Lecture Outline

- Introduction
- Lossy predicative coding
- Transform coding
- JPEG
- JPEG 2000
Spatial Prediction

Linear Estimator:  \( \hat{f}_K = a f_F + b f_G + c f_H + d f_J \)

Optimal MSE Predictor: The coefficients are determined to minimize the mean square prediction error.
A Typical Predictive Coder

Encoder

Input Samples → $f_p$ → $e_p$ → Quantizer → $\hat{e}_p$ → Entropy Coder → $B_e$

Decoder

$B_e$ → Entropy Decoder → $\hat{e}_p$ → Predictor → $\hat{f}$ → Output Samples

Closed-loop prediction
The Delta Modulator

• The simplest linear prediction.
  \[ \hat{f}_0 = f_1 \]

• The quantizer has only two levels.
  \[ \hat{e} = \begin{cases} 
  \xi & e > 0 \\
  -\xi & e < 0 
\end{cases} \]

\( \xi \) is appropriate
\( \xi \) is too small
Error Analysis of a Lossy Predictive Coder

• Let q represent the quantization error for e, then

\[ \hat{f} = f_p + \hat{e}_p = f_p + e_p - q = f - q \]

• Therefore, the error between the original and the reconstructed value \( e_f \) is exactly the same as the quantization error.

• Because the error usually has a non-uniform distribution, a non-uniform quantizer optimized for the distribution of the error signal is usually used.

• A vector quantizer can be used for the error.
Design of the Predictor

• For lossy predictive coders, the optimal predictor should be designed to minimize the MSE between the original $f_0$ and the predicted values $\hat{f}_0$.

$$\sigma_p^2 = E \left\{ \left| f_0 - \sum_k a_k \hat{f}_k \right|^2 \right\}$$

• Since $\hat{f}_k$ is related to $a_k$ and the quantizer in a complicated relation, the precise minimization of the error is difficult.

• In practice, one simply assumes that the quantization error is negligible.
Estimation of the Correlation Coefficients

• One can use spatial sample averaging to approximate statistical ensemble mean.
• To estimate the correlation between a pixel $f(m, n)$ and its neighbor at $f(m-k, n-l)$, denoted by $R_f(k, l)$, one can use

$$R_f(k, l) = \frac{1}{N_s} \sum_{m,n} f(m, n) f(m-k, n-l)$$

where $N_s$ is the number of pairs of $f(m, n)$ and $f(m-k, n-l)$ that are included in the summation.
Non-Adaptive v.s. Adaptive Predictive Coding

• In non-adaptive coders, a fixed set of prediction coefficients are used across the entire image.
• In adaptive coders, one updates the correlation coefficients $R(k, l)$ and hence the prediction coefficient $a_k$ based on local samples.
  – In *forward adaptive* predictive coder, for each block of pixels, the correlation coefficients are calculated for this block and the optimal coefficients are used.
  – In *backward adaptive* predictive coder, the correlation coefficients and consequently the prediction coefficients are updated based on the past reconstructed samples, in both the encoder and decoder.
Diagrams for Transform Coding System

**Encoder**

- \( \tilde{f} \) -> Block transform -> \( \tilde{t} \) -> Quantization of transform coefficients -> \( \hat{t} \) -> Coding of quantized coefficients -> \( B_t \)

**Decoder**

- \( B_t \) -> \( \hat{t} \) -> Dequantization of transform coefficients -> \( \hat{t} \) -> Inverse block transform -> \( \tilde{f} \)
Transform Coding

- Represent an image as the linear combination of some basis images and specify the linear coefficients.
Transform Basis Design

• Optimality Criteria:
  – *Energy compaction*: a few basis images are sufficient to represent a typical image.
  – *Decorrelation*: coefficients for separated basis images are uncorrelated.

• Karhunen Loeve Transform (KLT) is the Optimal transform for a given covariance matrix of the underlying signal.

• Discrete Cosine Transform (DCT) is close to KLT for images that can be modeled by a first order Markov process (*i.e.*, a pixel only depends on its previous pixel).
Basis Images of DCT

\[ h(m, n, u, v) = \alpha(u)\alpha(v) \cos \left( \frac{(2m+1)u\pi}{2N} \right) \cos \left( \frac{(2n+1)v\pi}{2N} \right) \]

where \( \alpha(u) = \begin{cases} 1 & u = 0 \\ \sqrt{\frac{1}{N}} & u = 1, \ldots, N-1 \end{cases} \)

\[
T(u, v) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n)h(m, n, u, v)
\]

\[
f(m, n) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v)h(m, n, u, v)
\]
DCT on a Real Image Block

\[
\begin{align*}
\text{imblock} &= \text{lena256}(128:135,128:135)-128 \\
\text{imblock} &= \\n54 & 68 & 71 & 73 & 75 & 73 & 71 & 45 \\
47 & 52 & 48 & 14 & 20 & 24 & 20 & -8 \\
20 & -10 & -5 & -13 & -14 & -21 & -20 & -21 \\
-24 & -22 & -26 & -24 & -33 & -30 & -23 \\
-29 & -13 & 3 & -24 & -10 & -42 & -41 & 5 \\
-16 & 26 & 26 & -21 & 12 & -31 & -40 & 23 \\
17 & 30 & 50 & -5 & 4 & 12 & 10 & 5 \\
\end{align*}
\]

\[
\begin{align*}
\text{dctblock} &= \text{dct2(imblock)} \\
\text{dctblock} &= \\n31.0000 & 51.7034 & 1.1673 & -24.5837 & -12.0000 & -25.7508 & 11.9640 & 23.2873 \\
\end{align*}
\]

In JPEG, “imblock-128” is done before DCT to shift the mean to zero
Quantization of DCT Coefficients

• Use uniform quantizer on each coefficient
• Different coefficient is quantized with different step-size (Q):
  – Human eye is more sensitive to low frequency components
  – Low frequency coefficients with a smaller Q
  – High frequency coefficients with a larger Q
  – Specified in a normalization matrix
  – Normalization matrix can then be scaled by a scale factor
Default Normalization Matrix in JPEG

Actual step size for $C(i,j)$: $Q(i,j) = QP*M(i,j)$

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

Fall 2003
EL512 Image Processing
Lecture 11, Page 16
Example: Quantized Indices

```matlab
>> dctblock = dct2(imblock)
dctblock =
     31.0000   51.7034   1.1673  -24.5837  -12.0000  -25.7508   11.9640    23.2873
    -1.4562  -13.3225  -0.8750   1.3248  10.3817   16.0762    4.4157   1.1041

>> QP=1;
>> QM=Qmatrix*QP;
>> qdct=floor((dctblock+QM/2)./(QM))
qdct =
     2     5     0    -2     0    -1     0     0
     9     1    -1     2     0     1     0     0
    14     1    -1     0    -1     0     0     0
     3    -1    -1    -1     0     0     0     0
     2    -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

Only 19 coefficients are retained out of 64
```
Example: Quantized Coefficients

>> iq dct = q dct .* Q M

% dequantized DCT block

\[
\begin{array}{cccccccc}
32 & 55 & 0 & -32 & 0 & -40 & 0 & 0 \\
108 & 12 & -14 & 38 & 0 & 58 & 0 & 0 \\
196 & 13 & -16 & 0 & -40 & 0 & 0 & 0 \\
42 & -17 & -22 & -29 & 0 & 0 & 0 & 0 \\
36 & -22 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Original DCT block

\[
\begin{array}{cccccccc}
\end{array}
\]
%reconstructed image block

```matlab
>> qimblock=round(idct2(iqdct))
qimblock=
    58   68   85   79   61   68   67   38
    45   38   39   33   22   24   19   -2
    21     2  -11  -12  -13  -19  -24  -27
    -8  -19  -31  -26  -20  -35  -37  -15
   -31  -17  -21  -20  -16  -39  -41     0
   -33     3  -14  -11  -37  -44     1
   -16    32   18   -10   1   -16  -30     8
     3    54   30   -6   16   11   -7    23
```

Original image block

```matlab
imblock=
    54   68   71   73   75   73   71   45
    47   52   48   14   20   24   20   -8
   20  -10  -13  -14  -21  -20  -21
  -24  -22  -22  -26  -24  -33  -30  -23
  -29  -13     3  -24  -10  -42  -41     5
  -16   26   26  -21   12  -31  -40   23
   17   30   50   -5     4   12   10     5
```
Zig-Zag ordering: converting a 2D matrix into a 1D array, so that the frequency (horizontal+vertical) increases in this order, and the coefficient variance decreases in this order.
Example

$$qdct =
\begin{array}{cccccccc}
2 & 5 & 0 & -2 & 0 & -1 & 0 & 0 \\
9 & 1 & -1 & 2 & 0 & 1 & 0 & 0 \\
14 & 1 & -1 & 0 & -1 & 0 & 0 & 0 \\
3 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

Run-length symbol representation:
\{2,(0,5),(0,9),(0,14),(0,1),(1,-2),(0,-1),(0,1),(0,3),(0,2),(0,-1),(0,2),(1,-1),(2,-1),(0,-1), (4,-1),(0,-1),(0,1),EOB\}

EOB: End of block, one of the symbol that is assigned a short Huffman codeword
Approximation by DCT Basis

Original

With 16/64 Coefficients

With 8/64 Coefficients

With 4/64 Coefficients

Fall 2003

EL512 Image Processing

Lecture 11, Page 23
Coding of DCT Coefficients

- Always code the first $L$ coefficients or coefficients in certain locations (**zonal-coding**)
- Code all non-zero coefficients after quantization (**run-length coding**) -- Used in JPEG standard
  - Runlength coding:
    - Ordering coefficients in the zig-zag order
    - Each symbol = (length-of-zero, non-zero-value)
    - Code each symbol using Huffman coding
What is JPEG

- The Joint Photographic Expert Group (JPEG), under both the International Standards Organization (ISO) and the International Telecommunications Union-Telecommunication Sector (ITU-T)
  - www.jpeg.org

- Has published several standards
  - JPEG: lossy coding of continuous tone still images
    - Based on DCT
  - JPEG-LS: lossless and near lossless coding of continuous tone still images
    - Based on predictive coding and entropy coding
  - JPEG2000: scalable coding of continuous tone still images (from lossy to lossless)
    - Based on wavelet transform
The 1992 JPEG Standard

Contains several modes:

- Baseline system (what is commonly known as JPEG!): lossy
  - Can handle gray scale or color images (8bit)
- Extended system: lossy
  - Can handle higher precision (12 bit) images, providing progressive streams, etc.
- Lossless version

Baseline version

- Each color component is divided into 8x8 blocks
- For each 8x8 block, three steps are involved:
  - Block DCT
  - Perceptual-based quantization
  - Variable length coding: Runlength and Huffman coding
Coding of Quantized DCT Coefficients

• DC coefficient: Predictive coding
  – The DC value of the current block is predicted from that of the previous block, and the error is coded using Huffman coding

• AC Coefficients: Runlength coding
  – Many high frequency AC coefficients are zero after first few low-frequency coefficients
  – Runlength Representation:
    • Ordering coefficients in the zig-zag order
    • Specify how many zeros before a non-zero value
    • Each symbol=(length-of-zero, non-zero-value)
  – Code all possible symbols using Huffman coding
    • More frequently appearing symbols are given shorter codewords
    • For more details on the actual coding table, see Handout (Sec.8.5.3 in [Gonzalez02])

• One can use default Huffman tables or specify its own tables.
• Instead of Huffman coding, arithmetic coding can be used to achieve higher coding efficiency at an added complexity.
Coding of DC Symbols

• Example:
  – Current quantized DC index: 2
  – Previous block DC index: 4
  – Prediction error: -2
  – The prediction error is coded in two parts:
    • Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code using a Huffman code (JPEG Default DC Code)
      – DC= -2 is in category “2”, with a codeword “100”
    • Which position it is in that category, using a fixed length code, length=category number
      – “-2” is the number 1 (starting from 0) in category 2, with a fixed length code of “10”.
      – The overall codeword is “10010”
### TABLE 8.17
JPEG coefficient coding categories.

<table>
<thead>
<tr>
<th>Range</th>
<th>DC Difference Category</th>
<th>AC Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>−1, 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>−3, −2, 2, 3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>−7, ..., −4, 4, ..., 7</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>−15, ..., −8, 8, ..., 15</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>−31, ..., −16, 16, ..., 31</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>−63, ..., −32, 32, ..., 63</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>−127, ..., −64, 64, ..., 127</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>−255, ..., −128, 128, ..., 255</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>−511, ..., −256, 256, ..., 511</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>−1023, ..., −512, 512, ..., 1023</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>−2047, ..., −1024, 1024, ..., 2047</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>−4095, ..., −2048, 2048, ..., 4095</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>−8191, ..., −4096, 4096, ..., 8191</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>−16383, ..., −8192, 8192, ..., 16383</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>−32767, ..., −16384, 16384, ..., 32767</td>
<td>F</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### TABLE 8.18
JPEG default DC code (luminance).

<table>
<thead>
<tr>
<th>Category</th>
<th>Base Code</th>
<th>Length</th>
<th>Category</th>
<th>Base Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>010</td>
<td>3</td>
<td>6</td>
<td>1110</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
<td>4</td>
<td>7</td>
<td>11110</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>5</td>
<td>8</td>
<td>111110</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>00</td>
<td>5</td>
<td>9</td>
<td>1111110</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>7</td>
<td>A</td>
<td>11111110</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>8</td>
<td>B</td>
<td>111111110</td>
<td>20</td>
</tr>
</tbody>
</table>
Coding of AC Coefficients

• Example:
  – First symbol (0,5)
    • The value ‘5’ is represented in two parts:
      • Which category it belongs to (Table of JPEG Coefficient Coding Categories), and code the “(runlength, category)” using a Huffman code (JPEG Default AC Code)
        – AC=5 is in category “3”,
        – Symbol (0,3) has codeword “100”
      • Which position it is in that category, using a fixed length code, length=category number
        – “5” is the number 4 (starting from 0) in category 3, with a fixed length code of “011”.
        – The overall codeword for (0,5) is “100011”
  – Second symbol (0,9)
    • ‘9’ in category ‘4’, (0,4) has codeword ‘1011’,’9’ is number 9 in category 4 with codeword ‘1001’ -> overall codeword for (0,9) is ‘10111001’
  – ETC
### JPEG Tables for Coding AC

**Run,Category Symbols**

<table>
<thead>
<tr>
<th>Run/Category</th>
<th>Base Code</th>
<th>Length</th>
<th>Run/Category</th>
<th>Base Code</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>1010 (= EOB)</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0/1</td>
<td>00</td>
<td>3</td>
<td>8/1</td>
<td>1111010</td>
<td>9</td>
</tr>
<tr>
<td>0/2</td>
<td>01</td>
<td>4</td>
<td>8/2</td>
<td>111111000000</td>
<td>17</td>
</tr>
<tr>
<td>0/3</td>
<td>100</td>
<td>6</td>
<td>8/3</td>
<td>1111111010111</td>
<td>19</td>
</tr>
<tr>
<td>0/4</td>
<td>1011</td>
<td>8</td>
<td>8/4</td>
<td>11111111011000</td>
<td>20</td>
</tr>
<tr>
<td>0/5</td>
<td>11010</td>
<td>10</td>
<td>8/5</td>
<td>1111111101101001</td>
<td>21</td>
</tr>
<tr>
<td>0/6</td>
<td>111000</td>
<td>12</td>
<td>8/6</td>
<td>111111111011010</td>
<td>22</td>
</tr>
<tr>
<td>0/7</td>
<td>1111000</td>
<td>14</td>
<td>8/7</td>
<td>1111111111011011</td>
<td>23</td>
</tr>
<tr>
<td>0/8</td>
<td>111110110</td>
<td>18</td>
<td>8/8</td>
<td>11111111011100</td>
<td>24</td>
</tr>
<tr>
<td>0/9</td>
<td>1111111110000010</td>
<td>25</td>
<td>8/9</td>
<td>111111110111101</td>
<td>25</td>
</tr>
<tr>
<td>0/A</td>
<td>1111111110000011</td>
<td>26</td>
<td>8/A</td>
<td>111111110111100</td>
<td>26</td>
</tr>
<tr>
<td>1/1</td>
<td>1100</td>
<td>5</td>
<td>9/1</td>
<td>111110000</td>
<td>10</td>
</tr>
<tr>
<td>1/2</td>
<td>11001</td>
<td>8</td>
<td>9/2</td>
<td>1111111011111</td>
<td>18</td>
</tr>
<tr>
<td>1/3</td>
<td>111001</td>
<td>10</td>
<td>9/3</td>
<td>11111111000000</td>
<td>19</td>
</tr>
<tr>
<td>1/4</td>
<td>11110110</td>
<td>13</td>
<td>9/4</td>
<td>11111111100001</td>
<td>20</td>
</tr>
<tr>
<td>1/5</td>
<td>111110110</td>
<td>16</td>
<td>9/5</td>
<td>11111111110001</td>
<td>21</td>
</tr>
<tr>
<td>1/6</td>
<td>11111111000010</td>
<td>22</td>
<td>9/6</td>
<td>11111111000011</td>
<td>22</td>
</tr>
<tr>
<td>1/7</td>
<td>111111110000101</td>
<td>23</td>
<td>9/7</td>
<td>111111111000100</td>
<td>23</td>
</tr>
<tr>
<td>1/8</td>
<td>111111110000110</td>
<td>24</td>
<td>9/8</td>
<td>111111111000101</td>
<td>24</td>
</tr>
<tr>
<td>1/9</td>
<td>111111110000111</td>
<td>25</td>
<td>9/9</td>
<td>111111111000110</td>
<td>25</td>
</tr>
<tr>
<td>1/A</td>
<td>111111110001000</td>
<td>26</td>
<td>9/A</td>
<td>111111111001111</td>
<td>26</td>
</tr>
<tr>
<td>2/1</td>
<td>11011</td>
<td>6</td>
<td>A/1</td>
<td>111110000</td>
<td>10</td>
</tr>
<tr>
<td>2/2</td>
<td>1111000</td>
<td>10</td>
<td>A/2</td>
<td>11111111001000</td>
<td>18</td>
</tr>
<tr>
<td>2/3</td>
<td>111110111</td>
<td>13</td>
<td>A/3</td>
<td>11111111010001</td>
<td>19</td>
</tr>
<tr>
<td>2/4</td>
<td>111111110001001</td>
<td>20</td>
<td>A/4</td>
<td>11111111010010</td>
<td>20</td>
</tr>
<tr>
<td>2/5</td>
<td>111111110001010</td>
<td>21</td>
<td>A/5</td>
<td>11111111010101</td>
<td>21</td>
</tr>
<tr>
<td>2/6</td>
<td>111111110001011</td>
<td>22</td>
<td>A/6</td>
<td>111111110101100</td>
<td>22</td>
</tr>
<tr>
<td>2/7</td>
<td>111111111000100</td>
<td>23</td>
<td>A/7</td>
<td>111111110110101</td>
<td>23</td>
</tr>
</tbody>
</table>

**Table 8.19**

JPEG default AC code (luminance) (continues on next page).
JPEG Performance for B/W images

65536 Bytes
8 bpp

4839 Bytes
0.59 bpp
CR=13.6

3037 Bytes
0.37 bpp
CR=21.6

1818 Bytes
0.22 bpp
CR=36.4

Fall 2003
JPEG for Color Images

- Color images are typically stored in (R,G,B) format
- JPEG standard can be applied to each component separately
  - Does not make use of the correlation between color components
  - Does not make use of the lower sensitivity of the human eye to chrominance samples
- Alternate approach
  - Convert (R,G,B) representation to a YCbCr representation
    - Y: luminance, Cb, Cr: chrominance
  - Down-sample the two chrominance components
    - Because the peak response of the eye to the luminance component occurs at a higher frequency (3-10 cpd) than to the chrominance components (0.1-0.5 cpd). (Note: cpd is cycles/degree)
- JPEG standard is designed to handle an image consists of many (up to 100) components
RGB <-> YCbCr Conversion

\[
\begin{bmatrix}
Y \\
C_b \\
C_r
\end{bmatrix} = \begin{bmatrix}
0.299 & 0.587 & 0.114 \\
-0.169 & -0.331 & 0.500 \\
0.500 & -0.419 & -0.081
\end{bmatrix} \begin{bmatrix}
R \\
G \\
B
\end{bmatrix} + \begin{bmatrix}
0 \\
128 \\
128
\end{bmatrix}
\]

\[
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix} = \begin{bmatrix}
1.000 & -0.001 & 1.402 \\
1.000 & -0.344 & -0.714 \\
1.000 & 1.772 & 0.001
\end{bmatrix} \begin{bmatrix}
Y \\
C_b - 128 \\
C_r - 128
\end{bmatrix}
\]

Note: Cb ~ Y-B, Cr ~ Y-R, are known as color difference signals.
**Chrominance Subsampling**

4:4:4
For every 2x2 Y Pixels
4 Cb & 4 Cr Pixel
(No subsampling)

4:2:2
For every 2x2 Y Pixels
2 Cb & 2 Cr Pixel
(Subsampling by 2:1 horizontally only)

4:1:1
For every 4x1 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 4:1 horizontally only)

4:2:0
For every 2x2 Y Pixels
1 Cb & 1 Cr Pixel
(Subsampling by 2:1 both horizontally and vertically)

- **Y Pixel**
- **Cb and Cr Pixel**

4:2:0 is the most common format
Each basic coding unit (called a data unit) is a 8x8 block in any color component. In the interleaved mode, 4 Y blocks and 1 Cb and 1 Cr blocks are processed as a group (called a minimum coding unit or MCU) for a 4:2:0 image.
# Default Quantization Table

<table>
<thead>
<tr>
<th>For luminance</th>
<th>For chrominance</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 11 10 16 24 40 51 61</td>
<td>17 18 24 47 99 99 99 99</td>
</tr>
<tr>
<td>12 12 14 19 26 58 60 55</td>
<td>18 21 26 66 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 22 29 51 87 80 62</td>
<td>47 66 99 99 99 99 99 99</td>
</tr>
<tr>
<td>18 22 37 56 68 109 103 77</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
<tr>
<td>24 35 55 64 81 104 113 92</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
<tr>
<td>49 64 78 87 103 121 120 101</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
<tr>
<td>72 92 95 98 112 100 103 99</td>
<td>99 99 99 99 99 99 99 99</td>
</tr>
</tbody>
</table>

The encoder can specify the quantization tables different from the default ones as part of the header information.
Performance of JPEG

- For color images at 24 bits/pixel (bpp)
  - 0.25-0.5 bpp: moderate to good
  - 0.5-0.75 bpp: good to very good
  - 0.75-1.5 bpp: excellent, sufficient for most applications
  - 1.5-2 bpp: indistinguishable from original

- For grayscale images at 8 bpp
  - 0.5 bpp: excellent quality
JPEG Performance

487x414 pixels,
Uncompressed, 600471 Bytes, 24 bpp
85502 Bytes, 3.39 bpp, CR=7

487x414 pixels
41174 Bytes, 1.63 bpp, CR=14.7
JPEG Pros and Cons

• Pros
  – Low complexity
  – Memory efficient
  – Reasonable coding efficiency

• Cons
  – Single resolution
  – Single quality
  – No target bit rate
  – Blocking artifacts at low bit rate
  – No lossless capability
  – Poor error resilience
  – No tiling
  – No regions of interest
JPEG2000 Features

• Improved coding efficiency
• Full quality scalability
  – From lossless to lossy at different bit rate
• Spatial scalability
• Improved error resilience
• Tiling
• Region of interests
• More demanding in memory and computation time
What is Scalability?

Quality Scalability of JPEG2000

Spatial Scalability of JPEG2000

18. Example of the progressive-by-resolution decoding for the color image “bike.” From [skodras01]
JPEG2000 vs. JPEG: Coding Efficiency

From [skodras01]


Fall 2003 EL512 Image Processing Lecture 11, Page 45
Example Image

20. Image “watch” of size $512 \times 512$ (courtesy of Kevin Odhner): (a) original, and reconstructed after compression at 0.2 $b/p$ by means of (b) JPEG and (c) JPEG 2000.

From [skodras01]
Another Example

21. Reconstructed image “ski” after compression at 0.25 b/p by means of (a) JPEG and (b) JPEG 2000.

From [skodras01]
How J2K Achieves Scalability?

• Core: Wavelet transform
  – Yields a multi-resolution representation of an original image

• Still a transform coder
  – Block DCT is replaced by a full frame wavelet transform
    • Also known as subband or wavelet coder
  – Wavelet coefficients are coded bit plane by bit plane
  – Spatial scalability can be achieved by reconstructing from only low resolution wavelet coefficients
  – Quality scalability can be achieved by decoding only partial bit planes
Wavelet Decomposition

6. Three-level dyadic wavelet decomposition of the image “Lena.”

From [skodras01]
Two Band Subband Decomposition

From [Vetterli01]

H0: Lowpass filter, \( y_0 \): a blurred version of \( x \)
H1: Highpass filter, \( y_1 \): edges in \( x \)
The above structure can be applied to rows of an image first, and then columns, forming 2D wavelet decomposition.

From [Vetterli01]
Wavelet Transform for Images

2D wavelet transform is accomplished by applying the 1D decomposition along rows of an image first, and then columns.

From [Usevitch01]
Wavelet Decomposition

6. Three-level dyadic wavelet decomposition of the image “Lena.”

From [skodras01]
- **Quantization:** Each subband may use a different step-size. Quantization can be skipped to achieve lossless coding.
- **Entropy coding:** Bit plane coding is used, the most significant bit plane is coded first.
- **Quality scalability is achieved by decoding only partial bit planes, starting from the MSB. Skipping one bit plane while decoding = Increasing quantization stepsize by a factor of 2.
1. Consider an image predictor: \( f(m, n) = \alpha f(m, n-1) + \beta f(m-1, n) + \gamma f(m-1, n-1) \), where \( m \) and \( n \) correspond to row and column indices, respectively. Assume that the image values have been scaled and shifted so that they have zero mean and unit variance. Suppose the correlation coefficients are \( \rho \) for two horizontally or vertically adjacent pixels, and \( \rho^2 \) for two diagonally or anti-diagonally adjacent pixels. Find the optimal values for \( \alpha, \beta, \) and \( \gamma \) that will minimize the mean square error of prediction, and determine the corresponding minimal error.

2. Consider the following image compression scheme. The pixels in an image is scanned from left to the right and from the top to the bottom. Each new pixel is predicted by the average of the pixel above and the one to the left. Let \( f \) and \( \hat{f} \) represent the original and predicted values, and \( e = f - \hat{f} \) the prediction error. The prediction error is quantized to “0”, “B”, or “-B” according to:

\[
\hat{e} = \begin{cases} 
- B & e < -T \\
0 & -T \leq e \leq T \\
B & e > T 
\end{cases}
\]

All the possible quantized symbols (“0”, “B” and “-B”) are assigned binary codewords using Huffman coding. For the image given in Fig. 2:

(a) Determine the predicted image. For simplicity, ignore the pixels in the first row and column. (i.e. Assuming those pixels are coded directly and losslessly). Also, for simplicity, always use the original values in the previous positions for prediction (i.e. using open loop predictive coding)

(b) Determine the prediction error image.

(c) Determine the quantized error image, assuming \( B = 3 \), \( T = 2 \).

(d) Determine the reconstructed image.

(f) Design a Huffman codebook for all the symbols based on their occurrence frequencies in this example. Calculate the average bit rate (bits/pixel) using this method.

(g) Note that in practice, one cannot assume that the original values for the previous pixels are available for prediction. Determine the reconstructed image without assuming the original values are available for the reference pixels used for prediction. That is, put the reconstructed value (the predicted value plus the quantized error values) for each pixel back to its position immediately, and when predicting a new pixel, use the reconstructed values for its past neighbors (This is known as closed-loop predictive coding).
Reading

• Prof. Yao Wang’s Lecture Notes, Section 9.4.2 – 9.5
• R. Gonzalez, “Digital Image Processing,” Section 8.5 - 8.6