CSE 490 G
Introduction to Data Compression
Winter 2006

Lossy Image Compression
Transform Coding
JPEG
Lossy Image Compression Methods

- DCT Compression
  - JPEG
- Wavelet Compression
  - SPIHT
  - UWIC (University of Washington Image Coder)
  - EBCOT (JPEG 2000)
- Scalar quantization (SQ).
- Vector quantization (VQ).
JPEG Standard

• JPEG - Joint Photographic Experts Group

• JPEG 2000 uses to wavelet compression.
Barbara

original
JPEG 32:1 compression ratio
.25 bits/pixel (8 bits)
VQ
Wavelet-SPIHT
SPIHT
Original
Images and the Eye

- Images are meant to be viewed by the human eye (usually).
- The eye is very good at “interpolation”, that is, the eye can tolerate some distortion. So lossy compression is not necessarily bad. The eye has more acuity for luminance (gray scale) than chrominance (color).
  - Gray scale is more important than color.
  - Compression is usually done in the YUV color coordinates, Y for luminance and U,V for color.
  - U and V should be compressed more than Y
  - This is why we will concentrate on compressing gray scale (8 bits per pixel) images.
Distortion

- Lossy compression: $x \neq \hat{x}$
- Measure of distortion is commonly mean squared error (MSE). Assume $x$ has $n$ real components (pixels).

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x_i})^2$$
PSNR

• Peak Signal to Noise Ratio (PSNR) is the standard way to measure fidelity.

\[
PSNR = 10 \log_{10}(\frac{m^2}{MSE})
\]

where \( m \) is the maximum value of a pixel possible. For gray scale images (8 bits per pixel) \( m = 255 \).

• PSNR is measured in decibels (dB).
  – .5 to 1 dB is said to be a perceptible difference.
  – Decent images start at about 30 dB
Rate-Fidelity Curve

Properties:
- Increasing
- Slope decreasing

PSNR

bits per pixel
PSNR is not Everything

\[ \text{PSNR} = 25.8 \text{ dB} \]

\[ \text{PSNR} = 25.8 \text{ dB} \]
PSNR Reflects Fidelity (1)

VQ

PSNR 25.8
.63 bpp
12.8 : 1
PSNR Reflects Fidelity (2)

VQ

PSNR 24.2
.31 bpp
25.6 : 1
PSNR Reflects Fidelity (3)

VQ

PSNR 23.2
.16 bpp
51.2 : 1
Idea of Transform Coding

- Transform the input pixels $x_0, x_2, \ldots, x_{N-1}$ into coefficients $c_0, c_1, \ldots, c_{N-1}$ (real values)
  - The coefficients are have the property that most of them are near zero
  - Most of the “energy” is compacted into a few coefficients

- Quantize the coefficients
  - This is where there is loss, since coefficients are only approximated
  - Important coefficients are kept at higher precision

- Entropy encode the quantization symbols
Decoding

- Entropy decode the quantized symbols
- Compute approximate coefficients $c'_0, c'_1, \ldots, c'_{N-1}$ from the quantized symbols.
- Inverse transform $c'_0, c'_1, \ldots, c'_{N-1}$ to $x'_0, x'_1, \ldots, x'_{N-1}$ which is a good approximation of the original $x_0, x_2, \ldots, x_{N-1}$. 
Block Diagram of Transform Coding

Encoder

input $x$ → transform $c$ → quantization bit allocation $s$ → entropy coding $b$

Decoder

entropy decoding $s$ → symbols $c'$ → inverse transform $x'$
Mathematical Properties of Transforms

• Linear Transformation - Defined by a real nxn matrix $A = (a_{ij})$

\[
\begin{bmatrix}
a_{00} & \cdots & a_{0,N-1} \\
\vdots & \ddots & \vdots \\
a_{N-1,0} & \cdots & a_{N-1,N-1}
\end{bmatrix}
\begin{bmatrix}
x_0 \\
\vdots \\
x_{N-1}
\end{bmatrix}
= 
\begin{bmatrix}
c_0 \\
\vdots \\
c_{N-1}
\end{bmatrix}
\]

• Orthonormality $A^{-1} = A^T$ (transpose)
Why Coefficients

\[ A^T c = x \]

\[
\begin{bmatrix}
    a_{00} & \cdots & a_{N-1,0} \\
    \vdots & \ddots & \vdots \\
    a_{0,N-1} & \cdots & a_{N-1,N-1}
\end{bmatrix}
\begin{bmatrix}
    c_0 \\
    \vdots \\
    c_{N-1}
\end{bmatrix}
= 
\begin{bmatrix}
    x_0 \\
    \vdots \\
    x_{N-1}
\end{bmatrix}
\]

basis vectors

coefficients
Why Orthonomality

• The energy of the data equals the energy of the coefficients

\[
\sum_{i=0}^{N-1} c_i^2 = c^T c = (Ax)^T (Ax)
\]

\[
= (x^T A^T)(Ax) = x^T (A^T A)x = x^T x = \sum_{i=0}^{N-1} x_i^2
\]
Squared Error is Preserved with Orthonormal Transformations

• In lossy coding we only send an approximation $c'_i$ of $c_i$ because it takes fewer bits to transmit the approximation.

Let $c_i = c'_i + \varepsilon_i$

$$\sum_{i=0}^{N-1} \varepsilon_i^2 = \sum_{i=0}^{N-1} (c_i - c'_i)^2 = (c - c')^T (c - c') = (Ax - Ax')^T (Ax - Ax')$$

$$= (A(x - x'))^T (A(x - x')) = ((x - x')^T A^T) (A(x - x'))$$

$$= (x - x')^T (A^T A)(x - x') = (x - x')^T (x - x')$$

$$= \sum_{i=0}^{N-1} (x_i - x'_i)^2$$

Squared error in original.
Compaction Example

\[ A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \]

\[ A^2 = A \Rightarrow A^{-1} = A \]

\[ A^\top = A = A^{-1} \quad \text{orthonormal} \]

\[
\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} b \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{2}b \\ 0 \end{bmatrix} \quad \text{compaction}
\]
Discrete Cosine Transform

\[
d_{ij} = \begin{cases} 
\sqrt{\frac{1}{N}} & i = 0 \\
\sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N} & i > 0 
\end{cases}
\]

\[N = 4\]

\[
D = \begin{bmatrix}
.5 & .5 & .5 & .5 \\
.65328 & .270598 & -.270598 & -.65328 \\
.5 & -.5 & -.5 & .5 \\
.270598 & -.65328 & .65328 & -.270598 
\end{bmatrix}
\]
Basis Vectors
Decomposition in Terms of Basis Vectors

\[
\begin{bmatrix}
.5 \\
.5 \\
.5 \\
.5 \\
\end{bmatrix}
\begin{bmatrix}
.5 \\
.270598 \\
-.270598 \\
-.653281 \\
\end{bmatrix}
\begin{bmatrix}
.5 \\
-.5 \\
-.5 \\
.5 \\
\end{bmatrix}
\begin{bmatrix}
.270598 \\
-.653281 \\
.653281 \\
-.270598 \\
\end{bmatrix}
\begin{bmatrix}
\cdots \\
x_0 \\
x_1 \\
x_2 \\
x_3 \\
\cdots \\
\end{bmatrix}
\]

DC coefficient \quad AC coefficients
Block Transform

Each 8x8 block is individually coded
2-Dimensional Block Transform

Block of pixels $X$

<table>
<thead>
<tr>
<th></th>
<th>$x_{00}$</th>
<th>$x_{01}$</th>
<th>$x_{02}$</th>
<th>$x_{03}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{10}$</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
<td></td>
</tr>
<tr>
<td>$x_{20}$</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
<td></td>
</tr>
<tr>
<td>$x_{30}$</td>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
<td></td>
</tr>
</tbody>
</table>

Transform

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Transform rows

$$r_{ij} = \sum_{k=0}^{N-1} a_{kj} x_{ik}$$

Transform columns

$$c_{ij} = \sum_{m=0}^{N-1} a_{im} r_{mj} = \sum_{m=0}^{N-1} a_{im} \sum_{k=0}^{N-1} a_{kj} x_{mk} = \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} a_{im} a_{kj} x_{mk}$$

Summary

$$C = AXA^T$$
8x8 DCT Basis
Importance of Coefficients

- The DC coefficient is the most important.
- The AC coefficients become less important as they are farther from the DC coefficient.
- Example Bit Allocation

```
00000000
00000000
00000001
00000012
00000123
00001235
00012357
00123578
```

Compression
55 bits for 64 pixels = .86 bpp
Quantization

- For a nxn block we construct a nxn matrix Q such that $Q_{ij}$ indicates how many quantization levels to use for coefficient $c_{ij}$.
- Encode $c_{ij}$ with the label

$$s_{ij} = \left\lfloor \frac{c_{ij}}{Q_{ij}} + 0.5 \right\rfloor$$

Larger $Q_{ij}$ indicates fewer levels.
- Decode $s_{ij}$ to

$$c'_{ij} = s_{ij}Q_{ij}$$
Example Quantization

- \( c = 54.2, \ Q = 24 \)
  \[
  s = \left\lfloor \frac{54.2}{24} + 0.5 \right\rfloor = 2
  \]
  \[
  c' = 2 \cdot 24 = 48
  \]

- \( c = 54.2, \ Q = 12 \)
  \[
  s = \left\lfloor \frac{54.2}{12} + 0.5 \right\rfloor = 5
  \]
  \[
  c' = 5 \cdot 12 = 60
  \]

- \( c = 54.2, \ Q = 6 \)
  \[
  s = \left\lfloor \frac{54.2}{6} + 0.5 \right\rfloor = 9
  \]
  \[
  c' = 9 \cdot 6 = 54
  \]
Example Quantization Table

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<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>
<td>61</td>
<td></td>
<td></td>
</tr>
<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>33</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>

Increase the bit rate = halve the table
Decrease the bit rate = double the table
Zig-Zag Coding

- DC label is coded separately.
- AC labels are usually coded in zig-zag order using a special entropy coding to take advantage the ordering of the bit allocation (quantization).
JPEG (1987)

- Let $P = [p_{ij}]$, $0 < i,j < N$ be an image with $0 < p_{ij} < 256$.
- Center the pixels around zero
  - $x_{ij} = p_{ij} - 128$
- Code 8x8 blocks of $P$ using DCT
- Choose a quantization table.
  - The table depends on the desired quality and is built into JPEG
- Quantize the coefficients according to the quantization table.
  - The quantization symbols can be positive or negative.
- Transmit the labels (in a coded way) for each block.
Block Transmission

• DC coefficient
  – DC coefficients don’t change much from block to neighboring block. Hence, their labels change even less.
  – Predictive coding using differences is used to code the DC label.

• AC coefficients
  – Do a zig-zag coding.
## Example Block of Labels

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |   |   |   |   |   |   |
| -8| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |   |   |   |   |   |   |
| 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |   |   |   |   |   |   |
| 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 | 0 |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |   |   |   |   |   |   |   |

### Coding order of AC labels

2 \(-8\) 3 0 0 0 0 0 1 1 0 0 1 0 0 .....
Coding Labels

• Categories of labels
  – 1    {0}
  – 2    {-1, 1}
  – 3    {-3,-2,2,3}
  – 4    {-7,-6,-5,-4,4,5 6 7}

• Label is indicated by two numbers C,B

• Examples

<table>
<thead>
<tr>
<th>label</th>
<th>C,B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3, 2</td>
</tr>
<tr>
<td>-4</td>
<td>4, 3</td>
</tr>
</tbody>
</table>
Coding AC Label Sequence

• A symbol has three parts (Z,C,B)
  – Z for number of zeros preceding a label $0 \leq Z \leq 15$
  – C for the category of the label
  – B for a C-1 bit number for the actual label

• End of Block symbol (EOB) means the rest of the block is zeros. EOB = (0,0,-)

• Example: $2 \ 8 \ 3 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ \ldots$
  $(0,3,2)(0,5,7)(0,3,3)(4,2,1)(0,2,1)(2,2,1)(0,0,-)$
Coding AC Label Sequence

• Z, C have a prefix code
• B is a C-1 bit number

Partial prefix code table

<table>
<thead>
<tr>
<th>Z</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>010</td>
<td>00</td>
<td>01</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>110</td>
<td>1101</td>
<td>111110001</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1110</td>
<td>1111101</td>
<td>1111110111</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>111010</td>
<td>111110111</td>
<td>11111110101</td>
<td></td>
</tr>
</tbody>
</table>

(0,3,2) (0,5,7) (0,3,3) (4,2,1) (0,2,1) (2,2,1) (0,0,-)

100 10 11010 0111 100 11 1111111000 1 01 1 11111001 1 1010

46 bits representing 64 pixels = .72 bpp
Notes on Transform Coding

• Video Coding
  – MPEG – uses DCT
  – H.263, H.264 – uses DCT

• Audio Coding
  – MP3 = MPEG 1- Layer 3 uses DCT

• Alternative Transforms
  – Lapped transforms remove some of the blocking artifacts.
  – Wavelet transforms do not need to use blocks at all.