The DCT and JPEG

4C8: Digital Image and Video Processing

Assistant Professor François Pitié

2024/2025

Department of Electronic & Electrical Engineering , Trinity College Dublin

adapted from original material written by Prof. Anil Kokaram.

Overview

- The DCT and its computation
- Basic elements of JPEG
- Quick overview of the JPEG entropy coding scheme.

JPEG

The main standard for image compression in current use is the JPEG (Joint Picture Experts Group) standard, devised and refined over the period 1985 to 1993. It is formally known as ISO Draft International Standard 10981-1 and ITU-T Recommendation T.81, and is described in depth in The JPEG Book by W B Pennebaker and J L Mitchell, Van Nostrand Reinhold 1993.

We shall briefly outline the baseline version of JPEG but first we consider its energy compression technique – the discrete cosine transform (DCT).

n-point transform

The 2-point Haar transform was introduced previously and it was shown that it can be easily inverted if the transform matrix T is orthonormal so that $T^{-1} = T^{\mathsf{T}}$.

If T is of size $n \times n$, where $n = 2^m$, then we may generate larger orthonormal matrices, which lead to definitions of larger transforms.

An n-point transform is defined as:

$$\begin{bmatrix} y(1) \\ \vdots \\ y(n) \end{bmatrix} = T \begin{bmatrix} x(1) \\ \vdots \\ x(n) \end{bmatrix} \qquad \text{where} \quad T = \begin{bmatrix} t_{11} & \cdots & t_{1n} \\ \vdots & & \vdots \\ t_{n1} & \cdots & t_{nn} \end{bmatrix}$$

n-point transform

For example, a 4-point orthonormal transform matrix that is equivalent to 2 levels of the Haar transform is:

$$T = \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{bmatrix}}_{\text{Haar level 2}} \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}}_{\text{Haar level 1}}$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

Similarly 3 and 4 level Haar transforms may be expressed using 8 and 16 point transform matrices respectively.

n-point transform

However for $n > 2$, there are better matrices than those based on the Haar transform, where better means with improved energy compression properties for typical images.

The Optimal Transform: The Karhunen Loève Transform

To preserve the energy, the transform must be orthogonal ($T^\top T = Id$):

$$\|y\|^2 = y^\top y = xT^\top Tx = x^\top x = \|x\|^2$$

There is an optimal orthogonal transform that achieves optimum energy concentration while minimising the entropy, it is called the Karhunen Loève Transform (KLT).

The optimal basis functions are the eigenvectors of the covariance matrix R_{xx} of the input signal:

$$R_{xx} = \mathbb{E}\left[(x - \mathbb{E}(x))\,(x - \mathbb{E}(x))^\top\right]$$

Karhunen Loève Transform

To compute the KLT basis, we need to collect a lot of signals x, build the covariance matrix R_{xx} and compute its eigenvectors.

The problem is that the KLT is not convenient to compute as it depends on the signal statistics. Also, both encoders and decoders must agree on the basis.

Karhunen Loève Transform vs DCT

In practice, performance of DCT is closest to the statistically optimal KLT based on a number of performance criteria such as variance distribution, energy packing efficiency, residual correlation, rate distortion, maximum reducible bits, etc.

Also, the performance hit is largely offset by a number of desirable characteristics for data compression such as data-independent basis functions that anybody can agree on and very fast implementation.

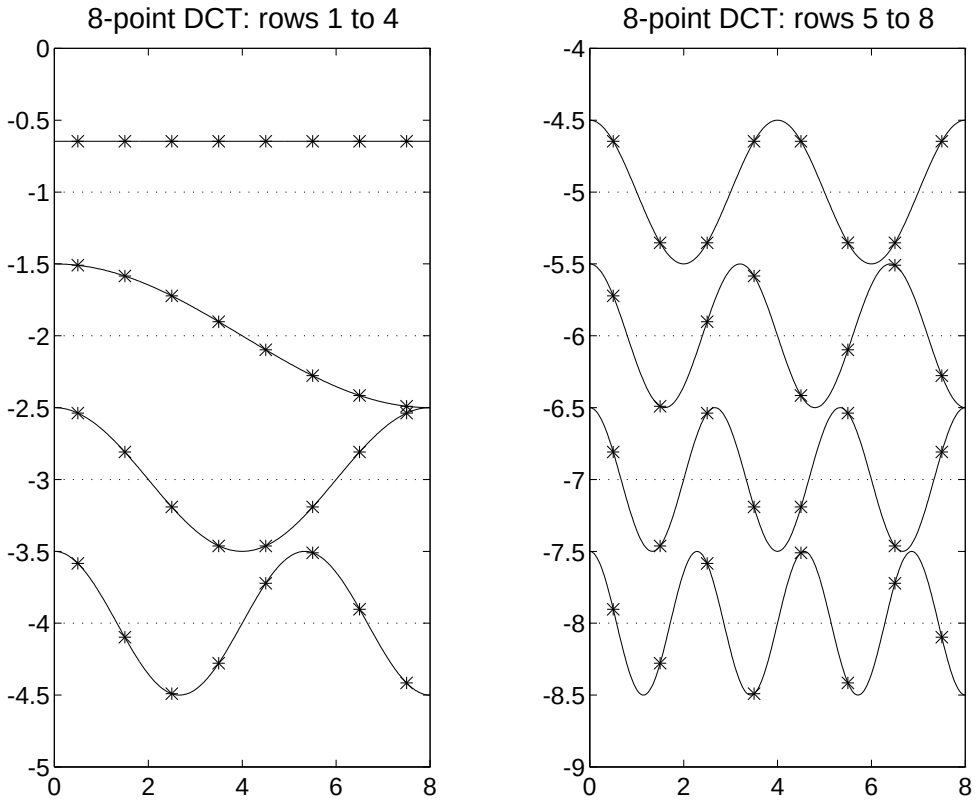
DCT

Discrete Cosine Transforms (DCTs) are simple to define and implement. The n rows of an n-point DCT matrix T are defined by:

$$t_{1i} = \sqrt{\frac{1}{n}} \quad \text{for } i \in [1:n]$$

$$t_{ki} = \sqrt{\frac{2}{n}} \cos\left(\frac{\pi(2i-1)(k-1)}{2n}\right) \quad \text{for } i \in [1:n], k \in [2:n]$$

DCT



The rows of T, known as basis functions, are plotted as asterisks (*) on the underlying continuous cosine functions. Only the amplitude scaling and the maximum frequency vary with the size n.

DCT

The 8-point DCT matrix ($n = 8$) is:

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

DCT

The DCT is closely related to the discrete Fourier transform (DFT). It represents the result of applying the $2n$-point DFT to a vector:

$$x_{2n} = c \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ x_n \\ \vdots \\ x_1 \end{bmatrix}$$ where c is some normalising constant

x_{2n} is symmetric about its centre and so the $2n$ Fourier coefficients are all purely real and symmetric about zero frequency. The n DCT coefficients are then the first n Fourier coefficients.

Fast DCT

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

The basic n-point DCT requires n^2 multiplications and $n(n-1)$ additions to calculate $y = Tx$ (64 mults and 56 adds for $n = 8$).

Fast DCT

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \left[\begin{array}{cccc|cccc} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

All the odd rows of T possess even symmetry about their centres and all the even rows possess odd symmetry.

Fast DCT

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \left[\begin{array}{cccc|cccc} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 \end{bmatrix} \begin{bmatrix} x_1 + x_8 \\ x_2 + x_7 \\ x_3 + x_6 \\ x_4 + x_5 \end{bmatrix}$$

$$\begin{bmatrix} y_2 \\ y_4 \\ y_6 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0.4904 & 0.4157 & 0.2778 & 0.0975 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 \end{bmatrix} \begin{bmatrix} x_1 - x_8 \\ x_2 - x_7 \\ x_3 - x_6 \\ x_4 - x_5 \end{bmatrix}$$

We can reduce the transform to 2 4×4 transforms. This reduces the computation to 8 add/subtract operations and 2×16 mults and 2×12 adds – almost halving the total computation load.

Fast DCT

Consider the green 4×4 matrix used to get $[y_1; y_3; y_5; y_7]$:

$$\begin{bmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 \end{bmatrix} \begin{bmatrix} x_1 + x_8 \\ x_2 + x_7 \\ x_3 + x_6 \\ x_4 + x_5 \end{bmatrix}$$

We can reduce this further by noticing this is the 4×4 DCT.

$$\begin{bmatrix} y_1 \\ y_5 \end{bmatrix} = \begin{bmatrix} 0.3536 & 0.3536 \\ 0.3536 & -0.3536 \end{bmatrix} \begin{bmatrix} x_1 + x_8 + x_4 + x_5 \\ x_2 + x_7 + x_3 + x_6 \end{bmatrix}$$

$$\begin{bmatrix} y_3 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0.4619 & 0.1913 \\ 0.1913 & -0.4619 \end{bmatrix} \begin{bmatrix} x_1 + x_8 - (x_4 + x_5) \\ x_2 + x_7 - (x_3 + x_6) \end{bmatrix}$$

This reduces further the complexity and the final computation for $n = 8$ is thus 28 add/subtract operations and 22 multiply operations. Much less than the naive approach (64 mults and 56 adds).

Fast DCT

This was only one way of simplifying the DCT computation. There are many other optimisation strategies, improving from a naive approach of $O(n^2)$ to $O(n \log n)$.

DCT and convolution

It is possible to use the DCT to perform image analysis as well as compression but note that unlike the DFT, convolution of two signals in space does not correspond to the product of their DCT Transforms.

DCT

The 8-point DCT is the basis of the JPEG standard, as well as several other standards such as MPEG-1, MPEG-2 (for TV and video) and H.263 and MPEG4/H.264, H.265, vp9, vpx. Hence we shall concentrate on it as our main example, but bear in mind that DCTs may be defined for a wide range of sizes n (4, 8, 16, 32, 64).

DCT in 2D

Recall from the previous handout:

$$\mathbf{y} = T\mathbf{x}T^{\mathsf{T}} \text{ and to invert: } \mathbf{x} = T^{\mathsf{T}}\mathbf{y}T$$

There is was shown that a 1-D transform could be extended to 2-D by pre- and post-multiplication of a square matrix $\mathbf{x}$ to give a matrix result $\mathbf{y}$. The example then used $\mathbf{2 \times 2}$ matrices, but this technique applies to square matrices of any size. Hence the DCT may be extended into 2-D by this method.

DCT in 2D

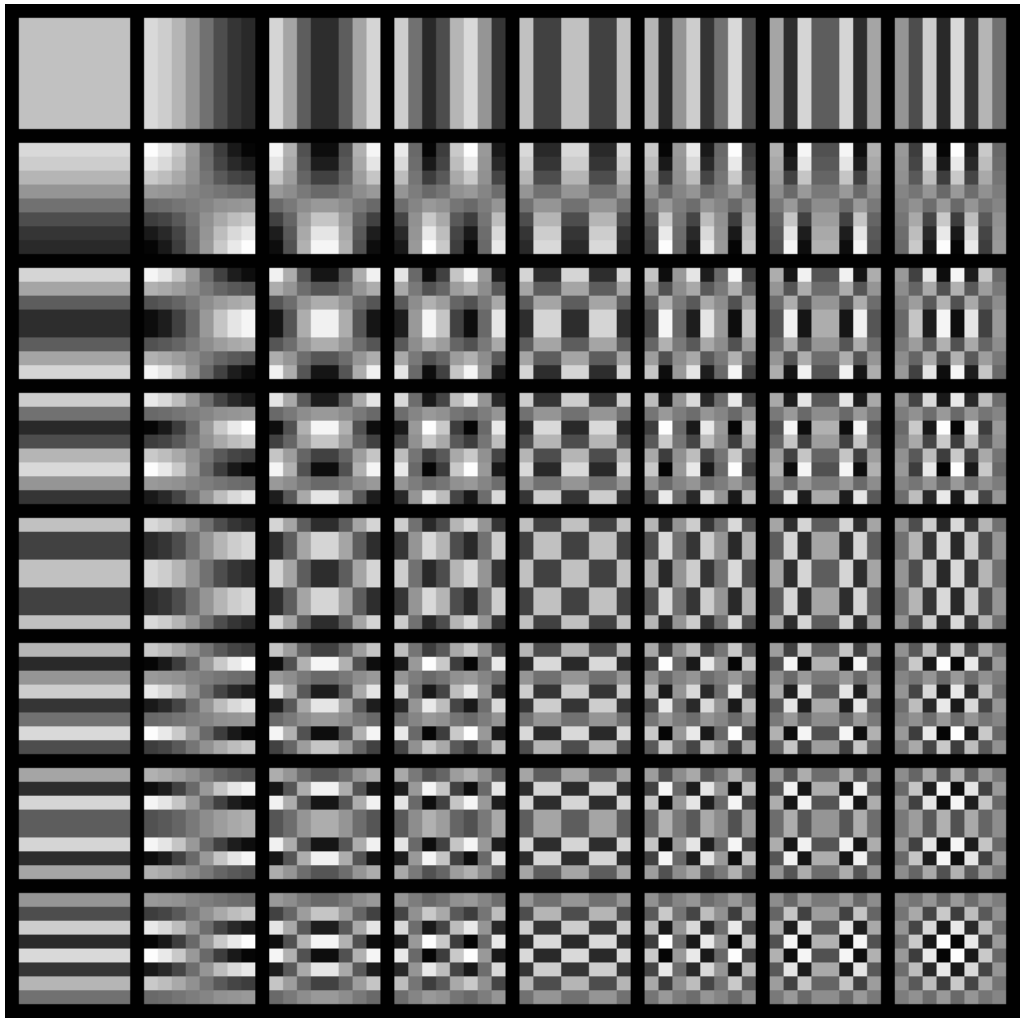
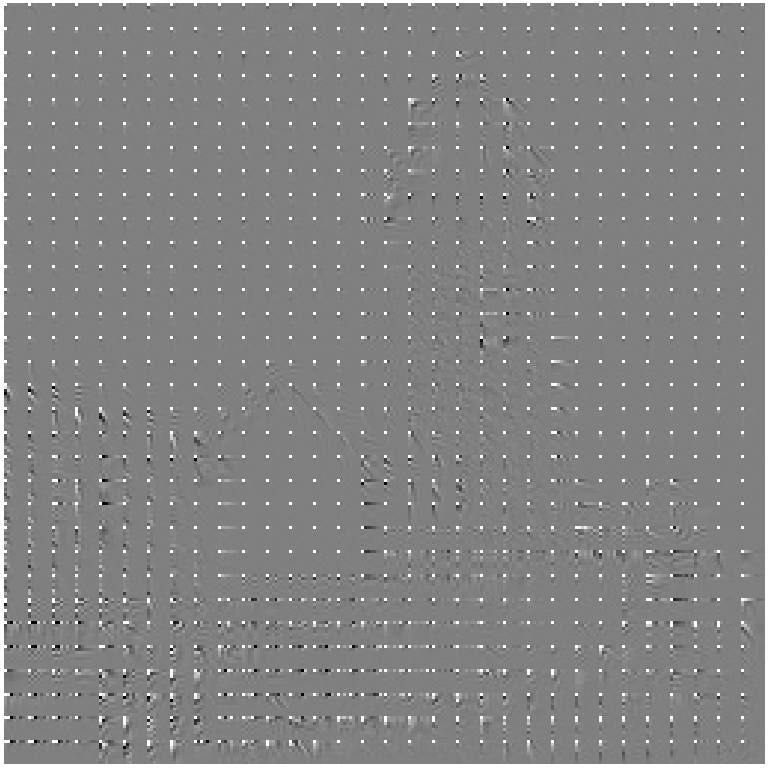


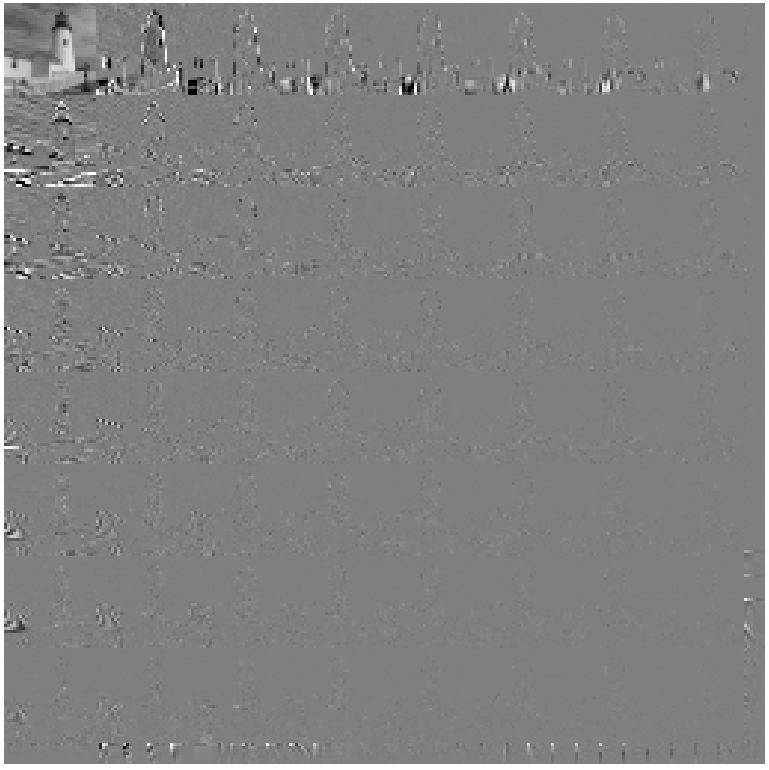
Figure: The 64 subimages basis of the 8×8 DCT.

DCT in 2D



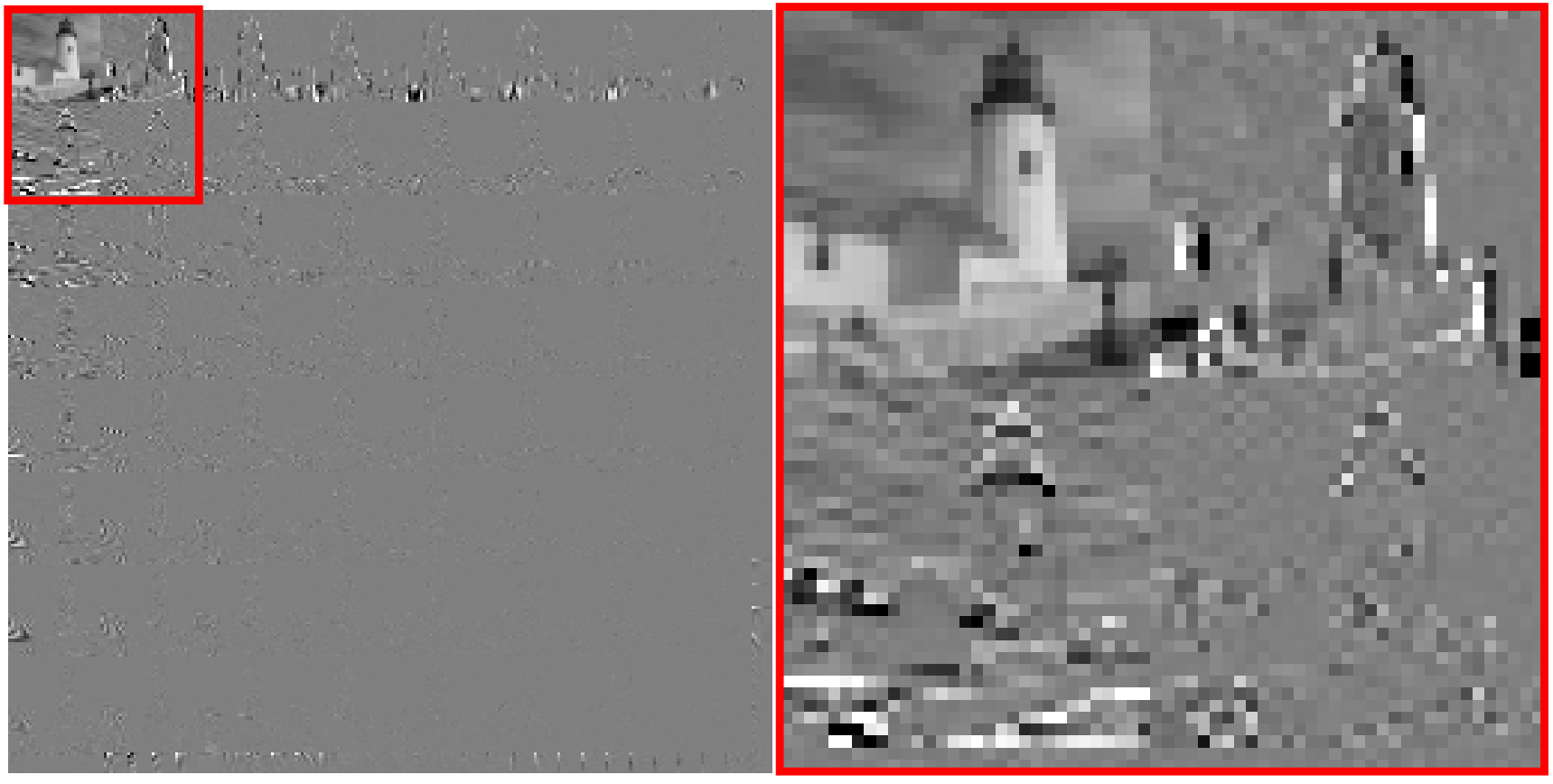
We apply the 8×8 DCT to each 8×8 block. (brightness and contrast adjusted as for Haar).

DCT in 2D



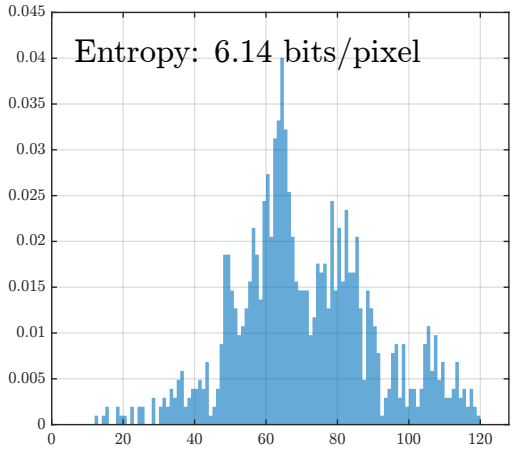
We can reorder the coefficients into subimages grouped by coefficient type.

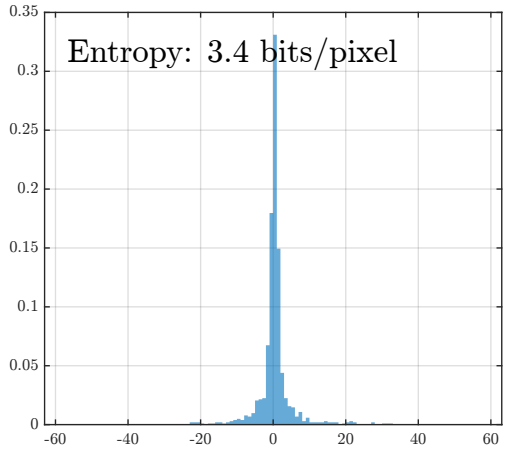
DCT in 2D

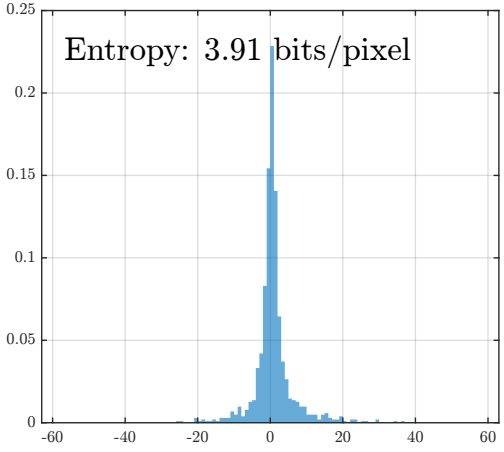


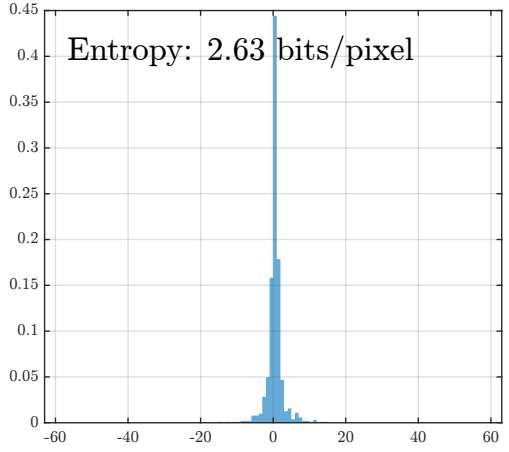
Here we've highlighted the DCT top left bands. They bear strong similarity with the Haar subimages.

DCT in 2D





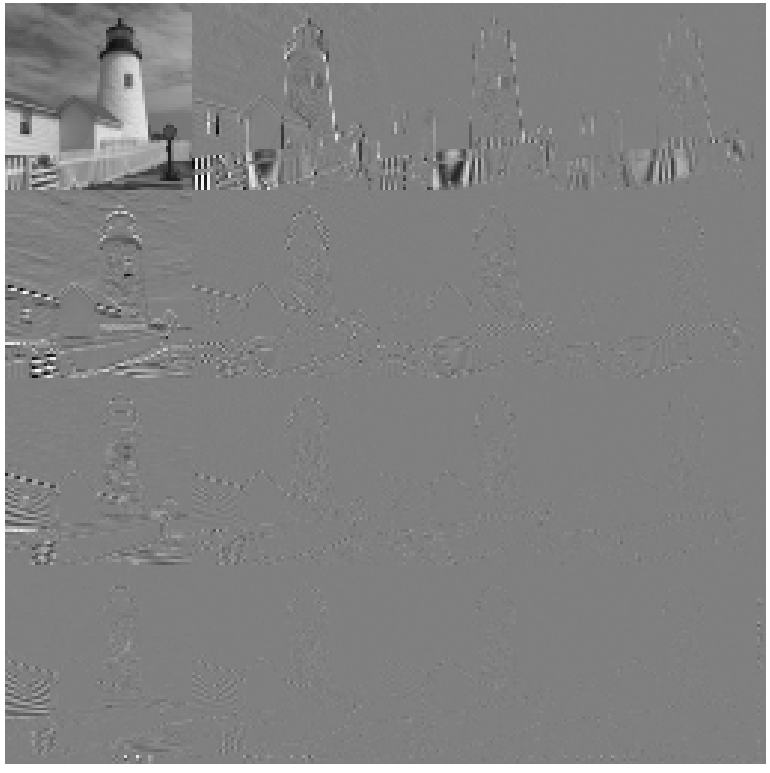




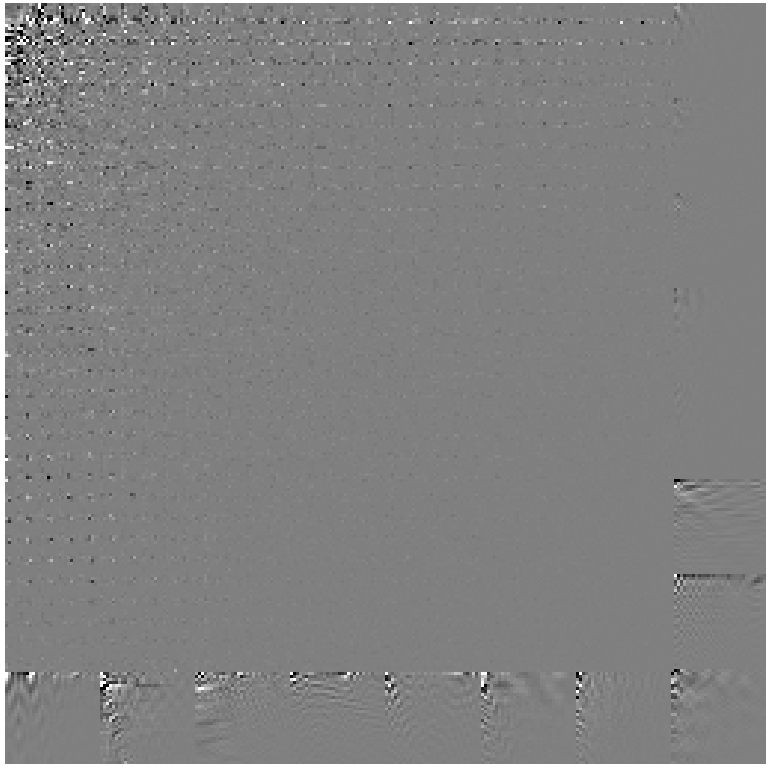
Entropies for these 4 subbands at $Q_{step} = 15$.

DCT Size

What is the optimum DCT size?



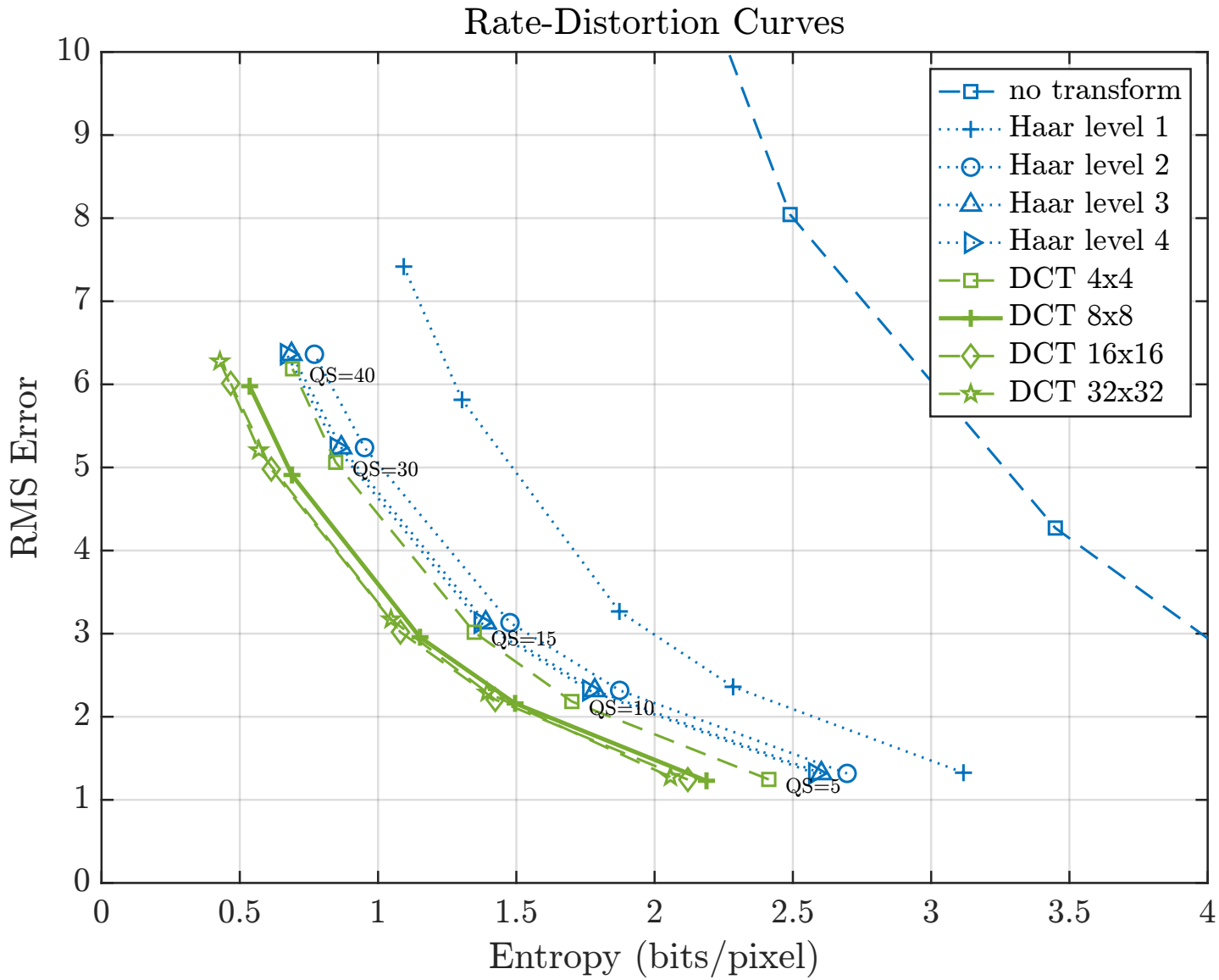
4×4 DCT



16×16 DCT

We can choose between 4×4, 8×8, 16×16, 32×32, 64×64.

RD Curves


Rate-Distortion Curves
RMS Error
Entropy (bits/pixel)
0
1
2
3
4
5
6
7
8
9
10
0
0.5
1
1.5
2
2.5
3
3.5
4
QS=40
QS=30
QS=15
QS=10
QS=5
no transform
Haar level 1
Haar level 2
Haar level 3
Haar level 4
DCT 4x4
DCT 8x8
DCT 16x16
DCT 32x32

RD Curves

The RD-Curves seem to indicate that for this picture, the 8×8, 16×16 and 32×32 perform best, with the DCT significantly surpassing Haar.

In practice, for a wide range of images and viewing conditions, 8×8 has been found to be the optimum DCT block size and is specified in most current coding standards.

Modern video codecs find a best compromise by splitting frames into blocks of different sizes (ranging from 4×4 to 64×64).

DCT Size

8×8 DCT at $Q_{step} = 15$

16×16 DCT at $Q_{step} = 15$

DCT Size

4×4 DCT at $Q_{step} = 15$

32×32 DCT at $Q_{step} = 15$

DCT Size

8×8 DCT at $Q_{step} = 15$

16×16 DCT at $Q_{step} = 15$

DCT Size

4×4 DCT at $Q_{step} = 15$

32×32 DCT at $Q_{step} = 15$

Color Images

JPEG is working on the YUV images (YUV will be referred to as YCbCr in the literature).

The chrominance channels are usually downsampled and there are 3 commonly used modes:

4:4:4 – no chrominance subsampling

4:2:2 – Every 2nd column in the chrominance channels are dropped.

4:2:0 – Every 2nd column and row is dropped.

The DCT is applied separately on each channel.

Quantisation of DCT Coefficients

Quantisation

Recall that the quantisation for a coefficient is given by:

$$Q(y) = \mathrm{round}\left(\frac{y}{Q_{step}}\right)$$

Quantisation

In JPEG it is standard to apply different thresholds to different bands

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

Thus the quantisation will change depending on the band. For a coefficient y in a band (i, j), the quantisation step is as follows:

$$Q_{lum}(y) = \text{round}\left(\frac{y}{Q_{lum}(i,j)}\right)$$

Quantisation

Here are the quantisation steps for the bands of the chrominance channels:

$$Q_{chr} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Quantisation

The Quantisation step values were obtained by perceptual tests.

Users were presented with an image and were asked to increase the gain of a given band until a difference is perceived.

So for a given band (i, j), for which the subimage is $\mu_{i,j}$, we have:

$$I_{vis}(x, y) = I_{ori}(x, y) + Q(i, j)\mu_{i,j}(x, y)$$

Quantisation

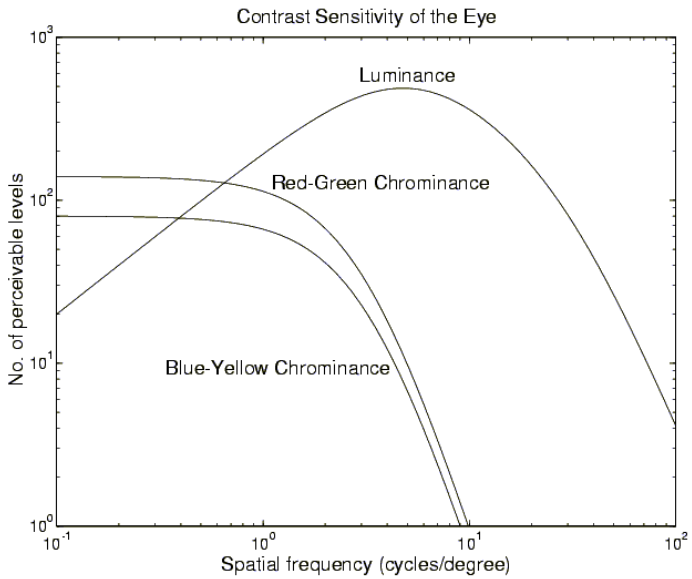
- The quantisation steps get bigger with the spatial frequency.
- There is a slight drop off from the luminance DC coefficient to low frequency coefficients.
- There is an asymmetry between the horizontal and vertical frequencies.
- The step sizes for the chrominance channels increase faster than for luminance.

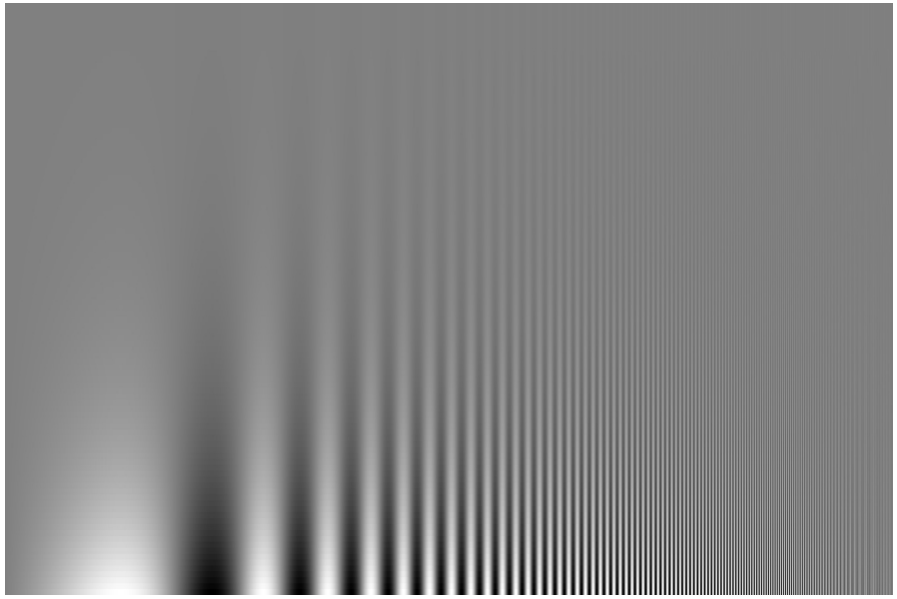
$$Q_{lum} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103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

$$Q_{chr} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Quantisation

all this is consistent with what we saw when covering the HVS:





Quantisation

original

8×8 DCT with $Q_{step} = 0.5 \times Q_{lum}$

Quantisation

original

8×8 DCT with $Q_{step} = 15$

Quantisation

original

8×8 DCT with $Q_{step} = Q_{lum}$

Quantisation

original

8×8 DCT with $Q_{step} = 2 \times Q_{lum}$

Quantisation

8×8 DCT with $Q_{step} = 15$

8×8 DCT with $Q_{step} = Q_{lum}$

Quantisation

8×8 DCT with $Q_{step} = 15$

8×8 DCT with $Q_{step} = 2Q_{lum}$

Quantisation

8×8 DCT with $Q_{step} = 15$

8×8 DCT with $Q_{step} = .5Q_{lum}$

JPEG Entropy Coding

JPEG Coding

Minimise average codeword length.

- We will use RLC to encode the zeros.
- We must take advantage of spatial and inter-band correlations.
- We need to consider how we order the data.

Minimise the coding overhead

- minimise the size of the huffman codetable
- we need to reduce the number of symbols we encode
- This can affect optimality

Correct for Synchronisation Errors

JPEG Coding

The obvious way would be to code each band separately ie. Huffman with RLC like we suggested with the Haar Transform. We could get close to the entropy.

This is not the way it is coded because it would require 64 different codes. High cost in computation and storage of codebooks (many extra bits to save).

It ignores the fact that the zero coefficients occur at the same positions in multiple bands.

JPEG Coding

Instead we code each block separately.

A block contains 64 coefficients, one from each band.

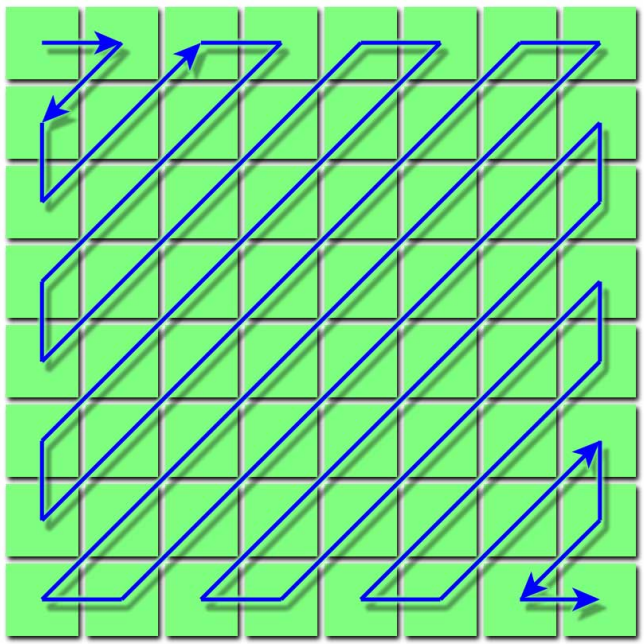
Each block contains 1 DC coefficient (from the top left band) and 63 AC coefficients

Two codebooks are used in total for all the blocks, one for the DC coefficients and the other for the AC coefficients.

At the end of each Block we insert an End Of Block (EOB) symbol in the datastream

JPEG Coding

Each block is a 8x8 grid of coeffs. A Zig-Zag scan converts them into a 1D stream:



As most non-zero values occur in the top left corner, the Zig-Zag scan maximises the lengths zero runs so improves efficiency of RLC.

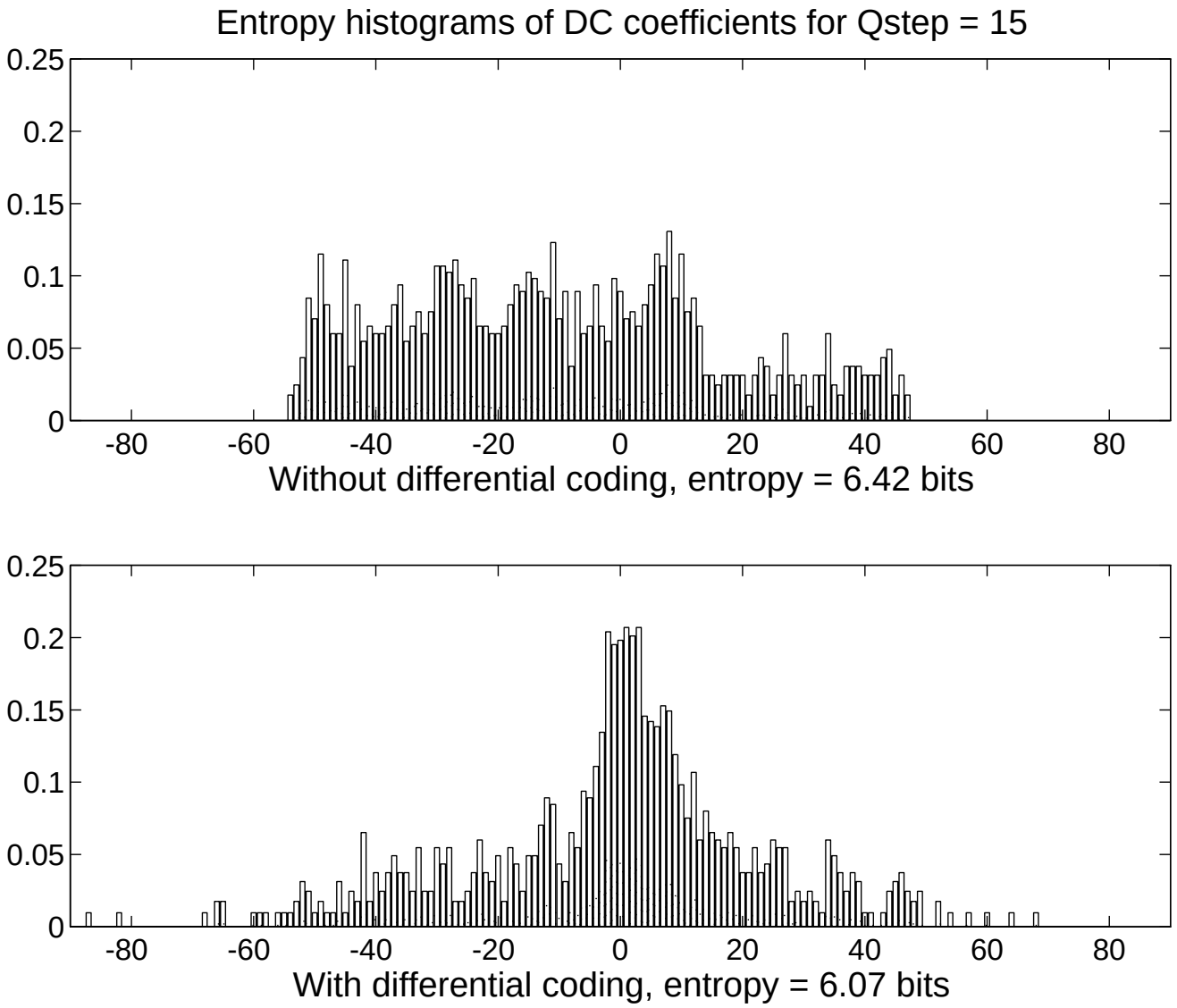
JPEG DC Coding

The DC coefficient (first coef of the Zig-Zag scan) exhibits different statistics from the remaining 63 AC coefficients. Thus a different coding is applied.

The DC coefs exhibit local correlations so differential coding is used in which the value to be coded is the difference between the current DC coef and the DC coef of the previous block.

The blocks are scanned from left to right, row by row. The first block in each row is coded with respect to zero.

JPEG DC Coding



JPEG DC Coding

The size of the differences can in theory be up to $\pm 255 \times 8 = \pm 2040$ if the input pels occupy the range -128 to +127 (the DCT has a gain of 8 at very low frequencies). Hence the Huffman code table would have to be quite large. JPEG adopts a much smaller code by using a form of floating-point representation, where Size is the base-2 exponent and Additional Bits are used to code the polarity and precise amplitude as follows.

DC Difference Categories

Category	Additional Bits	DC Value
0		0
1	0 1	-1 1
2	00,01 10,11	-3,-2 2,3
3	000,001,010,011 100,101,110,111	-7,-6,-5,-4 4,5,6,7
4	0000,...,0111 1000,...,1111	-15,...,-8 8,...,15
5	0 0000,... ...,1 1111	-31,...,-16 16,...,31
6	00 0000,... ...,11 1111	-63,...,-32 32,...,63
7	000 0000,... ...,111 1111	-127,...,-64 64,...,127
8	0000 0000,... ...,1111 1111	-255,...,-128 128,...,255
9	0 0000 0000,... ...,1 1111 1111	-511,...,-256 256,...,511
A	00 0000 0000,... ...,11 1111 1111	-1023,...,-512 512,...,1023
B	000 0000 0000,... ...,111 1111 1111	-2047,...,-1024 1024,...,2047

Suggested Huffman code for DC Coefficients

Category	Bits
0	00
1	010
2	011
3	100
4	101
5	110
6 bits	1110
7 bits	11110
8 bits	111110
9 bits	1111110
A bits	11111110
B bits	111111110

JPEG DC Coding

Example: if a DC component is 40 and the previous DC component is 48. The difference is -8. Therefore it is coded as:

1010111

101: the category from the same table reads 4. The corresponding code from the table is 101.

0111: the value for representing -8 (see size and value table in previous slide).

JPEG AC coding

AC components (range $-1023 \cdots 1023$) are coded as pairs of symbols:

`(RunLength,Size)(Value)`

`RunLength` is the length of the consecutive zero values.

`Size` is the number of bits needed to code the next nonzero AC component's value.

`Value` is the bit value of the AC component (see table in next slide).

The symbol `(RunLength,Size)` is encoded in 8 bits and can then be further encoded with Huffman. JPEG also provides a default coding scheme that is presented in the next slide.

JPEG AC Coding

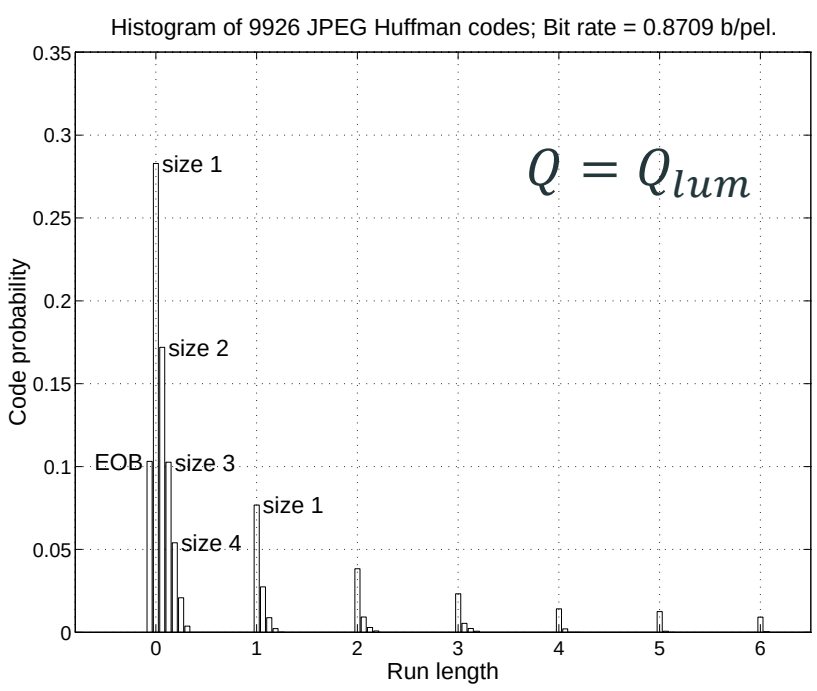
(Run,Size)	Code Word	(Run,Size)	Code Word
(0,1)	00	(0,6)	1111000
(0,2)	01	(1,3)	1111001
(0,3)	100	(5,1)	1111010
EOB	1010	(6,1)	1111011
(0,4)	1011	(0,7)	11111100
(1,1)	1100	(2,2)	11111001
(0,5)	11010	(7,1)	11111010
(1,2)	11010	(1,4)	111110110
(2,1)	11011		⋮
(3,1)	11100	ZRL	11111111001
(4,1)	111011		⋮

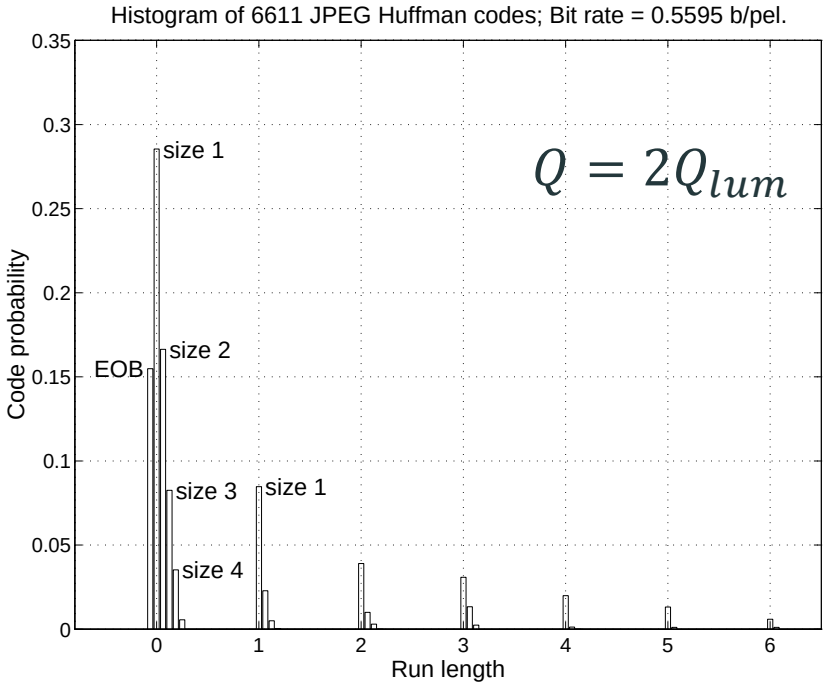
How good is this scheme?

We can compare the entropy after using RLC, Zig-Zag Scanning and AC and DC coefficient coding (with Huffman) strategies with the theoretical values of entropy.

Q Matrix	Mean Entropy (bit per pixel)	JPEG bit rate (bit per pixel)	JPEG efficiency
Q_{lum}	0.8595	0.8709	98.7%
$2Q_{lum}$	0.5551	0.5595	99.21%

AC Coefficient Probabilities





Even though doubling the quantisation sizes reduces the number of events the distribution of those events doesn't change much. Only the EOB probability changes significantly.

Therefore using the same codetable for both cases is reasonable.

How good is this scheme?

In fact using the same codetable for multiple images doesn't reduce the efficiency of the code much.

Q Matrix	JPEG efficiency	Efficiency when using default codetable
Q_{lum}	98.7%	97.35%
$2Q_{lum}$	99.21%	95.74%

Baseline JPEG

This section of the course has covered an introduction to a real image compression standard, JPEG. It is a widely used image format and it employs a building block (the DCT) which is also used in MPEG2 and MPEG4 for video compression.

The main components of JPEG are thus:

- Color Transform (RGB → YUV/YCbCr);
- Image Partition;
- Discrete Cosine Transform;
- Quantization;
- DC Coefficient Encoding;
- Zig-zag ordering of AC Coefficients;
- Entropy Coding.