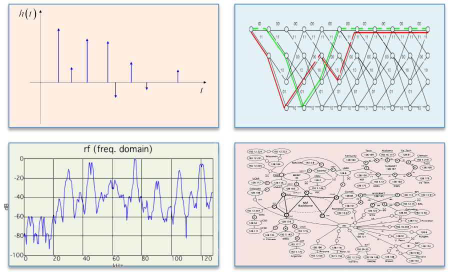


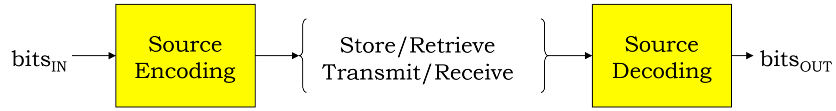
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Spring 2009
Lecture #25

- Lossless vs. lossy compression
- Perceptual models
- Selecting info to eliminate
- Quantization and entropy encoding

Lossless vs. Lossy Compression



- Huffman and LZW encodings are *lossless*, i.e., we can reconstruct the original bit stream exactly: $bits_{OUT} = bits_{IN}$.
  - What we want for "naturally digital" bit streams (documents, messages, datasets, …)
- Any use for *lossy* encodings: $bits_{OUT} \approx bits_{IN}$?
  - "Essential" information preserved
  - Appropriate for sampled bit streams (audio, video) intended for human consumption via imperfect sensors (ears, eyes).

Perceptual Coding

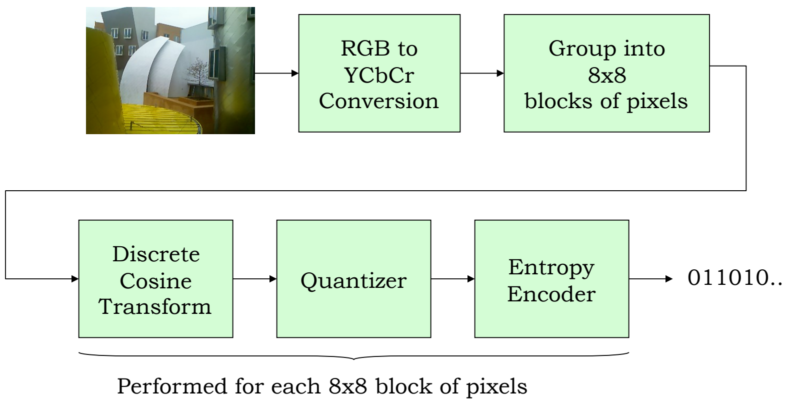
- Start by evaluating input response of bitstream consumer (eg, human ears or eyes), i.e., how consumer will perceive the input.
  - Frequency range, amplitude sensitivity, color response, …
  - Masking effects
- Identify information that can be removed from bit stream without perceived effect, e.g.,
  - Sounds outside frequency range, or masked sounds
  - Visual detail below resolution limit (color, spatial detail)
  - Info beyond maximum allowed output bit rate
- Encode remaining information efficiently
  - Use DCT-based transformations
  - Quantize DCT coefficients
  - Entropy code (eg, Huffman encoding) results

Perceptual Coding Example: Images

- Characteristics of our visual system ⇒ opportunities to remove information from the bit stream
  - More sensitive to changes in luminance than color ⇒ spend more bits on luminance than color (encode separately)
  - More sensitive to large changes in intensity (edges) than small changes ⇒ quantize intensity values
  - Less sensitive to changes in intensity at higher spatial frequencies ⇒ use larger quanta at higher spatial frequencies
- So to perceptually encode image, we would need:
  - Intensity at different spatial frequencies
  - Luminance (grey scale intensity) separate from color intensity

JPEG Image Compression

JPEG = Joint Photographic Experts Group

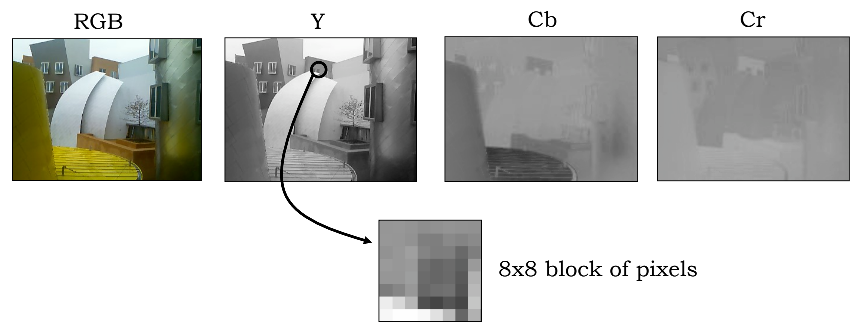


YCbCr Color Representation

JPEG-YCbCr (601) from "digital 8-bit RGB"

Y = 0.299*R + 0.587*G + 0.114*B
Cb = 128 - 0.168736*R - 0.331264*G + 0.5* B
Cr = 128 + 0.5*R - 0.418688*G - 0.081312*B

All values are in the range 0 to 255



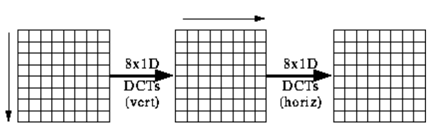
http://en.wikipedia.org/wiki/YCrCb

2D Discrete Cosine Transform (DCT2)

$$X_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x_{mn} \cos\left[\frac{\pi}{M}\left(m+\frac{1}{2}\right)p\right]\cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)q\right] \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M} & p = 0 \\ \sqrt{2/M} & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N} & q = 0 \\ \sqrt{2/N} & 1 \le q \le N-1 \end{cases}$$

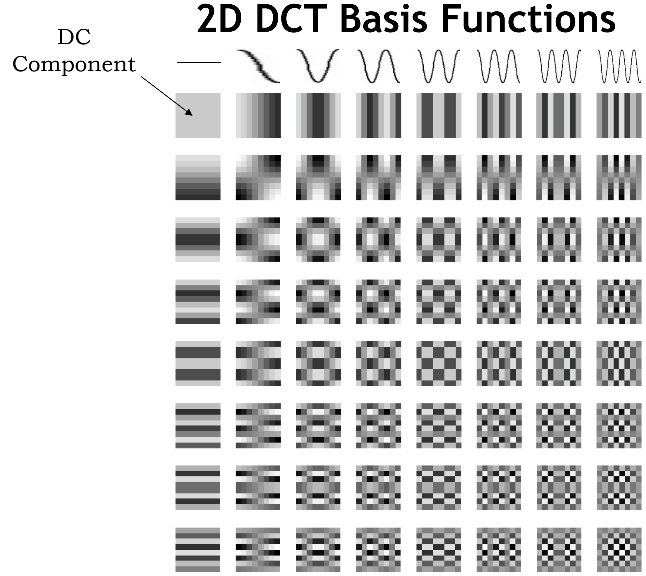
$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right]$$ 1D DCT (Type 2)



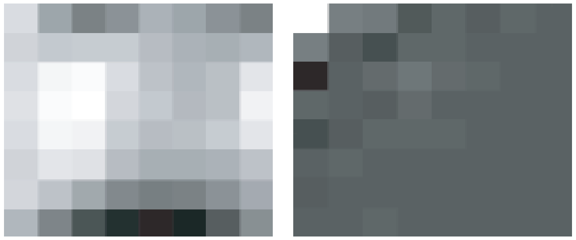
2D DCT

http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html

2D DCT Basis Functions



DCT Example



40	24	15	19	28	24	19	15
38	34	35	35	31	28	27	29
40	47	49	40	33	29	32	43
42	49	50	39	34	30	32	46
40	47	46	35	31	32	35	43
38	43	42	31	27	27	28	33
39	33	25	27	14	15	19	26
29	16	6	1	-4	0	7	18

Pixels

239	32	27	-12	3	-5	3	1
34	-3	-19	6	3	0	-1	1
-70	2	8	23	9	6	-1	-1
5	0	-6	11	-2	0	-1	1
-17	-3	6	6	3	-1	0	0
2	4	2	2	1	-2	0	1
-3	0	0	-1	-1	-1	0	0
1	-1	3	1	0	0	0	0

DCT Coefficients

Quantization (the "lossy" part)

Divide each of the 64 DCT coefficients by the appropriate quantizer value (Q_{lum} for Y, Q_{chr} for Cb and Cr) and round to nearest integer ⇒ many 0 values, many of the rest are small integers.

$$Q_{lum} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \quad Q_{chr} = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

Note fewer quantization levels in Q_{chr} and at higher spatial frequencies. Change "quality" by choosing different quantization matrices.

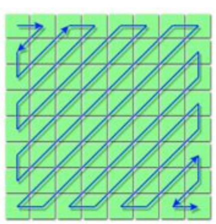
Quantization Example

239	32	27	-12	3	-5	3	1
34	-3	-19	6	3	0	-1	1
-70	2	8	23	9	6	-1	-1
5	0	-6	11	-2	0	-1	1
-17	-3	6	6	3	-1	0	0
2	4	2	2	1	-2	0	1
-3	0	0	-1	-1	-1	0	0
1	-1	3	1	0	0	0	0

DCT Coefficients

15	3	3	-1	0	0	0	0
3	0	-1	0	0	0	0	0
5	0	0	1	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

Quantized Coefficients

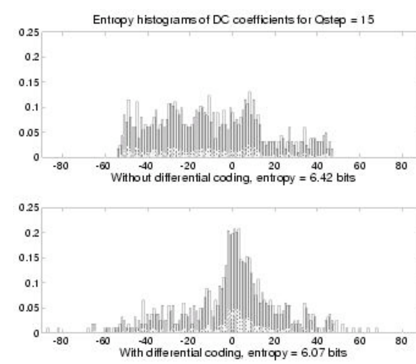


Visit coeffs in order of increasing spatial frequency ⇒ tends to create long runs of 0s towards end of list:

15 3 3 5 0 3 -1 -1 0 0 -1 0 0 0 0 0 0 1 0…

Entropy Encoding

Use differential encoding for first coefficient (DC value) -- encode difference from DC coeff of previous block.



Encode DC coeff as (N), coeff

N is Huffman encoded, differential coeff is an N-bit string

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

Encode AC coeffs as (run,N),coeff

Run = length of run of zeros preceding coefficient
N = number of bits of coefficient
Coeff = N-bit representation for coefficient

(run,N) pair is Huffman coded
(0,0) is EOB meaning remaining coeffs are 0
(15,0) is ZRL meaning run of 16 zeros

(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		⋮	
(3,1)	31	111010	(ZRL)	F0	11111111001
(4,1)	41	111011		⋮	

Entropy Encoding Example

Quantized coeffs:

```
15 3 3 5 0 3 -1 -1 0 0 -1 0 0 0 0 0 0 1 0…
```

DC: (N),coeff, all the rest: (run,N),coeff

```
(4) 15 (0,2) 3 (0,2) 3 (0,3) 5 (1,2) 3 (0,1) -1
(0,1) -1 (2,1) -1 (6,1) 1 EOB
```

Encode using Huffman codes for N and (run,N):

1011111011101111001011101111000000111000111101111010

Result: 8x8 block of 8-bit pixels (512 bits) encoded as 52 bits

10x compression!

To read more see "The JPEG Still Picture Compression Standard" by Gregory K. Wallace
http://white.stanford.edu/~brian/psy221/reader/Wallace.JPEG.pdf

JPEG Results

The source image (left) was converted to JPEG (q=50) and then compared, pixel-by-pixel. The error is shown in the right-hand image (darker = larger error).

http://en.wikipedia.org/wiki/JPEG