


\*Thanks to Greg Stitt for providing/assembling much of the included information.

## JPEG Image Compression

### Color Transform

- JPEG capable of using any color model
  - RGB, HSI, CMY, YUV, YCbCr
- RGB, HSI, CMY spread useful visual information across 3 components
  - Hard to select information to discard
- In YCbCr, most of useful information is in Y space
  - Some of Cb and Cr can be discarded
- JPEG can transform color model to improve compression

 **JPEG Image Compression**  
*RGB vs. YCbCr*

- RGB
  - R – Red
  - G – Green
  - B – Blue
- YCbCr
  - Y – Luma (Brightness)
  - Cb – Cb Chroma (Blue minus Luma)
  - Cr – Cr Chroma (Red minus Luma)
  
  - Luma - Amount of light that passes through or is emitted from a particular area
  - Chroma - Difference between a color and a chosen reference color of the same luminous intensity

3

 **JPEG Image Compression**  
*RGB vs. YCbCr*

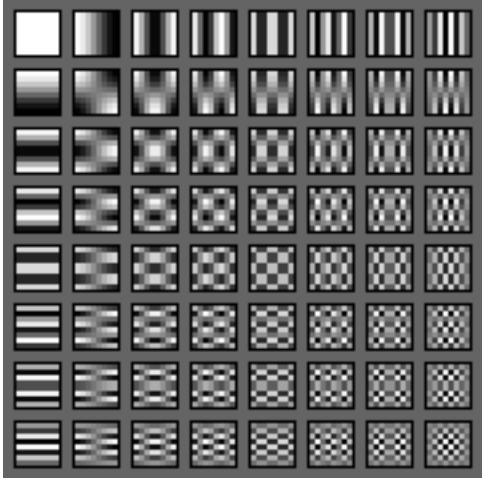
- RGB
  - 
  - 
  - 
- YCbCr
  - 
  - 
  - 

4



## JPEG Image Compression

*Frequency Domain*



The image displays an 8x8 grid of 64 DCT basis functions. Each cell in the grid contains a grayscale pattern representing a specific frequency component. The top-left cell is a solid white square, representing the DC component (0,0). As the frequency increases towards the bottom-right, the patterns become more complex, showing horizontal and vertical lines, and eventually checkerboard-like patterns.

7

## JPEG Image Compression

*DCT – Discrete Cosine Transform*

- Output of DCT is array of DCT coefficients representing different strengths of each frequency
  - For JPEG, DCT is done on 8x8 blocks
  - 8x8 DCT coefficients
- Top left is low frequencies
  - (0,0) is DC value, or average, of all pixels
- Bottom right is strength of cosine wave alternating from max to min every pixel

8



## JPEG Image Compression

### DCT – Discrete Cosine Transform

#### ■ DCT Equation:

$$F(u, v) = \frac{1}{4}C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 D[x, y] \cos\left(\frac{\pi(2x+1)u}{16}\right) \cos\left(\frac{\pi(2y+1)v}{16}\right)$$

$$C(h) = \begin{cases} \frac{1}{\sqrt{2}}, & h = 0 \\ 1, & \text{otherwise} \end{cases}$$

- $D[x, y]$  = original pixel value at location  $x, y$
- $F(u, v)$  = output coefficient at location  $x, y$



## JPEG Image Compression

### DCT – Discrete Cosine Transform

- DCT operates on 8x8 blocks,
  - $u, v, x,$  and  $y$  only range in value from 0 to 7
  - Precompute 64 possible cosine values and store in an 8x8 table,
- Rewrite DCT Equation:

$$F(u, v) = \frac{1}{4}C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 D[x, y] \cos[x, u] \cos[y, v]$$

- $D[x, y]$  = original pixel value at location  $x, y$
- $F(u, v)$  = output coefficient at location  $x, y$

## JPEG Image Compression

### *DCT – Discrete Cosine Transform*

- Reason for using DCT
  - Important to separate frequencies because human eye is more sensitive to low frequency information
  - Can discard some of high-frequency information

11

## JPEG Image Compression

### *Quantization*

- Divides each DCT coefficient by some value to reduce total number of bits
  - Many of the high frequency coefficients become 0
  - Other information is lost from integer division
  - This is where the loss in JPEG comes from

89	62	45	23	14	53	12	32
62	74	32	76	12	23	12	19
49	2	67	23	35	21	94	12
42	32	12	32	42	59	15	16
31	12	..	..				


/

54	44	40	64	98	160	204	244
48	48	56	76	104	232	240	220
56	52	64	96	160	220	276	224
50	68	96	116	204	300	300	248
72	96	148	224	252	300	300	300
96	140	220	256	300	300	300	300
104	256	300	300	300	300	300	300
288	300	300	300	300	300	300	300

=

1	1	1	0	0	0	0	0
1	1	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	0	0	0

12




## JPEG Image Compression

### Quantization

- Different Types
  - Constant amount
  - Varying amount for different frequencies
  - Separate values for luminance and chrominance
    - Larger amount for chrominance
  
- Quantization table stored in JPEG file
  - Generally a sample table is used
  - Different "Quality" jpegs use different quantization tables

13



## JPEG Image Compression

### Quantization

- Example quantization tables:

The *Luminance Quantization Table*  $q(u, v)$

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

The *Chrominance Quantization Table*  $q(u, v)$

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

14



## JPEG Image Compression

Quantization



*Qual = 100*  
*Size = 60.8 KB*



*Qual = 75*  
*Size = 12.8 KB*



*Qual = 50*  
*Size = 6.9 KB*



## JPEG Image Compression

Quantization



*Qual = 25*  
*Size = 2.6 KB*



*Qual = 10*  
*Size = 1.8 KB*



*Qual = 1*  
*Size = 1.6 KB*

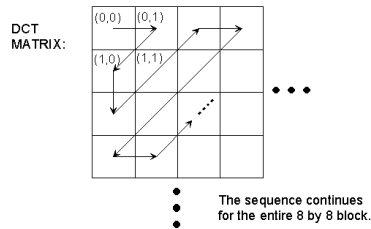


## JPEG Image Compression

### Encoding

- After quantization, many DCT coefficients are 0, especially for high frequencies
  - High frequencies appear in bottom right of 8x8 block
  - Grouping these 0s together makes compression better
  - Zig-zag converts 8x8 block into 16x1, in the following order:

FIGURE 2. ZIG-ZAG SEQUENCE FOR BINARY ENCODING



17

## JPEG Image Compression

### Encoding

- Run length encoding
  - For each non-zero DCT coefficient, JPEG records the number of zeros that preceded the number, the number of bits needed to represent the number's amplitude, and the amplitude itself.

18

## JPEG Image Compression

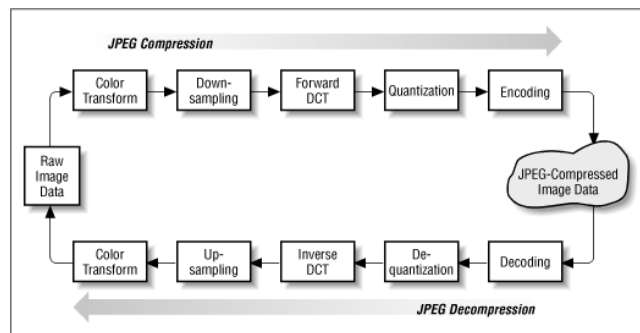
### Encoding

- After run length encoding, Huffman encoding further compresses the 8x8 block
- After compression, block is written to file along with end of block marker

19

## JPEG Image Decompression

- Basically opposite of encode
  - Decompress, Dequantize, Inverse DCT, Up-sampling, color transform



20