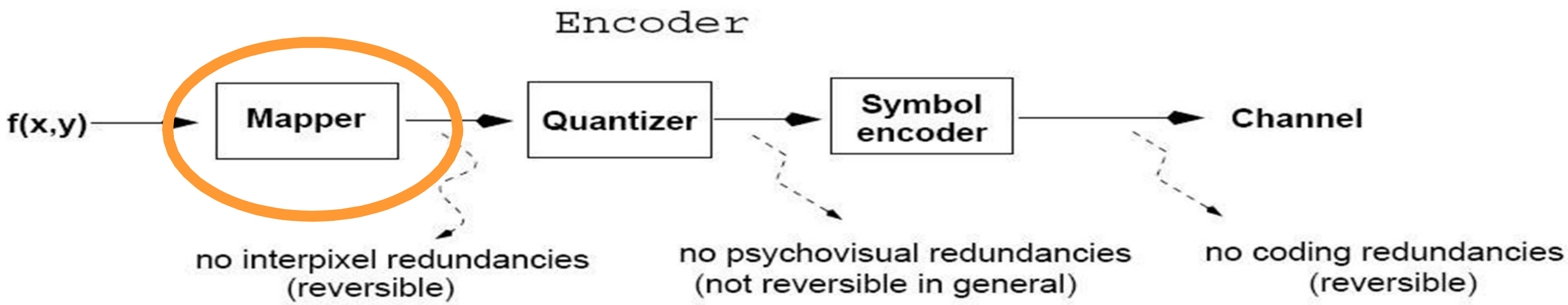
Image Compression Model



- **Mapper:** transforms input data in a way that facilitates reduction of interpixel redundancies (spatial and temporal redundancy)
- Example - Run length coding

Image Compression Model (cont'd)

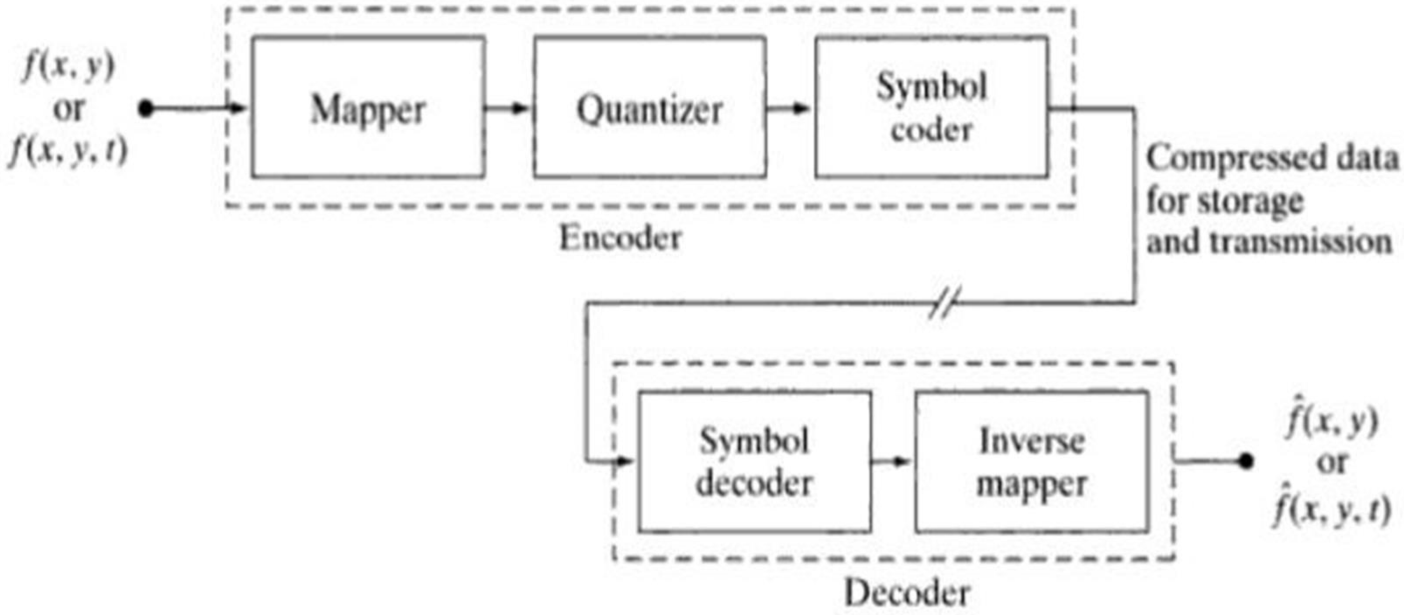
•**Quantizer :** the accuracy of the mapper's output in accordance with some pre-established fidelity criteria.

- The goal is to keep irrelevant information out of the compressed representation.
- It is irreversible but it must be omitted when error free compression is desired.

Image Compression Model (cont'd)

- **Symbol encoder:** assigns the shortest code to the most frequently occurring output values – thus minimizing coding redundancy.

Image Compression Models (cont'd)



- Inverse operations are performed.

Consider the simple 4×8, 8-bit image:

$$\begin{array}{cccccccc}
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\
21 & 21 & 21 & 95 & 169 & 243 & 243 & 243
\end{array}$$

(b) Compress the image using Huffman coding.

(c) Compute the compression achieved and the effectiveness of the Huffman coding.

21	3/8	3/8	5/8
243	3/8	3/8	3/8
95	1/8	2/8	
169	1/8		

Source reductions

21	1	1	0
243	00	00	1
95	000	01	
169	001		

Code assignments

(c) Using Eq. (8.1-4), the average number of bits required to represent each pixel in the Huffman coded image (ignoring the storage of the code itself) is

$$L_{avg} = 1\left(\frac{3}{8}\right) + 2\left(\frac{3}{8}\right) + 3\left(\frac{1}{8}\right) + 3\left(\frac{1}{8}\right) = \frac{15}{8} = 1.875 \text{ bits/pixel.}$$

Thus, the compression achieved is

$$C = \frac{8}{1.875} = 4.27.$$

Because the theoretical compression resulting from the elimination of all coding redundancy is $\frac{8}{1.811} = 4.417$, the Huffman coded image achieves $\frac{4.27}{4.417} \times 100$ or 96.67% of the maximum compression possible through the removal of coding redundancy alone.

Using the previously discussed Huffman decode the encoded string
0101000001010111110100

Original source		
Sym.	Prob.	Code
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010
a_5	0.04	01011

a3 a6 a6 a2 a5 a2 a2 a2 a4.

Decoding in LZW

Dictionary

0	a
1	b

0 1 2 4 3

Decoding in LZW

Dictionary

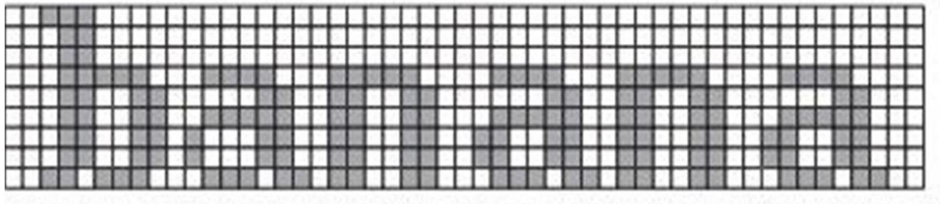
0	a
1	b

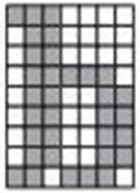
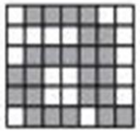
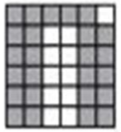
0 1 2 4 3

a b ab aba ba

Symbol-Based Coding

In *symbol-* or *token-based* coding, an image is represented as a collection of frequently occurring sub-images, called *symbols*. Each such symbol is stored in a *symbol dictionary* and the image is coded as a set of triplets $\{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots\}$, where each (x_i, y_i) pair specifies the location of a symbol in the image and *token* t_i is the address of the symbol or sub-image in the dictionary.



Token	Symbol
0	
1	
2	

Triplet
(0, 2, 0)
(3, 10, 1)
(3, 18, 2)
(3, 26, 1)
(3, 34, 2)
(3, 42, 1)

Compression ratio??

In this case, the starting image has 9x51x1 or 459 bits and, assuming that each triplet is composed of 3 bytes, the compressed representation has 6x3x8 +[(9x7)+(6x7)+(6x6)] or 285 bits; the resulting compression ratio= 1.61

Bit-Plane Coding

The symbol-based technique of the previous section can be applied to images with more than two intensities by processing their bit planes individually. The technique, called bit-plane coding, is based on the concept of decomposing a multilevel (monochrome or color) image into a series of binary images.

The intensities of an m-bit monochrome image can be represented in the form of the base-2 polynomial

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_1 2^1 + a_0 2^0$$

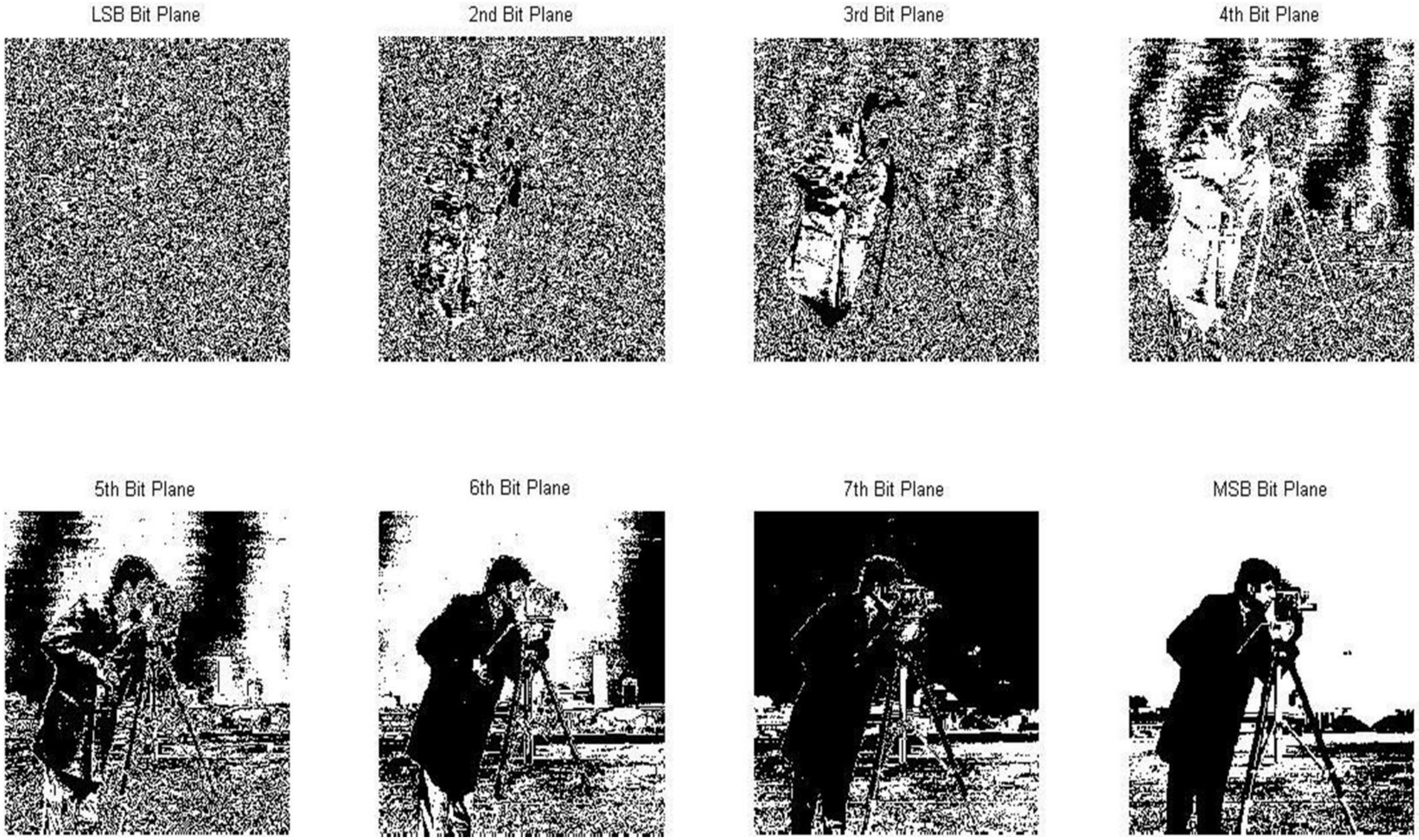
Example: display different bits as an individual image

```
>> clear
>> x=imread('cameraman.tif');
>> imshow(x);
>> x=double(x);
>> for k=1:256
    for j=1:256
        if x(k,j)>=128
            x1(k,j)=128;
        else
            x1(k,j)=0;
        end;
    end;
end;
>> figure
>> imshow(x1);
>> y1=imsubtract(x,x1);
>> for k=1:256
    for j=1:256
        if y1(k,j)>=64
            x2(k,j)=64;
        else
            x2(k,j)=0;
        end;
    end;
end;
>> figure
>> imshow(x2)
>>
```

original

Bit 7

Bit 6


LSB Bit Plane
2nd Bit Plane
3rd Bit Plane
4th Bit Plane
5th Bit Plane
6th Bit Plane
7th Bit Plane
MSB Bit Plane

Bit plane slicing: solution??

127	128	127
128	127	128
127	128	127

0	1	0
1	0	1
0	1	0

1	0	1
0	1	0
1	0	1

1	0	1
0	1	0
1	0	1

1	0	1
0	1	0
1	0	1

1	0	1
0	1	0
1	0	1

1	0	1
0	1	0
1	0	1

1	0	1
0	1	0
1	0	1

1	0	1
0	1	0
1	0	1

Gray codes instead of binary!

128⇒ 10000000 ⇒ 11000000

127⇒ 01111111⇒ 01000000

$$g_i = a_i \oplus a_{i+1} \quad 0 \leq i \leq m-2$$
$$g_{m-1} = a_{m-1}$$

Bit plane slicing: solution??

127	128	127
128	127	128
127	128	127

0	1	0
1	0	1
0	1	0

1	1	1
1	1	1
1	1	1

0	0	0
0	0	0
0	0	0

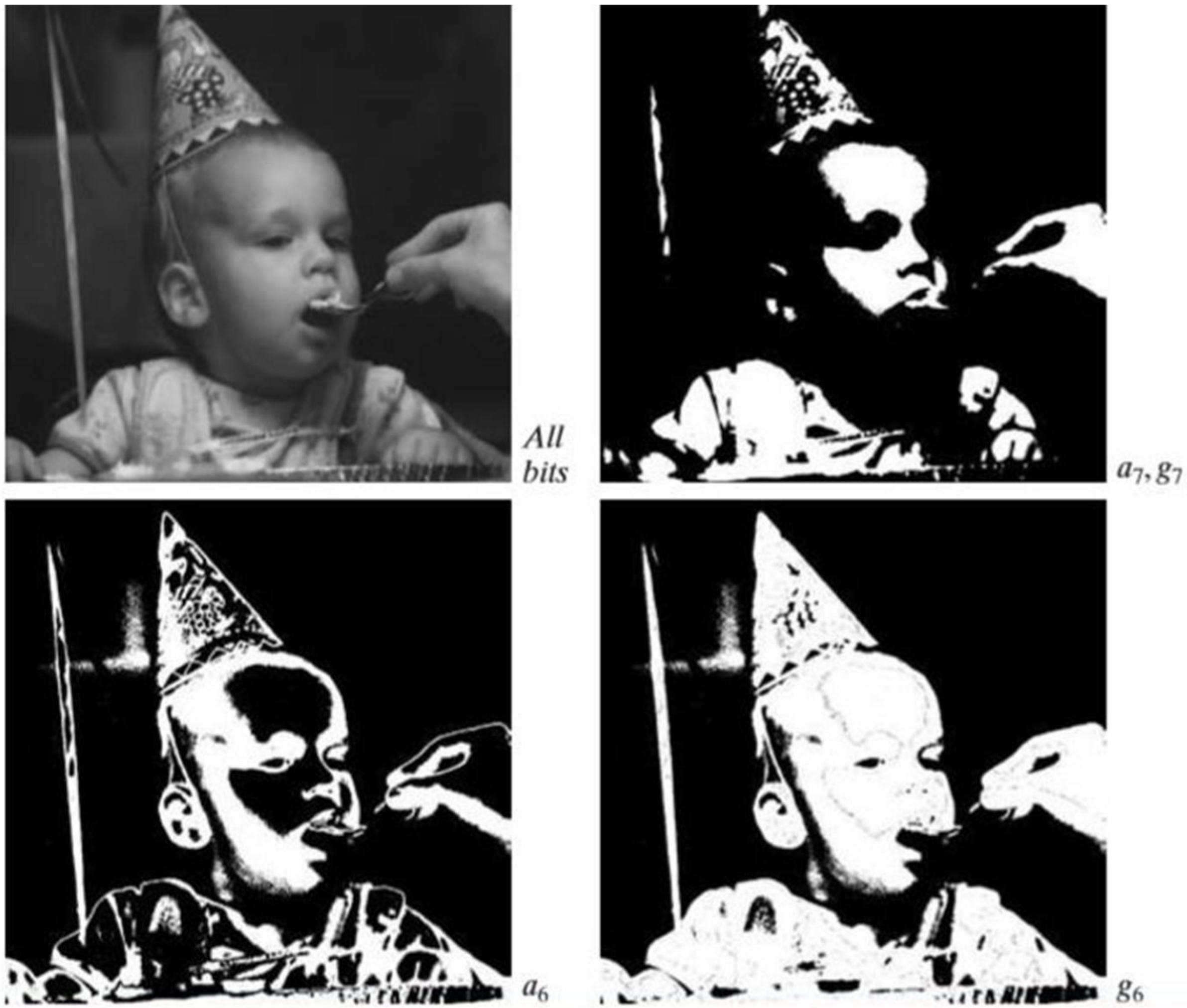
0	0	0
0	0	0
0	0	0

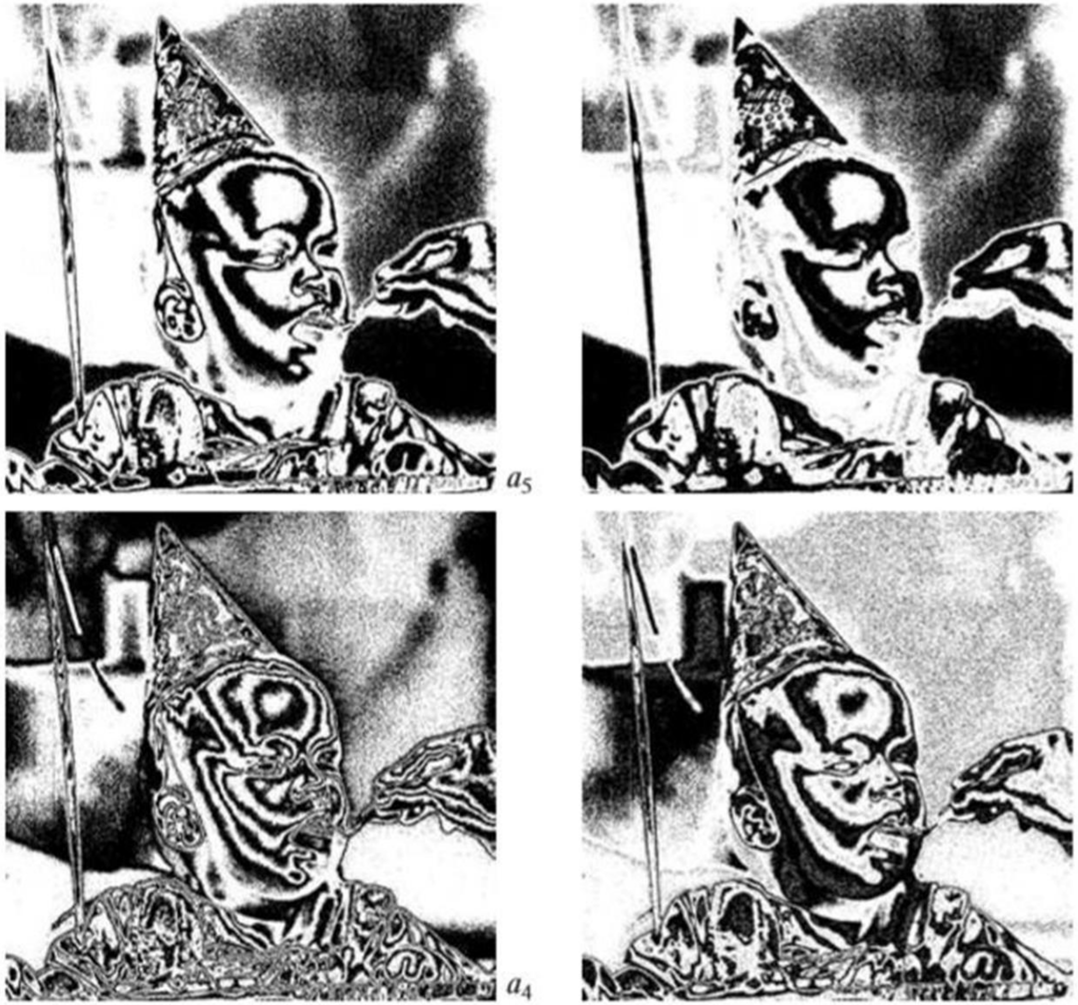
0	0	0
0	0	0
0	0	0

0	0	0
0	0	0
0	0	0

0	0	0
0	0	0
0	0	0

0	0	0
0	0	0
0	0	0

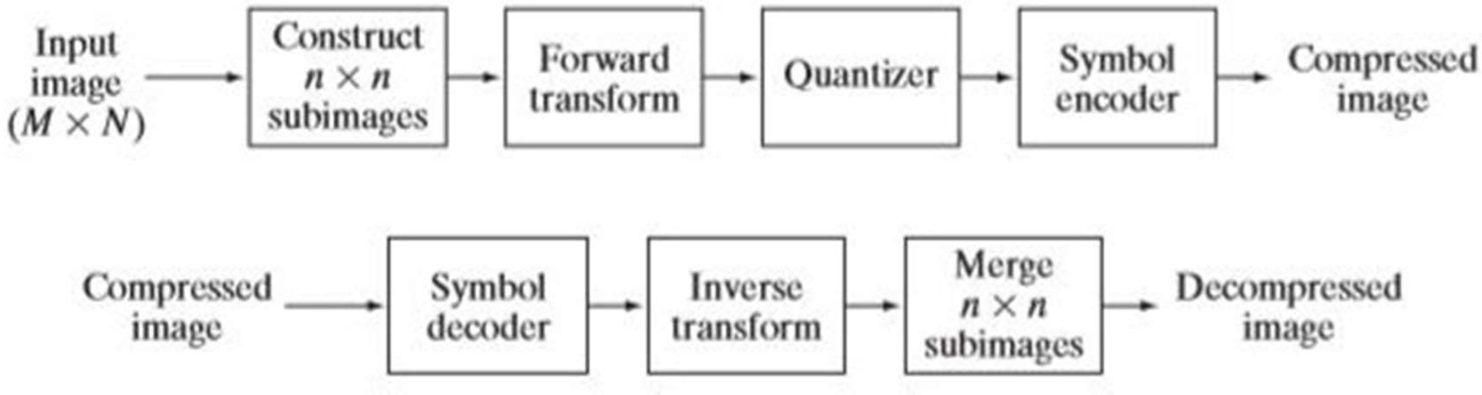

All bits
a_7, g_7
a_6
g_6



a_5 g_5

a_4 g_4

BLOCK TRANSFORM CODING



$g(x, y)$ of size $n \times n$ whose forward, discrete transform, $T(u, v)$, can be expressed in terms of the general relation

$$T(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x, y) r(x, y, u, v) \tag{8.2-10}$$

for $u, v = 0, 1, 2, \ldots, n - 1$. Given $T(u, v)$, $g(x, y)$ similarly can be obtained using the generalized inverse discrete transform

$$g(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) s(x, y, u, v) \tag{8.2-11}$$

for $x, y = 0, 1, 2, \ldots, n - 1$. In these equations, $r(x, y, u, v)$ and $s(x, y, u, v)$ are called the *forward* and *inverse transformation kernels*, respectively.

Walsh-Hadamard Transform

r(x, y, u, v)=$\frac{1}{n}(-1)^{\sum_{i=0}^{m-1} b_i(x)p_i(u)+bi(y)p_i(v)}$

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

$N=4=2^m \rightarrow m=2$

Walsh-Haramard Transform

x, y = 0,1,2,3

$$r(x, y, u, v)=\frac{1}{n}(-1)^{\sum_{i=0}^{m-1} b_i(x)p_i(u)+bi(y)p_i(v)}$$

u, v = 0, 0

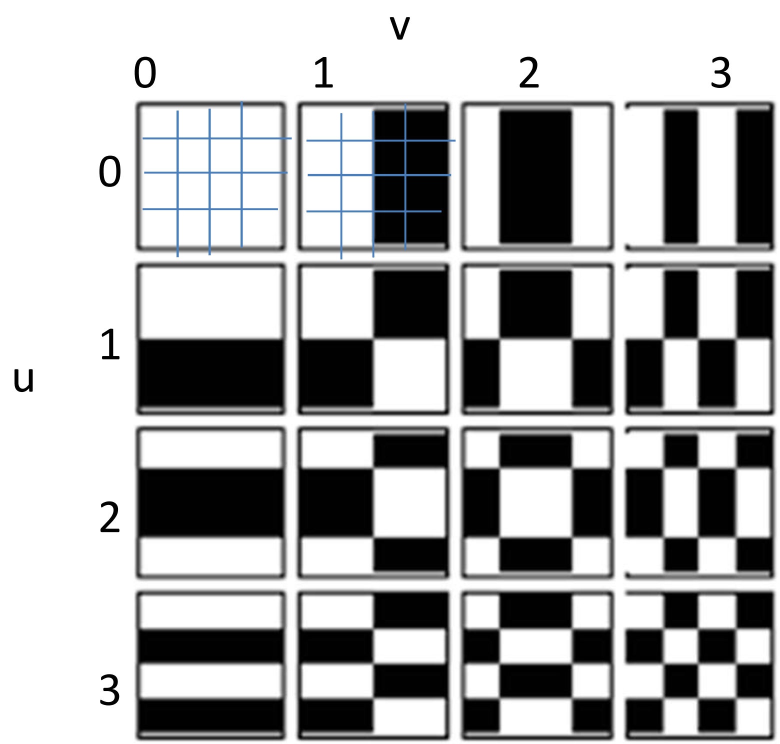
$R(0,0,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(0,1,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(0,2,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(0,3,0,0)= (1/4)(-1)^0 \rightarrow 1$

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



$N=4=2^m \rightarrow m=2$

Walsh-Haramard Transform

x, y = 0,1,2,3

u, v = 0, 0

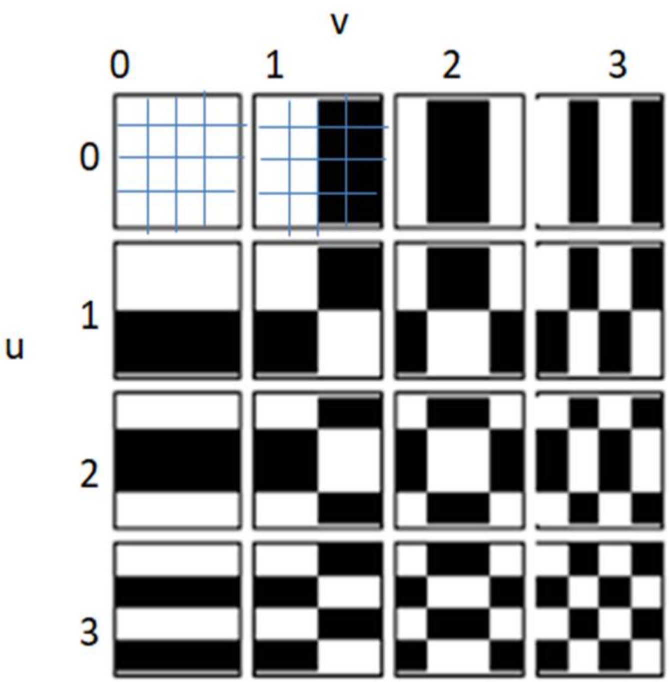
$$r(x, y, u, v)=\frac{1}{n}(-1)^{\sum_{i=0}^{m-1} b_i(x)p_i(u)+bi(y)p_i(v)}$$

$R(1,0,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(1,1,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(1,2,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(1,3,0,0)= (1/4)(-1)^0 \rightarrow 1$



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

$R(2,0,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(2,1,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(2,2,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(2,3,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(3,0,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(3,1,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(3,2,0,0)= (1/4)(-1)^0 \rightarrow 1$

$R(3,3,0,0)= (1/4)(-1)^0 \rightarrow 1$

N=4 =2^m → m=2

Walsh-Haramard Transform

x, y = 0,1,2,3

$$r(x, y, u, v)=\frac{1}{n}(-1)^{\sum_{i=0}^{m-1} b_i(x)p_i(u)+bi(y)p_i(v)}$$

u, v = 0, 1

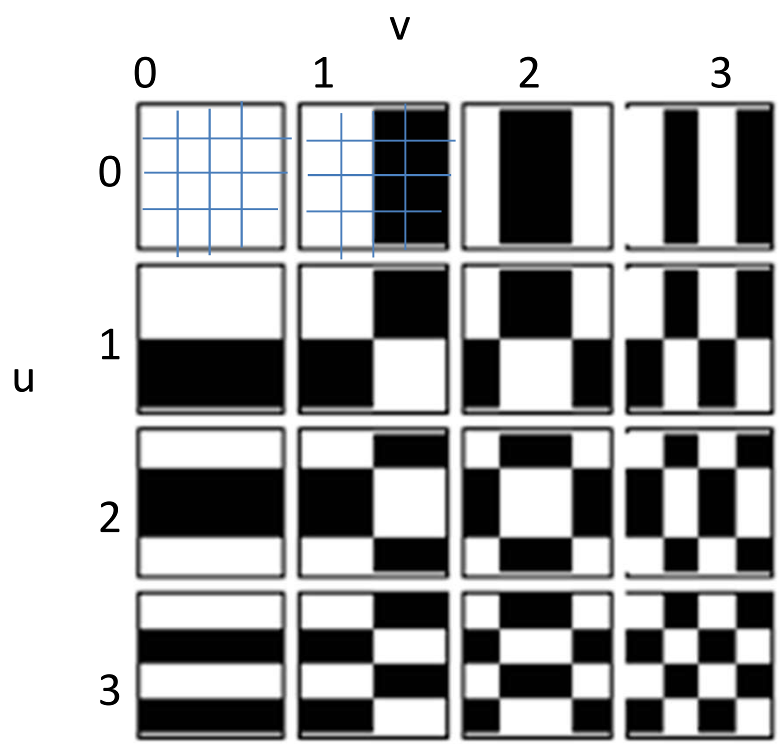
R(0,0,0,1)= (1/4)$(-1)^0$ → 1

R(0,1,0,1)= (1/4)$(-1)^0$ → 1

R(0,2,0,1)= (1/4)$(-1)^1$ → -1

R(0,3,0,1)= (1/4)$(-1)^1$ → -1

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



Walsh-Haramard Transform

$N=4=2^m \rightarrow m=2$

x, y = 0,1,2,3

u, v = 0, 1

$$r(x, y, u, v)=\frac{1}{n}(-1)^{\sum_{i=0}^{m-1} b_i(x)p_i(u)+bi(y)p_i(v)}$$

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

$R(1,0,0,1)= (1/4)(-1)^0 \rightarrow 1$

$R(1,1,0,1)= (1/4)(-1)^0 \rightarrow 1$

$R(1,2,0,1)= (1/4)(-1)^1 \rightarrow -1$

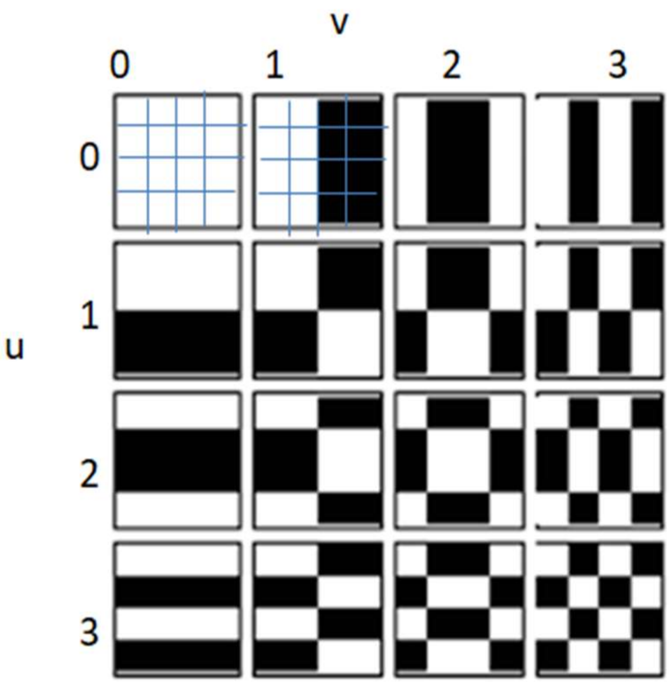
$R(1,3,0,1)= (1/4)(-1)^1 \rightarrow -1$

$R(2,0,0,1)= (1/4)(-1)^0 \rightarrow 1$

$R(2,1,0,1)= (1/4)(-1)^0 \rightarrow 1$

$R(2,2,0,1)= (1/4)(-1)^1 \rightarrow -1$

$R(2,3,0,1)= (1/4)(-1)^1 \rightarrow -1$



$R(3,0,0,1)= (1/4)(-1)^0 \rightarrow 1$

$R(3,1,0,1)= (1/4)(-1)^0 \rightarrow 1$

$R(3,2,0,1)= (1/4)(-1)^1 \rightarrow -1$

$R(3,3,0,1)= (1/4)(-1)^1 \rightarrow -1$

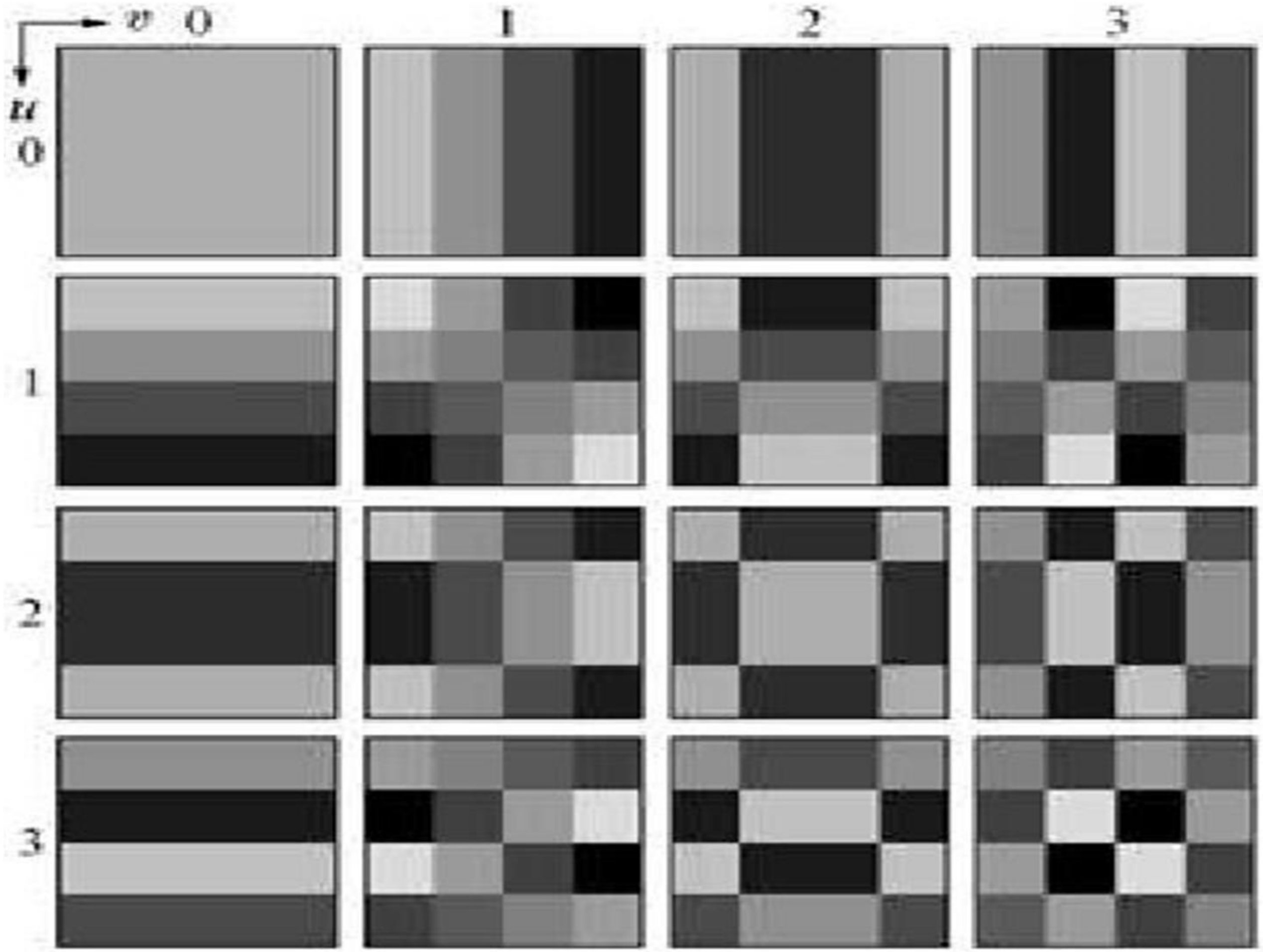
One of the transformations used most frequently for image compression is the *discrete cosine transform* (DCT). It is obtained by substituting the following (equal) kernels into Eqs. (8.2-10) and (8.2-11)

$$
\begin{aligned}
r(x, y, u, v) &= s(x, y, u, v) \\
&= \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2n}\right]\cos\left[\frac{(2y+1)v\pi}{2n}\right]
\end{aligned}
\tag{8.2-18}
$$

where

$$
\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{n}} & \text{for } u = 0 \\ \sqrt{\dfrac{2}{n}} & \text{for } u = 1, 2, \ldots, n-1 \end{cases}
\tag{8.2-19}
$$

and similarly for $\alpha(v)$. Figure 8.23 shows $r(x, y, u, v)$ for the case $n = 4$.


v
u
0
1
2
3
0
1
2
3

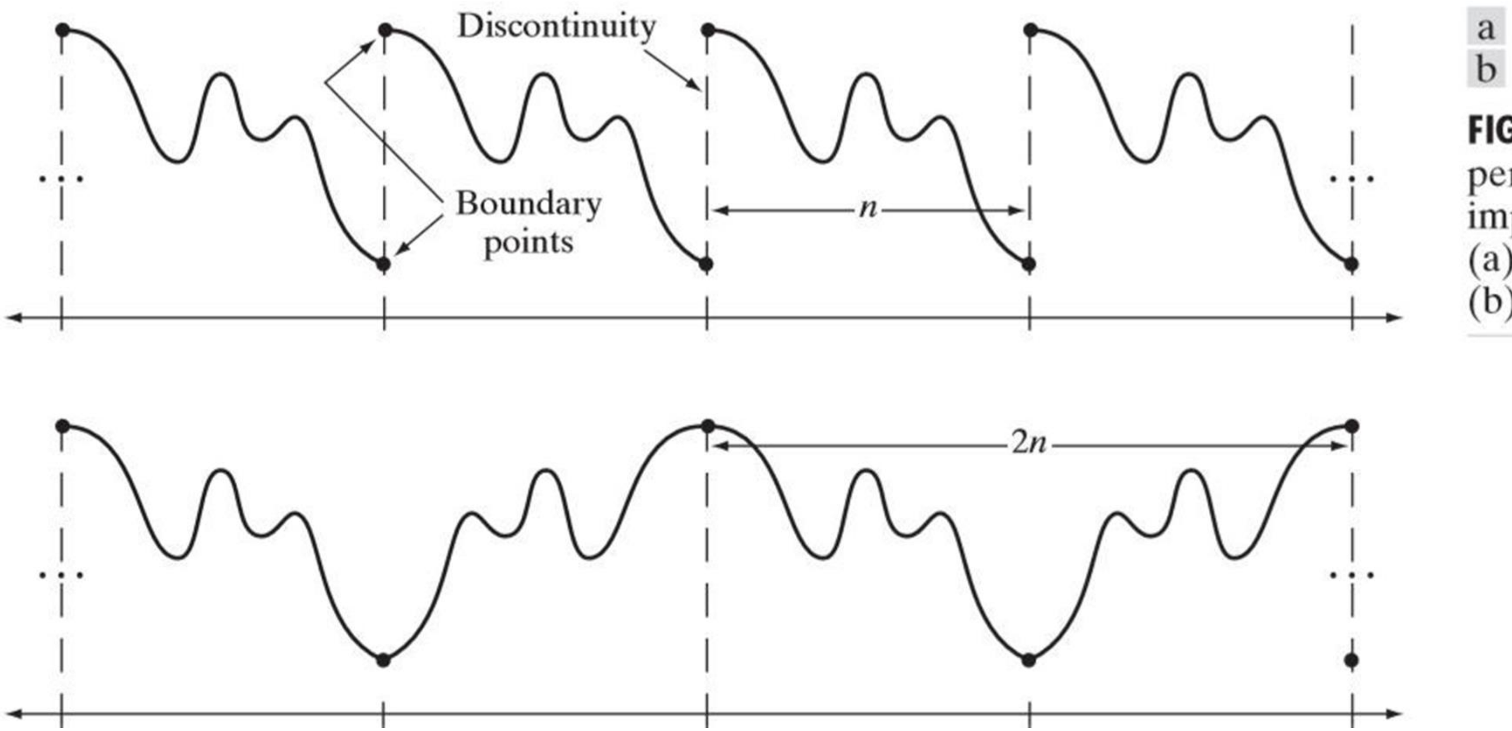


FIGURE 8.25 The periodicity implicit in the 1-D (a) DFT and (b) DCT.