7. The JPEG Image Compression Standard

7.1 Introduction

- **JPEG (Joint Photographic Experts Group)**
  - Image coding standard for still images
    - Both greyscale and colour
  - Finalised in the early 1990s
  - Objective: exhibit state-of-the-art image compression
    - Without excessive computational complexity!

7.2 Overview of JPEG

- **JPEG has four modes of operation…**
  - *Sequential encoding**
    - Each image component (block) is encoded in a single left-to-right, top-to-bottom scan
      - Means transmission of encoded blocks can begin before the full image is available
  - *Progressive encoding*
    - Image encoded in multiple scans (low freq coeffs, then higher freq)
      - Means the user can watch the image gradually build up in multiple 'coarse-to-fine' passes
  - *Lossless encoding*
    - Image encoded such that it can be recovered *exactly* at the decoder
  - *Hierarchical encoding*
    - Image encoded at different levels of resolution
      - Means lower resolution versions can be accessed first

7.2.1 DCT-Based Sequential Mode

- **Note → Only the decoding process is actually defined by JPEG**
  - JPEG does not care how you encode your image → as long as the resulting bitstream is compliant with what decoder expects!
- **Nonetheless, we illustrate a typical encoding process (5 steps)**
  1. Source image pixels are grouped into 8x8 non-overlapping blocks
     - If the image dimensions are not multiples of 8, rows/columns duplicated
  2. Pixel values are *level shifted*: e.g. [0,255] → minus 128 → [-128,127])
     - Attempt to exploit symmetry towards smaller req'd range for DC coeff
  3. The **8x8 2-D DCT** is used to transform the 8x8 pixel block values…

$$F(0,0) = \frac{1}{8}\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)$$

$$F(m,n) = \frac{1}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)\cos\frac{(2i+1)m\pi}{16}\cos\frac{(2j+1)n\pi}{16}$$

for $0 \le m,n \le 7$ and one of $m,n \neq 0$.

→ i.e. 1/8 * ∑ all 8*8 pixels => max range of DC-DCT values: [-1024,1016]

7.2.1 DCT-Based Sequential Mode (cont.)

- **Continuing…**
  - Resulting DCT coeffs represent relative amounts of horiz/vert spatial frequencies in the block (DCT basis images)
    - *DC coeff* (DCC) → effectively the (zero-freq) mean of all pixels
    - Remaining 63 coeffs known as the *AC-coeffs* (ACCs)

4. Each of the 64 coeffs is quantized using a uniform quantizer
   - User-specifies a 64-element qzn table (QT)

     → Elements of the QT are **integers** in the range {0, 255}, which determine **step-size** of each qzr
     - Quantization implemented by **dividing** each coeff by the corresponding QT element, then **rounding** (→ simple!)

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

5. Quantized coeffs entropy coded

7.2.1 DCT-Based Sequential Mode (cont.)

- **Notes…**
  - As JPEG is a decoding standard → the QTs are NOT specified!
    - Chosen by the user according to the design criteria
  - In practice…
    - Quantization is the coding step where compression is achieved by deliberately lossy means…
      - i.e. typically apply the NON-optimal division of bits across coeffs

        → Higher freq coeffs (considered less significant to HVS) typically quantized more severly (→ larger step-size)

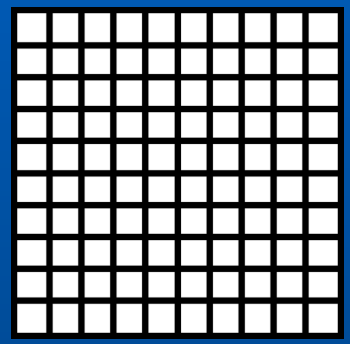
        → Typically reconstructed to zero; yielding compression! ☺

7. The JPEG Image Compression Standard

7.3 Entropy Coding

- **Entropy coding stage of JPEG compression…**
  - Relates to the task of *source coding* the quantized DCT coeffs
    - i.e. deploying pattern substitution methods towards data minimisation
- **Two choices of source coding methods…**
  - *Huffman coding*
    - The approach described here for JPEG compression!
  - *Arithmetic coding*
    - Tends to give better compression than Huffman
    - However, many JPEG implementations don't use it
      - The gain of 2%-10% is not very significant, and the algorithms are covered by US/Japanese patents
- **Note:** DCCs and ACCs are treated separately as they have very different characteristics!

7.3.1 DC Coefficients



- **Entropy coding DC coeffs…**

  1. DCCs of successive blocks are predictively encoded
     - Current DCC has the value of the previous block subtracted from it
       - Only the difference (‘***prediction residual***’) is retained for entropy encoding

         → Exploits the spatial correlation between neighbouring blocks towards a coding advantage

  2. DCC prediction residuals Huffman coded as follows…

7.3.1 DC Coefficients (cont.)

- **Huffman coding DCC residuals in JPEG**
  - Assuming 8-bit pixels are used → DCCs have range [-1024,1016]
    - => residual DCC values have range [-2040, 2040]
    - => substituting patterns for each value would require a table of over 4000 entries! → not implementation friendly!
  - Alternative approach employed…
    - Values first coded as (*length*, *magnitude*) pairs where…
      - *length* indicates the #bits used to encode the value (Huffman coded)
      - *magnitude* is the actual value (directly coded)

N.B. DCC range [-1024,1016] → requires 11-bit word length

→ Which actually covers range [-1024,1023]

Residual DCC range [-2040, 2040] → requires 12-bit word length

→ Which actually covers range **[-2047, 2047]**

7.3.1 DC Coefficients (cont.)

- **Details of Huffman coding DCC residuals in JPEG**
  - The range [-2047, 2047] is divided up into **12** *length* (or '*size*') categories, where the i^{th} category contains set of values which can be represented with *i* bits
    - → e.g. 4 values contained in cat. 2 can be represented by 2-bits, etc.
    - → e.g. 8 values contained in cat. 3 can be represented by 3-bits, etc.

    Allows the transmission of lower values without overhead of long bitwords ☺
  - *length* indicator → Huffman coded
    - i.e. represented by a static pattern
  - *magnitude* coded in straightforward binary

DC Coef Difference	Size
0	0
-1,1	1
-3,-2,2,3	2
-7,...,-4,4,...,7	3
-15,...-8,8,...,15	4
□	□
-1023,...-512,512,...,1023	10
-2047,...-1024,1024,...2047	11

- Now only 12 Huffman patterns used (as opposed to 4000+)…
  - Smaller implementation overhead! ☺

7.3.1 DC Coefficients (cont.)

- **Details of Huffman coding DCC residuals in JPEG (cont.)**
  - Typical Huffman *length* codes + corresponding binary *magnitude* codes…

N.B. Negative *magnitudes* coded in ones-complement form

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,...,-4,4,...,7	3	100	000,...,011,100,...111
-15,...-8,8,...,15	4	101	0000,...,0111,1000,...,1111
□	□	□	□
-1023,...-512,512,...,1023	10	1111 1110	00 0000 0000,...,11 1111 1111
-2047,...-1024,1024,...2047	11	1 1111 1110	000 0000 0000,...,111 1111 1111

  - E.g. value 7 coded in (*length*,*magnitude*) form = (3,111), i.e. (100,111)
  - E.g. value -15 coded in (*length*,*magnitude*) form = (101,0000)

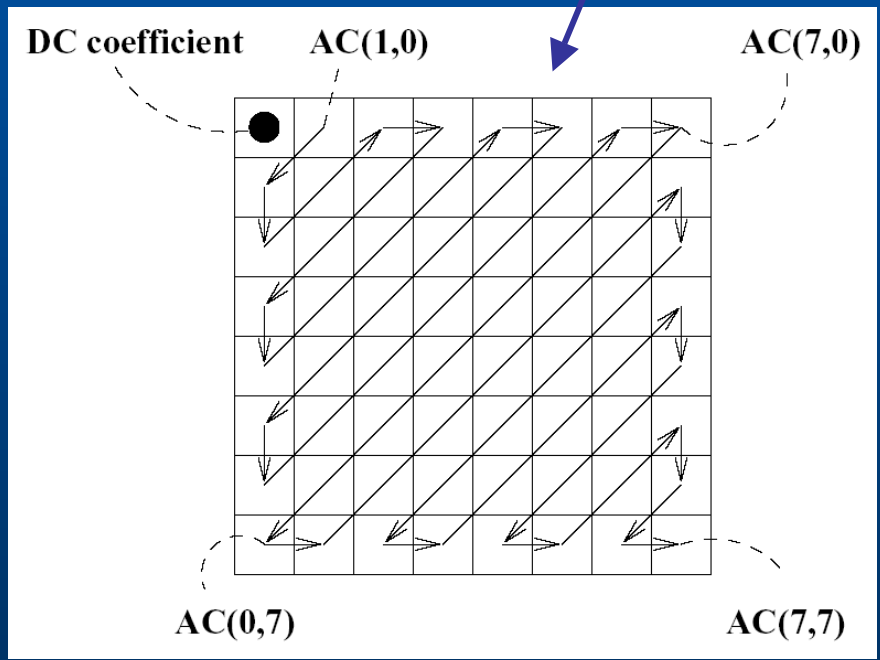
7.3.2 AC Coefficients

- **Entropy coding AC coeffs**
  - Similar to entropy coding of DCCs
    - i.e. length/magnitude style coding + Huffman
  - However, first…
    - ACCs of of a given block are transformed into a new set of events using RLE
  - Specifically…
    - Quantized coeffs are read out as a 1-D sequence in zig-zag scan fashion…

      → exploits long runs of zeroes along this path

- Coupled with strong quantization of higher-freq coeffs

  => Zig-zag → long runs of zeros



7.3.2 AC Coefficients (cont.)

- **Entropy coding AC coeffs (cont.)**
  - Following detection of zero 'runs' in the scan…
    - Each non-zero ACC is encoded as a (***run/length***, ***magnitude***) combination
      - ACC range {-1023, 1023} is divided up into **10** *length* categories, where the i^{th} category contains set of values which can be represented with *i* bits
      - ***run*** = # zero coeffs prior to the current non-zero coeff
    - And as before…
      - ***length*** indicates the #bits used to encode the value
      - ***magnitude*** is the actual value directly encoded
    - Value of ***run/length*** is then Huffman coded

AC coeff	Size
0	0
-1,1	1
-3,-2,2,3	2
-7,...,-4,4,...,7	3
-15,...-8,8,...,15	4
□	□
-1023,...-512,512,...,1023	10

7.3.2 AC Coefficients (cont.)

- **Entropy coding AC coeffs (cont.)**

AC coeff	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,...,-4,4,...,7	3	100	000,...,011,100,...111
-15,...-8,8,...,15	4	101	0000,...,0111,1000,...,1111
□	□	□	□
-1023,...-512,512,...,1023	10	1111 1110	00 0000 0000,...,11 1111 1111

  - E.g. AC coeff with value **-4**, preceded by **1 zero**
    - -4 falls into category3 (i.e. 3 bits allocated)
    - Binary code for **-4 = 011**
    - (*run/length*, *magnitude*) event thus represented by **(1/3, 011)**
    - Assuming Huffman code for symbol (1/3) is **1111001**

      → Overall code for this event **= 1111001,001**

7.3.2 AC Coefficients (cont.)

- **Summary…**
  - Quantization of DCT coeffs…
    - Primary source of compression in JPEG (lossy)
    - Renders many high frequency coefficients to zero
  - Entropy encoding (lossless)…
    - DCC prediction + Huffman
    - ACC zig-zag scanning + RLE, + Huffman
      - Also contributes significantly to reduction of bit rate

7.3.3 JPEG Coding of Colour Images

- **Image inter-component redundancy**
  - Images: Typically consist of parallel bands ('*channels*') of information (e.g. RGB)
  - Would be straightforward for JPEG to consider each separately
    - i.e. code each as a monochrome image
  - However redundancy exists between RGB channels
    - Should also be exploited, hence…
- **Image pre-processing (prior to JPEG encoding)**
  - Images transformed to **YUV** space
    (Inter-component correlation substantially reduced)
    - Chrominance channels (U&V) subsampled w.r.t. Y-channel
      - Not noticeable by the human eye
      - Even more bit-rate reduction!

7. The JPEG Image Compression Standard

7.4.3 Motion JPEG

- **JPEG…**
  - Developed specifically for still image compression
- **However…**
  - Also been used as a crude way for coding digital video
    - Known as '*MotionJPEG*' (MJPEG)
  - Each video sequence frame is encoded as a single baseline JPEG
    - Equivalent to I-frame only MPEG-1 video encoding (see later)
- **N.B.**
  - MJPEG not a mode described in the original JPEG standard
    - i.e. bit stream syntax not standardised

    → May vary from implementation to implementation

End of Chapter 7!