第十二章
影像與視訊壓縮

內容

- 12.1 前言
- 12.2 消息理論
- 12.3 不失真壓縮
- 12.4 向量量化法
- 12.5 單張影像壓縮
- 12.6 視訊壓縮
- 12.7 結論

12.1 前言

介紹消息理論和單張影像的壓縮原理。也介紹視訊的壓縮原理。例子：JPEG、 H.264/AVC 和 HEVC。

12.2 消息理論

定理12.2.1　給任意 n 個事件，其熵$H \le \log n$ 。

證明：

n 個事件且機率分別為p_1 、p_2 、...和 p_n。

$$H = -\sum_{i=1}^{n} p_i \log p_i$$

$$H - \log n = -\sum_{i=1}^{n} p_i \log p_i - \log n = -\sum_{i=1}^{n} p_i \log p_i - \sum_{i=1}^{n} p_i \log n$$

$$= -\sum_{i=1}^{n} p_i [\log p_i + \log n] = \sum_{i=1}^{n} p_i \log \frac{1}{p_i n}$$

$$\le \sum_{i=1}^{n} p_i \left(\frac{1}{p_i n} - 1\right) = \sum_{i=1}^{n} \frac{1}{n} - \sum_{i=1}^{n} p_i = 0$$

➔ $H - \log n \le 0$

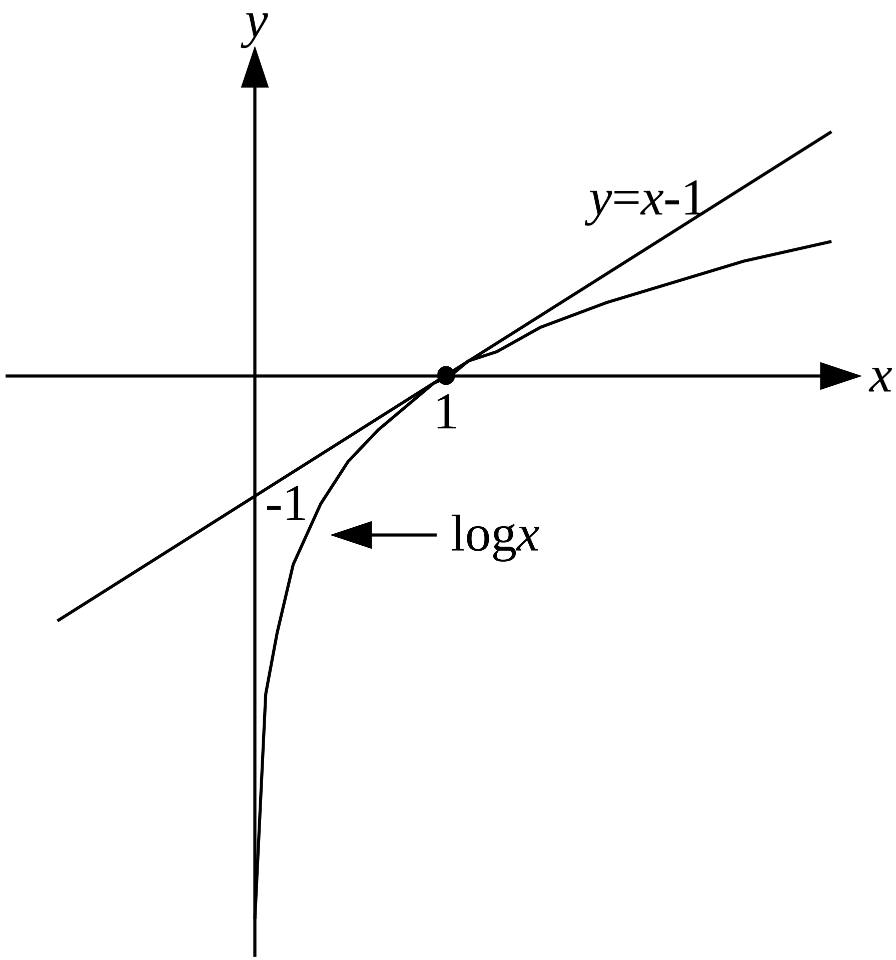


圖12.2.1 $\log x \le x - 1$的示意圖

- Kraft不等式

$S=\{a_1,a_2,...,a_n\}$ 。

a_i長度為d_i ，$d_1 \le d_2 \le ... \le d_n$

編碼a_i ： $C(a_i)$

假設完成了$C(a_1)$ 後，為避免發生$C(a_1)$ 為$C(a_2)$ 的前置碼 (Prefix Code)，則必須滿足條件 $2^{d_2} \ge 2^{d_2-d_1}+1$ ，這裡$2^{d_2-d_1}$ 為不合法的碼數。

同理，考慮時 $C(a_3)$，則需滿足 $2^{d_3} \ge 2^{d_3-d_2}+2^{d_3-d_1}+1$ 。不等式兩邊同除以 2^{d_3}，可得 $1 \ge 2^{-d_2}+2^{-d_1}+2^{-d_3}$ 。依此類推，可得下列Kraft不等式

$$1 \ge 2^{-d_1}+2^{-d_2}+...+2^{-d_n}$$

Kraft不等式將幫助證明熵可視為平均碼長的下限。

定理12.2.2　令$S=\{a_1,a_2,\ldots,a_n\}$且a_i已被編成長度為d_i的碼，則熵$H(S)\le L$，這裡L代表平均碼長。

證明：

已知 $L=\sum_{i=1}^{n} p_i d_i$

➔ $$H(S)-L=\sum_{i=1}^{n} p_i \log\frac{1}{p_i}-\sum_{i=1}^{n} p_i \log 2^{d_i}=\sum_{i=1}^{n} p_i \log\frac{1}{p_i 2^{d_i}}$$

$$\le\sum_{i=1}^{n} p_i\left(\frac{1}{p_i 2^{d_i}}-1\right)=\sum_{i=1}^{n}\frac{1}{2^{d_i}}-\sum_{i=1}^{n} p_i=\sum_{i=1}^{n}\frac{1}{2^{d_i}}-1\le 0$$

又 $$L=\sum_{i=1}^{n} p_i d_i<\sum_{i=1}^{n} p_i\left(1+\log\frac{1}{p_i}\right)=1+H(S)$$

$$L<1+H(S)$$

➔ $H(S)\le L<1+H(S)$

巨集符號(Macro Symbol)集的平均碼長

*n*個符號形成一個巨集符號(Macro Symbol)。

字母集 $=\{\overbrace{S_1S_1\ldots S_1}^{n},\ \ldots\ ,\overbrace{S_{|S|}S_{|S|}\ldots S_{|S|}}^{n}\}$

假設兩兩符號為彼此獨立

➔ $$H(S^{(n)})=-\sum_{i_1=1}^{|S|}\cdots\sum_{i_n=1}^{|S|}p(S_{i_1}S_{i_2}\ldots S_{i_{|n|}})\log p(S_{i_1}S_{i_2}\ldots S_{i_{|n|}})$$

$$=-\sum_{i_1=1}^{|S|}p(S_{i_1})\log p(S_{i_1})\overbrace{\left\{\sum_{i_2=1}^{|S|}\cdots\sum_{i_n=1}^{|S|}p(S_{i_2})\ldots p(S_{i_n})\right\}}^{=1}\cdots-\sum_{i_n=1}^{|S|}p(S_{i_n})\log p(S_{i_n})$$

$$=nH(S)$$

推得 $H(S^{(n)})\le L^{(n)}<1+H(S^{(n)})$

$\boldsymbol{L^{(n)}}$ 表一個巨集符號所需的位元長度。

➔ $$\begin{cases} nH(S)\le nL<1+nH(S) \\ H(S)\le L<\dfrac{1}{n}+H(S) \end{cases}$$ ➔ 若*n*趨近於無窮大，則 $L\approx H(S)$ 。

12.3 不失真壓縮
12.3.1 霍夫曼編碼

- 霍夫曼樹

符號集 $S=\langle S_1,S_2,\ldots,S_8\rangle$

對應頻率 $W=\langle 14,13,5,3,3,2,1,1\rangle$

Key: 每次挑目前頻率最小的二個符號編碼逐步建構出霍夫曼樹。

編成碼 $\langle C_1,C_2,\ldots,C_8\rangle$ =

$\langle 11,10,010,001,000,0111,01101,01100\rangle$

碼長度為

$\langle l_1,l_2,\ldots,l_8\rangle=\langle 2,2,3,3,3,4,5,5\rangle$

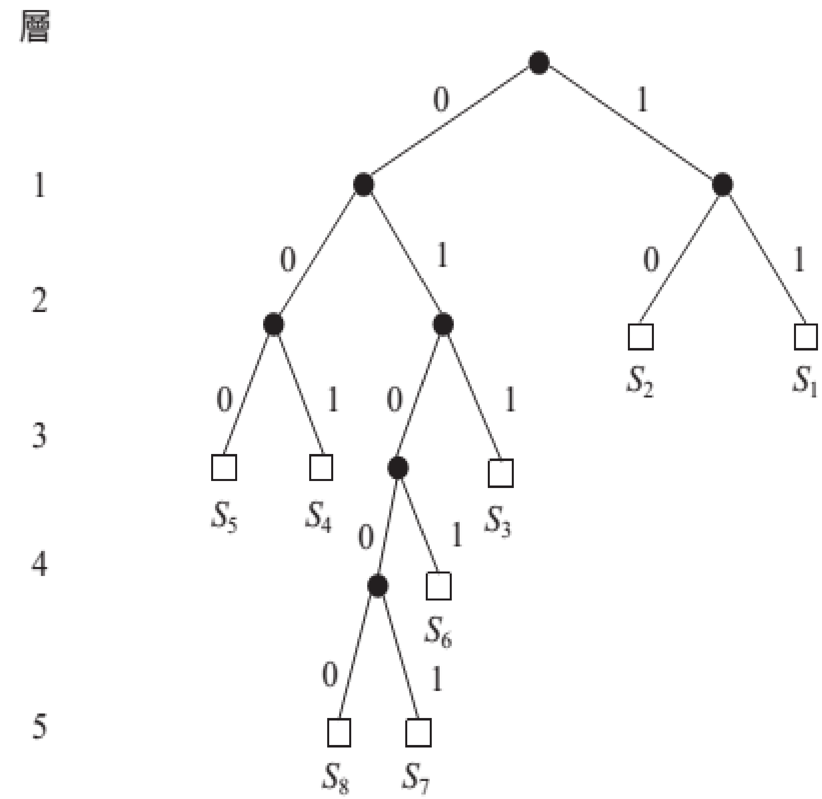
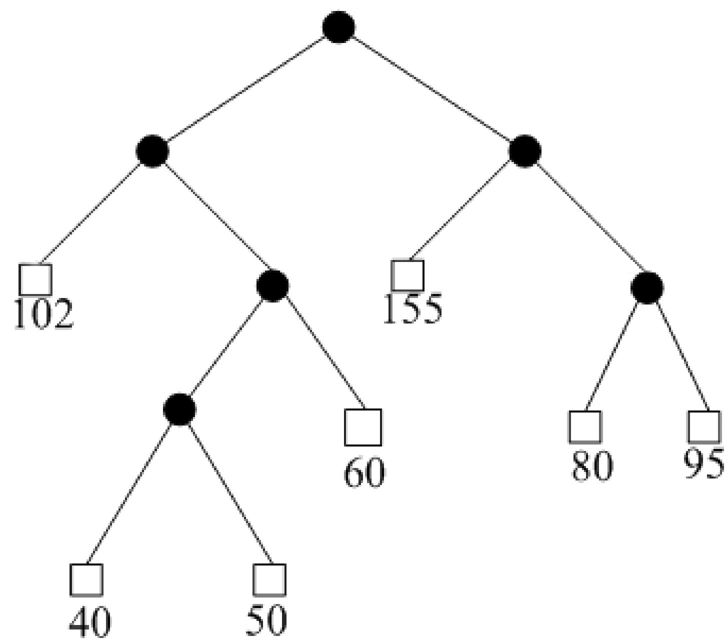


圖12.3.1.1 霍夫曼樹

範例1:給一4×4灰階影像，請建出霍夫曼樹並寫出灰階值50的霍夫曼碼長。

60	102	80	95
95	40	155	60
102	155	102	155
50	80	155	95

解答:S=<40,50,60,80,95,102,155>，而
W=<1,1,2,2,3,3,4>，霍夫曼樹如下:



灰階值50的霍夫曼碼長為4

解答完畢

單邊成長(Single-side Growing)霍夫曼樹

首先令 $C_1^{'}=11\ldots1$ 且 $|C_1^{'}|=l_1$ 。 $C_2^{'}=(C_1^{'}\times 2^{l_2-l_1})-1=11\ldots10$ 。單邊成長霍夫曼樹往左成長，所以 $C_i^{'}=(C_{i-1}^{'}\times 2^{l_i-l_{i-1}})-1$ 。可建出圖12.3.1.2的單邊成長霍夫曼樹且 $\langle C_1^{'},C_2^{'},\ldots,C_8^{'}\rangle=\langle 11,10,011,010,001,0001,00001,00000\rangle$

令 f_i 代表第*i*層的葉子樹。

令 I_i 代表第*i*層的內部節點數。

$\langle f_1,f_2,f_3,f_4,f_5\rangle=\langle 0,2,3,1,2\rangle$

$\langle I_0,I_1,I_2,I_3,I_4\rangle=\langle 1,2,2,1,1\rangle$

$A[0..7]=[S_2,S_1,S_5,S_4,S_3,S_6,S_8,S_7]$

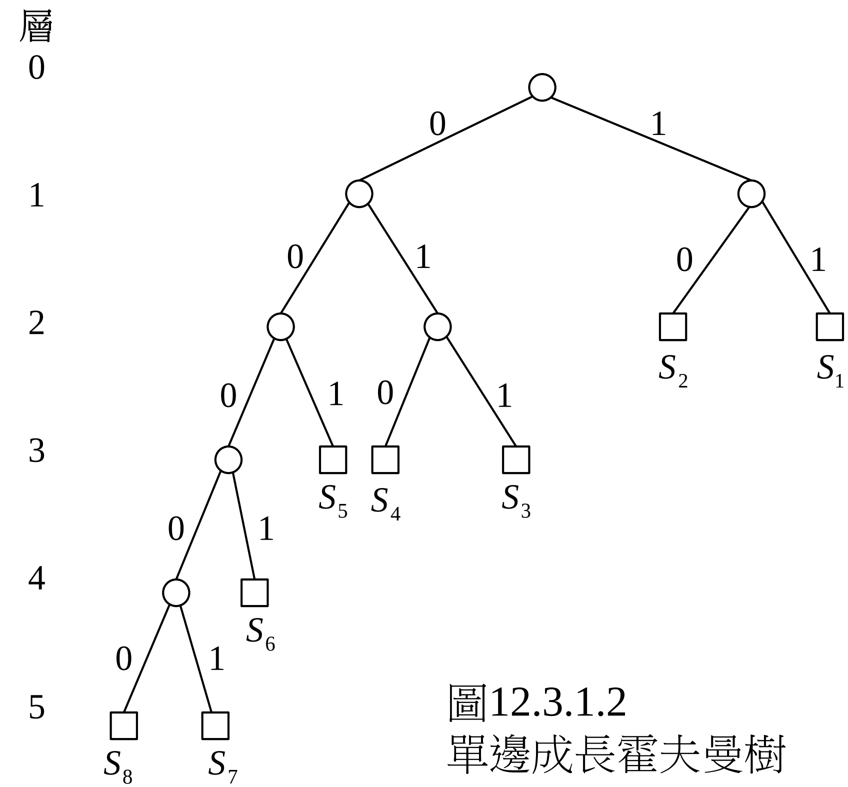


圖12.3.1.2
單邊成長霍夫曼樹

給 $H=001 \rightarrow H[1]=0$，$f_1=0$

$\rightarrow H[1..2]=00$ ，$f_2=2$
需跳過 $A[0..1]$ 中的兩個樹葉

$\rightarrow H[1..3]=001$，$f_3=3 \rightarrow A[2]=S_5$

解碼只需 $O(d)$ 的時間，*d*指的是單邊成長霍夫曼樹的深度。

速度最快的霍夫曼解碼器

在霍夫曼樹上進行廣先搜尋，在內部節點旁存上 $2l+r+1$ 的值，l代表位於同一層但在該內部節點左邊的內部節點數；r代表在同一層上，內部節點右邊的節點數。第0層到第2層形成了一個完全子樹，可利用變數$d^{'}=2$記錄這特性。儲存

$$CH\_array[0..11]=[S_7,S_8,2,3,S_4,S_5,2,S_6,2,S_3,S_1,S_2]$$

令輸入$=Huf_array[0..3]=1101$， $d^{'}=2$

$\Rightarrow$ $Huf\_array[0..1]=11$ ， $array\_ptr=3$

$\Rightarrow \begin{cases} CH\_array[3]=3 \text{ ， } code\_ptr=2 \\ Huf\_array[2]=0 \end{cases}$

$\Rightarrow \begin{cases} code\_ptr=3=2+1 \text{ ， } Huf\_array[3]=1 \\ array\_ptr=6=3+3 \text{ ， } CH\_array[6]=2 \end{cases}$

$\Rightarrow \begin{cases} array\_ptr=6+2+1=9 \\ S_3=CH\_array[9]=S_3 \end{cases}$

霍夫曼解碼可在 $O(d-d^{'})$ 的時間內完成。

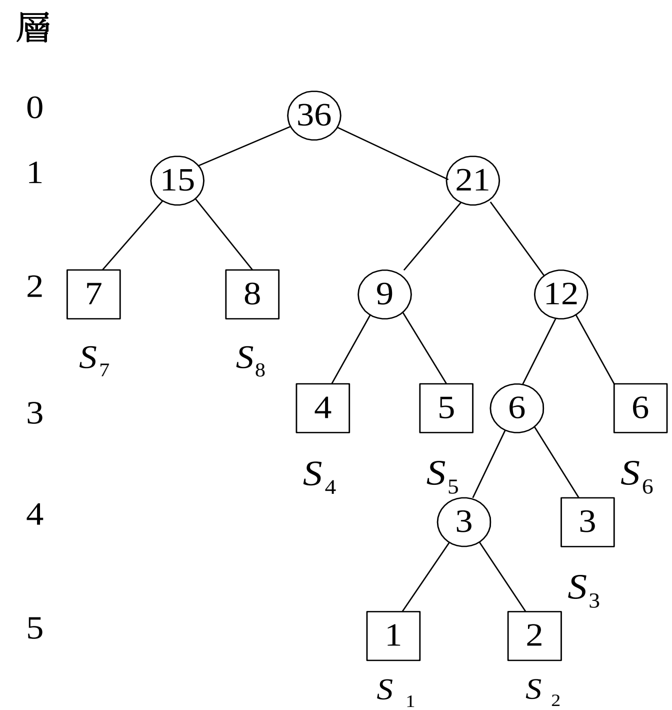
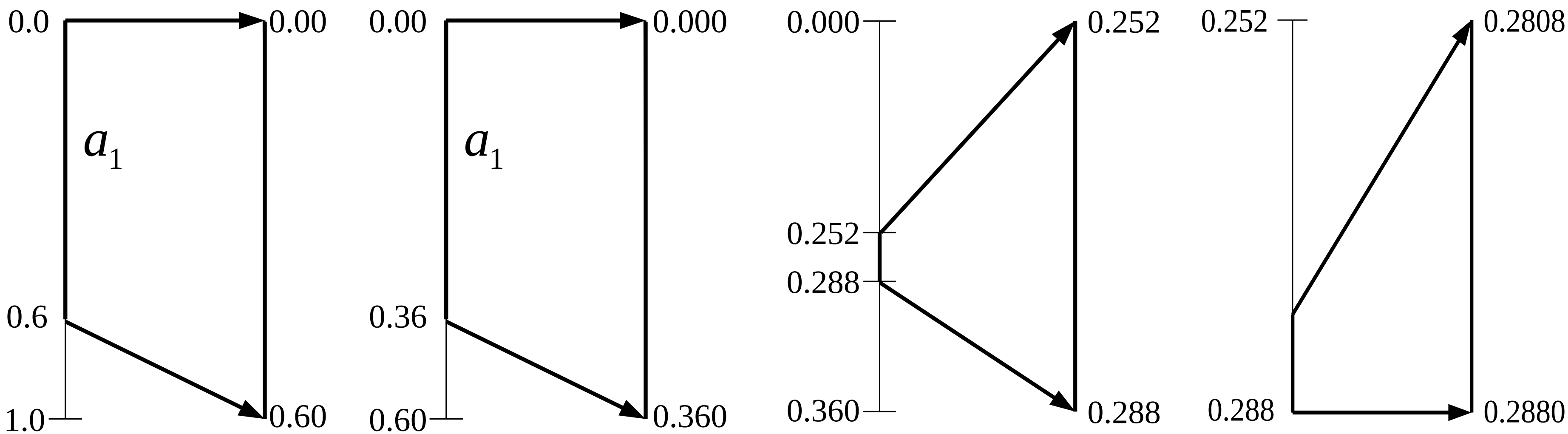


圖12.3.1.3 霍夫曼樹

12.3.2 算術碼

字母集$\{a_1,a_2,a_3,a_4\}$且字母的機率為$p(a_1)=0.6$、$p(a_2)=0.1$、$p(a_3)=0.1$和 $p(a_4)=0.2$。我們要編碼的訊息為 $a_1a_1a_3a_4$



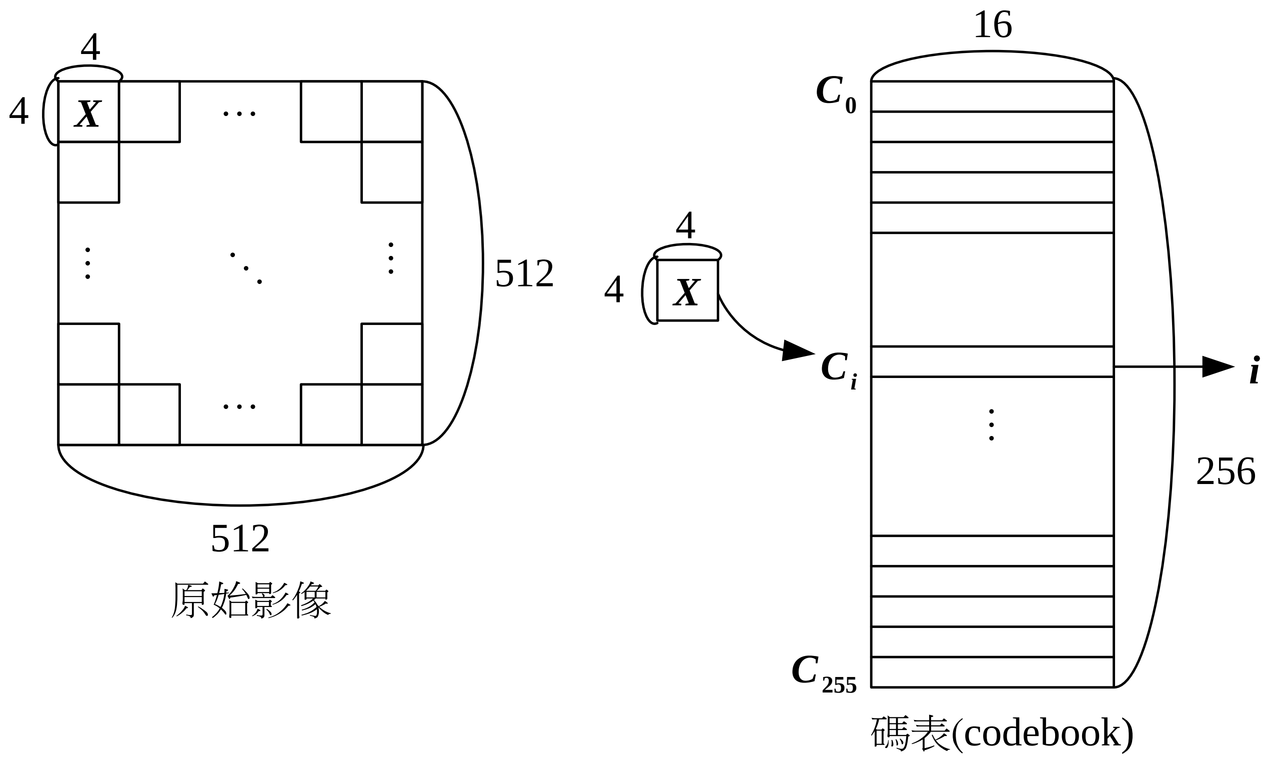
我們可用標簽的中間值0.2844表示原始之訊息。

收方收到的值是0.2844該如何解碼呢？因為0.2 $\in[0.0,0.6)$，可知第一個字母為 a_1；從0.28 $\in[0,0.36)$ 可知第二個字母亦為 a_1。最終可推得原訊息為 $a_1a_1a_3a_4$。

12.4 向量量化法

令碼表中的碼為 $\{C_i \mid 0 \le i \le 255\}$ 而待搜尋的區塊向量為 X，目標為找到 C_i 使得 $d^2(X, C_i) = \min_j \sum_{n=1}^{16} (X_n - C_{jn})^2$

這裏 $X = (X_1, X_2, \cdots, X_{16})$， $C_j = (C_{j1}, C_{j2}, \cdots, C_{j16})$。



- 金字塔式向量搜尋法

給二非負整數x和y

➔ $2x^2+2y^2-(x+y)^2 \;=x^2-2xy+y^2 \;=(x-y)^2 \geq 0$

➔ $2(\frac{x+y}{2})^2 \leq x^2+y^2$

對任意向量 $a=(a_1,a_2,\ldots,a_{16})$，令

$$f_1(a)=\left((\frac{a_1+a_2}{2}),(\frac{a_3+a_4}{2}),\ldots,(\frac{a_{15}+a_{16}}{2})\right) \quad ➔ \quad 2\| f_1(a)\|_2^2 \leq \| a\|_2^2$$

再令 $f_k(a)=f_1(f_{k-1}(a))$，$2\leq k\leq p$，

$$
\begin{aligned}
➔ \quad 2^p\left\|f_p(a)\right\|_2^2 &\leq 2^{p-1}\left\|f_{p-1}(a)\right\|_2^2 \\
&\vdots \\
&\leq 2\left\|f_1(a)\right\|_2^2 \\
&\leq \left\|a\right\|_2^2
\end{aligned}
$$

回到VQ的方法上，令 $a = X - C_i$

➔ $2^p 2^d (f_p(X), f_p(C_i))$

$$\begin{aligned} &\le 2^{p-1} d^2 (f_{p-1}(X), f_{p-1}(C_i)) \\ &\vdots \\ &\le 2d^2(f_1(X), f_1(C_i)) \\ &\le d^2(X, C_i) \end{aligned}$$

若每四個元素縮成一個平均值

➔ $4^q d^2 (f_q(X), f_q(C_i))$

$$\begin{aligned} &\le 4^{q-1} d^2 (f_{q-1}(X), f_{q-1}(C_i)) \\ &\vdots \\ &\le 4d^2(f_1(X), f_1(C_i)) \\ &\le d^2(X, C_i) \end{aligned}$$

其中q表示金字塔的高度。

不等式中，$f_1(x)$為 X縮小 1/4 後的上一層之向量，而$f_1(C_i)$ 為C_i 的上一層之向量，這裡 X和 C_i皆為最底層的向量。

每一個 C_i皆事先建好自己的金字塔。X也建出屬於自己的金字塔。

計算兩金字塔的頂端的對應值，即 $4^q d^2(f_q(X), f_q(C_i))$。若計算得到的值比目前暫時的最小值都來的大時，則 C_i 就不必再往金字塔的下層考慮了。

12.5 單張影像壓縮

JPEG：將影像切割成 8×8 的子影像集；
將RGB影像轉換為YC_bC_r影像。

(1) 將DCT作用在 8×8 的子影像上：每一像素皆先減去128，以下列的計算完成DCT

$$F(u,v)=\frac{1}{\sqrt{16}}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

79	75	79	82	82	86	94	94
76	78	76	82	83	86	85	94
72	75	67	78	80	78	74	82
74	76	75	75	86	80	81	79
73	70	75	67	78	78	79	85
69	63	68	69	75	78	82	80
76	76	71	71	67	79	80	83
72	77	78	69	75	75	78	78

(a) 8×8子影像

619	-29	8	2	1	-3	0	1
22	-6	4	0	7	0	-2	-3
11	0	5	-4	-3	4	0	-3
2	-10	5	0	0	7	3	2
6	2	-1	-1	-3	0	0	8
1	2	1	2	0	2	-2	-2
-8	-2	-4	1	2	1	-1	1
-3	1	5	-2	1	-1	1	3

(b) 8×8係數矩陣

圖12.5.1
經DCT 作用後的結果

(2)將第(1)步驟所得的頻率域值除以8 × 8量化表(Quantization Table)

圖12.5.2
量化表與量化後的結果

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

(a) 8 × 8量化表

39	-3	1	0	0	0	0	0
2	-1	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	0

(b) 8 × 8量化後DCT係數矩陣

(3)將第(2)步驟所得的結果四捨五入以取整數

圖12.5.2(b)的DCT係數矩陣經IDCT(Inverse DCT)

$$f(x,y)=\frac{1}{\sqrt{16}}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

作用後，可得解壓後的影像，如圖12.5.3所示。

74	75	77	80	85	91	95	98
77	77	78	79	82	86	89	91
78	77	77	77	78	81	83	84
74	74	74	74	76	78	81	82
69	69	70	72	75	78	82	84
68	68	69	71	75	79	82	85
73	73	72	73	75	77	80	81
78	77	76	75	74	75	76	77

圖12.5.3 8 × 8解壓後影像

(4)依據Zig-Zag的掃描次序，將第(3)步驟所得的結果依低頻為先的原則，圖12.5.2(b)的向量型式為(39,-3,2,1,-1,1,0,0,0,0,0,-1,0,0,0, … ,0,0,0)。

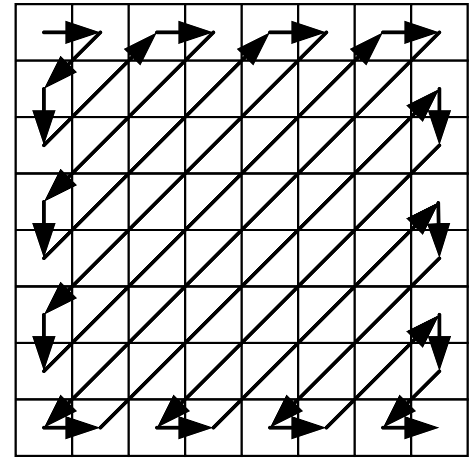


圖12.5.4 Zig-Zag掃描次序

(5)針對AC進行Run-Length編碼

可編碼為(0,-3)(0,2)(0,1) (0,-1)(0,1)(5,-1)EOB

在Run-Length編碼的格式(x,y)中，x 通常採用固定長度編碼，而y則依照事先建好的圖12.5.5表進行變動長度編碼。上述的向量型式進一步編成(0,2)(00)(0,2)(10)(0,1)(1)(0,1)(0)(0,1)(1)(5,1)(0)

位元數	y的範圍
0	0
1	-1,1
2	-3,-2,2,3
3	-7,...,-4,4,...,7
4	-15,...,-8,8,...,15
:	:
:	:

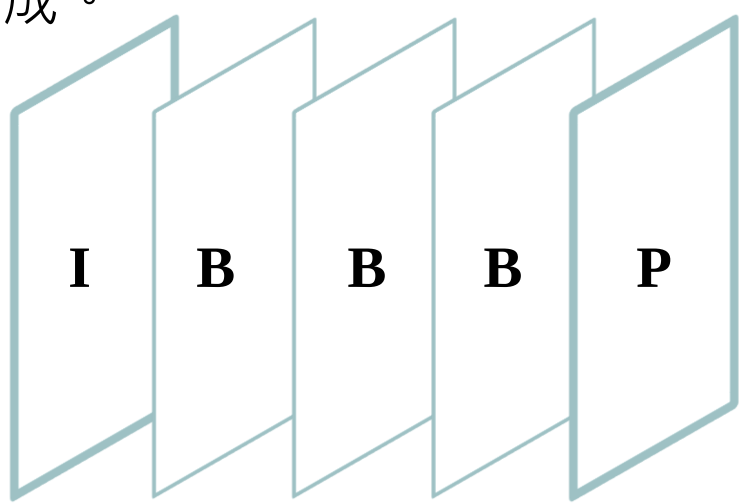
圖12.5.5 y的編碼對照表

(6)進行DPCM(Differential Pulse Code Modulation)和霍夫曼編碼(Huffman Encoding)

12.6 視訊壓縮

視訊壓縮 (Video Compression) 中，例如H.264/AVC和HEVC，我們先將視訊影像分成三類，分別為 I 、 P 和 B 影像。

- I 影像用Intra Mode壓縮即可。
- P 影像可利用前面的 I 影像，透過區塊匹配 (Block Matching) 和補償 (Compensation) 來壓縮。
- 夾在 I 和 P 之間的 B 影像之區塊就由 I 和 P 所匹配到的區塊內插而成。



12.6.1畫面間區塊匹配

- 區塊匹配是計算最花時間的部分。
- 區塊匹配是在前一張參考影像中找到某一區塊，使得找到的區塊和目前區塊最匹配。通常是採用在前張影像中先訂出一個搜尋視窗，在這搜尋視窗內包含許多與目前區塊相同大小的正方形區塊。因此進行區塊匹配前得先決定搜尋的範圍和區域。

- 假設目前區塊為 B_c，西邊鄰近區塊、西北邊鄰近區塊和北邊鄰近區塊會用來產生 B_c 的初始移動向量。接著，利用初始移動向量所得的區塊 B'_r，計算兩者的絕對差平均值(Mean Absolute Difference, MAD):

$$\mathrm{MAD}(B_c, B_r^{'}) = \frac{1}{N^2}\sum_{x=1}^{N}\sum_{y=1}^{N}\left|B_c(x,y) - B_r^{'}(x,y)\right|$$

  若得到的值很大，則 B_c 屬於高移動區塊，搜尋視窗為原始搜尋範圍。若是中等的值，則屬於中移動區塊，搜尋範圍為原始搜尋範圍的一半。否則屬於低移動區塊，搜尋範圍則為1/4的原始搜尋範圍。

- 應用到全搜尋(Full Search)演算法後，有60%以上的時間改良率。估計精確度和全搜尋演算法則是差不多。

- [24]實際分析圖12.6.1.2所示的五種視訊檔中的機率，給出類型配對和搜尋範圍間更合理的建議。而為了節省乘法和除法的計算，以累計絕對差 (Accumulated Absolute Difference, AAD) 當作區塊間的相似量度。定義如下：

$$\mathrm{AAD}(v_x, v_y) = \sum_{x=1}^{N}\sum_{y=1}^{N} \left| B_c(x, y) - B_r(x+v_x, y+v_y) \right|$$

由式子可知，對每個目前區塊算出參考影像中最匹配的區塊，然後紀錄 $D = \max(|v_x|, |v_y|)$。

根據實驗，發現*D*=4時，幾乎涵蓋大多數的最大絕對值位移。圖12.6.1.2為五種視訊檔的不同絕對位移分佈圖。

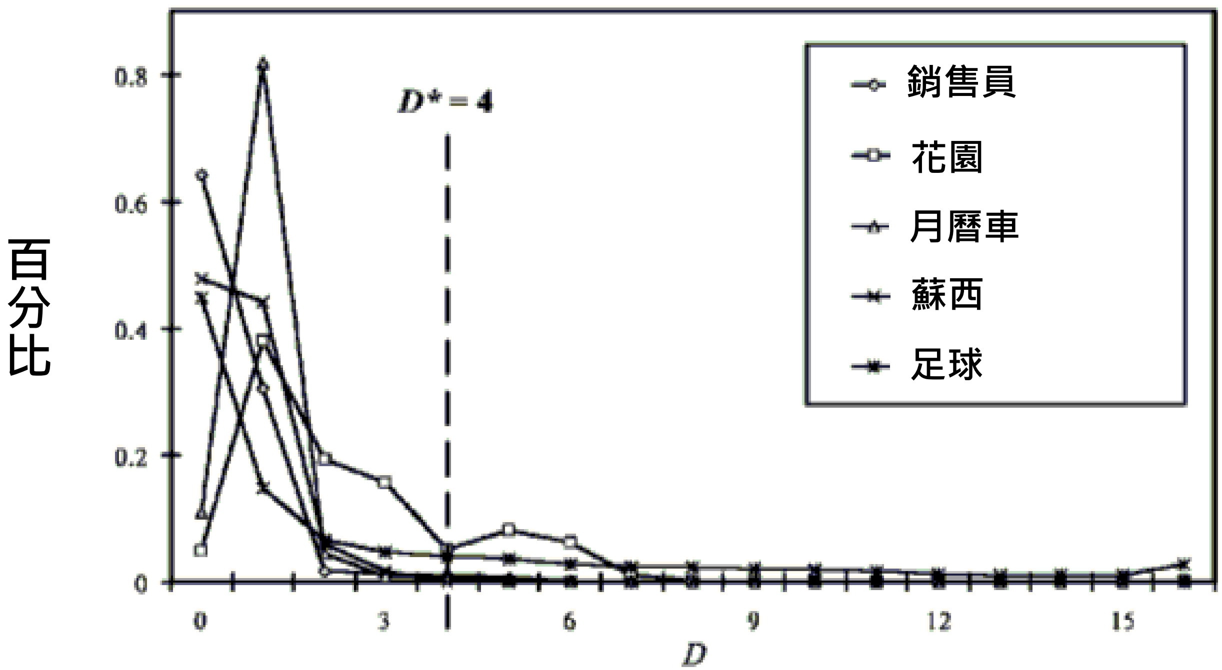


圖12.6.1.2　五種視訊檔的不同*D*分佈圖

令 $\Pr_{j,l}(D=i)$代表在視訊檔*l*中的第*j*張影像中隨機變數*D*的機率值。*D*的平均機率可表示為

$$\frac{1}{n_l}\sum_{j=2}^{n_l}\Pr_{j,l}(D=i)$$

針對五種視訊檔，圖12.6.1.3分別列出它們的*D*之平均機率。當 $0\le D\le 4$，把五種視訊的*D*之平均機率疊加起來，得

$$\frac{1}{5}\sum_{l}\sum_{i=0}^{4}\frac{1}{n_{l-1}}\sum_{j=2}^{n_l}\Pr_{j,l}(D=i)=91.17\%$$

由上式可知當*D*小於等於4時，平均的疊加機率高達91.17% 。令 $D^*=4$，這個值在決定最低搜尋範圍時會用到。

圖12.6.1.4　一個例子

(0,0) D=0	(0,0) D=0	(0,0) D=0	(0,0) D=0	(1,1) D=1
(0,0) D=0	(0,0) D=0	(0,0) D=0	(1,1) D=1	(1,0) D=1
(0,0) D=0	(1,0) D=1	(3,2) D=3	(2,0) D=2	(2,0) D=2
(1,0) D=1	(3,1) D=3	(5,3) D=5	(6,2) D=6	(2,1) D=2
(1,0) D=1	(2,1) D=2	(3,2) D=3	(2,2) D=2	(2,1) D=2

(a)參考影像

(3,2) B_1	(5,3) B_2	(4,1) B_3
(4,2) B_4	B_c	

(b)目前影像

假設在視訊檔 I 中的第 j 張影像已被分割成5×5個區塊，見圖12.6.1.4(a) ，圖中的(x,y)代表該區域的移動向量(Motion Vector)值。圖12.6.1.4(b) 這四個鄰近區塊的移動向量之平均值可用來預測 B_c 的初始移動後的 B_c 。

由圖12.6.1.4(a)可算得 $\Pr_{j,i}(D=0)$，$\Pr_{j,i}(D=1)$，$\Pr_{j,i}(D=2)$，$\Pr_{j,i}(D=3)$，$\Pr_{j,i}(D=4)$，$\Pr_{j,i}(D=5)$，$\Pr_{j,i}(D=6)$。利用式子

$$T_j = \min\{D^*, \arg\min_{T} \sum_{i=0}^{T} \Pr_{j,l}(D=i) \ge 0.9117\}$$

可得到 $T_j = 3 = \min\{4,3\}$。

- 完全搜尋(Full Search)

我們假定在搜尋範圍中的某一搜尋正方形，如圖12.6.1.5所示的中間較小框框 $\overline{w}$。若B_c在$\overline{w}$ 內找到最匹配的區塊，則完成匹配。否則在 $\overline{w}$ 的邊緣上找一暫時最匹配的區塊所在處定一3×3的視窗如圖12.6.1.5中以A點為中心的視窗。接著在這小視窗中找B_c的最佳匹配區塊。直到找到最佳的匹配區塊為止。

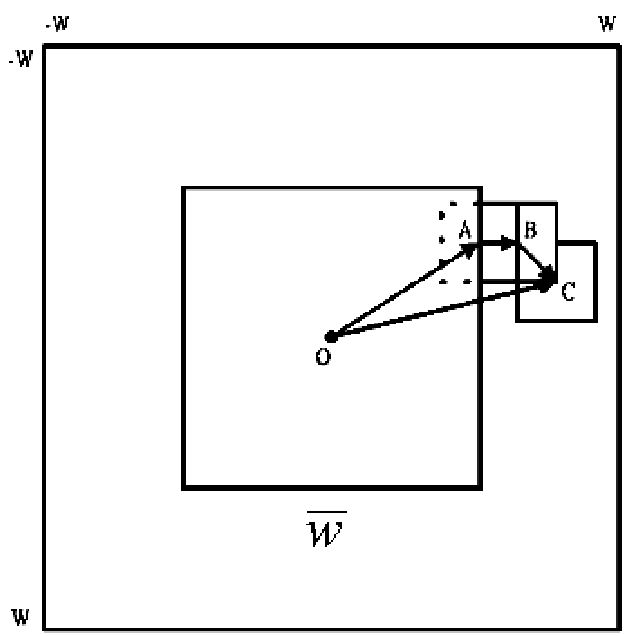


圖12.6.1.5　區塊匹配

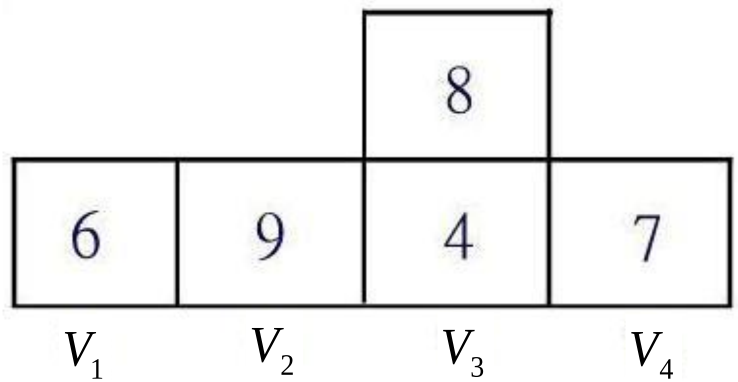
- 贏家修正策略(Winner-update Starategy)

沿用12.4節中的符號 X和 C_i，$1 \le i \le 256$，但 x 為目前影像中的待匹配區塊，而$\{C_i\}$為前一張參考影像中在搜尋範圍內的所有區塊，這裡假設十六維的向量 $V_i = |x - C_i|$，$1 \le i \le (2W+1) \times (2W+1)$ $V_i(j) = |x(j) - C_i(j)|$，$1 \le j \le 16$ 。為方便說明，假設 $1 \le i \le 4$，且各個向量只有三維。

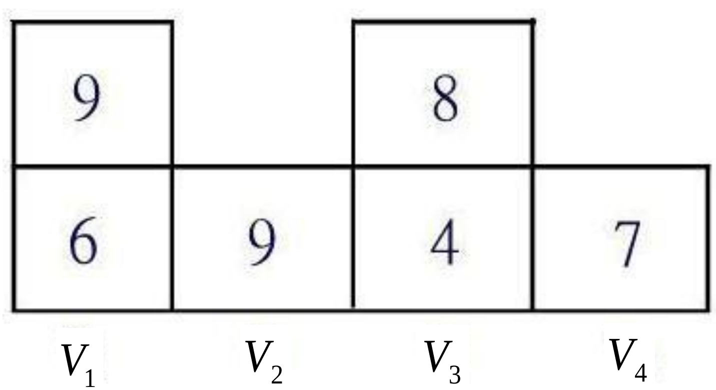
首先，我們檢查 V_1、V_2、V_3和 V_4的第一個元素，假如四個元素如下所示

6	9	4	7
V_1	V_2	V_3	V_4

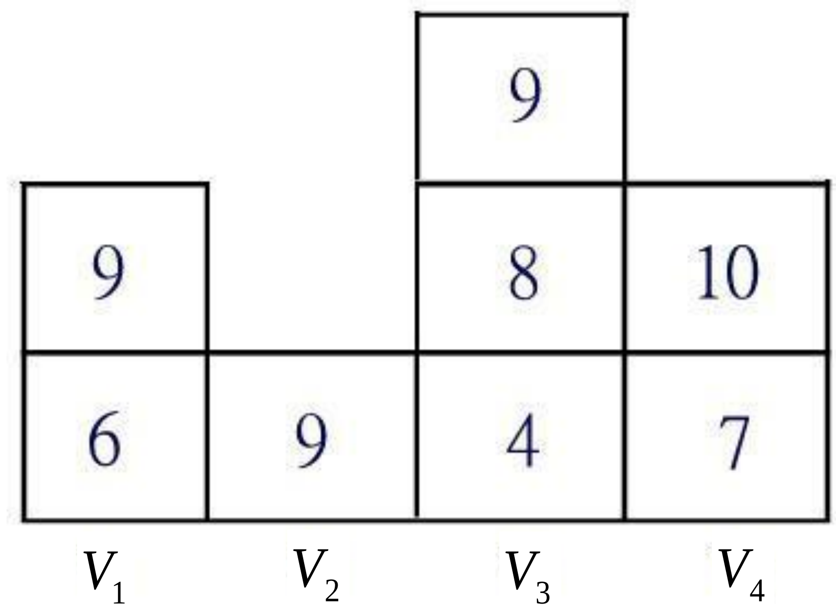
因為$V_3(1)=4$為最小者，就繼續看$V_3(2)$並且計算出$V_3(1)+V_3(2)$，假定目前的前置和如下所示



這時因為$V_1(1)=6$為前置和中最小值，所以計算$V_1(2)=V_1(2)+V_1(1)$ =3+6+9。目前的前置和如下所示



重複同樣的方式，假設最終的前置和如下所示



則由 V_3 可知 x 和 C_3 最匹配。

12.6.2畫面內預測模式

- 在 H. 264/AVC 中，畫面內預測分成 4× 4、 8× 8、 16× 16 三種不同大小的子區塊，其中 4× 4 和 8× 8 亮度區塊預測模式分成了垂直 (Vertical)、 水平(Horizontal)、 DC、 左下對角 (Diagonal Down-left)、 右下對角 (Diagonal Downright)、 右垂直 (Vertical-right)、下水平 (Horizontal-down)、 左垂直 (Vertical-left)和上水平 (Horizontal-up) 一共九種預測模式。

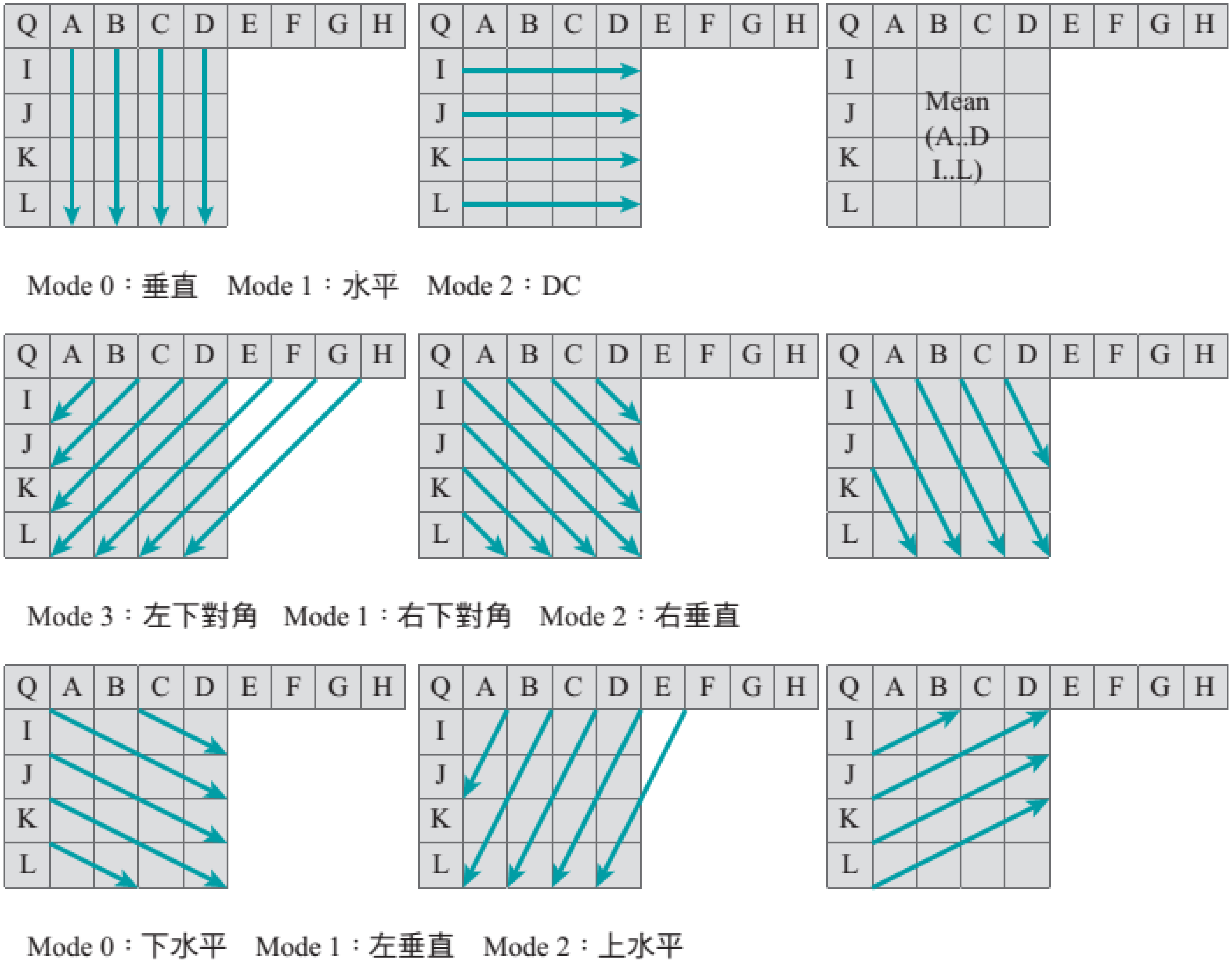


圖12.6.2.1 4x4區塊的九種畫面內的預測模式

在 16× 16 區塊的預測模式中，共分成垂直 (Vertical)、水平 (Horizontal)、DC、 平面 (Plane) 四種預測模式，如圖 12.6.2.2 所示。基本上， H. 264/AVC 會將 Inter Mode 和 Intra Mode 都做一遍，再從中挑 **RD** (Rate-Distortion) 花費較少的 Mode 為準。不管是 Inter Mode 或是 Intra Mode，系統都會計算殘量 (Residue) 的壓縮花費。

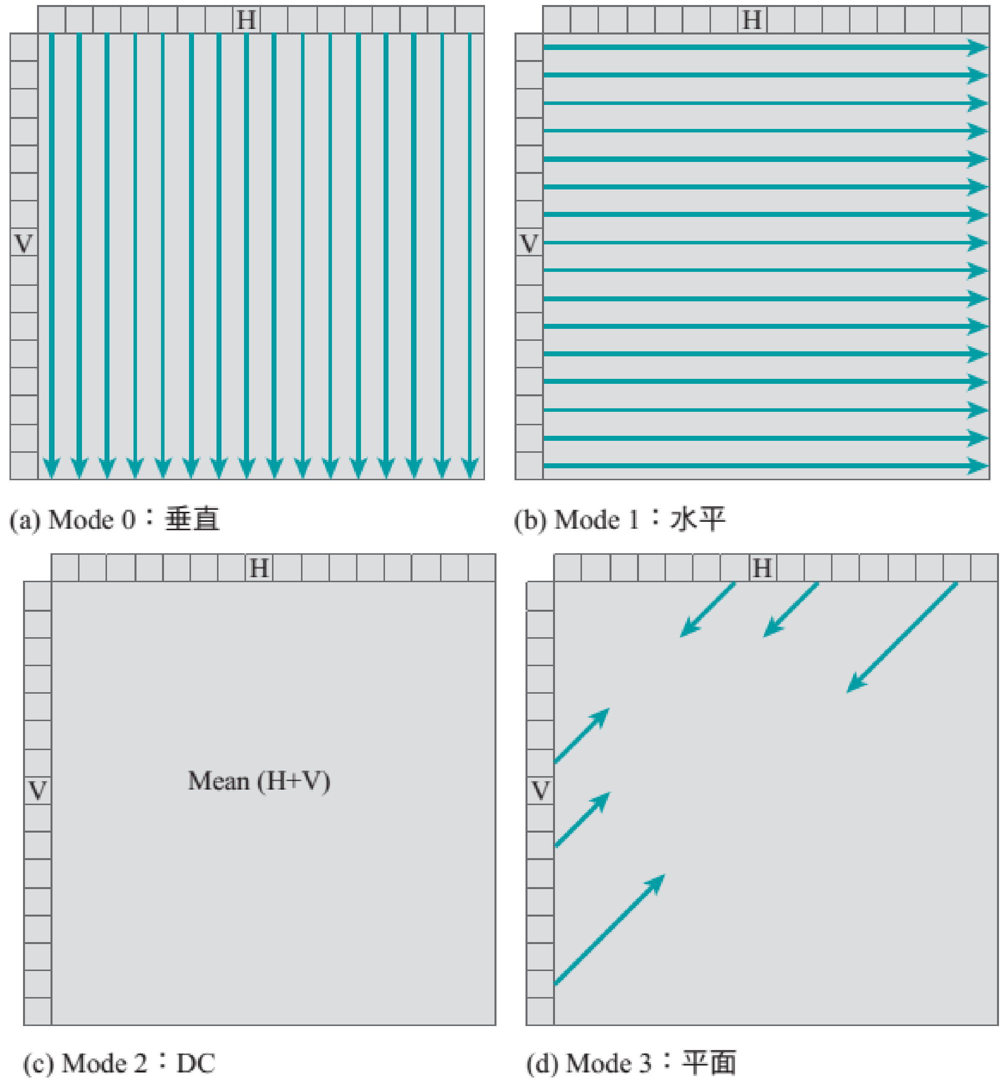


圖12.6.2.2 16x16 的四種畫面內的預測模式

■HEVC (High Efficiency Video Coding) 中，畫面內亮度區塊預測模式增加為 35 種，以藉此提升高解析度影片的壓縮效率，其中包含 DC、 Planar 及 33 種不同的預測方向。預測模式的編號與對應的預測方向表示在圖 12.6.2.3 中。預測區塊增加成 4× 4、 8× 8、 16× 16、 32× 32、64× 64 五種子區塊，不同區塊大小所使用的預測模式數目也不相同，參見圖 12.6.2.4。

區塊大小	畫面內預測模式數目
4×4	35
8×8	35
16×16	35
32×32	35
64×64	35

圖12.6.2.4

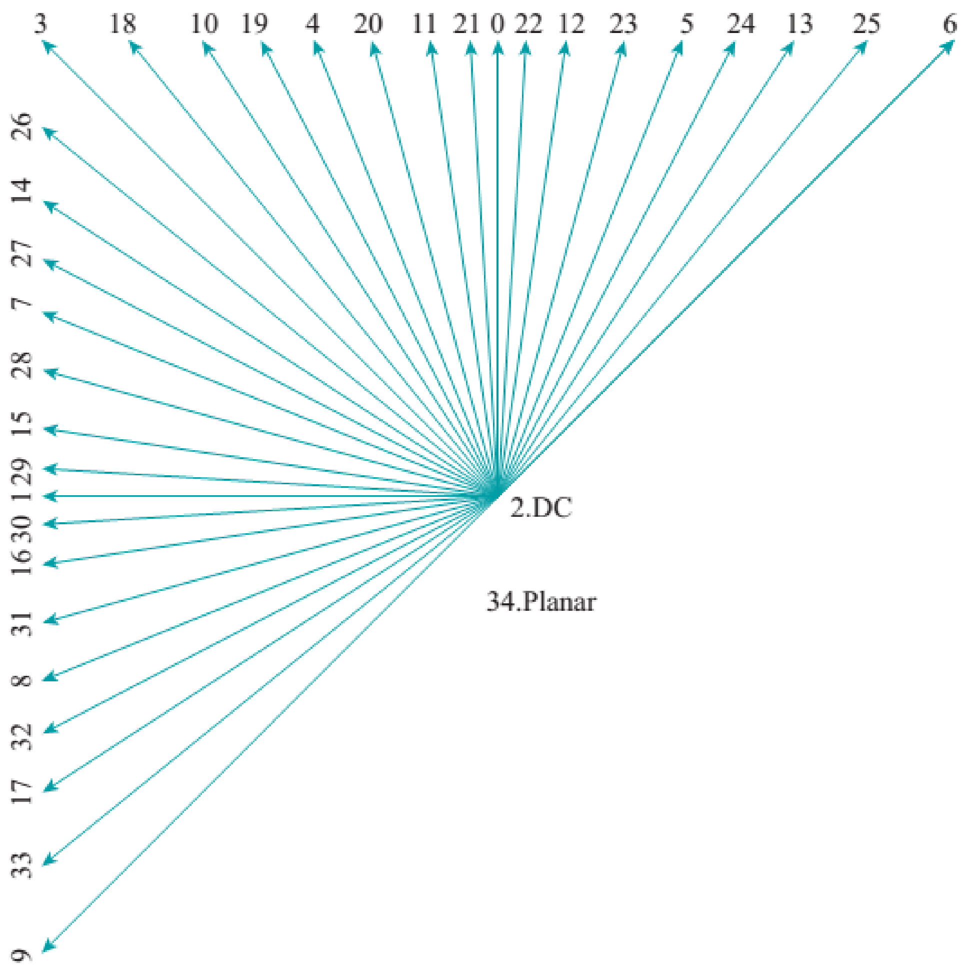


圖12.6.2.3
HEVC的35種
預測方式

■在 HEVC 中新增的 Planar 預測模式。

4× 4 的白色區塊為待預測區塊。綠色的已解碼像素當參考像素。預測 (2, 2) 位置的像素值為例，水平方向的預測是以 (4, - 1) 參考像素當作圖中 A 位置的像素值，再將 A 與考像素 (- 1, 2) 以線性內插法計算出 (2, 2) 的水平預測值；而後再計算垂直方向的預測值，將 (- 1, 4) 參考像素作為圖中 B 位置的像素值，再以像素 B 與 (2, - 1) 參考像素用線性內插法計算出目前像素的垂直預測值。最後，再將前面得到的水平預測值與垂直預測值取平均得到目前(2, 2) 位置的預測像素值。

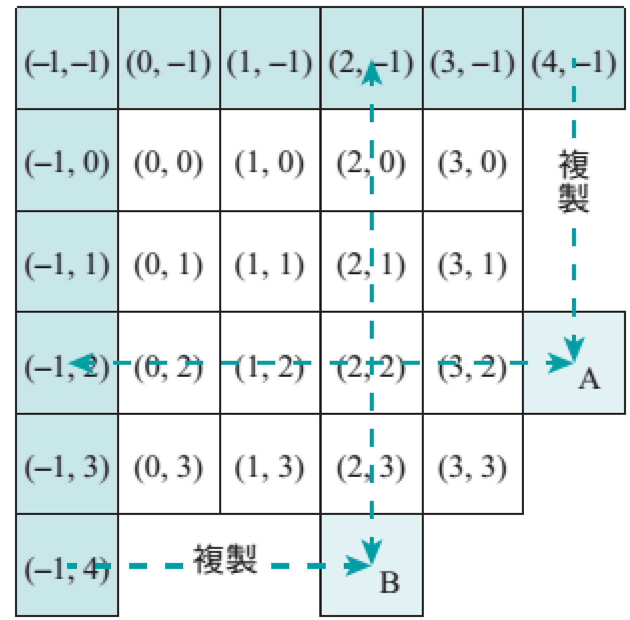


圖12.6.2.5 Planar 預測模式