有損失壓縮演算法

- 緒論
- 預測編碼法
- 量化
- 向量量化
- 轉換編碼

緒論

- 利用有損失壓縮演算法無法得到夠高的壓縮率，因此大部份的多媒體壓縮法是**有損失**的
- 何謂有損失壓縮演算法（ lossy compression ）
  - 壓縮後之資料與原資料並不相同，而是與原資料相似
  - 可得到比無損失壓縮法更高的壓縮率

失真量測

- 三種最常用於影像壓縮的方法分別為：
  - 均方差（MSE），σ^2

    $$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2$$

    其中 x_n，y_n 及N分別為原始資料串、解壓縮後的資料串及資料串的長度。
  - 訊雜比（SNR），單位為「單位分貝」（dB）

    $$SNR = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2}$$

    其中σ_x^2 為原始資料串的平均值的平方而 σ_d^2 為MSE值
  - 峰值訊號與雜訊比（PSNR）

    $$PSNR = 10\log\frac{x_{peak}^2}{\sigma_d^2}$$

預測編碼法

- Make use of the past history of the data being encoded to provide more efficient compression.
- For example:

Original sequence

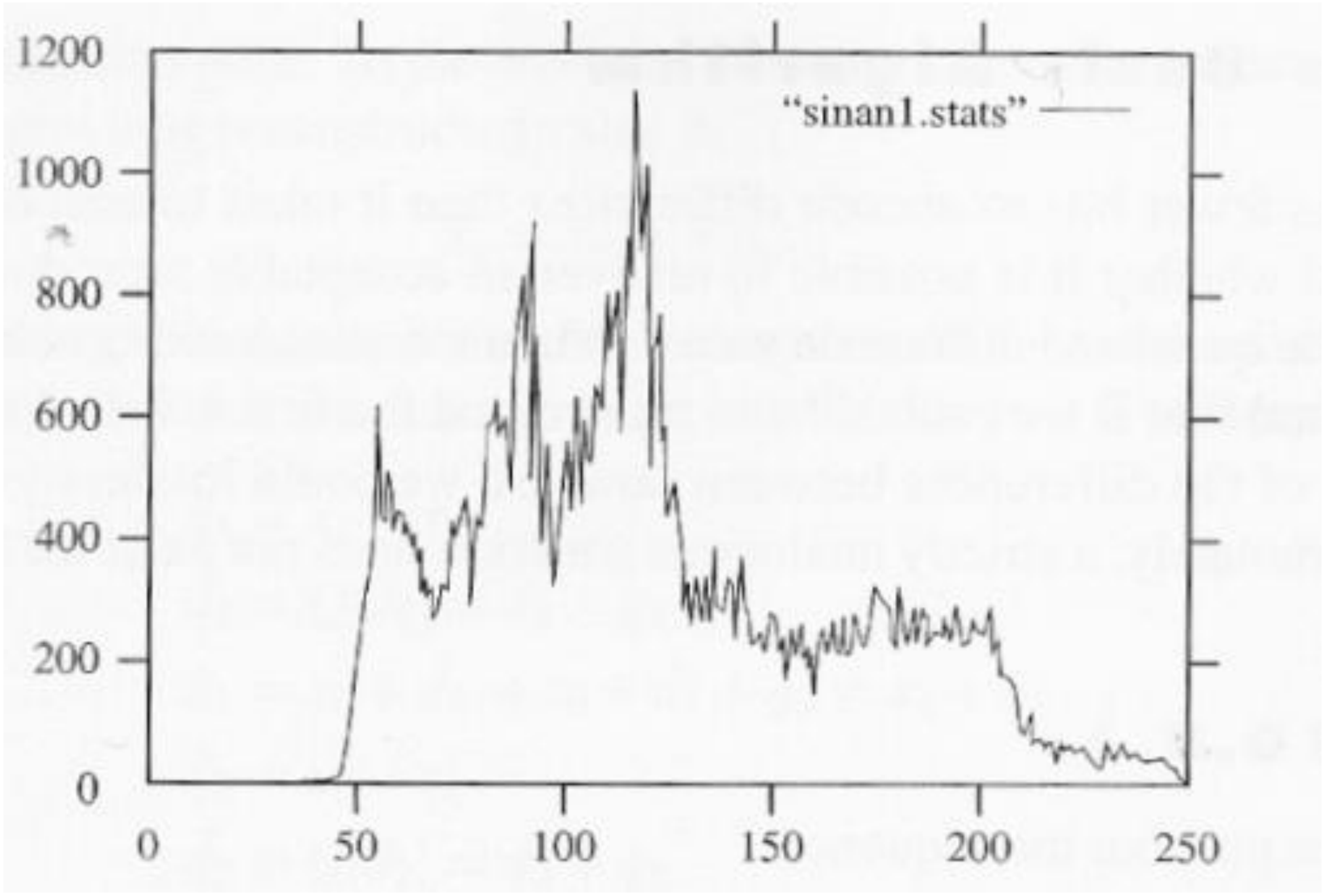
Prediction: Add 2 to the previous number and find the residual.

1	2	5	7	2	-2	0	-5	-3	-1	1	-2	-7	-4	-2	1	3	4
	3	4	7	9	4	0	2	-3	-1	1	3	0	-5	-2	0	3	5
	-1	1	0	-7	-6	0	-7	0	0	0	-5	-7	1	0	1	0	-1

Transmitted sequence (residual sequence)

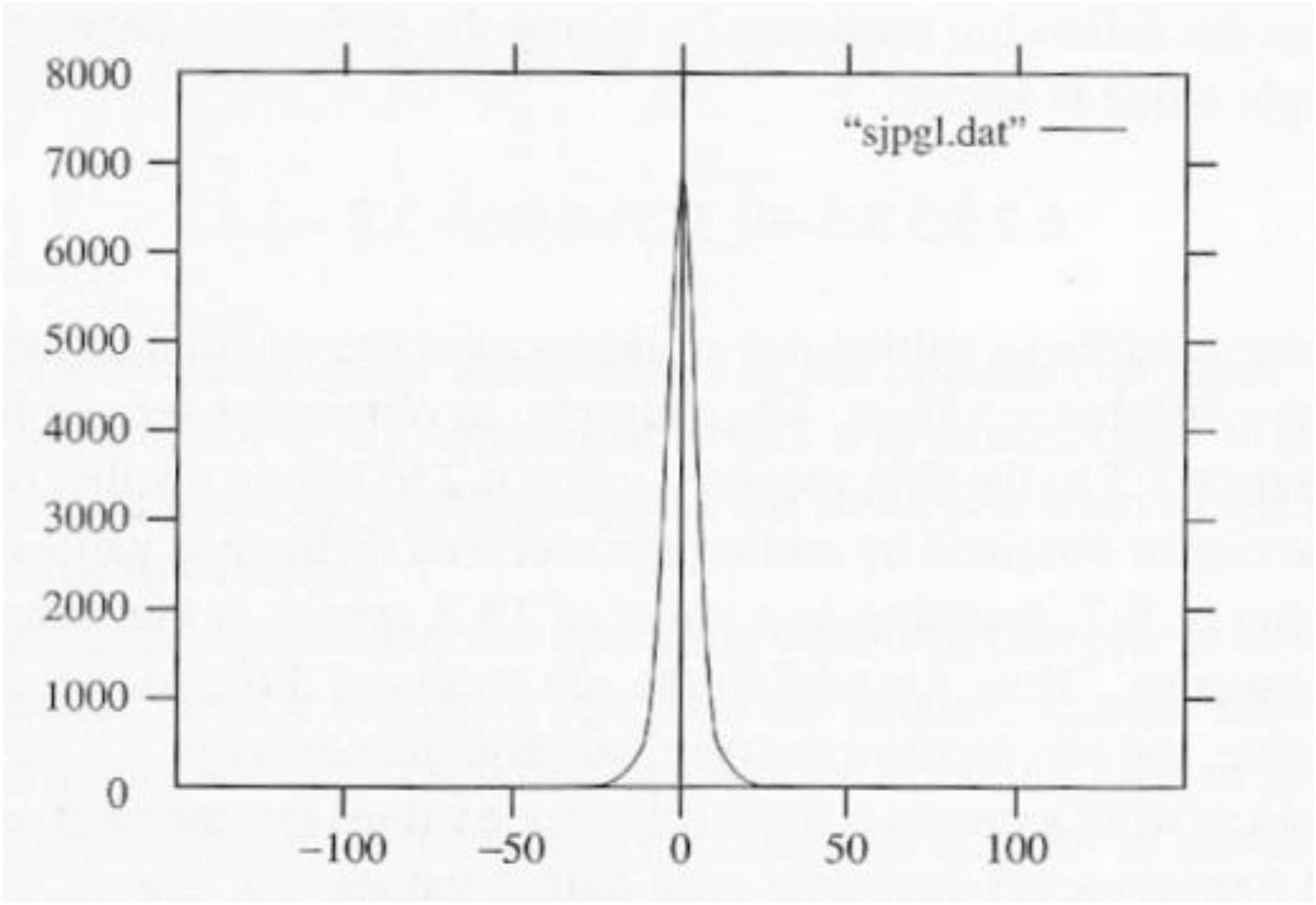
Differential encoding

- Histogram of the Sinan image



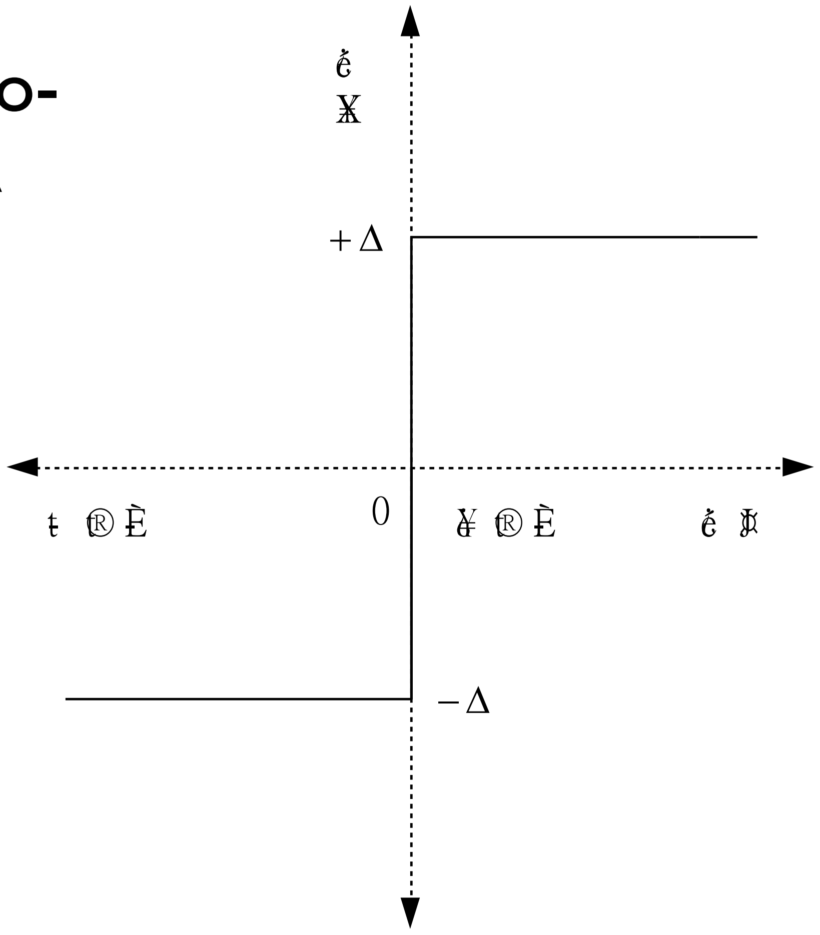
Differential encoding

- Histogram of pixel-to-pixel differences of the Sinan image

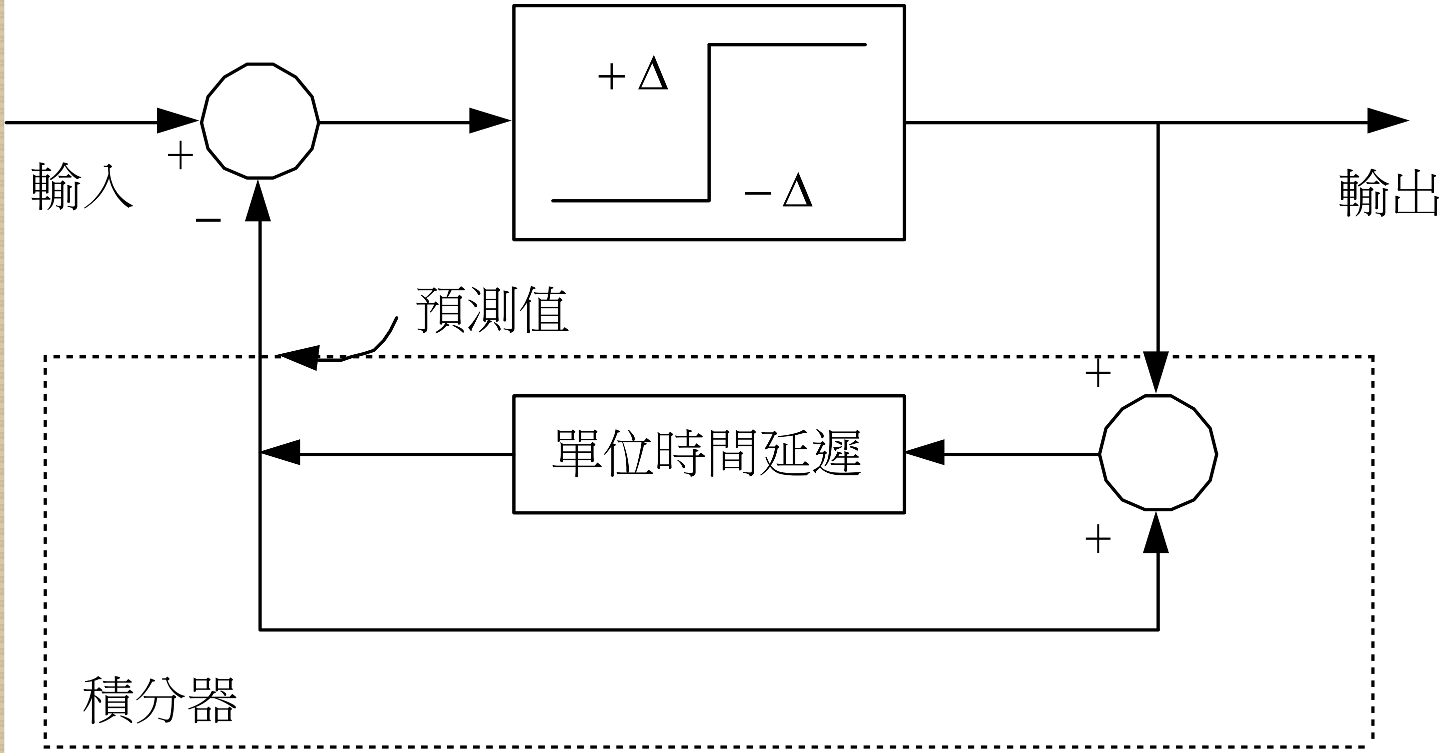


Delta Modulation (DM)

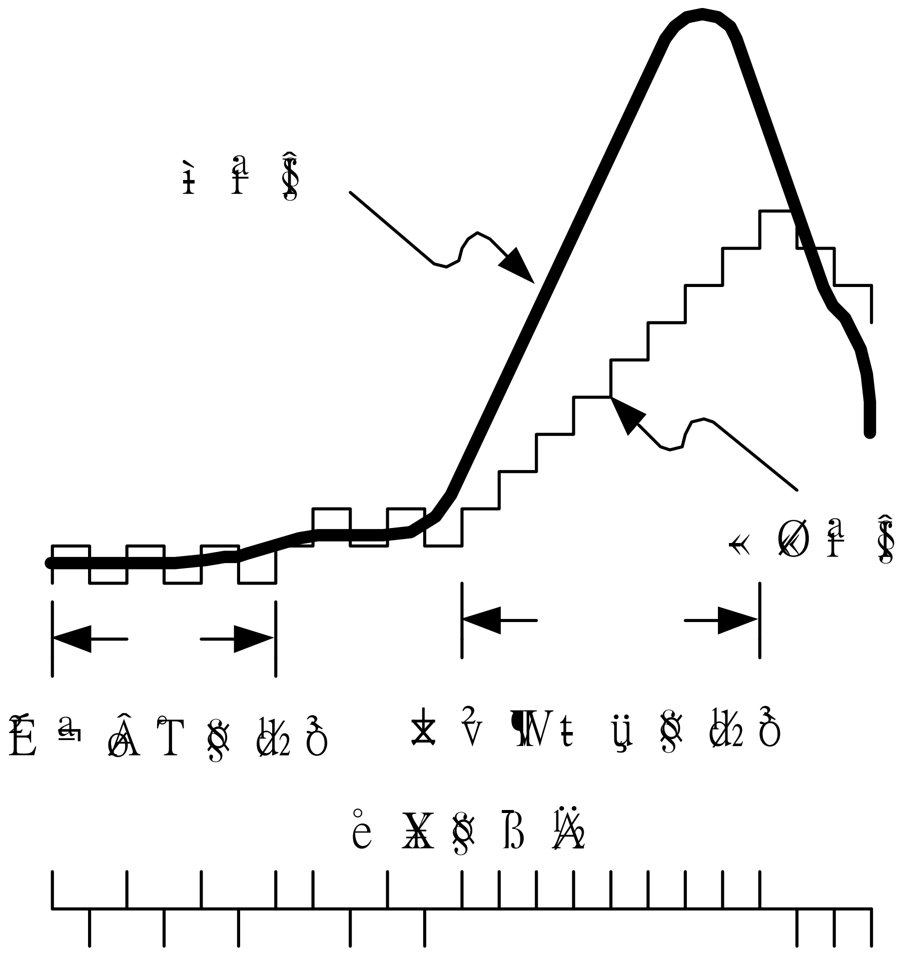
- To represent a sample-to-sample difference of $\pm\Delta$ with a 1-bit (two-level) quantizer.
- For example:
  - Input sequence
    - 3,4,5,6,4,3,3,4,5
  - Output sequence
    - 3,1,1,1,-1,-1,-1,1,1
  - Reconstruct
    - 3,4,5,6,5,4,3,4,5



Delta Modulation



Delta Modulation



Adaptive Delta Modulation

$$E_{k+1} = sign\ of\left[S_{k+1} - X_k\right]$$

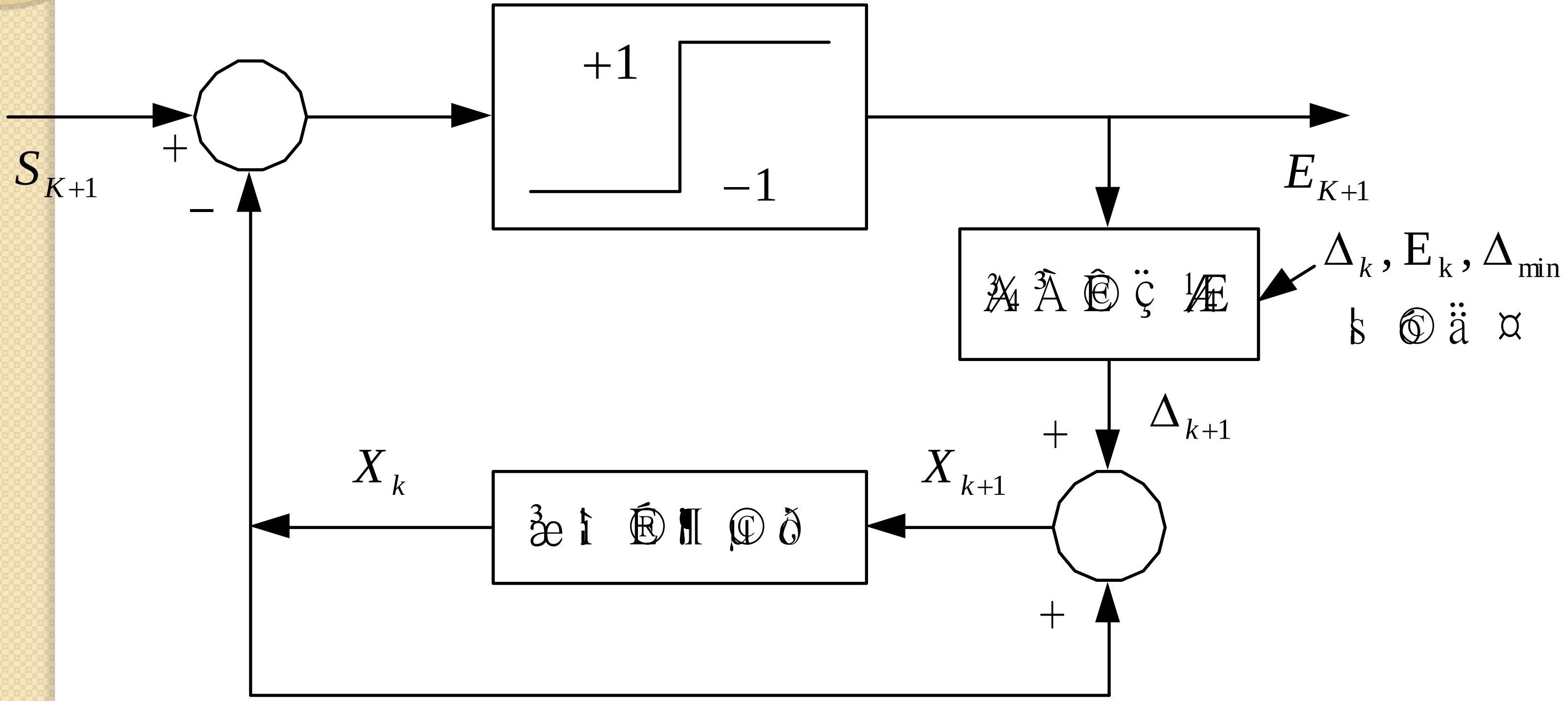
$$\Delta_{k+1} = \begin{cases} \left|\Delta_k\right|\left[E_{k+1} + \frac{1}{2}E_k\right] & if\left|\Delta_k\right| \geq \Delta_{\min} \\ \Delta_{\min}E_{k+1} & if\left|\Delta_k\right| < \Delta_{\min} \end{cases}$$

$$X_{k+1} = X_k + \Delta_{k+1}$$

S_{k+1}為輸入值

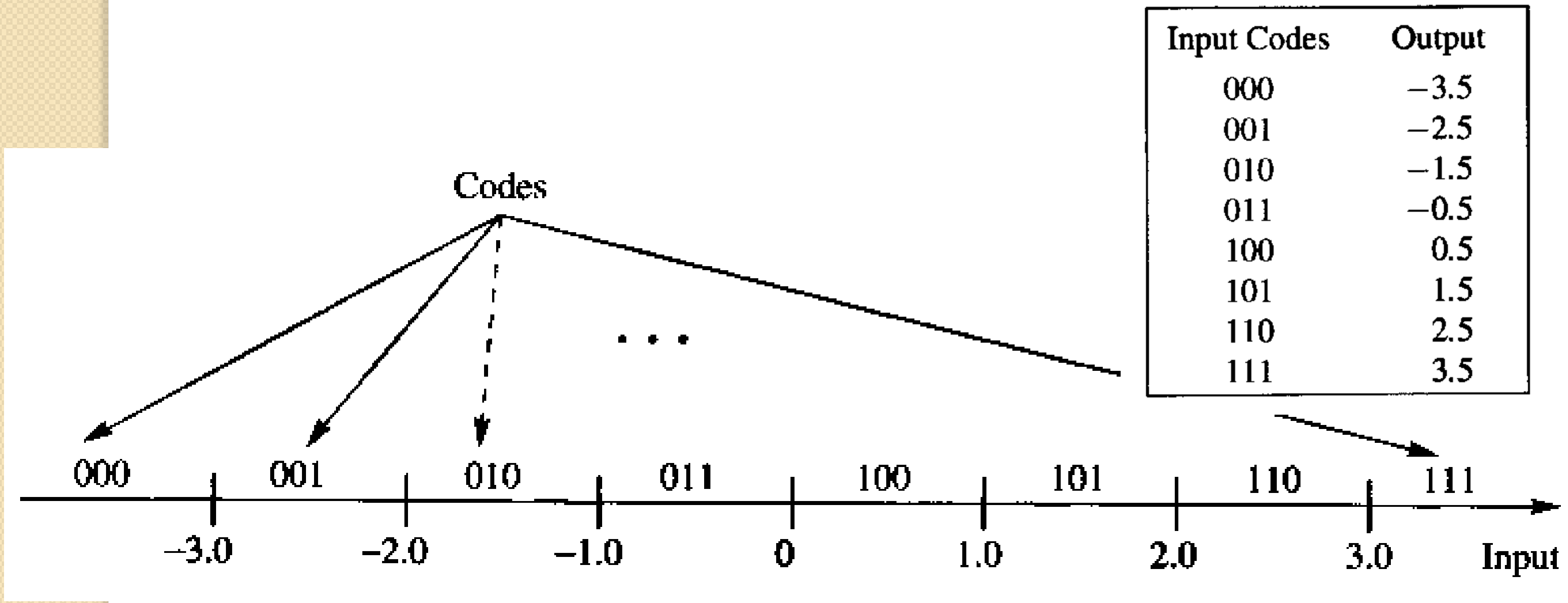
$\Delta_{\min}$ 為允許之最小Δ值

Adaptive Delta Modulation



量化

- To represent a large set of values with a much smaller set.



Input Codes	Output
000	−3.5
001	−2.5
010	−1.5
011	−0.5
100	0.5
101	1.5
110	2.5
111	3.5

量化

- Uniform quantizer

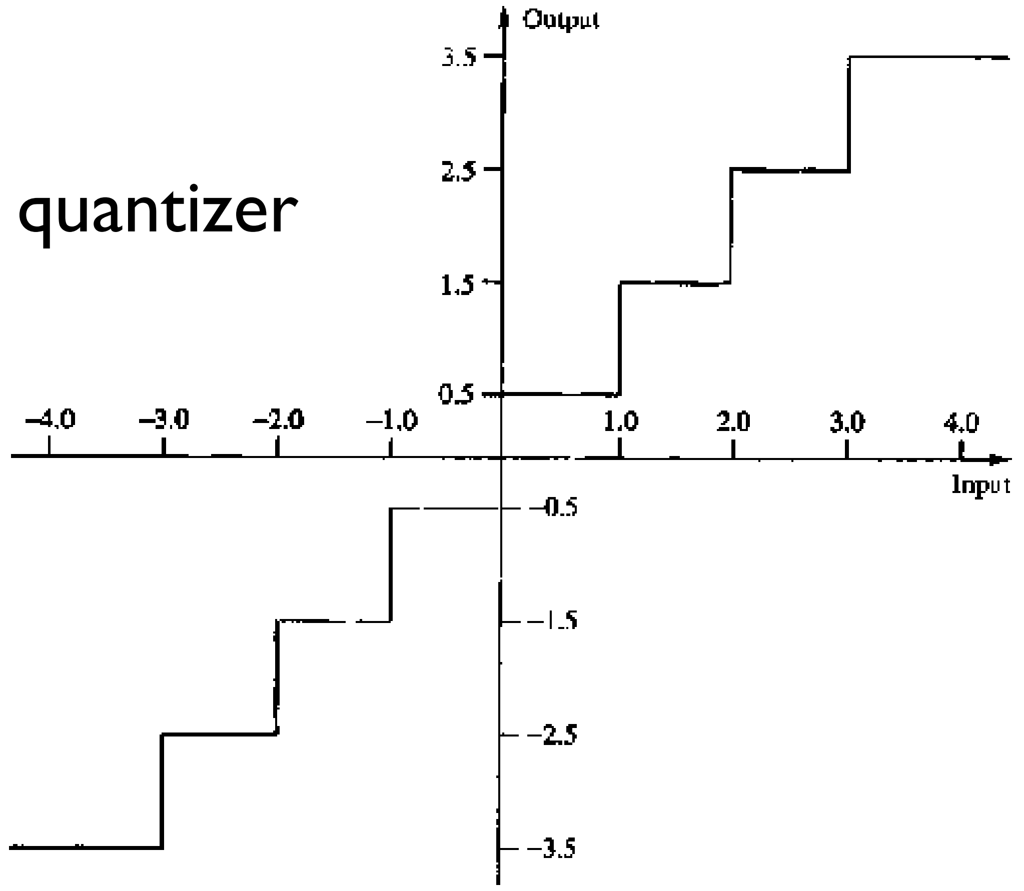


FIGURE 8.3 Quantizer Input-output map.

量化

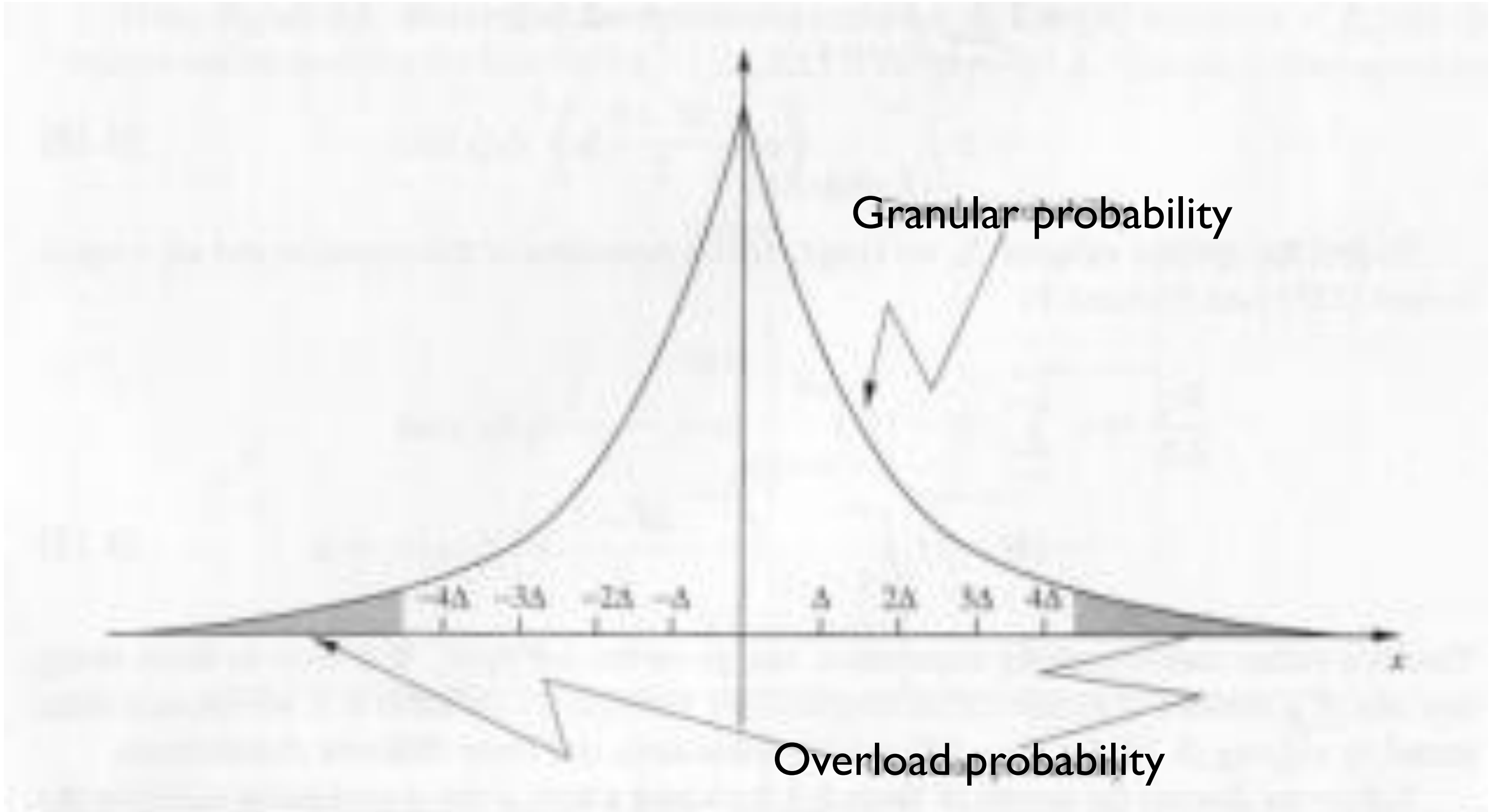
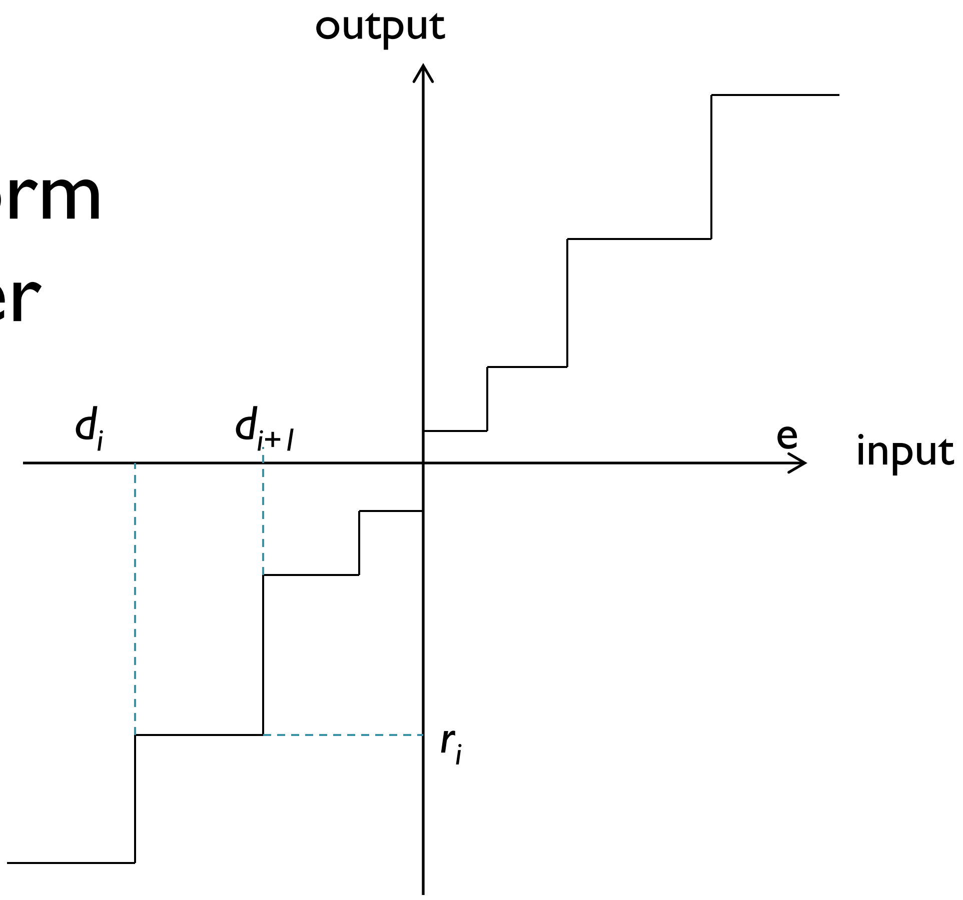


FIGURE 8.10 Overload and granular regions for a 3-bit uniform quantizer.

量化

- Nonuiform quantizer



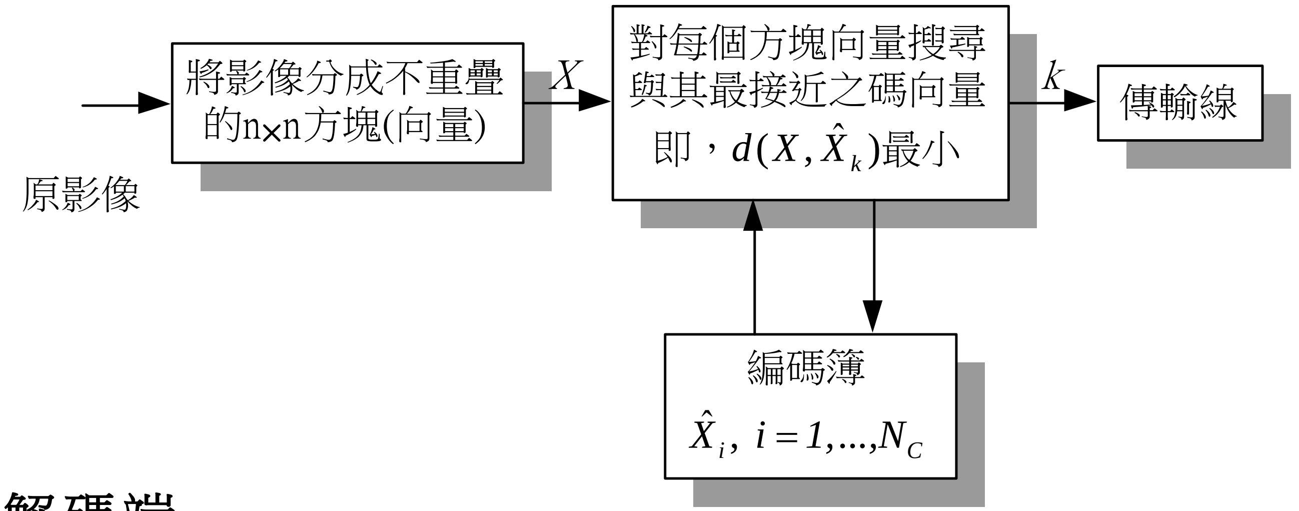
量化

i	$(d_i,d_{i+1}]$-> r_i	Probability	Huffman code
0	(-255,-16]-> -20	0.025	111111
1	(-16,-8]-> -11	0.047	11110
2	(-8,-4]-> -6	0.145	110
3	(-4,0]-> -2	0.278	00
4	(0,4]-> 2	0.283	10
5	(4,8]-> 6	0.151	01
6	(8,16]->11	0.049	1110
7	(16,255]->20	0.022	111110

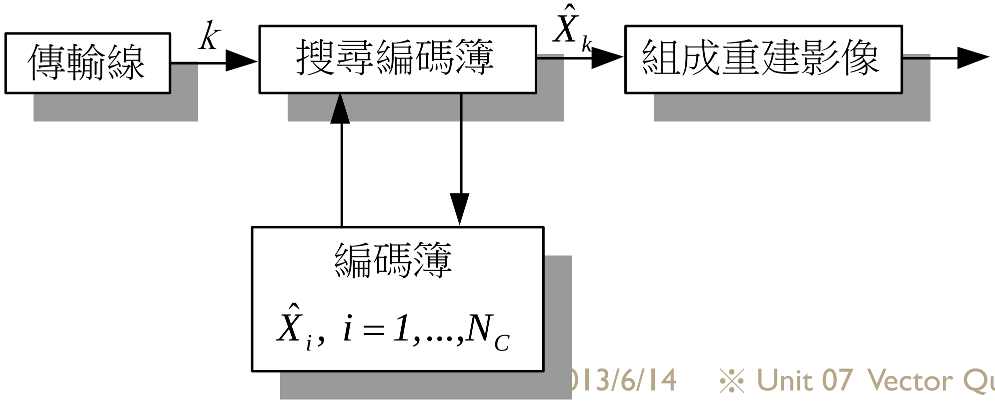
8-level Lloyd-max quantizer for Lena

向量量化 (Vector Quantization)

編碼端

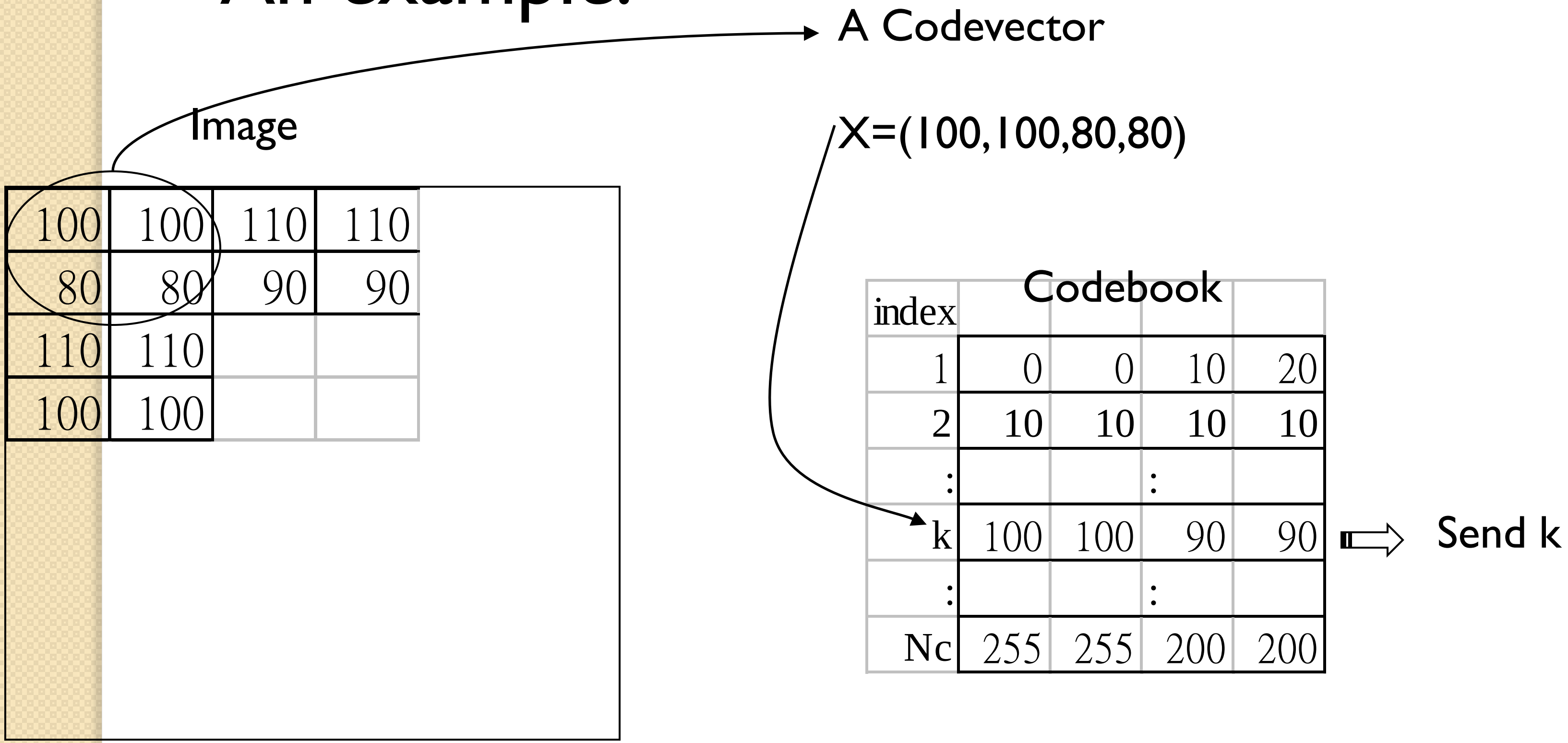

將影像分成不重疊的n×n方塊(向量)
原影像
X
對每個方塊向量搜尋與其最接近之碼向量即，$d(X,\hat{X}_k)$最小
k
傳輸線
編碼簿
$\hat{X}_i,\ i=1,\ldots,N_C$

解碼端



向量量化

- An example:



100	100	110	110	
80	80	90	90	
110	110			
100	100			

index				
1	0	0	10	20
2	10	10	10	10
:		:		
k	100	100	90	90
:		:		
Nc	255	255	200	200

向量量化

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n),\ \hat{\boldsymbol{X}} = (\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_n)$$

Let $\hat{X}_k$ be the codevector that satisfies

$d(X, \hat{X}_k) \le d(X, \hat{X}_j)$ for $j = 1, 2, \ldots, N_C$.

Then $\hat{X}_k$ can be expressed by using $\log_2 N_C$ bits

for the index of $\hat{X}_k$ in the codebook.

向量量化

where $d(X,\hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$ (MSE)

or $d_w(X,\hat{X}) = \frac{1}{n}\sum_{i=1}^{n} w_i (x_i - \hat{x}_i)^2$

(weighted MSE) w_i : *weighting* factor

向量量化

- Why VQ compress data ?
  - Number of codevectors : N_C
  - Input vector dimension: n
  - $(\log_2 N_C)/n$ bits/pixel
- **Example** : for a 4×4 blocks, $N_C = 128$, $\log_2 N_C = 7$
  bit rate=$7/(4\times4)$
- Two works in VQ :
  - Codebook generation
  - Speed up the search

Codebook Generation

- **Linde-Buzo-Gray(LBG) algorithm**

  Let the codebook size be N_C and the training vectors be $\{ x(n) | n=1,\ldots,M\}$

Step 1 Let the initial codebook C=$\{ y(i) \mid i = 1,\ldots,N_C\}$ be randomly selected from $\{ x(n) | n = 1,\ldots,M\}$.

Step 2 Cluster the training vectors into N_C groups $G(i), i = 1,\ldots,N_C$, where $G(i)=\{x(k) \mid d(x(k), y(i)) < d(x(k), y(j)),\ j \neq i$ and $d(p, q)$ denotes the distance between p and $q\}$

Step 3 Compute the distortion $$D = \sum_{i=1}^{N_c} \sum_{x(k)\in G(i)} d(x(k), y(i))$$

Step 4 If the distortion decreases, then go to Step 5 ; otherwise stop

Step 5 New $y(i) = \dfrac{\mathbf{1}}{|\boldsymbol{G(i)}|} \sum_{\boldsymbol{x(k)\in G(i)}} \boldsymbol{x(k)}$, where $|G(\mathrm{i})|$ = the number of vector in $G(i)$; go to Step 2

Codebook Generation

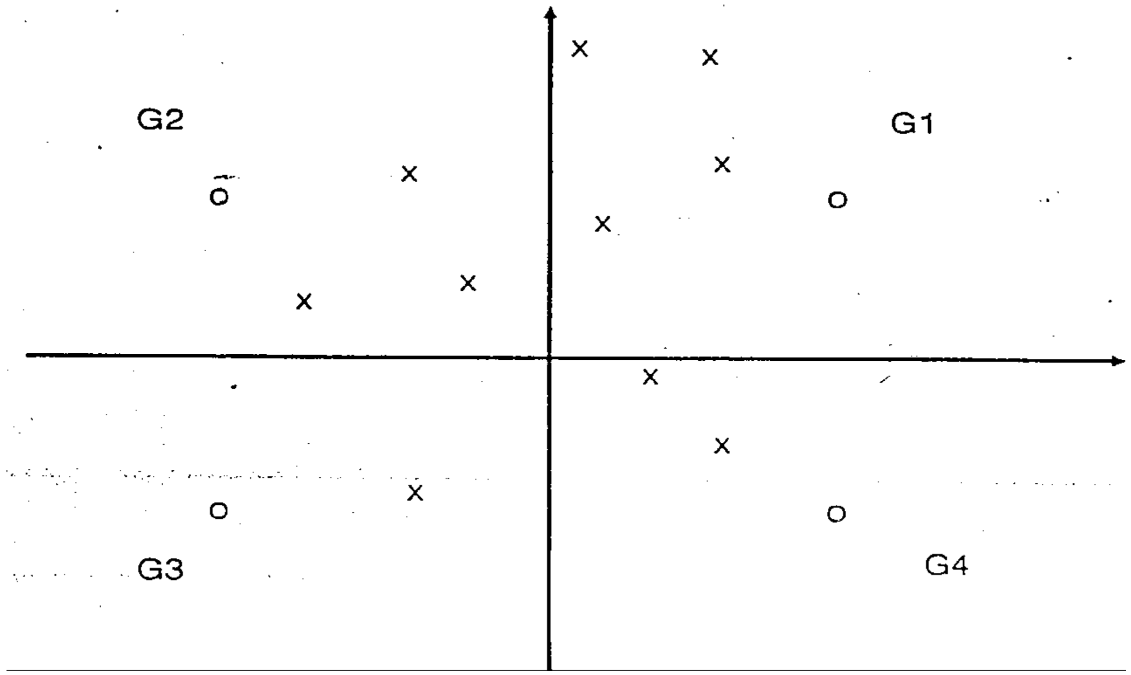
- **Linde-Buzo-Gray(LBG) algorithm**

  For each iteration, the codebook is modified, and the distortion should be decreased.

  A threshold ε is selected, and the iteration continue until

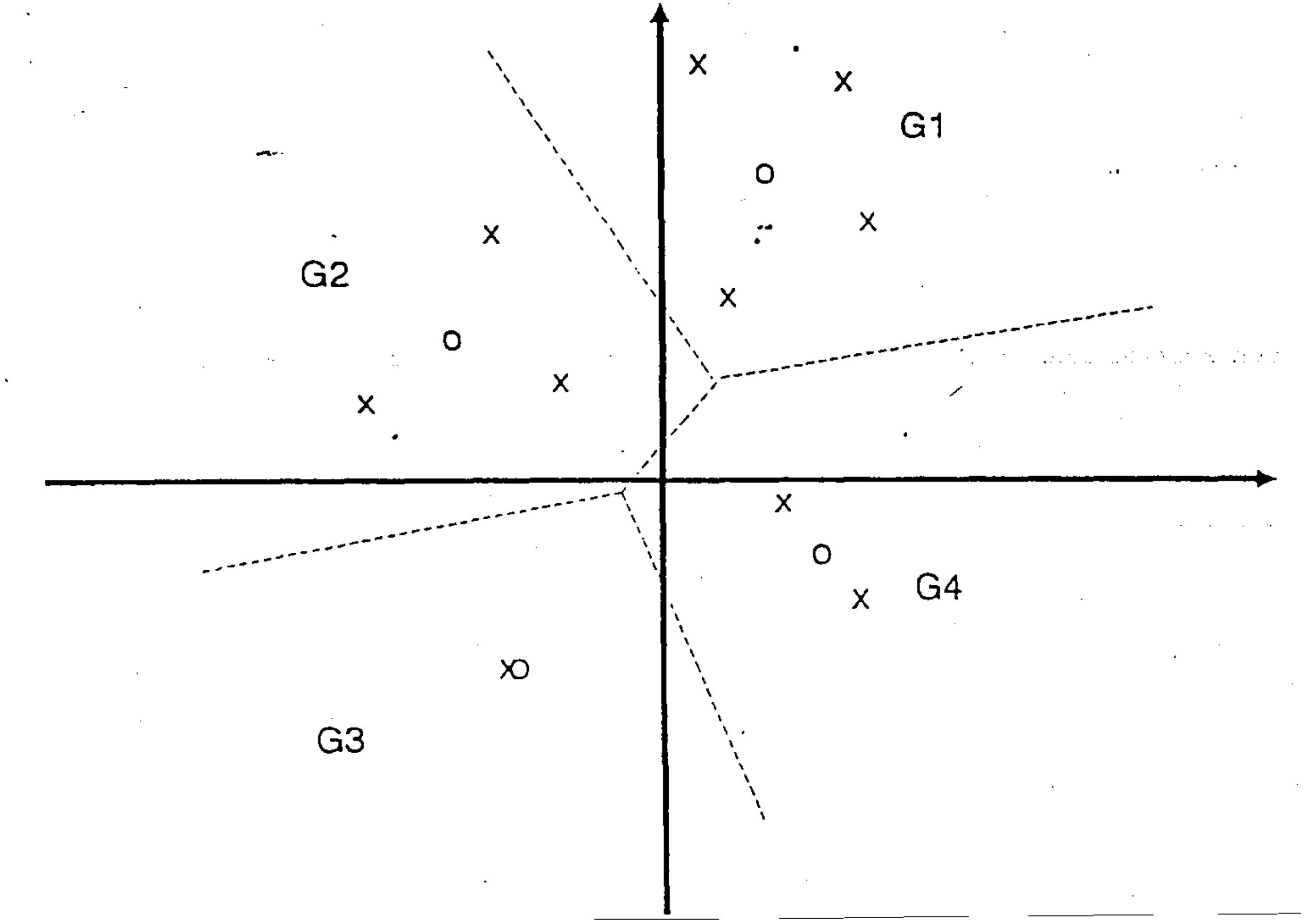
$$\frac{D^{(l-1)} - D^{l}}{D^{(l-1)}} \leq \varepsilon$$

Codebook Generation



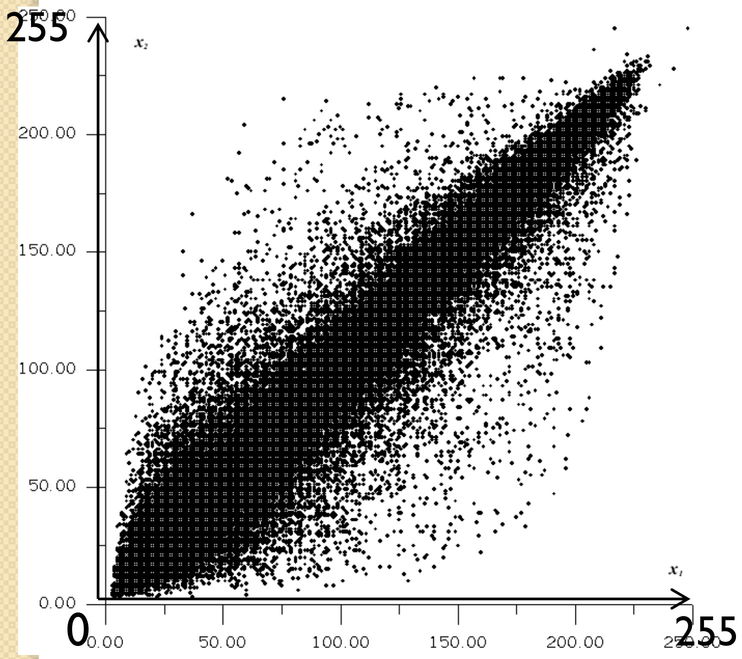
× = training codevectors
○ = codewords in the codebook
G*i* = region encoded into codeword *i*

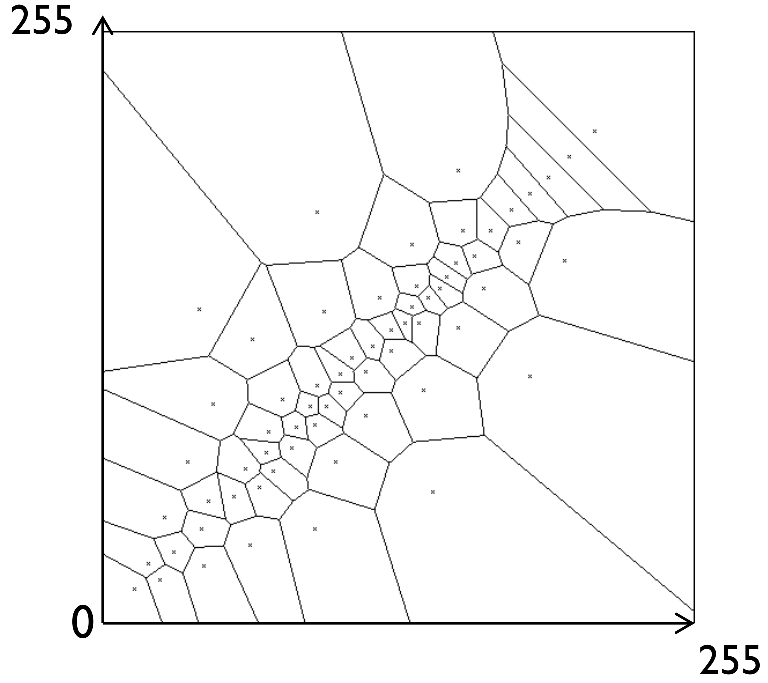
Codebook Generation



Codebook Generation

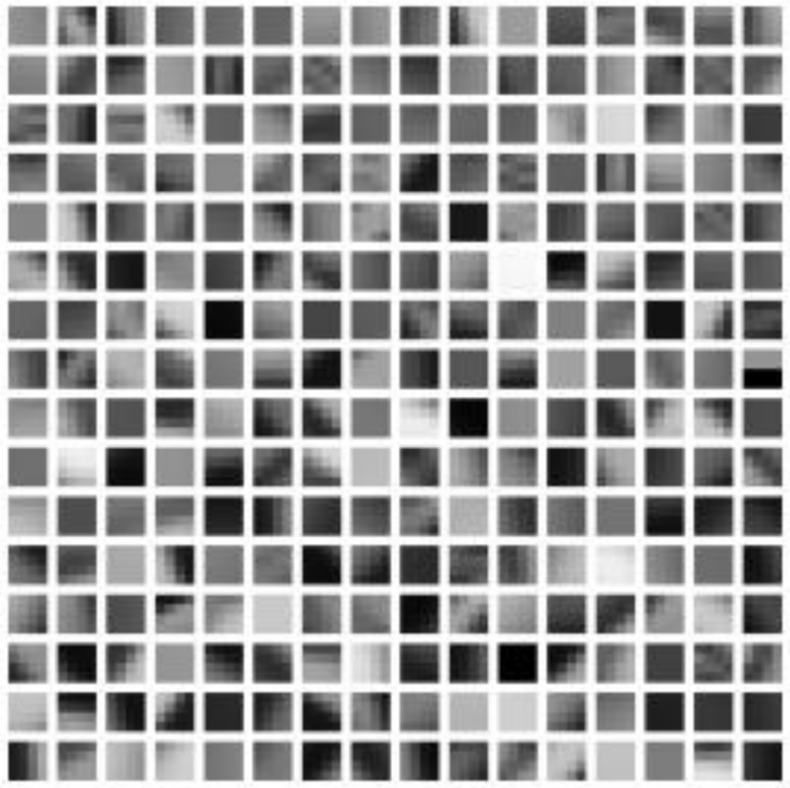
- An example of codebook using Lena
  - Block size is 1x2
  - N_c = 64





Codebook Generation

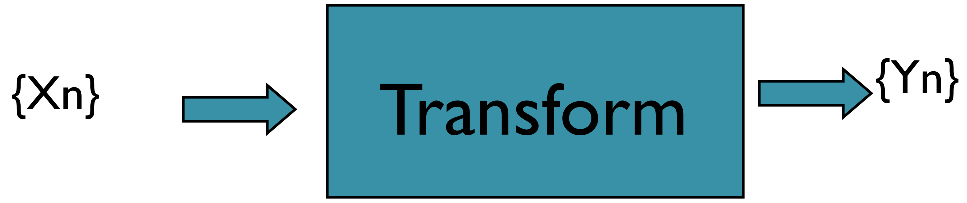
- A codebook example
  - Training set :
    - Lenna
    - Boots
  - Codebook size 256
  - Block size 4x4



Codebook Generation

- How to measure the quality of codebook
  - the quality of compressed image
    - SNR
    - bpp (bits per pixel)
    - Human visual system
- Codebook size
  - small codebook results in blocking artifacts
  - large codebook achieves better quality
    - higher bit rate
    - longer the search time

轉換編碼 (Transformation)



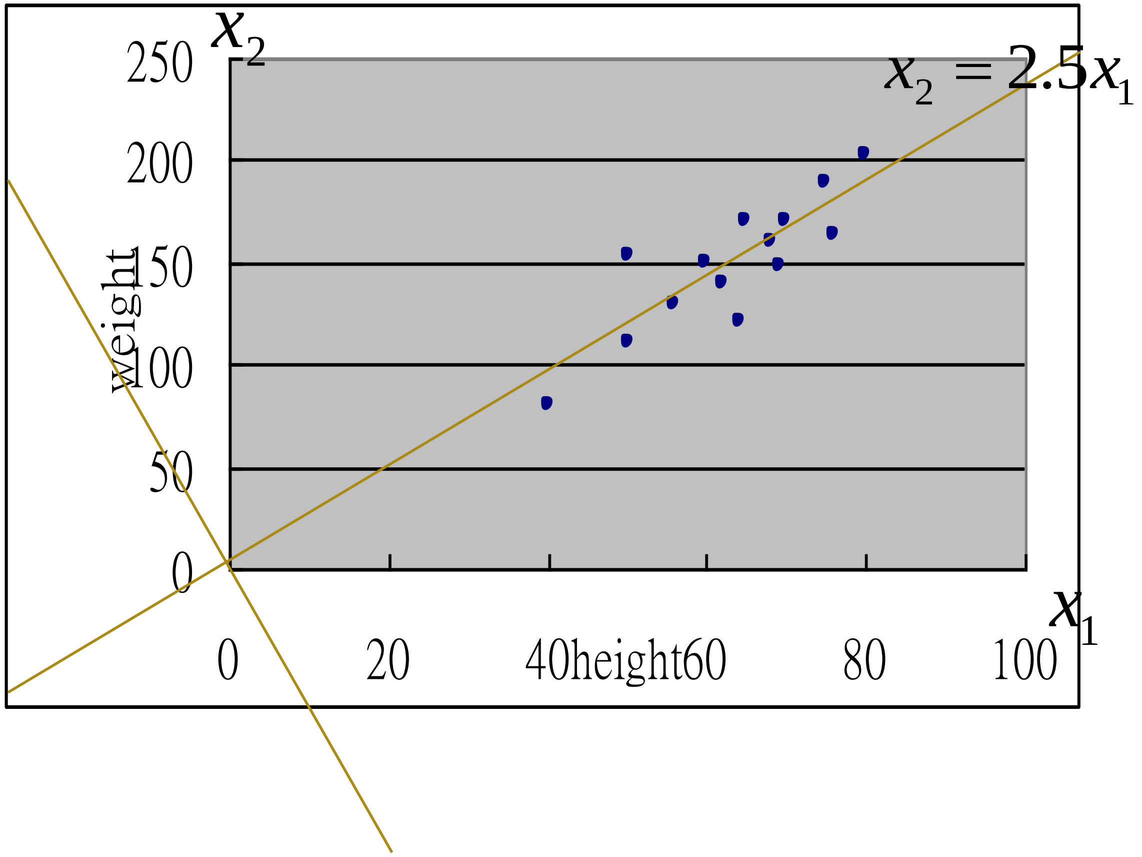
Y=AX

- Take a sequence of inputs and transform them into another sequence in which most of the information is contained in only a few elements.
- And, then discarding the elements of the sequence that do not contain much information, we can get a large amount of compression.

variance

轉換編碼

Height	Weight
65	170
75	188
60	150
70	170
56	130
80	203
68	160
50	110
40	80
50	153
69	148
62	140
76	164
64	120



轉換編碼

$$x_2 = 2.5x_1$$

$$\phi = \tan^{-1} 2.5 = 68.2$$

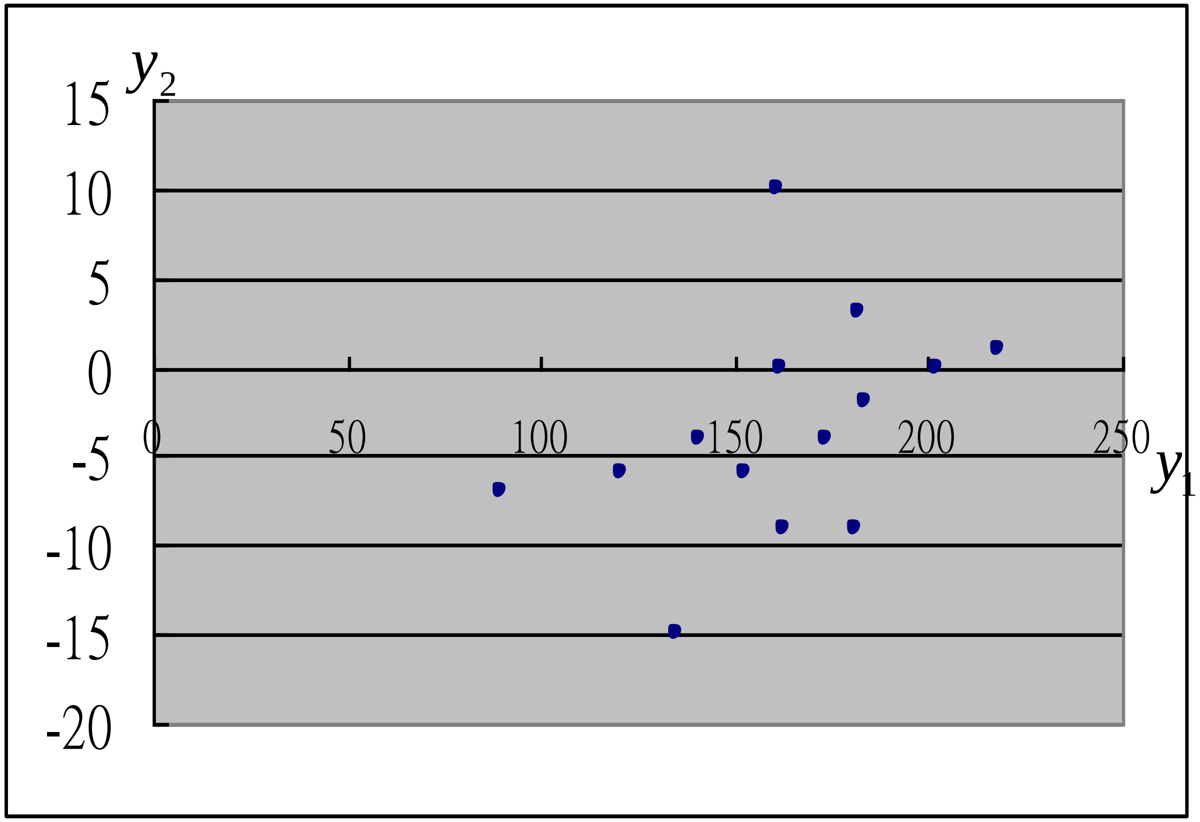
$$\mathbf{A} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} 0.37139068 & 0.92847699 \\ -0.92847699 & 0.37139068 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \qquad \mathbf{y} = \mathbf{Ax}$$

$$\mathbf{A}^{-1} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \qquad \mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$$

轉換編碼

Height	weight
182	3
202	0
162	0
184	-2
141	-4
218	1
174	-4
121	-6
90	-7
161	10
163	-9
153	-6
181	-9
135	-15



轉換編碼

- The transmitted sequence:

Height	weight
182	0
202	0
162	0
184	0
141	0
218	0
174	0
121	0
90	0
161	0
163	0
153	0
181	0
135	0

$\Rightarrow$

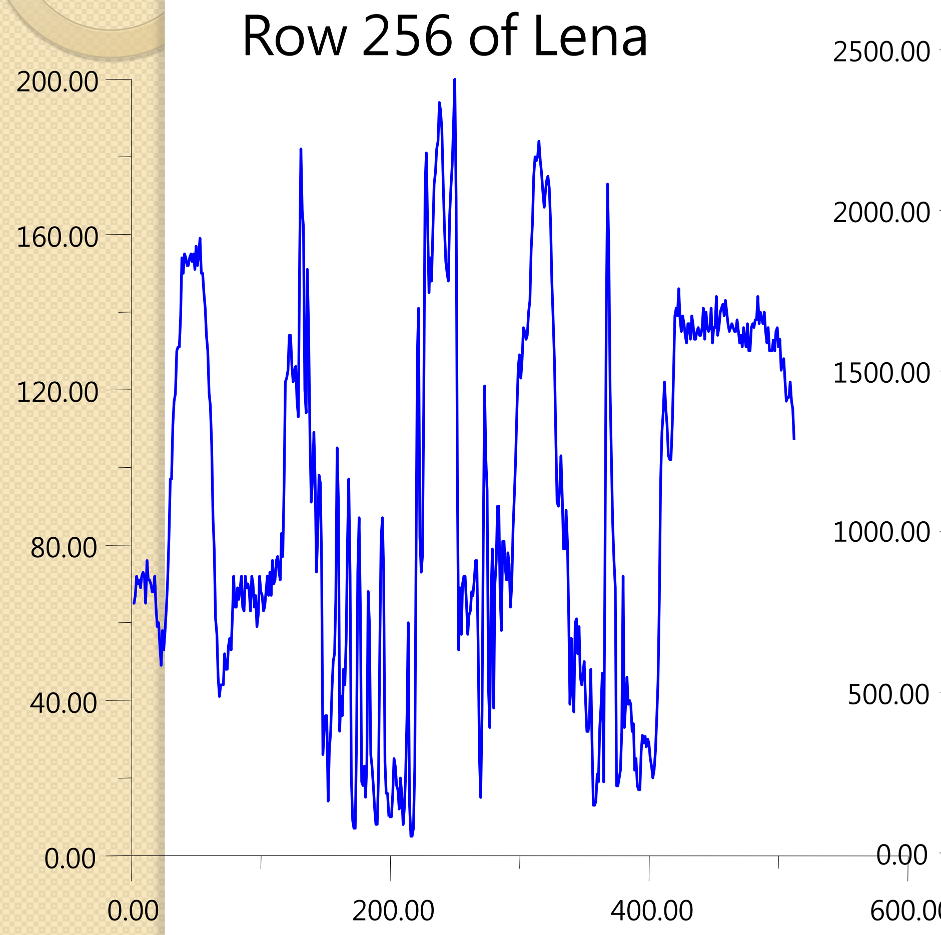
Height	weight
68	169
75	188
60	150
68	171
53	131
81	203
65	162
45	112
34	84
60	150
61	151
57	142
67	168
50	125

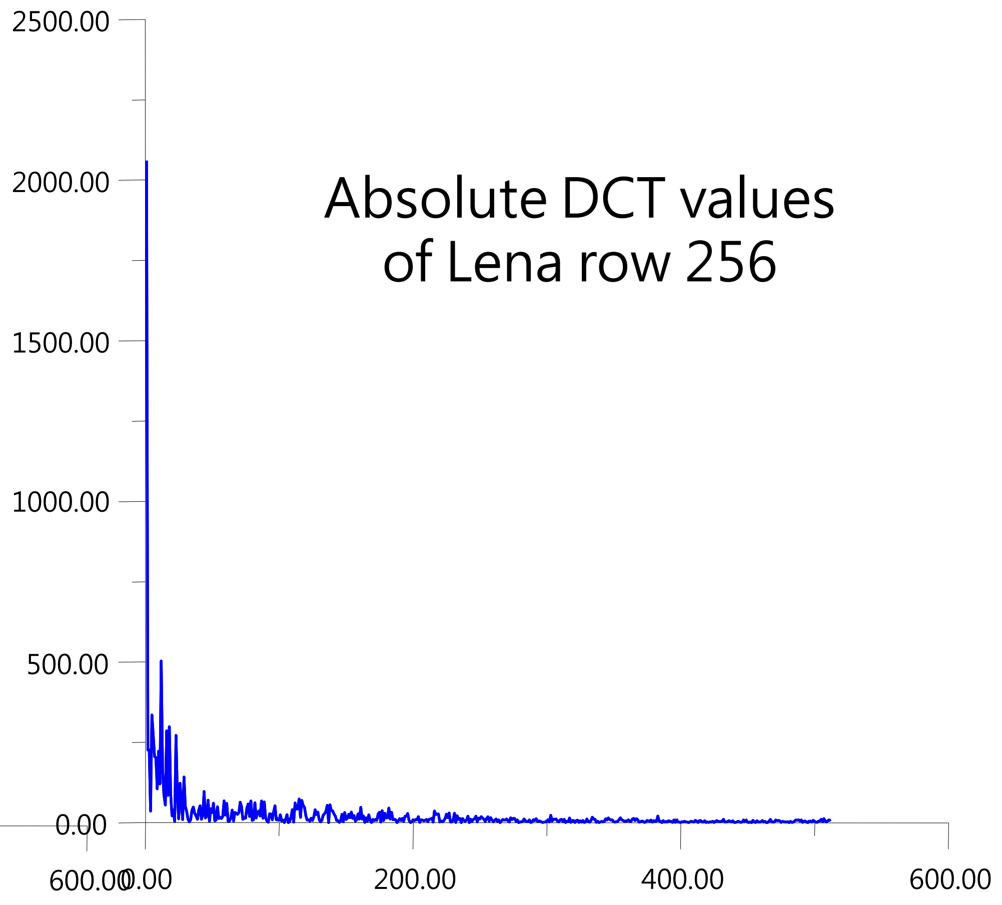
Original

Height	Weight
65	170
75	188
60	150
70	170
56	130
80	203
68	160
50	110
40	80
50	153
69	148
62	140
76	164
64	120

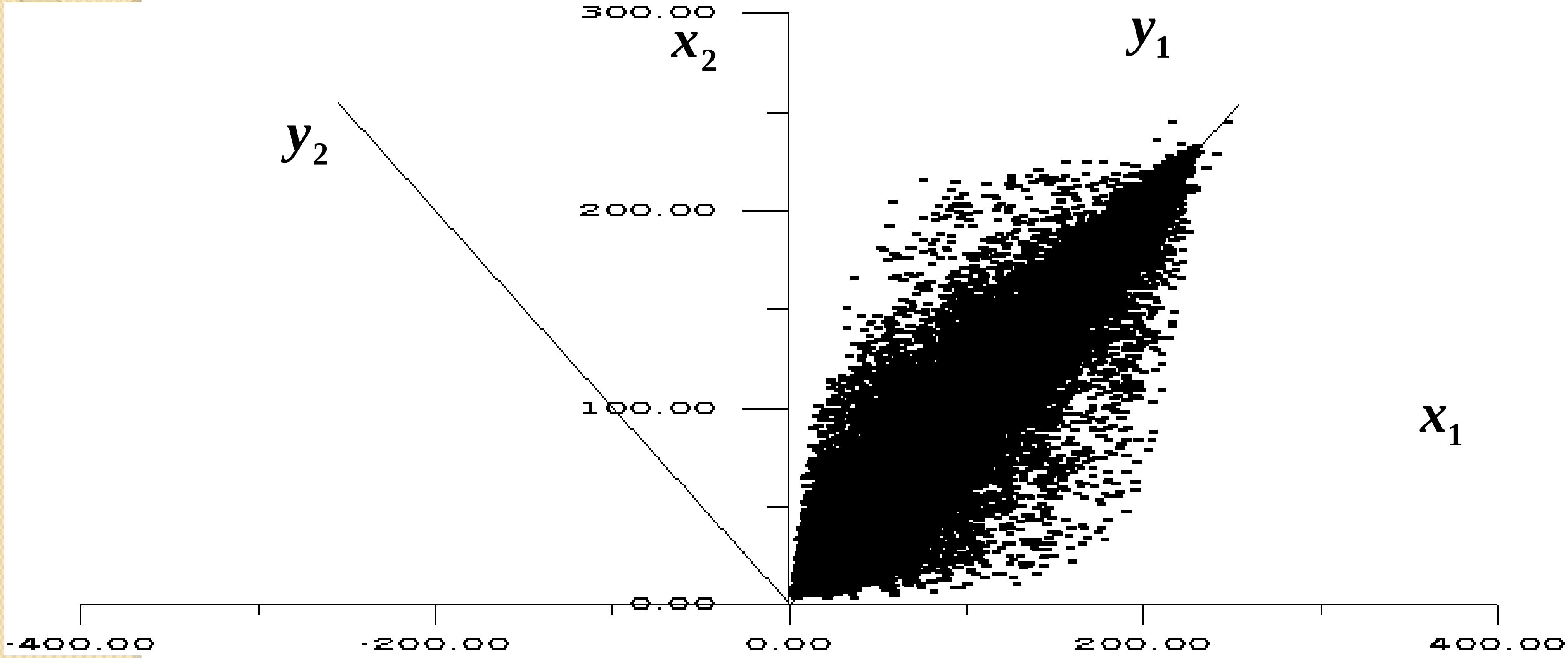
Introductions

Another example





Transforms as Coordinate Axes Rotations (座標軸旋轉)



Coordinate Axes Rotated by 45°

Transforms as Coordinate Axes Rotations (座標軸旋轉)

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$Y = AX$$

$\because$ A is unitary and orthogonal

$\therefore$ *Euclidean* distance is preserved

That is, $\sigma_{x_1}^2 + \sigma_{x_2}^2 = \sigma_{y_1}^2 + \sigma_{y_2}^2$

But, $\sigma_{y_1}^2 >> \sigma_{y_2}^2$

MSE is minimized $\left(= \sigma_{y_2}^2\right)$ if all y_{2i}'s are replaced by $\bar{y}_2$.

Note that $\sigma_{y_2}^2 << \sigma_{x_1}^2 \left(\sigma_{x_2}^2\right)$.

Transforms as Basis Function Decompositions (基底函數分解)

$$\underset{0\le j,k\le 7}{f(j,k)} = \mathrm{F}(0,0)\mathrm{B}(0,0) + \mathrm{F}(1,0)\mathrm{B}(1,0) + \ldots + \mathrm{F}(7,0)\mathrm{B}(7,0)$$

$$+ \mathrm{F}(0,1)\mathrm{B}(0,1) + \mathrm{F}(1,1)\mathrm{B}(1,1) + \ldots + \mathrm{F}(7,1)\mathrm{B}(7,1)$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$+ \mathrm{F}(0,7)\mathrm{B}(0,7) + \mathrm{F}(1,7)\mathrm{B}(1,7) + \ldots + \mathrm{F}(7,7)\mathrm{B}(7,7)$$

where

$\mathrm{B}(i, j)$: Basis function

$\mathrm{F}(i, j)$: Transform Coefficients.

DCT 轉換編碼

- Discrete Cosine Transform (DCT)

$$F(u,v)=\frac{4C(u)C(v)}{n^2}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1}f(j,k)\cos[\frac{(2j+1)u\pi}{2n}]\cos[\frac{(2k+1)u\pi}{2n}]$$

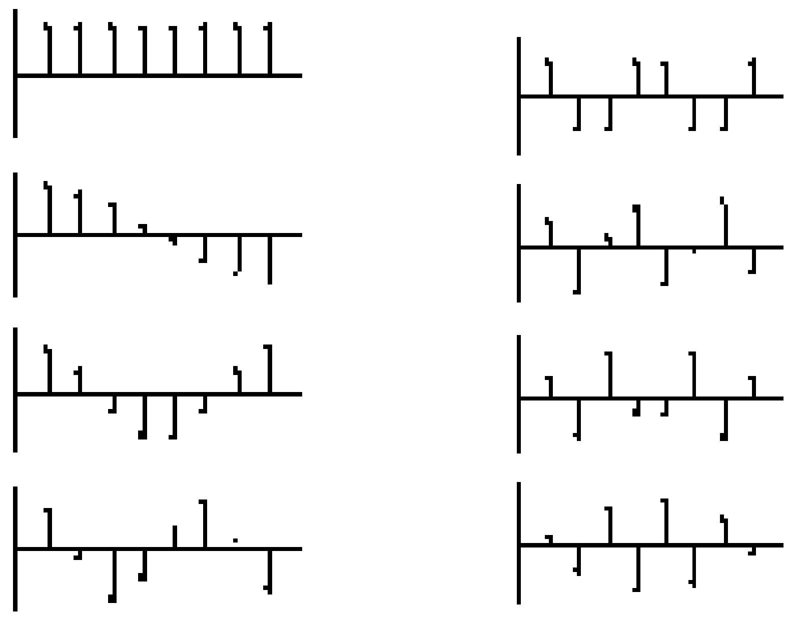
Inverse DCT

$$f(j,k)=\sum_{v=0}^{n-1}\sum_{u=0}^{n-1}C(u)C(v)F(u,v)\cos[\frac{(2j+1)u\pi}{2n}]\cos[\frac{(2k+1)u\pi}{2n}]$$

$$c(w)=\begin{cases}1/\sqrt{2} & \text{if } w=0\\ 1 \text{ if } w=1,2,\ldots,n\text{-}1\end{cases}$$

DCT 轉換編碼

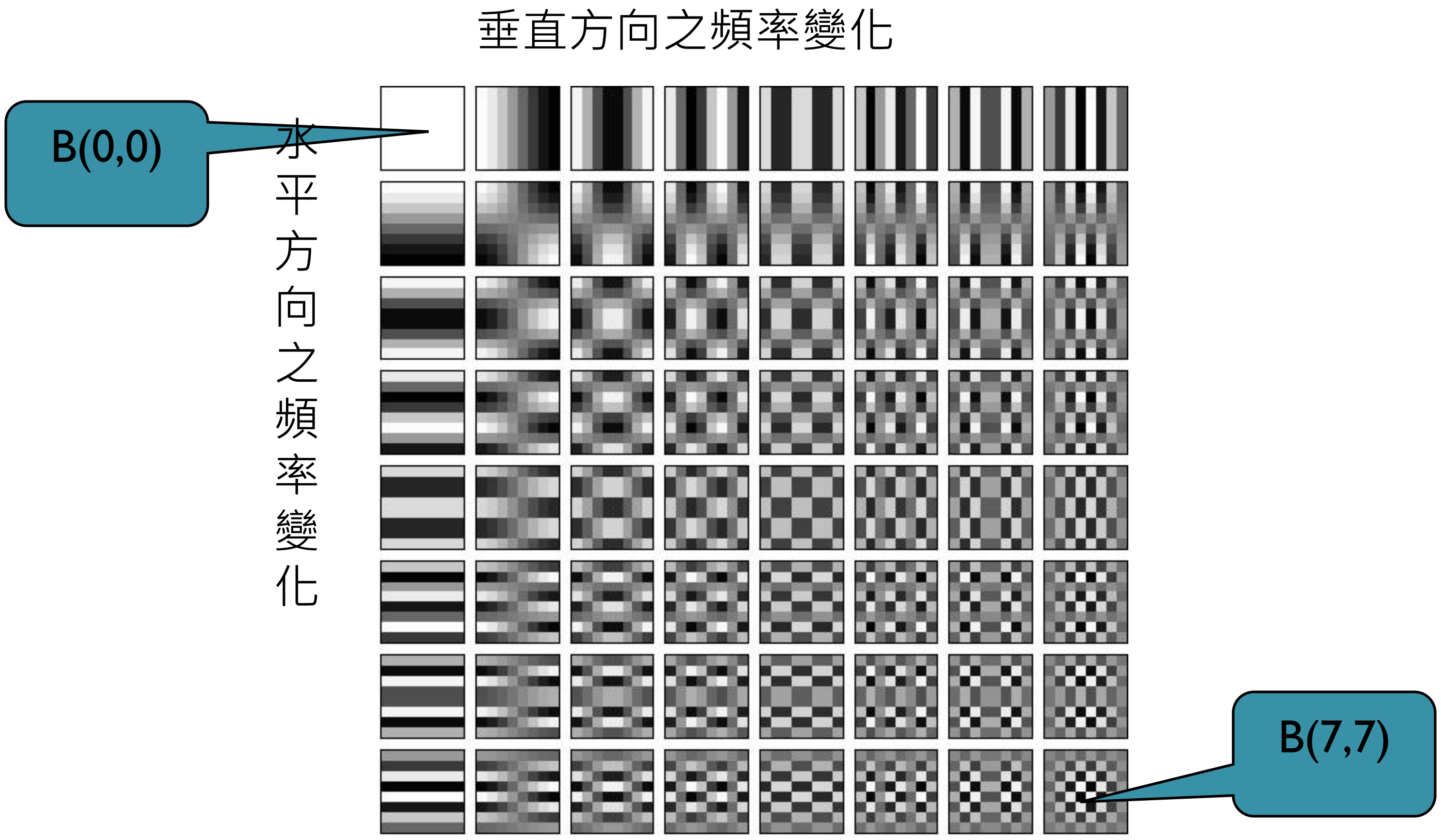
- One dimension DCT basis functions.



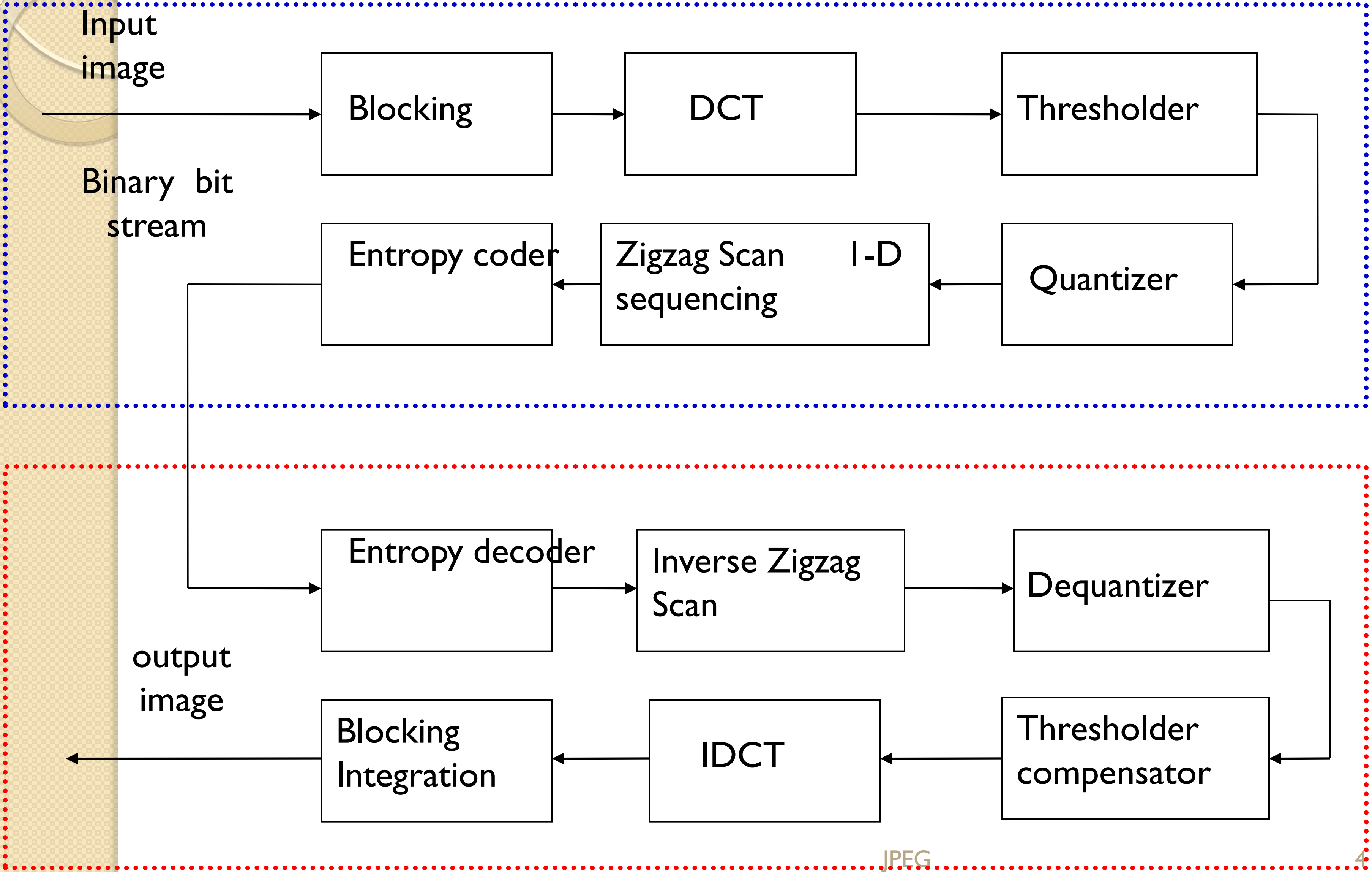
n=8

DCT 轉換編碼

- Two dimension DCT basis functions (8 x 8)



JPEG


Input image
Blocking
DCT
Thresholder
Binary bit stream
Entropy coder
Zigzag Scan 1-D sequencing
Quantizer
Entropy decoder
Inverse Zigzag Scan
Dequantizer
output image
Blocking Integration
IDCT
Thresholder compensator