Application: The DCT and JPEG
Image and Video Processing

Dr. Anil Kokaram* anil.kokaram@tcd.ie

1 The DCT and the JPEG Standard

This section continues to connect portions of this course and the course in Digital Communications in examining the use of DCT in the implementation of the JPEG standard for compression. The goals here are

- To introduce the DCT and its computation
- To introduce basic elements of JPEG
- To introduce the DC code for DC coefficents which will be re-used in the MPEG section of the course in this year and the next.

2 The DCT and the JPEG Standard

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 CCITT 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).

3 The Discrete Cosine Transform (DCT)

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

*Abridged version of material from Nick Kingsbury, Signal Processing Group, Cambridge University

If $\mathbf{T}$ 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:

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

A 4-point orthonormal transform matrix that is equivalent to 2 levels of the Haar transform is:

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

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

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.*

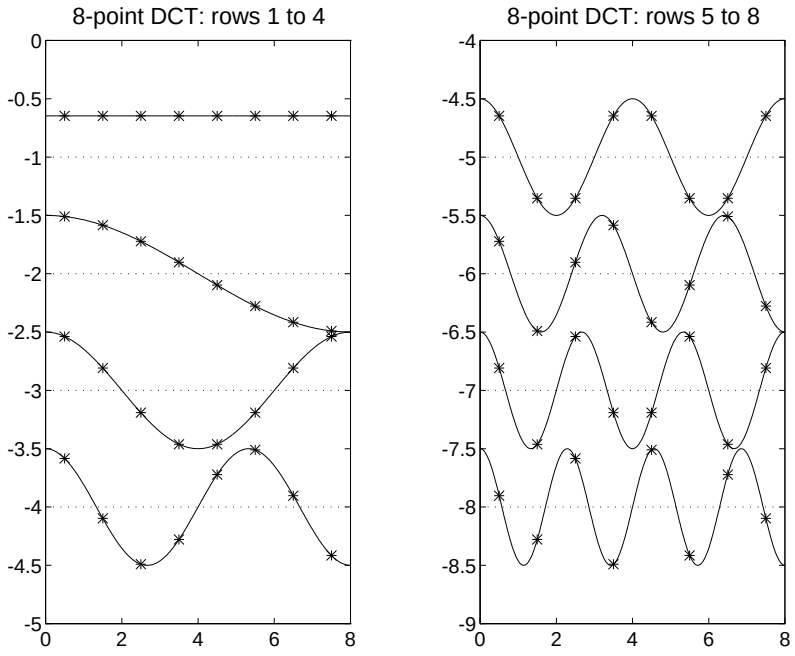


Fig 3.1: The 8-point DCT basis functions (*) and their underlying continuous cosine waves.

Discrete Cosine Transforms (DCTs) have some of these improved properties and are also simple

to define and implement. The n rows of an n-point DCT matrix $\mathbf{T}$ are defined by:

$$\begin{aligned} t_{1i} &= \sqrt{\frac{1}{n}} \qquad \text{for } i = 1 \rightarrow n, \\ t_{ki} &= \sqrt{\frac{2}{n}} \cos\left(\frac{\pi(2i-1)(k-1)}{2n}\right) \qquad \text{for } i = 1 \rightarrow n, k = 2 \rightarrow n. \end{aligned} \tag{3}$$

It is straightforward to show that this matrix is orthonormal for n even, since the norm of each row is unity and the dot product of any pair of rows is zero (the product terms may be expressed as the sum of a pair of cosine functions, which are each zero mean).

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

$$\mathbf{T} = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix} \tag{4}$$

The rows of $\mathbf{T}$, known as basis functions, are plotted as asterisks in fig 3.1. The asterisks are superimposed on the underlying continuous cosine functions, used in all sizes of DCT. Only the amplitude scaling and the maximum frequency vary with the size n.

When we take the transform of an n-point vector using $\mathbf{y} = \mathbf{T}\mathbf{x}$, $\mathbf{x}$ is decomposed into a linear combination of the basis functions (rows) of $\mathbf{T}$, whose coefficients are the samples of $\mathbf{y}$, because $\mathbf{x} = \mathbf{T}^T\mathbf{y}$.

The basis functions may also be viewed as the impulse responses of FIR filters, being applied to the data $\mathbf{x}$.

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

$$\mathbf{x}_{2n} = \begin{bmatrix} c\mathbf{x} \\ \mathbf{x}_{rev} \end{bmatrix} \qquad \text{where } \mathbf{x}_{rev} = \begin{bmatrix} cx(n) \\ \vdots \\ x(1) \end{bmatrix}$$

$\mathbf{x}_{2n}$ 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.

Note: the DFT must be defined with a half sample period offset on the indexing of the input samples for the above to be strictly true.

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. In effect this is due to the fact that the DCT is a transform which does not identify the orientation of features uniquely in the transform domain.

Standards

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 (for mobile phones). Even DivX is based on the DCT. 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.

3.1 Fast algorithms for the DCT

The basic n-point DCT requires n^2 multiplications and $n(n-1)$ additions to calculate $\mathbf{y} = \mathbf{T}\mathbf{x}$ (64 mults and 56 adds for $n = 8$).

From fig 3.1, it is clear that symmetries exist in the DCT basis functions. These can be exploited to reduce the computation load of the DCT.

All the odd rows of $\mathbf{T}$ in (4) possess even symmetry about their centres and all the even rows possess odd symmetry. Hence we may form:

$$u(i) = x(i) + x(9-i) \quad \text{and} \quad v(i) = x(i) - x(9-i) \quad \text{for } i = 1 \rightarrow 4 \tag{5}$$

and then form the odd and even terms in $\mathbf{y}$ from two 4×4 transforms:

$$\begin{bmatrix} y(1) \\ y(3) \\ y(5) \\ y(7) \end{bmatrix} = \mathbf{T}_{left,odd}\ \mathbf{u} \qquad \text{and} \qquad \begin{bmatrix} y(2) \\ y(4) \\ y(6) \\ y(8) \end{bmatrix} = \mathbf{T}_{left,even}\ \mathbf{v} \tag{6}$$

where $\mathbf{T}_{left,odd}$ and $\mathbf{T}_{left,even}$ are the 4×4 matrices formed by the left halves of the odd and even rows of $\mathbf{T}$.

This reduces the computation to 8 add/subtract operations for equations (5) and 2×16 mults and 2×12 adds for equations (6) – almost halving the total computation load.

The matrix $\mathbf{T}_{left,even}$ cannot easily be simplified much further, but $\mathbf{T}_{left,odd}$ can, as it possesses the same symmetries as $\mathbf{T}$ (it is equivalent to a 4-point DCT matrix). Hence we may use the same technique on this matrix to reduce the 16 mults and 12 adds for this product to 4 add/subtract operations followed by a pair of 2×2 matrix products, requiring 2×4 mults and 2×2 adds. Finally

two of these mults may be saved since one of the 2×2 matrices is just a scaled add/subtract matrix (like the Haar transform).

The total computation load for the 8×8 DCT then becomes:

- $(8 + 12) + (4 + 2 + 2) = 28$ add/subtract operations;
- $16 + 4 + 2 = 22$ multiply operations.

More complicated algorithms exist (JPEG Book, sections 4.3.2 to 4.3.5) which reduce the number of multiplies further. However these all require more intermediate results to be stored. In modern DSP chips this can cost more CPU cycles than the extra multiplications which can often be done simultaneously with additions. Hence the simple approach given above is frequently optimal.

3.2 The 2-dimensional DCT

Recall from the previous handout:

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

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 2×2 matrices, but this technique applies to square matrices of any size.

Hence the DCT may be extended into 2-D by this method.

E.g. the 8×8 DCT transforms a subimage of 8×8 pels into a matrix of 8×8 DCT coefficients.

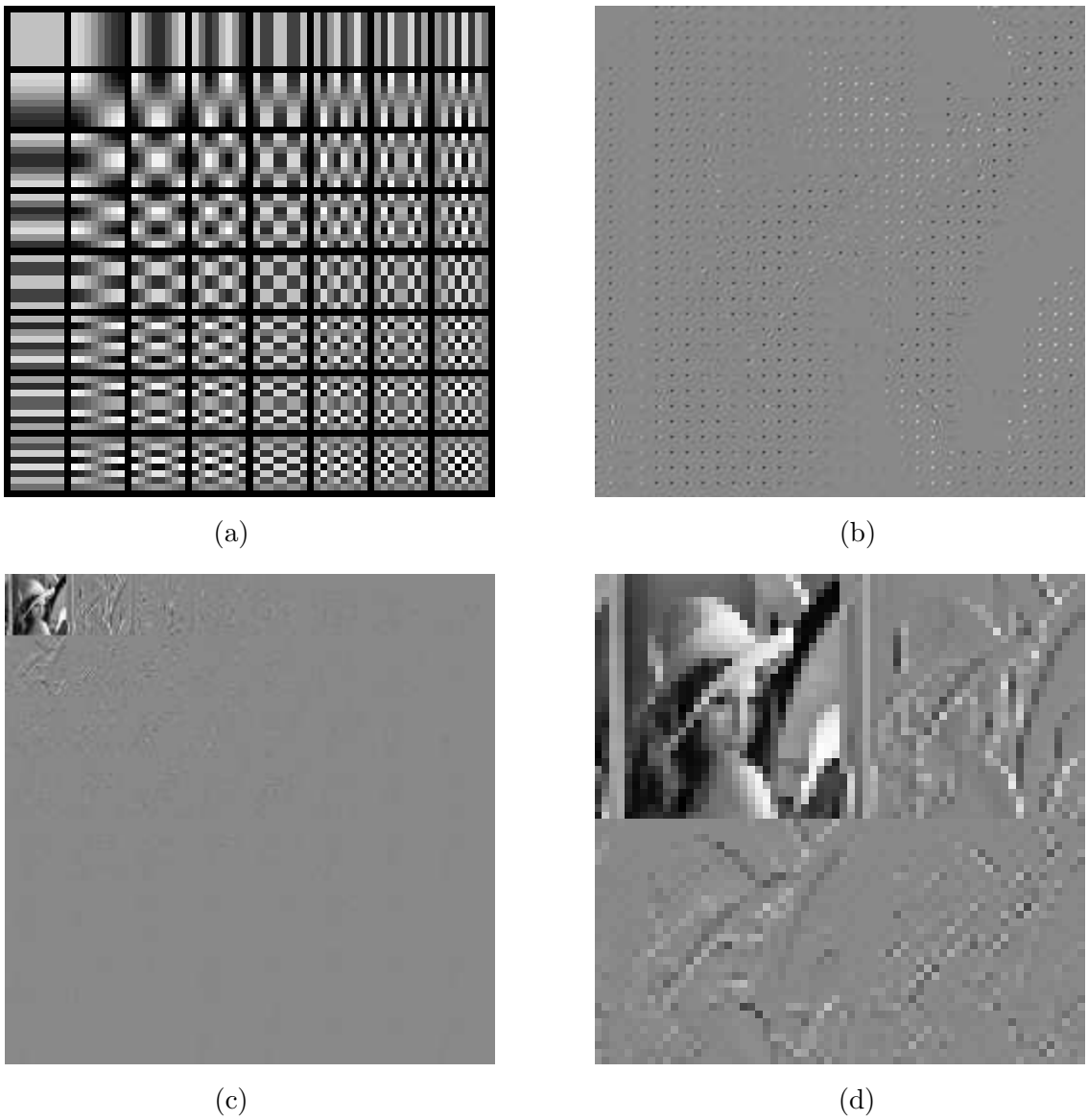
The 2-D basis functions, from which $\mathbf{x}$ may be reconstructed, are given by the n^2 separate products of the columns of $\mathbf{T}^T$ with the rows of $\mathbf{T}$. These are shown for $n = 8$ in fig 3.2a as 64 subimages of size 8×8 pels.

The result of applying the 8×8 DCT to the Lenna image is shown in fig 3.2b. Here each 8×8 block of pels $\mathbf{x}$ is replaced by the 8×8 block of DCT coefficients $\mathbf{y}$. This shows the 8×8 block structure clearly but is not very meaningful otherwise.

Fig 3.2c shows the same data, reordered into 64 subimages of 32×32 coefficients each so that each subimage contains all the coefficients of a given type – e.g: the top left subimage contains all the coefficients for the top left basis function from fig 3.2a. The other subimages and basis functions correspond in the same way.

We see the major energy concentration to the subimages in the top left corner. Fig 3.2d is an enlargement of the top left 4 subimages of fig 3.2c and bears a strong similarity to the group of third level Haar subimages in fig 2.7b. To emphasise this the histograms and entropies of these 4 subimages are shown in fig 3.3.

Comparing fig 3.3 with fig 2.9, the Haar transform equivalent, we see that the Lo-Lo bands have identical energies and entropies. This is because the basis functions are identical flat surfaces in both cases. Comparing the other 3 bands, we see that the DCT bands contain more energy and entropy than their Haar equivalents, which means *less* energy (and so hopefully less entropy) in the higher DCT bands (not shown) because the total energy is fixed (the transforms all preserve total energy). The mean entropy for all 64 subimages is 1.3622 bit/pel, which compares favourably with the 1.6103 bit/pel for the 4-level Haar transformed subimages using the same $Q_{step} = 15$.



(a) (b)

(c) (d)

Fig 3.2: (a) Basis functions of the 8×8 DCT; (b) Lenna transformed by the 8×8 DCT; (c) reordered into subimages grouped by coefficient type; (d) top left 4 subimages from (c).

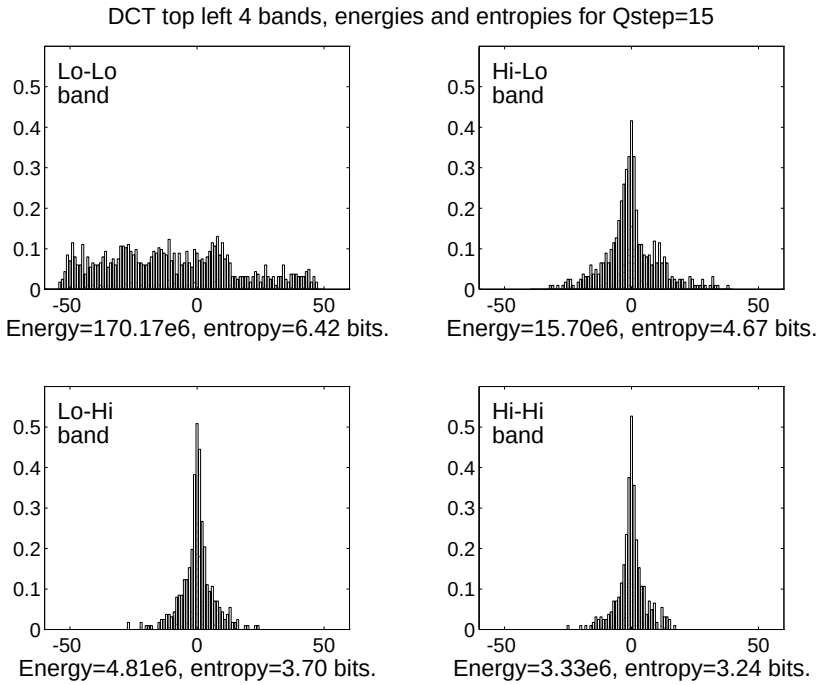
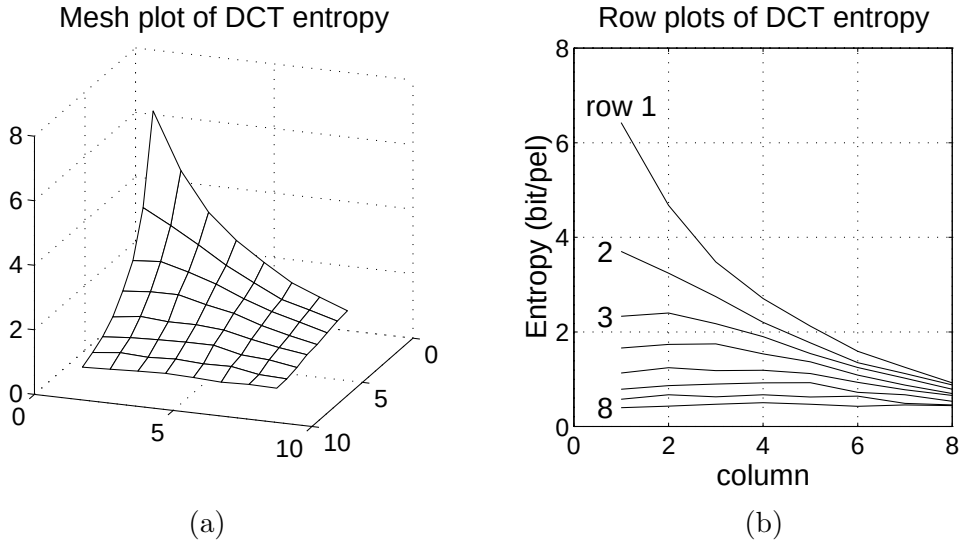
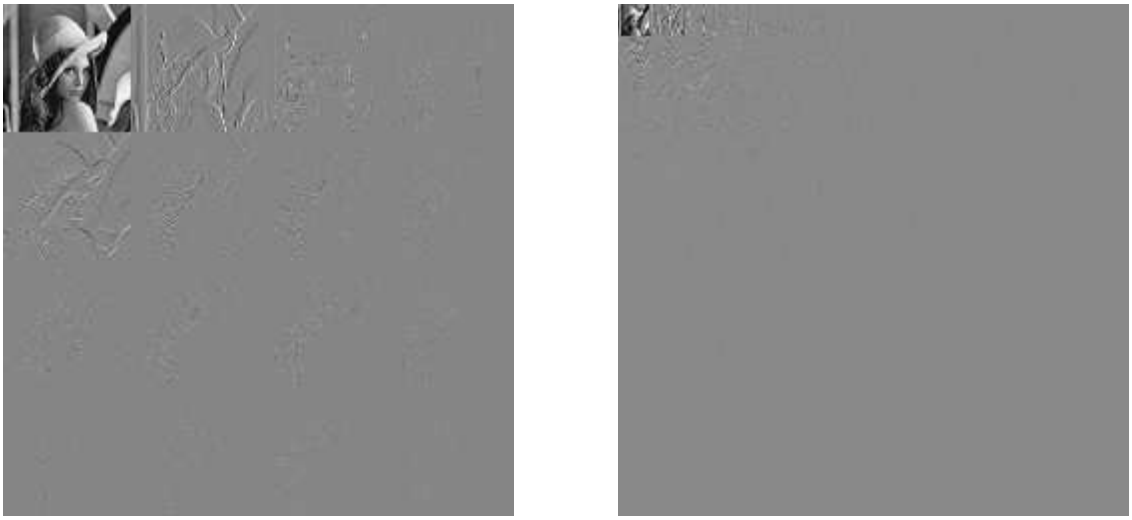


Fig 3.3: The probabilities p_i and entropies h_i for the 4 subimages from the top left of the 8×8 DCT (fig 3.2d).



(a) (b)

Fig 3.4: (a) Mesh and (b) row plots of the entropies of the subimages of fig 3.2c.



(a) (b)

Fig 3.5: Lenna transformed by the 4×4 DCT (a) and 16×16 DCT (b).

What is the optimum DCT size?

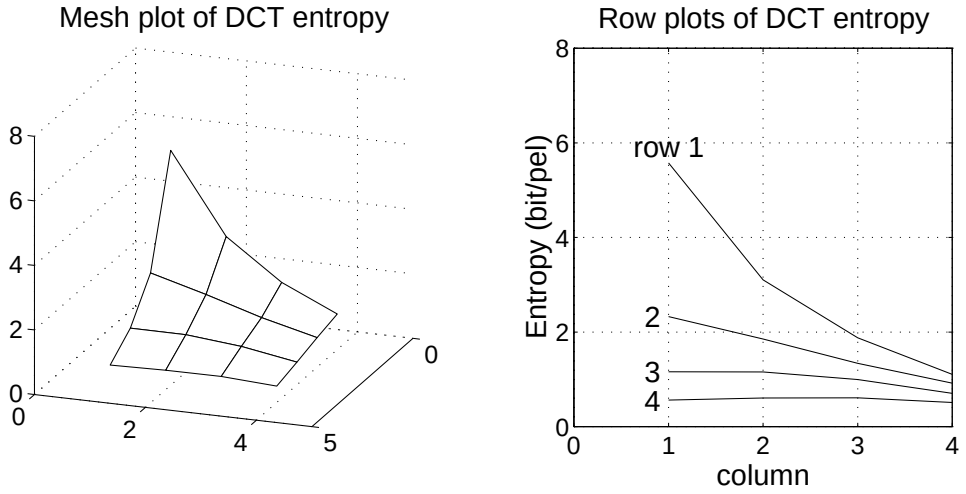
This is a similar question to: What is the optimum number of levels for the Haar transform?

We have analysed Lenna using DCT sizes from 2×2 to 16×16 to investigate this. Fig 3.5 shows the 4×4 and 16×16 sets of DCT subimages. The 2×2 DCT is identical to the level 1 Haar transform (see earlier handout) and the 8×8 set is in fig 3.2c.

Figs 3.6 and 3.7 show the mesh plots of the entropies of the subimages in fig 3.5.

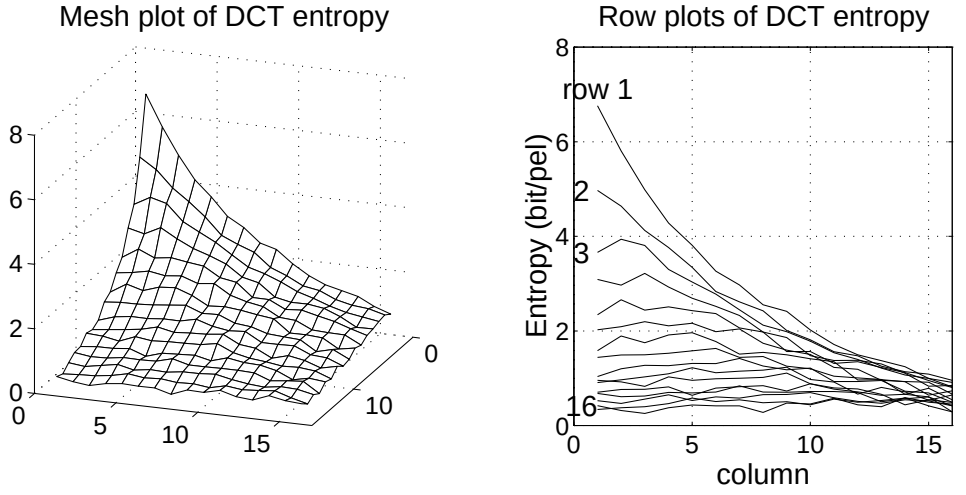
Fig 3.8 compares the total entropy per pel for the 4 DCT sizes with the equivalent 4 Haar transform sizes. We see that the DCT is significantly better than the rather simpler Haar transform.

As regards the optimum DCT size, from fig 3.8, the 16×16 DCT seems to be marginally better than the 8×8 DCT, but subjectively this is not the case since quantisation artefacts become more visible as the block size increases. In practise, 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.



(a) (b)

Fig 3.6: (a) Mesh and (b) row plots of the entropies of the 4×4 DCT in fig 3.5a.



(a) (b)

Fig 3.7: (a) Mesh and (b) row plots of the entropies of the 16×16 DCT in fig 3.5b.

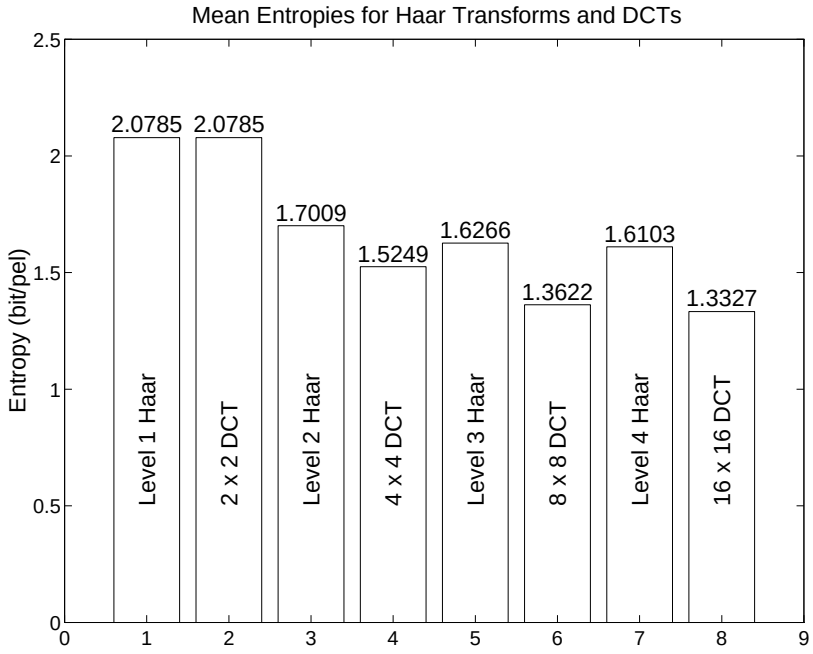


Fig 3.8: Comparison of the mean entropies of the Haar transform of Lenna at levels 1 to 4, and of the DCT for sizes from 2×2 to 16×16 pels with $Q_{step} = 15$.

3.3 Quantisation of DCT Coefficients

For the previous discussion we assumed a quantiser step size of 15 to allow direct comparison of entropies with the Haar transform. But what step size do we really need?

Figs 3.9a and 3.9b show images reconstructed from the 8×8 DCT of Lenna (fig 3.2c), when all the DCT coefficients are quantised with step sizes of 15 and 30 respectively. It is difficult to see quantising artefacts in fig 3.9a ($Q_{step} = 15$) but they are quite noticeable in fig 3.9b ($Q_{step} = 30$).

The visibility of the 8×8 DCT basis functions of fig 3.2a has been measured (for a 720×576 image viewed from 6 times the image width) and the minimum quantiser steps have been determined which will give artefacts just at the threshold of visibility. The matrices (JPEG Book, p37) for the luminance and chrominance threshold step sizes are:

$$\mathbf{Q}_{lum} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{7}$$

$$\mathbf{Q}_{chr} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \tag{8}$$

Fig 3.9c shows the reconstructed image when each subimage of fig 3.2c is quantised using the corresponding step size from $\mathbf{Q}_{lum}$. It is certainly difficult to detect any quantising artefacts, even though many of the step sizes are greater than $Q_{step} = 30$, used in fig 3.2b. Fig 3.9d is the reconstructed image using step sizes of $2 \times \mathbf{Q}_{lum}$ and the artefacts are still quite low.

(a) (b)

(c) (d)

Fig 3.9: Images reconstructed using the 8×8 DCT with (a) $Q_{step} = 15$, (b) $Q_{step} = 30$, (c) $\mathbf{Q}_{step} = \mathbf{Q}_{lum}$, the JPEG luminance matrix, and (d) $\mathbf{Q}_{step} = 2 \times \mathbf{Q}_{lum}$.

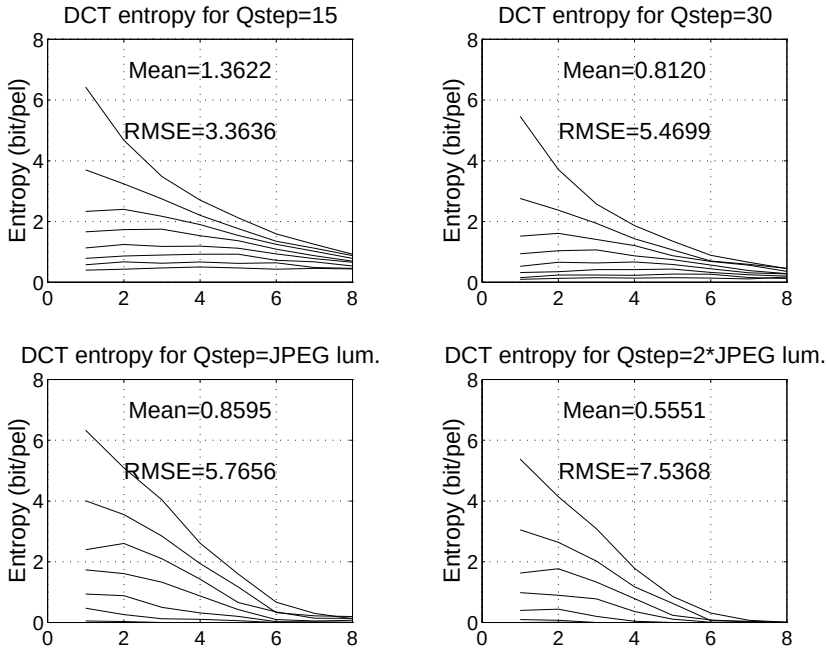


Fig 3.10: Plots of the entropies of the 8×8 DCT quantised subimages for the four reconstructed images of fig 3.9.

Fig 3.10 shows the entropies of the 64 quantised subimages used to reconstruct each of the four images in fig 3.9. Also given on each plot is the mean entropy (giving the bits/pel for the image) and the rms quantising error between the quantised image and the original.

We see that image (c) has about the same mean entropy and rms error as image (b), but that its quantising artefacts are much less visible. Image (d) has similar visibility of artefacts to (b), but has significantly lower entropy and hence *greater compression* (similarly for images (c) versus (a)).

This shows the distinct advantages of *subjectively weighted quantisation*, and also that it is unwise to rely too much on the rms error as a measure of image quality.

3.4 JPEG Entropy Coding

The entropy plots of the last section show the theoretical entropies of each DCT sub-band. In practise this would be a poor way to code the data because:

- 64 separate entropy codes would be required (each requiring many extra states to represent run-length coding of zeros).
- The statistics for each code are likely to vary significantly from image to image.
- To transmit the code table for each sub-band as header information would involve a large coding overhead (many extra bits).
- Coding the sub-bands separately does not take account of the correlations which exist beween the positions of the non-zero coefs in one sub-band with those of nearby sub-bands (see figs 3.2c and 3.2d).

JPEG uses a clever alternative method of coding, based on combining run-length and amplitude information into a single Huffman code for the whole of the image (except the DC sub-band which is coded separately because its statistics are so different).

The code is applied to each block of 8×8 quantised DCT coefs from a single 8×8 pel region. The blocks are the coefs *before reordering* as shown in fig 3.2b and comprise one coef from each of the 64 sub-bands.

Each block of 8×8 quantised coefs is formed into a 1-D vector by zig-zag scanning in the sequence:

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

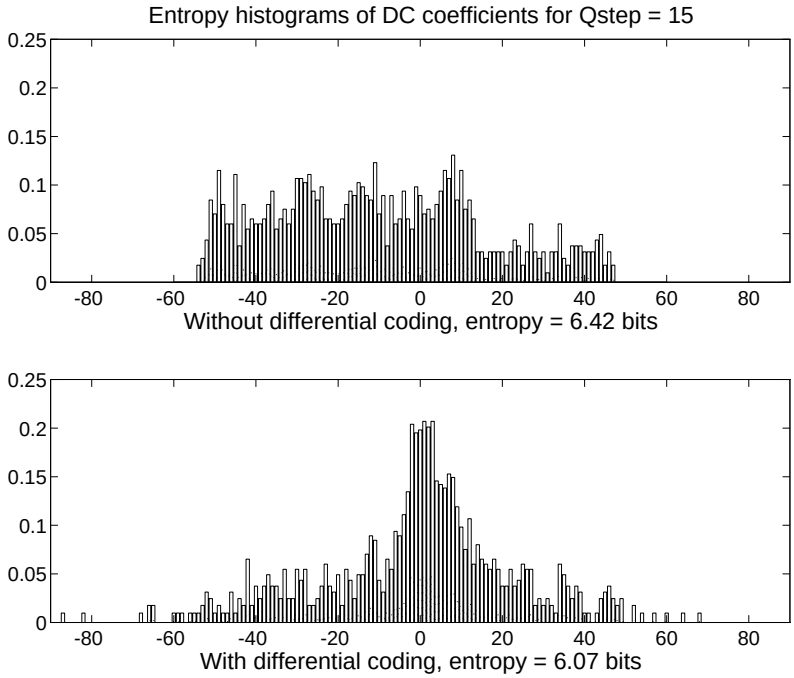


Fig 3.11: Histograms of the DC coefficients from the 8×8 DCT of Lenna, showing the entropy reduction with differential coding.

The JPEG Code for DC coefs

The first coefficient (0) of each block (vector) is the DC coef, which represents the mean value of the pels in the block (see the top left basis function in fig 3.2a).

The DC coefs still exhibit significant local correlations (top left of fig 3.2d), so differential coding is used in which the value to be coded is the *difference* between the current DC coef and that 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.

The histogram of entropies of the DC coef differences is compared in fig 3.11 with that of the raw DC coefs from fig 3.3. We note the histogram peak around zero and see that the entropy is reduced from 6.42 bits to 6.07 bits.

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 Coef Difference	Size	Typical Huffman codes for Size	Additional Bits (in binary)
0	0	00	$-$
$-1, 1$	1	010	$0, 1$
$-3, -2, 2, 3$	2	011	$00, 01, 10, 11$
$-7, \cdots, -4, 4, \cdots, 7$	3	100	$000, \cdots, 011, 100, \cdots, 111$
$-15, \cdots, -8, 8, \cdots, 15$	4	101	$0000, \cdots, 0111, 1000, \cdots, 1111$
$\vdots$	$\vdots$	$\vdots$	
$-1023, \cdots, -512, 512, \cdots, 1023$	10	$1111\,1110$	$00\,0000\,0000, \cdots, 11\,1111\,1111$
$-2047, \cdots, -1024, 1024, \cdots, 2047$	11	$1\,1111\,1110$	$000\,0000\,0000, \cdots, 111\,1111\,1111$

Only Size needs to be Huffman coded in the above scheme, since, within a given Size, all the input values have sufficiently similar probabilities for there to be little gain from entropy coding the Additional Bits (hence they are coded in simple binary as listed). Each coded Size is followed by the appropriate number of Additional Bits (equal to Size) to define the sign and magnitude of the coefficient difference exactly.

There are only 12 Sizes to be Huffman coded, so specifying the code table can be very simple and require relatively few bits in the header.

In JPEG all Huffman code tables are defined in the image header. Each table requires $16 + n$ bytes, where n is the number of codewords in the table.

The first 16 bytes list the number of codewords of each length from 1 to 16 bits (codewords longer than 16 bits are forbidden). The remaining n bytes list the decoded output values of the n codewords in ascending codeword order ($n < 256$).

Hence $16 + 12 = 28$ bytes are needed to specify the code table for DC coefficients.

The JPEG Run-Amplitude Code

The remaining 63 coefs (the AC coefs) of each 64-element vector usually contain many zeros and so are coded with a combined run-amplitude Huffman code.

The codeword represents the run-length of zeros before a non-zero coef **and** the Size of that coef. This is then followed by the Additional Bits which define the coef amplitude and sign precisely. Size and Additional Bits are defined just as for DC coefs.

This 2-dimensional Huffman code (Run, Size) is efficient because there is a strong correlation between the Size of a coef and the expected Run of zeros which precedes it – small coefs usually follow long runs; larger coefs tend to follow shorter runs. No single 2-D event is so probable that the Huffman code becomes inefficient.

In order to keep the code table size n below 256, only the following Run and Size values are coded:

$$\text{Run} = 1 \rightarrow 15 \qquad \text{Size} = 1 \rightarrow 10$$

These require 150 codes. Two extra codes, corresponding to (Run,Size) = (0,0) and (15,0) are used for EOB (End-of-block) and ZRL (Zero run length).

EOB is transmitted after the last non-zero coef in a 64-vector. It is the most efficient way of coding the final run of zeros. It is omitted in the rare case that the final element of the vector is non-zero.

ZRL is transmitted whenever Run > 15, and represents a run of 16 zeros (15 zeros and a zero amplitude coef) which can be part of a longer run of any length. Hence a run of 20 zeros followed by -5 would be coded as

(ZRL) (4,3) 010

When the code tables are defined in the image header, each codeword is assigned to a given (Run,Size) pair by making the decoded output byte *Code Byte* equal to $(16 \times \text{Run} + \text{Size})$.

The default JPEG code for (Run,Size) of AC luminance DCT coefficients is summarised below in order of decreasing code probability:

(Run,Size)	Code Byte (hex)	Code Word (binary)	(Run,Size)	Code Byte (hex)	Code Word (binary)
$(0,1)$	01	00	$(0,6)$	06	1111000
$(0,2)$	02	01	$(1,3)$	13	1111001
$(0,3)$	03	100	$(5,1)$	51	1111010
(EOB)	00	1010	$(6,1)$	61	1111011
$(0,4)$	04	1011	$(0,7)$	07	11111000
$(1,1)$	11	1100	$(2,2)$	22	11111001
$(0,5)$	05	11010	$(7,1)$	71	11111010
$(1,2)$	12	11011	$(1,4)$	14	111110110
$(2,1)$	21	11100		$\vdots$	
$(3,1)$	31	111010	(ZRL)	$F0$	11111111001
$(4,1)$	41	111011		$\vdots$	

As an example, let us code the following 8×8 block:

$$\begin{array}{cccccccc} -13 & -3 & 2 & 0 & 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Converting this to (DC Size) or (Run,Size) and values for the Additional Bits gives:

(4)	-13	$(0,2)$	-3	$(0,3)$	6	$(2,2)$	2	$(3,1)$	-1	(ZRL)	$(1,1)$	1	(EOB)
101	0010	01	00	100	110	11111001	10	111010	0	11111111001	1100	1	1010

The compressed bitstream for this block is listed on the lower line, assuming that the default Huffman code tables, given above, are used.

Fig 3.12 shows the histogram of probabilities for the (Run,Size) codewords used to code Lenna using the $\mathbf{Q}_{lum}$ quantisation matrix. The bin number represents the decoded byte value.

Fig 3.13 shows the equivalent histogram when the quantisation matrix is $2\mathbf{Q}_{lum}$.

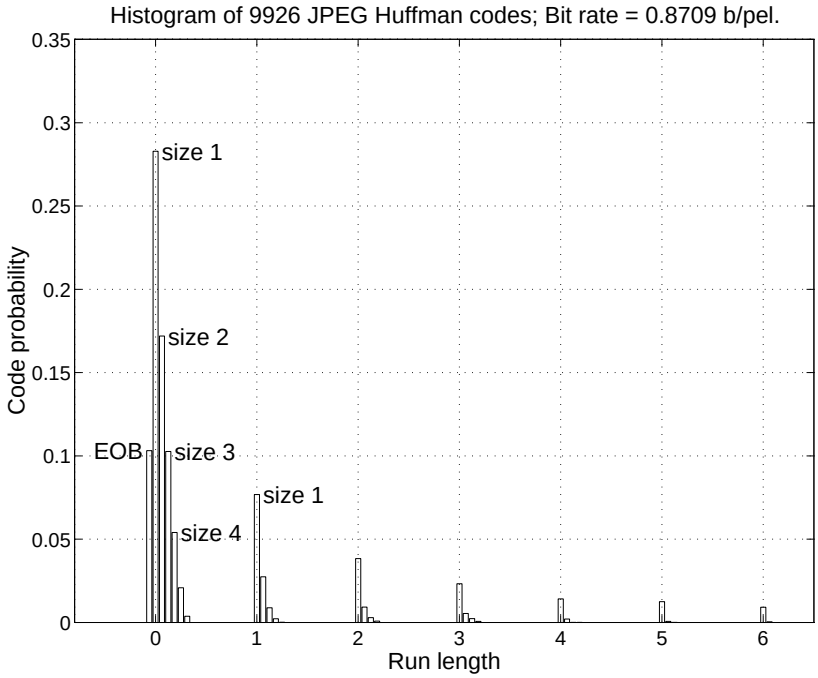


Fig 3.12: Histogram of the (Run,Size) codewords for the DCT of Lenna, quantised using $\mathbf{Q}_{lum}$.

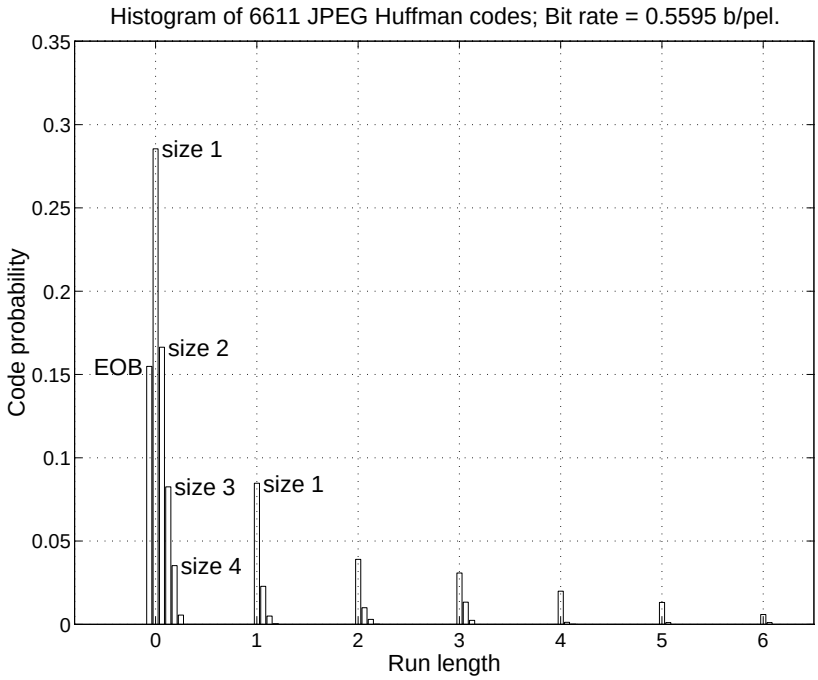


Fig 3.13: Histogram of the (Run,Size) codewords for the DCT of Lenna, quantised using $2\mathbf{Q}_{lum}$.

Note the strong similarity between these histograms, despite the fact that fig 3.13 represents only $\frac{2}{3}$ as many events. Only the EOB probability changes significantly, because its probability goes

up as the number of events (non-zero coefs) per block goes down.

It turns out that the (Run,Size) histogram remains relatively constant over a wide range of image material and across different regions of each image. This is because of the strong correlation between the run lengths and expected coef sizes. The number of events per block varies considerably depending on the local activity in the image, but the probability distribution of those events (except for EOB) changes much less.

Figs 3.12 and 3.13 also give the mean bit rates to code Lenna for the two quantisation matrices. Comparing these with the theoretical entropies from fig 3.10 (lower row) we get:

Q matrix	Mean Entropy b/pel	JPEG Bit Rate b/pel	JPEG efficiency
$\mathbf{Q}_{lum}$	0.8595	0.8709	98.7%
$2\mathbf{Q}_{lum}$	0.5551	0.5595	99.21%

Hence we see the high efficiency of the (Run,Size) code at two quite different compression factors. This tends to apply over a wide range of images and compression factors and is an impressive achievement.

There is even very little efficiency lost if a single code table is used for many images, which can avoid the need to transmit the $16 + n$ bytes (168 bytes) of code definition in the header of each image. Using the recommended JPEG default luminance tables (Annex K.3.3) the above efficiencies drop to 97.35% and 95.74% respectively.

3.5 Sync and Headers

We have described how individual 8×8 blocks of DCT coefficients are coded. Now we shall briefly look at the sync codes and header information that are needed in order to complete the coding process.

JPEG is rather complex in this aspect, so we shall just give an overview of the basic principles (see the JPEG Book, chapter 7 for the full picture).

JPEG data is divided into *segments*, each of which starts with a 2-byte *marker*.

All markers are byte-aligned – they start on the byte boundaries of the transmission/storage medium. Any variable-length data which precedes a marker is padded with extra ones to achieve this.

The first byte of each marker is FF_H. The second byte defines the type of marker.

To allow for recovery in the presence of errors, it must be possible to detect markers without decoding all of the intervening data. Hence markers must be unique. To achieve this, if an FF_H byte occurs in the middle of a segment, an extra 00_H *stuffed* byte is inserted after it and 00_H is never used as the second byte of a marker.

Some important markers in the order they are often used are:

Name	**Code** (hex)	**Purpose**
SOI	$FFD8$	Start of image.
COM	$FFFE$	Comment (segment ignored by decoder).
	L_{seg}, <Text comments>	
DQT	$FFDB$	Define quantisation table(s).
	L_{seg}, < $\mathbf{Q}_{lum}$, $\mathbf{Q}_{chr}$ ... >	
SOF_0	$FFC0$	Start of Baseline DCT frame.
	L_{seg}, <Frame size, no. of components (colours),	
		sub-sampling factors, Q-table selectors>
DHT	$FFC4$	Define Huffman table(s).
	L_{seg}, <DC Size and AC (Run,Size) tables for each component>	
SOS	$FFDA$	Start of scan.
	L_{seg}, <Huffman table selectors for each component>	
	<Entropy coded DCT blocks>	
EOI	$FFD9$	End of image.

In this table the data which follows each marker is shown between <> brackets. The first 2-byte word of most segments is the length (in bytes) of the segment, L_{seg}. The length of <Entropy coded DCT blocks>, which forms the main bulk of the compressed data, is not specified explicitly, since it may be determined by decoding the entropy codes. This also allows the data to be transmitted with minimal delay, since it is not necessary to determine the total length of the compressed data before any of the DCT block data can be sent.

Long blocks of entropy-coded data are rather prone to being corrupted by transmission errors. To mitigate the worst aspects of this, Restart Markers ($FFD0 \rightarrow FFD7$) may be included at regular intervals (say at the start of each row of DCT blocks in the image) so that separate parts of the entropy coded stream may be decoded independently of errors in other parts. The restart interval, if required, is defined by a DRI ($FFDD$) marker segment. There are 8 restart markers, which are used in sequence, so that if one (or more) is corrupted by errors, its absence may be easily detected.

The use of multiple scans within each image frame and multiple frames within a given image allows many variations on the ordering and interleaving of the compressed data. For example:

- Chrominance and luminance components may be sent in separate scans or interleaved into a single scan.

- Lower frequency DCT coefs may be sent in one or more scans before higher frequency coefs.
- Coarsely quantised coefs may be sent in one or more scans before finer (refinement) coefs.
- A coarsely sampled frame of the image may be sent initially and then the detail may be progressively improved by adding differentially-coded correction frames of increasing resolution.

4 Summary

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.

Lim does not cover JPEG itself, but there is a good coverage of the DCT. See pages 148–162