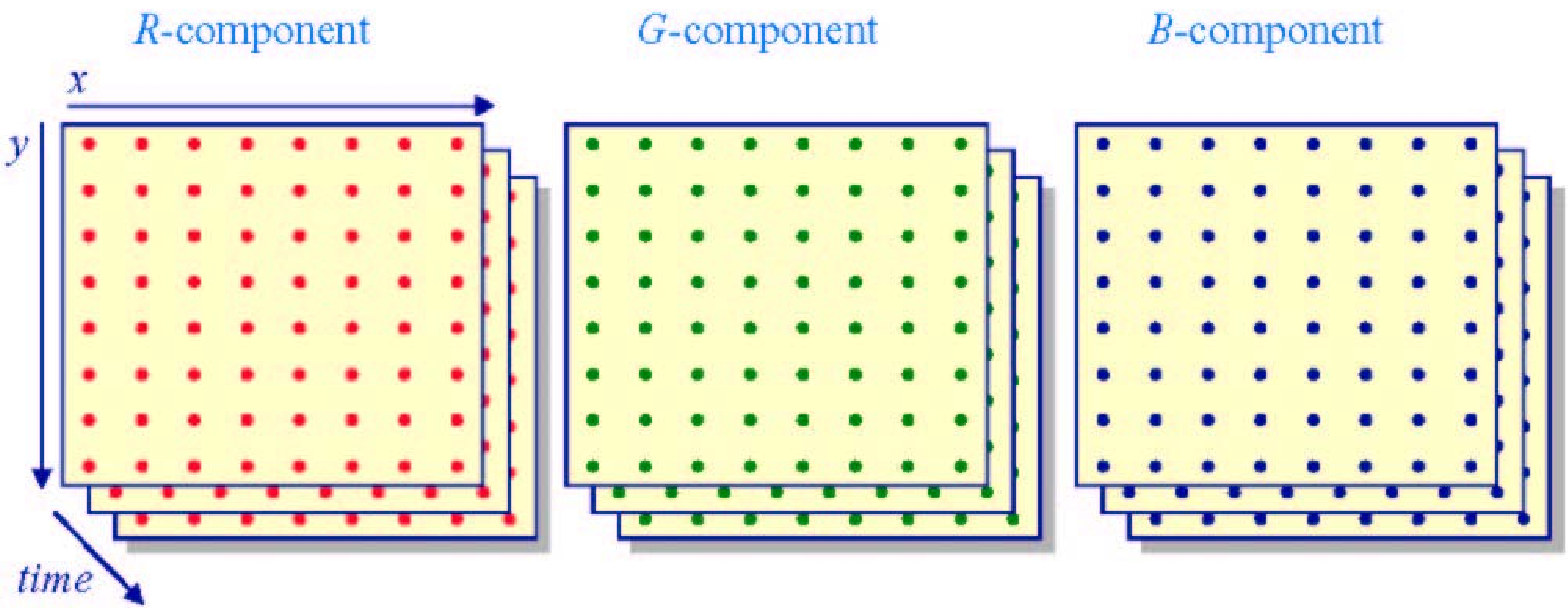
Image and Video Compression

Digital Image Processing

Image and Video

- Video is a multi-dimensional signal



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

» $Y = 0.30R + 0.59G + 0.11B$

» $I = 0.60R - 0.28G - 0.32B$

» $Q = 0.21R - 0.52G + 0.31B$

» $Y = 0.3R + 0.6G + 0.1B$

» $U = (B - Y) \times 0.493$

» $V = (R - Y) \times 0.877$

Compression Issues

- 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

Compression Basics

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

Compression Basics

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

Bit Reduction

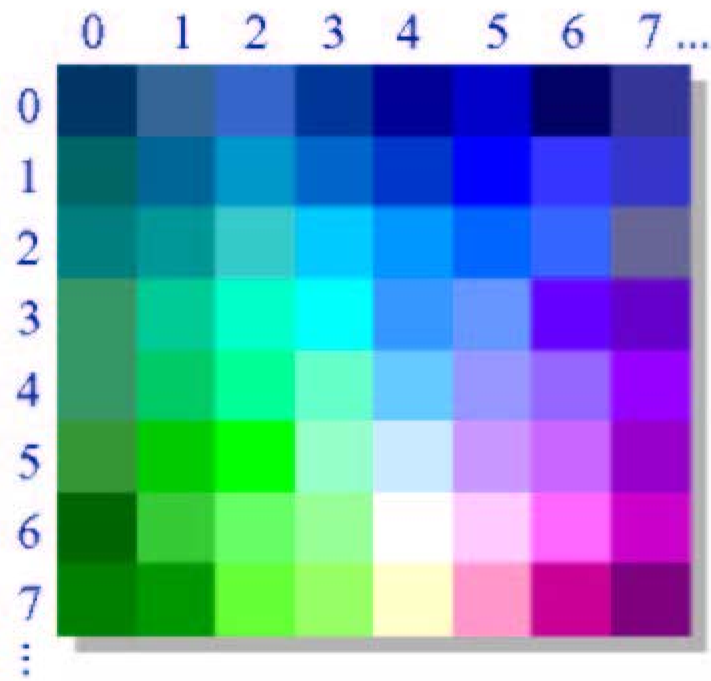
- Reducing the number of bits per pixel
  - » Throw away the least significant bits of each sample value
- Example
  - » Go from *RGB* at 8 bits/component sample (8:8:8) to 5 bits (5:5:5)
    - Go from 24 bpp to 15 bpp
    - This gives "acceptable results"
  - » Go from *YUV* at 8 bits/component sample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

Compression Basics

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

Color Look-Up-Table (Statistical)

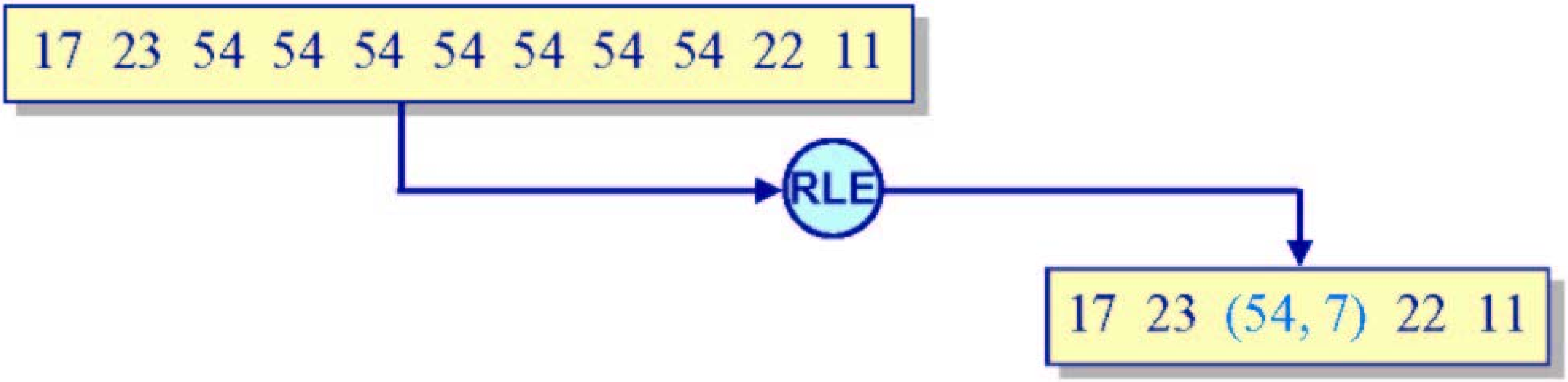
- Quantize coarser in the color domain
  - » Pixel values represent indices into a color table
  - » Tables can be optimized for individual images
- Entries in color table stored at "full resolution" (*e.g.* 24 bits)



- Example:
  - » 8-bit indices (256 colors) gives

$$(440 \times 480) \times 8 + (24 \times 256) = 1.7 \times 10^{6} \text{ bits/sec}$$

Run Length Encoding



- 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*)

Statistical Compression

Huffman coding

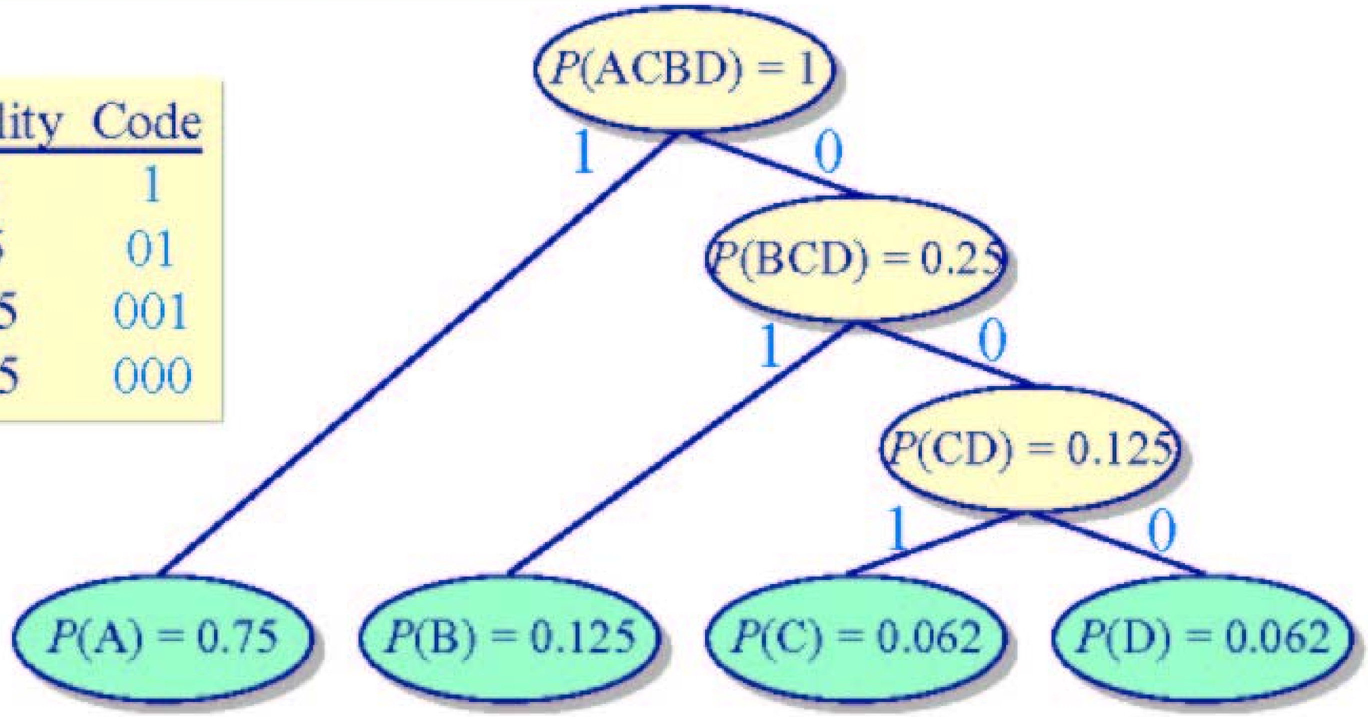
- Exploit the fact that not all sample values are equally likely
  - Samples values are non-uniformly distributed
  - Encode "common" values with fewer bits and less common values with more bits
- Process each image to determine the statistical distribution of sample values
  - Generate a *codebook* — 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



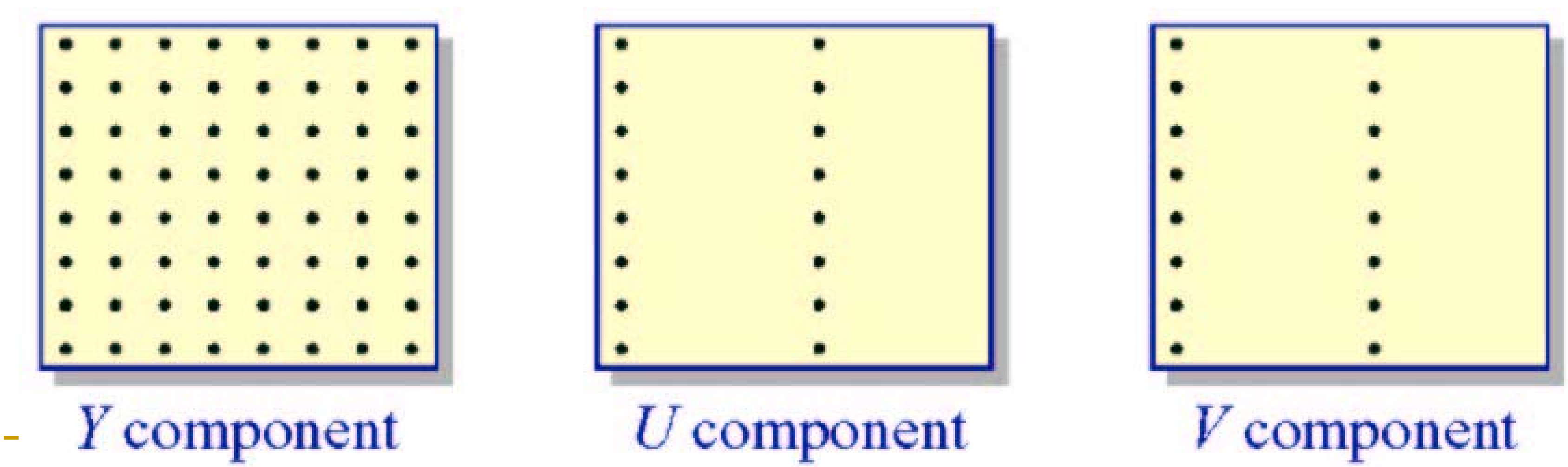
- Order all possible sample values in a binary tree by combining the least likely samples into a sub-tree
- Label the branches of the tree with 1's and 0's
  - » Huffman code is the sequence of 1's and 0's on the path from the root to the leaf node for the symbol

Compression Basics

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

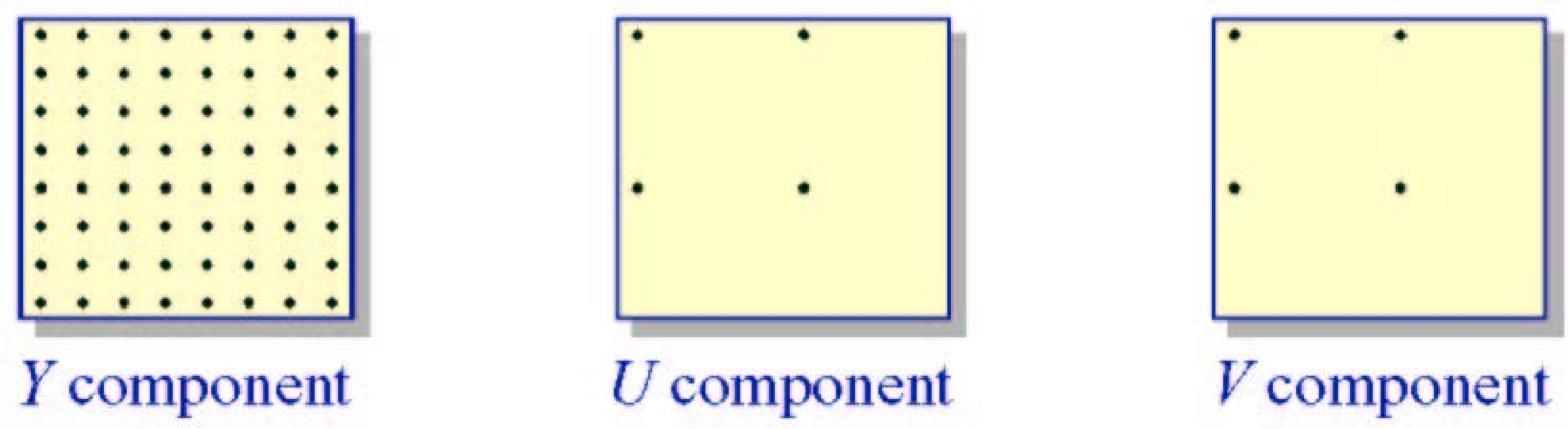
Interpolative Compression

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



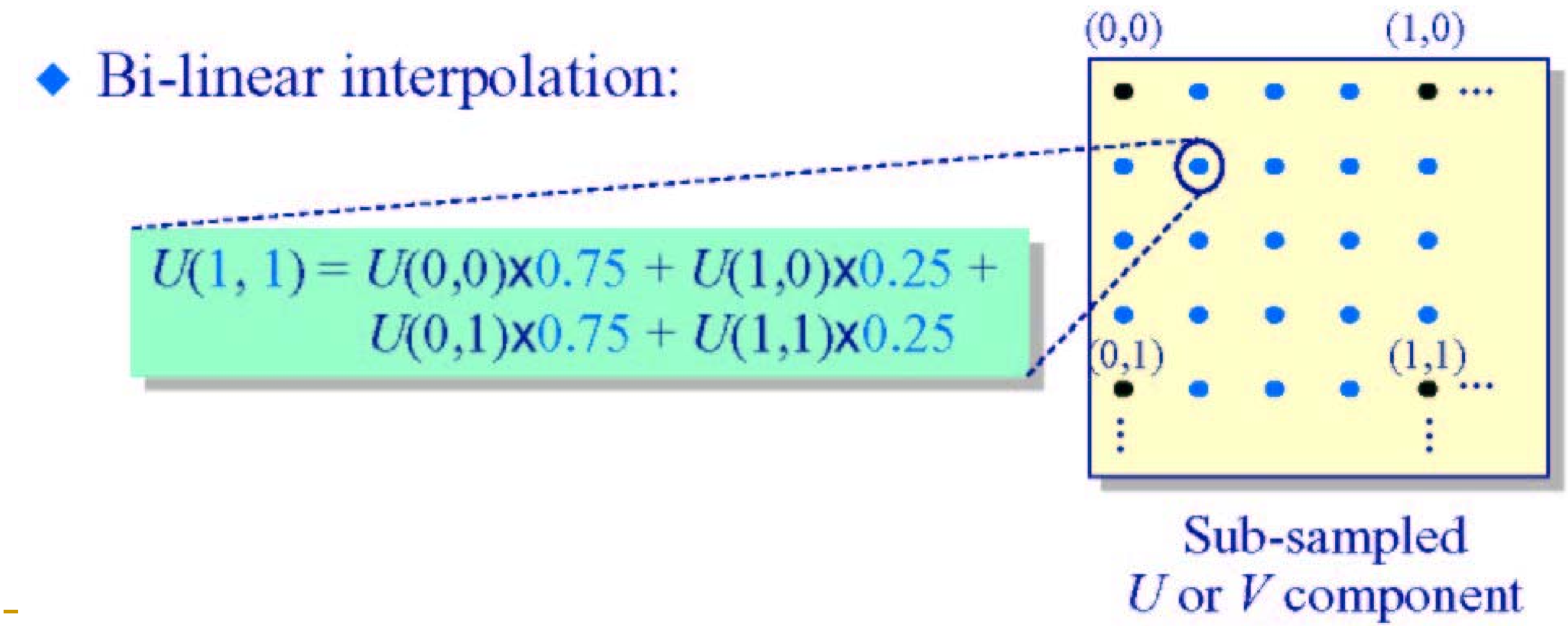
Interpolative Compression

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

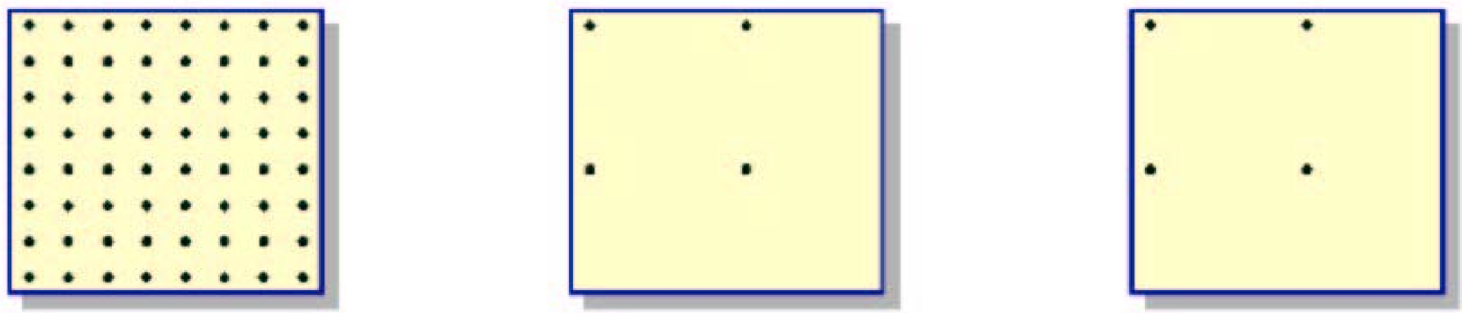


Reconstruction

- Using bilinear interpolation
- Gives excellent results



Significant Compression



Y component   *U* component   *V* component

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

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

$$= 9$$

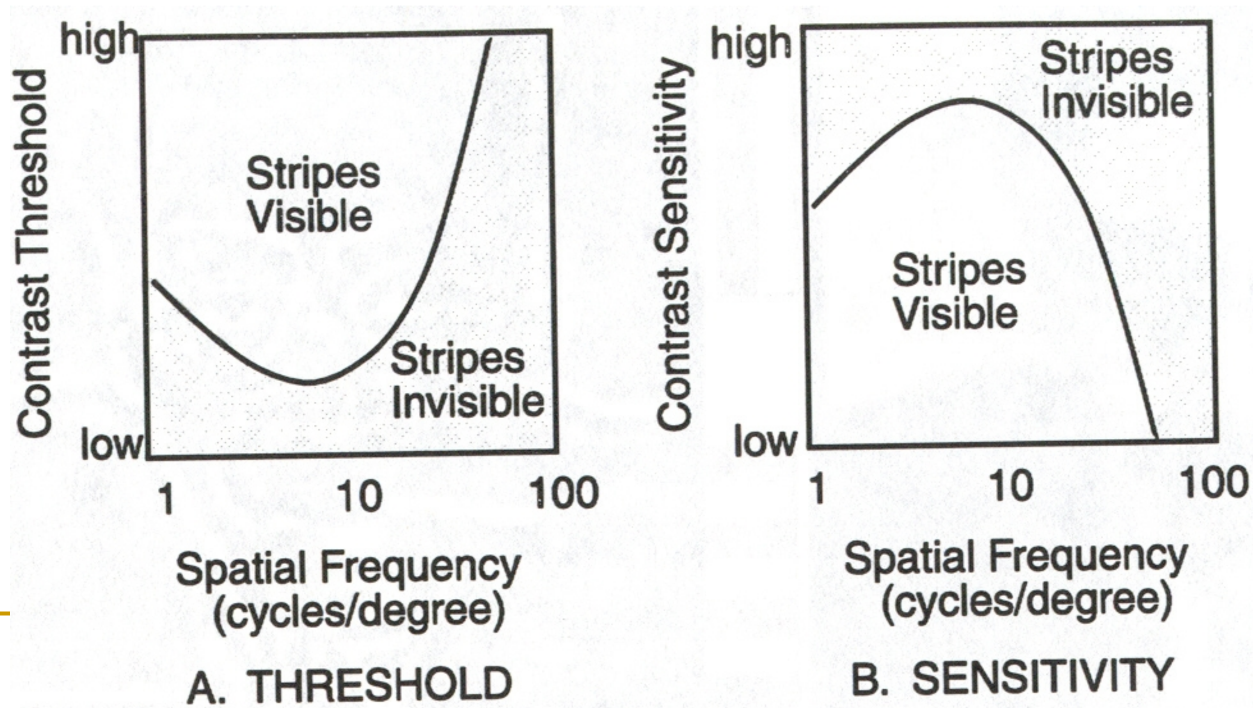
  - » A 62.5% reduction overall

Compression Basics

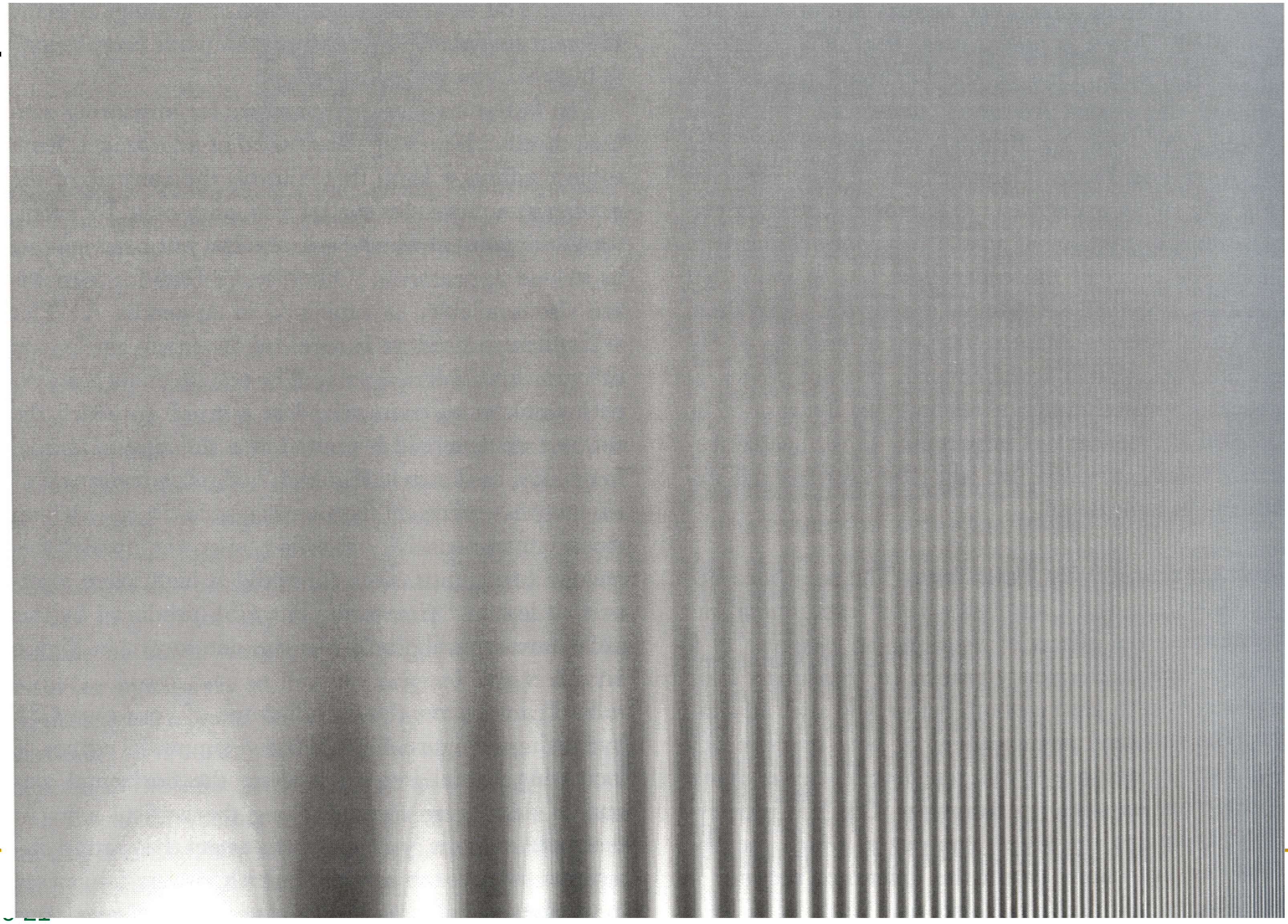
- Simple compression
- Statistical techniques
- Interpolation-based techniques
- Transforms

Luminance Contrast Sensitivity

- Minimum contrast required to *detect* a particular frequency
- Maximum sensitive at 4-5 cycles per degree

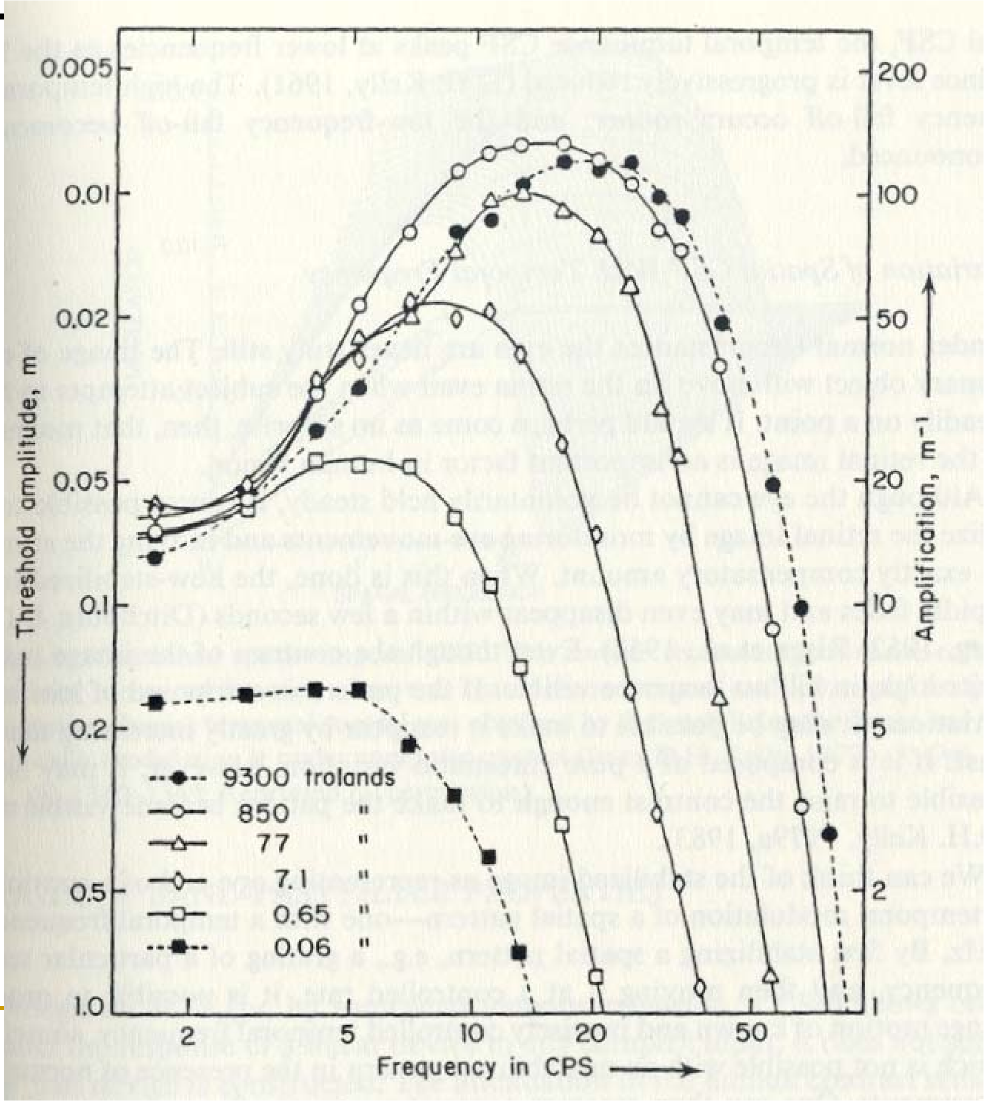


Testing Contrast Sensitivity



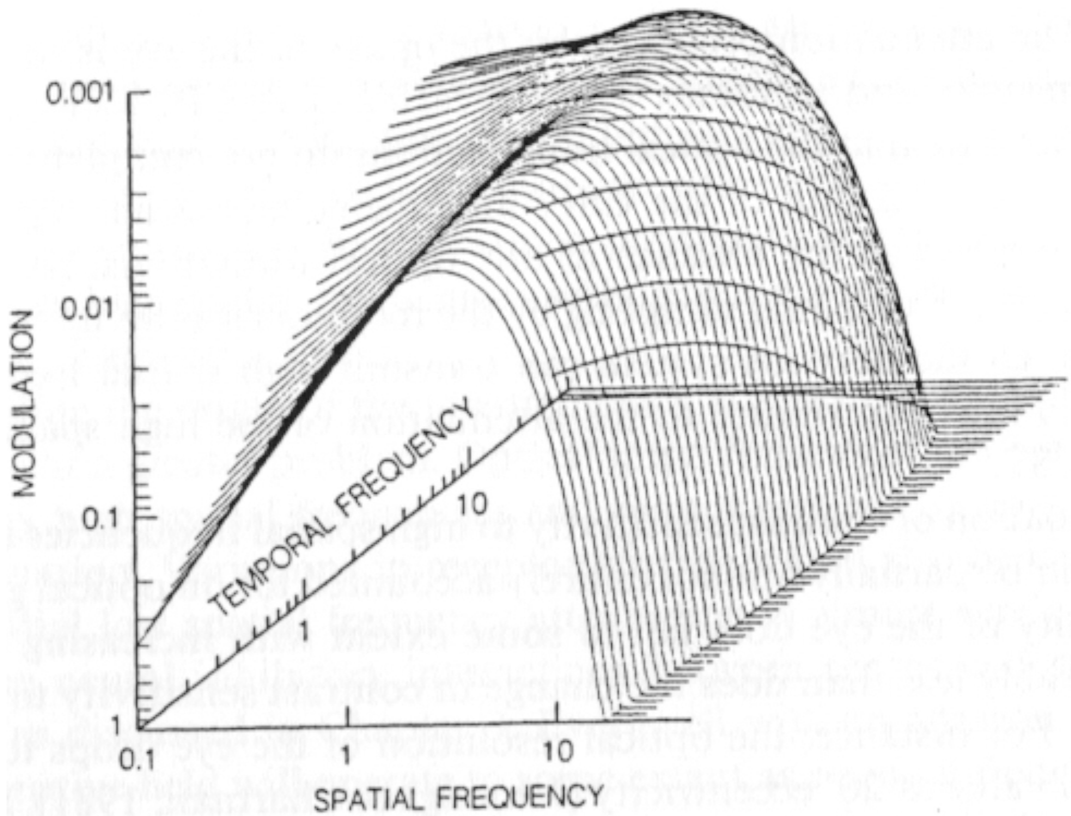
Temporal Contrast Sensitivity

- Present image of flat fields temporally varying in intensity like a sine wave
- If the flicker is detectable
- Cycles per second



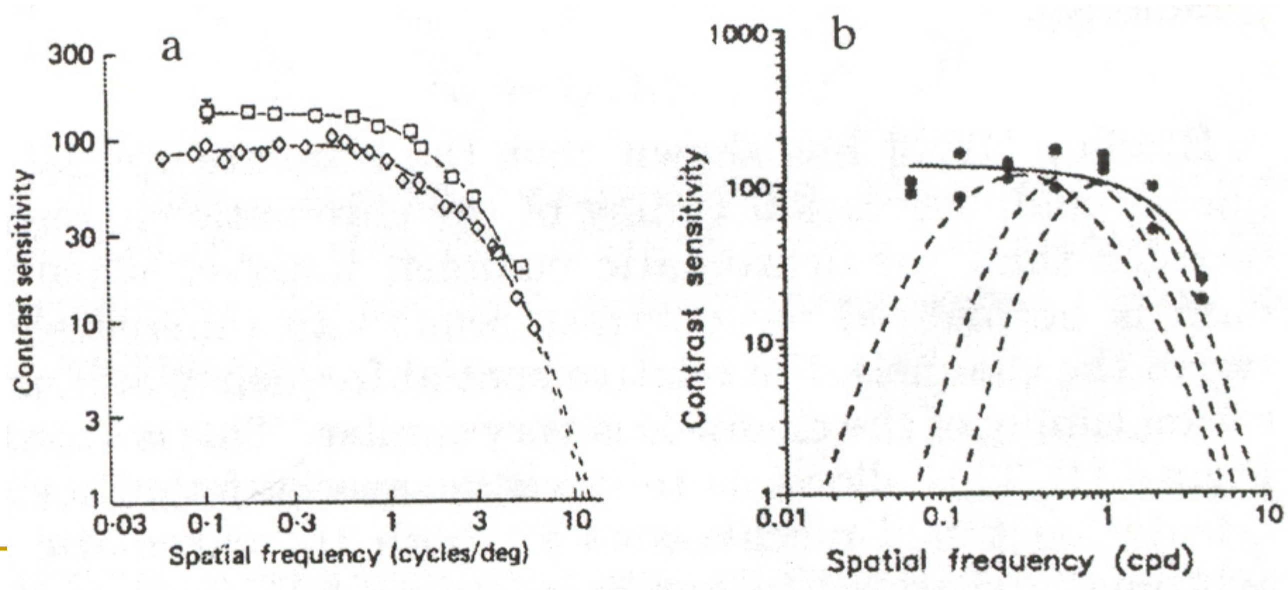
CSF and filters

- Both spatial and temporal CSF act as band pass filters
- How do they interact?
  - At higher temporal frequency, acts as low pass filter



Chrominance Contrast Sensitivity

- Gratings
  - Red-Green (602, 526nm)
  - Blue-Yellow (470, 577nm)



Compare with luminance CSF

- Low pass filter rather than bandpass filter
- Sensitivity is lower
  - More sensitive to luminance change than to chrominance change
- High frequency cut-off is 11 cycles per degree rather than 30 cycles per degree
  - Color acuity is lower than luminance acuity

Important points

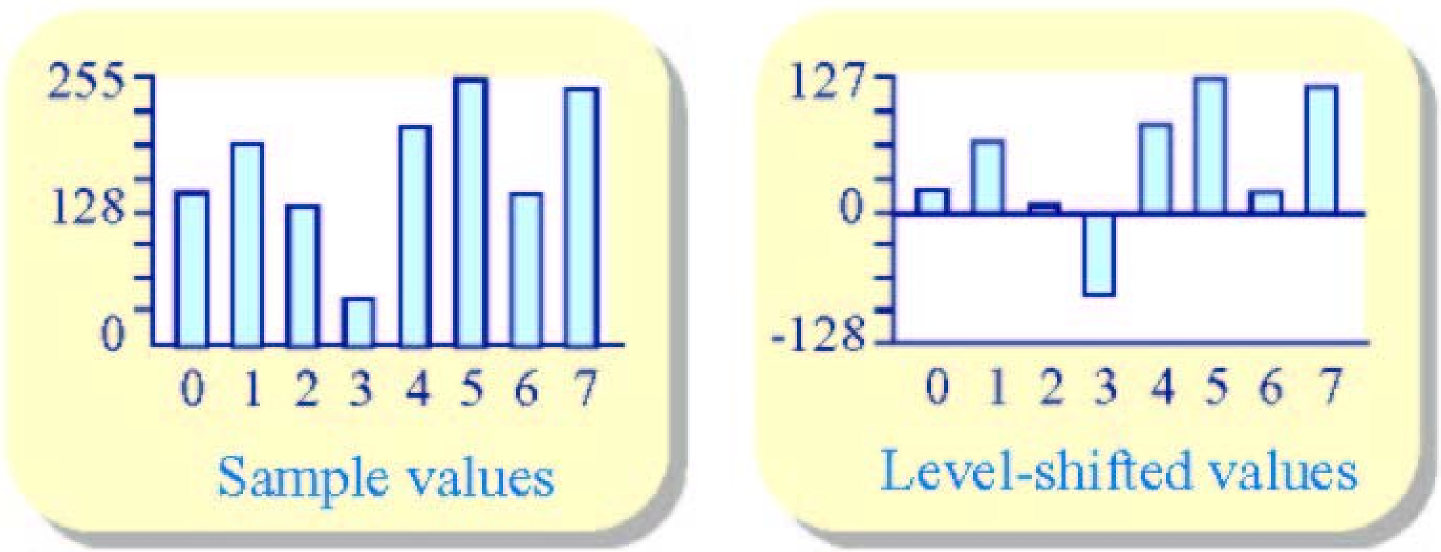
- We are more sensitive to lower frequencies than to higher frequencies in luminance
- We are less sensitive to chrominance than to luminance
- We are less sensitive to high temporal frequency

Transform-Based Compression

The Discrete Cosine Transform (DCT)

-

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

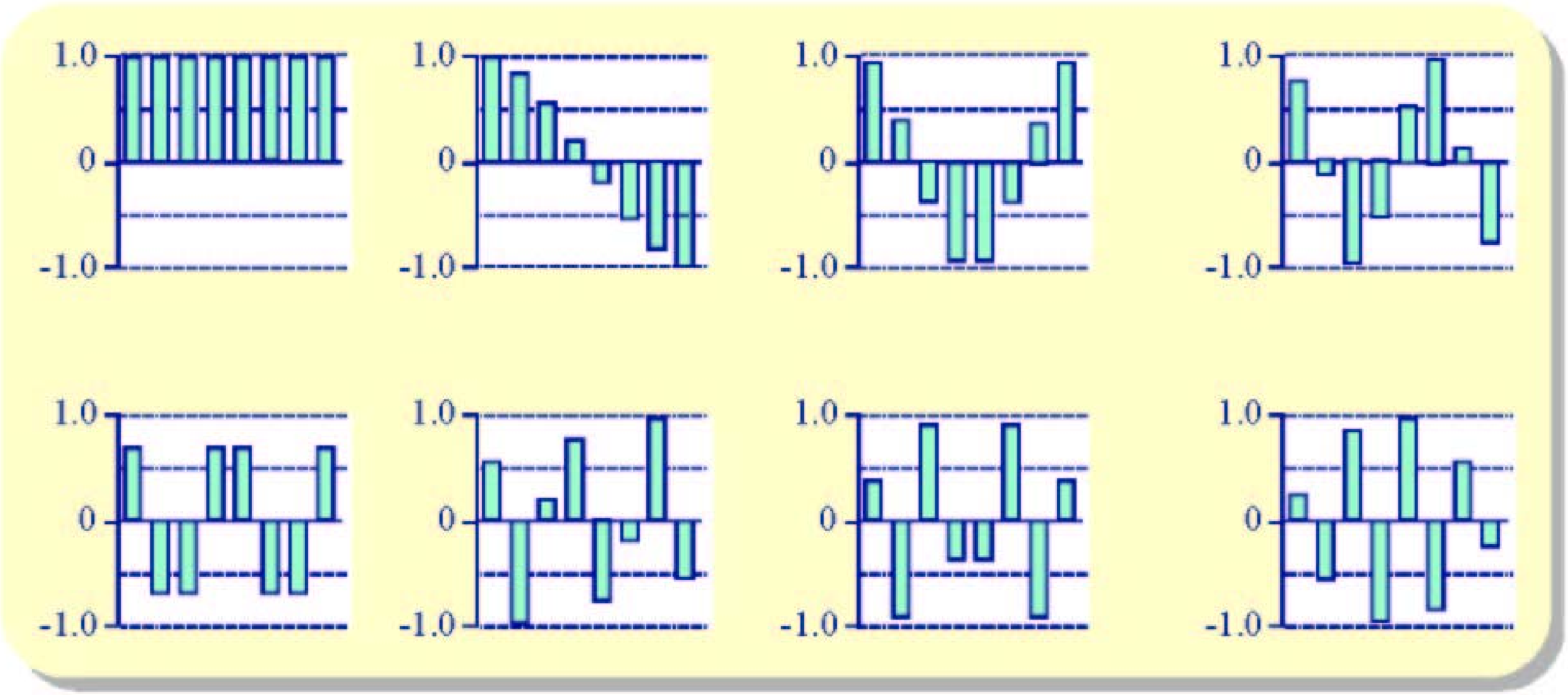


- What is the most compact way to represent this signal?

Transform-Based Compression

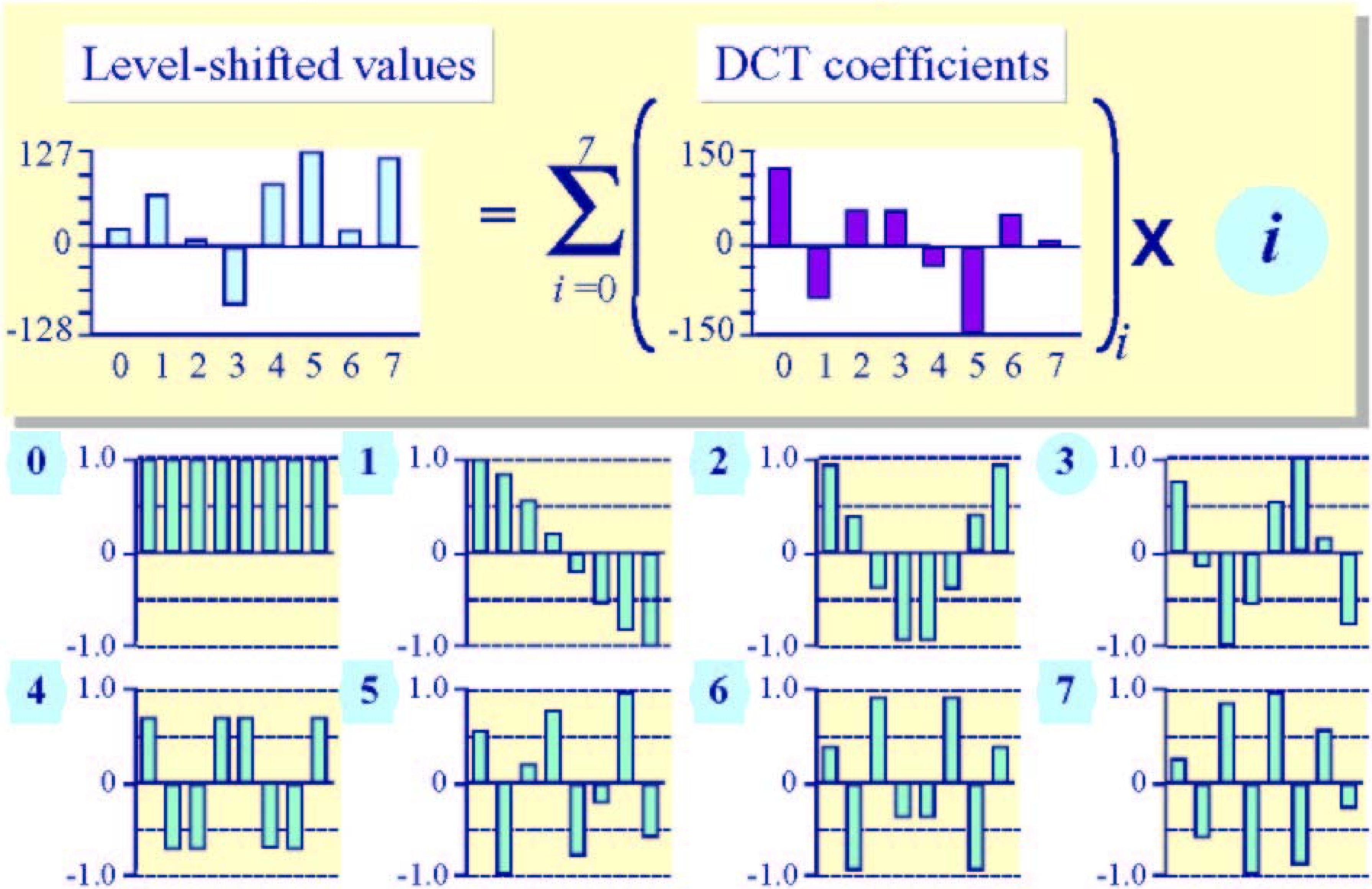
The Discrete Cosine Transform (DCT)

- 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



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)

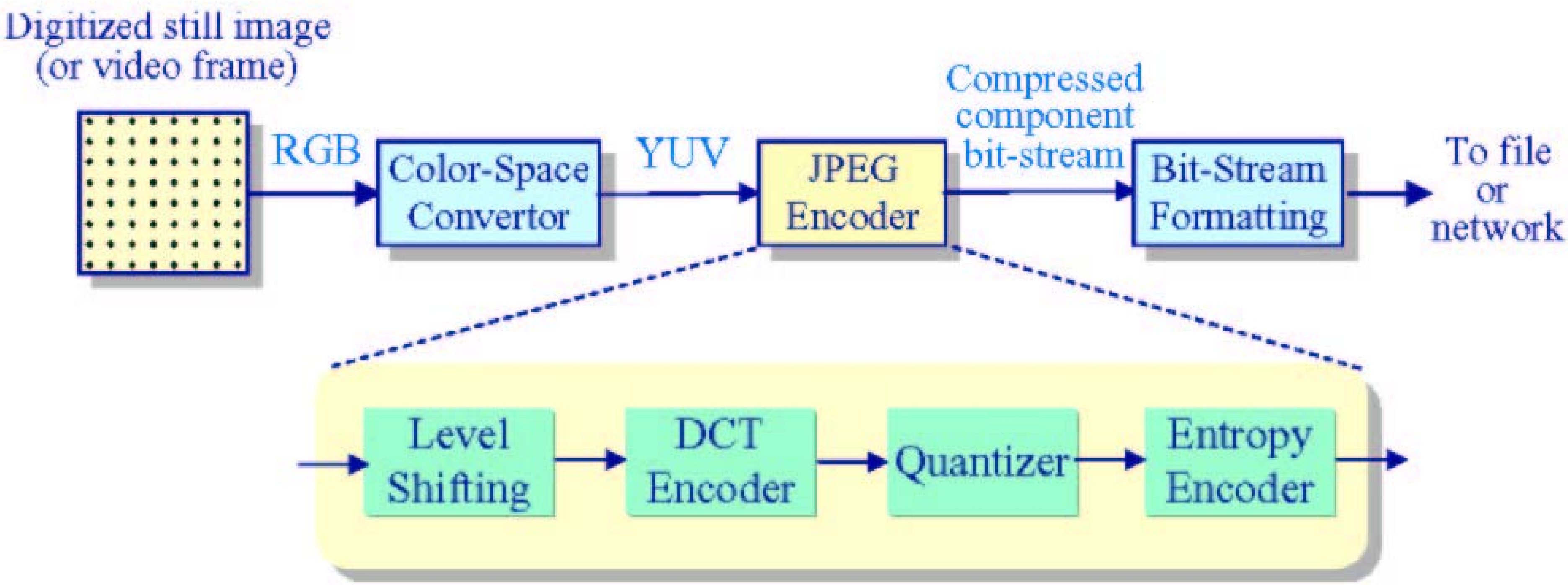
- The 2-dimensional (spatial) transform:

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

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

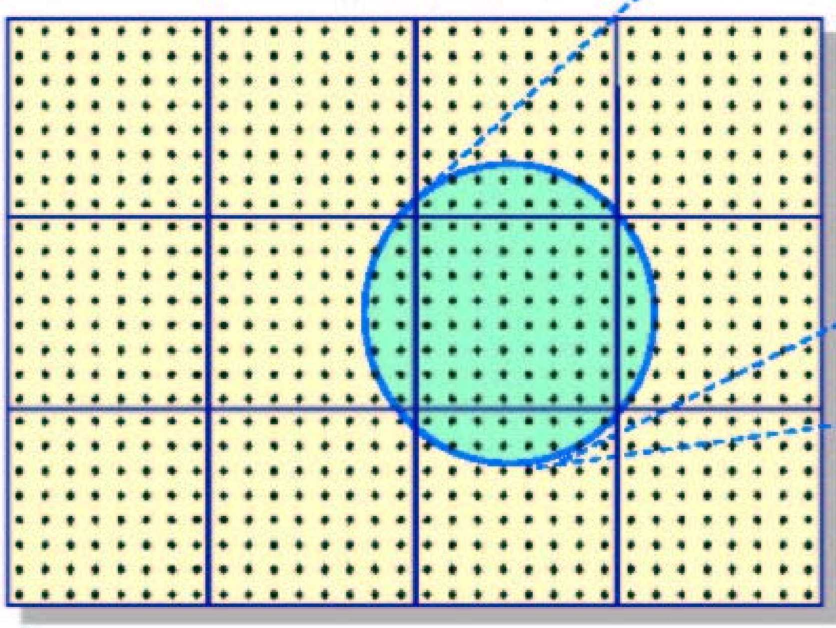


Transform-Based Compression

The two-dimensional DCT

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

Video Frame



172	-18	15	-8	23	-9	-14	19
21	-34	24	-8	-10	11	14	7
-9	-8	-4	6	-5	4	3	-1
-10	6	-5	4	-4	4	2	1
-8	-2	-3	5	-3	3	4	6
4	-2	-4	6	-4	4	2	-1
4	-3	-4	5	6	3	1	1
0	-8	-4	3	2	1	4	0

DCT Coefficients

JPEG Compression

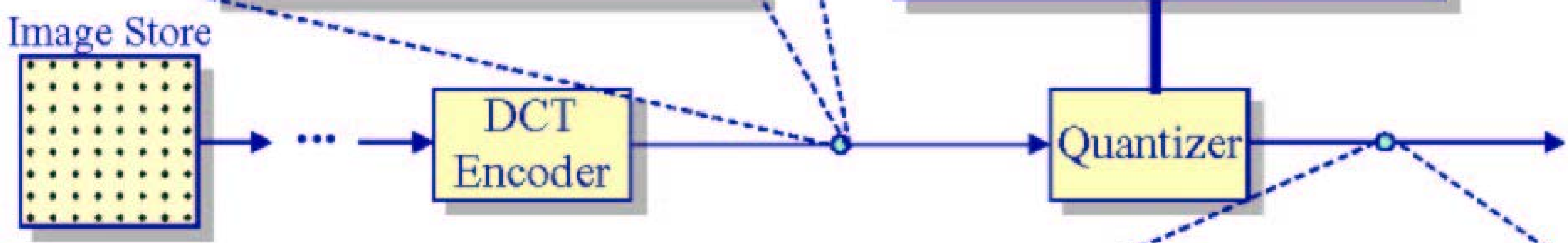
Quantization example

DCT coefficients

172	-18	15	-8	23	-9	-14	19
21	-34	24	-8	-10	11	14	7
-9	-8	-4	6	-5	4	3	-1
-10	6	-5	4	-4	4	2	1
-8	-2	-3	5	-3	3	4	6
4	-2	-4	6	-4	4	2	-1
4	-3	-4	5	6	3	1	1
0	-8	-4	3	2	1	4	0

Step-size table

4	7	10	13	16	19	22	25
7	10	13	16	19	22	25	28
10	13	16	19	22	25	28	31
13	16	19	22	25	28	31	34
16	19	22	25	28	31	34	37
19	22	25	28	31	34	37	40
22	25	28	31	34	37	40	43
25	28	31	34	37	40	43	46

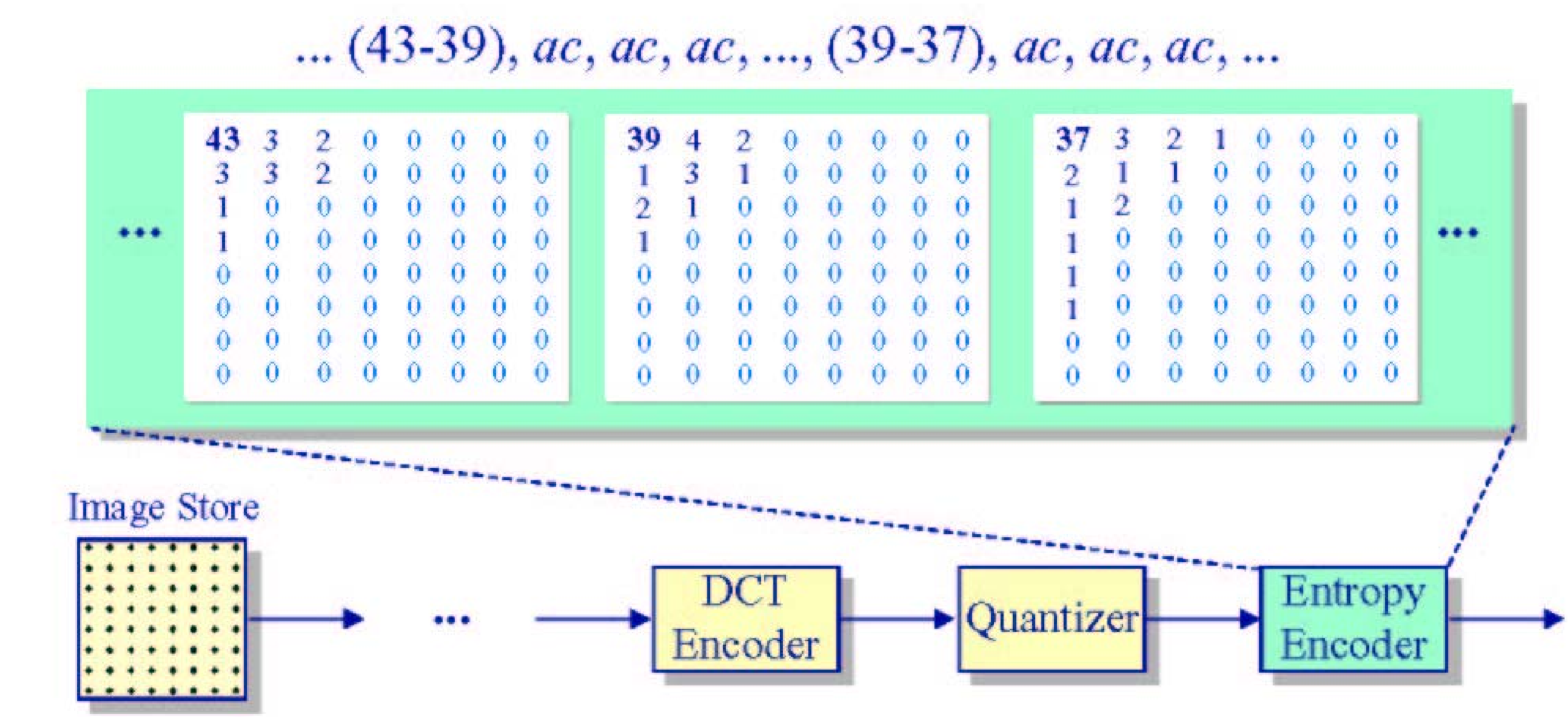


43	3	2	1	1	0	0	0
3	3	2	1	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

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

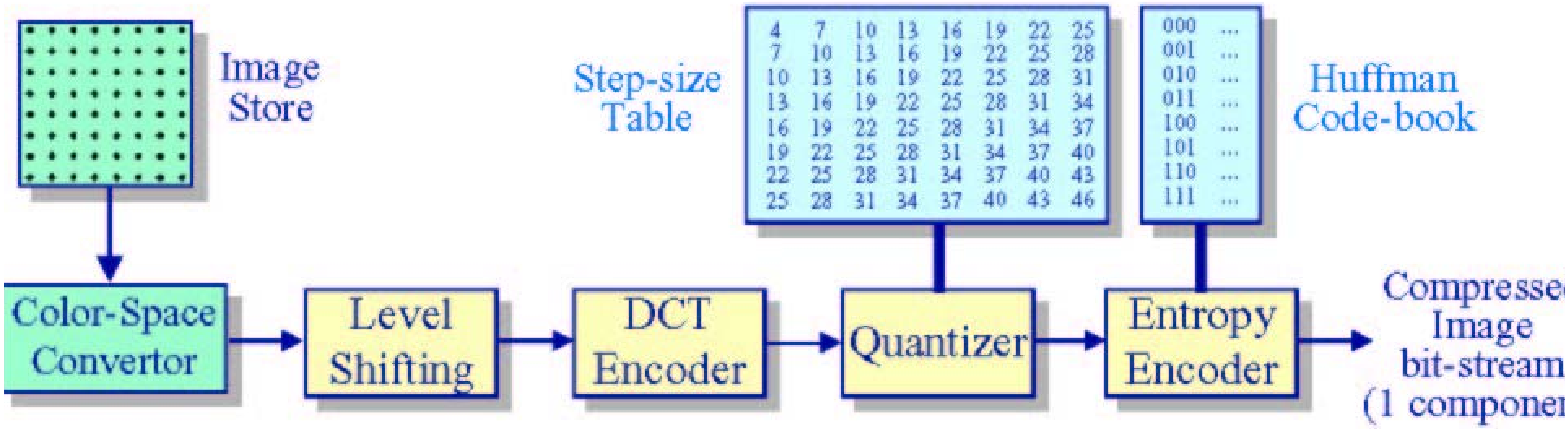
Humans are not sensitive to high frequencies

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

JPEG Compression

Examples of quality *v.* bpp