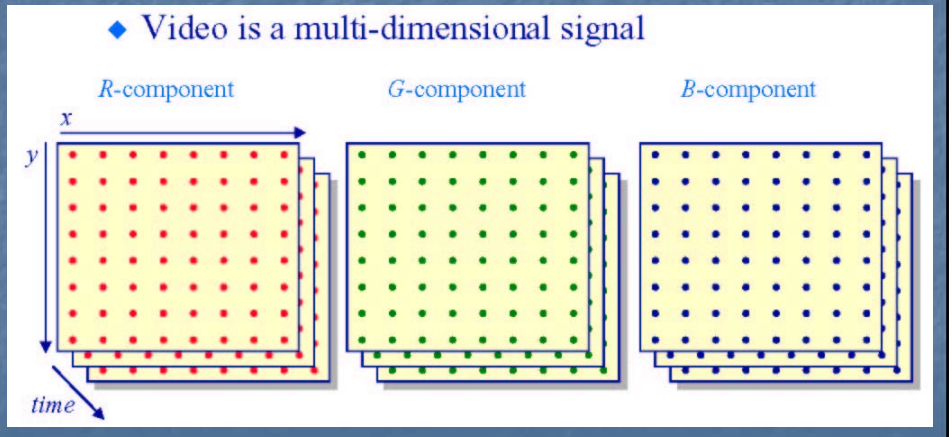
Image and Video Compression

ICS 280: Visual Perception

Image and Video



Formats

- RGB is not used for transmission of signals between capture and display devices
  - Too expensive, needs too much bandwidth
- Converted to luminance and chrominance formats
  - Use standard YIQ or YUV format

» $Y = 0.30R + 0.59G + 0.11B$
» $I = 0.60R - 0.28G - 0.32B$
» $Q = 0.21R - 0.52G + 0.31B$

» $Y = 0.3R + 0.6G + 0.1B$
» $U = (B - Y)$ x 0.493
» $V = (R - Y)$ x 0.877

Compression Basics

- Simple compression
- Interpolation-based techniques
- Transforms
- Statistical techniques

Compression Techniques

- How many bits we need?
  - Deals with perceptible color resolution
  - Has to do with difference threshold
- How much frequency do we need?
  - Deals with perceptible frequency
  - Both spatial and temporal
- How much can we perceive?

Compression Issues

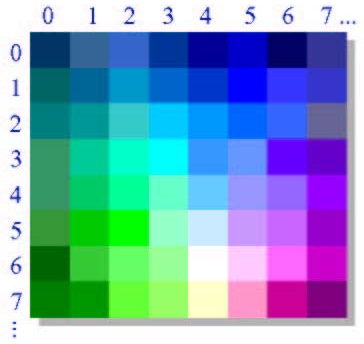
- Bandwidth requirements of resulting stream
  - Bits per pixel (bpp)
- Image quality
  - Compression/decompression speed
  - Latency
  - Cost
  - Symmetry
- Robustness
  - Tolerance of errors and loss
- Application requirements
  - Live video
  - Stored video

Bit Reduction

- Reducing the number of bits per pixel
  - » Throw away the least significant bits of each sample value
- Example
  - » Go from *RGB* at 8 bits/component sample (8:8:8) to 5 bits (5:5:5)
    - Go from 24 bpp to 15 bpp
    - This gives "acceptable results"
  - » Go from *YUV* at 8 bits/component sample 6:5:5 (16 bpp)
- Advantage — simple!

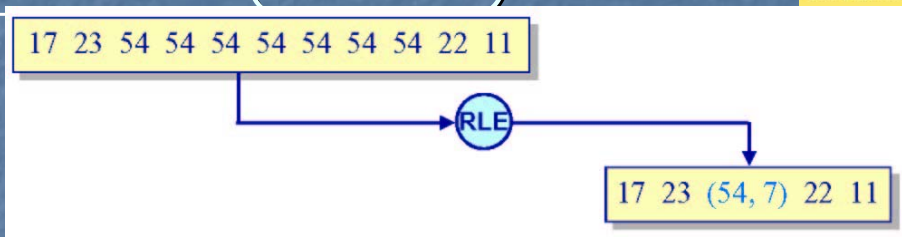
Color Look-Up-Table (Statistical)

- Quantize coarser in the color domain
  - » Pixel values represent indices into a color table
  - » Tables can be optimized for individual images
- Entries in color table stored at "full resolution" (*e.g.* 24 bits)



- Example:
  - » 8-bit indices (256 colors) gives
    $(440 \times 480) \times 8 + (24 \times 256) = 1.7 \times 10^6$ bits/sec

Run Length Encoding (Statistical)

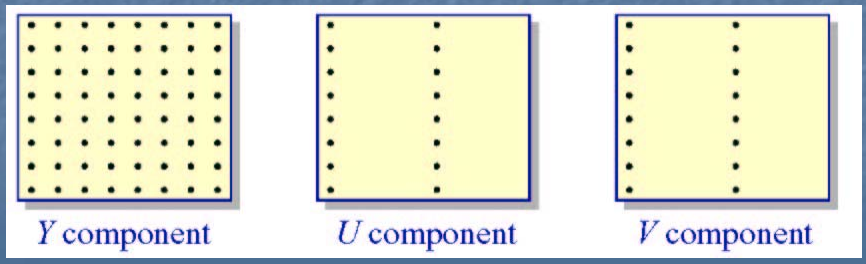


- Replace sequences of pixel components with identical values with a pair (*value*, *count*)
- Works well for computer-generated images, cartoons. works less well for natural video
- Also works well with CLUT encoded images

  (*i.e., multiple techniques may be effectively combined*)

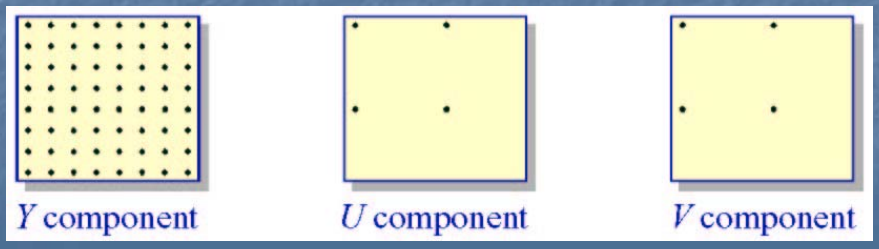
Interpolative Compression

- Acquire chrominance at lower resolution
  - Humans have lower chrominance acquity
- Sub-sample by a factor of four in horizontal and vertical direction



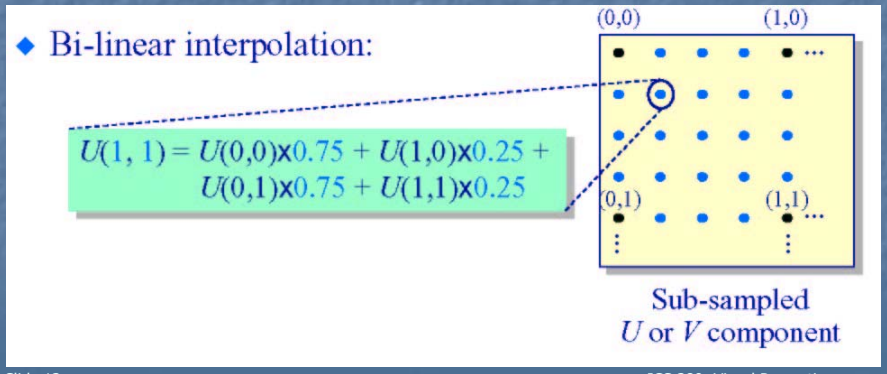
Interpolative Compression

- Acquire chrominance at lower resolution
  - Humans have lower chrominance acquity
- Sub-sample by a factor of four in horizontal and vertical direction

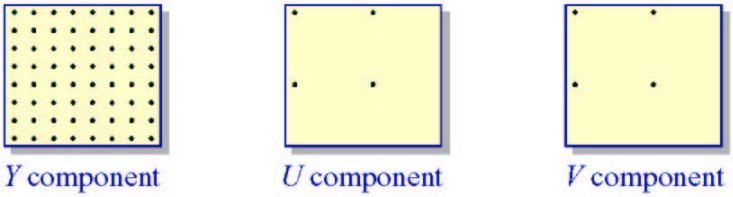


Reconstruction

- Using bilinear interpolation
- Gives excellent results



Significant Compression



Y component *U* component *V* component

- Storage/transmission requirements reduction:
  - » Within a 4x4 pixel block:

$$\text{bpp} = \frac{(8 \text{ bpp luminance}) \times 16 \text{ samples} + (8 \text{ bpp chrominance}) \times 2}{16}$$

$$= 9$$

  - » A 62.5% reduction overall

Statistical Compression

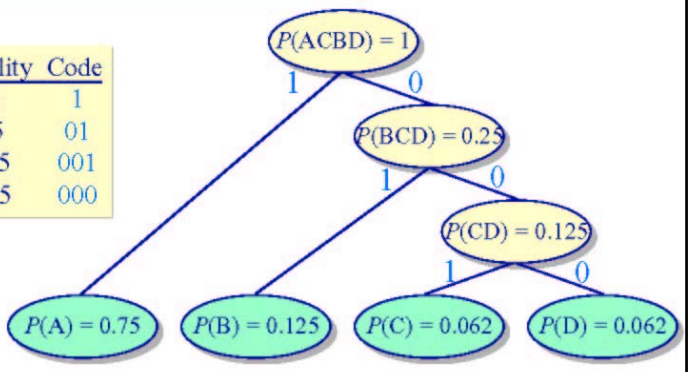
Huffman coding

- Exploit the fact that not all sample values are equally likely
  - » Samples values are non-uniformly distributed
  - » Encode "common" values with fewer bits and less common values with more bits
- Process each image to determine the statistical distribution of sample values
  - » Generate a *codebook* — a table used by the decoder to interpret variable length codes
  - » Codebook becomes part of the compressed image

Statistical Compression

Huffman coding

Symbol	Probability	Code
A	0.75	1
B	0.125	01
C	0.0625	001
D	0.0625	000

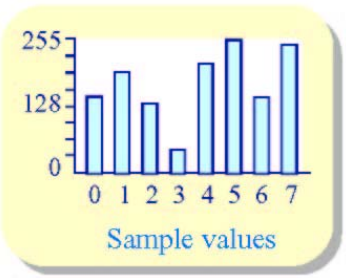


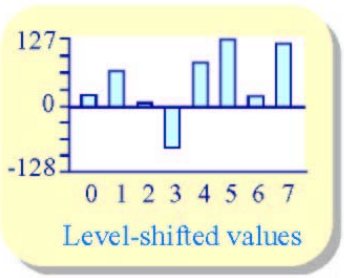
- Order all possible sample values in a binary tree by combining the least likely samples into a sub-tree
- Label the branches of the tree with 1's and 0's
  - » Huffman code is the sequence of 1's and 0's on the path from the root to the leaf node for the symbol

Transform-Based Compression

The Discrete Cosine Transform (DCT)

- A transformation into the frequency domain
- Example: 8 adjacent pixel values (*e.g.*, luminance)



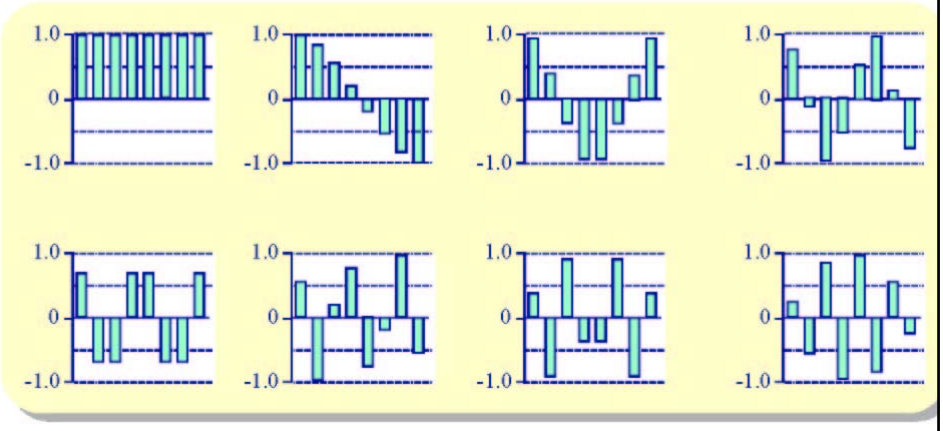


- What is the most compact way to represent this signal?

Transform-Based Compression

The Discrete Cosine Transform (DCT)

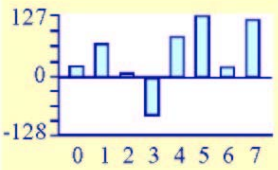
- Represent the signal in terms of a set of *cosine basis functions*



The Discrete Cosine Transform

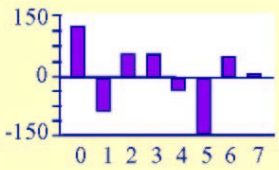
Represent input as a sum of scaled basis functions

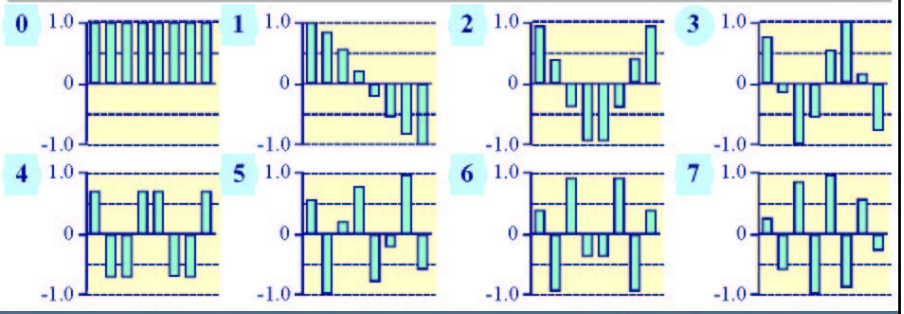
Level-shifted values



$$= \sum_{i=0}^{7} \left(\right)_i \times i$$

DCT coefficients





Transform-Based Compression

The Discrete Cosine Transform (DCT)

- The 1-dimensional transform:

$$F(\mu) = \frac{C(\mu)}{2} \sum_{x=1}^{7} f(x) \cos \frac{(2x+1)\mu\pi}{16}$$

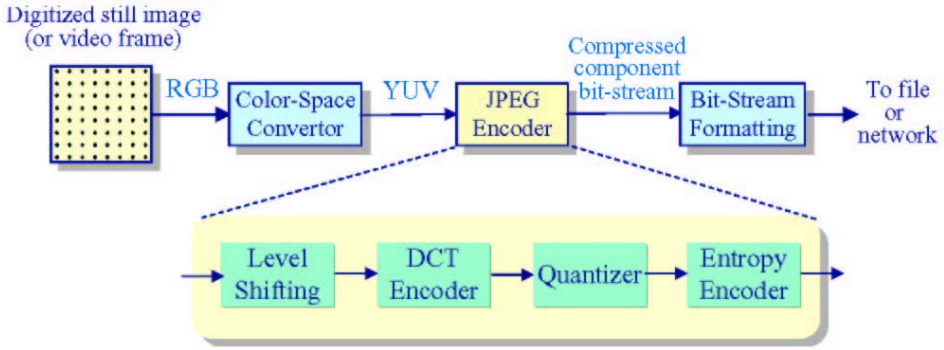
  - » $F(\mu)$ is the DCT coefficient for $\mu = 0..7$
  - » $f(x)$ is the x^{th} input sample fot $x = 0..7$
  - » $C(\mu)$ is a constant (equal to $2^{-0.5}$ if $\mu = 0$ and 1 otherwise)
- The 2-dimensional (spatial) transform:

$$F(\mu,\nu) = \frac{C(\mu)}{2}\frac{C(\nu)}{2} \sum_{y=1}^{7}\sum_{x=1}^{7} f(x,y) \cos \frac{(2x+1)\mu\pi}{16} \cos \frac{(2y+1)\nu\pi}{16}$$

JPEG Compression

Encoder architecture — sequential mode

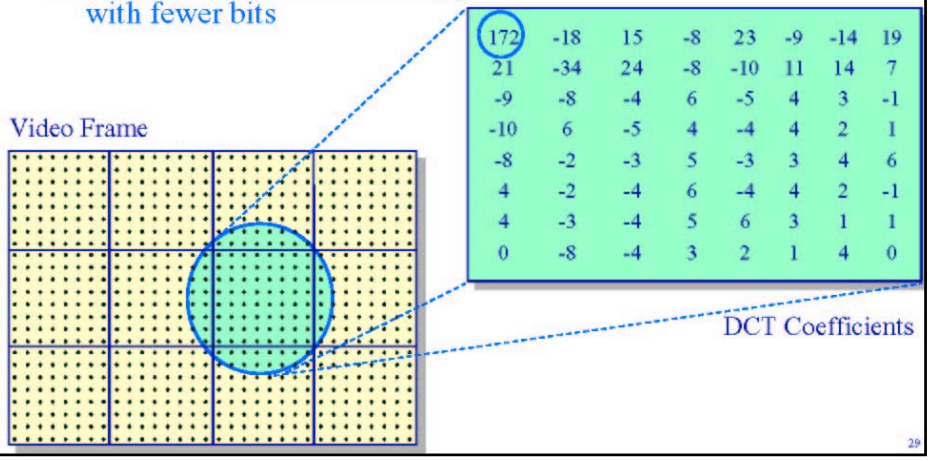
- Inputs are 8 or 12-bit samples
  - » baseline = 8-bit samples
- Image components are compressed separately
  - » DCT operates on 8x8 pixel blocks



Transform-Based Compression

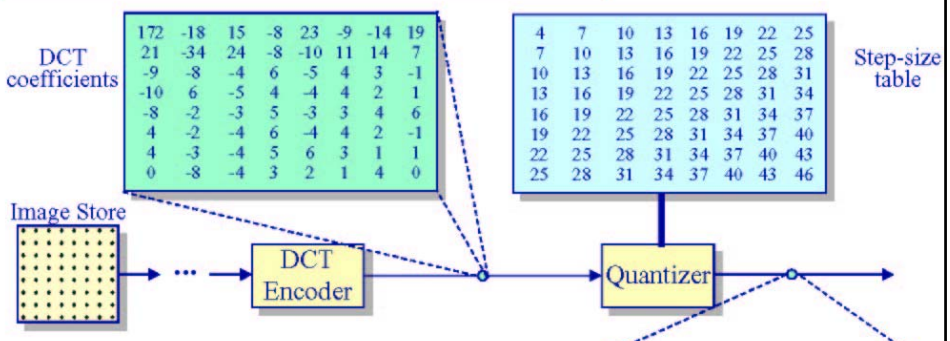
The two-dimensional DCT

- Apply the DCT in *x* and *y* dimensions simultaneously to 8x8 pixel blocks
  - » Code coefficients individually with fewer bits



JPEG Compression

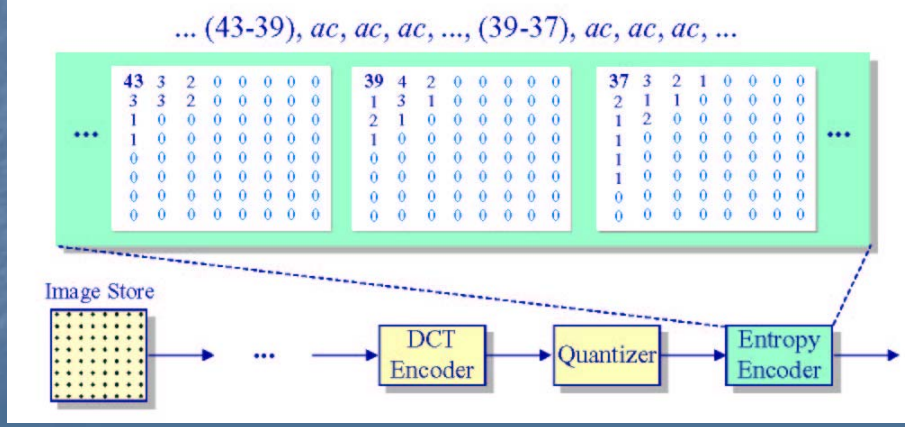
Quantization example



- Step-size table can be scaled to control degree of compression

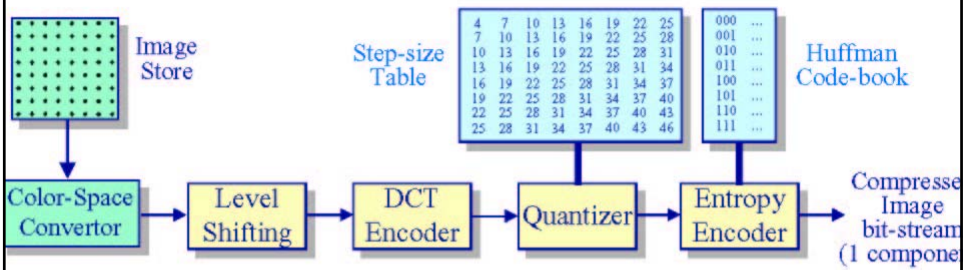
Humans are not sensitive to high frequencies

DC Coefficients DPCM



Sequential JPEG Compression Summary

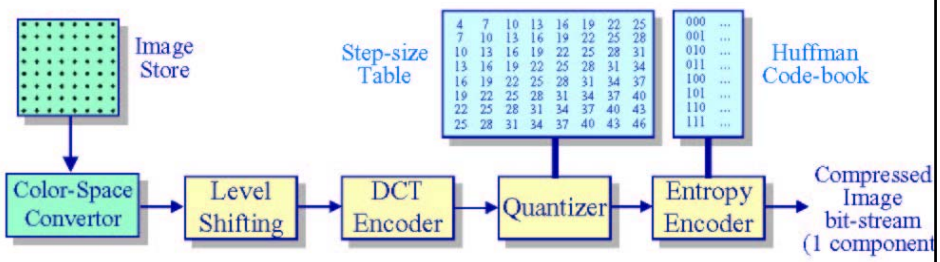
Complete compression pipeline



- Compression comes from:
  - » Chrominance subsampling
  - » DCT coefficient quantization
  - » Difference coding DC coefficients
  - » Statistical & run-length coding of AC coefficients
- Qualitative results:
  - » 0.25 - 0.5 bpp — ok for some applications
  - » 0.5 - 0.75 bpp — ok for many
  - » 0.75 - 1.5 bpp — excellent
  - » 1.5 - 2.0 — indistinguishable

Sequential JPEG Compression Summary

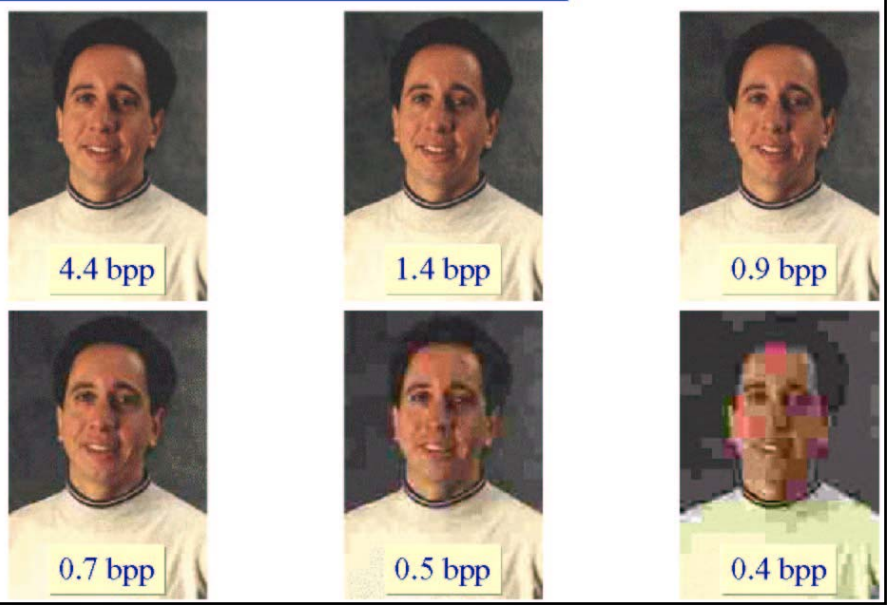
Complete compression pipeline



- Compression comes from:
  - » Chrominance subsampling
  - » DCT coefficient quantization
  - » Difference coding DC coefficients
  - » Statistical & run-length coding of AC coefficients
- Qualitative results:
  - » 0.25 - 0.5 bpp — ok for some applications
  - » 0.5 - 0.75 bpp — ok for many
  - » 0.75 - 1.5 bpp — excellent
  - » 1.5 - 2.0 — indistinguishable

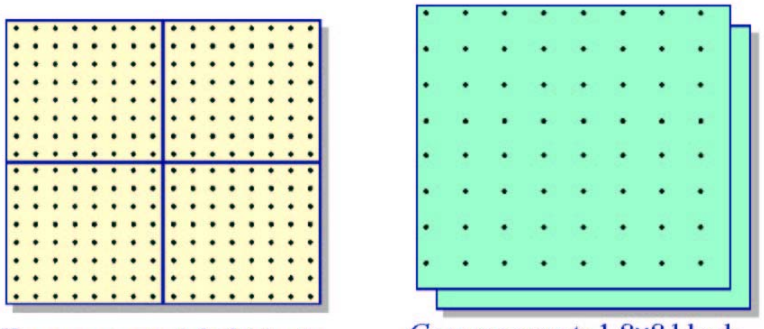
JPEG Compression

Examples of quality *v.* bpp



MPEG Video Compression

- Chrominance components are subsampled 2:1 horizontally & vertically
- Each video frame is subdivided into 16x16 *macroblocks*



Y component: 4 8x8 blocks

Cr component: 1 8x8 block
(same for *Cb*)

27

Pipeline