第十章
空間資料結構設計與應用

內容

- 10.1 前言
- 10.2 黑白影像的空間資料結構表示法
- 10.3 高灰階影像的空間資料結構表示法
- 10.4 基本影像運算之應用
- 10.5 結論

10.1 前言

- 主要介紹黑白及灰階影像的空間資結構表示法。另外介紹一些應用。

10.2 黑白影像的空間資料結構表示法

10.2.1 四分樹表示法

- 四分樹切割

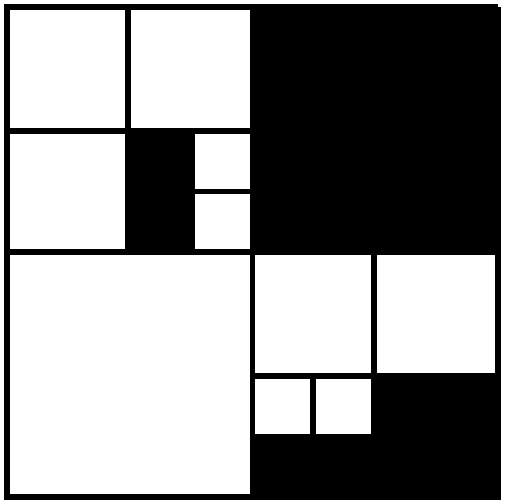


圖10.2.1.1 黑白影像

圖10.2.1.2 四分樹表示法

- 四分樹的正規化
  圖10.2.1.3需16個葉子點。往東南方向移動一格，只需七個葉子點。

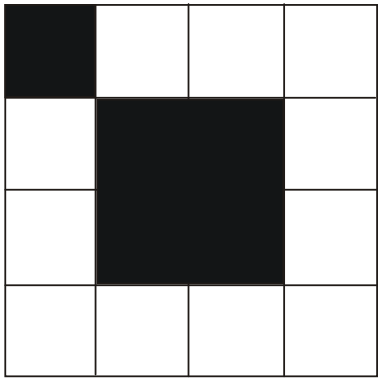
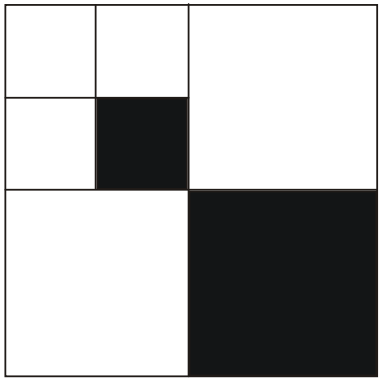
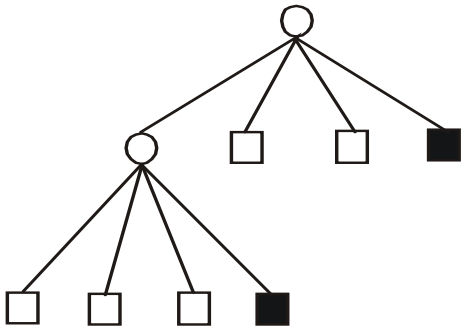


圖10.2.1.3 4×4黑白影像



(a) 移動後的結果



(b) 移動後的四分樹表示法

圖10.2.1.4移位後的效果

適當的移位可以減少葉子數量來達到節省記憶體的功效。

10.2.2 深先表示法

圖10.2.1.2的四分樹，用深先搜尋法

內部節點 ➔ 輸出G
白色外部節點 ➔ 輸出W
黑色外部節點 ➔ 輸出B

得到GGWWWGBWBWBWGWWGWWBBB。

可改成((000 (101010(00(00111。

圖10.2.1.2四分樹表示法

10.2.3 線性四分樹表示法

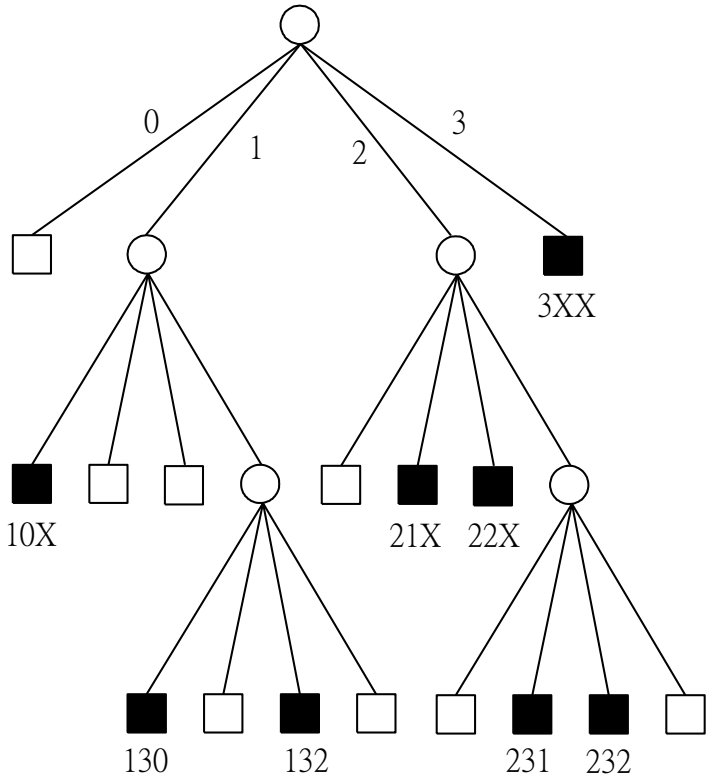
- 線性四分樹
  圖10.2.1.2的可表示為 030，032，1XX ， 322，323，33X。

圖10.2.1.2 四分樹表示法

兩組不同的線性四分樹編碼，還原出一樣的四分樹。

10X, 130, 132, 21X, 22X, 231, 232, 3XX

3XX, 10X, 21X, 22X, 130, 132, 231, 232



- 實驗

照原圖儲存共需65536個位元。實驗結果：

- 深先表示法需花19024位元
- JBIG來壓縮圖需花10976位元 (難運算)

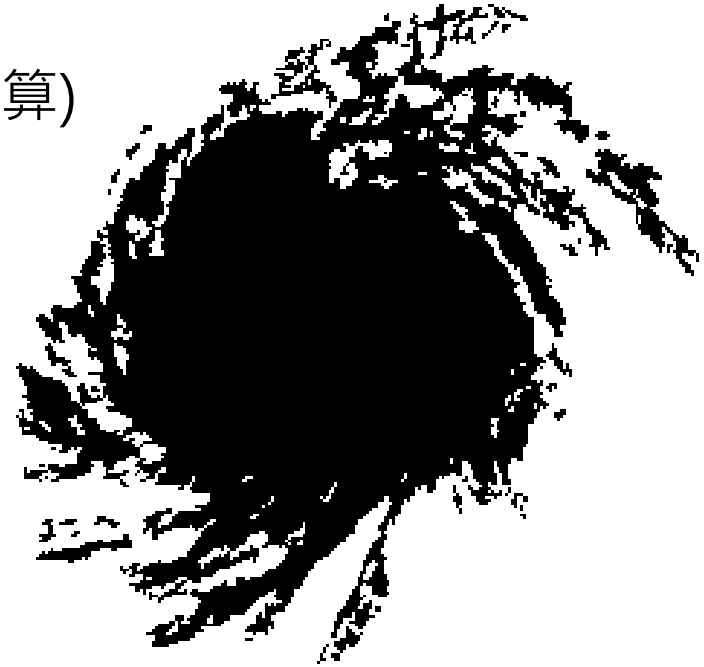


圖10.2.3.1　256 × 256颱風影像

10.2.4 S樹表示法

以深先搜尋表示

- 線性樹表
  - 內部節點 ➔ 輸出0
  - 外部節點 ➔ 輸出1

  線性樹表可表示為 00010101110100100011011011。
- 顏色表
  - 白色葉子 ➔ 輸出0
  - 黑色葉子 ➔ 輸出1

  顏色表可表示為
  0010010010101。

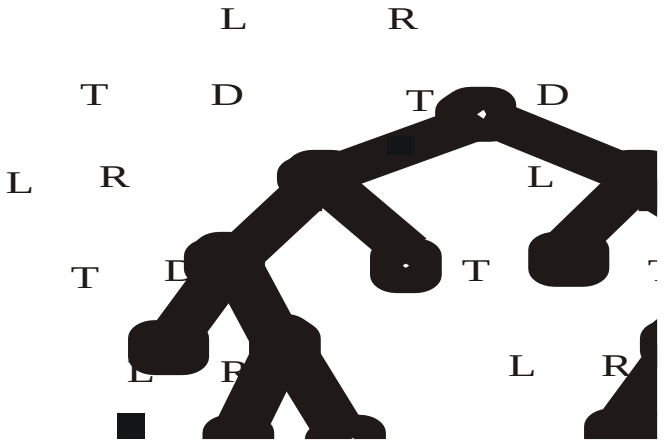


圖10.2.4.1　圖10.2.1.1的二分樹表示法

10.3 高灰階影像的空間資料結構表示法

- 一維線性內插

O點=(1,5)，此處1表示 x 軸的位置而5表示灰階值；C點=(11,13)。A點=(4,?)，則

由 $\frac{\overline{OA}}{\overline{OC}} = \frac{\overline{AB}}{\overline{CD}}$

得 $\overline{AB} = \frac{\overline{OA} \times \overline{CD}}{\overline{OC}} = \frac{3 \times 8}{10} = 2.4$

A點的灰階值約為7=(5+2)。

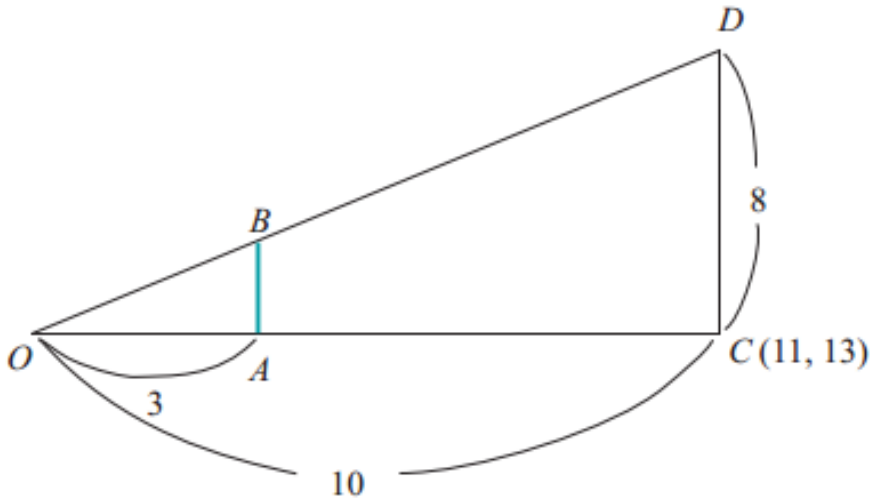


圖10.3.1 一維的線性內插

- 二分樹切割的條件

$$|g(x,y)-g_{est}(x,y)|\leq \varepsilon$$

此處
- $g(x,y)$: 原始灰階值
- ε:誤差容忍度
- $g_{est}(x,y)$: 線性內插得到的估計灰階值

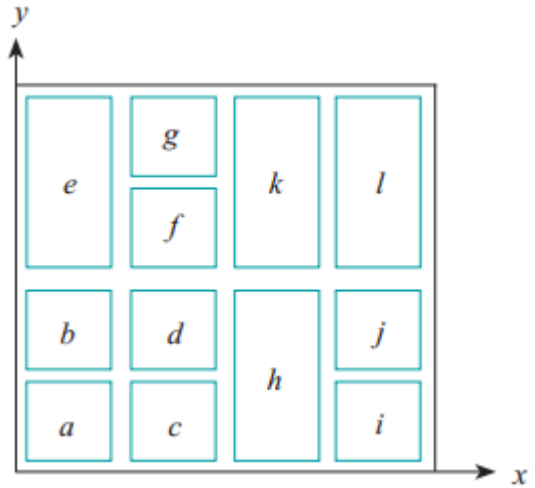


圖10.3.2 同質的區塊分割圖

圖10.3.3 二分樹表示法

- 求算 $g_{est}(x,y)$

  假設一個區塊的四個角點分別如下

	位置	灰階值
左上	(x_1, y_1)	g_1
右上	(x_2, y_1)	g_2
左下	(x_1, y_2)	g_3
右下	(x_2, y_2)	g_4

  利用一維線性內插可以得到 $g_{est}(x,y) = g_5 + \frac{g_6 - g_5}{y_2 - y_1}(y - y_1)$

  此處 $g_5 = g_1 + \frac{g_2 - g_1}{x_2 - x_1}(x - x_1)$ 和 $g_6 = g_3 + \frac{g_4 - g_3}{x_2 - x_1}(x - x_1)$ 。

- 廣先搜尋

  圖10.3.3的二分樹用 S 樹表示如下

線性樹表：00000000010101111111111

顏色表：$(e_{ul}, e_{ur}, e_{bl}, e_{br})$, $(h_{ul}, h_{ur}, h_{bl}, h_{br})$, … , $(j_{ul}, j_{ur}, j_{bl}, j_{br})$

- **重疊策略**(Overlapping Strategy)

  由於區塊與區塊之間是分開的，會造成**區塊效應**(Blocking Effect)，採用重疊策略來降低區塊效應的影響。

  原先$2^n \times 2^n$大小的影像放大成$(2^n + 1) \times (2^n + 1)$的大小。

  像素分享的特色配合線性內插的平滑性，解壓出來後可降低區塊效應。

- 實驗

在$\varepsilon = 21$時，圖10.3.4的 S 樹所需的bpp(Bit Per Pixel)約為1.35位元，這與原始影像一個像素需8個位元相比，壓縮改良率為83%。

圖10.3.4 ε=21得到的還原影像圖

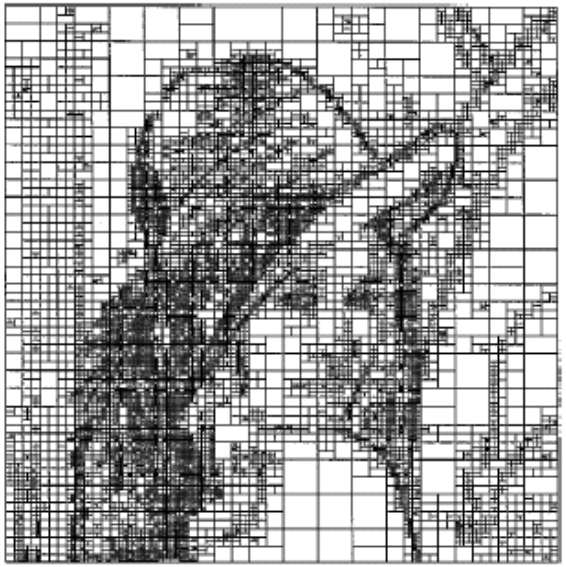


圖10.3.5 二元分割後的區塊示意圖

在壓縮比上不如JPEG (Joint Photographic Experts Group)來的好，但在解碼的時間(Decoding Time)上快3~4倍。

10.4 基本影像運算之應用
10.4.1 影像加密

網路傳輸前，將資料加密(Encrypt)，使攔截者無法有效的解密(Decrypt)。

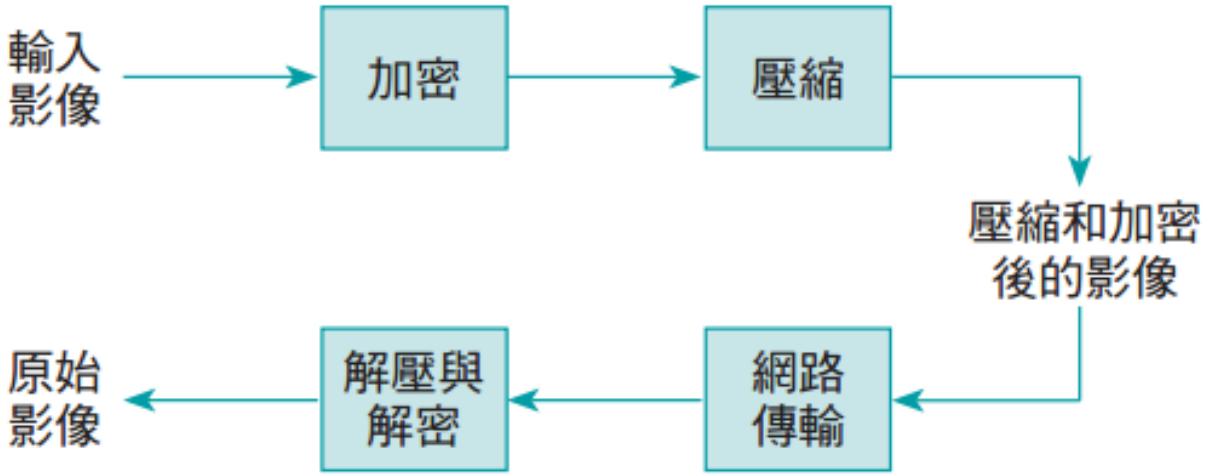
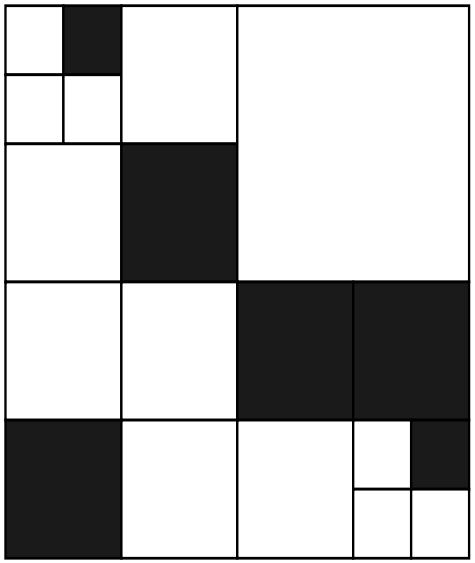
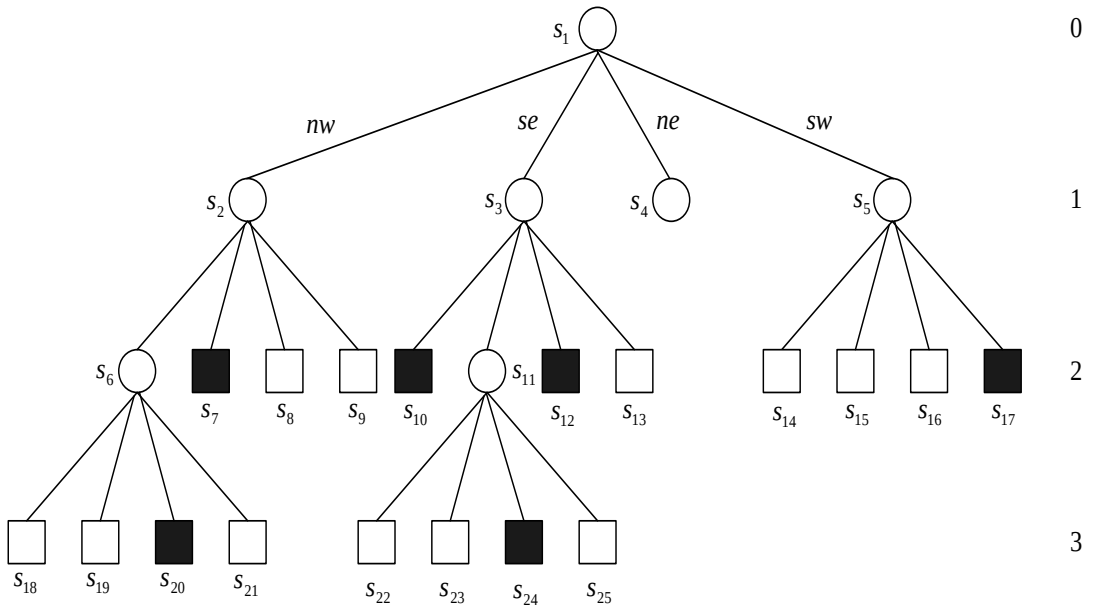


圖10.4.1.1 影像加密系統



(a) 8×8黑白影像



(b) 四分樹結構

圖10.4.1.2 影像加密的例子

- 定義掃瞄語言

假設影像的大小為 $2^n \times 2^n$，掃瞄語言可被定義為文法 $G = \langle V_N, V_T, P, S \rangle$

$V_N = \left\{ S, \bigcup_{i=1}^{n} L_i \right\}$ 代表非終結符號集

L_i 代表四分樹中第 i 層的掃瞄圖案

$V_T = \left\{ \bigcup_{i=1}^{n} \Omega_i^{4^{i-1}} \middle| \Omega_i = \left\{ R_j^i \middle| 1 \le j \le 4^{i-1} \right\} \right\}$ 終結符號集

$\Omega_i^{4^{i-1}} = \left\{ \Omega_i \Omega_i \cdots \Omega_i \ (\Omega_i \text{連乘} 4^{i-1} \text{次}) \right\}$

R_j^i 圖9.4.1.3中定義的24個掃瞄圖案中的一個

S 代表起始符號

P 代表文法 G 中的產生規則

SP_i 為24個掃瞄圖案中的第 i 個掃瞄圖案

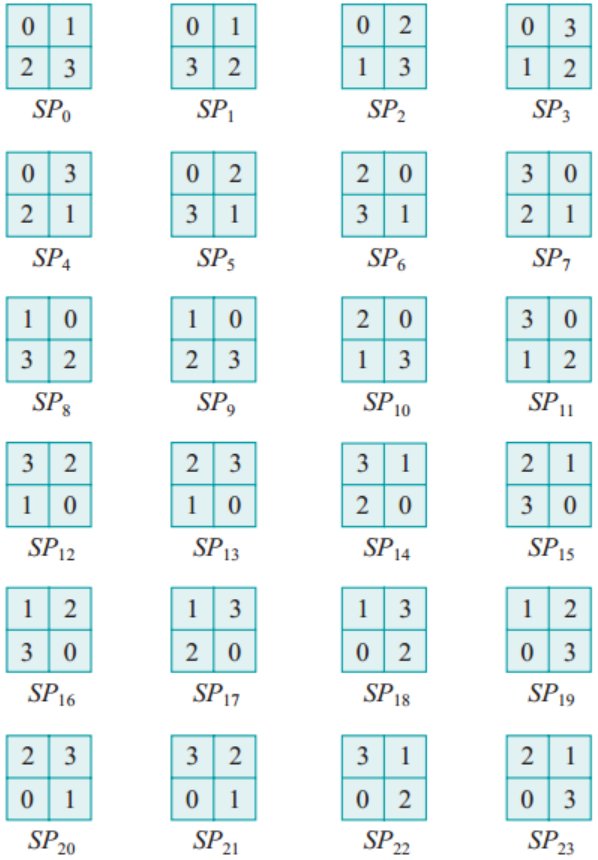


圖10.4.1.3 24個掃瞄圖案

- 例子

給定一組產生規則如下

$$S \rightarrow L_1 L_2 L_3$$

$$L_1 \rightarrow R_1^1$$

$$L_2 \rightarrow R_1^2 R_2^2 R_3^2 R_4^2$$

$$L_3 \rightarrow R_1^3 R_2^3 R_3^3 R_4^3 R_5^3 R_6^3 R_7^3 R_8^3 R_9^3 R_{10}^3 R_{11}^3 R_{12}^3 R_{13}^3 R_{14}^3 R_{15}^3 R_{16}^3$$

$$R_1^1 = SP_1$$

$$R_1^2 = SP_{23},\quad R_2^2 = SP_2,\quad R_3^2 = SP_4,\quad R_4^2 = SP_7$$

$$R_1^3 = SP_1,\quad R_2^3 = SP_{11},\quad R_3^3 = SP_{13},\quad R_4^3 = SP_1$$

$$R_5^3 = SP_4,\quad R_6^3 = SP_1,\quad R_7^3 = SP_0,\quad R_8^3 = SP_7$$

$$R_9^3 = SP_1,\quad R_{10}^3 = SP_{10},\quad R_{11}^3 = SP_1,\quad R_{12}^3 = SP_{21}$$

$$R_{13}^3 = SP_{11},\quad R_{14}^3 = SP_1,\quad R_{15}^3 = SP_{13},\quad R_{16}^3 = SP_{15}$$

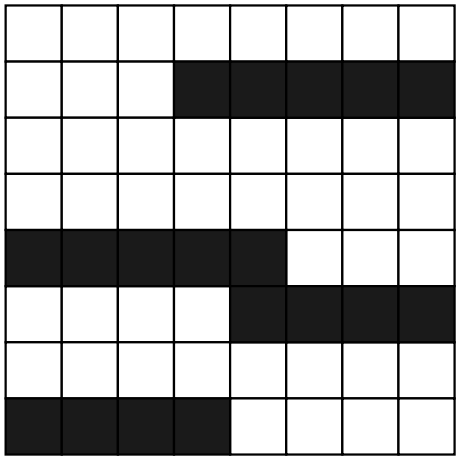


圖10.4.1.4 加密後的結果

則圖10.4.1.2(a)的黑白影像被加密成圖10.4.1.4。

利用列掃瞄的方式，

圖10.4.1.4可表示成0000000000011111000000000000000111110000000111100000000011110000，進而用011516574844來表示。