

Digital Image Processing

Image Compression and Watermarking (II)

Dr. Tun-Wen Pai

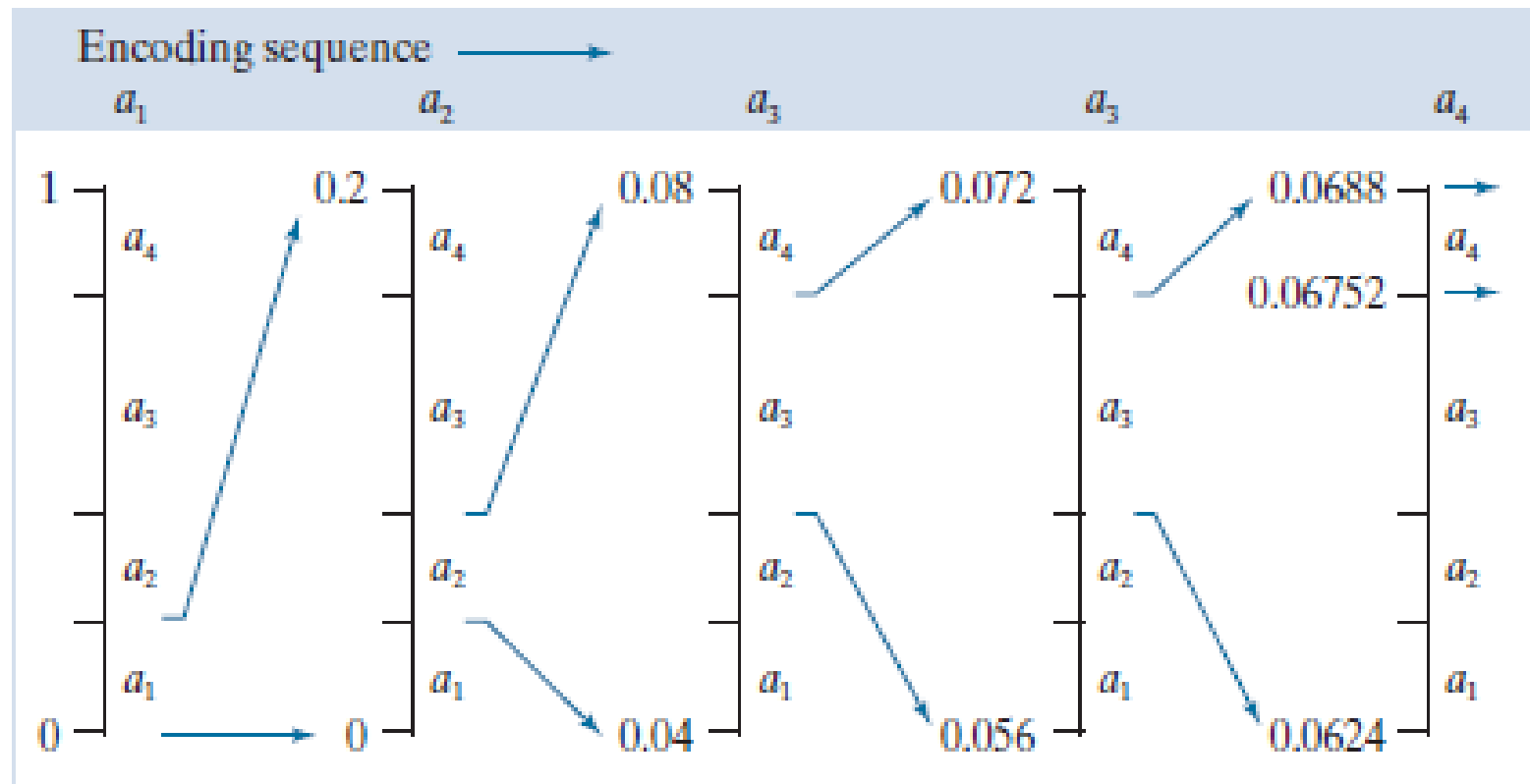
- 1) Some other coding techniques**
- 2) Understand transform coding**
- 3) Walsh Transform / Discrete Cosine Transform**
- 4) JPEG compression**
- 5) Bit Allocation/Zonal Coding/Threshold Coding**

TABLE 8.7

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)

Arithmetic coding procedure.



Elias (1963)

TABLE 8.7
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	$[0.8, 1.0)$

generating nonblock codes
 → entire sequence of source symbols (messages)
 → assigned a single arithmetic code word.

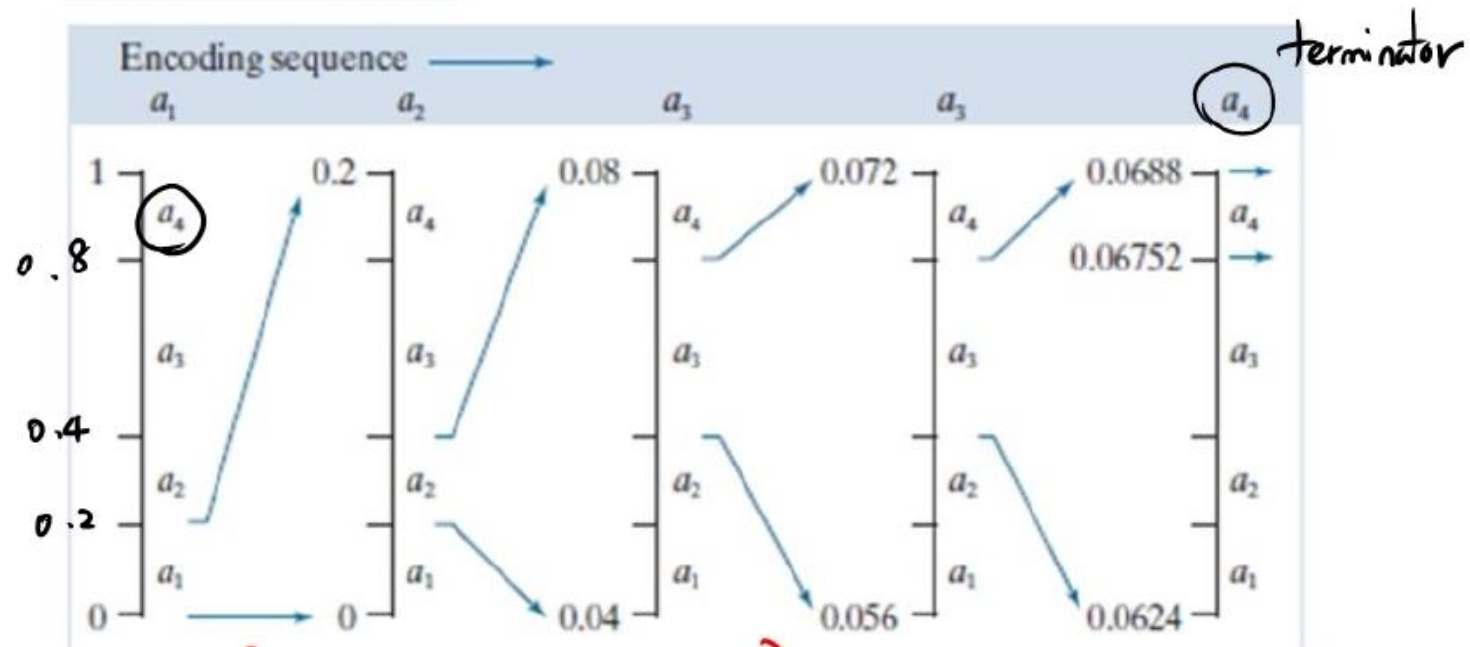
ex) $a_1 a_2 a_3 a_4 \rightarrow 0.068$

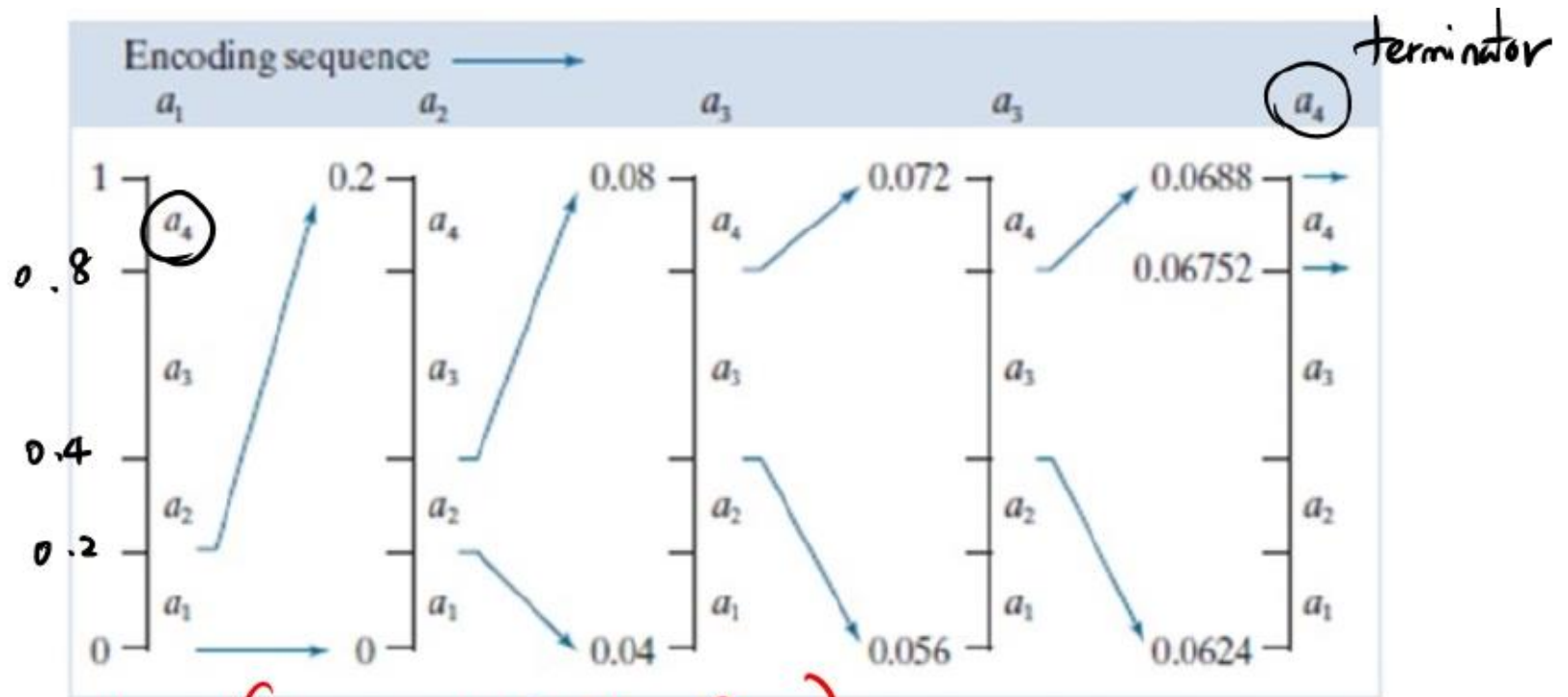
Notes:

- (1) the addition of the end-of-message indicator (termination char) is needed to separate one message from another
- (2) the use of finite precision arithmetic (scaling strategy & rounding strategy)

message
 $\Rightarrow a_1 a_2 a_3 a_3 a_4$

FIGURE 8.12
Arithmetic coding
procedure.





encoding. (Low = 0, High = 1, loop)

loop {

- Range = High - Low
- High = Low + range * high-range of the symbol being coded
- Low = Low + range * low-range of " " " "

$$a_3: [0.056, 0.072) \rightarrow \text{range} = 0.072 - 0.056 = 0.016$$

$$\text{High} = 0.056 + 0.016 \times 0.8 = 0.0688$$

$$\text{Low} = 0.056 + 0.016 \times 0.4 = 0.0624$$

$$a_4: [0.0624, 0.0688) \rightarrow \text{range} = 0.0688 - 0.0624 = 0.0064$$

$$\text{High} = 0.0624 + 0.0064 \times 1 = 0.0688$$

$$\text{Low} = 0.0624 + 0.0064 \times 0.8 = 0.06752$$

$$\Rightarrow a_1 a_2 a_3 a_4 \rightarrow [0.06752, 0.0688)$$

any number in this interval is OK.

$$\Rightarrow \boxed{0.068} \quad (\text{small \# of decimal digit is better})$$

Decode loop.

$$\left\{ \begin{array}{l} \bullet \text{ range} = \text{high} - \text{range of the symbol} - \text{Low-range of the symbol} \\ \bullet \text{ number} = \text{number} - \text{Low-range of the symbol} \\ \bullet \text{ number} = \text{number} / \text{range} \end{array} \right.$$

Decode loop.

- $\text{range} = \text{high_range of the symbol} - \text{Low_range of the symbol}$
- $\text{number} = \text{number} - \text{Low_range of the symbol}$
- $\text{number} = \text{number} / \text{range}$

$$0.068 \Rightarrow 0.068 \in [0, 0.2) \Rightarrow a_1$$

$$\text{range} = 0.2 - 0.0 = 0.2$$

$$\text{number} = 0.068 - 0 = 0.068,$$

$$\text{number} = 0.068 / 0.2 = 0.34$$

$$0.34 \in [0.2, 0.4) \Rightarrow a_2$$

$$\text{range} = 0.4 - 0.2 = 0.2$$

$$\text{number} = 0.34 - 0.2 = 0.14$$

$$\text{number} = 0.14 / 0.2 = 0.7$$

$$0.7 \in [0.4, 0.8) \Rightarrow a_3$$

$$\text{range} = 0.8 - 0.4 = 0.4$$

$$\text{number} = 0.7 - 0.4 = 0.3$$

$$\text{number} = 0.3 / 0.4 = 0.75$$

$$0.75 \in [0.4, 0.8) \Rightarrow a_3$$

$$\text{range} = 0.8 - 0.4 = 0.4$$

$$\text{number} = 0.75 - 0.4 = 0.35$$

$$\text{number} = 0.35 / 0.4 = 0.875$$

$$0.875 \in [0.8, 1) \Rightarrow a_4$$

terminated!

TABLE 8.8
LZW Coding
example.

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		

Dictionary-Based Compression

- This family of algorithm does not encode single symbols as bit streams, instead they encode phrases of variable length as single tokens.
- LZ77 (1977 by Ziv and Lempel)
- LZW (Lempel-Ziv-Welch) by Terry Welch in 1984 – IBM patent(USA).
- Fixed-length codes for variable symbols/codewords
- Dynamic dictionary table construction for both encoding/decoding sides
- Good compression for long/repeat contents

LZW encoding algorithm

BEGIN

 s = next input character;

 while not EOF

 { c = next input character;

 if s + c exists in the dictionary

 s = s + c;

 else

 { output the code for s;

 add string s + c to the dictionary with a new code;

 s = c;

 }

 }

 output the code for s;

END

- ABABBABCABABBA

string table

code	string
1	A
2	B
3	C

1	A
2	B
3	C

Encoded output

1 2 4 5 2 3 4 6 1

Required 9 characters

Instead of 14 characters

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

- LZW Decoding algorithm

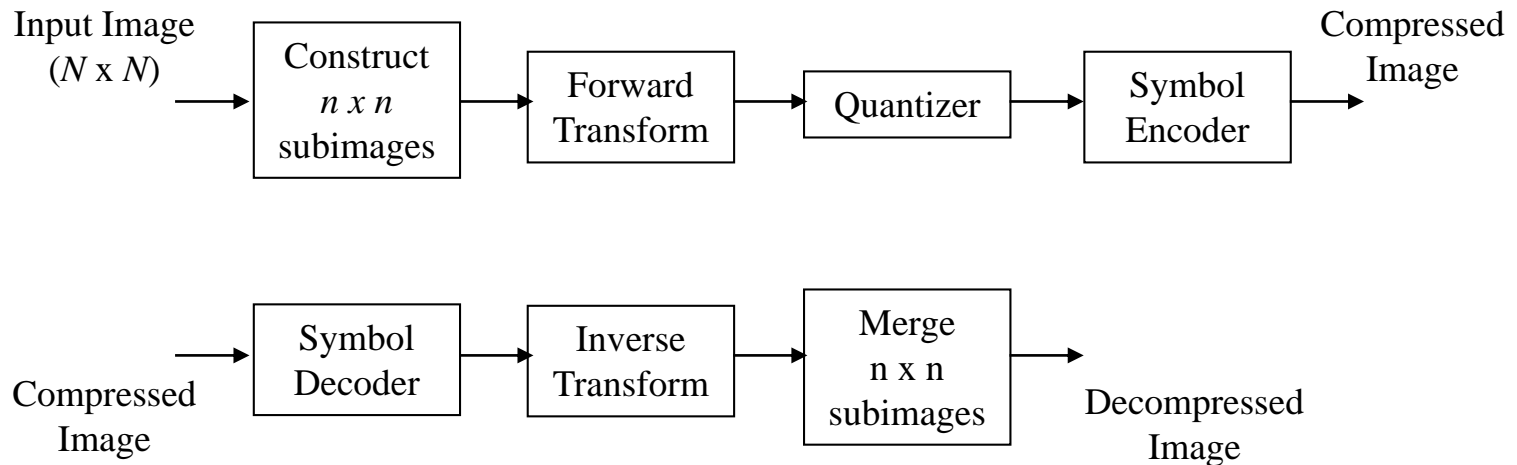
```
BEGIN
  s = NIL;
  while not EOF
    {
      k = next input code;
      entry = dictionary entry for k;
      output entry;
      if (s != NIL)
        add string s + entry[0] to dictionary with a new code;
      s = entry;
    }
  END
```


s	k	entry/output	code	string
<hr/>				
			1	A
			2	B
			3	C
<hr/>				
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

Decoded output: ABABBABCABABBA

Transform Coding

- Predictive coding technique is a spatial domain technique since it operates on the pixel values directly.
- Transform coding techniques operate on a reversible linear transform coefficients of the image (ex. DCT, DFT, Walsh etc.)



- Input $N \times N$ image is subdivided into subimages of size $n \times n$.
- $n \times n$ subimages are converted into transform arrays. This tends to decorrelate pixel values and pack as much information as possible in the smallest number of coefficients.
- Quantizer selectively eliminates or coarsely quantizes the coefficients with least information.
- Symbol encoder use a variable-length code to encode the quantized coefficients.
- Any of the above steps can be adapted to each subimage (adaptive transform coding), based on local image information, or fixed for all subimages.

Walsh Transform (1-D case)

- Given a one-dimensional image (sequence) $f(m)$, $m = 0, 1, \dots, N-1$, with $N = 2^q$, its **Walsh transform** $W(u)$ is defined as:

$$W(u) = \frac{1}{N} \sum_{m=0}^{N-1} f(m) \prod_{t=0}^{q-1} (-1)^{b_t(n)b_{q-1-t}(u)}, \quad u = 0, 1, \dots, N-1$$

$b_k(z)$: k^{th} bit (from LSB) in the binary representation of z .

- Note that the Walsh-Hadamard transform discussed in the text is very similar to the Walsh transform defined above.
- Example: Suppose $z = 6 = 110$ in binary representation. Then $b_0(6) = 0$, $b_1(6) = 1$ and $b_2(6) = 1$.
- The **inverse Walsh transform** of $W(u)$ is given by

$$f(m) = \sum_{u=0}^{N-1} W(u) \prod_{t=0}^{q-1} (-1)^{b_t(m)b_{q-1-t}(u)}, \quad m = 0, 1, \dots, N-1$$

- Verify that the “inverse works.” Let

$$\begin{aligned} g(m) &= \sum_{u=0}^{N-1} \left[\underbrace{\frac{1}{N} \sum_{n=0}^{N-1} f(n) \prod_{t=0}^{q-1} (-1)^{b_t(n)b_{q-1-t}(u)}}_{\text{This is } W(u) \text{ with } m \text{ replaced with } n.} \right] \prod_{t=0}^{q-1} (-1)^{b_t(m)b_{q-1-t}(u)} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} f(n) \sum_{u=0}^{N-1} \prod_{t=0}^{q-1} (-1)^{(b_t(n)+b_t(m))b_{q-1-t}(u)} \end{aligned}$$

Walsh Transform (2-D case)

- Given a two-dimensional $N \times N$ image $f(m, n)$, with $N = 2^q$, its **Walsh transform** $W(u, v)$ is defined as:

$$W(u, v) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) \prod_{t=0}^{q-1} (-1)^{b_t(n)b_{q-1-t}(u) + b_t(n)b_{q-1-t}(v)}$$

$u, v = 0, 1, \dots, N-1$.

- Similarly, the **inverse Walsh transform** is given by

$$f(m, n) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} W(u, v) \prod_{t=0}^{q-1} (-1)^{b_t(m)b_{q-1-t}(u) + b_t(n)b_{q-1-t}(v)}$$

$m, n = 0, 1, \dots, N-1$

- The Walsh transform is
 - Separable (can perform 2-D transform in terms of 1-D transform).
 - Symmetric (the operations on the variables m, n are identical).
 - Forward and inverse transforms are identical.
 - Involves no trigonometric functions (just + 1 and -1), so is computationally simpler.

Discrete Cosine Transform (DCT)

- Given a two-dimensional $N \times N$ image $f(m, n)$, its **discrete cosine transform (DCT)** $C(u, v)$ is defined as:

$$C(u, v) = \alpha(u)\alpha(v) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) \cos\left[\frac{(2m+1)u\pi}{2N}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

$u, v = 0, 1, \dots, N-1$, where

$$\alpha(u) = \begin{cases} 1/\sqrt{N}, & u = 0 \\ \sqrt{2/N}, & u = 1, 2, \dots, N-1 \end{cases}$$

- Similarly, the inverse discrete cosine transform (IDCT) is given by

$$f(m, n) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u, v) \cos\left[\frac{(2m+1)u\pi}{2N}\right] \cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

$m, n = 0, 1, \dots, N-1$

- The DCT is
 - Separable (can perform 2-D transform in terms of 1-D transform).
 - Symmetric (the operations on the variables m, n are identical)
 - Forward and inverse transforms are identical
- The DCT is the most popular transform for image compression algorithms like JPEG (still images), MPEG (motion pictures).
- The more recent JPEG2000 standard uses wavelet transforms instead of DCT.

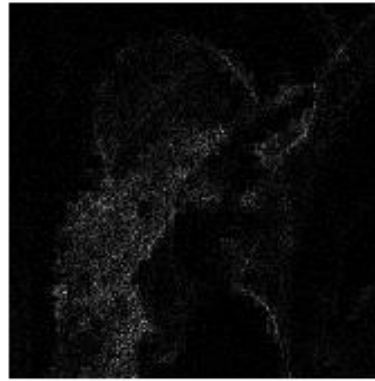
Transform Selection

- Commonly used ones are Karhunen-Loeve (Hotelling) transform (KLT), discrete cosine transform (DCT), discrete Fourier transform (DFT), Walsh-Hadamard transform (WHT).
- Choice depends on the computational resources available and the reconstruction error that can be tolerated.
- This step by itself is **lossless** and does not lead to compression. The quantization of the resulting coefficients results in compression.
- The KLT is optimum in terms of packing the most information for any given fixed number of coefficients.
- However, the KLT is data dependent. Computing it requires computing the correlation matrix of the image pixel values.
- The “non-sinusoidal” transforms like the WHT are easy to compute and implement (no multiplications and no trigonometric function evaluation).
- Performance of “sinusoidal” transforms like DCT, DFT, in terms of information packing capability, closely approximates that of the KLT.
- DCT is by far the most popular choice and is used in the **JPEG** (Joint Photographic Experts Group) image standard.

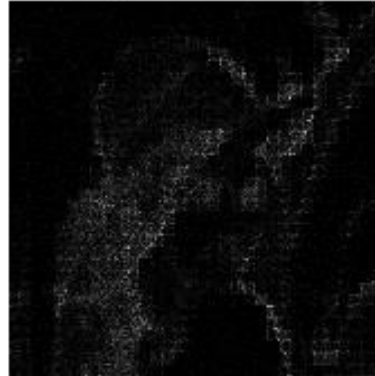
Reconstructed Image



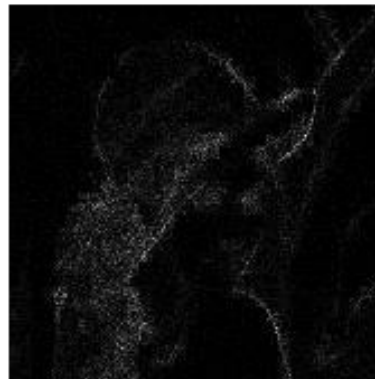
Error Image



DCT
RMSE = 0.018



DFT
RMSE = 0.028

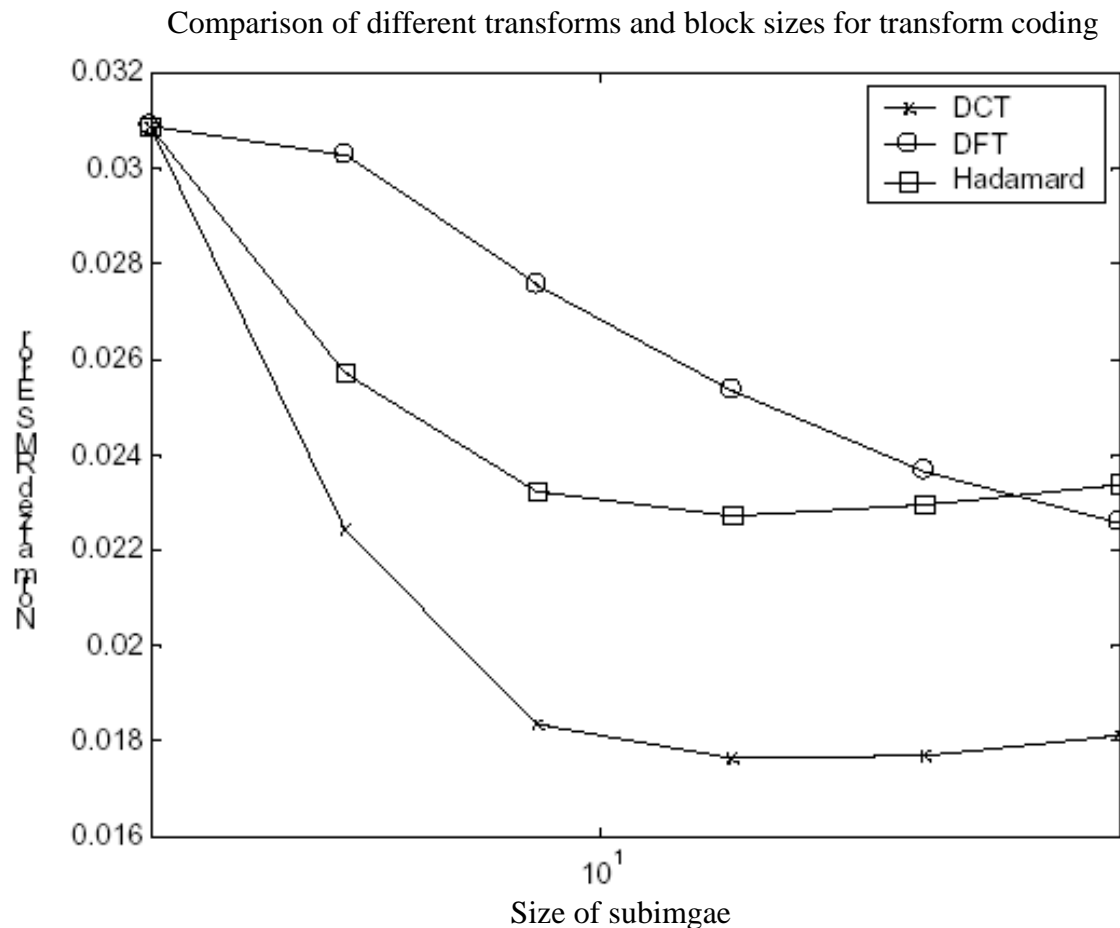


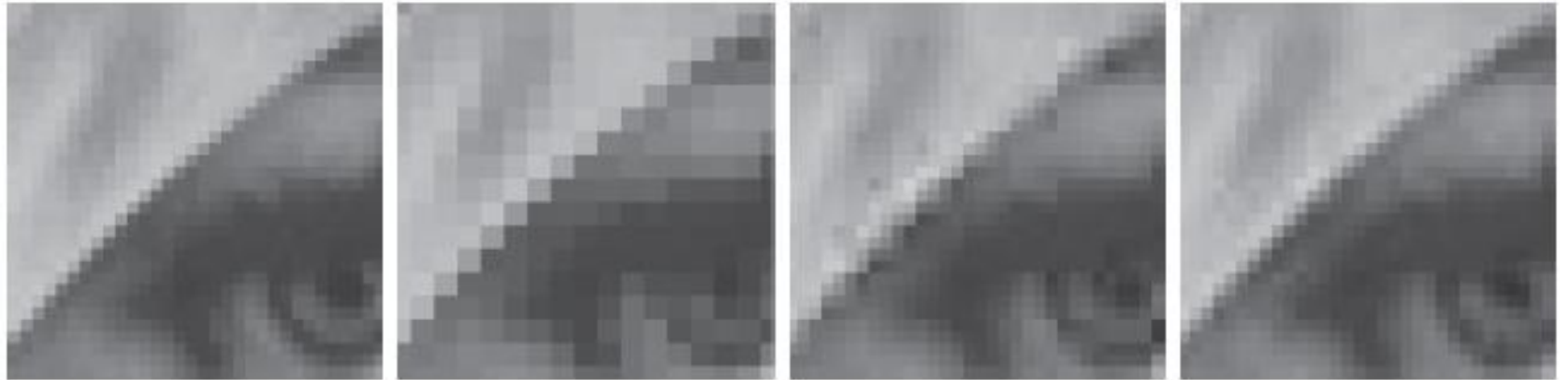
WHT
RMSE = 0.023

DCT/DFT/WHT comparison for 8 x 8 subimages, 25% coefficients (with largest magnitude) retained. Note also the blocking artifact.

Subimage Size Selection

- Images are subdivided into subimages of size $n \times n$ reduce the correlation (redundancy) between adjacent subimages.
- Usually $n = 2k$, for some integer k . this simplifies the computation of the transforms (ex. FFT algorithm).
- Typical block sizes used in practice are 8×8 and 16×16 .





a b c d

FIGURE 8.24 Approximations of Fig. 8.24(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).

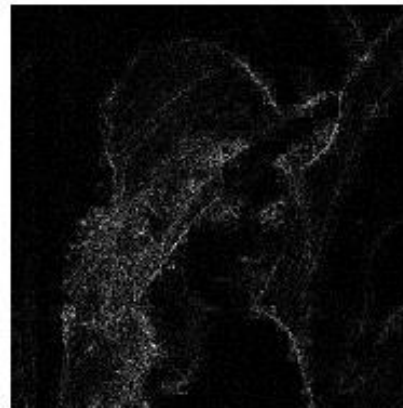
Bit Allocation

- After transforming each subimage, only a fraction of the coefficients are retained. This can be done in two ways:
 - **Zonal coding:** Transform coefficients with large variance are retained. Same set of coefficients retained in all subimages.
 - **Threshold coding:** Transform coefficients with large magnitude in each subimage are retained. Different set of coefficients retained in different subimages.
- The retained coefficients are quantized and then encoded.
- The overall process of truncating, quantizing, and coding the transformed coefficients of the subimage is called **bit-allocation**.

Reconstructed Image

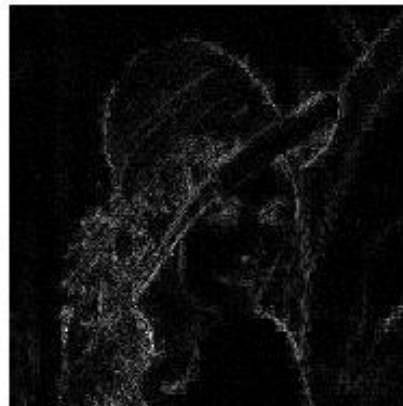


Error Image



RMSE = 0.029

Threshold coding



RMSE = 0.038

Zonal coding

Comparison of Zonal and Threshold coding for 8 x 8 DCT subimages, with 12.5% coefficients retained in each case.

Zonal Coding:

- Transform coefficients with large variance carry most of the information about the image. Hence a fraction of the coefficients with the largest variance is retained.
- The variance of each coefficient is calculated based on the ensemble of $(N/n)^2$ transformed subimages, or using a statistical image model.
- The coefficients with maximum variance are usually located around the origin of an image transform. This is usually represented as a 0-1 mask.
- The same set of coefficients are retained in each subimage.
- The retained coefficients are then quantized and coded. Two possible ways:
 - The retained coefficients are normalized with respect to their standard deviation and they are all allocated the same number of bits. A uniform quantizer then used.
 - A fixed number of bits is distributed among all the coefficients (based on their relative importance). An optimal quantizer such as a Lloyd-Max quantizer is designed for each coefficient.

Threshold Coding:

- In each subimage, the transform coefficients of largest magnitude contribute most significantly and are therefore retained.
- A different set of coefficients is retained in each subimage. So this is an adaptive transform coding technique.
- The thresholding can be represented as $T(u, v)m(u, v)$, where $m(u, v)$ is a masking function:

$$m(u, v) = \begin{cases} 0 & \text{if } T(u, v) \text{ satisfies some truncation criterion} \\ 1 & \text{otherwise} \end{cases}$$

- The elements of $T(u, v)m(u, v)$ are reordered in a predefined manner to form a 1-D sequence of length n^2 . This sequence has several long runs of zeros, which are run-length encoded.

1	1	1	0	1	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
1	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

Typical threshold mask

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

Zigzag ordering of coefficients

a b
c d

FIGURE 8.26

A typical
(a) zonal mask,
(b) zonal bit allo-
cation,
(c) threshold
mask, and
(d) thresholded
coefficient order-
ing sequence.
Shading highlights
the coefficients
that are retained.

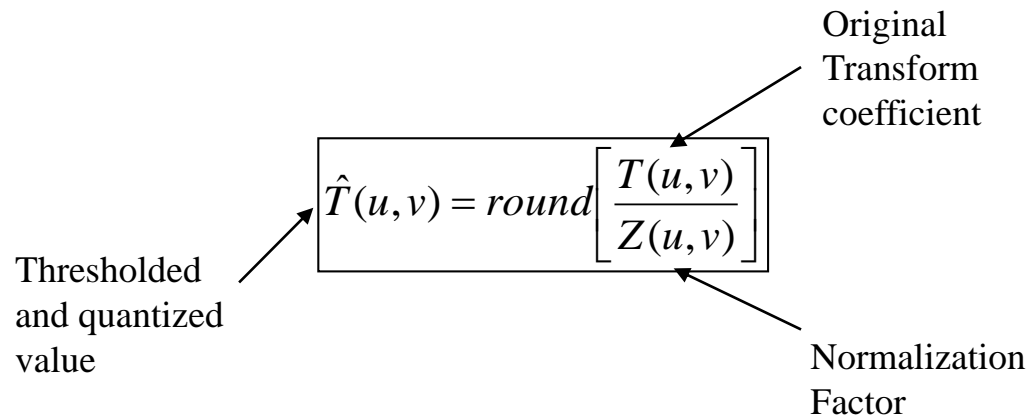
1	1	1	1	1	0	0	0
1	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

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

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

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

- The retained coefficients are encoded using a suitable variable-length code.
- The thresholding itself can be done in three different ways, depending on the “truncation criterion:”
 - A **single global threshold** is applied to all subimages. The level of compression differs from image to image depending on the number of coefficients that exceed the threshold.
 - **N-largest coding:** The largest N coefficients are retained in each subimage. Therefore, a different threshold is used for each subimage. The resulting code rate (total # of bits required) is fixed and known in advance.
 - **Threshold is varied** as a function of the location of each coefficient in the subimage. This results in a variable code rate (compression ratio).
 - Here, the thresholding and quantization steps can be together represented as:



The diagram illustrates the process of thresholding and quantization. It features a central rectangular box containing the mathematical expression $\hat{T}(u, v) = \text{round} \left[\frac{T(u, v)}{Z(u, v)} \right]$. Three arrows point to different parts of this expression: one from the left points to the entire expression and is labeled "Thresholded and quantized value"; one from the top right points to the numerator $T(u, v)$ and is labeled "Original Transform coefficient"; and one from the bottom right points to the denominator $Z(u, v)$ and is labeled "Normalization Factor".

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

Thresholded and quantized value

Original Transform coefficient

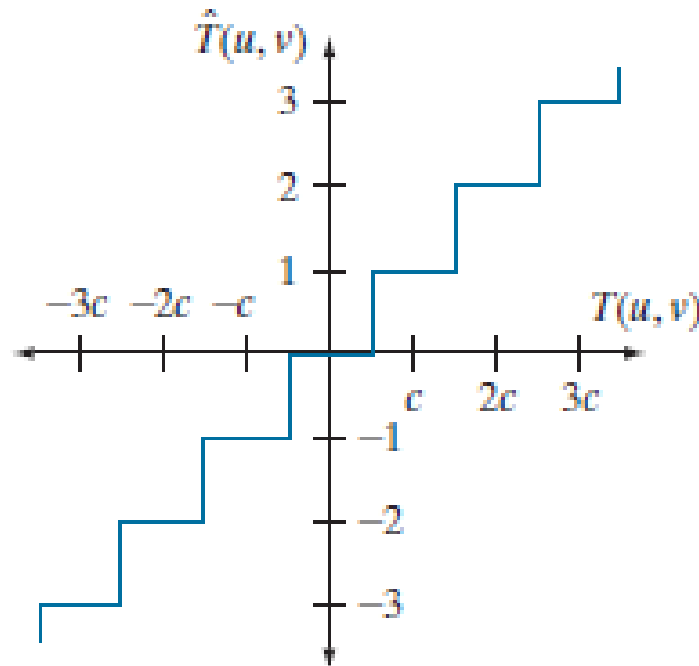
Normalization Factor

- $Z(u, v)$ is transform normalization matrix. Typical example is shown below.

a b

FIGURE 8.27

(a) A threshold coding quantization curve [see Eq. (8-28)]. (b) A typical normalization matrix.



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

- The values in the Z matrix weight the transform coefficients according to heuristically determined perceptual or psychovisual importance. Large the value of $Z(u, v)$, smaller the importance of that coefficient.
- The Z matrix maybe scaled to obtain different levels of compression.
- At the decoding end, $T(u, v) = \hat{T}(u, v)Z(u, v)$ is used to denormalize the transform coefficients before inverse transform.



a	b	c
d	e	f

FIGURE 8.28 Approximations of Fig. 8.9(a) using the DCT and normalization array of Fig. 8.27(b): (a) Z , (b) $2Z$, (c) $4Z$, (d) $8Z$, (e) $16Z$, and (f) $32Z$.



a	b	c
d	e	f

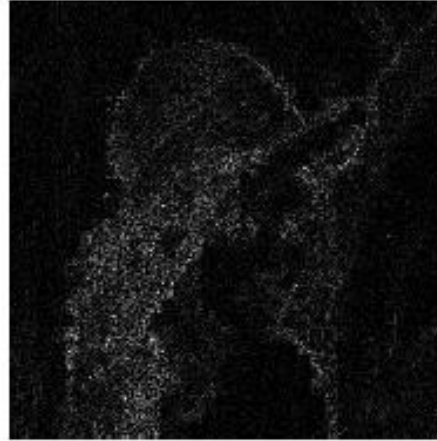
FIGURE 8.29 Two JPEG approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image.

Example: Coding with different Z

Reconstructed Image



Error Image



RMSE = 0.023

Quantization matrix Z
(9152 non-zero coefficients)



RMSE = 0.048

Quantization matrix 8Z
(2389 nonzero coefficients)