6.02 Spring 2010
Lecture #23

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

Perceptual Coding

- Start by evaluating input response of bitstream consumer (e.g., 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 bitstream 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 (real instead of complex)
  - Quantize DCT coefficients
  - Entropy code (e.g., Huffman encoding) results

Perceptual Coding Example: Images

- Characteristics of our visual system ⇒ opportunities to remove information from the bitstream
  - 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

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} ≠ bits_{IN}?
  - “Essential” information preserved
  - Appropriate for sampled bit streams (audio, video) intended for human consumption via imperfect sensors (ears, eyes).
JPEG Image Compression

JPEG = Joint Photographic Experts Group

YCbCr Color Representation

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

\[
\begin{align*}
Y &= 16 + 0.299 \times R + 0.587 \times G + 0.114 \times B \\
Cb &= 128 - 0.168736 \times R - 0.331264 \times G + 0.5 \times B \\
Cr &= 128 + 0.5 \times R - 0.418688 \times G - 0.081312 \times B
\end{align*}
\]

All values are in the range 16 to 235

2D Discrete Cosine Transform (DCT2)

\[
X_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{mn} \cos \left( \frac{\pi}{M} \left( m + \frac{1}{2} \right) \right) \cos \left( \frac{\pi}{N} \left( n + \frac{1}{2} \right) \right) \\
\alpha_p = \begin{cases} 
1/\sqrt{M} & p = 0 \\
\sqrt{2/M} & 1 \leq p \leq M-1
\end{cases} \\
\alpha_q = \begin{cases} 
1/\sqrt{N} & q = 0 \\
\sqrt{2/N} & 1 \leq q \leq N-1
\end{cases}
\]

\[
X_i = \alpha_i \sum_{k=0}^{K-1} x_k \cos \left( \frac{\pi}{K} \left( n + \frac{1}{2} \right) \right)
\]

1D DCT (Type 2)

2D DCT

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

Ms. Söderberg as guest at 50th Annual Conference of the Society for Imaging Science and Technology!
2D DCT Basis Functions

DC Component

DCT Example

Lenna DCT Coeffs from each 8x8 block

DC coeffs carry a lot of the picture info!

Low freqs contain major edge info (note more H and V...)

High freqs contain fine detail (e.g., texture of feather)

Quantization (the “lossy” part)

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

Note fewer quantization levels in $Q_{\text{chr}}$ and at higher spatial frequencies. Change “quality” by choosing different quantization matrices.
### Quantization Example

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

-14
-13 13
0 -1 4
-2 -2 6 0
0 2 -1 0 -1
0 -1 -1 1 0 0
0 0 0 -1 0 0 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

<table>
<thead>
<tr>
<th>DC Coef Difference</th>
<th>Size</th>
<th>Typical Huffman codes for Size</th>
<th>Additional Bits (in binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>010</td>
<td>0 1</td>
</tr>
<tr>
<td>-1, -2, 2, 3</td>
<td>2</td>
<td>011</td>
<td>00, 01, 10, 11</td>
</tr>
<tr>
<td>-7, -8, 9, 10, 11</td>
<td>3</td>
<td>101</td>
<td>000, 010, 100, 111</td>
</tr>
<tr>
<td>-15, -16, 17, 18, 19</td>
<td>4</td>
<td>101 (0000, 0100, 1000, 1110)</td>
<td></td>
</tr>
<tr>
<td>-32, -33, 34, 35, 36</td>
<td>5</td>
<td>101 (00000, 00100, 01000, 10000, 11100)</td>
<td></td>
</tr>
<tr>
<td>1023, -1024, 1025</td>
<td>10</td>
<td>1111 1110</td>
<td>00 00000 000000 11 1111 1111</td>
</tr>
<tr>
<td>-2047, -1024, 1025</td>
<td>11</td>
<td>1111 1110</td>
<td>000 00000 0000000 11 1111 1111</td>
</tr>
</tbody>
</table>

### 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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,1)</td>
<td>01</td>
<td>00</td>
<td>(64)</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>(12)</td>
<td>02</td>
<td>01</td>
<td>(53)</td>
<td>03</td>
<td>09</td>
</tr>
<tr>
<td>(53)</td>
<td>03</td>
<td>09</td>
<td>(33)</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>(128)</td>
<td>00</td>
<td>1000</td>
<td>(41)</td>
<td>04</td>
<td>0101</td>
</tr>
<tr>
<td>(41)</td>
<td>04</td>
<td>0101</td>
<td>(156)</td>
<td>04</td>
<td>1010</td>
</tr>
<tr>
<td>(156)</td>
<td>04</td>
<td>1010</td>
<td>(206)</td>
<td>05</td>
<td>10100</td>
</tr>
<tr>
<td>(206)</td>
<td>05</td>
<td>10100</td>
<td>(206)</td>
<td>05</td>
<td>10100</td>
</tr>
<tr>
<td>(0)</td>
<td>00</td>
<td>0000</td>
<td>(206)</td>
<td>05</td>
<td>10100</td>
</tr>
<tr>
<td>(101)</td>
<td>101</td>
<td>101</td>
<td>(101)</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>(206)</td>
<td>05</td>
<td>10100</td>
<td>(206)</td>
<td>05</td>
<td>10100</td>
</tr>
<tr>
<td>(206)</td>
<td>05</td>
<td>10100</td>
<td>(206)</td>
<td>05</td>
<td>10100</td>
</tr>
</tbody>
</table>
Entropy Encoding Example

Quantized coeffs:

$-14 -13 13 0 -1 4 -2 6 0 0 2 -1 0 -1 0 -1 1 0 0 0 0 -1 0 0 0..$

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

(4) $-14$ (0, 4) $-13$ (0, 4) $13$ (1, 1) $-1$ (0, 3) $4$ (0, 2) $-2$ (0, 2) $-2$

(0, 3) $6$ (2, 2) $2$ (0, 1) $-1$ (1, 1) $-1$ (0, 1) $1$ (0, 1) $1$ (5, 1) $-1$ EOB

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

1010001 10110010 10111101 11000 100100 0101 0101
100110 1111101110 000 11000 000 001 11110100 1010

Result: 8x8 block of 8-bit pixels (512 bits) encoded as 84 bits

6x compression!

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).

JPEG Forensics


Bust resized
Eye & tooth color changed
Skin color changed
Arm reshaped
Background blurred
Handbag removed (missing grout!)
Dress color changed