Subsampling

- Selecting one single value to represent several values in a part of the image.
  - For example, use top left corner of 2X2 block to represent the block
  - Compression ratio 75%

11	15	19	55	→	11	11	19	19
13	14	21	32		11	11	19	19
39	17	24	76		39	39	24	24
43	34	27	80		39	39	24	24

Subsampling

- A better way- averaging
- Compression ratio 75%

11	15	19	55		13	13	32	32
13	14	21	32	→	13	13	32	32
39	17	24	76		33	33	51	51
43	34	27	80		33	33	51	51

Subsampling

- Subsample in non-square blocks
- Different components may be subsampled at different frequencies
- Works best for images with low frequency components
- Suitable when target output resolution is lower than source resolution
- Works poorly on images with fine details, including text

Quantization

- Mapping of a large range of possible sample values into a smaller range of values or codes.
- Fewer bits are required to encode the quantized sample.
- Examples
  - -Letter grades (A, B, C, D, F)
  - Rounding of person’s age, height, or weight

Quantization

- Truncation and Rounding
- Quantized levels need not be evenly spaced
- Can be used for relative as well as absolute information
- Information is lost in quantiztion, but the error can be recovered

Truncation

- Discard lower-order bits
  – average error 1/2 LSB of target resolution
- Example

9	11	17	21		0	10	10	20
19	51	33	14	→	10	50	30	10
19	23	18	15		10	20	10	10
53	47	12	43		50	40	10	40

Rounding

- Add 5 and then truncate the result.
  - One more LSB participate than in truncation
  - average error 1/4 LSB

13	19	9	5		10	20	10	10
14	17	8	15		10	20	10	20
52	49	53	47	→	50	50	50	50
50	58	51	42		50	60	50	40

Random Rounding

- Add a random number in the range [0,LSB], then truncate to LSB.
- Decimal number "43" has 70% chance of being "40" and 30% chance of being "50".
- Information in all bits participates.
- Average error 1/3 LSB (higher than rounding, but results look better.)
- Identical pixels may be rounded to different values.
- Colors not available in target color space may result.

Error Diffusion

- Quantize the number, subtract quantization error from adjacent pixels that have not been quantized.
  - Preserves color levels over very localized areas
  - Every bit contributes to the final image

Error Diffusion

17	14	12	19
20	11	12	19
	10	13	19
		10	22
			20

Result 20 10 10 20

Floyd-Steinberg

- Qunatize one pixel
- Distribute error to four of its neighbors (scan order).

	E	7/16
3/16	5/16	1/16

Example

200	500	800	400	800
800	600	553	A	613
12	B	C	D	423
612	916	453	395	532

→

200	500	800	400	800
800	600	600	A-21	613
12	B-9	C-15	D-2	423
612	916	453	395	532

Delta Coding

- Code the difference between adjacent pixels.
- Since adjacent pixels are similar, the difference is normally small, and requires fewer bits to code.
- A typical pixel value requires 8 bits.
- The difference between any 8 bit pixels is in the range [-255,255], which needs 9 bits!

Delta Coding

- But most deltas will be small.
  - Smaller deltas can be assigned shorter codes
  - Smaller deltas can be ignored completely
  - smaller deltas can be quantized more finally for better quality
- Complementary delta values can share a code; e.g., +1 and -255 yield same result in 8 bit positive value.
- 9 bits are not required!

Encoding with quantization loss

- Encoder must calculate incorrect pixel value that the decoder will decode, and use that value in computing the next delta, to minimize the quantization loss.

Prediction

- Prediction further reduces delta values.
- In delta coding prediction is the last pixel
- Better prediction algorithm means better compression ratio.
- It can improve picture quality

Prediction

- Use left pixel (delta coding)
- Use linear interpolation (left+(left-previous))
- Use 2d interpolation (left+above-corner)

Run-length Encoding (RLE)

- Image compression method that works by counting the number of adjacent pixels with the same gray levels values.
- Many consecutive zeros in deltas resulting from prediction can be coded compactly.

RLE: Example

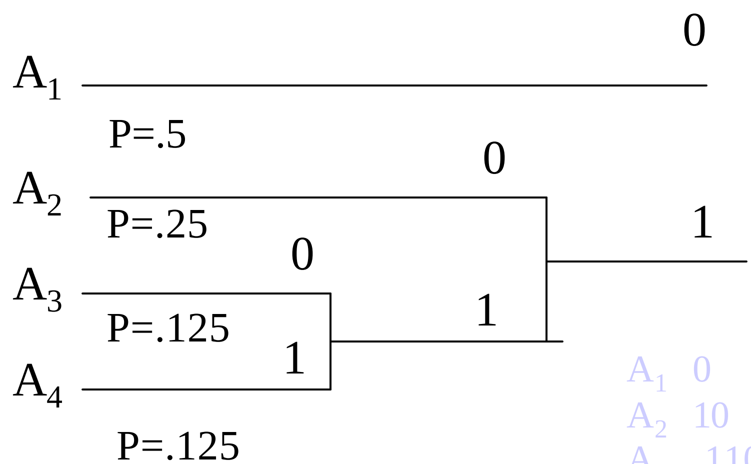
8
0, 4, 4
1, 2, 5
1, 5, 2
1, 3, 2, 1, 1
2, 1, 2, 2, 1
0, 4, 1, 1, 2
8

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Huffman Coding

- Given “n” possible symbols we need $\log_2(n)$ bits to code them using binary system.
- If probability of occurrence of these symbols is not uniform, then we can code them using variable number of bits.
- This is lossless and efficient coding.
- Assign shorter codes to more frequent symbols, and longer codes to less frequent.

Huffman Coding



A_1 0
A_2 10
A_3 110
A_4 111

Huffman Coding

Entropy

$$H = -.5\log.5 - .25\log.25 - .125\log.125 - .125\log.125 = 1.75$$

Codeword length

$$R = .5\times1 + .25\times2 + .125\times3 + .125\times3 = 1.75$$

Image Compression

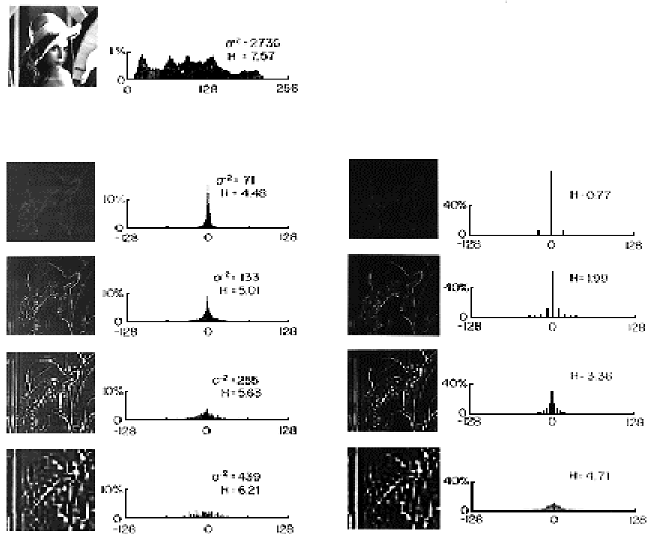
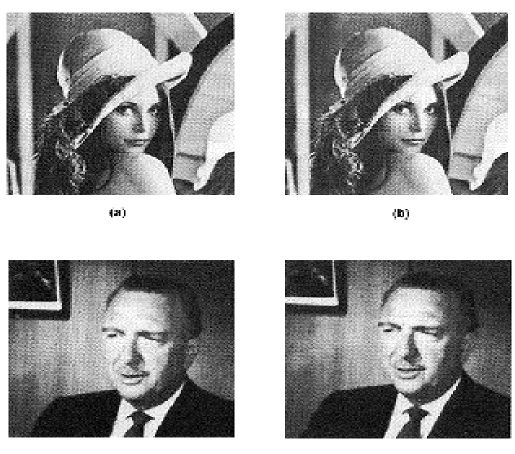


Image Compression

1.58

1

.73



Variable Length Coding (Vector Deltas)

0	1
1	010
2	0010
3	00010
4	0000110
5	00001010
….	
15	000000011010

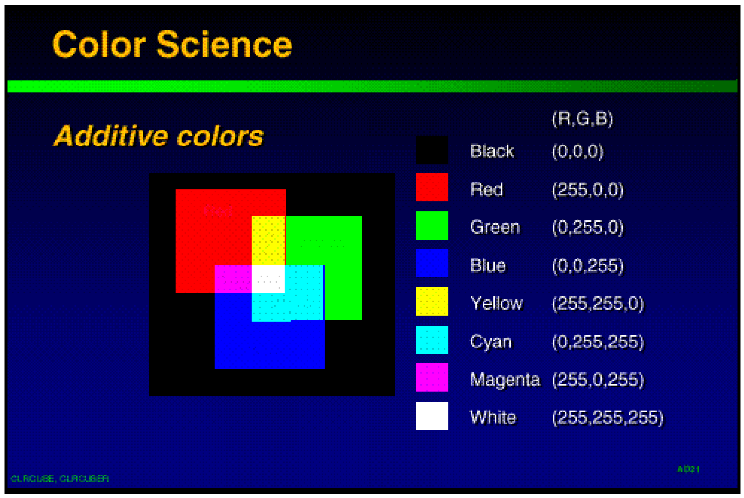
Variable Length Coding (DCT AC Coefficients)

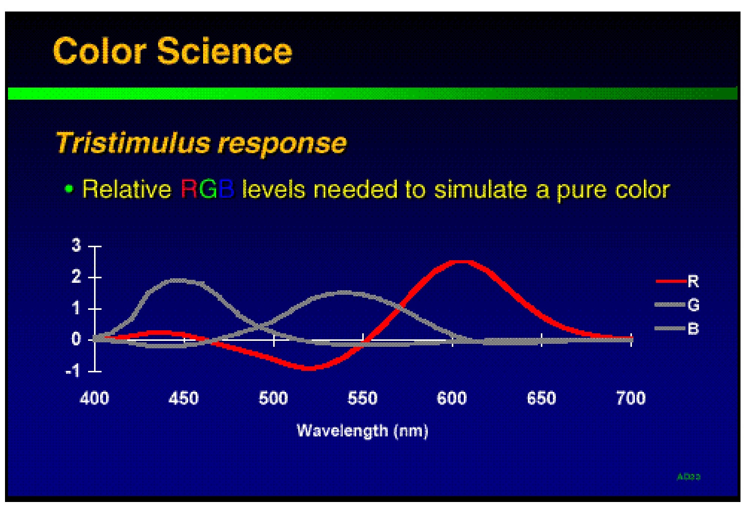
0,1	110
1,1	0110
0,-1	111
7,-1	0001001
EOB	10

Color Images

Color Science

- “Colors” as we perceive them are weighted sum of multiple wavelengths.
- Three types of photoreceptors in eye roughly correspond to Red, Green and Blue.
- We simulate colors by hitting those photoreceptors with calculated amounts of Red, Green and Blue.


Color Science
Additive colors
(R,G,B)
Black (0,0,0)
Red (255,0,0)
Green (0,255,0)
Blue (0,0,255)
Yellow (255,255,0)
Cyan (0,255,255)
Magenta (255,0,255)
White (255,255,255)


Color Science
Tristimulus response
• Relative RGB levels needed to simulate a pure color
3
2
1
0
-1
400
450
500
550
600
650
700
Wavelength (nm)
R
G
B

Color Spaces

- R, G, B
- Y, Cb, Cr
- Y, I, Q
- C, M, Y
- I, H, S
- Y, U, V

Luma & Chroma

$$Y = .3R + .6G + .1B$$

$$C_b = \frac{R - Y}{1.6} + .5$$

$$C_r = \frac{B - Y}{2} + .5$$

Y, I, Q

$$Y = .3R + .59G + .11B$$

$$I = .6R + .28G - .32B$$

$$Q = .21R - .52G + .31B$$

I=Red-Cyan
Q=magenta-green
Y=white-black

C, M, Y

$$C = 1 - R$$

$$M = 1 - G$$

$$Y = 1 - B$$

Cyan, Magenta and Yellow: Primary colors of pigments.

Intensity, Hue and Saturation

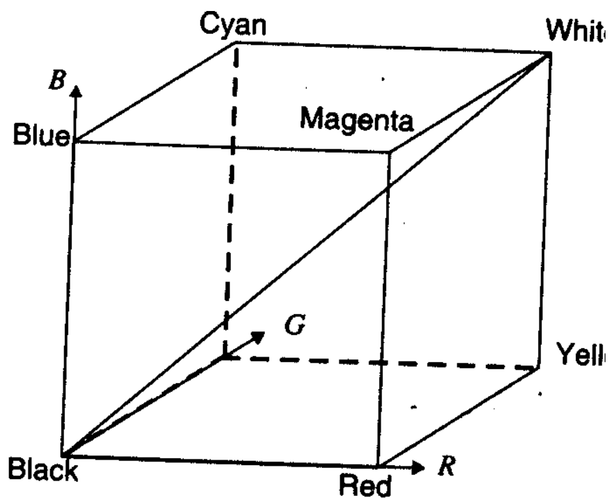
$$I = R + G + B$$

$$S = 1 - 3\frac{\min(R,G,B)}{I}$$

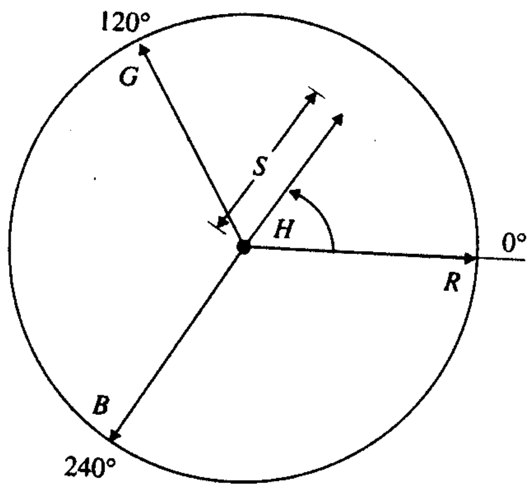
$$h = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

Saturation measures lack of whiteness in the color.
Hue is proportional to the average wavelength of the color. (A “deep”, “bright” “orange”.) (245,110,20)

The Color Cube



The Color Circle



The Color Cylinder

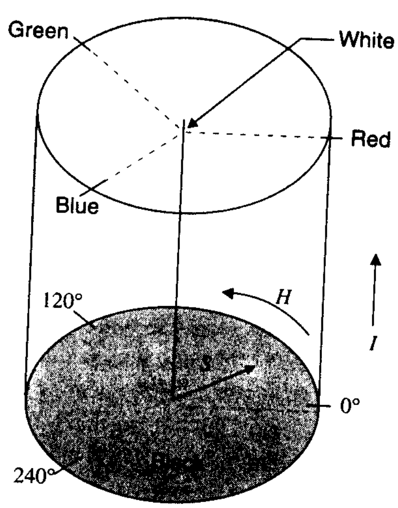


Figure 21

Y, U, V

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} .299 & .587 & .114 \\ -.169 & -.331 & .5 \\ .5 & -.419 & -.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Y represents the brightness of a pixel.
U, V represent how far blue and red are from white.

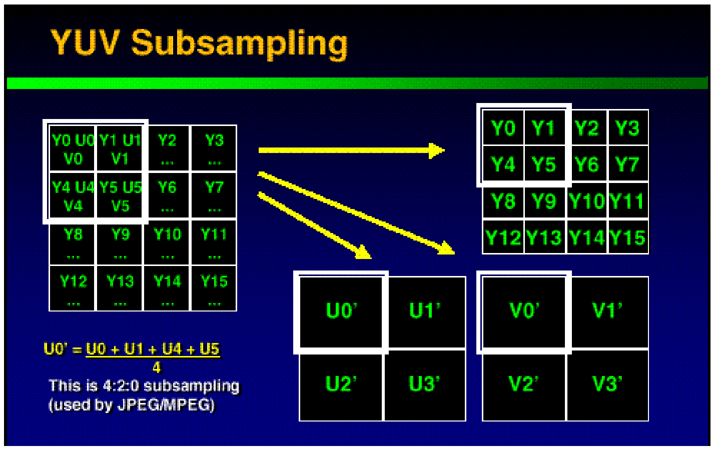
Average Delta Values for Adjacent Pixels

Y=13
U=1
V=1
YUV=13

R=13
G=13.2
B=12.7
RGB=13

We can sub-sample U & V over a number of pixels without loss of picture quality.

YUV Subsampling



Simple Compression Scheme

- Convert RGB to YUV space
- Predict each pixel's value from adjacent pixels
- Calculate deltas from the predicted values
- Quantize the differences
- Encode the quantized deltas, including run-length encoding
- Diffuse quantization error to nearby pixels

Decompression Scheme

- Predict each pixel's value components from adjacent pixels
- Decode the stored quantized difference (deltas)
- Add decoded delta to the predicted values
- Convert each pixel to RGB space
- Filter result to recapture lost information