Image Compression & Coding

Why Compression?

- There are too many data around.
- **Data ≠ Information**; the difference is the **redundancy**.
- Compression: The process of reducing the amount of data while preserving as much information as possible.
- Types of compression:
  - Lossless:
  - Lossy:

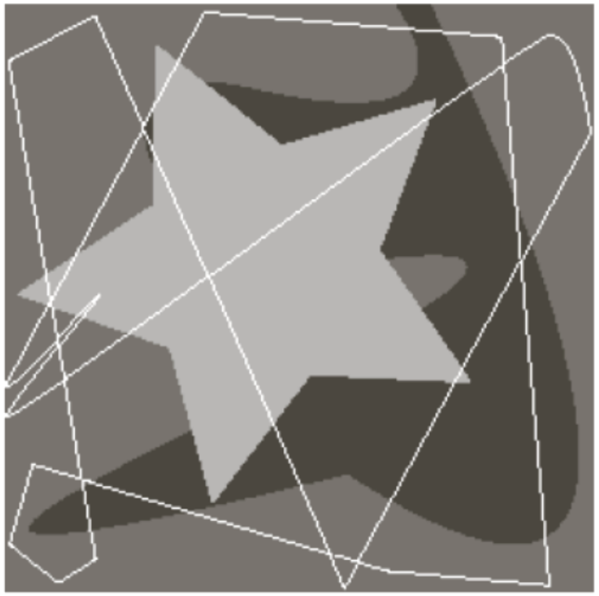
Data Redundancy

- Compression is achieved by the reduction of redundancy in data.
- Three types of data redundancy involved in image compression:
  - Coding redundancy:
  - Spatial and temporal redundancy:
  - Irrelevant information:

Image Data Redundancy

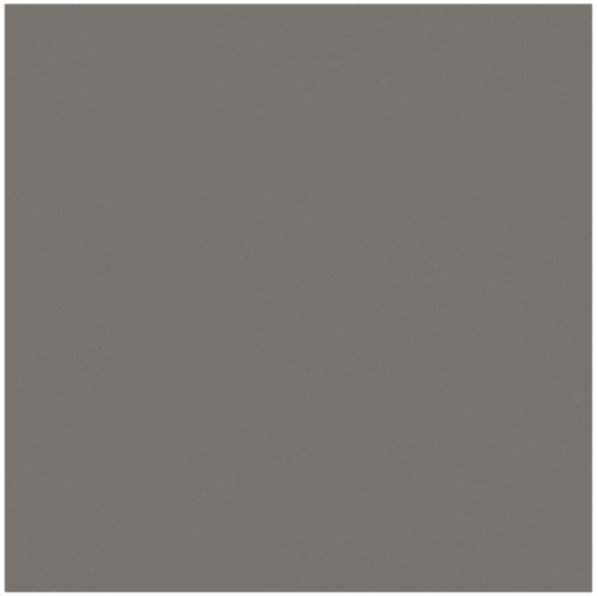
Examples:

Coding Redundancy



Spatial Redundancy

Irrelevant Information



Coding Redundancy

Average length of code (bits per symbol):

$$L_{avg} = \sum_{k} l(r_k)\, p_r(r_k)$$

probability of symbol

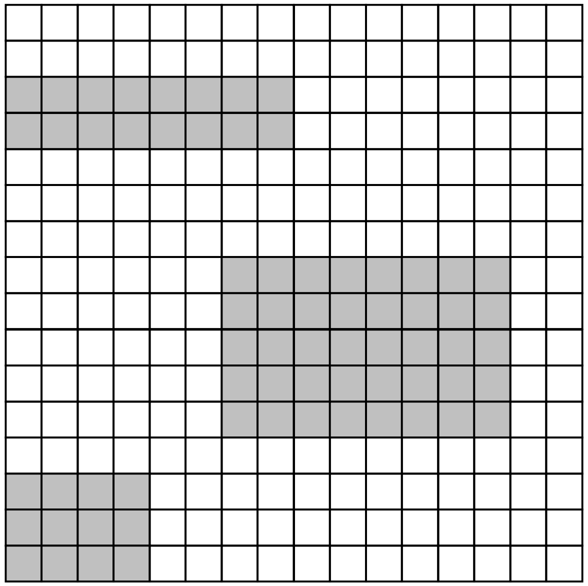
code length of symbol

An example of VLC (variable-length coding):

r_k	$p_r(r_k)$	**Code 1**	$l_1(r_k)$	**Code 2**	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

What are the resulting compression ratios of the two codes?

Spatial Redundancy



This is a 16x16 binary image.

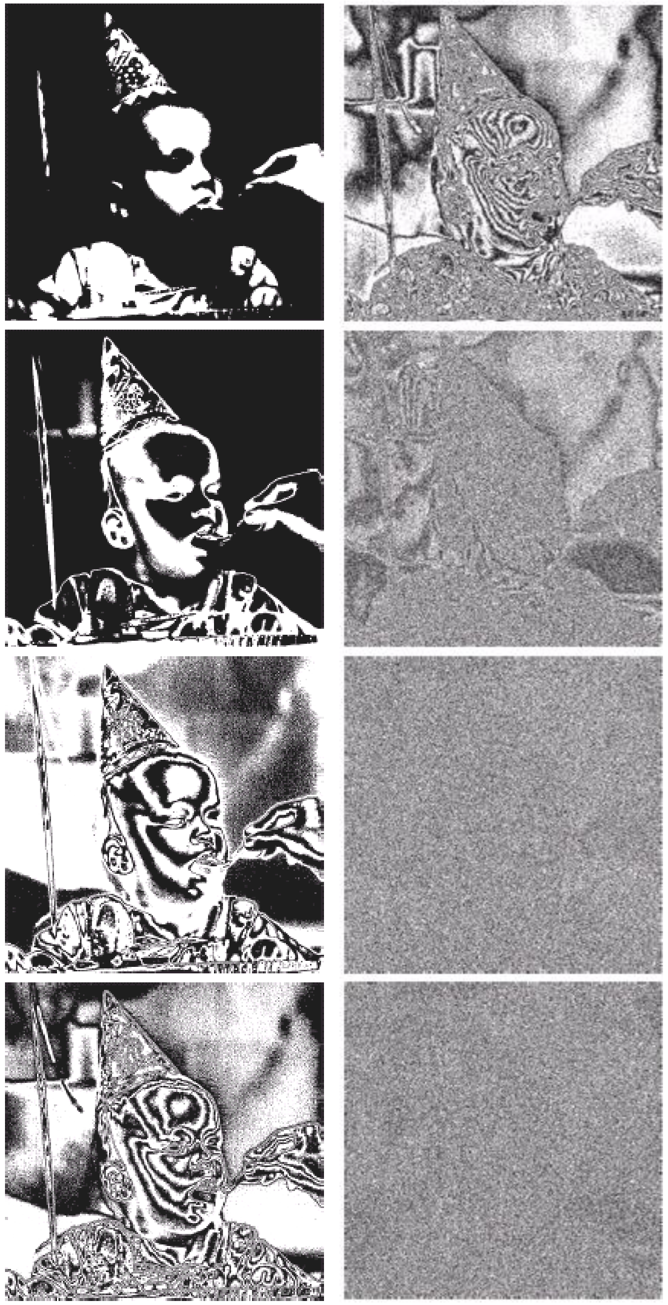
Amount of data (uncompressed):

What is an efficient way to represent the content of this image?

Irrelevant Information

For the viewer/user of a picture, not every bit of the data is equally important. This means that we can discard some (less important) data while preserving as much visual quality or useful information as possible.

An example of visually important and unimportant data involves the bit planes of an image:



Fidelity Criteria

For lossy coding, we need some criteria to quantify the resulting "information loss". Below are a few common ones:

- Root-mean-square (RMS) error:

$$e_{rms} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2 \right]^{1/2}$$

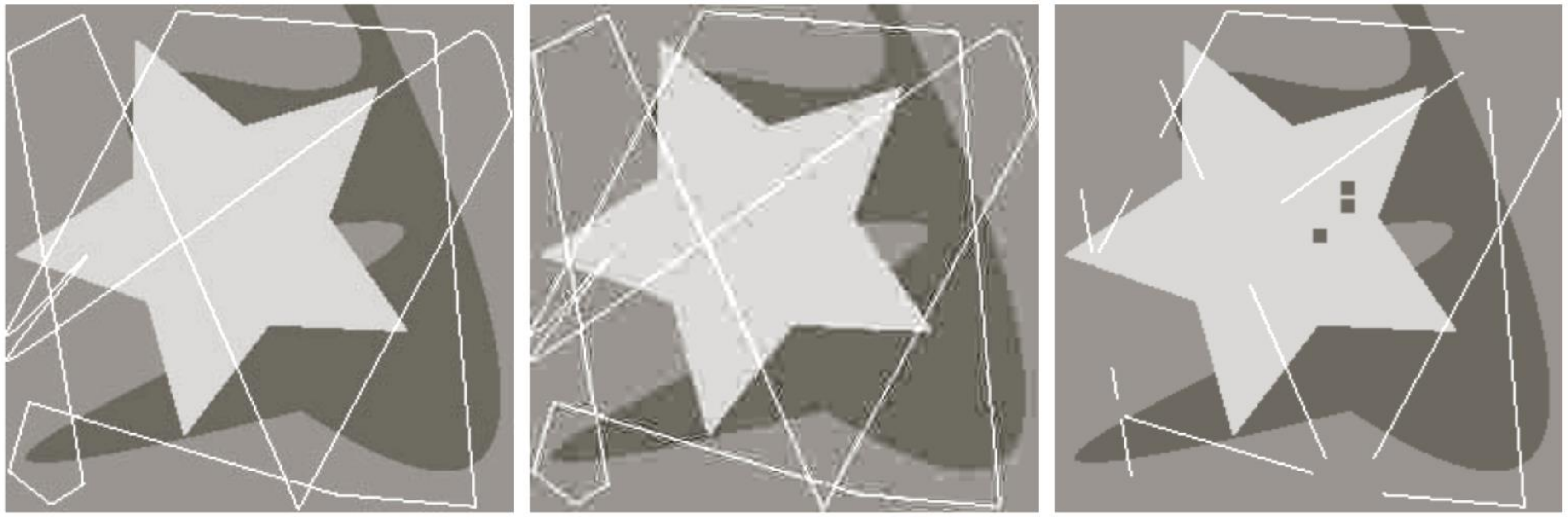
- Signal-to-noise ratio (SNR):

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) \right]^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2}$$

- Subjective evaluation by human viewers:

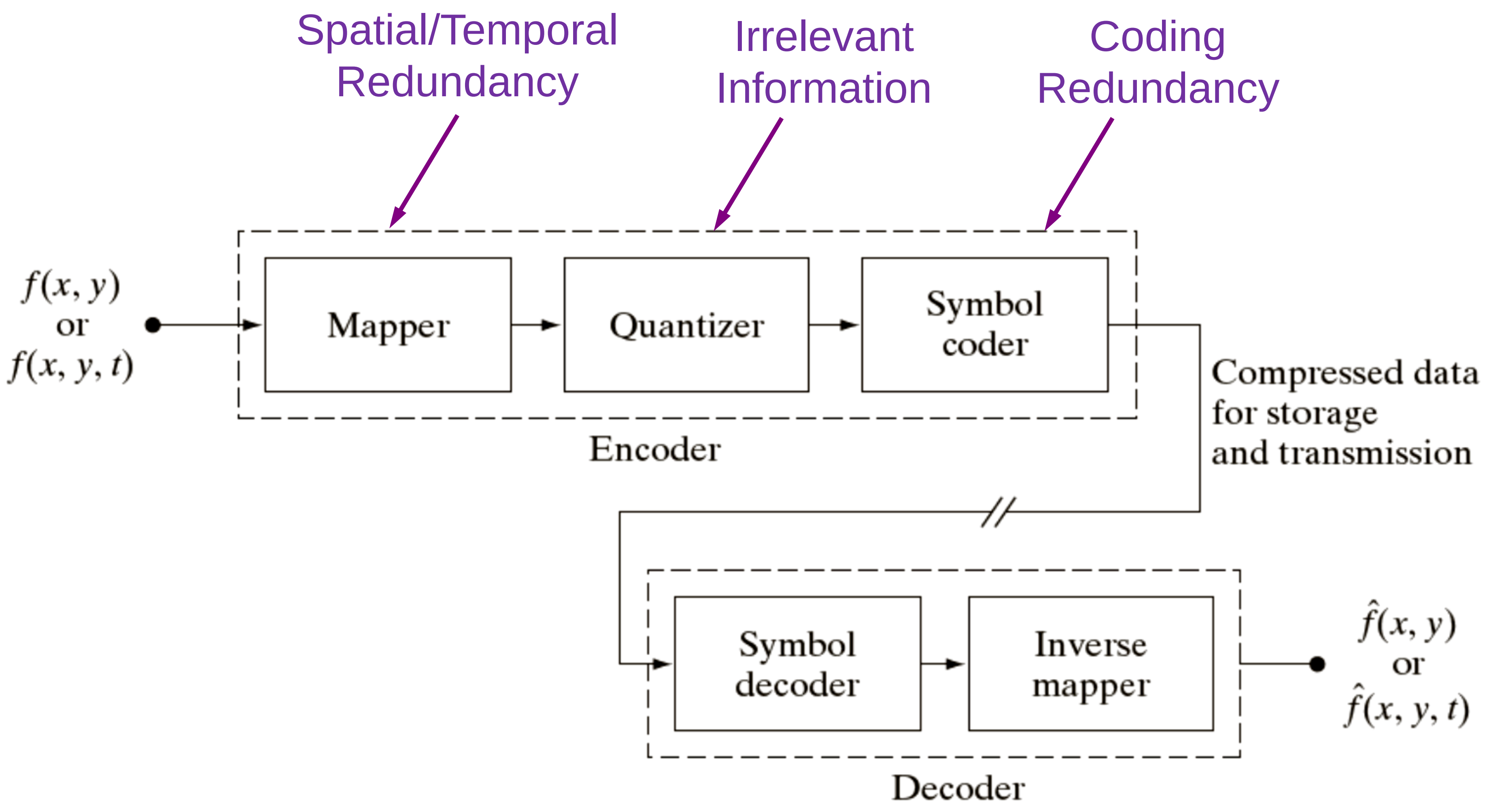
Fidelity Criteria

An illustration of the limitations of objective criteria:



For the two images after compression (middle and right), which one is better according to RMS or SNR? Which one is better at retaining visual information?

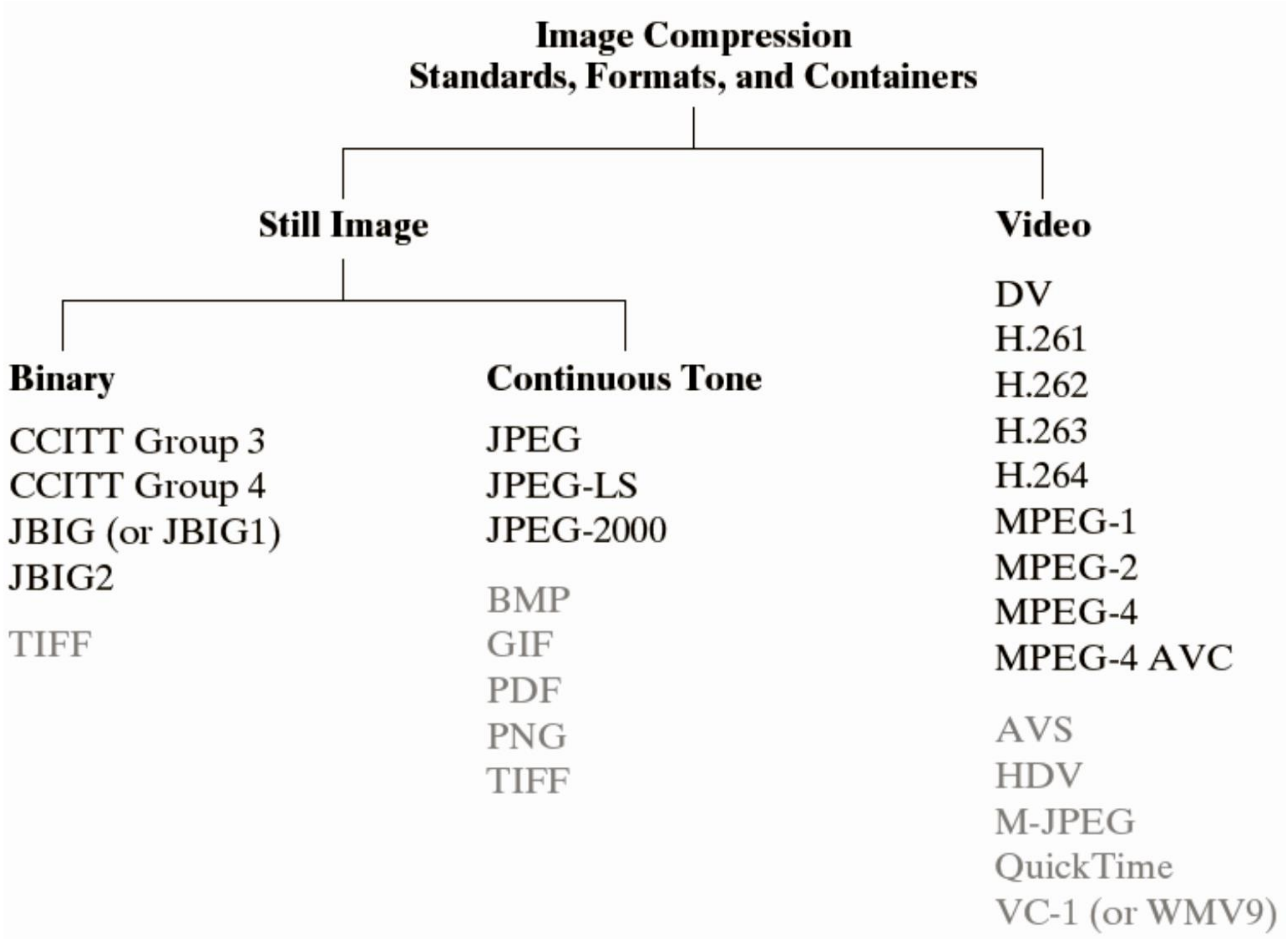
Encoder/Decoder Models


Spatial/Temporal Redundancy
Irrelevant Information
Coding Redundancy
$f(x, y)$ or $f(x, y, t)$
Mapper
Quantizer
Symbol coder
Encoder
Compressed data for storage and transmission
Symbol decoder
Inverse mapper
Decoder
$\hat{f}(x, y)$ or $\hat{f}(x, y, t)$

Coding Techniques - Overview

- Symbol encoder/decoder: Symbols are replaced with "codewords", exploiting non-random statistics of the original symbols.
  - Example: Huffman coding
- Quantizer (lossy): Symbols are replaced with a reduced set of symbols. This step is irreversible.
  - Example: 24-bit color image to 8-bit color image
- Mapper / Inverse mapper: Convert the data to a new representation that can be coded more efficiently.
  - Run-length coding and extensions: Exploit situations when recurring symbols are common.
  - Predictive coding: Exploit spatial and temporal coherence; just code the differences from predictions.
  - Transform coding: Use a set of transform coefficients to represent the data.

Common Image Coding Standards



Symbol Coder / Decoder

Idea: Replace "symbols" in data with "codewords", so that the overall data require fewer bits to represent.

- Fixed-length input strings, variable-length codewords: **Huffman coding** and similar methods
- Variable-length input strings, fixed-length codewords: **LZW** and similar methods
- Variable-length input strings, variable-length codewords: **Arithmetic coding**

The symbol coder/decoder is a part of all coding systems; many other methods such as predictive coding are used to convert the data into a form more suitable for symbol coding.

Entropy and Coding

The entropy of a discrete random variable (RV) F:

$$H(F) = -\sum_{f \in A} p_F(f) \log_2 [p_F(f)]$$

The entropy is a measure of the uncertainty about (and therefore the amount of information content of) a RV.

Simple examples:

- An image with random pixel values between 0 and 255:
- An image with a constant pixel value:
- An image with random pixel values between 240 and 255:

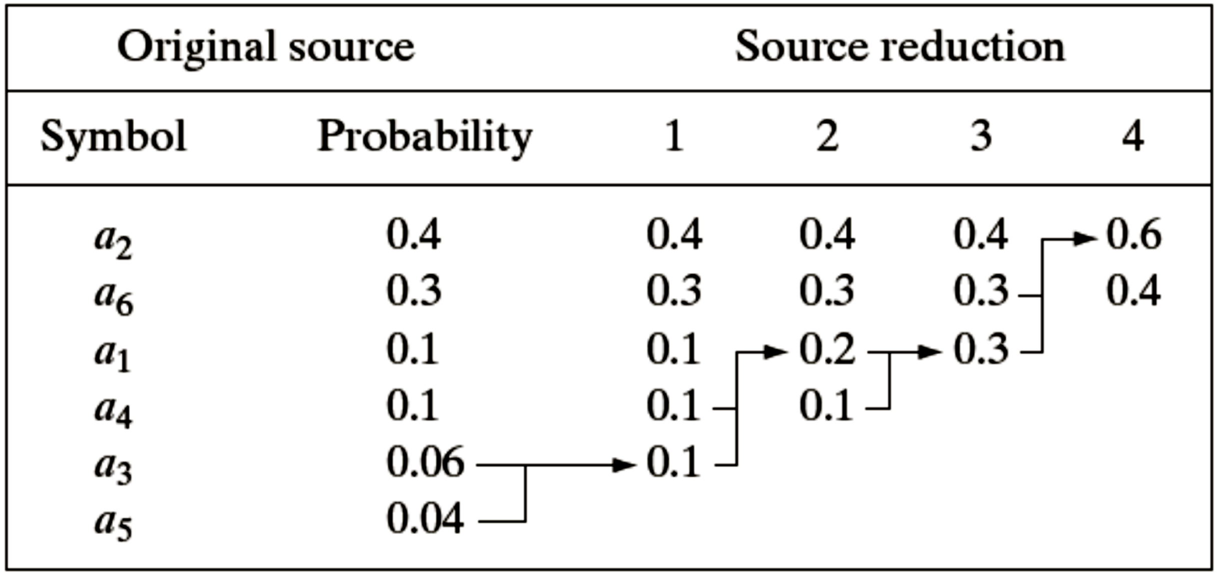
Minimum bit per sample required in lossless coding:

$$H(F) \leq R(F) \leq H(F) + 1$$

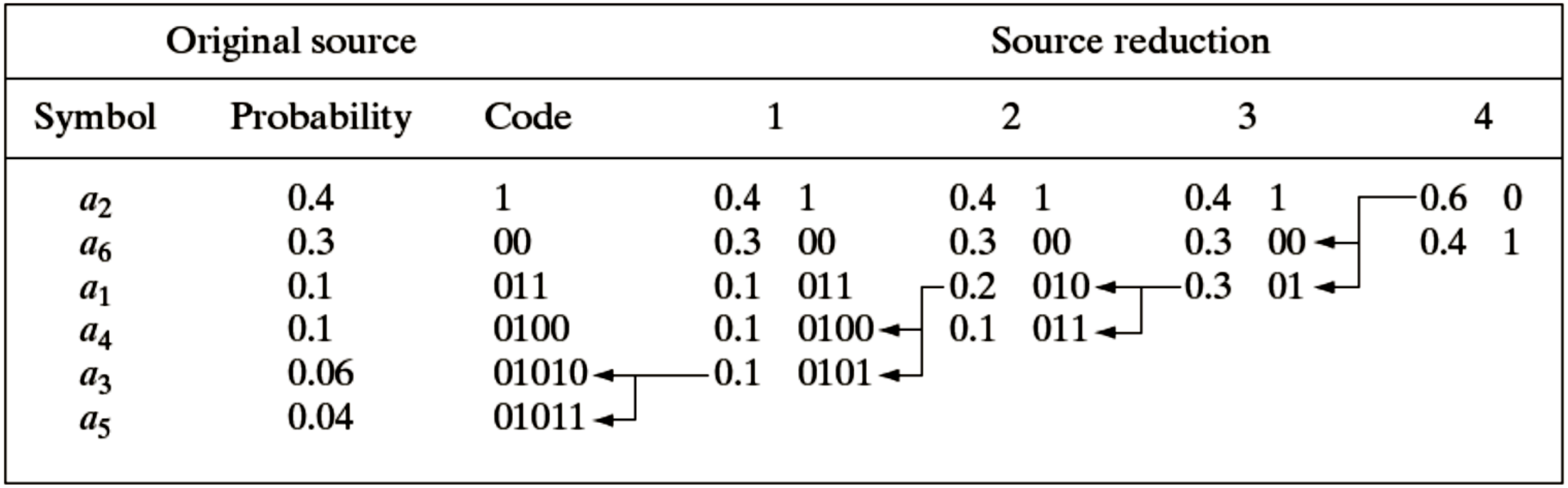
Huffman Coding

- Determine the probability of each symbol in the source. Assign a node for each symbol.
- Loop while there are at least two nodes:
  - Find the two nodes with the lowest probabilities and randomly assign 0 and 1 to them.
  - Merge these two nodes, with the probability of the new node being the total probability of the two nodes.
- The codeword of a symbol is the list of assigned bits on the path from the root to the leaf node corresponding to that symbol.
- Coding efficiency / compression ratio is limited by the requirement that each symbol is coded separately.

Huffman Coding: Example



Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				



Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

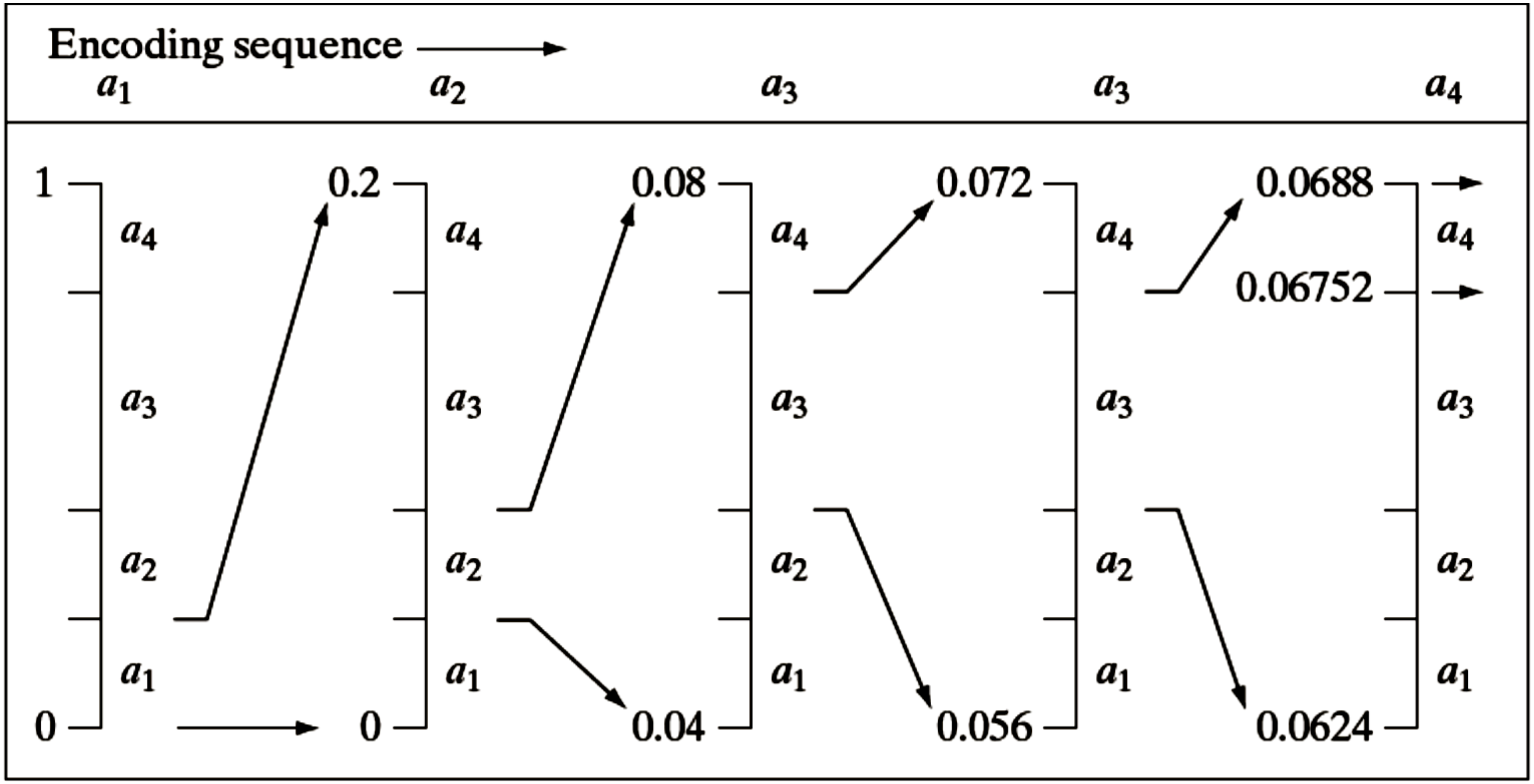
Now try to decode 1000110101011011.

Arithmetic Coding

- Each input symbol is given a (non-overlapping) interval within [0,1) according to its probability.
- Each codeword (an interval within [0,1)) can represent a sequence of several symbols (a "message").
- The coding of a message involves a series of interval subdivision.
- A single number within the interval can be used to represent the codeword. However, every message now requires an extra special end-of-message symbol.
- The number of bits in a codeword is determined by the necessary precision for uniquely decoding the message.
- Without the "one codeword per symbol" constraint of Huffman coding, arithmetic coding can usually achieve a compression efficiency closer to the entropy limit.

Arithmetic Coding: Example

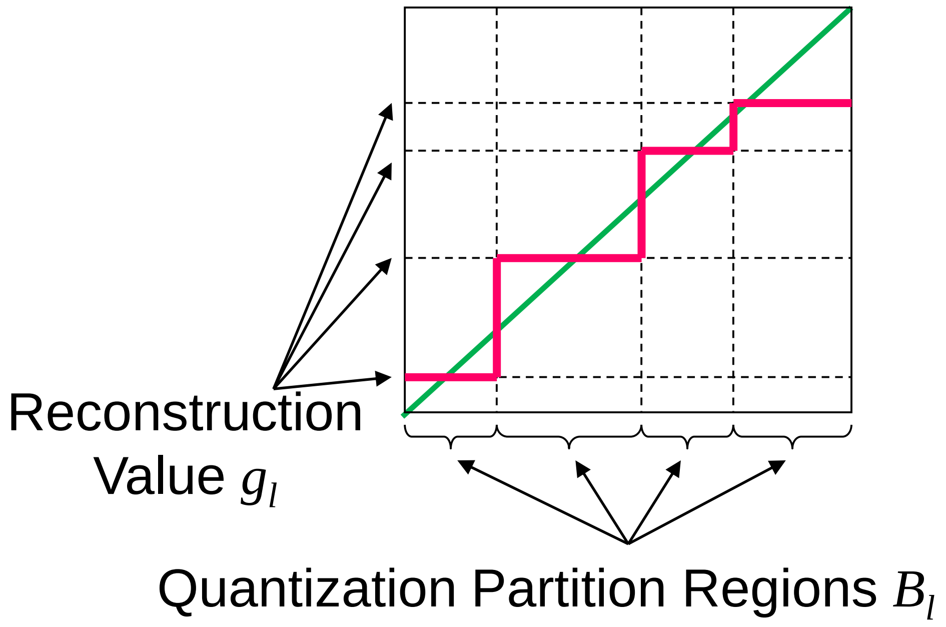
Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Quantization

- A codeword is used to represent a set of possible inputs.
- A "lossy" step: Given a codeword, we are unable to determine the exact original input.
  - Instead, each codeword is mapped to a reconstruction value, which approximates the corresponding input, during decoding.

Scalar quantization example:


Reconstruction Value g_l
Quantization Partition Regions B_l

Uniform quantization:

quantization stepsize
= size of each partition region
= separation between reconstruction values.

Run-Length Coding (RLC)

Let us consider binary images for now. This is the basis of compression standards such as those used for FAX.

Example: Consider the run-length coding of the following pixel sequence: 1111100011111111100111111010111

The run-length codes can be further compressed with a symbol coder. In fact, we can use separate codebooks for black and white runs to take advantage of their different statistics (e.g. for text documents).

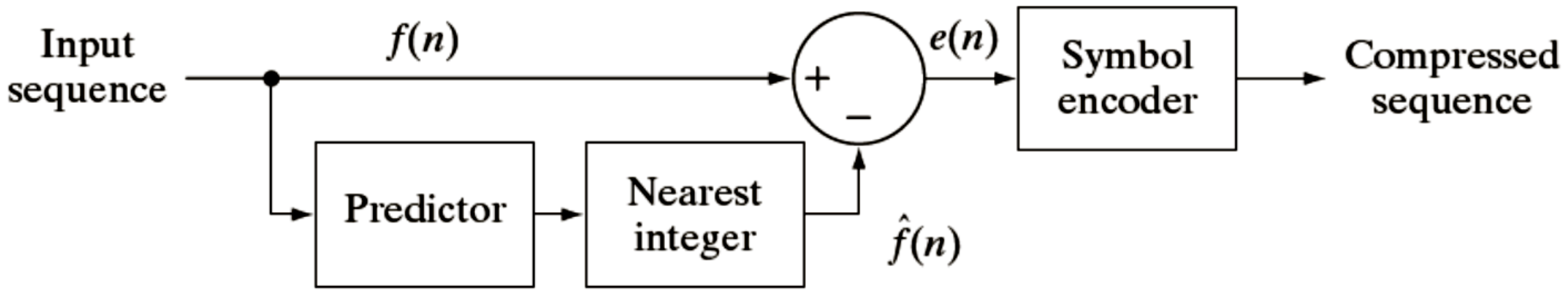
Bit-Plane Coding

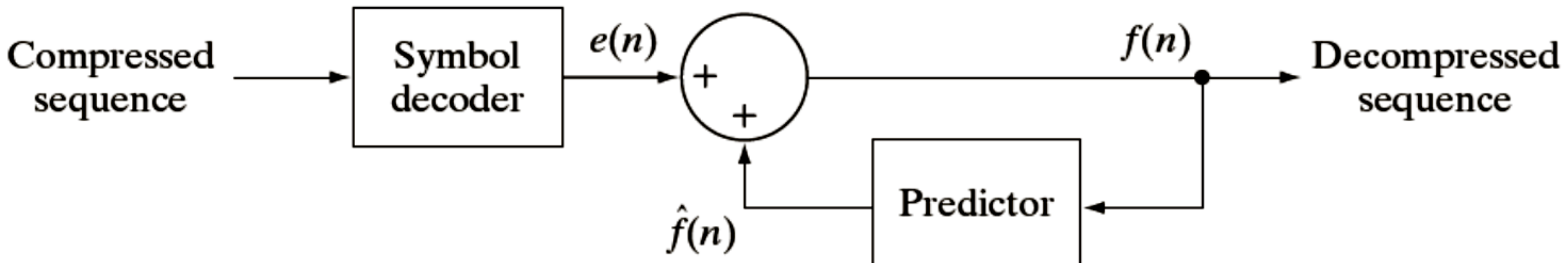
- Each bit plane is a binary image.
  - Code bit planes using any binary image coding method (e.g., run-length coding).
- For best coding efficiency, it is often a good idea to convert the pixel values to **Gray codes** before the decomposition into bit planes.
  - Reason: Gray codes minimize the number of bits changed by small changes in gray values. Fewer bits changed means longer runs of 0 and 1.

Predictive Coding

- Goal: To exploit the correlation between nearby samples in the source. (General idea: Difference has lower variance and therefore can be coded more efficiently.)
- Can be lossless or lossy.
- Linear predictor is most often used: $s_p = \sum_k a_k s_k$
- Higher-order nonlinear predictors can achieve very good coding efficiency (even better than transform coding) but is not often used in practical systems.

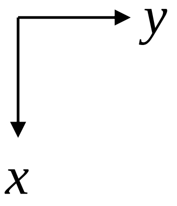
Predictive Coding - Lossless

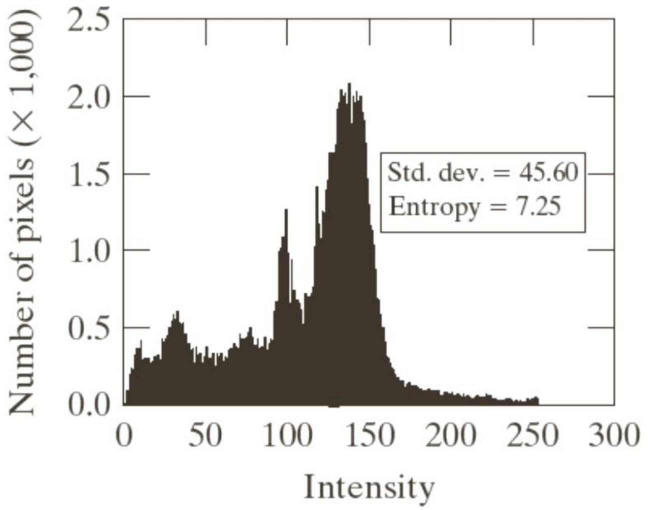


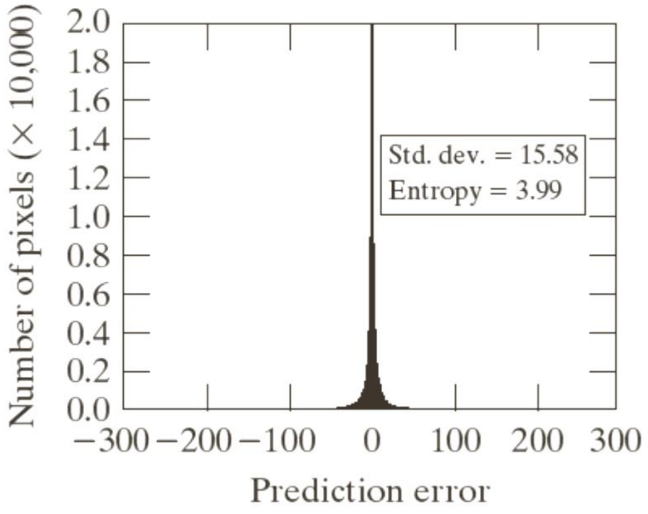


Predictive Coding - Lossless

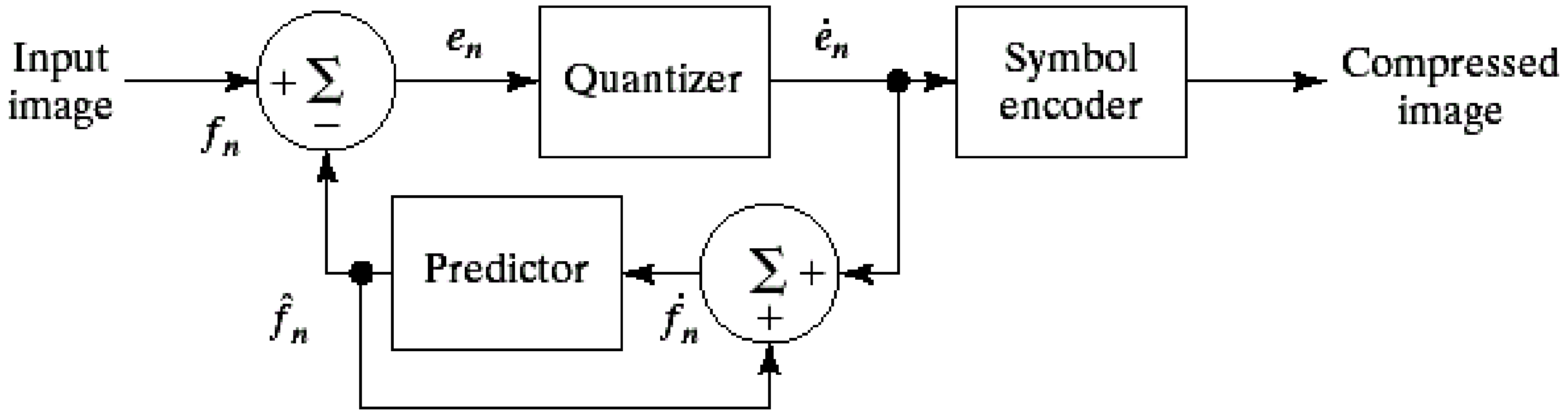
$$\hat{f}(x,y) = f(x,y-1)$$

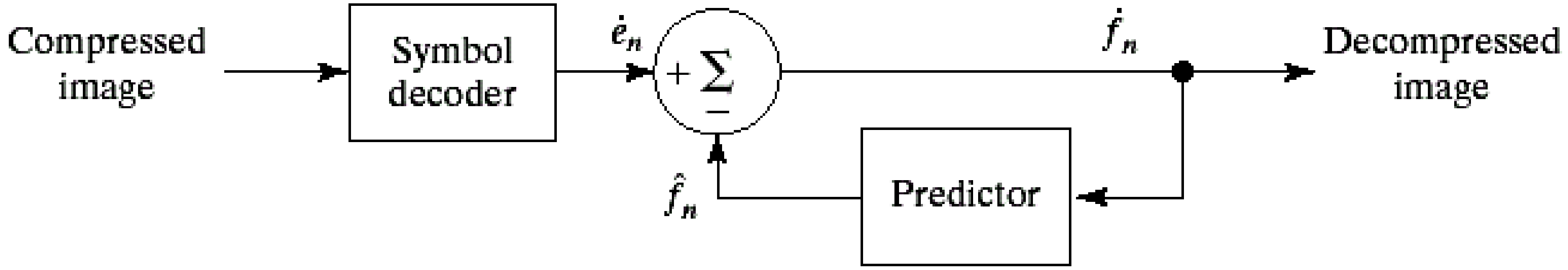






Predictive Coding - Lossy





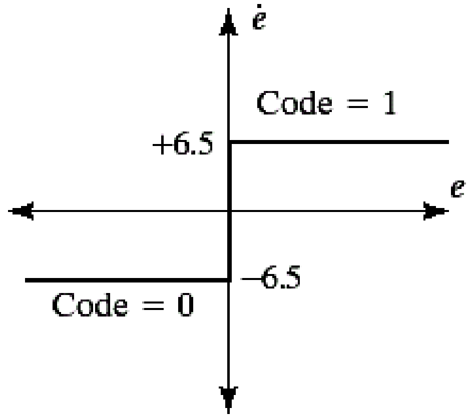
Predictive Coding - Lossy

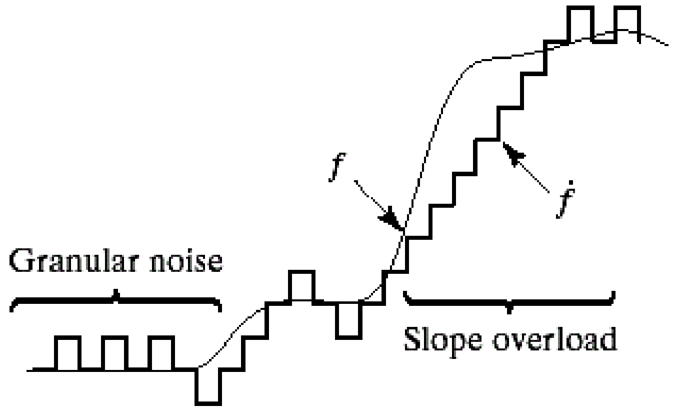
Example: Delta modulation

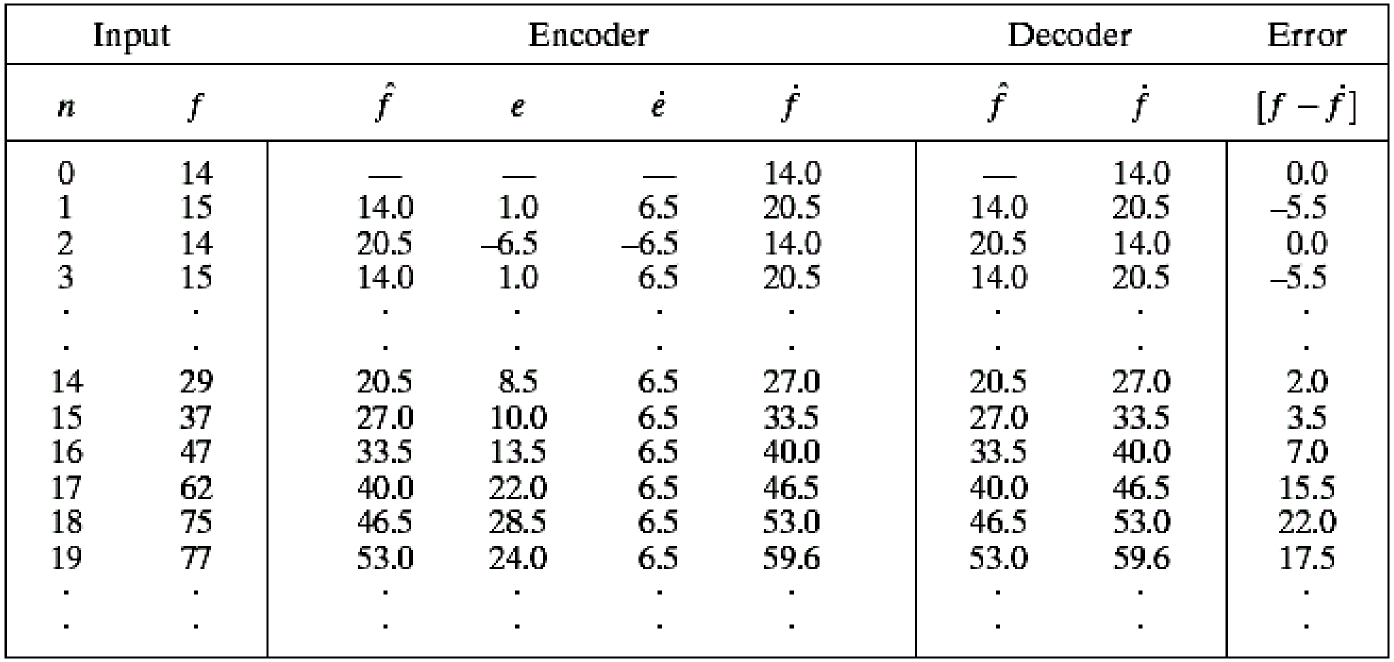
$$\hat{f}_n = \alpha\, f_{n-1}$$

$$(\alpha \le 1)$$

$$\dot{e}_n = \begin{cases} +\xi & e_n > 0 \\ -\xi & \text{otherwise} \end{cases}$$



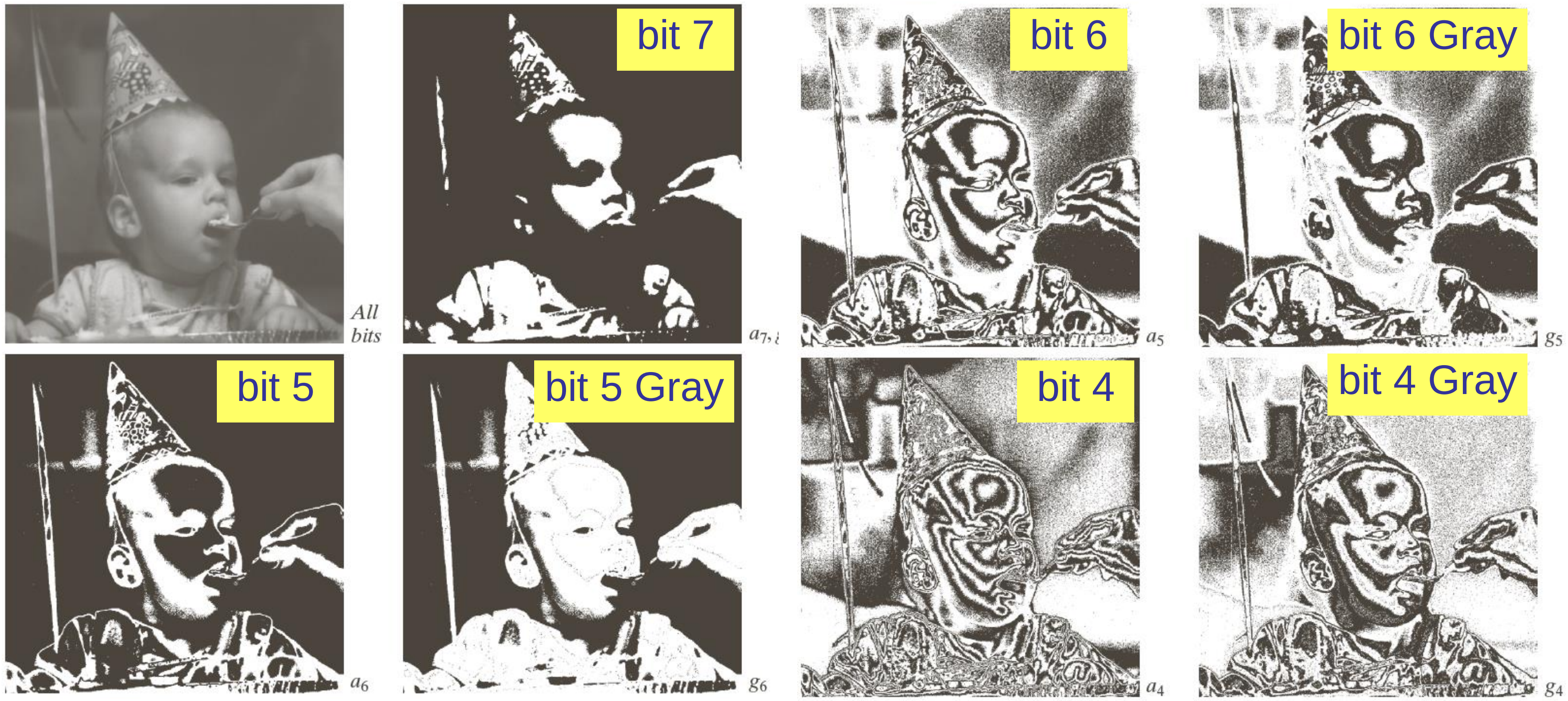




Input		Encoder				Decoder		Error
n	f	$\hat{f}$	e	$\dot{e}$	$\dot{f}$	$\hat{f}$	$\dot{f}$	$[f-\dot{f}]$
0	14	—	—	—	14.0	—	14.0	0.0
1	15	14.0	1.0	6.5	20.5	14.0	20.5	–5.5
2	14	20.5	–6.5	–6.5	14.0	20.5	14.0	0.0
3	15	14.0	1.0	6.5	20.5	14.0	20.5	–5.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
14	29	20.5	8.5	6.5	27.0	20.5	27.0	2.0
15	37	27.0	10.0	6.5	33.5	27.0	33.5	3.5
16	47	33.5	13.5	6.5	40.0	33.5	40.0	7.0
17	62	40.0	22.0	6.5	46.5	40.0	46.5	15.5
18	75	46.5	28.5	6.5	53.0	46.5	53.0	22.0
19	77	53.0	24.0	6.5	59.6	53.0	59.6	17.5
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Bit-Plane Coding

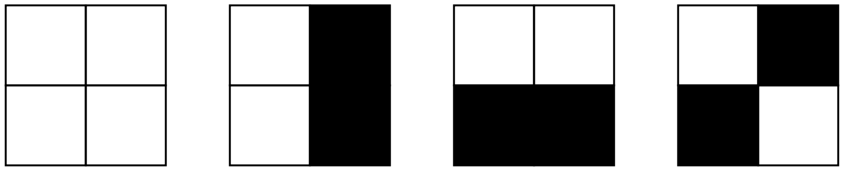
Comparisons of the four most significant binary (original) and Gray-coded bit plains.



Block Transform Coding

The transform projects a vector in the source (such as colors of all the pixels in a block) onto another set of basis vectors.

Example: A set of "basis blocks" for a 2x2 block



These are the basis blocks of the **Walsh-Hadamard Transform (WHT)** for 2x2 blocks. (White = +1/2, Black = -1/2)

One advantage of block transform coding is the easiness to implement in hardware.

Block Discrete Cosine Transform

The basis vectors (2-D):

$$r(x,y,u,v)=\alpha(u)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\alpha(v)\cos\left[\frac{(2y+1)v\pi}{2n}\right],$$

$$x,y,u,v=0,1,\ldots,n-1$$

with the normalization factors:

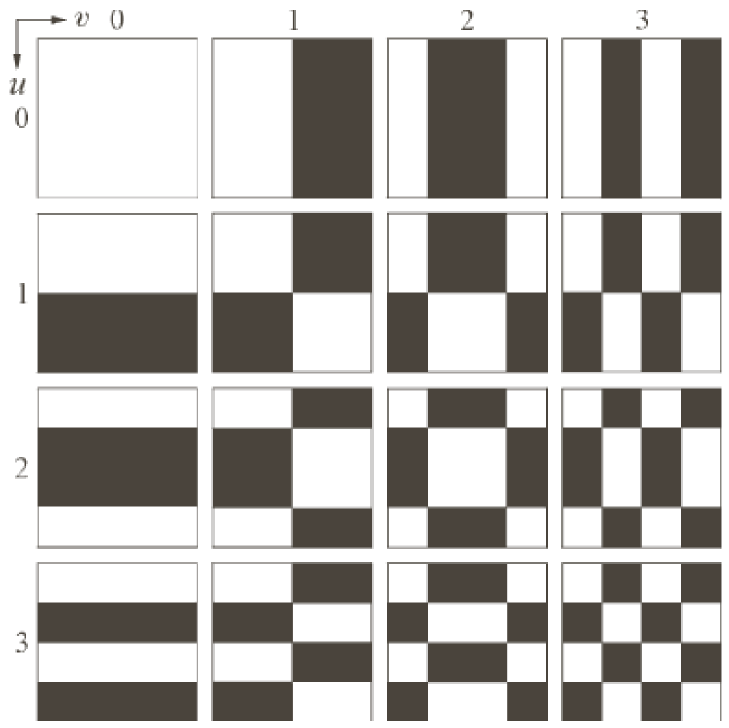
$$\alpha(u)=\begin{cases}\sqrt{1/n}, & u=0\\ \sqrt{2/n}, & \text{otherwise}\end{cases}$$

(similar for $\alpha(v)$)

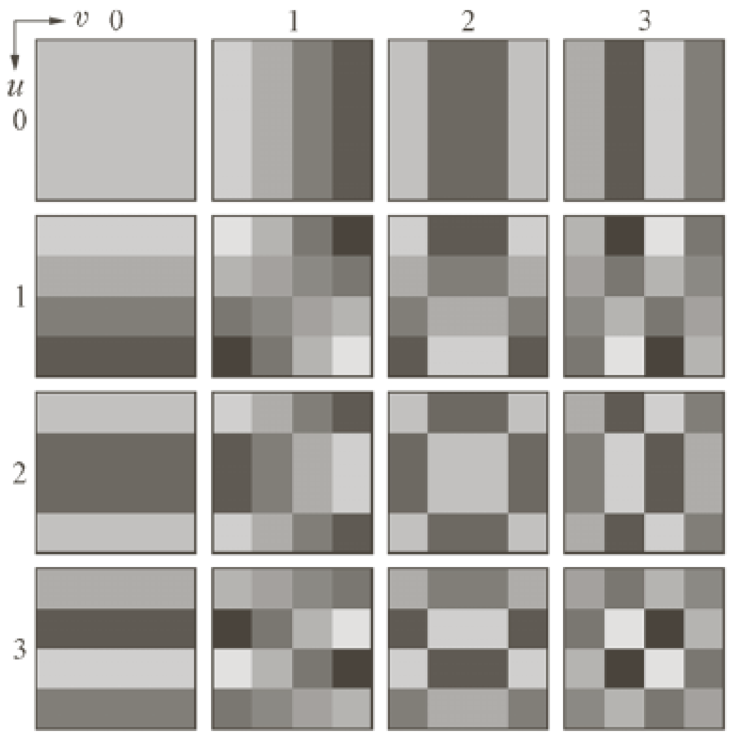
The first basis vector (u=v=0) is the DC part and the resulting coefficient the DC coefficient.

2-D Basis Blocks

$n = 4$



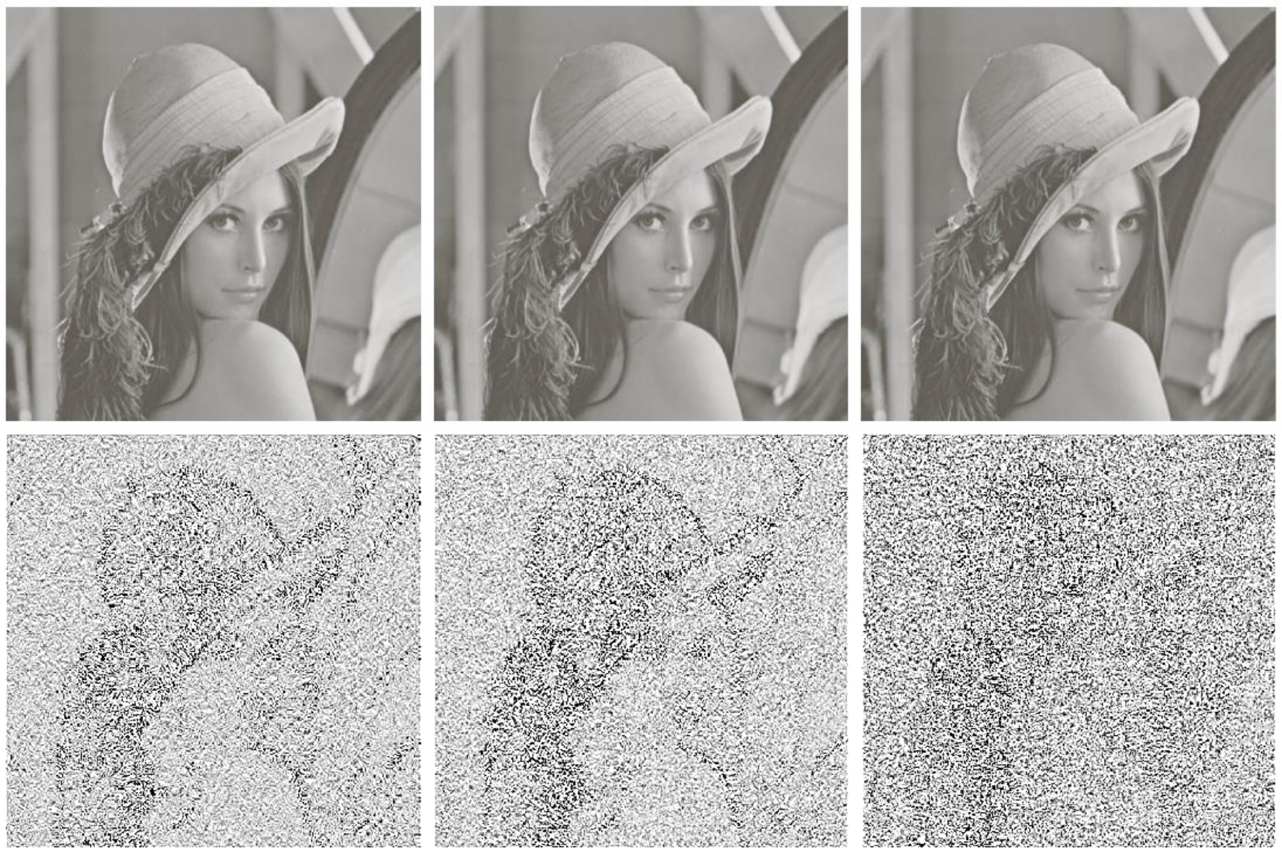
WHT



DCT

In general, coefficients closer to the origin (with smaller $u+v$, therefore lower spatial frequencies) are more important.

Comparing Transforms



DFT $e_{rms} = 2.32$ | WHT $e_{rms} = 1.78$ | DCT $e_{rms} = 1.13$

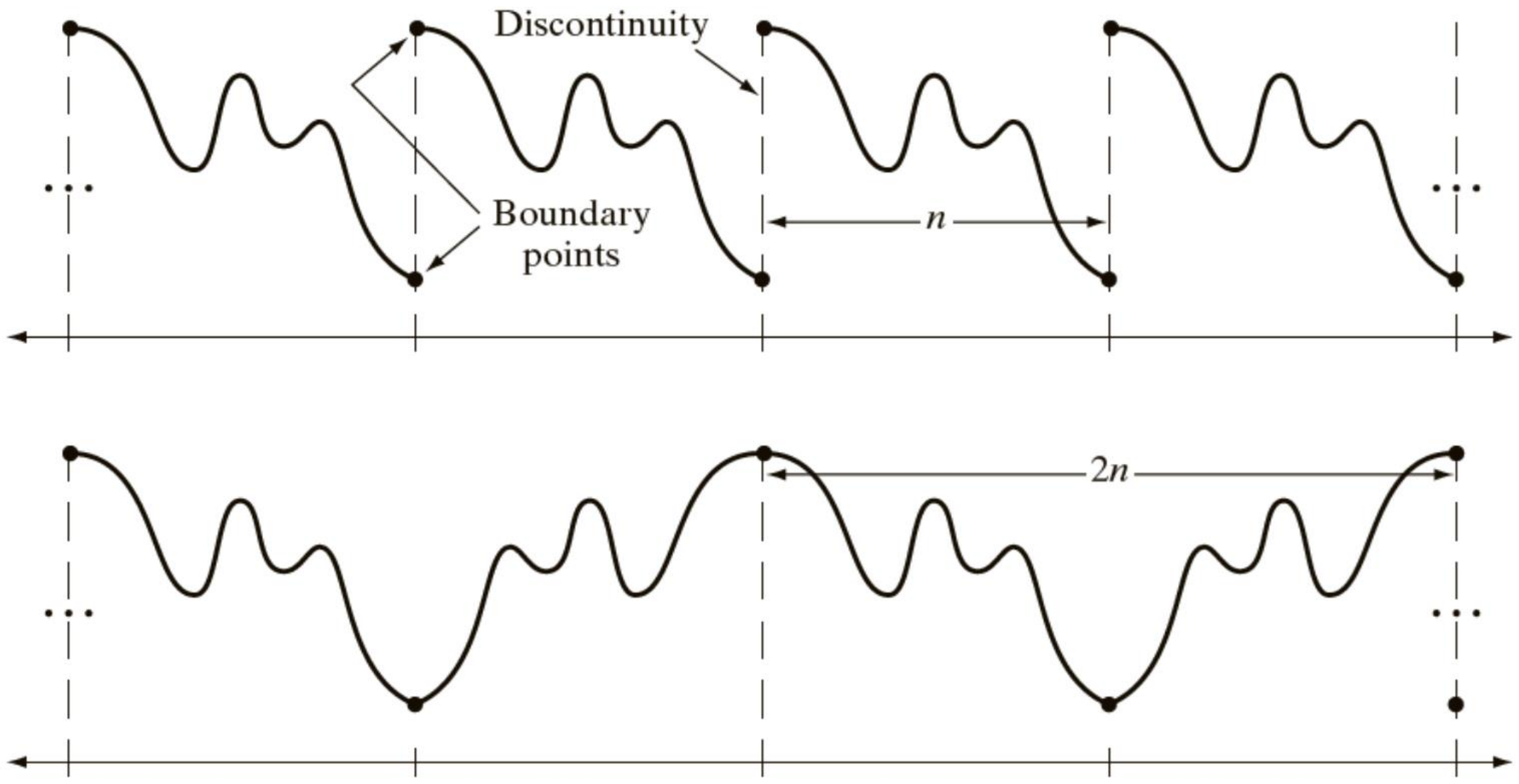
(Retaining the first 32 of the 64 coefficients.)

Comparing DFT and DCT

The implicit periodicity:

DFT

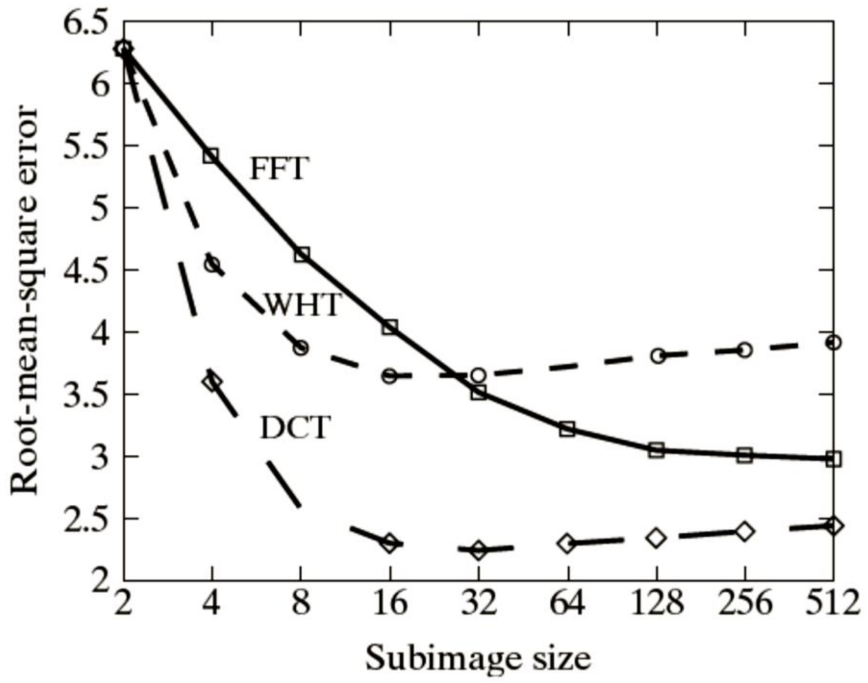
DCT

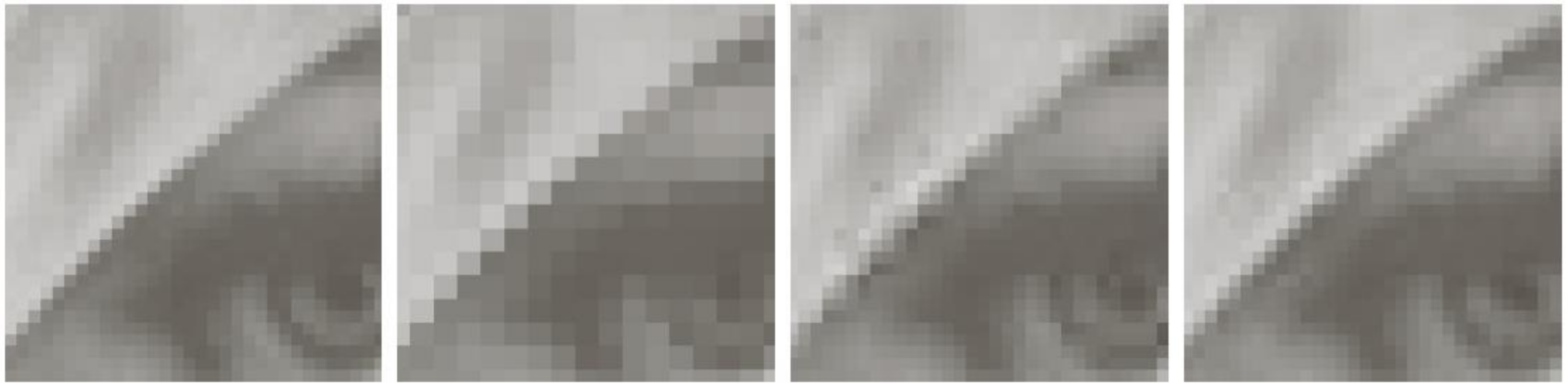


The discontinuities in DFT can cause undesired effects (wraparound errors).

Subimage (Block) Size Selection

- Examples here are based on retaining 25% of coefficients.
- 8 and 16 are the most commonly used block sizes in practice.



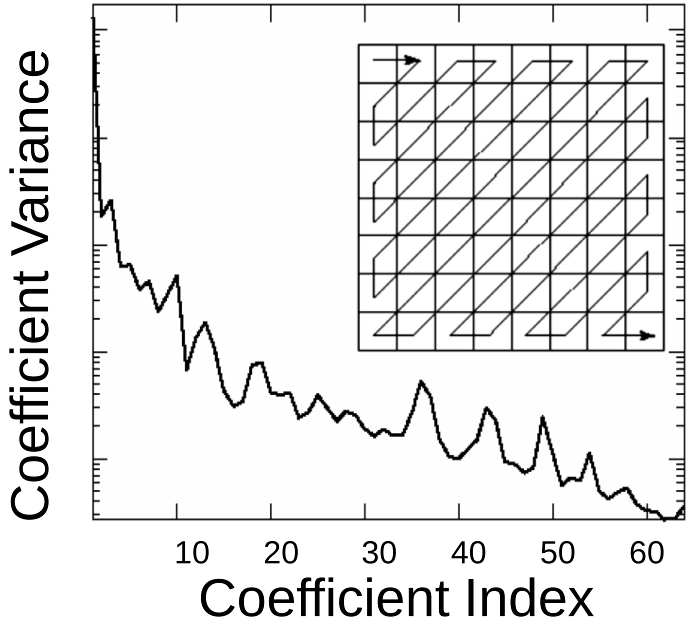


$n = 2$ $n = 4$ $n = 8$

Coefficient Quantization

- With transform coding, the quantization step is applied to the transform coefficients, not the original pixel values.
- The general idea is that the more important coefficients should be allocated more bits during quantization.
- Two main approaches:
  - Zonal coding: Allocate more bits for the coefficients with larger variances (computed over all the blocks); the same "zonal mask" for all the blocks.
  - Threshold coding: Retain only the coefficients that are sufficiently large; the retained coefficients vary from block to block.

Zonal Coding



A mask directly marking retained and discarded coefficients.

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

Zonal Bit Allocation

8	7	6	4	3	2	1	0
7	6	5	4	3	2	1	0
6	5	4	3	3	1	1	0
4	4	3	3	2	1	0	0
3	3	3	2	1	1	0	0
2	2	1	1	1	0	0	0
1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0

Threshold Coding

Three basic approaches:

- A single global threshold for each block.
- Retain a fixed number of coefficients for each block.
- A coefficient-dependent threshold (the same for all the blocks):
  - The method in JPEG
  - Thresholding and quantization handled in one step.

Coefficient Quantization in JPEG

- JPEG uses a quantization matrix which specifies the quantization step size for each coefficient.
- The quantization matrix ($Z(u,v)$) is the normalization matrix (example shown below) multiplied by a positive constant, which corresponds to the "quality factor".
- The normalization matrix is empirically determined based on their "psychovisual importance".

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

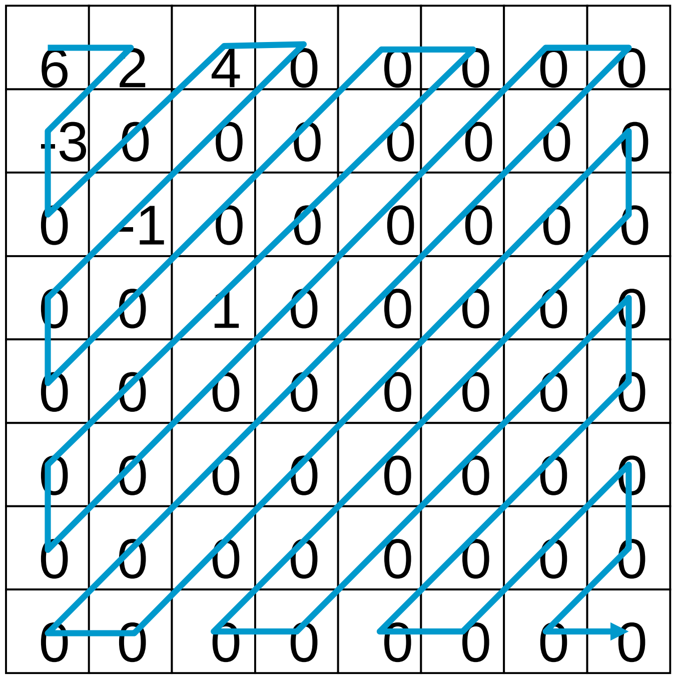
$$\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$$

JPEG Coder

For each block:

- Intensity shifting (e.g., to -128 to 127 for 8-bit images)
- DCT
- Coefficient quantization

The run-length coding of DCT coefficients, an example:



6	2	4	0	0	0	0	0
-3	0	0	0	0	0	0	0
0	-1	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
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

6 2 -3 004 00 -10 00000 000100 ...

6 (0,2) (0,-3) (2,4) (2,-1) (9,1) EOB

JPEG Coder

To complete the description of still-image coding...

6 (0,2) (0,-3) (2,4) (2,-1) (9,1) EOB

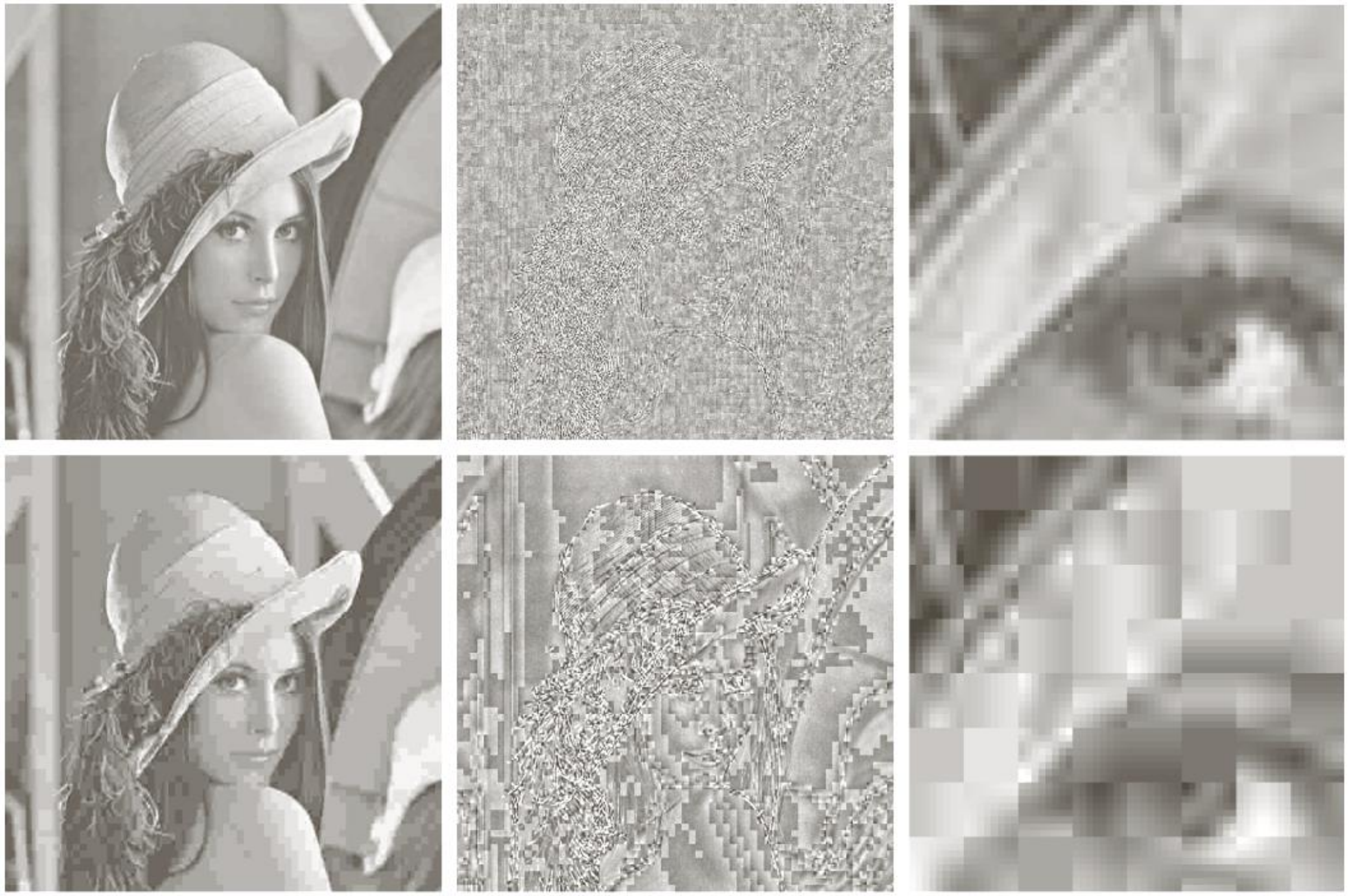
- DC term (the mean value of the block):
  - Difference coded from the previous block (predictive coding)
  - [Category → Huffman code] + [LSB]
- [run-length + AC] terms and EOB
  - [Category/Run → Huffman code] + [LSB]

JPEG

Example results:

25:1

52:1



JPEG for Color Images

- Decorrelation: RGB to YP_bP_r (floating-point YC_bC_r)

$$\begin{bmatrix} Y \\ P_b \\ P_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41896 & 0.08131 \end{bmatrix} \begin{bmatrix} \tilde{R} \\ \tilde{G} \\ \tilde{B} \end{bmatrix}$$

  - This is often used but not required, as this is irreversible due to rounding errors.

- JPEG allows chromatic subsampling. For example, chromatic components are sampled at half the resolution in both directions (4:2:0).