Overview of JPEG

What is JPEG?
- Works with colour and greyscale images, e.g., satellite, medical, ...

Motivation

- The compression ratio of lossless methods (e.g., Huffman, Arithmetic, LZW) is not high enough for image and video compression, especially when the distribution of pixel values is relatively flat.
- JPEG uses transform coding, it is largely based on the following observations:
  - Observation 1: A large majority of useful image contents change relatively slowly across images, i.e., it is unusual for intensity values to alter up and down several times in a small area, for example, within an 8 x 8 image block. Translate this into the spatial frequency domain, it says that, generally, lower spatial frequency components contain more information than the high frequency components which often correspond to less useful details and noises.
  - Observation 2: Psychophysical experiments suggest that humans are more receptive to loss of higher spatial frequency components than loss of lower frequency components.

JPEG Encoding

- There are FOUR (4) modes:
  - Lossy sequential DCT (supported by all)
  - Extended lossy DCT (further enhancements)
  - Lossless mode
  - Hierarchical mode
- The baseline process takes the following techniques: Block, MCU, FDCT, RLE and Huffman. We shall discuss the details later.

JPEG Decoding

How to decode the JPEG Pictures?
Reverse the Process!
- i.e. The flow arrow will point backwards
- This will imply that the transform function needs to have an equivalent inverse.
Image Preparation

A source image consists of 1..255 components/planes

- Color has its 3 components with equal resolution

Sampling factors

- X, Y are maximum value
- H = Horizontal sampling ratio,
- V = Vertical sampling ratio
- Hmax = Horizontal sampling ratio,
- Vmax = Vertical sampling ratio

\[
X_i = \frac{X \times H_i}{H_{\text{max}}} \\
Y_i = \frac{Y \times V_i}{V_{\text{max}}}
\]

Interleaving of data units

- Non-interleaved order of data
- Strictly Left-to-Right and Top-to-Bottom

Example:

- Given X = 512, Y=512, H_{\text{max}} = 4 \rightarrow V_{\text{max}} = 2
- Level 0: H_0 = 4, V_0=1
  \[X_0 = \frac{X \times H_0}{H_{\text{max}}} = \frac{512 \times 4}{4} = 512\]
  \[Y_0 = \frac{Y \times V_0}{V_{\text{max}}} = \frac{512 \times 1}{2} = 256\]
- Level 1: H_1 = 2, V_1=2
  \[X_1 = \frac{X_0 \times H_1}{H_{\text{max}}} = \frac{512 \times 2}{4} = 256\]
  \[Y_1 = \frac{Y_0 \times V_1}{V_{\text{max}}} = \frac{256 \times 2}{2} = 256\]
- Level 2: H_2 = 1, V_2=1
  \[X_2 = \frac{X_1 \times H_2}{H_{\text{max}}} = \frac{256 \times 1}{4} = 128\]
  \[Y_2 = \frac{Y_1 \times V_2}{V_{\text{max}}} = \frac{256 \times 1}{2} = 256\]
Interleaving of data units

MCU (Minimum Coded Units) = combination of interleaved data units of different components.

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \vline & 4 & 5 \\
0 & 1 & 2 & 3 \vline & 4 & 5 \\
0 & 1 & 2 \vline & 4 & 5 \\
1 & 0 & 3 & 4 & 5 \\
1 & 0 & 3 & 4 & 5 \\
1 & 0 & 3 & 4 & 5 \\
1 & 0 & 3 & 4 & 5 \\
1 & 0 & 3 & 4 & 5 \\
\end{array}
\]

Note: The data units from the same component are shown with the same color.

d1: H=2, V=2;  d2: H=2, V=1;  d3: H=1, V=2;  d4: H=1, V=1

Major Steps

1. DCT (Discrete Cosine Transformation)
2. Quantization
3. Zigzag Scan
4. DPCM on DC component
5. RLE on AC Components
6. Entropy Coding

Discrete Cosine Transform (DCT)

The transform function maps the pixel value at (i,j) in the range of [1,255] to [-128, 127]

Discrete Cosine Transform (DCT):

\[
F(u,v) = \frac{1}{\sqrt{8}} \sum_{i=0}^{7} \sum_{j=0}^{7} \frac{\cos \left(\frac{(2i+1)\pi}{16} u\right) \cos \left(\frac{(2j+1)\pi}{16} v\right)}{16} \cdot f(i,j)
\]

\[
A(\xi) = \begin{cases} 
\frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\
\frac{1}{2} & \text{otherwise}
\end{cases}
\]

Inverse Discrete Cosine Transform (IDCT):

\[
h(i,j) = \frac{1}{\sqrt{8}} \sum_{u=0}^{7} \sum_{v=0}^{7} A(\xi) A(\eta) \cos \left(\frac{(2i+1)\pi}{16} u\right) \cos \left(\frac{(2j+1)\pi}{16} v\right) \cdot F(u,v)
\]

\[
A(\xi) = \begin{cases} 
\frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\
\frac{1}{2} & \text{otherwise}
\end{cases}
\]
**Discrete Cosine Transform (DCT)**

**Example:**

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>8x8 DCT</th>
<th>64x64 DCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

**The 64 (8 x 8) DCT basis functions:**

<table>
<thead>
<tr>
<th>u</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

**Factoring reduces problem to a series of 1D DCTs:**

\[ F[u, v] = \frac{1}{2} \sum_j A(u) \cos \left( \frac{2\pi}{16} j \right) G[v, j] \]

\[ G[v, j] = \frac{1}{2} \sum_i A(v) \cos \left( \frac{2\pi}{16} i \right) f[i, j] \]

**Most software implementations use fixed point arithmetic. Some fast implementations approximate coefficients so all multiplies are shifts and adds.**

**Quantization**

**Formula:**

\[ F'[u, v] = \text{round} \left( \frac{F[u, v]}{q[u, v]} \right) \]

**Why?**

To reduce number of bits per sample

*Example:* \(101101 = 45\) (6 bits).

\(q[u, v] = 4\) → Truncate to 4 bits: 1011 = 11.

**Quantization error is the main source of the Lossy Compression.**

**Uniform Quantization**

Each \(F[u, v]\) is divided by the same constant \(N\).

**Non-uniform Quantization using Quantization Tables**

Eye is most sensitive to low frequencies (upper left corner), less sensitive to high frequencies (lower right corner)

**The Luminance Quantization Table \(q(u, v)\)**

<table>
<thead>
<tr>
<th>u</th>
<th>v</th>
<th>16</th>
<th>11</th>
<th>10</th>
<th>16</th>
<th>24</th>
<th>40</th>
<th>51</th>
<th>61</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td>26</td>
<td>58</td>
<td>60</td>
<td>55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>16</td>
<td>24</td>
<td>40</td>
<td>57</td>
<td>69</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>22</td>
<td>29</td>
<td>51</td>
<td>87</td>
<td>80</td>
<td>62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>22</td>
<td>37</td>
<td>56</td>
<td>68</td>
<td>109</td>
<td>103</td>
<td>77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>35</td>
<td>55</td>
<td>64</td>
<td>81</td>
<td>104</td>
<td>113</td>
<td>92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>64</td>
<td>78</td>
<td>87</td>
<td>103</td>
<td>121</td>
<td>120</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>92</td>
<td>95</td>
<td>98</td>
<td>112</td>
<td>100</td>
<td>103</td>
<td>99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Quantization

The Chrominance Quantization Table \( q(u, v) \)

<table>
<thead>
<tr>
<th></th>
<th>17</th>
<th>18</th>
<th>24</th>
<th>47</th>
<th>99</th>
<th>99</th>
<th>99</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>21</td>
<td>26</td>
<td>66</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>24</td>
<td>26</td>
<td>56</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
</tbody>
</table>

The numbers in the above quantization tables can be scaled up (or down) to adjust the so-called quality factor. Custom quantization tables can also be put in image/scan header.

Zig-zag Scan

Why?
1. To group low frequency coefficients in top of vector.
2. Maps 8 x 8 to a 1 x 64 vector

Differential Pulse Code Modulation (DPCM) on DC component

- DC component is large and varied, but often close to previous value.
- Encode the difference from previous 8 x 8 blocks \( \rightarrow \) DPCM

\[
\text{DC}_{i-1} \quad \text{DC}_i \quad \text{DC}_{i+1}
\]

DIFF(i-1,i) = DC\(_i\) - DC\(_{i-1}\)

Block i-1  Block i  Block i+1

Run Length Encode (RLE) on AC components

- 1 x 64 vector has lots of zeros in it
- Keeps skip and value, where skip is the number of zeros and value is the next non-zero component.
- Send (0,0) as end-of-block sentinel value.

Entropy Coding

- Categorize DC values into SIZE (number of bits needed to represent) and actual bits.

<table>
<thead>
<tr>
<th>Size</th>
<th>BCDC</th>
<th>Diff DC Coefficient Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>-1, 1</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>-3, -2, 2, 3</td>
</tr>
<tr>
<td>3</td>
<td>0111</td>
<td>-7, -6, -5, -4, -3</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>-15, -14, -13, -12, -11</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>-31, -30, -29, -28, -27</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>-63, -62, -61, -60, -59</td>
</tr>
</tbody>
</table>

Example: If DC value is 4, 3 bits are needed. Send off SIZE as Huffman symbol, followed by actual 3 bits.

Entropy Coding

- For AC components two symbols are used: Symbol_1: \((\text{skip}, \text{SIZE})\), Symbol_2: actual bits.
- Symbol_1 \((\text{skip}, \text{SIZE})\) is encoded using the Huffman coding, Symbol_2 is not encoded.
- Huffman Tables can be custom (sent in header) or default.
A Glance at the JPEG Bitstream

- A "Frame" is a picture, a "scan" is a pass through the pixels (e.g., the red component), a "Segment" is a group of blocks, a "Block" is an 8 x 8 group of pixels.
- Frame header:
  - sample precision
  - (width, height) of image
  - number of components
  - unique ID (for each component)
  - horizontal/vertical sampling factors (for each component)
  - quantization table to use (for each component)
- Scan header:
  - Number of components in scan
  - component ID (for each component)
  - Huffman table for each component (for each component)
- Misc. (can occur between headers)
  - Quantization tables
  - Huffman Tables
  - Arithmetic Coding Tables
  - Comments
  - Application Data

The Four JPEG Modes (Revisited)

The four JPEG modes:
- Sequential Mode
- Loss-less Mode
- Progressive Mode
- Hierarchical Mode

Note: In "Motion JPEG", Sequential JPEG is applied to each image in a video.

Sequential Mode JPEG

- Each image component is encoded in a single left-to-right, top-to-bottom scan.
- Baseline Sequential Mode, the one that we described above, is a simple case of the Sequential mode:
  - It supports only 8-bit images (not 12-bit images)
  - It uses only Huffman coding (not Arithmetic coding)

Lossless Mode JPEG

A special case of the JPEG where indeed there is no loss. Its block diagram is as below:

- It does not use DCT-based method! Instead, it uses a predictive (differential coding) method:
  - A predictor combines the values of up to three neighbouring pixels (not blocks as in the Sequential mode) as the predicted value for the current pixel, indicated by "X" in the figure on next page.
  - The encoder then compares this prediction with the actual pixel value at the position "X", and encodes the difference (prediction residual) losslessly.

It can use any one of the following seven predictors:

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>A + B - C</td>
</tr>
<tr>
<td>5</td>
<td>(A + B - C) / 2</td>
</tr>
<tr>
<td>6</td>
<td>B + (A - C) / 2</td>
</tr>
<tr>
<td>7</td>
<td>(A + B) / 2</td>
</tr>
</tbody>
</table>
Since it uses only previously encoded neighbours, the very first pixel I(0, 0) will have to use itself. Other pixels at the first row always use P1, at the first column always use P2.

Effect of Predictor (test result with 20 images is shown on next page)

Note: “2D” predictors (4-7) always do better than “1D” predictors.

Comparison with Other Lossless Compression Programs (compression ratio):

<table>
<thead>
<tr>
<th>Compression Program</th>
<th>Lena</th>
<th>football</th>
<th>F-18</th>
<th>flowers</th>
</tr>
</thead>
<tbody>
<tr>
<td>lossless JPEG</td>
<td>1.45</td>
<td>1.54</td>
<td>2.29</td>
<td>1.26</td>
</tr>
<tr>
<td>optimal lossless JPEG</td>
<td>1.49</td>
<td>1.67</td>
<td>2.71</td>
<td>1.33</td>
</tr>
<tr>
<td>compress (LZW)</td>
<td>0.86</td>
<td>1.24</td>
<td>2.21</td>
<td>0.87</td>
</tr>
<tr>
<td>gzip (Lempel-Ziv)</td>
<td>1.08</td>
<td>1.36</td>
<td>3.10</td>
<td>1.05</td>
</tr>
<tr>
<td>gzip -9 (optimal Lempel-Ziv)</td>
<td>1.08</td>
<td>1.36</td>
<td>3.13</td>
<td>1.05</td>
</tr>
<tr>
<td>pack (Huffman coding)</td>
<td>1.02</td>
<td>1.12</td>
<td>1.19</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Goal: display low quality image and successively improve.

Two ways to successively improve image:

1. **Spectral selection**: Send DC component and first few AC coefficients first, then gradually some more ACs.
2. **Successive approximation**: send DCT coefficients MSB (most significant bit) to LSB (least significant bit). (Effectively, it is sending quantized DCT coefficients first, and then the difference between the quantized and the non-quantized coefficients with finer quantization stepsize.)

(a) Down-sample by factors of 2 in each dimension, e.g., reduce 640 x 480 to 320 x 240
(b) Code smaller image using another JPEG mode (Progressive, Sequential, or Lossless).
(c) Decode and up-sample encoded image
(d) Encode difference between the up-sampled and the original using Progressive, Sequential, or Lossless.

Can be repeated multiple times.

Good for viewing high resolution image on low resolution display.