Chapter 6

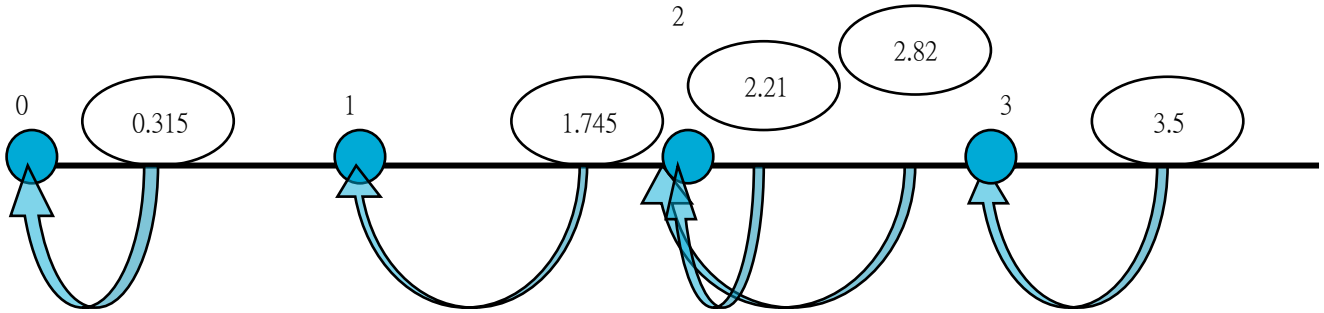
Data Compression: Source Coding

Isn't Entropy Coding Good Enough?

- In practice, loseless entropy coding such as Huffman, arithmetic and LZW coding usually reduce the data size by half (actual ratio depends on data properties)
- 2:1 ratio obviously is not good enough for multimedia data such as images and video.
- These coding techniques do not make use of the nature of the media and the inability of human perception.
- If loss is allowed, we can achieve even higher ratio with unnoticeable (to most people) artifact.
- The techniques discussed below may or may not be lossy.
- Let's call them source coding.

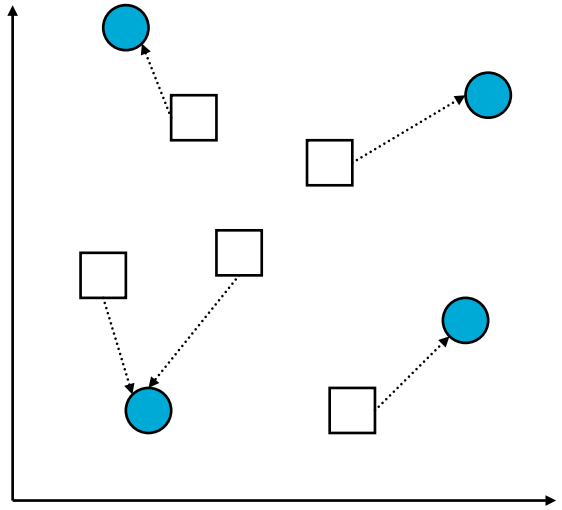
Vector Quantization (VQ)

- Scalar quantization
  - e.g. A/D conversion in audio and video sampling.
  - e.g. 0.315 -> 0, 1.745 -> 1, etc.

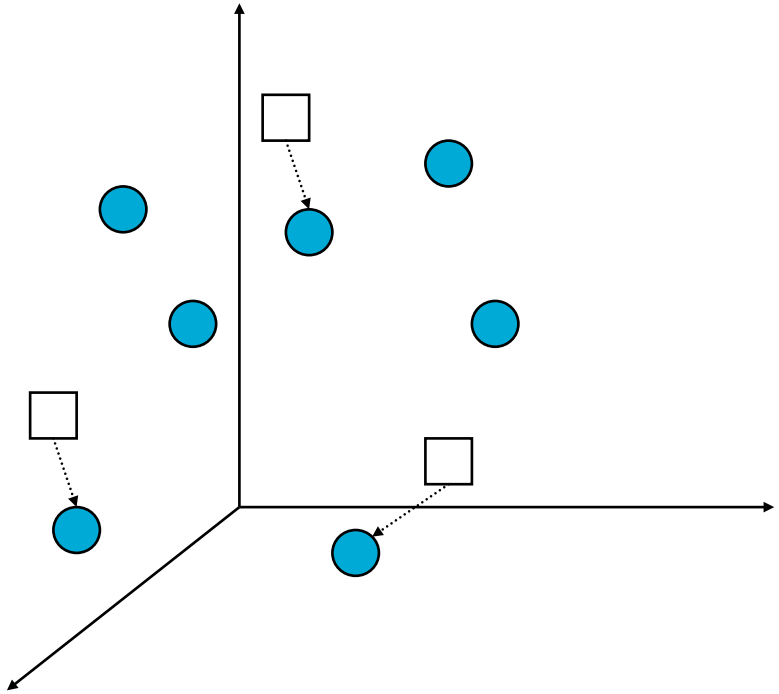


VQ (2)

- Generalized to 2D data space.

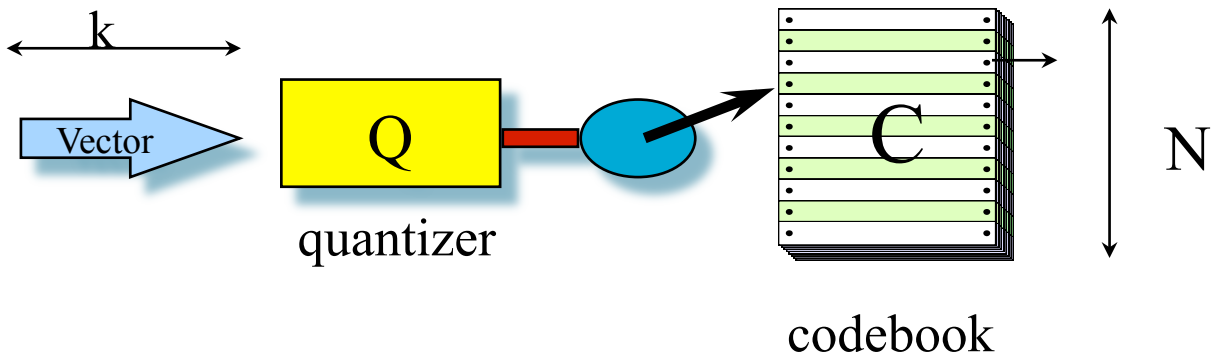


- Generalize to 3D data space.



VQ (3)

- VQ - subject to be quantized is vector.
- VQ as a form of pattern recognition
  - Input pattern (a vector) is approximated by one of the predetermined set of standard patterns.
- Lossy - Information lose as input vector is mapped to the standard vector.
- Process summarized as follows:



Quantization Error

- Quantization
  - Mapping all data points within a group onto the representative point is called quantization.
- Error term
  - x is the data point, $\hat{x}$ is the group centroid (representative point),
  - k is the number of groups.
  - *Squared error distortion* measures the goodness of quantization mapping:

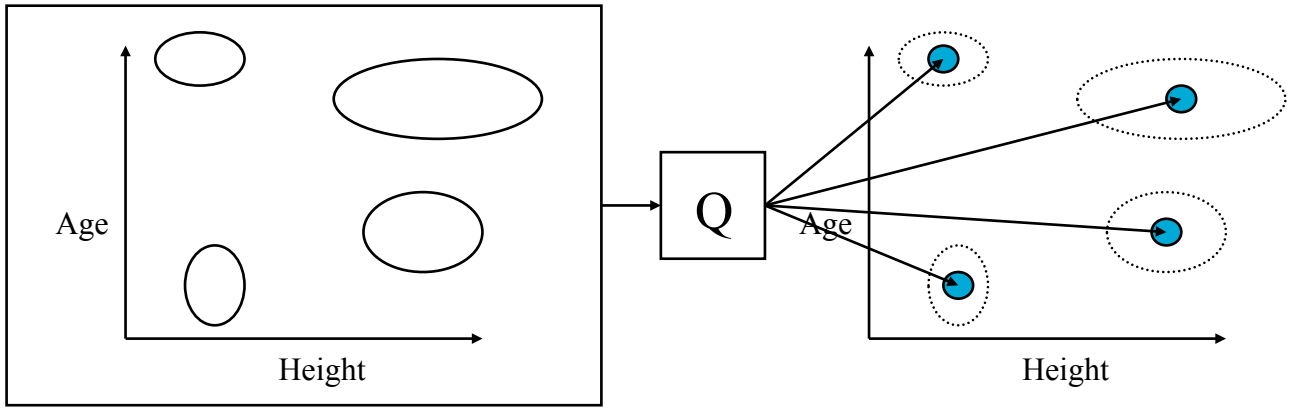
$$d(x,\hat{x}) = \|x - \hat{x}\|^2 = \sum_{i=0}^{k-1} (x_i - \hat{x}_i)^2$$

VQ Example 1

- Age/height group quantization
  - Population $\boldsymbol{S}$ { (a_1,h_1), (a_2,h_2), (a_3,h_3), …, (a_n,h_n) }
  - Quantized values $\boldsymbol{G}$ { $g_1(a^*_1,h^*_1)$, $g_2(a^*_2,h^*_2)$, $g_3(a^*_3,h^*_3)$, $g_4(a^*_4,h^*_4)$ }
  - Quantizer $\boldsymbol{Q}$ maps $\boldsymbol{S}$ onto $\boldsymbol{G}$ such that distortion measure is minimized.
    - i.e. For a data point $s_i(a_i,h_i)$, $\boldsymbol{Q}$ maps it to g_j so that

$$| s_i - g_j | = \min_k \{ | s_i - g_k | \}$$

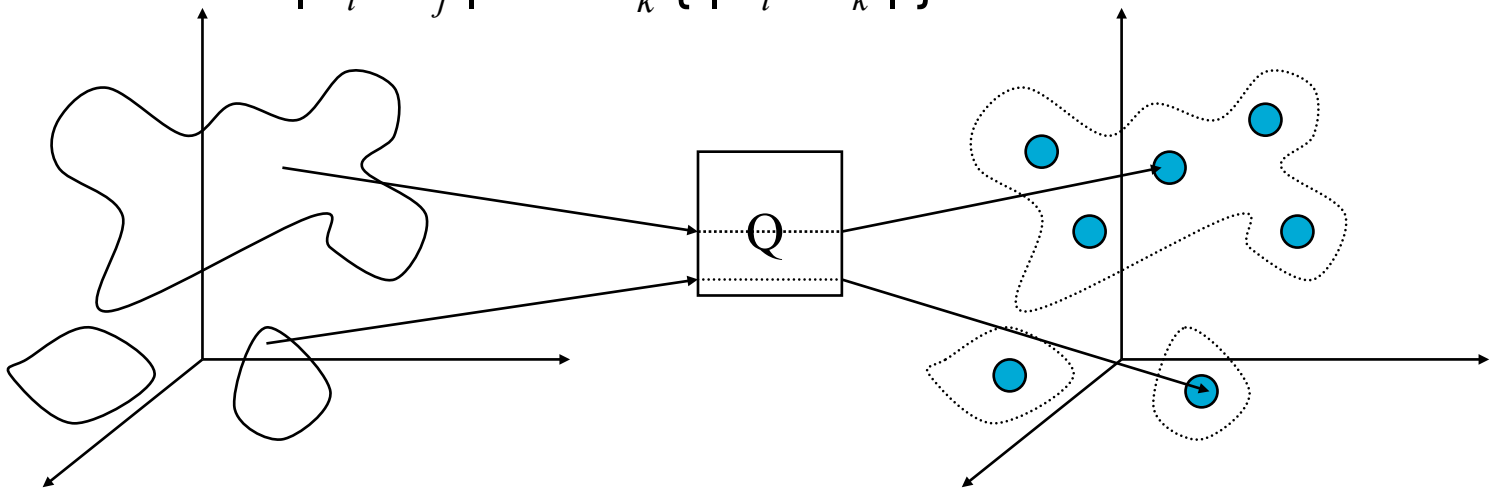
In other words, "map to the closest representative"



VQ Example 2

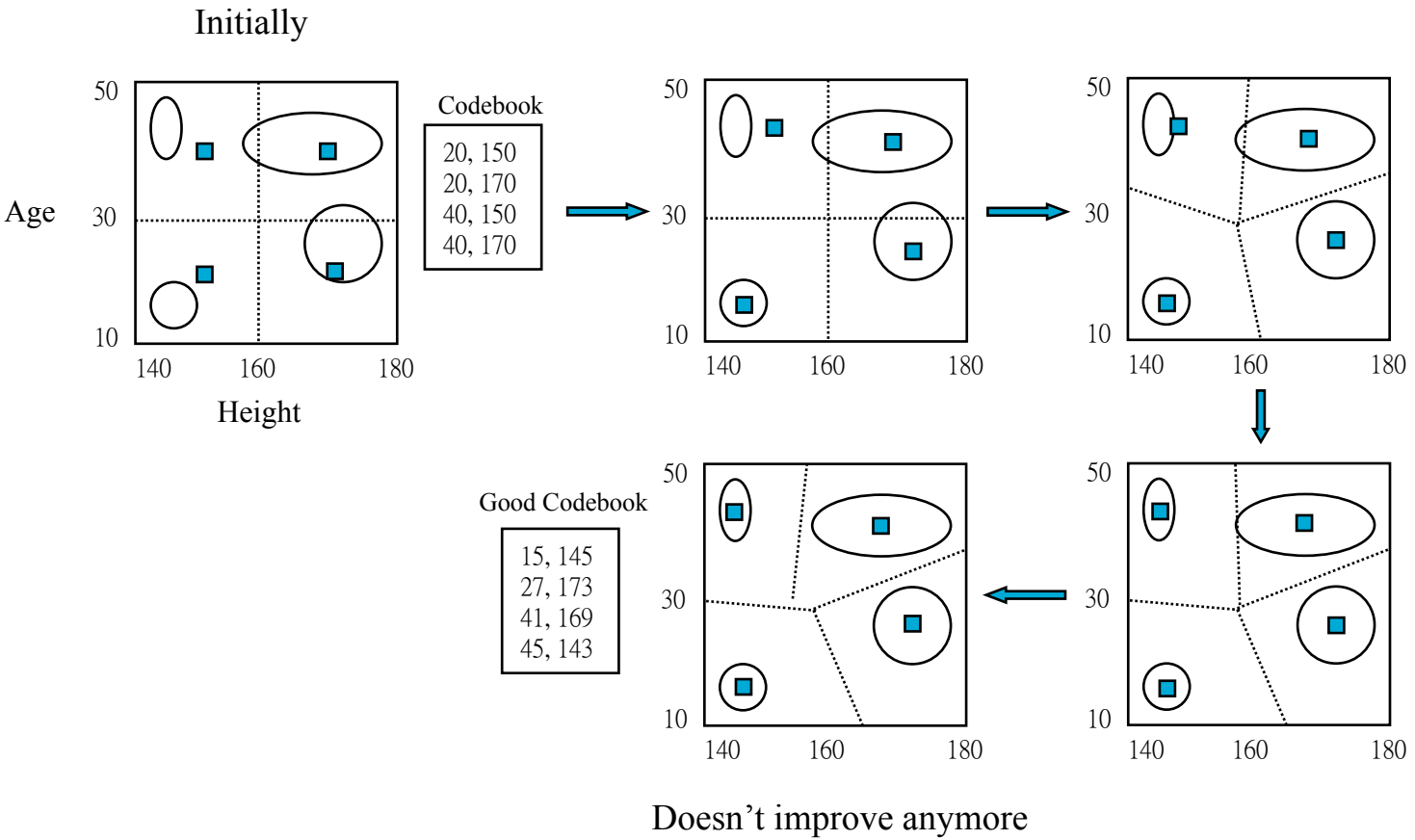
- Color quantization (VQ in color domain)
  - Consider RGB image, each pixel has a triplet (r,g,b).
  - Image to be quantized to carry 256 colors only.
  - $\boldsymbol{S}$ { $(r_1, g_1, b_1), (r_2, g_2, b_2)$, …, (r_n, g_n, b_n) }
  - Quantized $\boldsymbol{H}$ {(r^*_1, g^*_1, b^*_1), …, $(r^*_{256}, g^*_{256}, b^*_{256})$ }
  - Quantizer $\boldsymbol{Q}$ maps $\boldsymbol{S}$ onto $\boldsymbol{G}$ such that distortion measure is min.
    - i.e. For a data point $s_i(r_i,g_i,b_i)$, $\boldsymbol{Q}$ maps it to h_j so that

$$| s_i - h_j | = \min_k \{ | s_i - h_k | \}$$



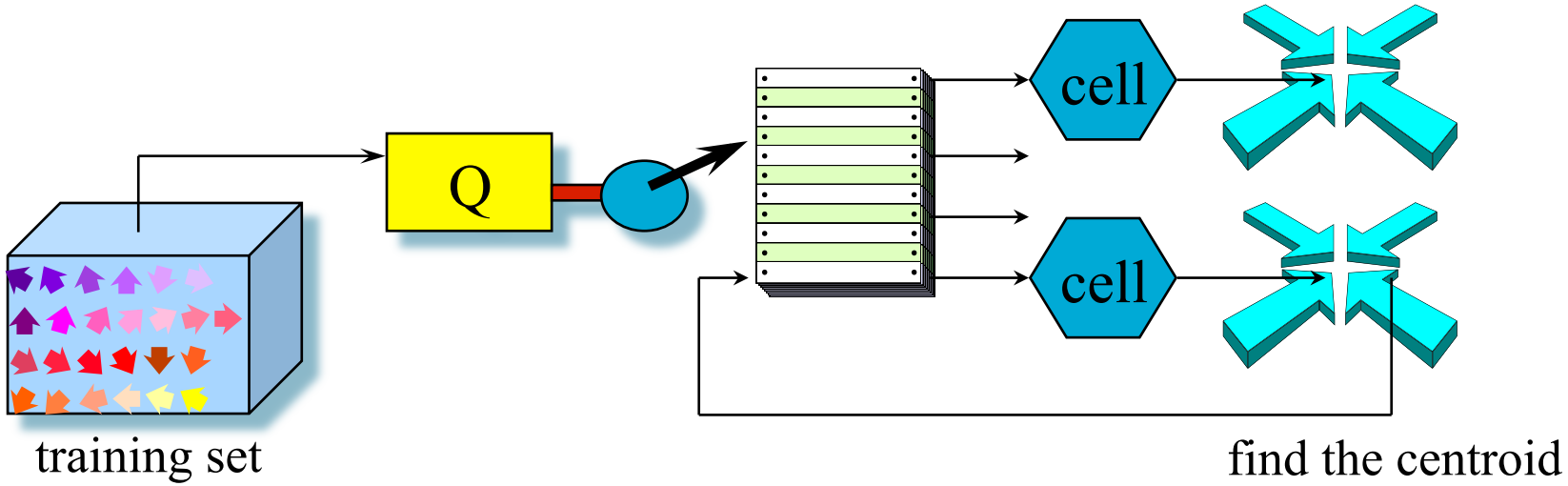
How to derive a good Codebook?

- Codebook - the table of all representatives
- LBG algorithm (similar to K-means method in data clustering)



Codebook Training

- Get a large sample of data (the training set).
- Pick an initial set of code vectors.
- Partition the training set into cells.
- Use the cells to tune the codebook.
- Repeat.
- The LBG (Linde-Buzo-Gray) Algorithm:



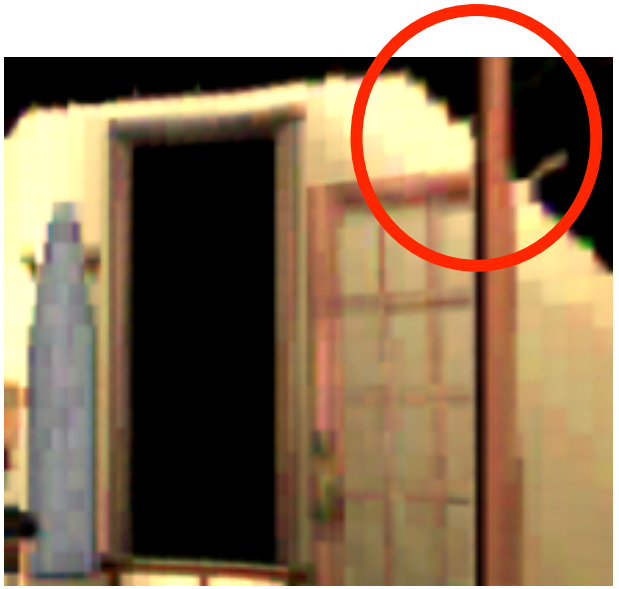
LBG Algorithm

- Step 1
  - Given a training set, X, with M vectors
  - Let d = the mean square distortion measure
  - Let the iteration index be j and set j=1
  - Select an initial codebook $\mathbf{Y}0$
  - Set initial distortion d0 = infinity
  - Pick a convergence threshold E.
- Step 2
  - Optimally encode all x within X using codebook $\mathbf{Y}$(j-1)
  - Assign x to cell P(i,j-1) if x is quantized as codevector y(i,j-1) where i is the i-th cell or i-th codevector
  - Compute d(j) = (sum((x-y)^2))/M
  - If (d(j-1)-d(j))/d(j) < E then quit with codebook = $\mathbf{Y}$(j-1), otherwise goto step 3.
- Step 3
  - Update the codevectors as
    y(i,j) = sum(x) for all x in P(i,j-1) cell / (number of x in P(i,j-1))
    i.e., the centroid of the cell.
  - j++; go to Step 2.

VQ Example 3

- VQ on image
- Subdivide the image into blocks e.g. 4 x 4, then it is treated as a vector with 16 elements
- Why?
- So you can make use of the coherence between adjacent pixels
- Feed the vectors (blocks) to VQ

Original image



VQ encoded image

Pros and Cons of VQ

Advantage:

- Extremely fast to decode. Just lookup the table (codebook)
- Easy to implement

Disadvantages:

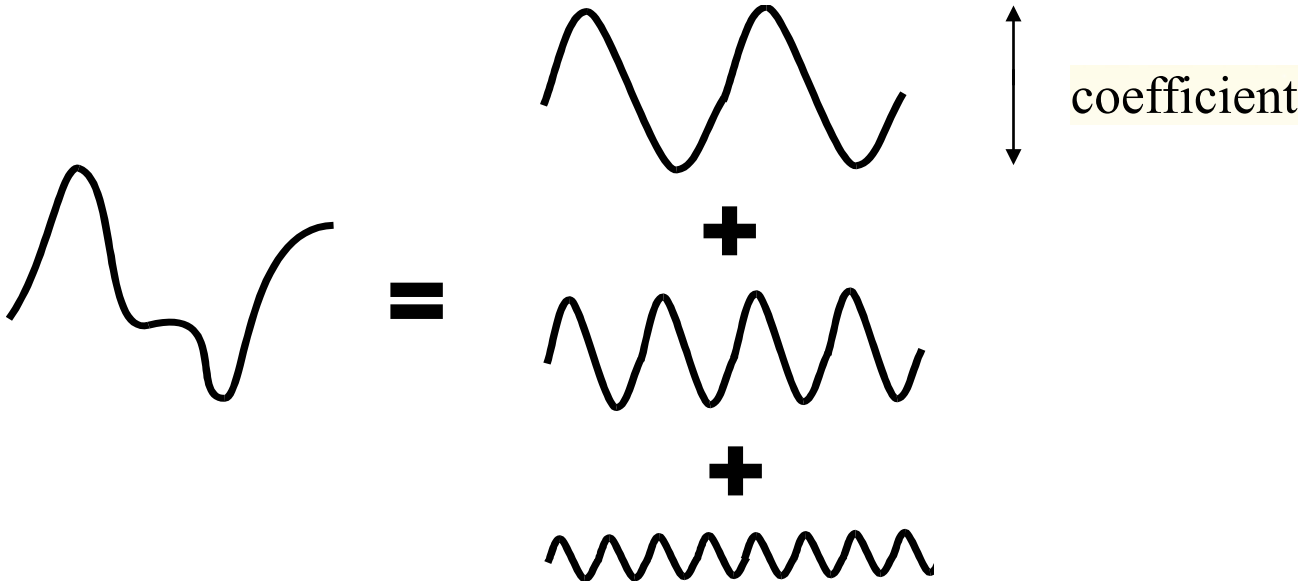
- When applied to image, there is blocky artifact.
- But the most seriously problem is its computational expensiveness.
- The running time grows exponentially when the codebook size increases.

Transform Coding

- Up to this moment, we compress all values/signal in spatial/temporal domain.
  Spatial - e.g. 2D image as 2D array of RGB values (samples along the space dimension)
  Temporal - e.g. 1D audio as sequence of samples (samples along the time axis)
- Is spatial/temporal domain the best place to compress?
- Not necessary

Transform Coding (2)

- From signal processing, we know that a signal can be represented as a sum of sinusoidal signal components.



- Each component has different frequency and can be mathematically represented. (basis functions)
- The amplitude of each frequency can be controlled by a coefficient. (weight)

Transform Coding (3)

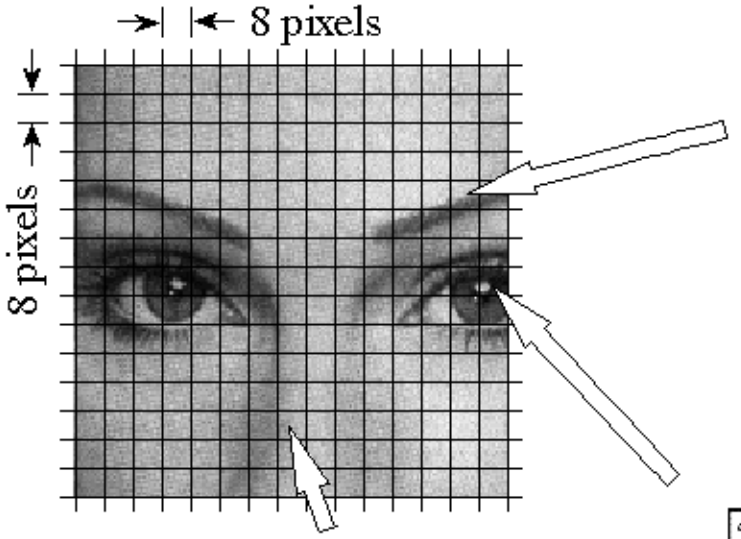
- In other words, the same data (signal) can simply be represented as a set of coefficients (or weights) and the basis functions.
- Once we all agreed to use one set of basis functions, there is no need to explicitly stored the basis functions.
- Only the weights are stored.
- The conversion from spatial samples to these weights (with respect to certain basis functions) is called transform. There are many different transforms.
- One well-known transform is Fourier transform.
- There is no loss in Fourier transform.
- And there is no reduction in storage if only transform is done.

Discrete Cosine Transform (DCT)

- Fourier transform uses both sine and cosine waves as its basis functions.
- In fact, Fourier transform is seldom used in image compression.
- Another transform, discrete cosine transform, is widely used in compressing visual media due to its capability in “squeezing energy” (explained later)
- It uses only cosine waves.

DCT(2)

- You can perform DCT on the whole image, but it is time consuming.
- An 8x8 image block is a 2D function f(x,y) in spatial domain (or time domain as a term borrowed from signal processing).



231	224	224	217	217	203	189	196
210	217	203	189	203	224	217	224
196	217	210	224	203	203	196	189
210	203	196	203	182	203	182	189
203	224	203	217	196	175	154	140
182	189	168	161	154	126	119	112
175	154	126	105	140	105	119	84
154	98	105	98	105	63	112	84

154	154	175	182	189	168	217	175
154	147	168	154	168	168	196	175
175	154	203	175	189	182	196	182
175	168	168	168	140	175	168	203
133	168	154	196	175	189	203	154
168	161	161	168	154	154	189	189
147	161	175	182	189	175	217	175
175	175	203	175	189	175	175	182

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

DCT (4)

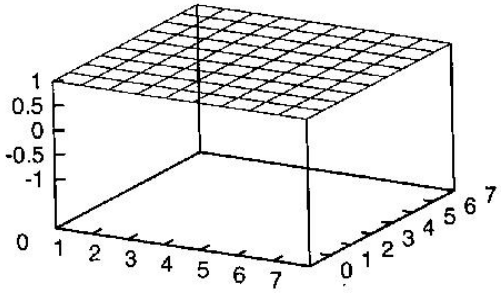
- Define basis functions. For frequency variables u, v in the 2 dimensions respectively:

$$b_{u,v}(x,y) = \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

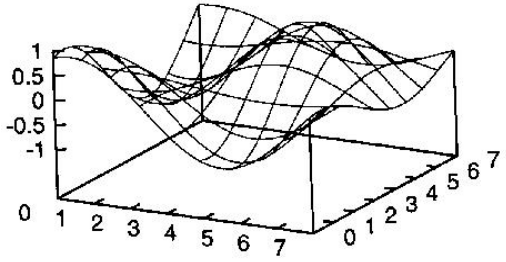
- These are wave functions of successively increasing frequencies. *(Imagine them as undulating surfaces of increasingly frequent ups and downs.)*
- Given a 2D function *(imagine it as a 2D surface)*, one can decompose it into a linear combination of these wave functions.
- So, DCT is a frequency (uv coordinates) representation of a spatial (xy coordinates) function.

DCT Basis Functions

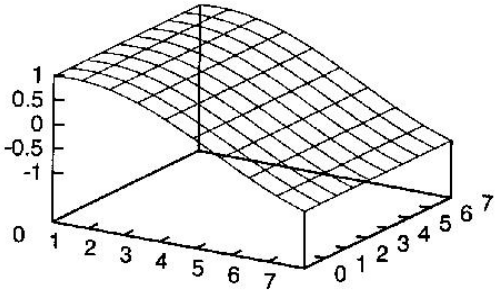
u0v0



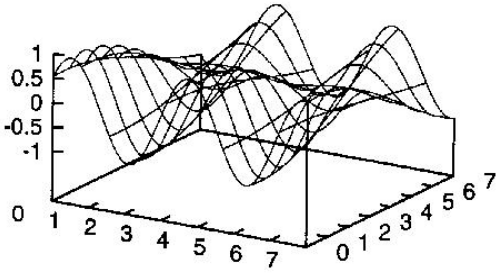
u2v2



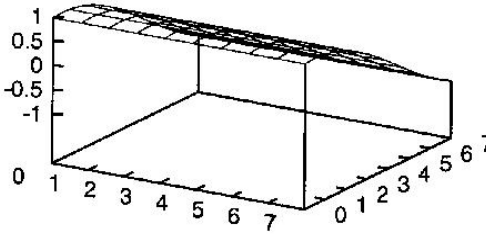
u1v0



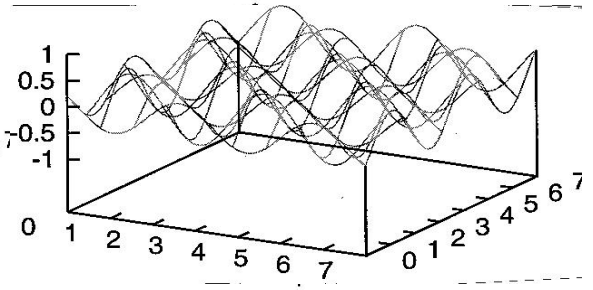
u5v1



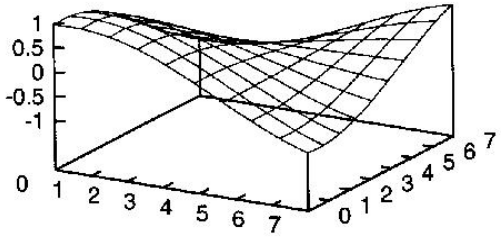
u0v1



u6v3

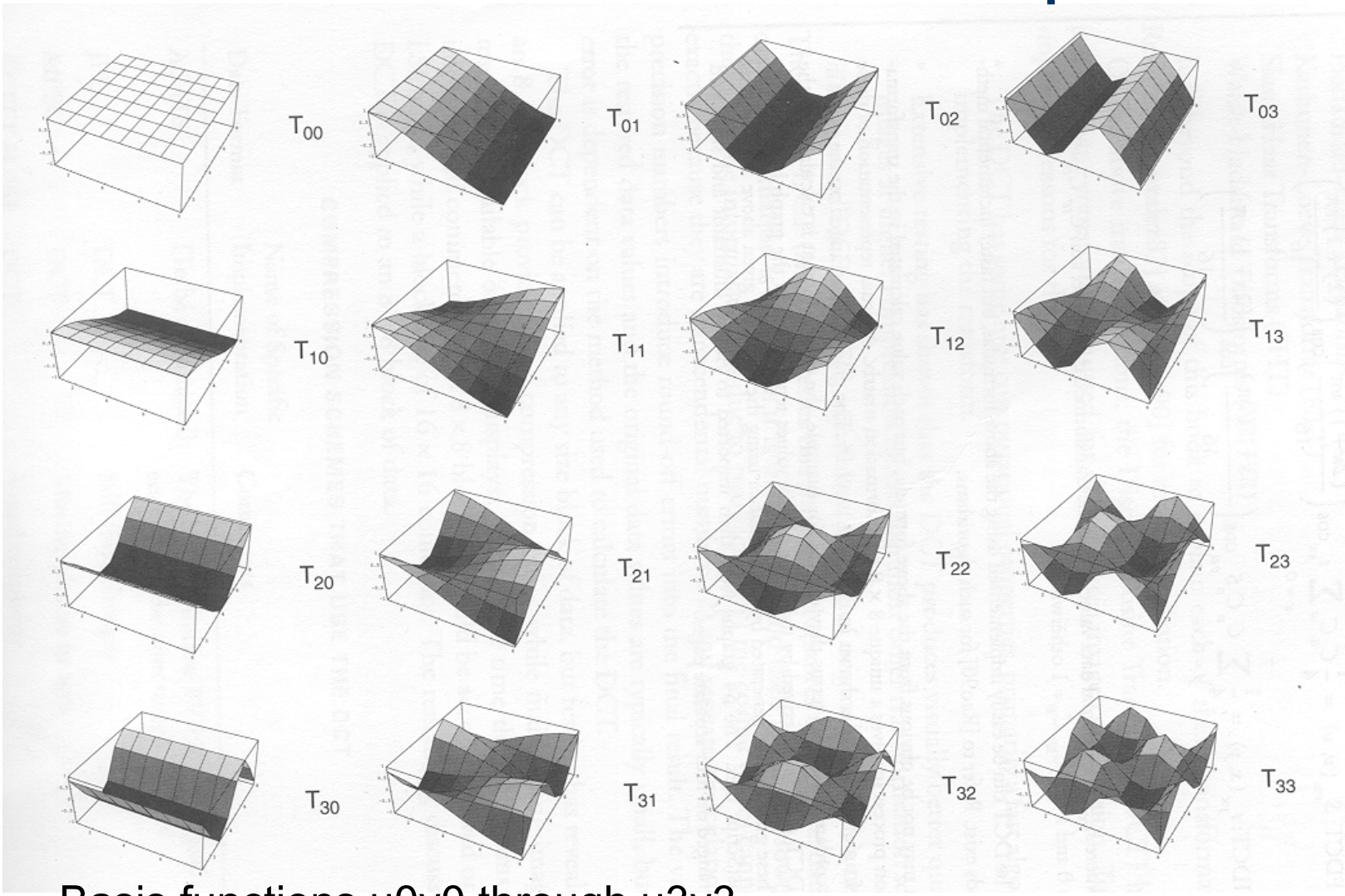


u1v1



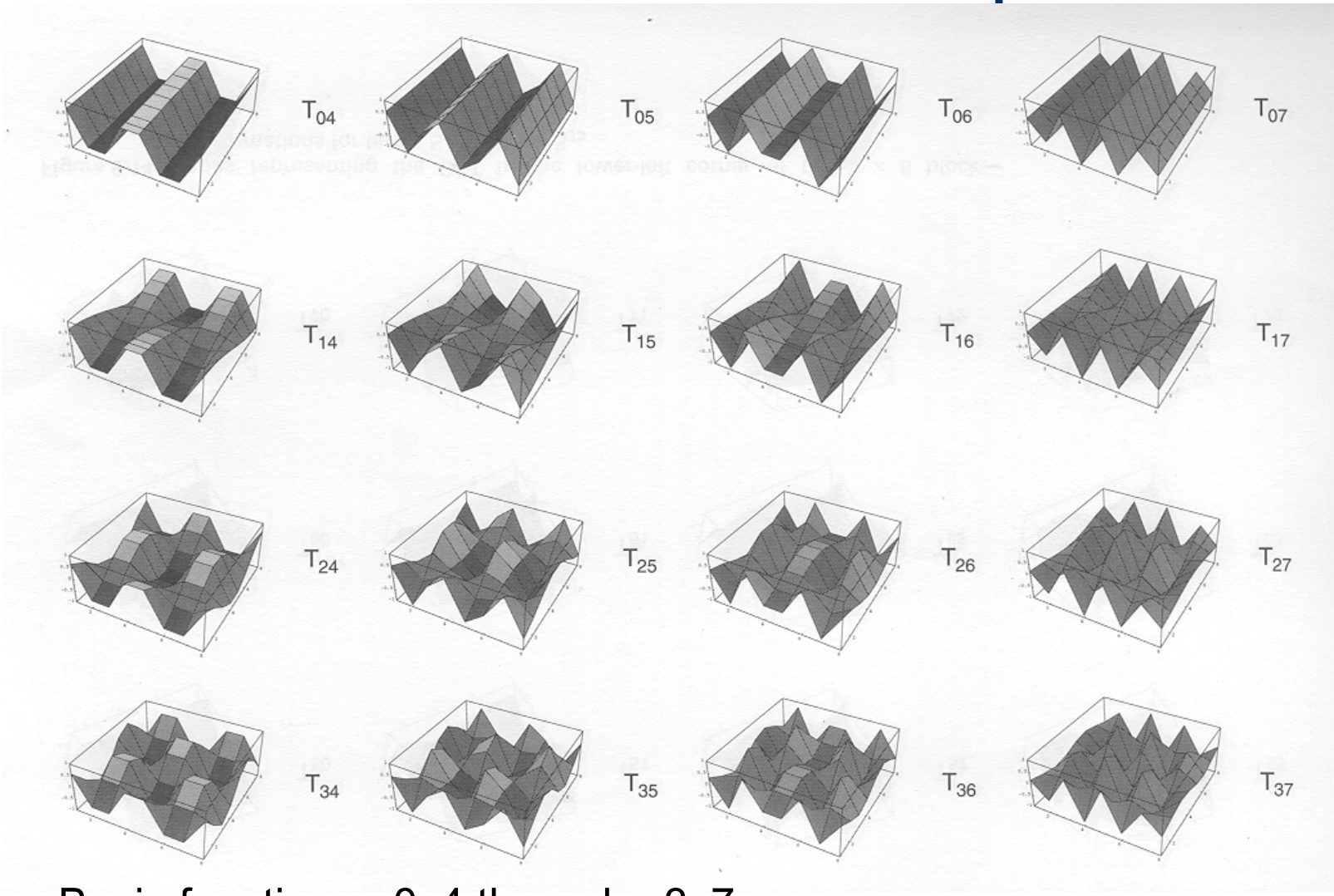
Some 2-D Basis Functions

DCT Basis Functions - Complete Set



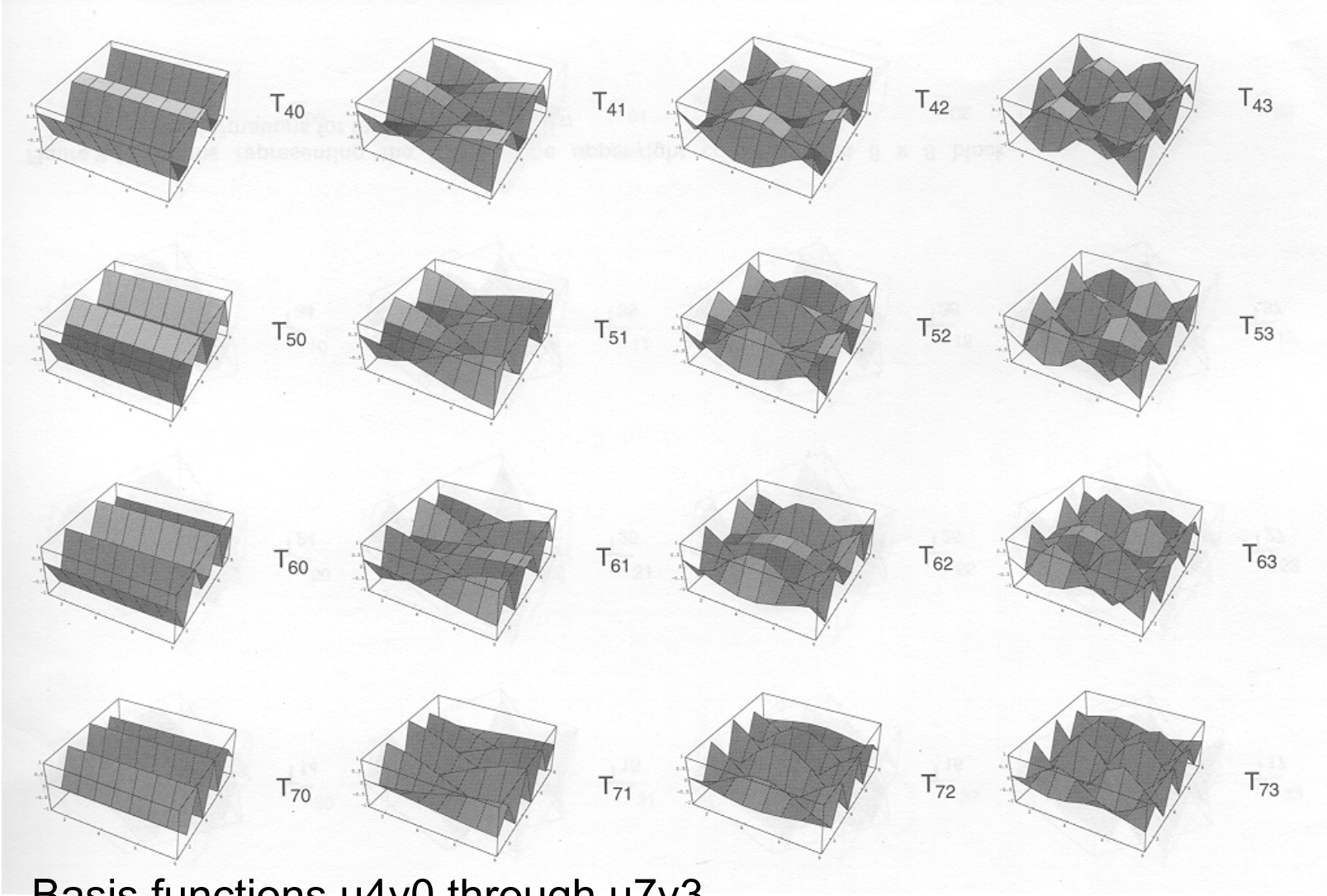
Basis functions u0v0 through u3v3

DCT Basis Functions - Complete Set (2)



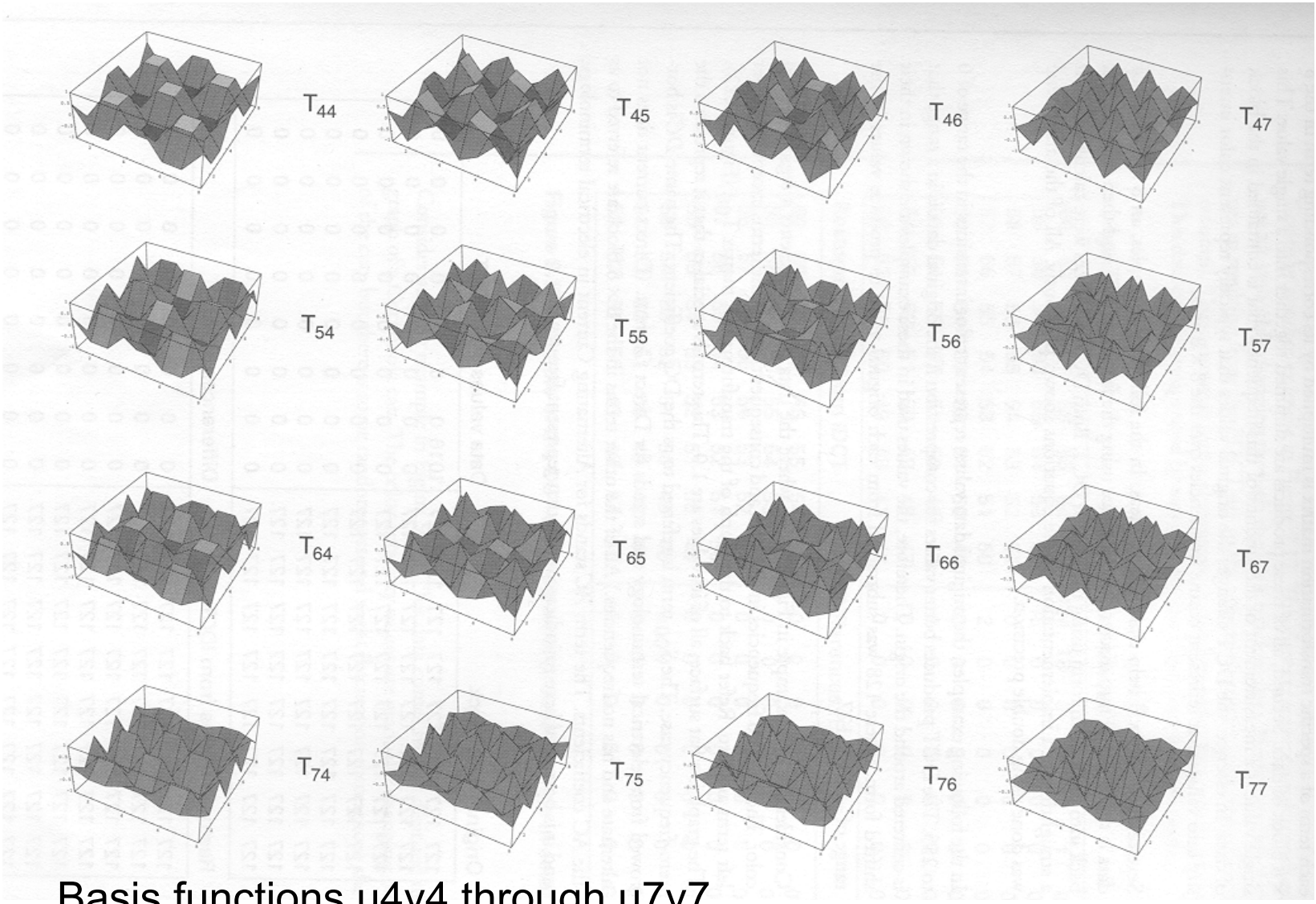
Basis functions u0v4 through u3v7

DCT Basis Functions - Complete Set (3)



Basis functions u4v0 through u7v3

DCT Basis Functions - Complete Set (3)



Basis functions u4v4 through u7v7

DCT

- From the original spatial function f(x,y), extract the frequency components by multiplying f(x,y) with these basis functions.

$$F(u,v) = \frac{1}{4} c_u c_v \sum_{x,y} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} f(x,y)$$

$$where \quad c_u, c_v = \begin{cases} 1/\sqrt{2} & for\ u,v = 0 \\ 1 & otherwise \end{cases}$$

- The result is a function F(u,v) in frequency domain, 64 (8x8) coefficients representing the 64 frequency components of the original image function.
  - F(0,0) is due to the basis function at u=0, v=0, which is a flat wave function.
    F(0,0) is known as the DC-coefficient.
  - Other coefficients are called the AC-coefficients as they are not flat.

DCT Coefficients, An Example

An 8x8 block

Large values accumulated at the top left corner

DCT coefficients after transformation

139	144	149	153	155	155	155	155
144	151	153	156	149	146	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

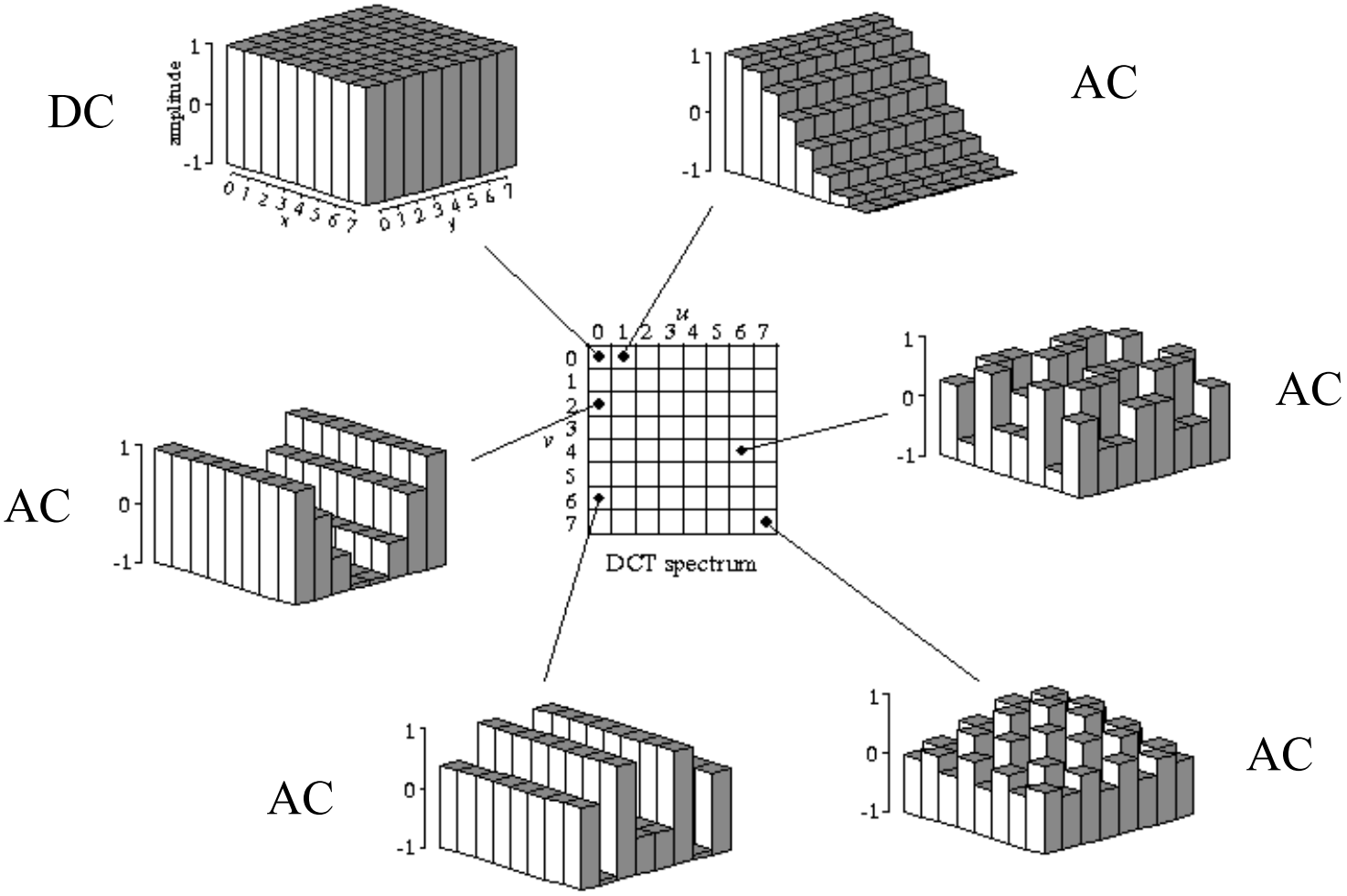
in x,y co-ordinates

233.1	0.3	-9.8	-7.9	2.1	-0.1	-3.7	1.1
-25.5	-15.9	-3.5	-6.4	-2.9	2.1	-0.7	-1.5
-12.3	-8.5	-0.3	0.1	0.2	0.0	-1.1	-0.2
-6.4	-2.3	-0.4	2.2	0.9	-0.6	0.2	0.4
1.9	-2.2	-0.8	4.3	-0.1	-2.5	1.6	1.5
5.2	-2.0	-1.6	3.4	-0.8	-1.0	2.4	-0.6
2.0	-2.1	-3.3	2.1	-0.5	-0.6	2.3	-0.4
-0.6	0.5	-5.6	0.3	1.9	-0.2	0.2	-0.2

in u,v co-ordinates

DCT Coefficients (2)

- The value in the element (u,v) is the coefficient (or amplitude) for the (u,v)-basis function.



DCT Coefficients (3)

- DC component determines the fundamental gray (color) intensity of the 8x8 pixels.
- AC components add the intensity variation to the pixel values to give the original image function.
- Typical image consists of large regions of single color. DCT thus concentrates most of the signal in the lower spatial frequencies. Many of the high-frequency coefficients are of very low values.
- In other words, the “energy” are “squeezed” at the top left corner of the DCT spectrum
- Entropy encoding applied to the DCT coefficients would normally achieve high data reduction
- That is why we choose DCT instead of Fourier Transform

Inverse DCT (IDCT)

- The inverse of DCT (IDCT) takes the 64 DCT coefficients and reconstructs a 8 x 8 output image by summing the basis signals.

$$f(x,y) = \frac{1}{4}\sum_{u,v} c_u c_v \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}F(u,v)$$

$$where \quad c_u, c_v = \begin{cases} 1/\sqrt{2} & for\ u,v = 0 \\ 1 & otherwise \end{cases}$$

- Imagine adding up the respective undulating surfaces to yield the original surfaces.

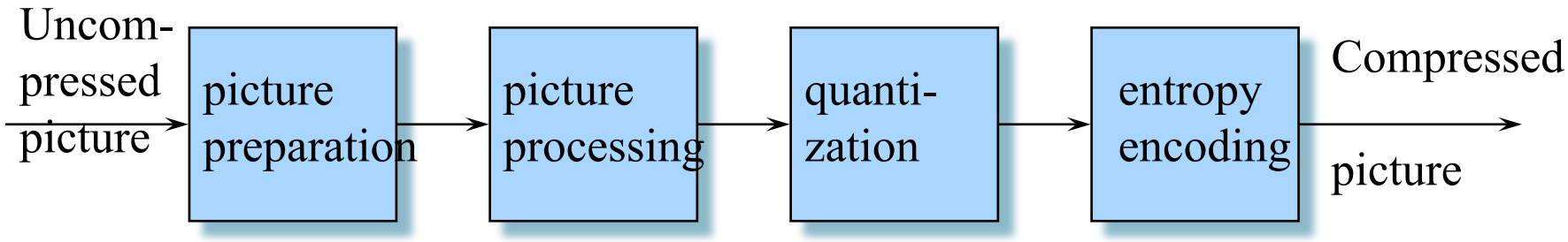
Quantization

- If DCT and IDCT were calculated *with full precision*, it would be possible to reconstruct the 64 pixels exactly.
- In practice, the DCT is quantized to throw away bits, and that is the main source of lossiness.
- Uniform quantization
  - DCT can be divided by a constant N and the result is rounded.
  - Equal treatment to all DCT coefficients.
- Quantization table
  - Each of the 64 coefficients can be adjusted separately. Specific frequencies can be given more importance than others according to the characteristics of the original image. (This is used in JPEG)

JPEG Standard

- DCT is extensively used in image compression due to the wide acceptance of JPEG standard
- JPEG stands for “Joint Photographic Experts Group”.
- A standard for compressing and encoding continuous-tone still images.
- It is not just a format!
- It is not just DCT, but also quantization, coding, ...
- Adjustable compression/quality.
- 4 modes of operations:
  - Sequential (line-by-line)
  - Progressive (blur-to-clear)
  - Lossless (pixel-for-pixel)

JPEG - Major Steps



1. *Preparation*
   - Includes analog-to-digital conversion. Image can be separated in YIQ components to facilitate subsampling on the chrominance components. The image is segmented into 8x8 blocks.
2. *Processing*
   - Sophisticated algorithms, such as transformation from time to frequency domain using DCT.

3. *Quantization*
   - Mapping real-number values from the previous step to integers. This process results in loss of precision, but achieves data compression.
   - It specifies the granularity of the mapping, allowing control of the precision carried in the compressed data.
   - Different levels of quantization are applied to the luminance and chrominance components, exploiting the sensitivity of human perception.
4. *Entropy encoding*
   - It compresses a sequential data stream without loss. Steps of zigzag scan to linearize the data. DPCM and RLE are used to encode the DC and AC components. Finally, Huffman scheme to encode the data.

JPEG - Major Steps (2)

- The schematic diagram:

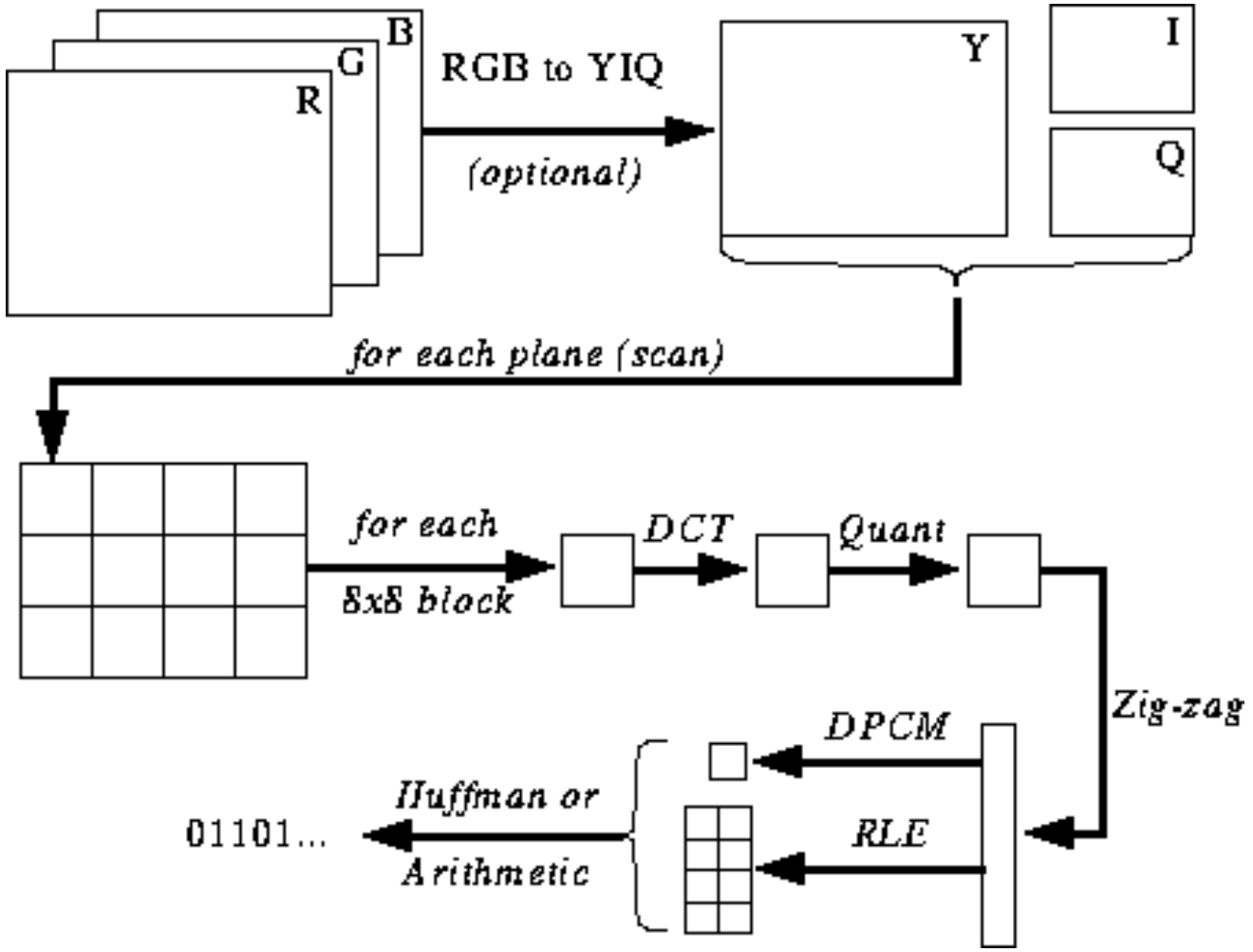


Image Preparation

- Each image consists of a number of components (e.g., YUV, YIQ, RGB).
- Divide each component into 8x8 blocks.
- Each block is a “data unit” subject to DCT compression.
- The values in a block are shifted from unsigned integers with range $[0, 2^p - 1]$ to signed integers with range $[-2^{p-1}, 2^{p-1} - 1]$

DCT

- Next step is DCT for each 8x8 image block as illustrated before

Quantization

- Quantization table
  - In JPEG, each $F(u,v)$ is divided by a different quantizer step size $Q(u,v)$ given in a 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

$$F_Q(u,v) = \text{round}\left(\frac{F(u,v)}{Q(u,v)}\right)$$

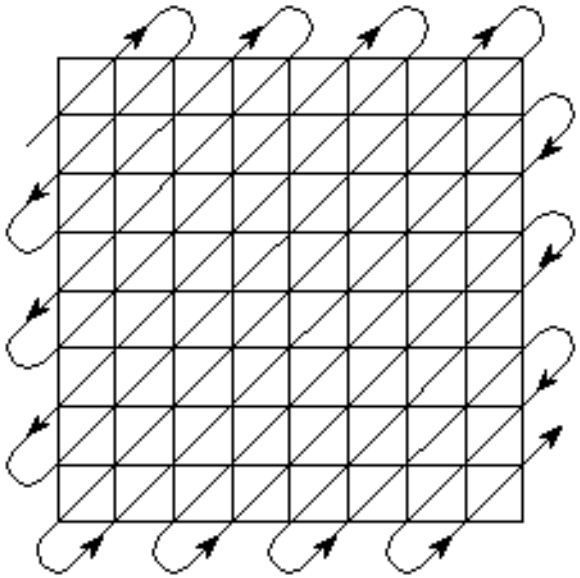
Quantization (2)

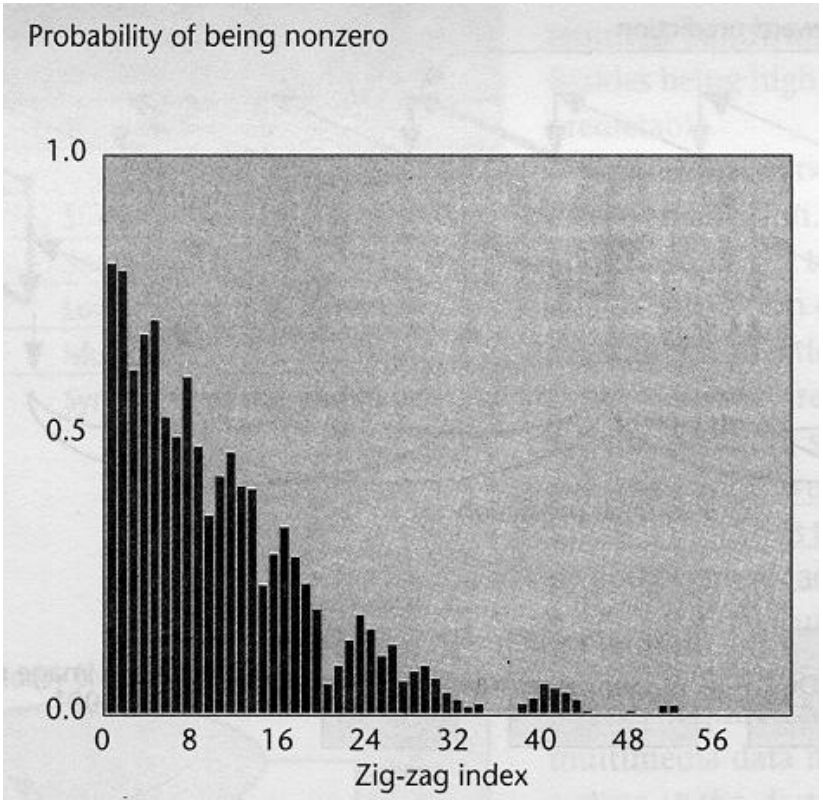
- The eye is most sensitive to low frequencies (upper left corner), less sensitive to high frequencies (lower right corner).
- JPEG standard defines 2 default quantization tables, one for luminance (above), one for chrominance.
- Quality factor:
  - How would scaling the quantization numbers affect the image, say if I double them all?
  - In most implementations, quality factor is the scaling factor for the default quantization table.

Zig-Zag Scan

- This step linearizes the 8x8 block of DCT coefficients. It maps 8x8 to 64-byte stream.
- The RLE and Entropy encoding methods are then applied on the byte stream.
- Why zig-zag? It is to group the coefficients from low to high frequencies, so that
  - Zeros in high frequencies are grouped together. Consecutive zeros would be effectively compressed using RLE.
  - High frequencies can be truncated in easy operations.

Zig-Zag Scan (2)





Entropy Encoding

- DC component encoded with DPCM
  - DC coefficient determines the average color (or intensity) of the 8x8 block.
  - Between adjacent blocks, the variation is fairly small.
  - Encode the difference between the current DC coefficient and the one of the previous block.
- AC components encoded with RLE
  - The 63-number stream has lots of zeros in it.
  - Encode as *(skip,value)* pairs, where *skip* is the number of zeros and *value* is the next non-zero component.

Entropy Encoding (2)

- Convert the DCT coefficients after quantization into a compact binary sequence in 2 steps:
  - forming an intermediate symbol sequence
  - converting the sequence into binary using Huffman tables
- Intermediate Symbol Sequence
  - each AC coefficient is represented by a pair of symbols:
    - Symbol-1 (*Runlength*, *Size*)
    - Symbol-2 (*Amplitude*)

AC Encoding

- *Runlength* is the # of consecutive 0-valued AC coefficients preceding the nonzero AC coefficient. *Runlength* is in the range 0 to 15.
- *Size* is the # of bits used to encode *Amplitude. Amplitude* can use up to 10 bits.
- *Amplitude* is the amplitude of the nonzero AC coefficient in the range of [-1024,+1023] => 10 bits.
- e.g., given the sequence:
  ....., 0, 0, 0, 0, 0, 0, 476 => (6,9)(476) // 2 symbols
- If *Runlength* > 15, use Symbol-1 (15,0) => more than 15 0's
- e.g., what is the sequence represented by:
  (15,0) (15,0) (7,4) (12)?
- (0,0) = End of Block

DC Encoding

- Categorize DC values into *Size* (number of bits needed to represent, symbol-1) and the *Amplitude* (symbol-2).

Amplitude	Size
0	0
-1, 1	1
-3, -2, 2, 3	2
-7, . . . , -4, 4, . . . , 7	3

- If DC is 4, 3 bits are needed. Have *Size* as Huffman symbol, then the actual 3 bits.
- Since DC are differentially encoded, its range is [-2048,2047].

JPEG Encoding (Example)

(a) Original 8x8 block

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

(b) Shifted block

12	16	19	12	11	27	51	47
16	24	12	19	12	20	39	51
24	27	8	39	35	34	24	44
40	17	28	32	24	27	8	32
34	20	28	20	12	8	19	34
19	39	12	27	27	12	8	34
8	28	-5	39	34	16	12	19
20	27	8	27	24	19	19	8

(c) Block after FDCT Eq. (5.1)

185	-17	14	-8	23	-9	-13	-18
20	-34	26	-9	-10	10	13	6
-10	-23	-1	6	-18	3	-20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

(d) Quantization Table (quality=2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

(e) Block after quantization Eq. (5.2)

61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	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

(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
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

(g) Intermediate symbol sequence

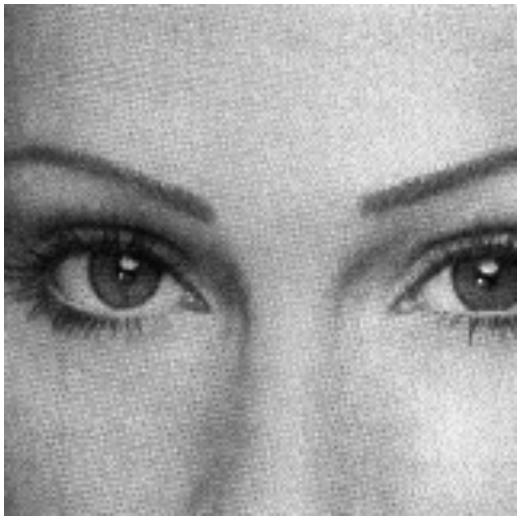
(6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
(0,2)(-2),(5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

(e) Encoded bit sequence (total 98 bits)

111011101001001000001000110110110111001011111111011
1101110101111101101110001110110111101001010

How Poor the Image Can Be?

- High compression ratio can introduce noticeable artifacts. For example,



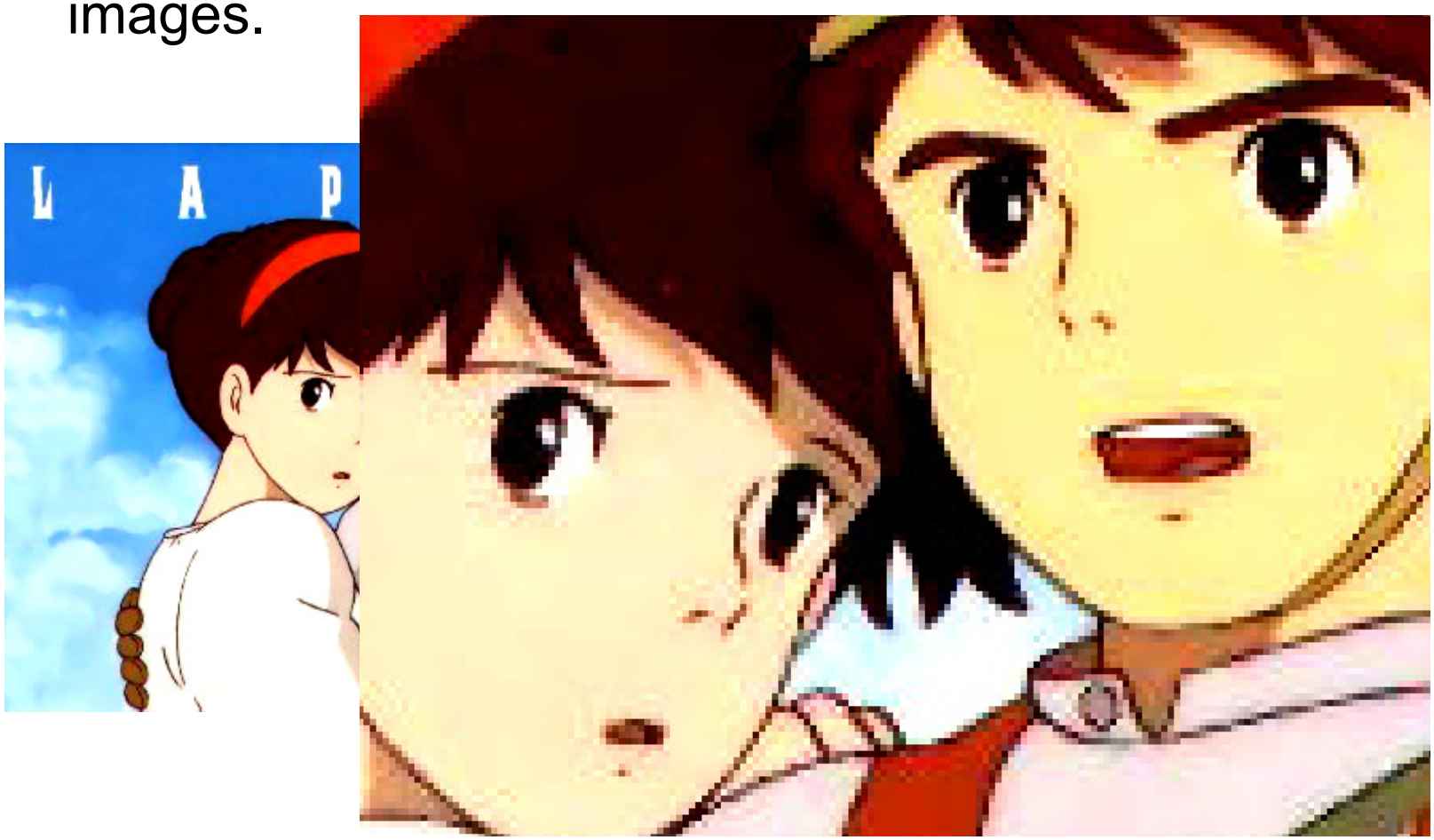
original

10:1

45:1

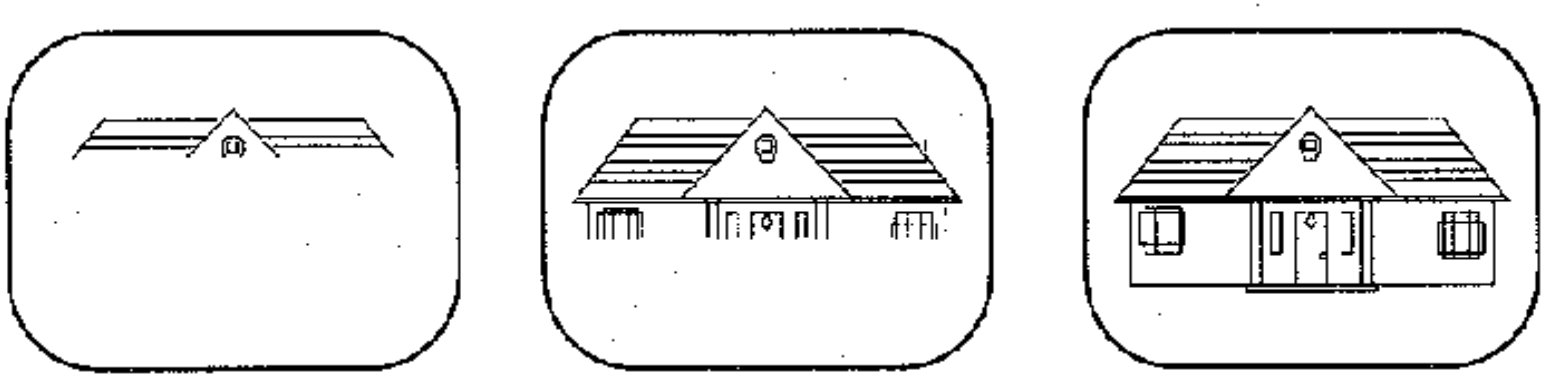
How Poor the Image Can Be?

- DCT is poor in handling sharp edges, e.g those cartoon images.



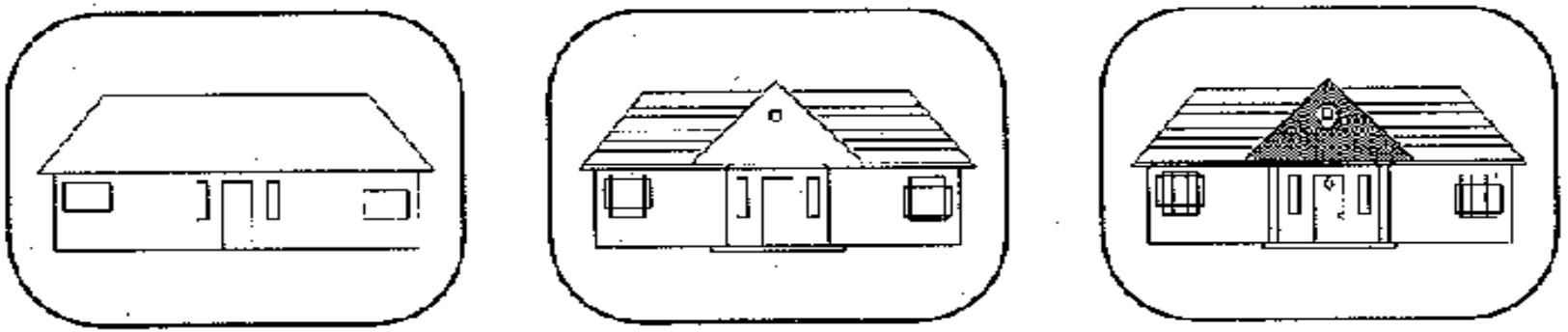
Sequential Encoding

- In sequential encoding, the whole image is encoded and decoded in a single run. It allows decoding with immediate presentation, but in top-to-bottom sequence.



Progressive Encoding

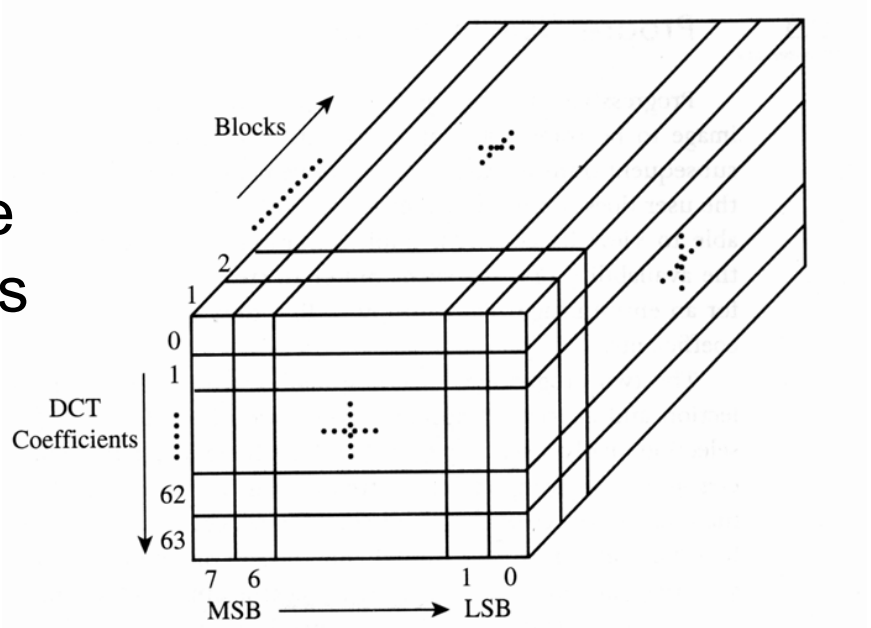
- Progressive mode encodes and reconstructs the image with a very rough representation, and refines it during successive steps.
- Also known as layered coding.
- Frequently used in web-based application due to the low network bandwidth.



Successive Refinement

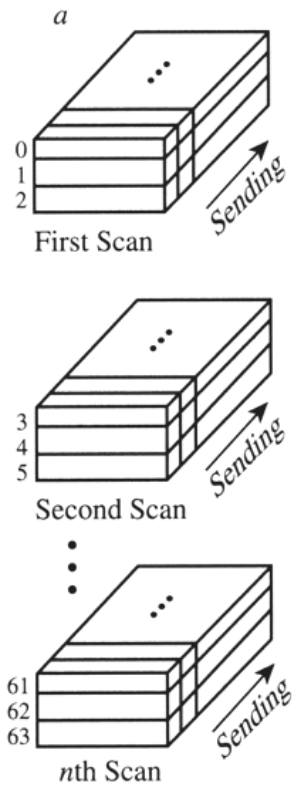
- 2 ways to successive refinement:
  - *Spectral selection*. Send DC component to the entropy encoding. Then first few AC, some more AC, etc.
  - *Successive approximation*. Send all DCT coefficients in each run, but single bits within a coefficient are processed in different runs. The most-significant bits encoded first, then the less-significant bits.

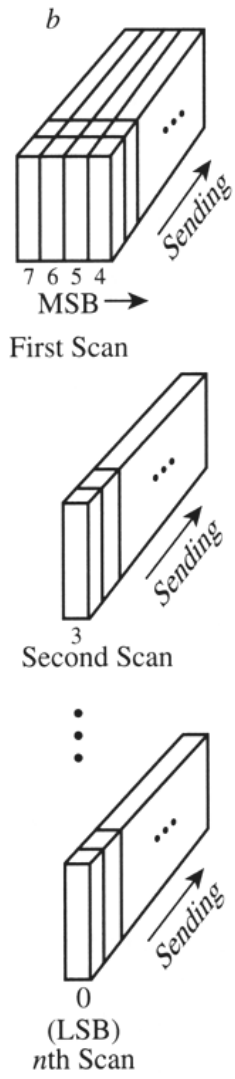
- Large buffer is needed to decode the progressive image while small buffer is enough for sequential decoding.



Successive Refinement (2)

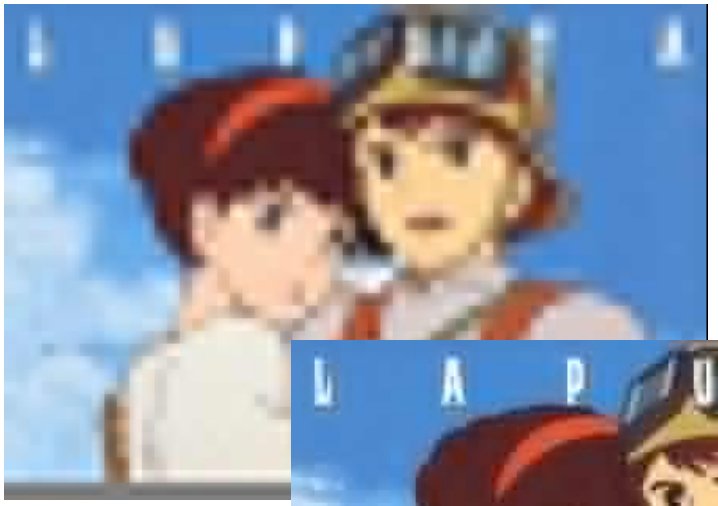
Spectral Selection





Successive Approximation

Successive Refinement (3)



10% sent

20% sent

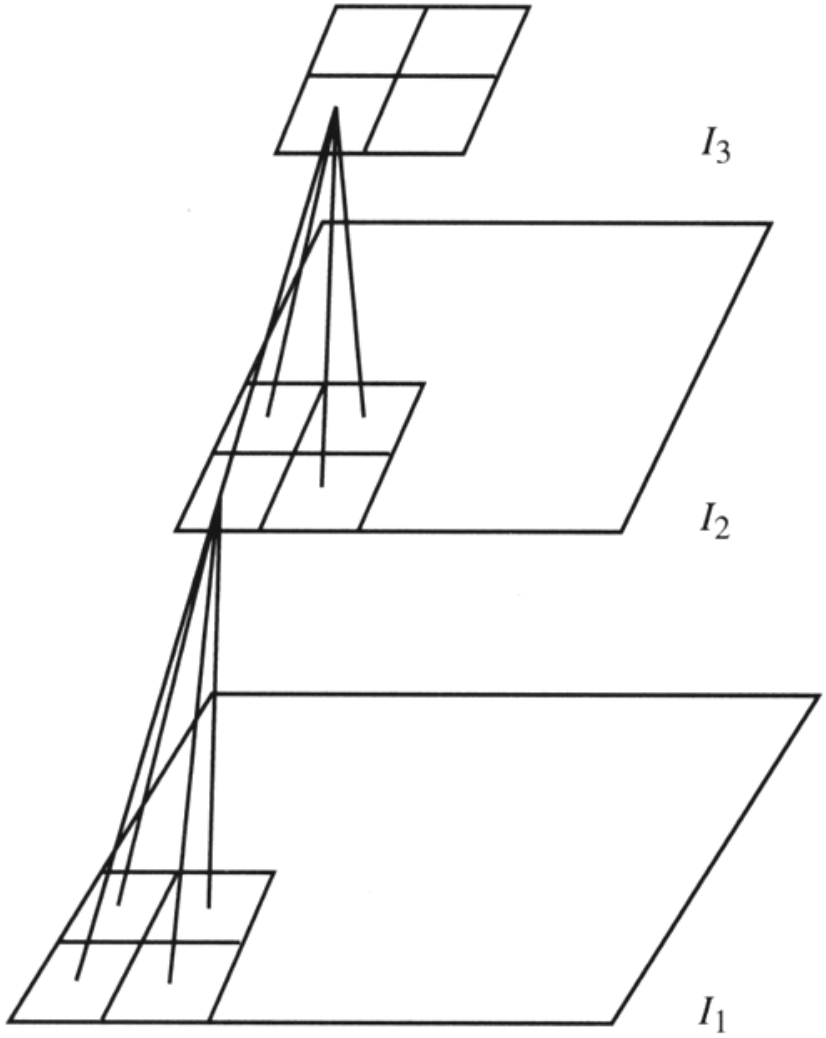
Original

50% sent

Hierarchical Mode

- Down-sample by factors of 2 in both directions. E.g., reduce 640x480 to 320x240 to 160x120, etc.
- Repeat the following process recursively until the full resolution image is compressed.
  - Initially, encode the smallest image. Then at each level:
    - Decode and up-sample the smaller image.
    - Encode the difference between the up-sampled and the original images.
- Since the original image is encoded at different resolutions,
  - It requires more storage for multiple resolutions.
  - Advantage: It is immediately available at different resolutions. Scaling is cheap when display system works only with a lower resolution.

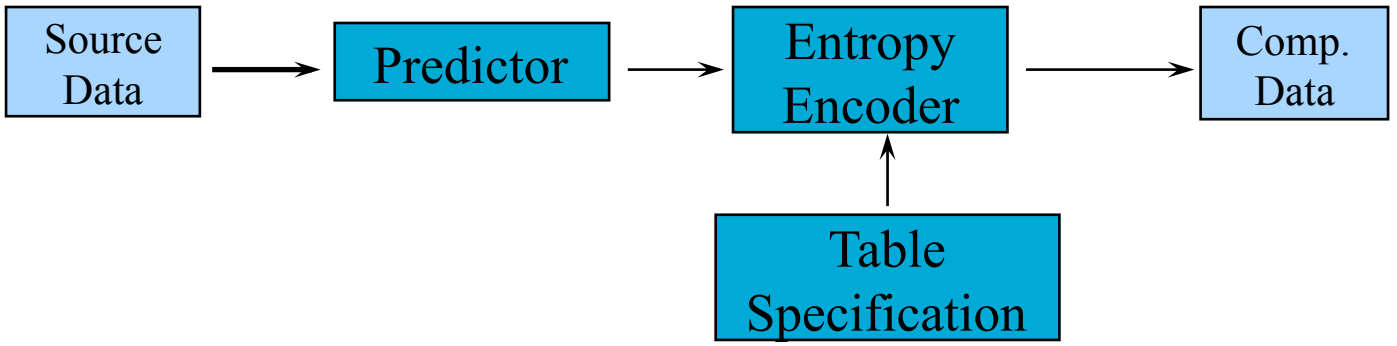
Hierarchical Mode (2)



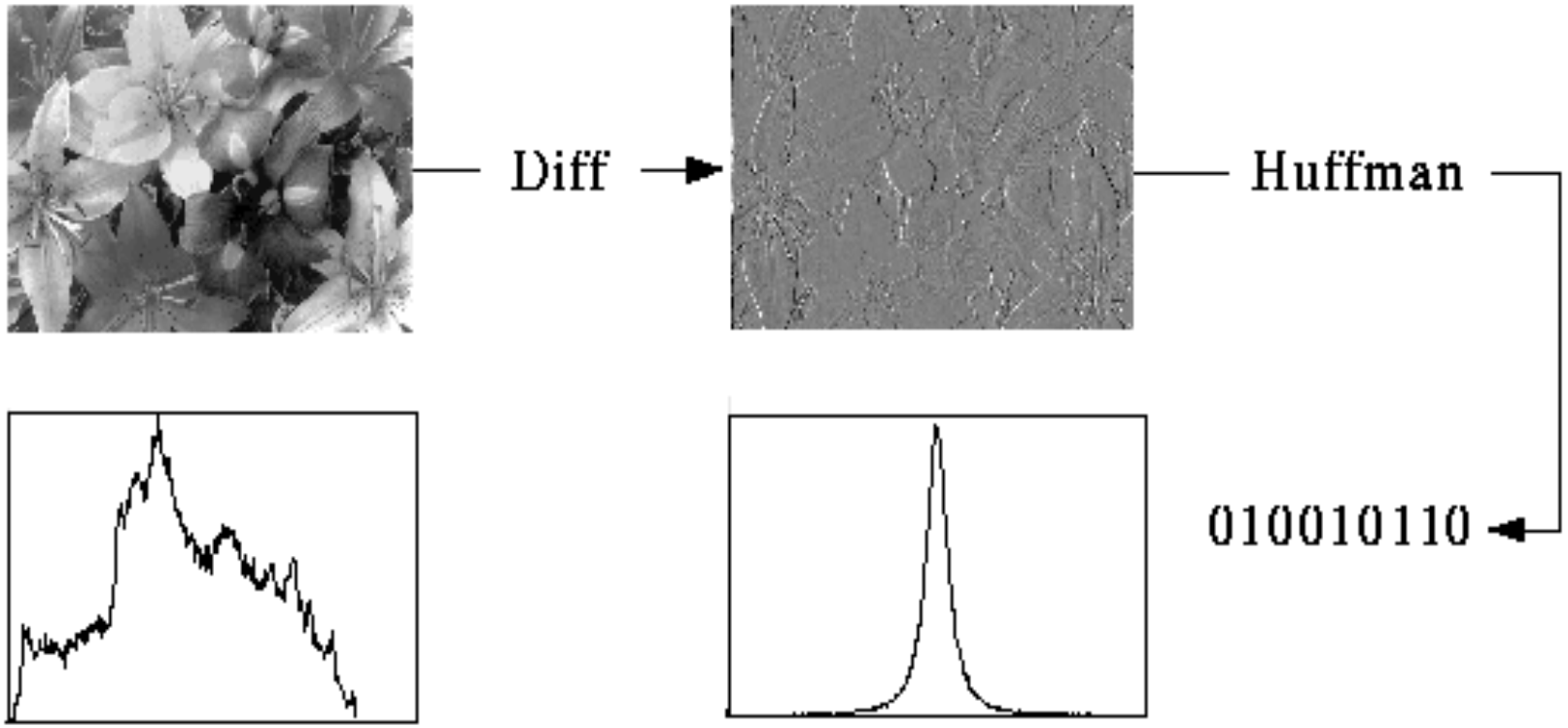
Pyramidal Filtering and Decimation of an Image

Lossless Mode of JPEG Compression

- A special case of JPEG where there is no loss in the encoding process. No DCT involved.
- In this mode, image processing and quantization use a predictive technique instead of transformation encoding.
- Neighboring pixels are taken as predictors, and the difference between the predicted and the actual ones are encoded using Huffman methods.

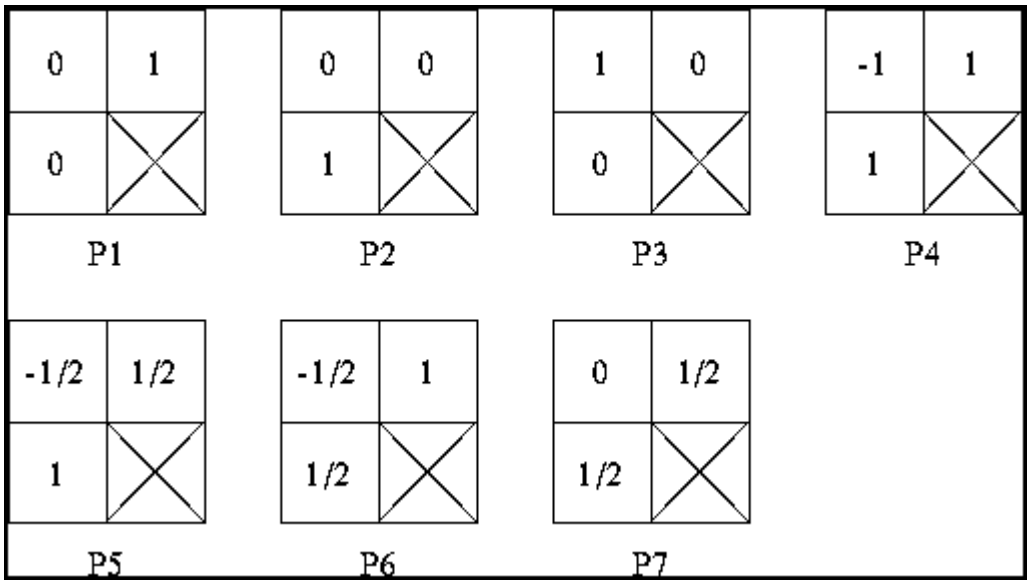


Lossless JPEG (2)



Lossless JPEG (3)

- Normally, pixel values do not vary by much except at intensity (color) edges. The differences have small values in most regions of the image. Effective entropy compression is possible.
- For each pixel, the predictor uses a linear combination of previously encoded neighbors. The typical predictor functions used are:



Lossless JPEG (4)

- 2D predictors (4-7) always do better than 1D predictors. (P0 is “no prediction”)
- Comparison with other lossless compressions:

Compression Program	Compression Ratio			
	Lena	Football	F-18	Flowers
lossless JPEG	1.49	1.67	2.71	1.33
Unix compress (LZW)	0.86	1.24	2.21	0.87
gzip (also LZW-based)	1.08	1.36	3.13	1.05
pack (Huffman coding)	1.02	1.12	1.19	1.00

- Typical compression ratio 2:1, similar to normal entropy coding.
- Seldom used.