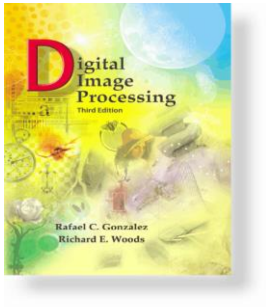


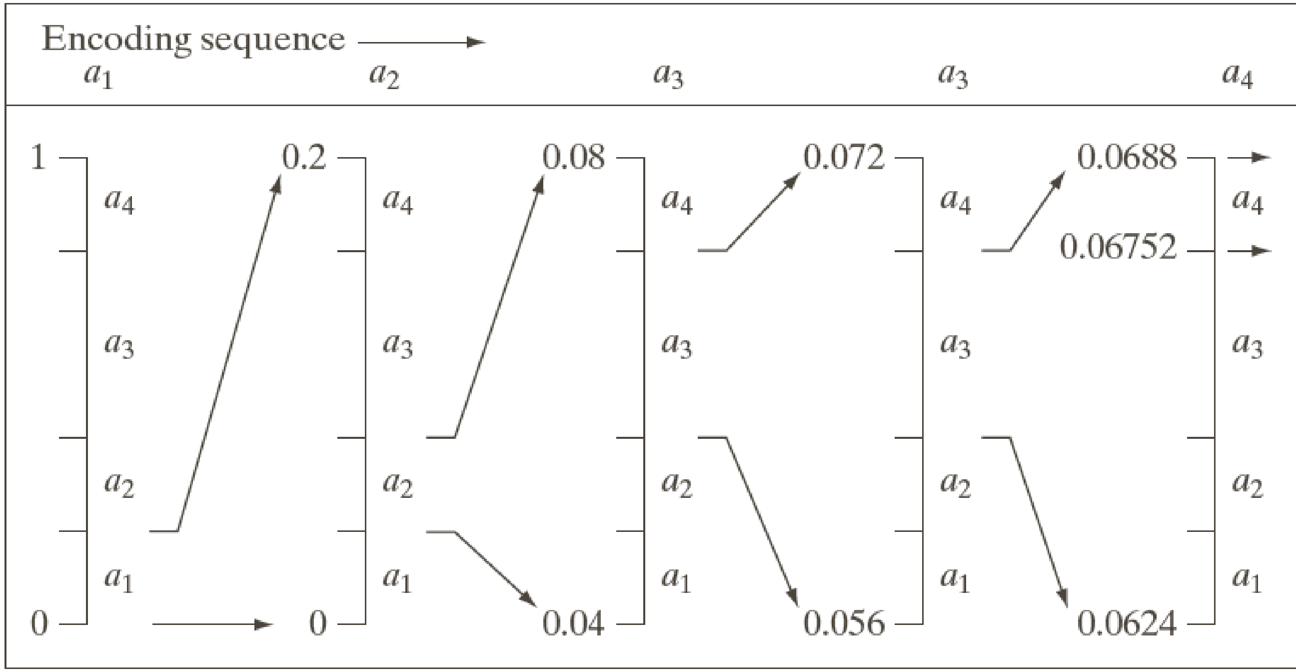
Image Compression

Arithmetic Coding

Source: $a_1a_2a_3a_3a_4$

Any number $\in$[0.06752, 0.0688) can be used to represent this message. Ex: 0.068

Three decimal digits for five symbols. So, 0.6 decimal digits per symbol.



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)

LZW Coding

Lempel-Ziv-Welch Coding

4 x 4, 8 bit image of a vertical edge: →

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

Original:
128-bit

New:
10 9-bit=90 bit

Run-Length Coding

Bit Map: BMP

Codes in every two bytes: first byte tells number of pixels of the value of second byte. .

In case, the first byte is 0: absolute mode: must be aligned to 16-bit word boundary.

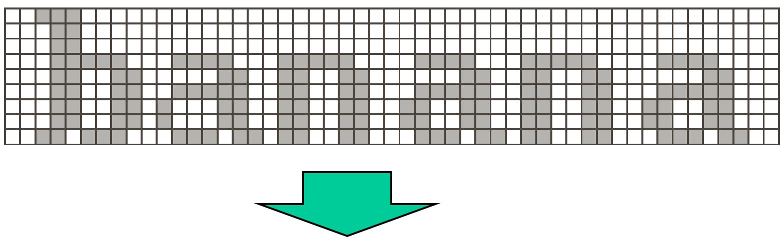
Second Byte Value	Condition
0	End of line
1	End of image
2	Move to a new position
3–255	Specify pixels individually

Decode:

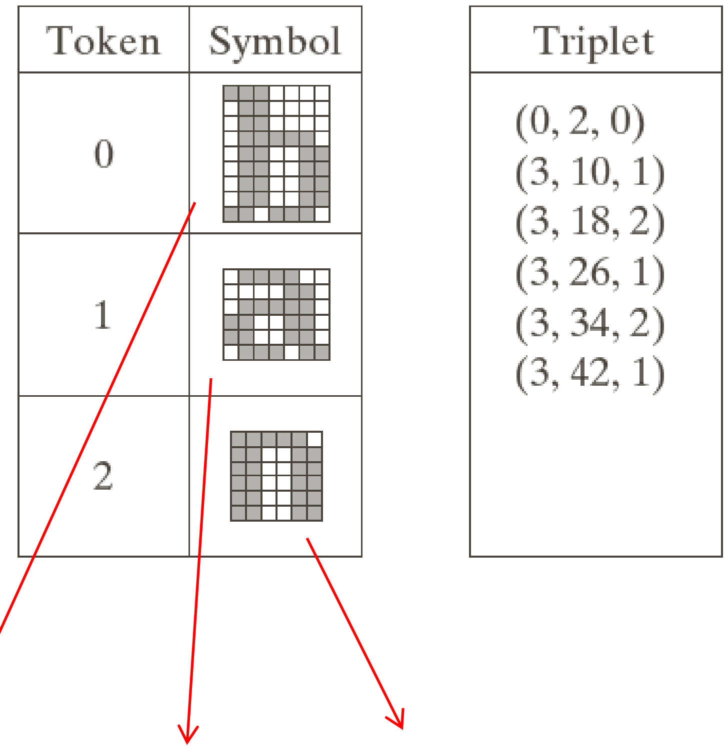
{3, 4, 5, 6, 0, 3, 103, 125, 67, 0, 2, 47}

{4, 4, 4, 6, 6, 6, 6, 6, 103, 125, 67, 47, 47}

Symbol-Based Coding



$9 \times 51 \times 1$ or 459 bits

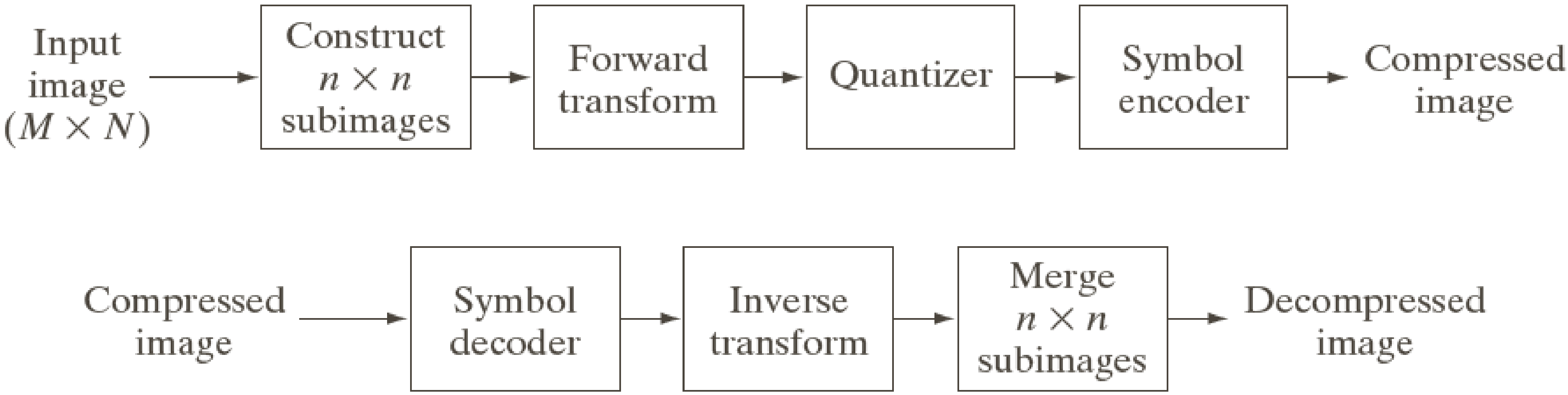


$(6 \times 3 \times 8)+[(9 \times 7)+(6 \times 7)+(6 \times 6)]$ or 285 bits

Each triplet has 3 bytes.

Compression ratio = 1.61

Block Transform Coding



Forward transform: $$T(u,v)=\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} g(x,y)\,r(x,y,u,v)$$

Inverse transform: $$g(x,y)=\sum_{u=0}^{n-1}\sum_{v=0}^{n-1} T(u,v)\,s(x,y,u,v)$$

Transformation - I

WHT: $$r(x,y,u,v) = s(x,y,u,v) = \frac{1}{n}(-1)^{\sum_{i=0}^{m-1}\left[b_i(x)p_i(u)+b_i(y)p_i(v)\right]}$$

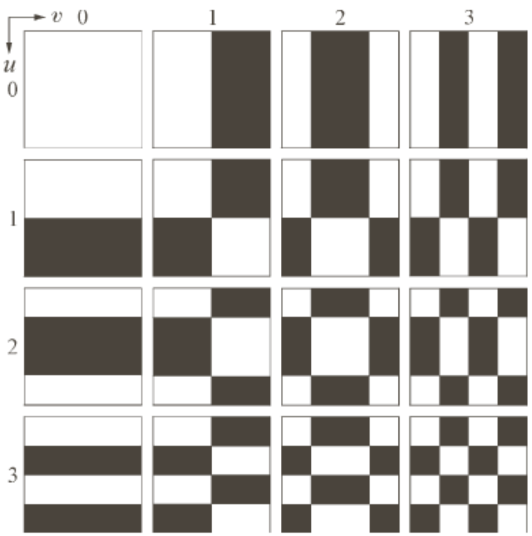
(Walsh-Hadamard Transform)

$$n = 2^m$$

$b_k(z)$ is the *k*th bit in the binary form of *z*.

Ex: If $m = 0$ and $z = 6$ (110), then $b_0(z) = 0$, $b_1(z)= 1$, $b_2(z) =1$.

$$p_0(u) = b_{m-1}(u)$$
$$p_1(u) = b_{m-1}(u) + b_{m-2}(u)$$
$$p_2(u) = b_{m-2}(u) + b_{m-3}(u)$$
$$\vdots$$
$$p_{m-1}(u) = b_1(u) + b_0(u)$$



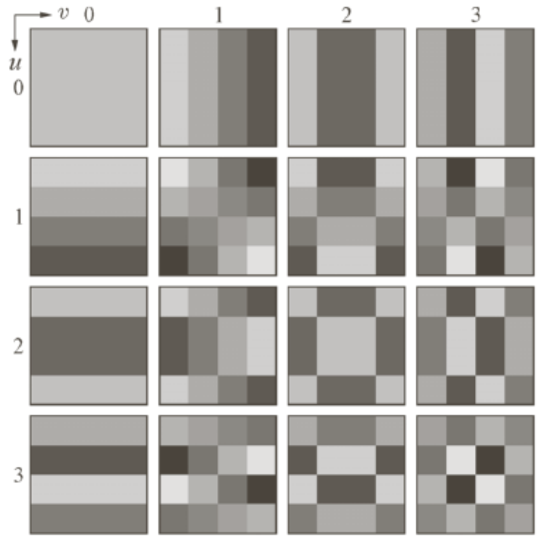
Kernel

Transformation - II

Discrete Cosine Transform (DCT)

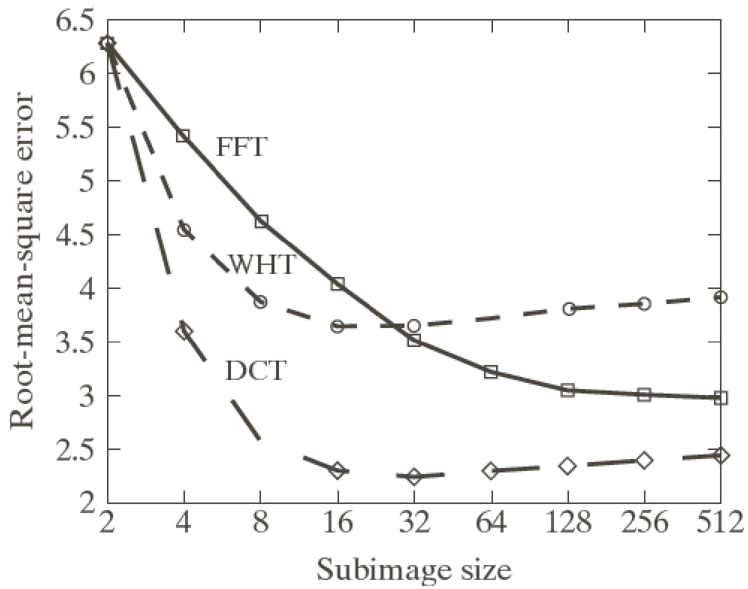
$$r(x,y,u,v) = s(x,y,u,v) = \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]$$

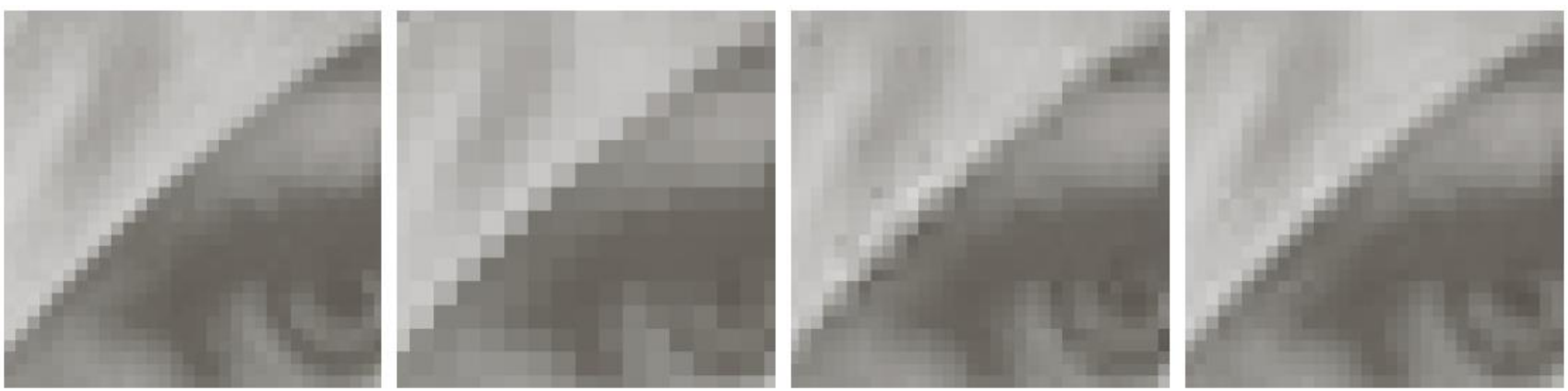
Where: $$\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & \text{for } u = 0 \\ \sqrt{\frac{2}{n}} & \text{for } u = 1,2,..,n-1 \end{cases}$$



Kernel

Subimage Size





a b c d

FIGURE 8.27 Approximations of Fig. 8.27(a) using 25% of the DCT coefficients and (b) 2 × 2 subimages, (c) 4 × 4 subimages, and (d) 8 × 8 subimages. The original image in (a) is a zoomed section of Fig. 8.9(a).

JPEG - I

Subimage:

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

256 or 2^8 possible intensities.

Shift by -128 or -2^7 .

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

JPEG - II

Applying DCT with n = 8.

−415	−29	−62	25	55	−20	−1	3
7	−21	−62	9	11	−7	−6	6
−46	8	77	−25	−30	10	7	−5
−50	13	35	−15	−9	6	0	3
11	−8	−13	−2	−1	1	−4	1
−10	1	3	−3	−1	0	2	−1
−4	−1	2	−1	2	−3	1	−2
−1	−1	−1	−2	−1	−1	0	−1

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

Normalization matrix

Round(-415/16)

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	1	2	−1	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

After normalization

JPEG - III

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Ordering sequence

Resulting 1-D coefficient sequence:

[-26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 -1 -1 EOB]

DC coefficient = -26.
Let, DC coefficient of its immediate left subimage is -17.
Difference is -9 → category 4. [Table A.3]

Base code 101 (3-bit) and total length = 7 bits. [Table A.4]

For *K* category, *K* additional bit → *K* LSB for +difference

→ *K* LSB for –difference - 1

For, difference = -9:
(0111) - 1 = 0110.

Complete DC code: 1010110

JPEG - IV

First nonzero coefficient = -3 → category 2 (code 01), length = 4

Last two bits (00) are obtained by DC diff. code.

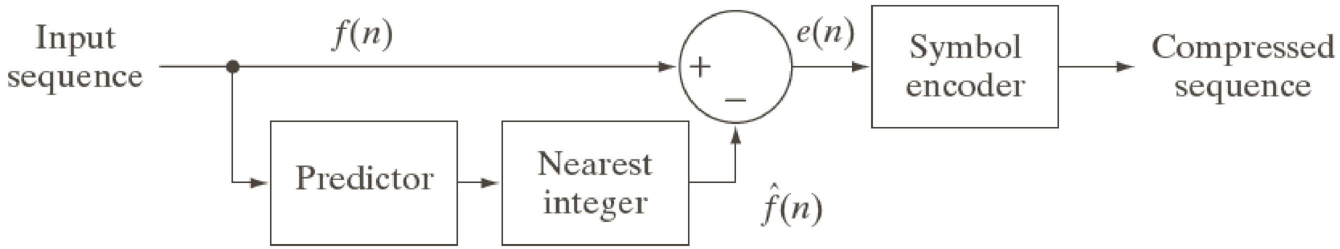
Use the reverse direction for decompression.

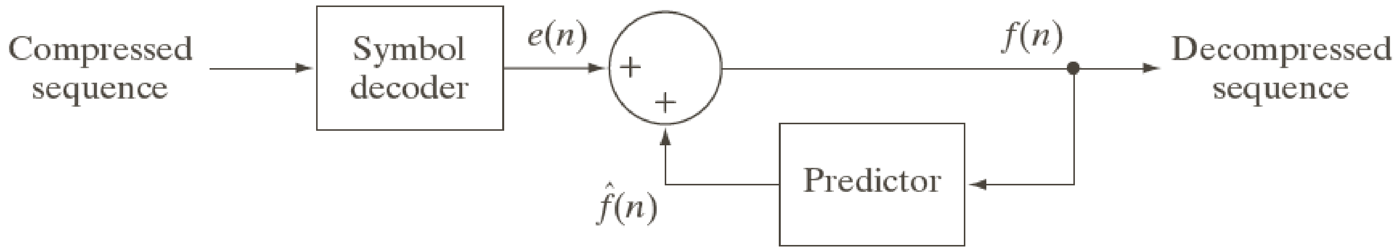
Predictive Coding - I

Linear predictor: $$\hat{f}(n) = round\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$

m = order of the linear predictor, α_i : prediction coefficients.

Prediction error: $$e(n) = f(n) - \hat{f}(n)$$

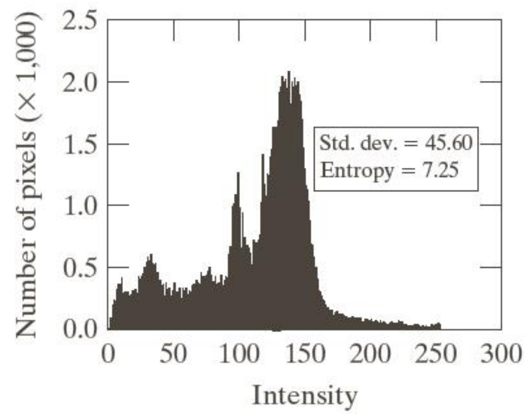




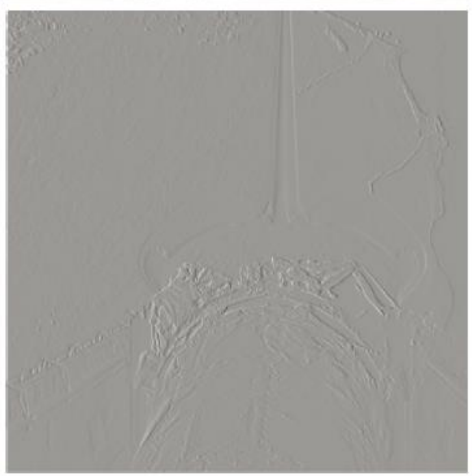
Predictive Coding - II

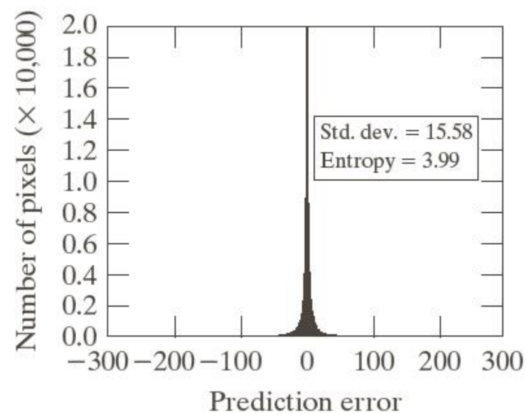
First order linear predictor: $\hat{f}(x,y) = round\left[\alpha f(x, y-1)\right]$

Original image



Prediction error image





SD is less.
Entropy is less

Digital Image Watermarking

Inserting information (watermark) into images in such a way that the watermark is inseparable from the images.

- Copyright identification.
- User identification.
- Authenticity determination.
- Automated monitoring.
- Copy protection.

$$f_w = (1 - \alpha) f + \alpha w$$

Watermarked image

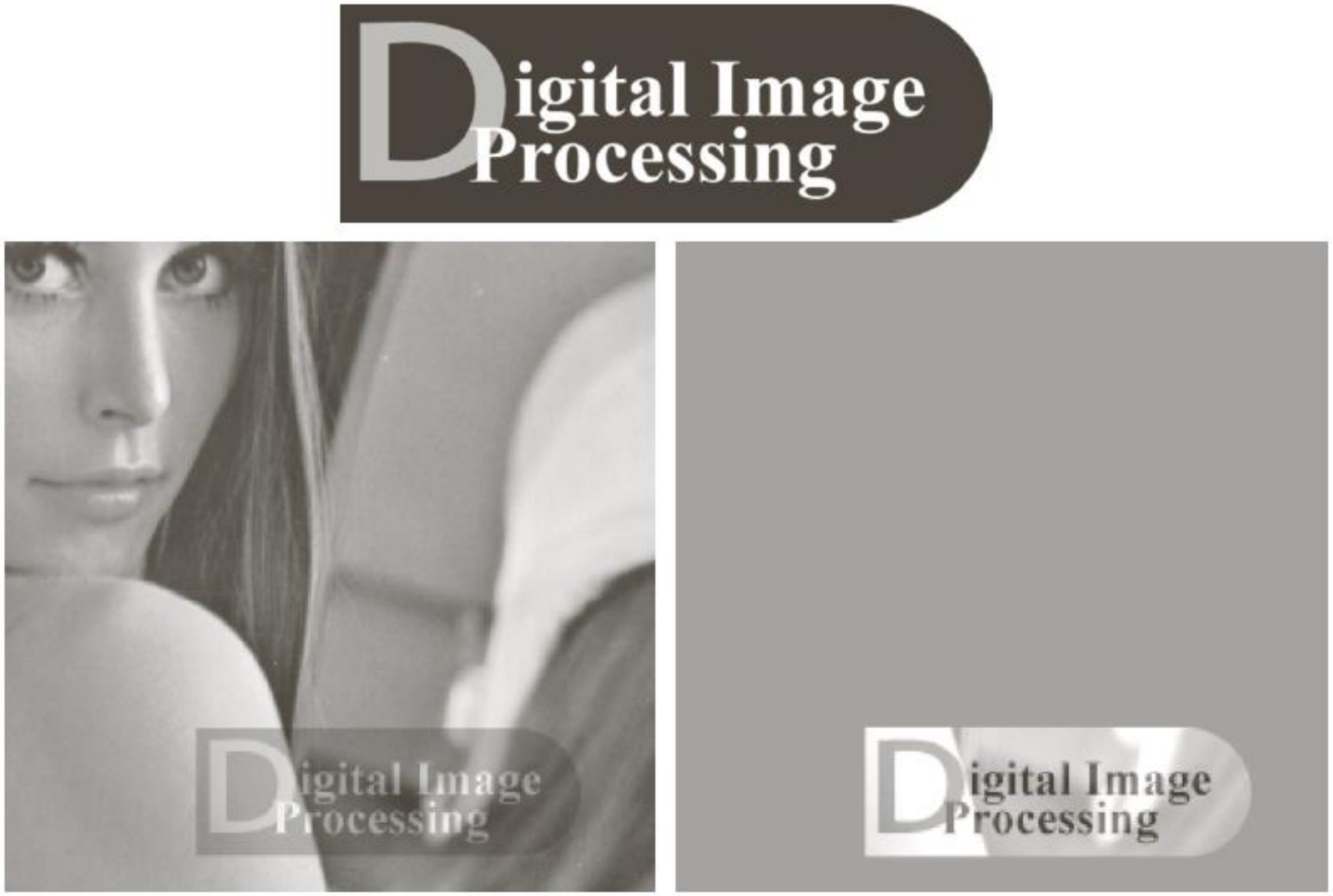
Original image

Watermark

Controls relative visibility of watermark and image

Example of Watermarking - I

Visible



a
b c

FIGURE 8.50
A simple visible watermark: (a) watermark; (b) the watermarked image; and (c) the difference between the watermarked image and the original (non-watermarked) image.

$\alpha = 0.3$

Example of Watermarking - II

Invisible

Watermark is inserted in image's two LSBs.

$$f_w = 4\left(\frac{f}{4}\right) + \frac{w}{64}$$

Sets two LSBs of image to 00.

Shifts two MSBs into two LSBs.

By zeroing 6 MSB and scaling the remains to full intensity level, we get

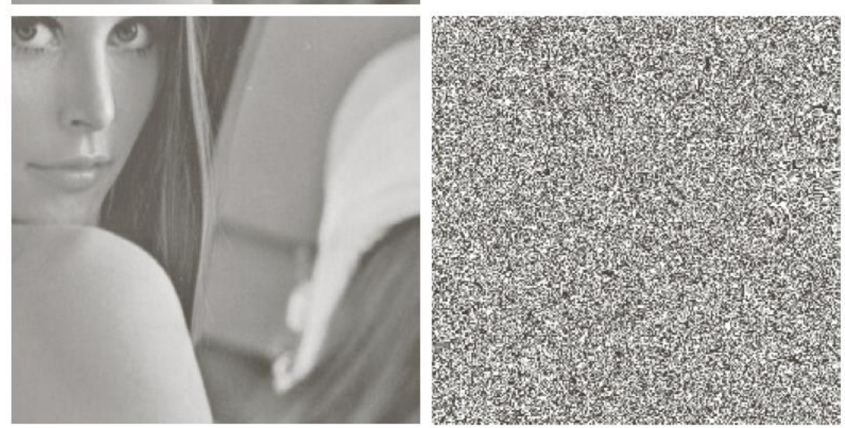
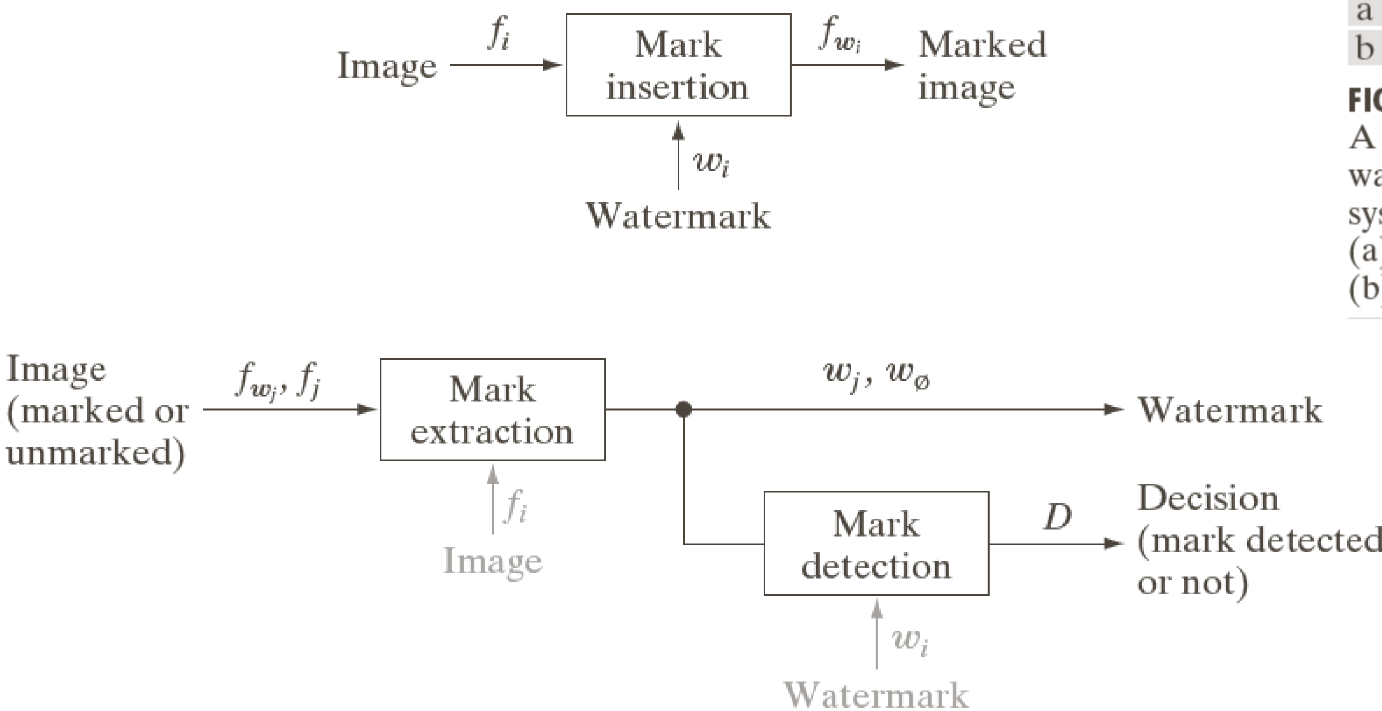


Image Watermarking System



a
b

FIGURE 8.52 A typical image watermarking system: (a) encoder; (b) decoder.

Private / restricted key system: f_i and w_i are used.

Public / unrestricted key system: f_i and w_i are unused.

DCT-Based Invisible Robust Watermark

Step 1: Compute DCT of the image to be watermarked.

Step 2: Locate the *K* largest coefficients $c_1, c_2, \ldots, c_k$ by magnitude.

Step 3: Generate a watermark with *K* elements pseudo-random numbers $w_1, w_2, \ldots, w_k$ from a Gaussian distribution with *mean* = 0 and *variance* = 1.

Step 4: Embed the watermark into the image using the following equation:

$$c_i' = c_i \bullet (1 + \alpha w_i) \qquad 1 \le i \le K$$

Step 5: Compute inverse DCT of the result from step 4.

DCT-Based Watermark: Why Robust?

1. Watermark has no obvious structure, (pseudo-random).

2. Watermark is embedded in multiple frequency components. So we cannot know its location.

3. Attacks against watermark tend to degrade the image as well (most important frequency components must be altered).

Checking an Image:

1. DCT (Image).
2. Extract c'_k, k = 1,2…K. If $c_k = c'_k$, no change; if $c_k \approx c'_k$, attack.
3. Compute $w'_i = c'_i - c_i$.
4. Compute correlation between w_i and w'_i.
5. Take a decision.