651 lecture | Fr 1/8/99

JPEG Lecture

(Ref's: Sayood: Section 12.6, Rabbani and Jones, the standard, G.K. Wallace, "The JPEG Still Picture Compression Standard," *Comm. ACM*, vol. 34, pp. 30-44, April 1991.)

The first and still principal image compression standard. circa 1990.

Conceptually fairly simple

There is a baseline version and extensions. I'll describe just the baseline version for black-white images.

Overview of JPEG Encoding

1. Segment, 2. Transform, 3. Quantize, 4. Binary encode

Overview of JPEG Decoding

4. Binary decode, 3. Inverse quantize, 2. Inverse transform, 1. Reassemble segments into image

Draw Block Diagram

Images

Black-white **image** $X = [X(i,j),\ i=1,...,n,\ j=1,...,n]$ as a **matrix**, e.g. 512x512.

i denotes row, j denotes column

Ordinarly, each $X(i,j) \in \{0, 1, 2, ...,255\}$; i.e. 8 bits per pixel, which is what JPEG ordinarly expects. 0 = black, 255 = white

We could reduce the rate by dropping bits. R = 7 looks very good (128 levels), R = 6 looks pretty good (64), R = 5 or below shows false contouring.

Details

1. **Segment** into 8x8 square blocks (each will be processed independently)

   Let $B = [B(i,j),\ i=0,...,7,\ j=0,...,7]$ denote a one of the blocks. we'll describe how to quantize it.

2. **Transformation:**

encoder does **DCT** transform of each 8x8:

$$C(u,v) = 4\, c_u\, c_v \frac{1}{n^2} \sum_{i=0}^{7} \sum_{j=0}^{7} B(i,j) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16}, \quad u,v = 0,\ldots,7$$

$$\text{where } c_u = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & u = 1,\ldots,7 \end{cases}$$

decoder does inverse DCT:

$$B(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} c_u c_v C(u,v) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16}, \quad i,j = 0,\ldots,7$$

this shows the "meaning" of the coeffiecients, specfically we interpret C(u,v) as a frequency coefficient

C(u,v) = strength of sinusoidal component with horizontal frequency u and vertical frequency v

$$(u,v) \text{ th component} = C(u,v) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16}$$

show picture of frequency basis matrices

Notes:

there is fast implementation based on FFT

$C(u,v) \in \{-1024, \ldots, 1023\}$

can also view as a matrix multiply $C = T\,B$, where T is a 64x64 orthogonal matrix and C and B are 64-dimensional column vectors.

3. Quantization

encoder produces *index* that describes coefficient:

$$I(u,v) = \text{round}\big(B(u,v)/M(u,v)\big) \qquad \text{(round to nearest integer)}$$

example: $I(u,v) = 7$ means that $6.5 \le M(u,v) \le B(u,v) \le 7.5\, M(u,v)$

M is called "the quantization matrix". It's a table of integer quantizer accuracies.

"quantization matrix" is term used only by JPEG

Small values lead to fine quantizations.

decoder converts index to a quantized coefficient:

$$\hat{C}(u,v) = I(u,v)\, M(u,v)$$

M can be specified by the encoder. There is no official standard, but JPEG did list an example M_o that most implementations use. Actually, they typcially, use

$M = M_o/f$, where f is quality factor. Large f leads to high quality.

Handout quantization matrix

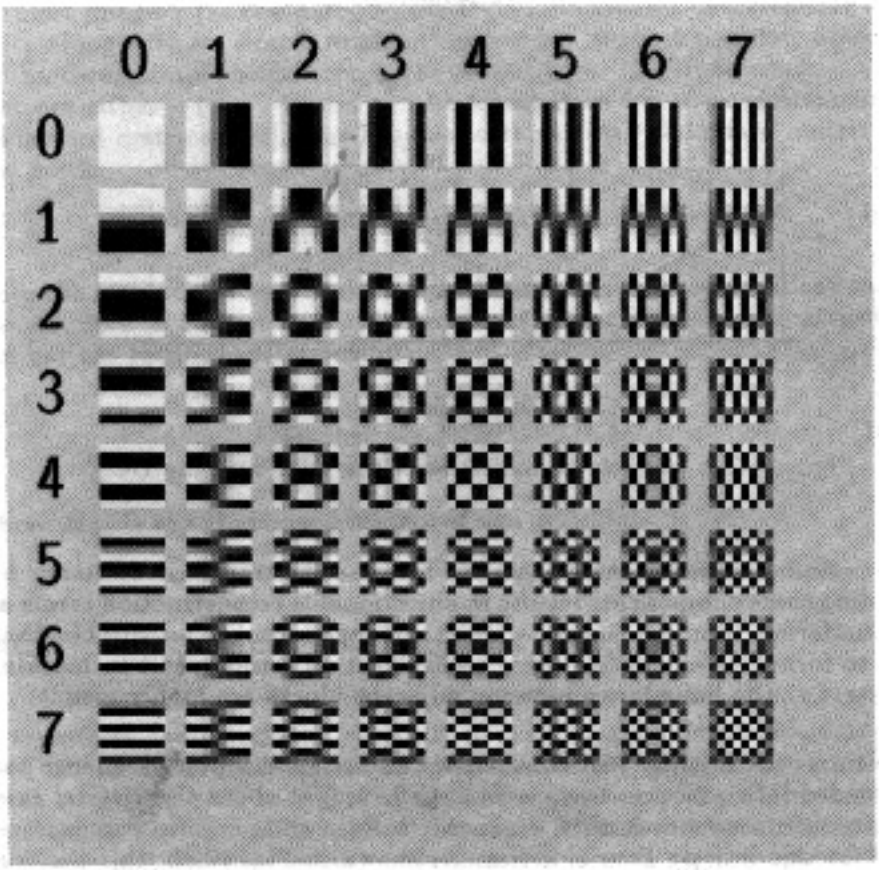


Figure 10.3: 8×8 DCT basis functions.

4. **binary encode**

DC coeff's

C(0,0) = avg of block B = DC coefficient

neighborhing blocks are similar
binary encode I(0,0) - previous I(0,0) using variable length code

C(0,0) - previous $\hat{C}(0,0)$
an example of predictive encoding (DPCM)

AC coeff's (the other coefficients)

zig-zag runlength encoding -- arrange coefficients in zig-zig order $Z_1\ Z_2\ Z_3\ Z_4$ ...

many of these are zero, so

transform into value and runlength sequence $V_1,R_1,\ V_2,R_2,$ ...

V_i = ith nonzero value, R_i = number of zeros since last value. R_i limited to 16.

transform value V into sign, size/category S and value W

$V = \text{sign}(V)\ (2^{S-1} + W),\quad S = 1,...,10,\quad W = 0,...,2^{S-1}-1$

V ranges from 2^{S-1} to 2^S-1 and the negative of these

there are 2^{S-1} values so it takes S-1 bits plus sign bit

encode (S_i,R_i) together using variable-length encoding table. (**show table**)

encode W_i using S_i-1 bits.

Performance

rate and distortion

Complexity

encoder: 2.25 multiplies, 9.25 additions, 1 comparision per pixel
(Rabbani and Jones, p. 122)

Issues:

Why? transform
scalar quantize one at a time
uniform quantizer
why this binary coding techniques

we see the pattern -- processing, quantization, binary coding

Annex K (informative)

Examples and guidelines

This annex provides examples of various tables, procedures, and other guidelines.

K.1 Quantization tables for luminance and chrominance components

Two examples of quantization tables are given in tables K.1 and K.2. These quantization values have been used with good results on 8 bit per sample Y,Cb,Cr images of the format illustrated in Figure 13. Note that these quantization values are appropriate for the DCT normalization defined in A.3.3.

Table K.1 Luminance quantization table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table K.2 Chrominance quantization table

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

If these quantization values are divided by 2, the resulting reconstructed image is usually nearly indistinguishable from the source image.

K.2 A procedure for generating the lists which specify a Huffman code table

A Huffman table is generated from a collection of statistics in two steps. The first step is the generation of the list of lengths and values which are in accord with the rules for generating the Huffman code tables. The second step is the generation of the Huffman code table from the list of lengths and values.

The first step, the topic of this section, is needed only for custom Huffman table generation and is done only in the encoder. In this step the statistics are used to create a table associating each value to be coded with the size (in bits) of the corresponding Huffman code. This table is sorted by code size.

A procedure for creating a Huffman table for a set of up to 256 symbols is shown in Figure K.1. Three vectors are defined for this procedure:

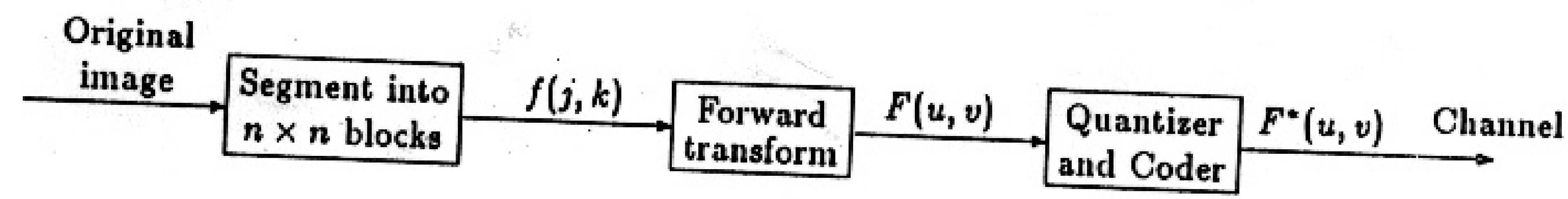
```
FREQ(V)       frequency of occurrence of symbol V.
CODESIZE(V)   code size of symbol V.
OTHERS(V)     index to next symbol in chain of all symbols in
              current branch of code tree.
```

where V goes from 0 to 256.

Discussion In Class 455 ~~Monday 2-12-96~~

651
W97
Fri, 9-5-97

Transmitter



Receiver

An Image Block

$$f(j,k) = \begin{bmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

The DCT transform of the image block

$$F(u,v) = \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}.$$

The usual quantization matrix

$$Q(u,v) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

The quantized indices

$$F^*(u,v) = \begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

zig-zag ordering

$$\begin{bmatrix}
0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\
2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\
3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\
9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\
10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\
20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\
21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\
35 & 36 & 48 & 49 & 57 & 58 & 62 & 63
\end{bmatrix}$$

Category	AC Coefficient Range
1	$-1, 1$
2	$-3, -2, 2, 3$
3	$-7, \cdots, -4, 4, \cdots, 7$
4	$-15, \cdots, -8, 8, \cdots, 15$
5	$-31, \cdots, -16, 16, \cdots, 31$
6	$-63, \cdots, -32, 32, \cdots, 63$
7	$-127, \cdots, -64, 64, \cdots, 127$
8	$-255, \cdots, -128, 128, \cdots, 255$
9	$-511, \cdots, -256, 256, \cdots, 511$
10	$-1023, \cdots, -512, 512, \cdots, 1023$

Table 10.1: AC coefficient grouping.

Category	AC Coefficient Range
1	$-1, 1$
2	$-3, -2, 2, 3$
3	$-7, \cdots, -4, 4, \cdots, 7$
4	$-15, \cdots, -8, 8, \cdots, 15$
5	$-31, \cdots, -16, 16, \cdots, 31$
6	$-63, \cdots, -32, 32, \cdots, 63$
7	$-127, \cdots, -64, 64, \cdots, 127$
8	$-255, \cdots, -128, 128, \cdots, 255$
9	$-511, \cdots, -256, 256, \cdots, 511$
10	$-1023, \cdots, -512, 512, \cdots, 1023$

Table 10.1: AC coefficient grouping.

Zero Run	Category	Code length	Codeword
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
.	.	.	.
.	.	.	.
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
.	.	.	.
.	.	.	.
2	1	5	11011
2	2	8	11111000
.	.	.	.
.	.	.	.
3	1	6	111010
3	2	9	111110111
.	.	.	.
.	.	.	.
4	1	6	111011
5	1	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
11	1	9	111111010
.	.	.	.
16 0's	.	.	ZRL 111111001
End of Block (EOB)		4	1010

Table 10.2: Example of JPEG AC Huffman code table.

zeroes to the end

reconstructed image block

$$\hat{f}(j,k) = \begin{bmatrix} 144 & 146 & 149 & 152 & 154 & 156 & 156 & 156 \\ 148 & 150 & 152 & 154 & 156 & 156 & 156 & 156 \\ 155 & 156 & 157 & 158 & 158 & 157 & 156 & 155 \\ 160 & 161 & 161 & 162 & 161 & 159 & 157 & 155 \\ 163 & 163 & 164 & 163 & 162 & 160 & 158 & 156 \\ 163 & 163 & 164 & 164 & 162 & 160 & 158 & 157 \\ 160 & 161 & 162 & 162 & 162 & 161 & 159 & 158 \\ 158 & 159 & 161 & 161 & 162 & 161 & 159 & 158 \end{bmatrix}$$

errors in reconstructed block

$$e(j,k) = \begin{bmatrix} -5 & -2 & 0 & 1 & 1 & -1 & -1 & -1 \\ -4 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\ -5 & -1 & 3 & 5 & 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & -2 & -1 & 0 & 2 & 4 \\ -4 & -3 & -3 & -1 & 0 & -5 & -3 & -1 \\ -2 & -2 & -3 & -3 & -2 & -3 & -1 & 0 \\ 2 & 1 & -1 & 1 & 0 & -4 & -2 & -1 \\ 4 & 3 & 0 & 0 & 1 & -3 & -1 & 0 \end{bmatrix},$$