JPEG Image Compression

Color Transform

- JPEG capable of using any color model
  - RGB, HSI, CMY, YUV, YCbCr
- RGB, HSI, CMY spread useful visual information across 3 components
  - Hard to select information to discard
- In YCbCr, most of useful information is in Y space
  - Some of Cb and Cr can be discarded
- JPEG can transform color model to improve compression

*Thanks to Greg Stitt for providing/assembling much of the included information.*
JPEG Image Compression
RGB vs. YCbCr

- RGB
  - R - Red
  - G - Green
  - B - Blue

- YCbCr
  - Y - Luma (Brightness)
  -Cb - Cb Chroma (Blue minus Luma)
  -Cr - Cr Chroma (Red minus Luma)

Luma - Amount of light that passes through or is emitted from a particular area
Chroma - Difference between a color and a chosen reference color of the same luminous intensity
JPEG Image Compression

Down Sampling

- Luminance more important than chrominance
  - Store less information for chrominance
- Example: 1000x1000 image
  - Store 1000x1000 luminance pixels
  - Store 500x500 chrominance pixels
  - Each chrominance pixel corresponds to actual 2x2 block
  - => Each 2x2 block requires 6 pixels instead of 12

JPEG Image Compression

DCT - Discrete Cosine Transform

- Convert spatial/time domain to frequency domain
  - Time domain plots how some value changes over time
  - Frequency domain decomposes signal from time domain into sine and cosine waves with various amplitudes
- Spatial frequency
  - The frequency over distance (instead of time)
  - The frequency domain is a space in which the value at each image position represents the amount that the intensity values in the image vary over a specific distance
  - Any image can be formed by adding different spatial frequencies
JPEG Image Compression
Frequency Domain

Output of DCT is array of DCT coefficients representing different strengths of each frequency
- For JPEG, DCT is done on 8x8 blocks
- 8x8 DCT coefficients

Top left is low frequencies
- (0,0) is DC value, or average, of all pixels

Bottom right is strength of cosine wave alternating from max to min every pixel
JPEG Image Compression

DCT – Discrete Cosine Transform

DCT Equation:

\[ F(u, v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^{8} \sum_{y=0}^{8} D[x, y] \cos \left( \frac{\pi (2x + 1) u}{16} \right) \cos \left( \frac{\pi (2y + 1) v}{16} \right) \]

\[ C(h) = \begin{cases} 
\frac{1}{\sqrt{2}}, & h = 0 \\
1, & \text{otherwise}
\end{cases} \]

- \( D[x,y] \) = original pixel value at location \( x, y \)
- \( F(u,v) \) = output coefficient at location \( x, y \)

DCT operates on 8x8 blocks,
- \( u, v, x, \) and \( y \) only range in value from 0 to 7
- Precompute 64 possible cosine values and store in an 8x8 table,

Rewrite DCT Equation:

\[ F(u, v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^{8} \sum_{y=0}^{8} D[x, y] \cos \left[ x \cdot u \right] \cdot \cos \left[ y \cdot v \right] \]

- \( D[x,y] \) = original pixel value at location \( x, y \)
- \( F(u,v) \) = output coefficient at location \( x, y \)
JPEG Image Compression

**DCT – Discrete Cosine Transform**

- **Reason for using DCT**
  - Important to separate frequencies because human eye is more sensitive to low frequency information
  - Can discard some of high-frequency information

---

JPEG Image Compression

**Quantization**

- Divides each DCT coefficient by some value to reduce total number of bits
  - Many of the high frequency coefficients become 0
  - Other information is lost from integer division
  - This is where the loss is JPEG comes from

![Matrix representation of DCT coefficients and quantization process]
JPEG Image Compression

Quantization

- Different Types
  - Constant amount
  - Varying amount for different frequencies
  - Separate values for luminance and chrominance
    - Larger amount for chrominance

- Quantization table stored in JPEG file
  - Generally a sample table is used
  - Different “Quality” jpegs use different quantization tables

Example quantization tables:

<table>
<thead>
<tr>
<th>The Luminance Quantization Table q(u, v)</th>
<th>The Chrominance Quantization Table q(u, v)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 12 10 16 24 40 51 61</td>
<td>17 18 24 97 99 99 99 99</td>
</tr>
<tr>
<td>12 12 14 19 26 58 60 55</td>
<td>18 21 26 99 99 99 99 99</td>
</tr>
<tr>
<td>14 13 16 24 40 57 69 56</td>
<td>24 26 56 99 99 99 99 99</td>
</tr>
<tr>
<td>14 17 23 29 51 87 89 65</td>
<td>47 66 99 99 99 99 99 99</td>
</tr>
<tr>
<td>30 22 27 96 90 109 103 77</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
<tr>
<td>24 35 65 64 81 104 113 92</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
<tr>
<td>49 64 70 97 100 121 120 101</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
<tr>
<td>72 52 95 98 112 130 133 99</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
</tbody>
</table>

|----------------------------------------|-------------------------------------------|
JPEG Image Compression
Quantization

Qual = 100
Size = 60.8 KB

Qual = 75
Size = 12.8 KB

Qual = 50
Size = 6.9 KB

Qual = 25
Size = 2.6 KB

Qual = 10
Size = 1.8 KB

Qual = 1
Size = 1.6 KB
After quantization, many DCT coefficients are 0, especially for high frequencies
- High frequencies appear in bottom right of 8x8 block
- Grouping these 0s together makes compression better
- Zig-zag converts 8x8 block into 16x1, in the following order:

Run length encoding
- For each non-zero DCT coefficient, JPEG records the number of zeros that preceded the number, the number of bits needed to represent the number's amplitude, and the amplitude itself.
JPEG Image Compression

Encoding

- After run length encoding, Huffman encoding further compresses the 8x8 block
- After compression, block is written to file along with end of block marker

JPEG Image Decompression

- Basically opposite of encode
  - Decompress, Dequantize, Inverse DCT, Up-sampling, color transform