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

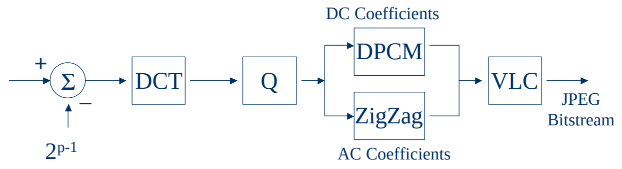
<u>Baseline</u>

1. sequential and

<u>Extended</u> versions:

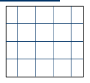
1. progressive (multiple scans at same spatial resolution)
2. lossless
3. hierarchical (multiple spatial resolutions)

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:

$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

- Inverse:

$$f(x) = \sum_{u=0}^{N-1} \alpha(u)C(u)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

- Where:

$$\alpha(u) = \begin{cases} \sqrt{1/N}; & \text{for } u = 0 \\ \sqrt{2/N}; & \text{for } u \geq 1 \end{cases}$$

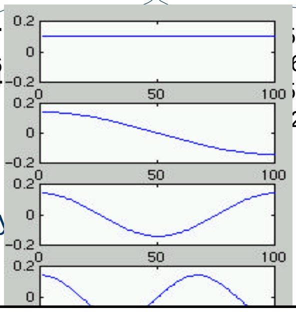
Definition, matrix form:

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

$$A = \frac{1}{\sqrt{N}}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \sqrt{2}\cos(3\pi/2N) & \cdots & \sqrt{2}\cos((2N-1)\pi/2N) \\ \vdots & & \ddots & \vdots \\ 1 & \sqrt{2}\cos(3(N-1)\pi/2N) & \cdots & \sqrt{2}\cos((2N^2-3N+1)\pi/2N) \end{pmatrix}$$

Example 4x4 Cosine transform

-



Cosine basis

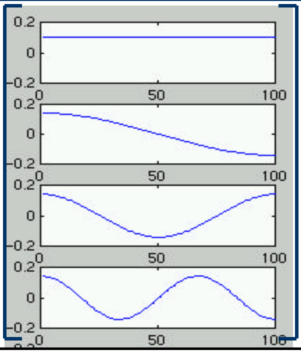
Test "image"

$$\begin{bmatrix} & & & 5 \\ & & & 65 \\ & & & 5 \\ & & & 27 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Transformed Im

- Why

Why?? Transforms graphically

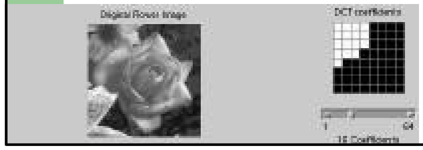


$$\begin{bmatrix} 1 \\ 0.5 \\ -0.5 \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1.58 \\ 0 \\ -0.11 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 5 \\ -2.23 \\ 0 \\ -0.16 \end{bmatrix}$$

Quantization

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

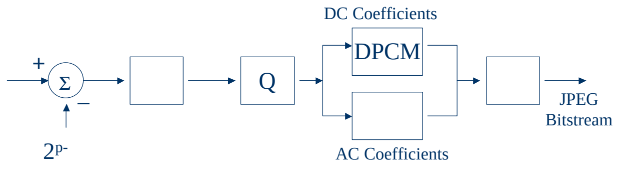


45	23	3	2	12
22	12	1	2	1
7	5	5	9	1
3	1	3	1	1
1	2	11	1	1

Example Z mask

17	18	24	47	99	99	99	99
18	21	26	66	99	99	60	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99		99	99		99	99	

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

- See Gonzalez table 6.14 and 6.15

DC range coding

Hex	Binary	Difference Range
0	00	0
1	010	-1,1
2	011	-3...-2,2...3
3	100	-7...-4,4...7
4	101	-15...-8,8...15
5	110	-31...-16,16...31
6	1110	-63...-32,32...63
⋮	⋮	⋮
B	111111110	-2047...-1024,1024...2047

Range + difference

- 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

$50 \rightarrow$ '1110' $diff_{LB}$ = 50-32=18$\rightarrow$ '110010'

$9 \rightarrow$ '101' $diff_{LB}$ =9-8=1$\rightarrow$ '1001'

$0 \rightarrow$ '00' (no $diff_{LB}$)

$-19 \rightarrow$ '110' $diff_{LB}$ = -19+16= -3 $\rightarrow$ '01100'

$1 \rightarrow$ '010' $diff_{LB}$ is sign bit '1'

$-1 \rightarrow$ '010' $diff_{LB}$ is sign bit '0'

$I_{DC}(x,y)$

50	59	59
40	41	40

$E_{DC}(x,y)$

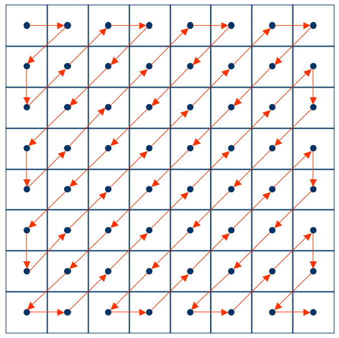
50	9	0
-19	1	-1

Encoded data = 1110 110010 101 1001 00 110 01100 010 1 010 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 AC_0 , $AC_1 \ldots AC_n$
- Run-length encode AC coefficients using format: $(R_0S_0)LB_0$, (R_1S_1) $LB_1 \ldots$ $(R_nS_n)LB_n$,EOB
  - R_i = 4-bit number of zero coefficients between AC_i and AC_{i-1} (AC_{-1} = DC)
  - S_i = size category for quantized coefficient AC_i
  - EOB = ‘1010’
  - R_iS_i = F0 is used to extend runs of longer than 15

Example: AC coefficients

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

0,2 $\rightarrow$ ’01’	-3+2 = -1 $\rightarrow$ ’00’
0,2 $\rightarrow$ ’01’	2-2 = 0 $\rightarrow$ ’10’
0,1 $\rightarrow$ ’00’	just sign ‘1’
0,1 $\rightarrow$ ’00’	just sign ‘0’
0,1 $\rightarrow$ ’00’	just sign ‘1’
5,1 $\rightarrow$ ’1111 010’	just sign ‘0’
0,0 $\rightarrow$ 1010 (EOB=End of Block/defined 0,0)	

Enc. data:

$\underbrace{\text{1110 100111}}_{\text{DC}}\ \underbrace{\text{0100 0110 001 000 001 11110100 1010}}_{\text{AC component}}$

JPEG File Format

JPEG Stand-alone Markers

Value	Symbol	Description
FF01	TEM	Temporary (used for Arithmetic coding)
FFD0-FFD7	RST_0-RST_7	Reset Marker
FFD8	SOI	Start of Image
FFD9	EOI	End of Image

JPEG File Format

JPEG Data Markers (continued)

FF02-BF	RES	Reserved
FFC0	SOF_0	Start of Frame, SOF baseline
FFC1	SOF_1	SOF extended sequential mode
FFC2	SOF_2	SOF progressive mode
FFC3	SOF_3	SOF lossless mode
FFC4	DHT	Define Huffman Table
FFC5	SOF_5	SOF differential sequential mode
FFC6	SOF_6	SOF differential progressive mode
FFC7	SOF_7	SOF differential lossless mode
FFC8	JPG	Reserved
FFC9	SOF_9	SOF extended sequential, arithmetic coding mode
FFCA	SOF_{10}	SOF progressive, arithmetic coding
FFCB	SOF_{11}	SOF lossless arithmetic mode

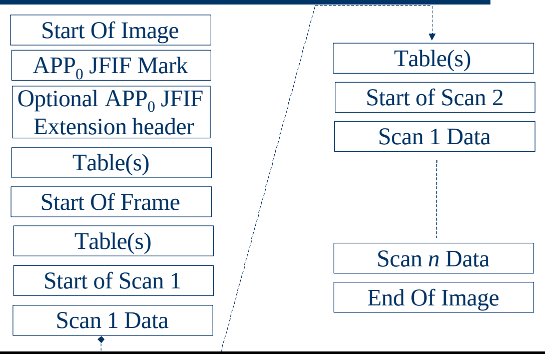
JPEG File Format

- 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 File Format
JPEG Data Markers

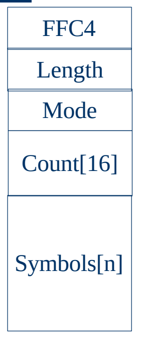
FFCC	DAC	Define arithmetic conditions
FFCD	SOF_{13}	SOF differential sequential, arithmetic
FFCE	SOF_{14}	SOF differential progressive, arithmetic
FFCF	SOF_{2}	SOF differential lossless, arithmetic
FFDA	SOS	Start of Scan
FFDB	DQT	Define Quantization Table
FFDC	DNL	Define Number of Lines
FFDD	DRI	Define Restart Interval
FFDE	DHP	Define Hierarchial progression
FFDF	EXP	Expand Reference Components
FFE0-EF	APP_0-APP_{15}	Application Specific Data
FFF0-FD	JPG_0-JPG_{13}	Reserved
FFFE	COM	Comment

JPEG File Format
JPEG Data Sequence



JPEG File Format
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]

JPEG File Format
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 = 1or 3)
- Component Specific Areas (1or 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)

JPEG File Format
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)

FFDA
length
chan
CSA(1)
CSA(2)
CSA(3)
SSS
SSE
SA

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

SOI
APP0
Length
ID
Version
Units
Xdensity
Ydensity
Xthumbnail
Ythumbnail
thumbimage

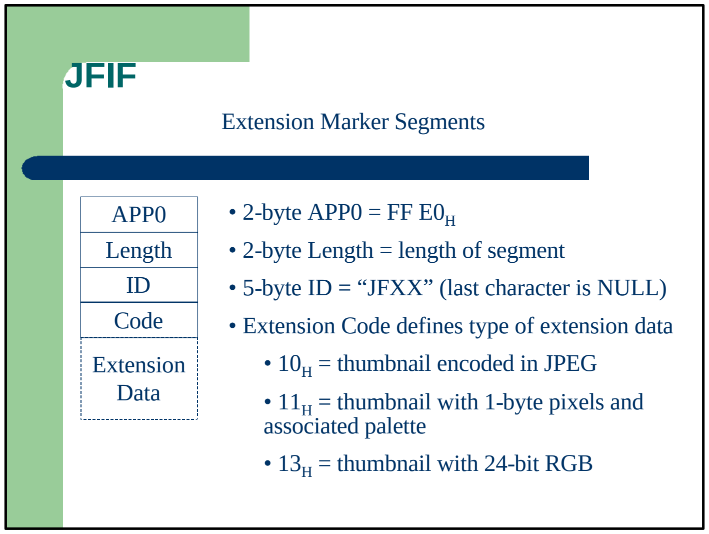
- 2-byte SOI = FF $D8_H$
- 2-byte APP0 = FF $E0_H$
- 2-byte Length = length of segment
- 5-byte ID = “JFIF” (last character is NULL)
- 2-byte Version (major and minor revision #s)
- 1-byte Units
  - 00_H = Not defined
  - 01_H = PEL/inch
  - 02_H = PEL/cm
- 2-byte Xdensity/Ydensity is Resolution in Units
- 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

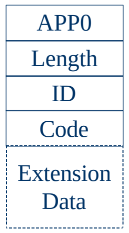
JFIF

- 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



JFIF

Extension Marker Segments



- 2-byte APP0 = FF $E0_H$
- 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