Perceptual Coding

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

Lossless vs. Lossy Compression

- Huffman and LZW encodings are lossless, i.e., we can reconstruct the original bit stream exactly: \( \text{bits}_{\text{OUT}} = \text{bits}_{\text{IN}} \).
  - What we want for “naturally digital” bit streams (documents, messages, datasets, …)
- Any use for lossy encodings: \( \text{bits}_{\text{OUT}} \approx \text{bits}_{\text{IN}} \)?
  - “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 (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 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 (e.g., 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**

- **RGB to YCbCr Conversion**
- **Group into 8x8 blocks of pixels**
- **Discrete Cosine Transform**
- **Quantizer**
- **Entropy Encoder**

Performed for each 8x8 block of pixels

---

**YCbCr Color Representation**

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

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

All values are in the range 0 to 255
2D Discrete Cosine Transform (DCT2)

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

where:

\[ \alpha_p = \begin{cases} 1/\sqrt{M} & p = 0 \\ \sqrt{2/M} & 1 \leq p \leq M - 1 \end{cases} \]

\[ \alpha_q = \begin{cases} 1/\sqrt{N} & q = 0 \\ \sqrt{2/N} & 1 \leq q \leq N - 1 \end{cases} \]

1D DCT (Type 2)

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

2D DCT Basis Functions

DC Component
DCT Example

Pixels

<table>
<thead>
<tr>
<th>40</th>
<th>24</th>
<th>15</th>
<th>19</th>
<th>28</th>
<th>24</th>
<th>19</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>34</td>
<td>35</td>
<td>35</td>
<td>31</td>
<td>28</td>
<td>27</td>
<td>29</td>
</tr>
<tr>
<td>40</td>
<td>47</td>
<td>49</td>
<td>40</td>
<td>33</td>
<td>29</td>
<td>32</td>
<td>43</td>
</tr>
<tr>
<td>42</td>
<td>49</td>
<td>50</td>
<td>39</td>
<td>34</td>
<td>30</td>
<td>32</td>
<td>46</td>
</tr>
<tr>
<td>40</td>
<td>47</td>
<td>46</td>
<td>35</td>
<td>31</td>
<td>32</td>
<td>35</td>
<td>43</td>
</tr>
<tr>
<td>38</td>
<td>43</td>
<td>42</td>
<td>31</td>
<td>27</td>
<td>27</td>
<td>28</td>
<td>33</td>
</tr>
<tr>
<td>39</td>
<td>33</td>
<td>25</td>
<td>27</td>
<td>14</td>
<td>15</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td>29</td>
<td>16</td>
<td>6</td>
<td>1</td>
<td>-4</td>
<td>0</td>
<td>7</td>
<td>18</td>
</tr>
</tbody>
</table>

DCT Coefficients

<table>
<thead>
<tr>
<th>239</th>
<th>32</th>
<th>27</th>
<th>-12</th>
<th>3</th>
<th>-5</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>-3</td>
<td>-19</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-70</td>
<td>2</td>
<td>8</td>
<td>23</td>
<td>9</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-6</td>
<td>11</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-17</td>
<td>-3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 ⇒ many 0 values, many of the rest are small integers.

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


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

<table>
<thead>
<tr>
<th>239 32 27 -12 3 -5 3 1</th>
<th>15 3 3 -1 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>34 -3 -19 6 3 0 -1 1</td>
<td>3 0 -1 0 0 0 0 0</td>
</tr>
<tr>
<td>-70 2 8 23 9 6 -1 -1</td>
<td>5 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>5 0 -6 11 -2 0 -1 1</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>-17 -3 6 6 3 -1 0 0</td>
<td>-1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>2 4 2 2 1 -2 0 1</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>-3 0 0 -1 -1 -1 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 -1 3 1 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

DCT Coefficients  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 1 0...

Entropy Encoding

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

![Entropy histograms of DC coefficients for Codap = 15](http://cnx.org/content/m11096/latest/)
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>-1,1</td>
<td>1</td>
<td>010</td>
<td>0,1</td>
</tr>
<tr>
<td>-3,-2,2,3</td>
<td>2</td>
<td>011</td>
<td>00,01,10,11</td>
</tr>
<tr>
<td>-7,-6,-5,-4,-3,7</td>
<td>3</td>
<td>100</td>
<td>000,011,100,111</td>
</tr>
<tr>
<td>-15,-8,8,-15</td>
<td>4</td>
<td>101</td>
<td>0000,0111,1000,1111</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>-1023,...,-512,512,...,1023</td>
<td>10</td>
<td>1111 1110</td>
<td>00 0000 0000,...,11 1111 1111</td>
</tr>
<tr>
<td>-2047,...,-1024,1024,...,2047</td>
<td>11</td>
<td>1 1111 1110</td>
<td>00 0000 0000,...,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

(run,N) pair is Huffman coded
(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>(6,1)</td>
<td>01</td>
<td>00</td>
<td>(0,5)</td>
<td>05</td>
<td>1111000</td>
</tr>
<tr>
<td>(6,2)</td>
<td>02</td>
<td>01</td>
<td>(1,3)</td>
<td>13</td>
<td>1111001</td>
</tr>
<tr>
<td>(6,3)</td>
<td>03</td>
<td>100</td>
<td>(5,1)</td>
<td>51</td>
<td>1111010</td>
</tr>
<tr>
<td>(EOB)</td>
<td>00</td>
<td>1010</td>
<td>(6,1)</td>
<td>61</td>
<td>1111011</td>
</tr>
<tr>
<td>(6,4)</td>
<td>04</td>
<td>1011</td>
<td>(6,7)</td>
<td>67</td>
<td>11111000</td>
</tr>
<tr>
<td>(1,1)</td>
<td>11</td>
<td>1100</td>
<td>(2,2)</td>
<td>22</td>
<td>11111001</td>
</tr>
<tr>
<td>(6,5)</td>
<td>05</td>
<td>11010</td>
<td>(7,1)</td>
<td>71</td>
<td>11111010</td>
</tr>
<tr>
<td>(1,2)</td>
<td>12</td>
<td>11011</td>
<td>(1,4)</td>
<td>14</td>
<td>111110110</td>
</tr>
<tr>
<td>(2,1)</td>
<td>21</td>
<td>11100</td>
<td></td>
<td></td>
<td>111110110</td>
</tr>
<tr>
<td>(3,1)</td>
<td>31</td>
<td>111010              (ZRL)</td>
<td>30</td>
<td>11111110001</td>
<td></td>
</tr>
<tr>
<td>(4,1)</td>
<td>41</td>
<td>111011</td>
<td></td>
<td></td>
<td>11111110001</td>
</tr>
</tbody>
</table>
Entropy Encoding Example

Quantized coeffs:
15 3 3 5 0 3 -1 -1 0 0 -1 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):
101111101110111100101110111000011100111111010

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

6.02 Spring 2008 Perceptual Coding, Slide 15

Big Ideas in 6.02

Multiple representations
• time vs. frequency

Communicating Information
• managing signal vs. noise
• encode as voltages, amplitude, phase
• power, bandwidth, channel capacity

Digital Abstraction
• errors accumulate
• continuous $\rightarrow$ discrete
• restoration of values

Information & Encodings
• info can be measured: entropy
• add redundancy $\rightarrow$ correct errors
• remove redundancy $\rightarrow$ compression

Systems engineering
• modularity, composable behaviors
• uniform representations
• latency, throughput
• layered architectures