

影像處理

(Image Processing)

真理大學 資訊工程系 吳汶涓老師

Course 14

影像編碼與壓縮



Outline

14.1 無失真與失真壓縮

14.2 Huffman編碼

14.3 行程長度編碼

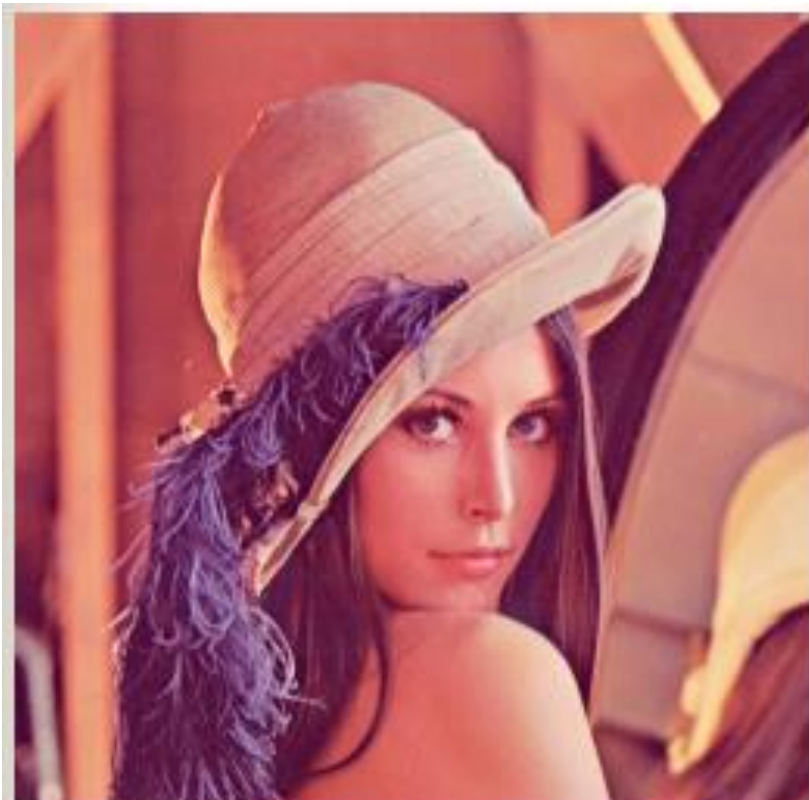
14.4 JPEG演算法



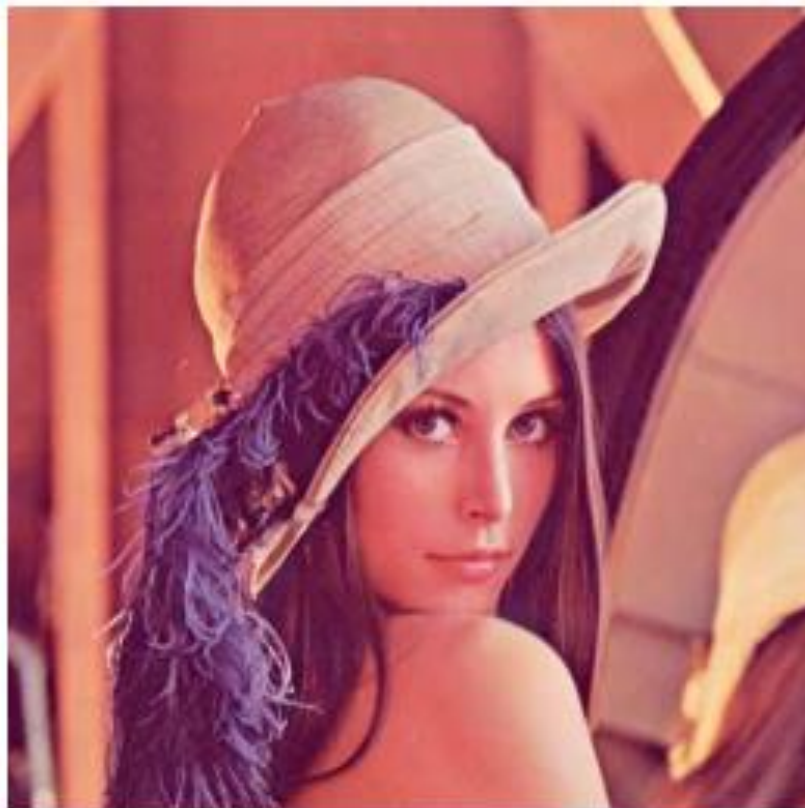
14.1 無失真與失真壓縮

- 影像檔案可能會十分龐大，不管是為了儲存或傳輸，都會希望檔案能夠縮小。
- 兩種不同類型的壓縮方法：
 - **無失真壓縮** (lossless compression)
 - 保留所有的資訊
 - 特別是醫學影像 (X光片)
 - **失真壓縮** (lossy compression)
 - 某些資訊會被省略
 - 得到較高的**壓縮比** (原影像的資料量 / 壓縮影像資料量)

為什麼需要資料壓縮？

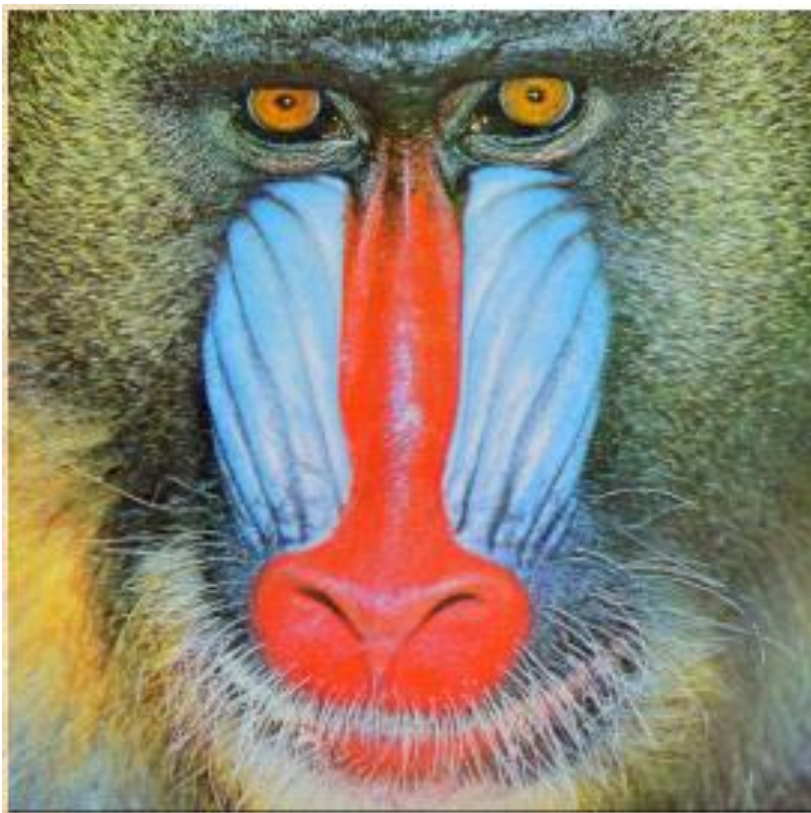


786488 bytes

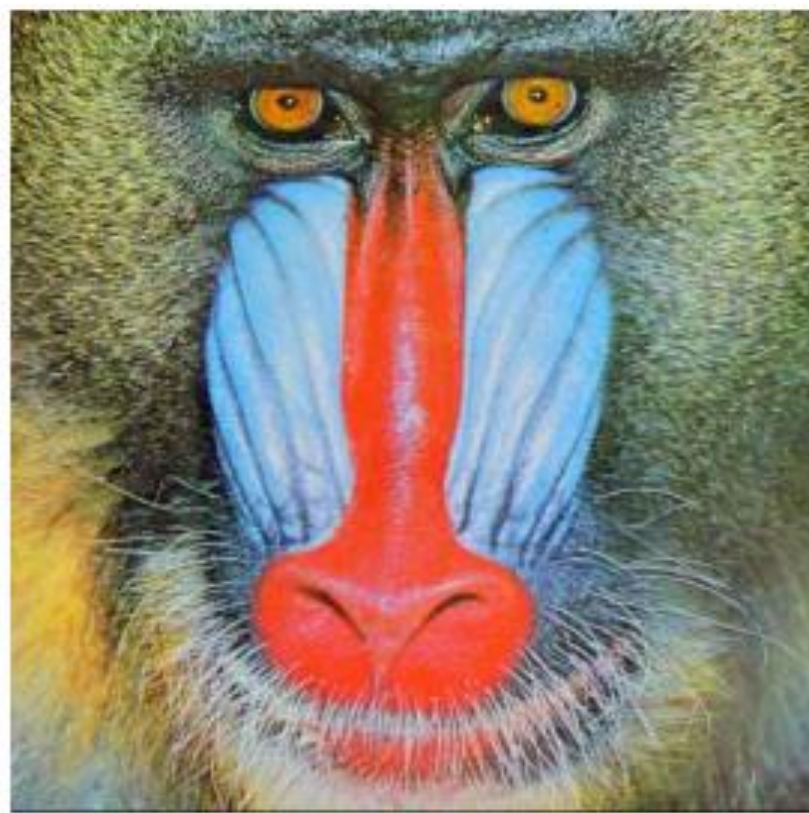


23116 bytes, Cr=34.0

為什麼需要資料壓縮？



786488 bytes



49746 bytes, Cr=15.80

14.2 Huffman編碼

灰階值	機率	固定長度編碼	變動長度編碼
0	0.2	00	000
1	0.4	01	1
2	0.3	10	01
3	0.1	11	001

- 每個像素的平均位元數按照**機率**計算：

$$(0.2 \times 3) + (0.4 \times 1) + (0.3 \times 2) + (0.1 \times 3) = 1.9$$

$$H = - \sum_{i=0}^{L-1} p_i \log_2(p_i) \Rightarrow H = -(0.2 \log_2(0.2) + 0.4 \log_2(0.4) + 0.3 \log_2(0.3) + 0.1 \log_2(0.1)) = 1.8464$$

熵：在不流失任何資訊下，每個像素所需的最小位元數

求得影像Huffman編碼的方法如下：

1. 決定影像中灰階值的出現機率。
2. 取最小的兩個數值，將機率兩兩相加，建構二元編碼樹。
3. 從編碼樹頂端開始，每一分支任意指定0與1。
4. 從上至下讀取編碼。

Gray value	0	1	2	3	4	5	6	7
probability	0.19	0.25	0.21	0.16	0.08	0.06	0.03	0.02

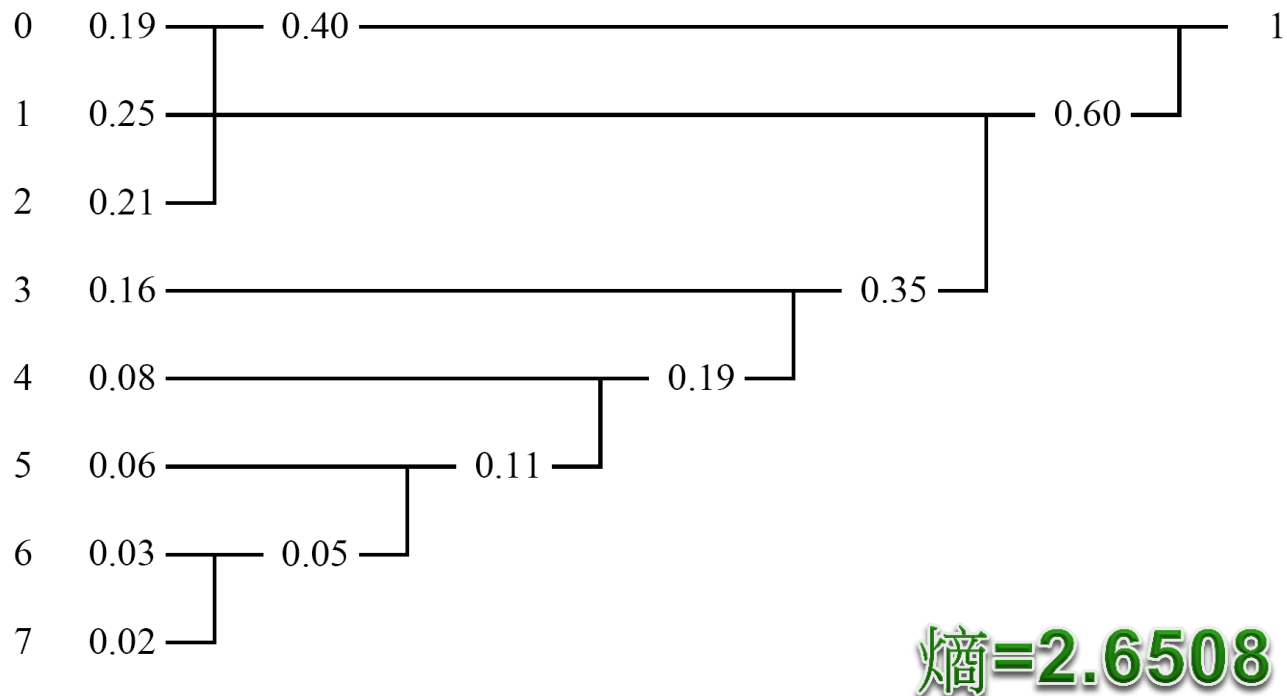
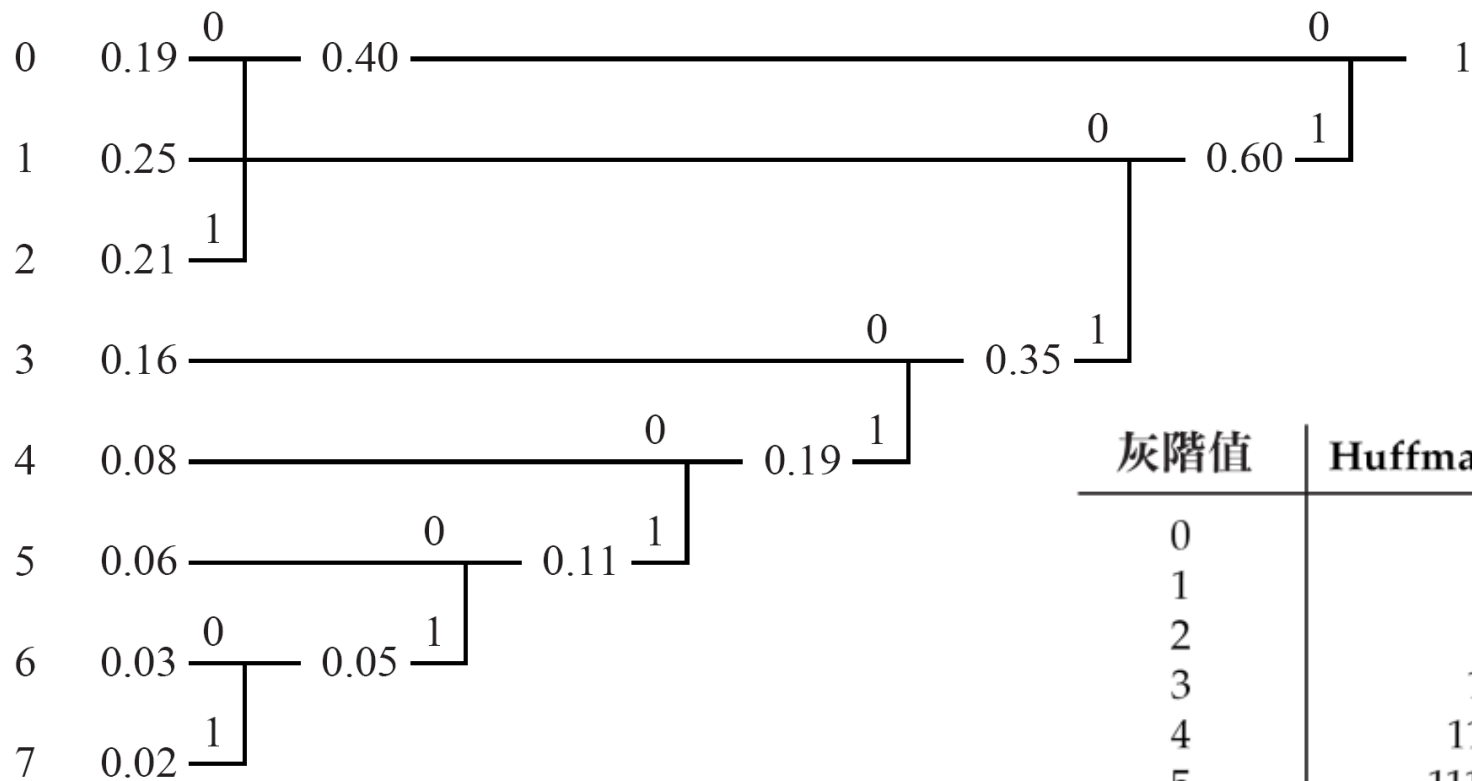


圖 14.1 建構 Huffman 編碼樹



灰階值	Huffman 編碼
0	00
1	10
2	01
3	110
4	1110
5	11110
6	111110
7	111111

圖 14.2 分支指定 0 與 1

- 我們可以計算像素平均位元數：

$$(0.19 \times 2) + (0.25 \times 2) + (0.21 \times 2) + (0.16 \times 3) \\ + (0.08 \times 4) + (0.06 \times 5) + (0.03 \times 6) + (0.02 \times 6) = 2.7$$

- Huffman 編碼符合**唯一解碼**性質，即字串的解碼結果只有一種


$\underbrace{1 \ 1 \ 0}_3 \quad \underbrace{1 \ 1 \ 1 \ 0}_4 \quad \underbrace{0 \ 0}_0 \quad \underbrace{0 \ 0}_0 \quad \underbrace{1 \ 0}_1 \quad \underbrace{0 \ 1}_2 \quad \underbrace{1 \ 1 \ 1 \ 1 \ 0}_5$

灰階值	Huffman 編碼
0	00
1	10
2	01
3	110
4	1110
5	11110
6	111110
7	111111

14.3 行程長度編碼

- 行程長度編碼(run-length encoding, RLE)的概念：
按照字串中0與1重複的長度進行編碼
 - 應用於傳真機上。
 - 其中一種方法是從0的長度開始計算，對各列分別進行編碼。


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



(123)(231)(0321)(141)(33)(0132)

- 另一種編碼方法是將各列編成一組組的數字。第一個數字代表1 開始的位置，第二個數字則是1 字串的長度。

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



(22)(33)(1361)(24)(43)(1152)

- 灰階影像可以分解成位元平面再加以編碼

10	7	8	9	1010	0111	1000	1001
11	8	7	6	1011	1000	0111	0110
9	7	5	4	1001	0111	0101	0100
10	11	2	1	1010	1011	0010	0001

0	1	0	1	1	1	0	0	0	1	0	0	1	0	1	1
1	0	1	0	1	0	1	1	0	0	1	1	1	1	0	0
1	1	1	0	0	1	0	0	0	1	1	1	1	1	0	0
0	1	0	1	1	1	1	0	0	0	0	0	0	1	1	0
第零平面				第一平面				第二平面				第三平面			
LSB												MSB			

每個平面可以使用適合的RLE 運算方式分別進行編碼

- 但是，使用位元平面有個問題就是灰階值小幅的變動可能會造成位元完全改變。
 - 可使用灰階值的二元Gray碼 (Gray codes)來改善
 - 將固定長度的二位元字串依某種順序排列的編碼，使得兩個相鄰的字串之間只有一個位元不同

15	1	0	0	0	8	8	7	8	1000	1000	0111	1000	1100	1100	0100	1100
14	1	0	0	1	8	7	8	7	1000	0111	1000	0111	1100	0100	1100	0100
13	1	0	1	1	7	7	8	7	0111	0111	1000	0111	0100	0100	1100	0100
12	1	0	1	0	7	8	7	7	0111	1000	0111	0111	0100	1100	0100	0100
11	1	1	1	0					Binary code				Gray code			
10	1	1	1	1	0	0	1	0	0	0	1	0	1	1	0	1
9	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
8	1	1	0	0	1	1	0	1	1	1	0	1	0	0	1	0
7	0	1	0	0	1	0	1	1	1	0	1	1	0	1	0	0
6	0	1	0	1	第零平面				第一平面				第二平面			
5	0	1	1	1												
4	0	1	1	0												
3	0	0	1	0												
2	0	0	1	1												
1	0	0	0	1												
0	0	0	0	0												

■ MATLAB中的行程長度編碼

```
L=prod(size(im));  
im=reshape(im',1,L);
```

```
min(find(im==1))
```

影像	尋找	位置	RLE 輸出
			[]
[0 0 1 1 1 0 0 0 1]	1	3	[2]
[1 1 1 0 0 0 1]	0	4	[2 3]
[0 0 0 1]	1	4	[2 3 3]
[1]	0	未找到	[2 3 3 1]

```

function out=rle(image)
% Example:
%   rle([1 1 1 0 0;0 0 1 1 1;1 1 0 0 0])
%
%   ans = 0     3     4     5     3
%
L=prod(size(image));
im=reshape(image',1,L);
x=1;
out=[];
while L ~= 0,
    temp=min(find(im == x));
    if isempty(temp),
        break
    end;
    out=[out L];
    x=1-x;
    im=im(temp:L);
    L=L-temp+1;
end;

```

二元黑白圖

```

> c=imread('circles.tif');
> cr=rle(c);
> whos c cr

```

Name	Size	Bytes	Class
c	256x256	65536	uint8 array (logical)
cr	1x693	5544	double array

```

>> cr=uint16(cr);
>> whos cr

```

Name	Size	Bytes	Class
cr	1x693	1386	uint16 array

$$65536/8 = 8,192 \text{ bytes}$$

圖 14.3 求得二元數位影像行程長度編碼的 MATLAB 函數

```
>> t=imread('text.tif');  
>> tr=rle(t);  
>> whos t tr
```

Name	Size	Bytes	Class
t	256x256	65536	uint8 array (logical)
tr	1x2923	23384	double array

```
>> tr=uint16(tr);  
>> whos tr
```

Name	Size	Bytes	Class
tr	1x2923	5846	uint16 array

14.4 JPEG演算法

- 失真壓縮是容許資料的損失換取更高的壓縮率。
- 其中以聯合影像專家小組（Joint Photographic Experts Group, **JPEG**）所發展的演算法最為普遍。
- 這演算法使用的是**轉換編碼**（transform coding），不是直接以像素值來編碼，而是使用轉換的結果運算。
 - **離散餘弦轉換**（discrete cosine transform, DCT）

❑ 離散餘弦轉換 (discrete cosine transform, DCT)

- 若 $f(j, k)$ 為一 8×8 區塊，則正向 (二維) DCT 定義如下：

$$F(u, v) = \frac{C(u)C(v)}{4} \sum_{i=0}^7 \sum_{k=0}^7 f(j, k) \cos \left(\frac{(2j+1)u\pi}{16} \right) \cos \left(\frac{(2k+1)v\pi}{16} \right)$$

- 而對應的反DCT 為：

$$f(j, k) = \sum_{u=0}^7 \sum_{v=0}^7 f(u, v) C(u) c(v) \cos \left(\frac{(2j+1)u\pi}{16} \right) \cos \left(\frac{(2k+1)v\pi}{16} \right)$$

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}} & \text{若 } w = 0 \\ 0 & \text{其他} \end{cases}$$

■ DCT 的幾項性質特別適合用於壓縮：

1. 均為實數，因此不須處理複數。
2. 濃縮能力強，只要幾個係數就可以承載大量資訊。
3. 在硬體中的運算效率高。
4. 和FFT一樣，轉換法另有「快速」版本。
5. 轉換所使用的基底函數與輸入資料無關。

```
>> a=[10:15:115]
```

```
a =
```

```
10    25    40    55    70    85   100   115
```

```
>> fa=fft(a);  
>> fa(5:8)=0;  
>> round(abs(ifft(fa)))
```

```
ans =
```

```
49    41    56    57    71    70    85    90
```

```
>> da=dct(a);  
>> da(5:8)=0;  
>> round(idct(da))
```

```
ans =
```

```
11    23    41    56    69    84   102   114
```

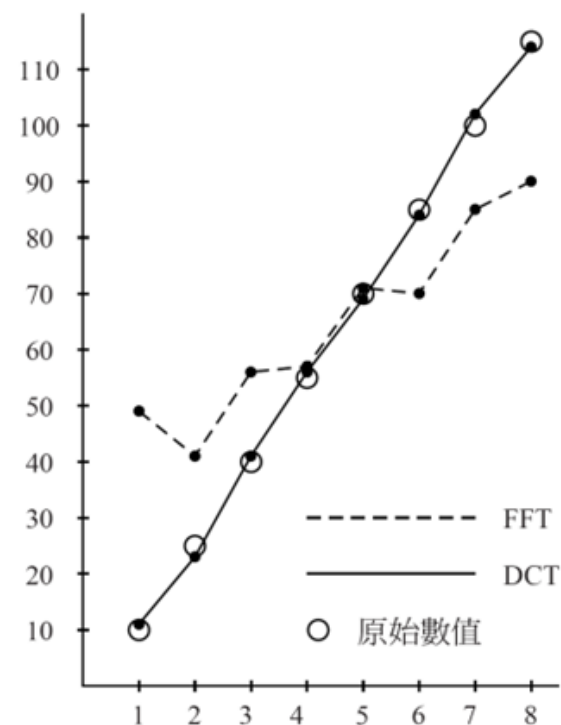
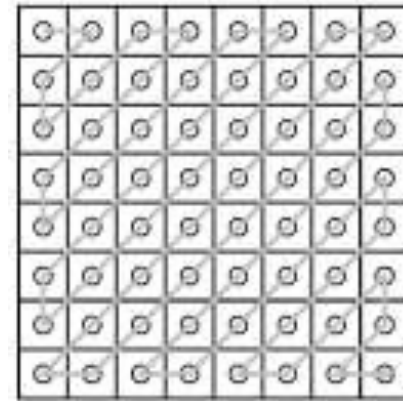


圖 14.4 比較 FFT 與 DCT

■ JPEG的演算法

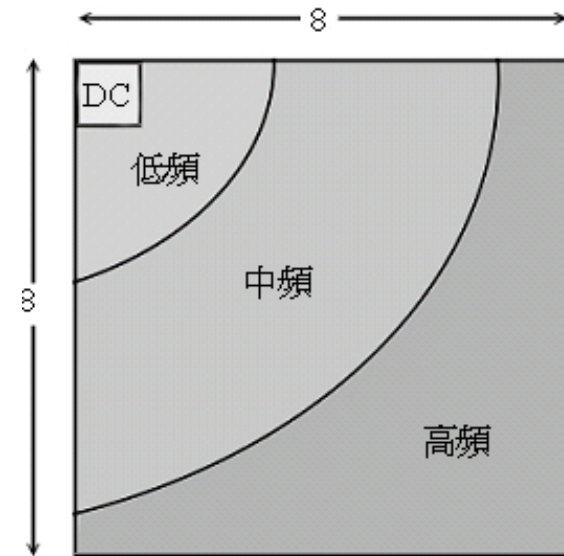
1. 影像切割成8×8 大小的區塊，分別轉換各區塊。
2. 每個區塊的數值均平移（減去）128。
3. 將平移後的區塊代入DCT。(matlab: dct()函式)
4. 將DCT 值除以正規化矩陣 Q 。正規化的動作讓區塊裡大部分的元素都變成零，因此壓縮了影像。(Quantization)
5. 將矩陣轉換為一維向量，從左上角開始以鋸齒狀的方式讀取所有非零數值。(Zig-zag scan)

$$Q : \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$



5. 每個向量的第一個係數是向量中最大的元素，稱為**DC 係數**。
DC 的編碼是使用每個區塊的DC 係數與前一區塊DC 係數之間的差值，再使用RLE 壓縮數值。
6. 其他數值稱為**AC 係數**（AC coefficients），使用Huffman 編碼進行壓縮。

DC	AC ₁	AC ₃	AC ₆	...
AC ₂	AC ₄	AC ₇	...	
AC ₅	AC ₈	...		
AC ₉	...			
⋮				



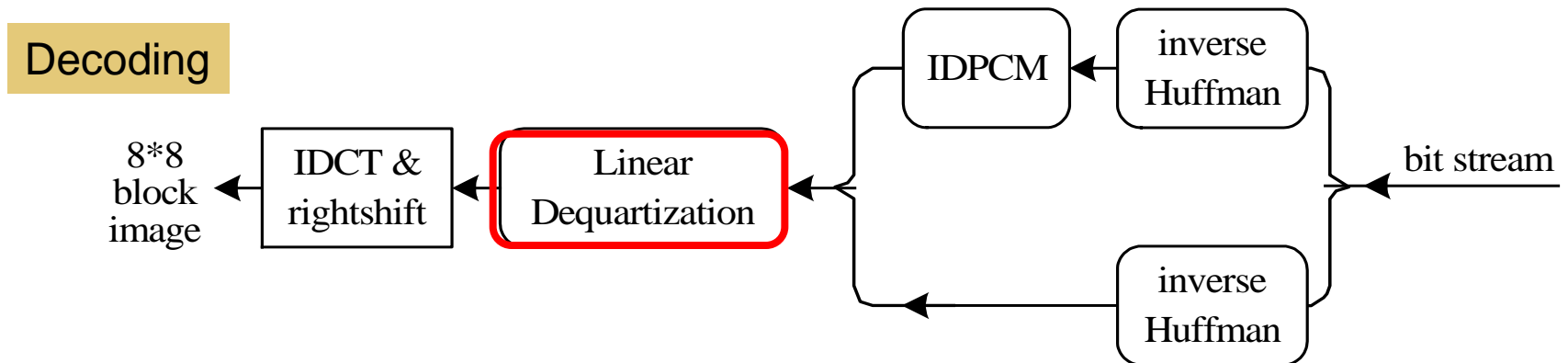
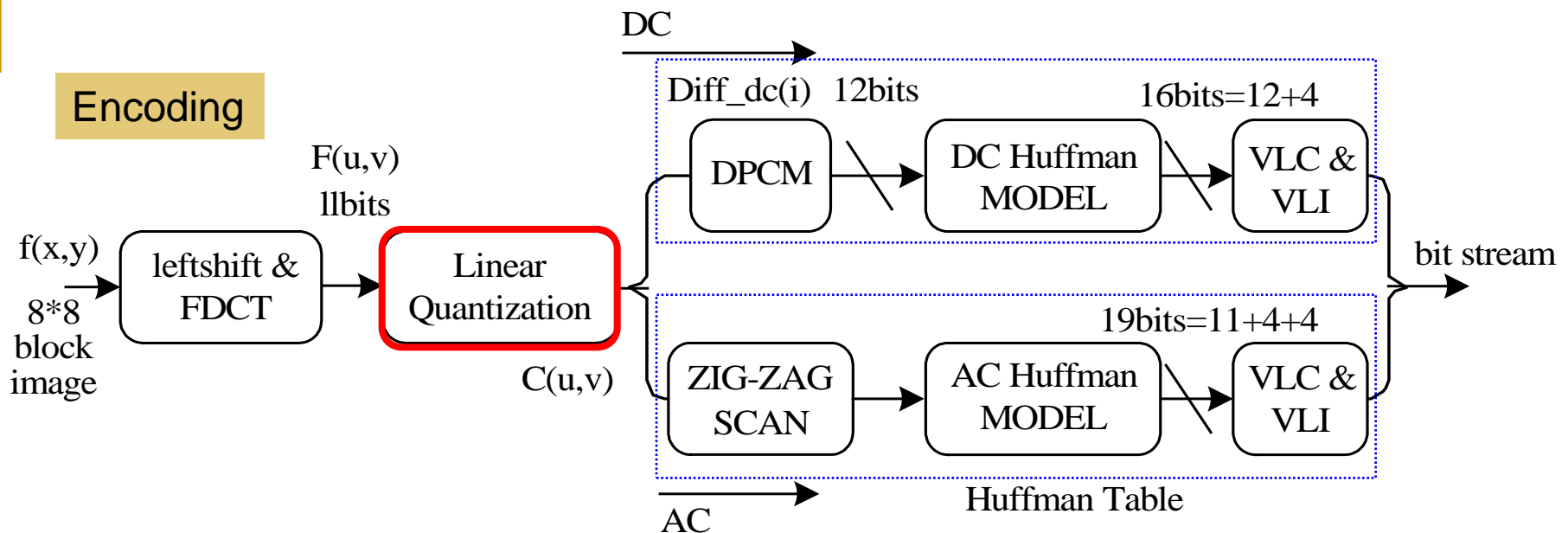


圖7-12 JPEG 壓縮流程圖

壓縮範例：

```
>> c=imread('caribou.tif');  
>> x=151;y=90;  
>> block=c(x:x+7,y:y+7)
```



block =

87	95	92	73	59	57	57	55
74	71	68	59	54	54	51	57
64	58	57	55	58	65	66	65
57	63	68	66	74	89	98	104
95	109	117	114	119	134	145	140
128	139	146	139	140	148	151	143
137	135	125	118	137	156	154	132
122	119	113	110	128	144	140	142

```
>> b=double(block)-128
```

b =

-41	-33	-36	-55	-69	-71	-71	-73
-54	-57	-60	-69	-74	-74	-77	-71
-64	-70	-71	-73	-70	-63	-62	-63
-71	-65	-60	-62	-54	-39	-30	-24
-33	-19	-11	-14	-9	6	17	12
0	11	18	11	12	20	23	15
9	7	-3	-10	9	28	26	4
-6	-9	-15	-18	0	16	12	14



```
>> bd=dct2(b)
```

bd =

-225.3750	-30.7580	17.3864	5.6543	-22.3750	-1.8591	3.7575	1.7196
-241.5333	52.0722	0.8745	-21.2434	8.1434	1.8639	0.9420	-1.3369
-2.5427	50.9316	5.0847	9.1573	1.5820	-3.8454	1.5706	-0.6043
102.5557	23.3927	-11.5151	-12.7655	-10.6629	2.8179	-3.6743	1.2462
-2.3750	-20.7081	3.5090	-10.3182	-1.3750	-2.4723	0.3054	-0.7308
-12.7510	1.5740	2.7664	8.1034	-5.2779	1.0922	-1.6694	1.0561
6.6005	7.8668	-4.9294	-7.0092	2.1860	0.8872	0.6653	-0.1783
10.6630	0.4486	-0.1019	7.9728	-4.0241	2.4364	-2.3823	0.6011


```
>> q = [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];
>> bq=round(bd./q)
```

正規化矩陣

bq =

-14	-3	2	0	-1	0	0	0
-20	4	0	-1	0	0	0	0
0	4	0	0	0	0	0	0
7	1	-1	0	0	0	0	0
0	-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

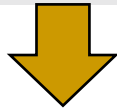
-14 -3 -20 0 4 2 0 0 4 7 0 1 0 -1 -1 0 0 0 -1 0 -1 EOF

解壓縮範例

```
>> bq2=bq.*q
```

```
bq2 =
```

-224	-33	20	0	-24	0	0	0
-240	48	0	-19	0	0	0	0
0	52	0	0	0	0	0	0
98	17	-22	0	0	0	0	0
0	-22	0	0	0	0	0	0
-24	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



```
>> bd2=idct2(bq2)
```

```
bd2 =
```

-48.1431	-39.4257	-39.8246	-53.5852	-65.6253	-68.5089	-70.4960	-74.9017
-52.5762	-46.8345	-50.6187	-65.1825	-74.3228	-71.8983	-68.6282	-69.8981
-70.6699	-66.1335	-69.6750	-80.0362	-81.2435	-69.9392	-59.8115	-57.6095
-68.4457	-61.8134	-60.1283	-62.2478	-54.7898	-37.7751	-26.0007	-24.2342
-29.6526	-21.6215	-16.7263	-14.7648	-5.2632	9.2177	14.8476	11.3981
1.6297	7.3329	9.2805	8.8036	15.4106	24.9217	24.0240	15.7152
3.0533	4.5797	1.2799	-2.1680	4.6076	15.8252	16.2949	8.4867
-2.8366	-4.0640	-10.3394	-14.0550	-3.8001	13.2610	19.3977	14.9470

```
>> b2=round(bd2+128)
```

```
b2 =
```

80	89	88	74	62	59	58	53
75	81	77	63	54	56	59	58
57	62	58	48	47	58	68	70
60	66	68	66	73	90	102	104
98	106	111	113	123	137	143	139
130	135	137	137	143	153	152	144
131	133	129	126	133	144	144	136
125	124	118	114	124	141	147	143



```
function out=jpg_in(x,n)
q=[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];
bd=dct2(double(x)-128);
out=round(bd./(q*n));
```

```
function out=jpg_out(x,n)
q=[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];
out=round(idct2(x.*q*n)+128);
```

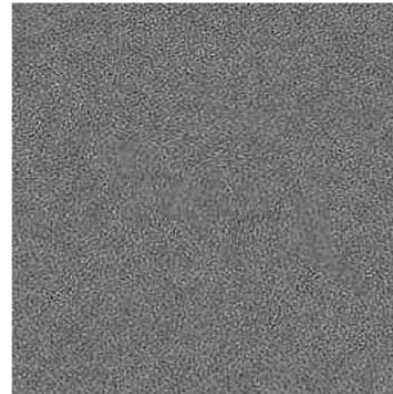
圖 14.5 執行 JPEG 壓縮的 MATLAB 函數



(a)



(b)



(a)與(b)的差異

影像執行 JPEG 壓縮與解壓縮之前後對照

- 參數 n 為2，讓更多的DCT 數值變成0

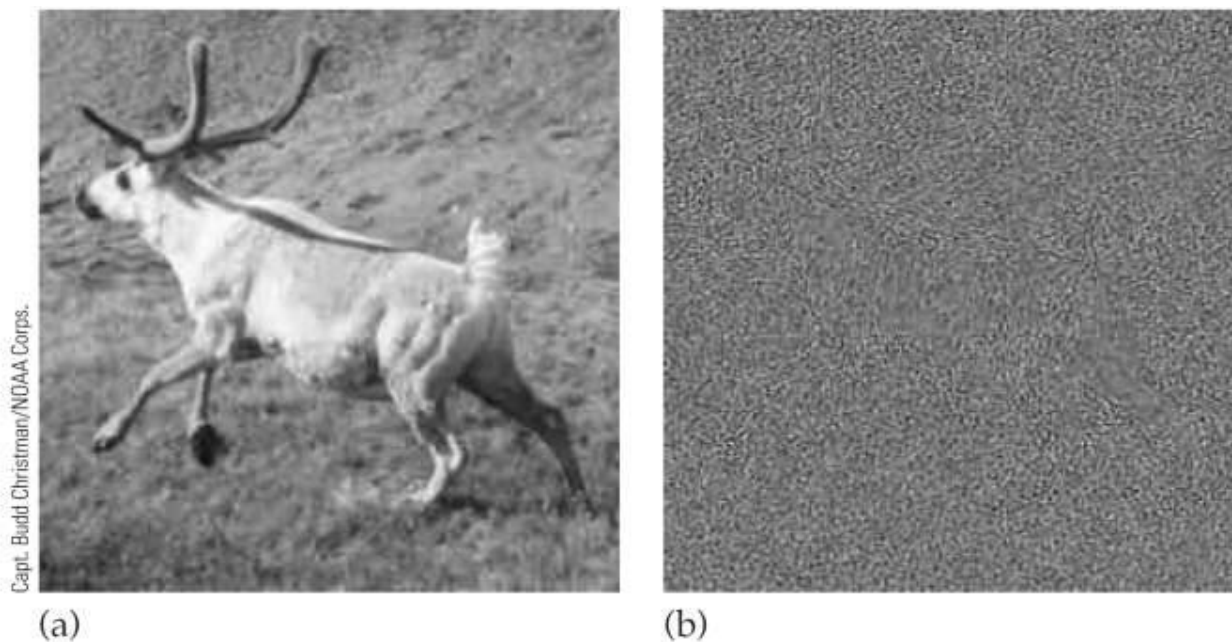


圖 14.8 縮放係數為 2 的 JPEG 壓縮

```
>> cj1=blkproc(c,[8,8],'jpg_in',1);  
>> length(find(cj1==0))
```

ans =

51940

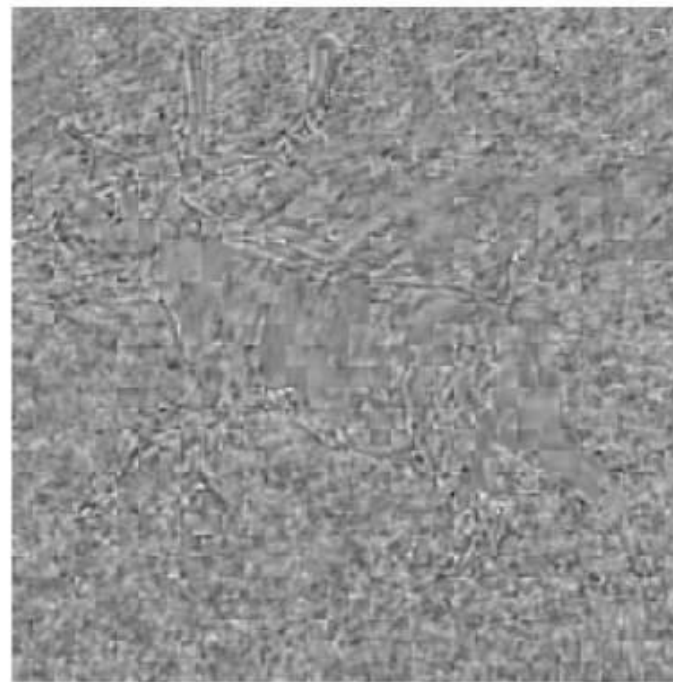
```
>> cj2=blkproc(c,[8,8],'jpg_in',2);  
>> length(find(cj2==0))
```

ans =

56729



(a)



(b)

圖 14.10 縮放係數為 10 的 JPEG 壓縮

練習

- 使用灰階的engineer.tif影像，並對影像使用JPEG壓縮，自設定五個不同的參數，來比較壓縮後的結果，其中每個不同的參數下有多少資訊流失(為0)，有多少個資訊留下呢?
另請問你覺得影像仍可辨識的最大量化縮放係數為何？

PS: 上傳時，需繳交.m與相關的程式、word報告