbf{S} = \begin{bmatrix} -17 & 2 & 5 & 0 & 0 & 0 & 0 & 0 \\ 15 & -9 & -2 & -1 & 0 & 0 & 0 & 0 \\ 8 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ -2 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Evidently, if a quantization matrix with a reduced quality level is used, $\mathbf{S}$ will contain more zeros, thus a higher compression can be achieved at the cost of reduced image quality.

Step 3: Coding

The quantized matrix $\mathbf{S}$ is then converted by an encoder to a stream of binary data like $\{1001101\ldots\}$. Detailed coverage of the coding process is beyond the scope of this experiment. Instead, we mention the following.

- The DC coefficients of the 8×8 blocks (see Figure 6a) are encoded by a differential pulse-code modulation (DPCM) coding (also referred to as *DC prediction*).
- Because the location of the retained coefficients vary from block to block, the quantized AC coefficients are zigzag scanned (see Figure 6b) and ordered into (run, level) pairs, where "level" means the value of a nonzero coefficient, and "run" means the number of zero coefficients preceding it.
- These (run, level) pairs are entropy coded, that is, longer codes for less frequent pairs and vice versa.



(a) (b)

Figure 6

Step 4: Decompression

This is a step to be carried out at the receiver to reconstruct the image. It consists of the following three simple operations.

- Pointwise multiplication of matrix $\mathbf{S}$ with the quantization matrix $\mathbf{Q}$ to get an image block $\mathbf{R} = \mathbf{Q} \odot \mathbf{S}$. In our example, the $\mathbf{Q}$ matrix was taken to be $\mathbf{Q_{50}}$ that yields,

$$\mathbf{R} = \begin{bmatrix} -272 & 22 & 50 & 0 & 0 & 0 & 0 & 0 \\ 180 & -108 & -28 & -19 & 0 & 0 & 0 & 0 \\ 112 & 13 & -32 & 24 & 0 & 0 & 0 & 0 \\ -28 & 102 & 0 & 0 & 0 & 0 & 0 & 0 \\ -54 & -44 & 37 & 0 & 0 & 0 & 0 & 0 \\ 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

- Apply 2-D inverse DCT to matrix $\mathbf{R}$. Note that this can be done by using the MATLAB command `idct2`. Alternatively, it can be done using $\mathbf{D_8^T R D_8}$. This gives

$$\mathbf{E} = \mathbf{D_8^T R D_8}$$
$$= \begin{bmatrix} 2.7181 & 2.1228 & 2.2693 & 4.2600 & 7.6484 & 10.8548 & 12.7203 & 13.3406 \\ -18.1034 & -10.6760 & -0.3337 & 7.7581 & 11.5839 & 12.5496 & 12.8140 & 13.1153 \\ -43.2294 & -34.5413 & -23.6264 & -15.8552 & -9.7946 & -0.4317 & 12.7343 & 23.0141 \\ -57.9809 & -58.9924 & -63.0283 & -67.8737 & -64.7907 & -46.4675 & -17.7381 & 4.4189 \\ -40.7650 & -50.9610 & -69.2674 & -88.8143 & -98.8380 & -92.2600 & -73.6202 & -57.6914 \\ 0.1073 & -12.7805 & -34.0432 & -57.0831 & -76.3805 & -89.4098 & -96.5154 & -99.3926 \\ 6.4904 & -8.6805 & -28.5866 & -43.2078 & -52.0062 & -61.1538 & -73.3190 & -83.0164 \\ -25.3942 & -44.5619 & -65.1582 & -70.4424 & -60.8203 & -50.0267 & -47.8854 & -50.9752 \end{bmatrix}.$$

- Finally, the effect of the "level-off" operation in Step 1 is taken into account by adding 128 to the entries of the matrix E obtained above. This yields the reconstructed image block that mimics the image block **B** as,

$$\mathbf{B} = \mathbf{E} + 128$$

$$= \begin{bmatrix} 130.7181 & 130.1228 & 130.2693 & 132.2600 & 135.6484 & 138.8548 & 140.7203 & 141.3406 \\ 109.8966 & 117.3240 & 127.6663 & 135.7581 & 139.5839 & 140.5496 & 140.8140 & 141.1153 \\ 84.7706 & 93.4587 & 104.3736 & 112.1448 & 117.2054 & 127.5683 & 139.7343 & 151.0141 \\ 70.0191 & 69.0076 & 64.9717 & 60.1263 & 63.2093 & 81.5325 & 110.2619 & 132.4189 \\ 87.2350 & 77.0390 & 58.7326 & 39.1857 & 29.1620 & 35.7400 & 54.3798 & 70.3086 \\ 128.1073 & 115.2195 & 93.9568 & 70.9069 & 51.6195 & 38.5902 & 31.4846 & 28.6074 \\ 134.4904 & 119.3195 & 99.4134 & 84.7922 & 75.9938 & 66.8462 & 54.6810 & 44.9836 \\ 102.6058 & 83.4381 & 62.8418 & 57.5576 & 67.1797 & 77.9733 & 80.1146 & 77.0248 \end{bmatrix}$$

On comparing the block $\tilde{\mathbf{B}}$ with block **B** (given in Step 1), we see $\tilde{\mathbf{B}}$ approximates **B** quite well.

For an image of size $M \times N$, the number of blocks is $M/8 \times N/8$, and the steps described above is applied to each of these blocks for a selected quality level. We count the number of zeros contained in the **S** matrices (see Step 2) in order to compute the compression factor because the total number of zeros indicates how the image has been compressed. As expected, the use of reduced quality levels leads to a higher percentage of total zeros at the cost of a degraded image. Figures 7a, 7b, 7c and 7d show the results of the DCT-based compression method that was applied to image Lena of size 256×256 with 3 quality levels (50, 30, and 10).

3.2 Peak Signal to Noise Ratio (PSNR)

The PSNR is defined as;

$$PSNR = 20 \log_{10} \left(\frac{\psi_{\max}}{\sigma_e} \right) \tag{11}$$

where $\psi_{\max}$ denotes the maximum light intensity of the original image and and σ_e^2 denotes the mean squared error. For an 8-bit digital image, $\psi_{\max} = 255$. Furthermore, σ_e^2 can be computed in MATLAB applying the command `mean` two times (over the two dimensions) on a squared error matrix. The error matrix is given by $I_{rec} - I_{original}$, where I_{rec} and $I_{original}$ are the reconstructed and original images.

3.3 Tasks

1. Download 3 images assigned to your index number. You have to select one more image of your choice, which is not in the provided set of images.

2. Compress each of these images by the DCT-based method described above with three quality levels. The quality levels are also assigned based on your index number. Here, step 3 (i.e., coding) is not required to perform, and you can proceed into the step 4 after the step 2.

(a) Original image

(b) Compressed, quality level = 50, zeros: 84.9%

(c) Compressed, quality level = 30, zeros: 88.8%

(d) Compressed, quality level = 10, zeros: 94.4%

Figure 7

3. Observe the results in terms of
   - percentage of zeros,
   - quality in terms of peak signal-to-noise ratio (see above), and
   - visual quality of the compressed images as related to the quality level.
4. Some images are more difficult to compress than others in the sense that the visual (subjective) quality cannot be maintained as easily as others for a given compression ratio. Observe the compression results obtained in the task (2) from this perspective and explain why.

3.4 Useful MATLAB commands

- plot: plots a function
- fft: calculates the DFT
- dct2: returns the two-dimensional discrete cosine transform of the input
- idct2: returns the two-dimensional inverse discrete cosine transform of the input
- load: load contents from a file into a MATLAB workspace.
- norm: returns the norm of a vector input.

4 References

1. Alan Oppenheim, Ronald Schafer, "Discrete Time Signal Processing", Prentice Hall Signal Processing (3rd edition), 2009.
2. S.A. Khayam, "The discrete cosine transform (DCT): Theory and Application", Report, Dept of ECE, Michigan State University, March 2003.
3. K. Cabeen and P. Gent, "Image compression and the discrete cosine transform", College of the Redwoods.
4. A related demo by Berkeley EECS: JPEG DCT Demo