**Jpeg Image Compression**

C306
Martin Jagersand

**Jpeg overview**

- Probably most useful and popular compression method for natural images.
- Based on the Discrete Cosine Transform (DCT) with other smarts added

**Baseline**
1. sequential and

**Extended versions:**
1. progressive (multiple scans at same spatial resolution)
2. lossless
3. hierarchical (multiple spatial resolutions)

**Compression block diagram**

- 8x8 DCT transform (as defined in the previous lecture)
- Q: Quantization of DCT coefficients
- Predictive coding (DC) or Zig-Zag scan (AC) of Quantized DCT coefficients
- Variable length coding of the quantized AC and DC coefficients
- Decoding performs operations in reverse

**DCT of block image**

- Image divided, 8x8 pixel blocks I =
- Subtraction I = I – 128
- Cos Trans IC = DCT(I) (= A*I)
- Now have to code the 64 transform coefficients in IC!
Remember definition, Gonzalez
Discrete Cosine Transform, DCT

- **Forward Transform:**
  \[ C(u) = \hat{e}(u) \times f(x) \cos \frac{(2x+1)u}{2N} \]
- **Inverse:**
  \[ f(x) = \sum_{u=0}^{N-1} \hat{e}(u) \times C(u) \cos \frac{(2x+1)u}{2N} \]
- **Where:**
  \[ \hat{e}(u) = \begin{cases} \frac{1}{\sqrt{2N}} & \text{for } u = 0 \\ \frac{1}{\sqrt{N}} & \text{for } u > 0 \end{cases} \]

Definition, matrix form:

- Let \( C = Af \)
- \( f = A'C \) (‘= transpose), where:

\[
A = \begin{bmatrix}
\frac{1}{\sqrt{N}} & \cos(3u=2N) & \cdots & \cos((2N^2-3N+1)u=2N) \\
1 & \cos(3u=1u=2N) & \cdots & \cos((2N^2-3N+1)u=2N) \\
\vdots & \vdots & \ddots & \vdots \\
1 & \cos(3u=1u=2N) & \cdots & \cos((2N^2-3N+1)u=2N)
\end{bmatrix}
\]

Example
4x4 Cosine transform

- Cosine basis
- Test “image”
- Why

Transformed Im

Why??
Transforms graphically
Quantization

- Remember: DCT compacts the energy into a few coefficients in IC
- Quant: $Q = \text{round}(IC(u,v)/Z(u,v))$
- Each number uniformly quantized
- But different $Z(u,v)$ weights importance

Example Z mask

<table>
<thead>
<tr>
<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>60</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>
</tr>
<tr>
<td>47</td>
<td>66</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
</tbody>
</table>

Encoding of Quantized

- DC (first DCT component) and AC (remaining 63 components) are encoded differently.

DC Coefficients

- DC coefficients represent the average intensity value for the 8x8 block
- Are differentially coded using previous DC value (in raster order)
- DCT values have dynamic range of 11 bits, thus difference values are encoded at 12 bits
- Encoded as range followed by remaining bits
DC range coding

- See Gonzalez table 6.14 and 6.15

<table>
<thead>
<tr>
<th>Hex</th>
<th>Binary</th>
<th>Difference Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>-1,1</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>-2,2...3</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>-3...-4,4...7</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>-5,...-8,8...15</td>
</tr>
<tr>
<td>5</td>
<td>110</td>
<td>-7...-16,16...31</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>-15...-32,32...63</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>11111110</td>
<td>-2047...-1024,1024...2047</td>
</tr>
</tbody>
</table>

Difference Range

- Difference value is encoded as the Huffman code for range, followed by remaining bits of precision to reconstruct actual value.
- If value is positive, precede value with sign bit of 1, followed by value of low bits.
- If value is negative, precede value with sign bit of 0, followed by 1’s compliment (inverse) of low bit value.

Example: DC Coefficient coding

<table>
<thead>
<tr>
<th>DC(x,y)</th>
<th>diff_{lb}</th>
<th>DC(x,y)</th>
<th>diff_{lb}</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1110</td>
<td>-19</td>
<td>101</td>
</tr>
<tr>
<td>9</td>
<td>101</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>-19</td>
<td>110</td>
<td>-19</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>-1</td>
<td>00</td>
</tr>
</tbody>
</table>

Encoded data = 1110 110010 101 1001 00 110 01100 010 0 010 0

35 bits required, would have taken 6x11 = 66 bits w/o VLC

AC coefficients

- Zig-zag scan
AC coefficients: coding

- Quantization creates long runs of zero coefficients
- Non-zero coefficients are labelled AC₀, AC₁,…ACₙ
- Run-length encode AC coefficients using format: (RᵢSᵢ)LB₀, (RᵢSᵢ)LB₁,… (RᵢSᵢ)LBₙ,EOB
  - Rᵢ = 4-bit number of zero coefficients between ACᵢ and ACᵢ₋₁ (AC₀ = DC)
  - Sᵢ = size category for quantized coefficient ACᵢ
  - EOB = ‘1010’
  - RᵢSᵢ = F0 is used to extend runs of longer than 15

Example: AC coefficients

DCT values = (39 -3 2 1 1 0 0 0 0 -1 0 0 0 0 . . . 0)

- 0.2 → '01'  - 3+2 = -1 → '00'
- 0.2 → '01'  - 2-2 = 0 → '10'
- 0.1 → '00'  - just sign '1'
- 0.1 → '00'  - just sign '0'
- 0.1 → '00'  - just sign '1'
- 5.1 → '1111 0010' - just sign '0'
- 0.0 → 1010 (EOB=End of Block/defined 0,0)

Enc. data:
1110 100111 0100 0110 001 000 001 11110100 1010

DC  AC component

JPEG File Format

JPEG Stand-alone Markers

<table>
<thead>
<tr>
<th>Value</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF01</td>
<td>TEM</td>
<td>Temporary (used for Arithmetic coding)</td>
</tr>
<tr>
<td>FFD0-FFD7</td>
<td>RST₀-RST₇</td>
<td>Reset Marker</td>
</tr>
<tr>
<td>FFD8</td>
<td>SOI</td>
<td>Start of Image</td>
</tr>
<tr>
<td>FFD9</td>
<td>EOI</td>
<td>End of Image</td>
</tr>
</tbody>
</table>

JPEG File Format

JPEG Data Markers (continued)

<table>
<thead>
<tr>
<th>Value</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF02-BF</td>
<td>RES</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFC0</td>
<td>SOF₀</td>
<td>Start of Frame, SOF baseline</td>
</tr>
<tr>
<td>FFC1</td>
<td>SOF₁</td>
<td>SOF extended sequential mode</td>
</tr>
<tr>
<td>FFC2</td>
<td>SOF₂</td>
<td>SOF progressive mode</td>
</tr>
<tr>
<td>FFC3</td>
<td>SOF₃</td>
<td>SOF lossless mode</td>
</tr>
<tr>
<td>FFC4</td>
<td>DHT</td>
<td>Define Huffman Table</td>
</tr>
<tr>
<td>FFC5</td>
<td>SOF₅</td>
<td>SOF differential sequential mode</td>
</tr>
<tr>
<td>FFC6</td>
<td>SOF₆</td>
<td>SOF differential progressive mode</td>
</tr>
<tr>
<td>FFC7</td>
<td>SOF₇</td>
<td>SOF differential lossless mode</td>
</tr>
<tr>
<td>FFC8</td>
<td>JPG</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFC9</td>
<td>SOF₉</td>
<td>SOF extended sequential, arithmetic coding mode</td>
</tr>
<tr>
<td>FFCA</td>
<td>SOF₁₀</td>
<td>SOF progressive, arithmetic coding</td>
</tr>
<tr>
<td>FFCB</td>
<td>SOF₁₁</td>
<td>SOF lossless arithmetic mode</td>
</tr>
</tbody>
</table>
JPEG File Format

- Markers are used to break up the JPEG stream into blocks.
- Markers always begin with FF_H.
- Markers can be preceded by as many FF’s as desired, preceding FFs are ignored.
- Two types:
  - stand alone markers
  - data markers (2-byte length field)

JPEG Data Markers

- FFCC  = Define arithmetic conditions
- FFCD  = SOF, SOF differential sequential, arithmetic
- FFCE  = SOF, SOF differential progressive, arithmetic
- FFCF  = SOF, SOF differential lossless, arithmetic
- FFDA  = SOS, Start of Scan
- FFDB  = DQT, Define Quantization Table
- FFDC  = DNL, Define Number of Lines
- FFDD  = DRI, Define Restart Interval
- FFDE  = DHP, Define Hierarchical progression
- FFDF  = EXP, Expand Reference Components
- FFEO-EF = APP, Application Specific Data
- FFFE  = COM, Comment
- FFFF  = DCH, Marker

JPEG Data Sequence

- Start Of Image
- APP, JFIF Mark
  - Optional APP, JFIF Extension header
  - Table(s)
- Start Of Frame
  - Table(s)
- Start of Scan 1
  - Scan 1 Data
  - End Of Image
  - Table(s)
  - Start of Scan 2
  - Scan 1 Data
  - Scan n Data

DCH Marker

- FFC4 = Define Huffman Table
- Mode byte:
  - high 4-bits define type (0=DC, 1=AC)
  - low 4-bits define table ID (0-1 baseline)
- Count array of sixteen 8-bit values:
  - Huffman codes are < 16 bits in length
  - Sorted variable length list of symbol values:
    - DCT Coefficients range from 0-255
    - byte values are included in ascending order
    - number of values equal to sum of count array
JPEG File Format

DQT Marker
- FFDB = Define Quantization Table
- 1-byte mode
  - High 4-bit = SIZE
    - size = 0 → 1-byte values
    - size = 1 → 2-byte values
  - Low 4-bit = ID
- Followed by the quantization values
  - 64 values (one or two bytes each)
  - unsigned values

FFDB

Length

Mode

Q[64]

SOF_n Marker
- FFCX = Start of Frame
- 2-byte length field
- 1-byte sample precision (= 8 or 10)
- 2-byte image height
- 2-byte image width
- 1-byte chan - number of color chan (n = 1 or 3)
- Component Specific Areas (1 or 3)
  - 1-byte identifier Y=1, C_b = 2, and C_r =3
  - 1-byte sampling (4-bit horz then 4-bit vert)
  - 1-byte Qtable ID

FFCX

length

precision

height

width

chan

CSA(1)

CSA(2)

CSA(3)

FFDA

length

chan

CSA(1)

CSA(2)

CSA(3)

SSS

SSE

SA

SOS Marker
- FFDA = Start of Scan
- 2-byte length field
- number color channels in this scan
- Specific Areas
  - 1-byte component ID
  - 1-byte Huffman select (4-bit DC then 4-bit AC)
- Spectral Selection Start (= 0)
- Spectral Selection End (= 63)
- Successive Approximation (= 0)

JFIF
JPEG only specifies a method of image compression, not a file format for transmission, storage, or exchange between computers.

JPEG File Interchange Format (JFIF)

Developed by C-cube Microsystems for the purpose of storing and sharing JPEG encoded images.
JFIF

- 16-bit words stored in Big Endian form
- JPEG is stored as a stream of blocks, each block with a marker value
- First two bytes are Start Of Image (SOI) marker values
- Next is a JFIF application marker segment
- Finally, there may be one or more JFIF extension marker segments
- JPEG bitstream begins after application and extension marker segments

JFIF

- 2-byte SOI = FF D8
- 2-byte APP0 = FF E0
- 2-byte Length = length of segment
- 5-byte ID = “JFIF” (last character is NULL)
- 2-byte Version (major and minor revision #s)
- 1-byte Units
  - 00 = Not defined
  - 01 = PEL/inch
  - 02 = PEL/cm
- 2-byte Xdensity /Ydensity is Resolution in Units
- 1-byte Xthumbnail /Ythumbnail is width and height of thumbnail image

JFIF

- JFIF header may not be complete, JPEG data stream may start immediately after the stream FF D8 FF (if next byte is not E0)
- Length is size remainder (inclusive of length) of application marker segment
  - usually equal to 16
- When Units are "not defined", Xresolution and Yresolution indicate the aspect ratio
- Thumbnail image may follow
  - uncompressed
  - RGB format
  - 8-bit precision
  - less than 64k

JFIF

- Thumbnails (previews/icons) were added as extension marker segments in versions later than 1.01
- Permits multiple thumbnail formats
  - JPEG compressed
  - Color Palette
  - Multiple resolutions
- Extension segments can be used to store other user-defined data
Extension Marker Segments

- 2-byte APP0 = FF E0
- 2-byte Length = length of segment
- 5-byte ID = “JFXX” (last character is NULL)
- Extension Code defines type of extension data
  - 10_H = thumbnail encoded in JPEG
  - 11_H = thumbnail with 1-byte pixels and associated palette
  - 13_H = thumbnail with 24-bit RGB