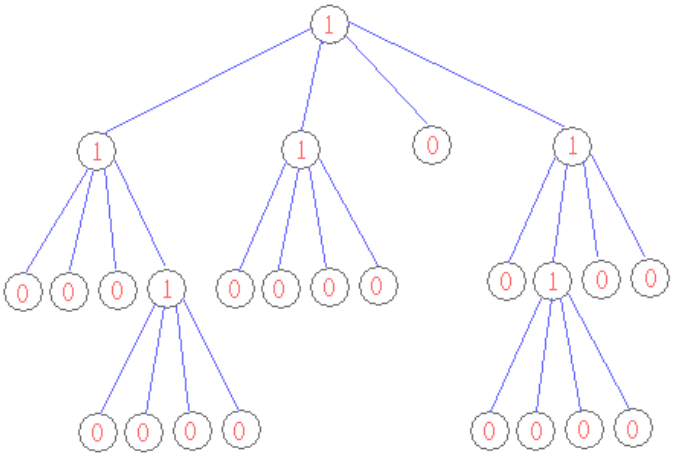
PDA 上四元樹影像解壓縮暨漸進式影像傳輸系統



由左邊影像切割出來的若干影像區塊所形成的一棵四元樹

組別:J 組

指導老師:曾修宜

組員: 資四 B 93156233 柯國偉

資四 A 91156160 李楊國

資四 A 93156183 謝翰澤

【目錄】

一、系統摘要 2-4
1.1 系統名稱 2
1.2 系統目標 2
1.3 系統使用者 2
1.4 系統功能 2
1.5 系統限制 2
1.6 系統開發環境 3
1.7 系統藍圖 3
1.8 系統藍圖概說 4
二、系統功能 5-6
2.1 系統功能結構圖 5
2.2 系統功能 6
2.2.1 Server端 6
2.2.2 Client端 6
三、系統設計 7-11
3.1 系統類別與模組 7-8
3.1.1 Server端 7
3.1.2 Client端 8
3.2 類別模組關聯圖 9
3.2.1 Server端 9
3.2.2 Client端 9
3.3 類別與模組說明 10-11
3.3.1 Client端 10
3.3.2 Server端 11
四、四元樹影像解壓縮暨漸近式傳輸 12-24
4.1 四元樹影像壓縮技術淺談 12-14
4.2 影像解壓縮 15
4.3 影像精鍊 16-20
4.4 漸近式影像傳輸 20-21
4.5 影像區域精鍊 21-24
五、系統操作介面與使用 25-29
5.1 連至Server端功能執行過程 25-29
5.2 本機端功能執行過程 29
六、四元樹影像壓縮率暨系統執行效能評估 30-31
七、結論 31

【一、系統摘要】

【1.1 系統名稱】:

PDA 上四元樹影像解壓縮暨漸進式影像傳輸系統

【1.2 系統目標】:

我們的系統係以在 PDA 上實現**四元樹影像壓縮檔的解壓縮**以及針對 Server 端影像壓縮檔的**漸進式影像傳輸**。另外還有針對解壓縮後的還原影像進行**影像精鍊**的動作，以得到使用者所想要的影像清晰度。

四元樹影像壓縮是一種失真的高壓縮率影像壓縮技術，我們的系統是在 PDA 上解壓縮由這門技術所產生的影像壓縮檔，還原成失真的原始影像。

四元樹影像壓縮可以依照不同的影像失真程度，壓縮成若干個影像壓縮檔。在還原影像時，我們會先解壓縮最失真的壓縮檔; 爲了讓影像更清晰，於是我們再依序解壓縮其他影像失真程度較低的壓縮檔，讓原本模糊的影像愈來愈清晰，而這種動作，就稱叫**影像精鍊**。

每位元使用者對於影像的清晰度有不同的接受程度，我們所有經過壓縮過的影像壓縮檔會置放在我們的 Server 端上; 而我們的系統會經過網路連線，從 Server 端下載壓縮檔至 PDA 上，然後進行解壓縮的動作。當使用者不滿意影像的清晰度時，便會再次下載較不失真的影像壓縮檔，然後讓之前的圖片變得更清晰，直到滿意爲止。而這種功能，就是我們系統所欲實現的**漸進式影像傳輸**。

【1.3 系統使用者】:

四元樹影像壓縮雖是一種高壓縮率的影像壓縮技術，但在解壓縮的過程中必須依照我們的壓縮格式進行解壓縮，在執行效能上會付出一些代價; 因此我們的系統是針對在儲存空間較小的行動裝置 PDA 上，也就是我們的系統使用者就是**想要在 PDA 上儲存多數影像**的使用者。

【1.4 系統功能】:

依照使用者的需求，我們基本的系統功能如下:

(1)PDA 取得與 PC 的連線，並且能夠從 PC 上下載影像壓縮檔至 PDA 上。

(2)在 PDA 上解壓縮我們所下載過的影像壓縮檔。

(3)實現漸近式影像傳輸功能。

另外，根據使用者的特殊需求，我們加上了另一項功能:

(4)還原影像時，只讓影像的某區塊變得更清晰。

【1.5 系統限制】:

我們使用的影像壓縮檔是由其他專案小組開發出來的四元樹影像壓縮系統，而且四元樹影像壓縮技術有一些使用條件，因此我們有一些系統限制:

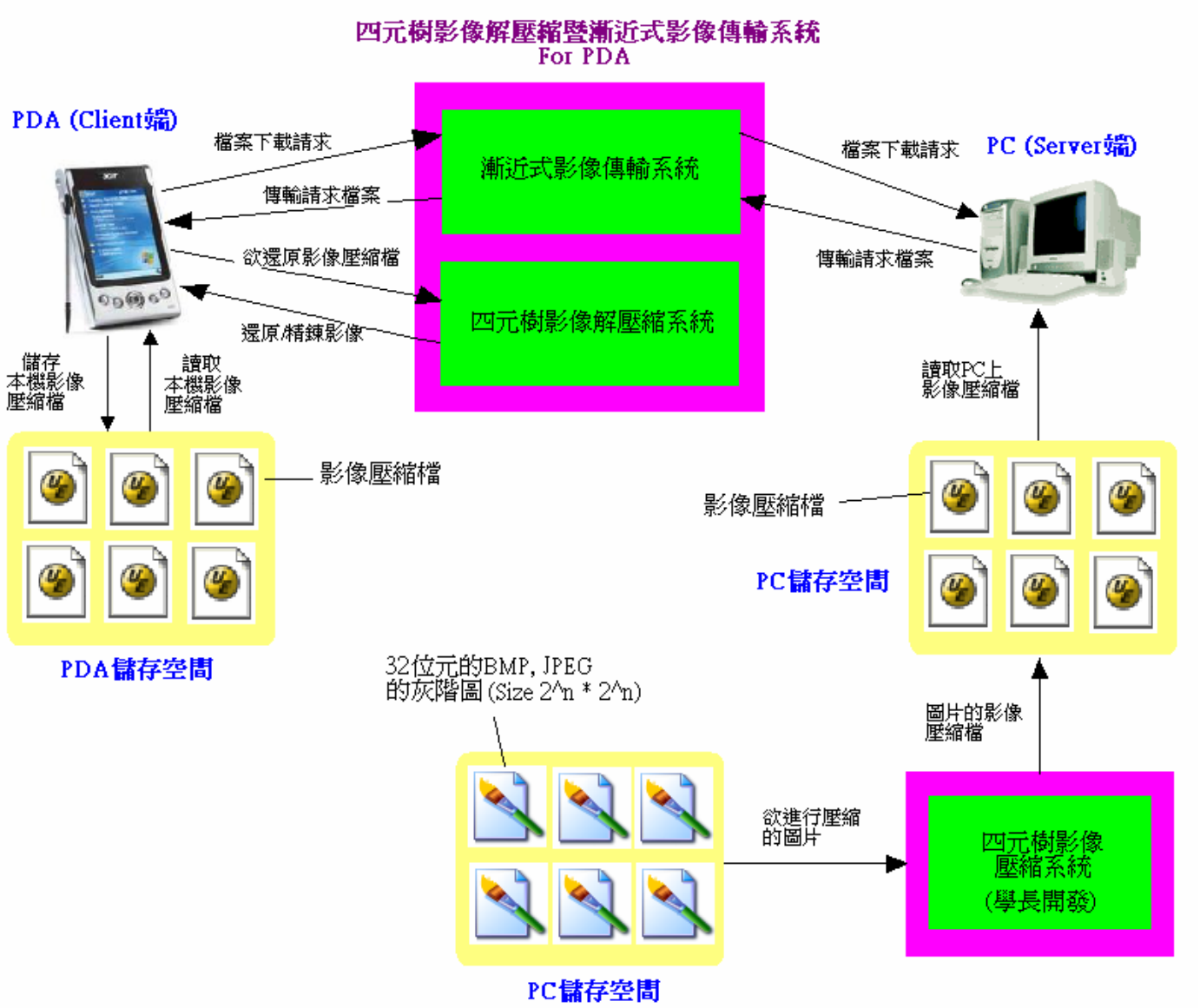
(1)所壓縮的影像只能爲 32 位元元的 bmp, jpeg 灰階圖，且影像的大小只能爲 $2^n * 2^n$。

(2)四元樹影像壓縮技術本身是一種會讓影像失真的壓縮技術。
(3)硬體限制上，PDA 本身 CPU 執行效能和記憶體空間大小相對 PC 要差得多，因此系統在 PDA 上執行效能會較差。

【1.6 系統開發環境】:

(1)開發環境: Microsoft Visual Studio 2003。
(2)開發語言: Visual Basic.NET。

【1.7 系統藍圖】:



影像壓縮檔爲*.byte 的二進位檔案格式，右下角的**四元樹影像壓縮系統**是由以前學長開發的系統。而我們的系統分爲**漸近式影像傳輸系統**以及**四元樹影像解壓縮系統**兩個子系統，前者用來做爲 PDA 以及 Server 端傳遞檔案或訊息的平臺; 後者則是能夠將存在 PDA 上的影像壓縮檔進行解壓縮，以達到還原失真的原始影像和影像精鍊的功能。

【1.8 系統藍圖概說】:

我們的使用者，如果想要在自己的 PDA 上儲存多數的圖片，但不想因爲這些圖片吃掉許多儲存空間; 所以想利用四元樹影像壓縮技術，將這些圖片壓縮成影像壓縮檔存在 PDA 上，然後利用我們的系統來瀏覽這些壓縮過的圖片，那麼:

(1)使用者必須先使用學長開發的**四元樹影像壓縮系統**，將 Server 端想要儲存在 PDA 上的圖片壓縮成影像壓縮檔。

(2)使用者想要這些影像壓縮檔存放在 PDA 上，但是只想存自己認爲清晰程度夠好的影像，所以利用我們在 PDA 上的系統，連接到 Server 端，然後下載我們壓縮過的影像壓縮檔。

(3)我們的系統在下載影像壓縮檔後，除了會存在 PDA 的儲存空間，並且會立即解壓縮這個影像壓縮檔。

(4)若使用者覺得影像不夠清晰，則再從 Server 端下載清晰度較高的影像壓縮檔; 若使用者覺得影像夠清晰了，則跳至步驟(7)。

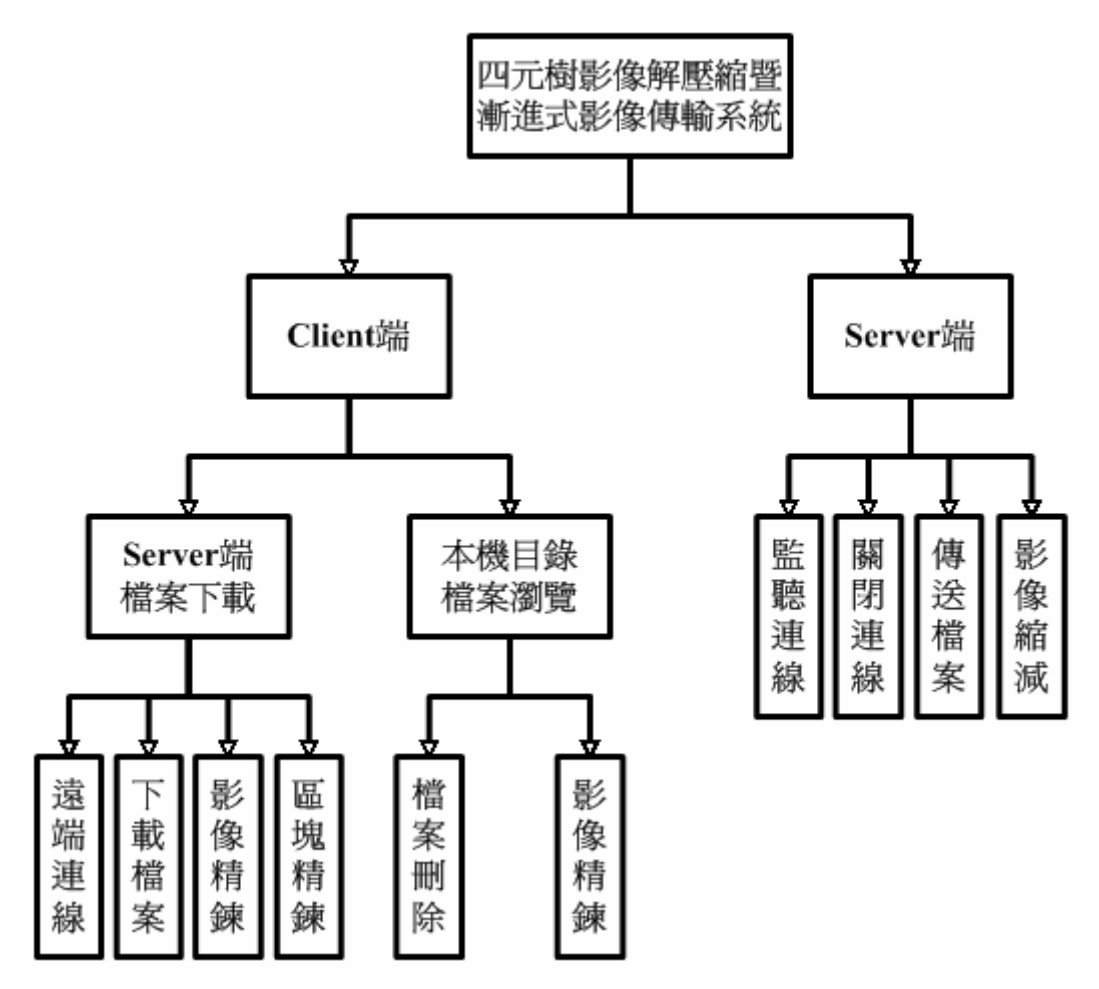
(5)若影像已達到當初壓縮時的最清晰程度，則 Server 端傳「檔案不存在」的訊息給系統; 否則 Server 端會傳較清晰的影像壓縮檔給系統，然後系統再**精鍊**原本較模糊的圖片，讓它變得更清晰。

(6)重覆第(4)個步驟。

(7)影像已經根據我們想要的清晰程度下載到 PDA 上了，我們的系統允許本機瀏覽的功能，所以以後如果想要重新瀏覽圖片，使用我們的系統就可以了。

【二、系統功能】

【2.1 系統功能結構圖】:



【2.2 系統功能】:

(2.2.1 Server 端)

(1) 監聽連線：

開啓本機 port，以提供服務讓 Client 端可以下載影像壓縮檔。

(2) 關閉連線：

關閉本機提供下載壓縮檔的服務。

(3) 傳送檔案：

當 Client 端發出壓縮檔下載請求，傳送指定壓縮檔給 Client 端。

(4) 影像縮減：

當 Client 端所要求的還原影像是區域精鍊時，將指定影像壓縮檔進行四元樹縮減的動作，以求得該精鍊區域的壓縮資料。

(2.2.2 Client 端)

(1) Server 端檔案下載：

(a) 遠端連線：

連線到開啓下載服務的 Server 端，以取得壓縮檔下載服務。

(b) 下載檔案：

向 Server 端提出指定壓縮檔下載請求，並在本機進行解壓縮的動作，以瀏覽影像。

(c) 影像精鍊：

從 Server 端下載瀏覽指定影像後，便可針對指定影像向 Server 端提出下載精鍊壓縮檔的要求，然後在本機上精鍊指定影像。

(d) 區塊精鍊：

在 Client 端選擇一塊影像區域給 Server 端，以取得該影像區域的影像壓縮資料。

(2) 本機目錄檔案瀏覽：

(a) 檔案刪除：

將 Client 端某個存在的壓縮影像相關壓縮檔進行刪除的動作。

(b) 影像精鍊：

從本機上瀏覽/精鍊所存放的影像壓縮檔。

【三、系統設計】

【3.1 系統類別與模組】:

(3.1.1 Server 端)

介面:

- Init
- ServerListen
- ServerCloseListen

FormTCPListener:

- StartListen
- SendDIRList
- CloseListen
- SendFileStream
- SendBlockStream
- WriteData

DownSizeQuadTree:

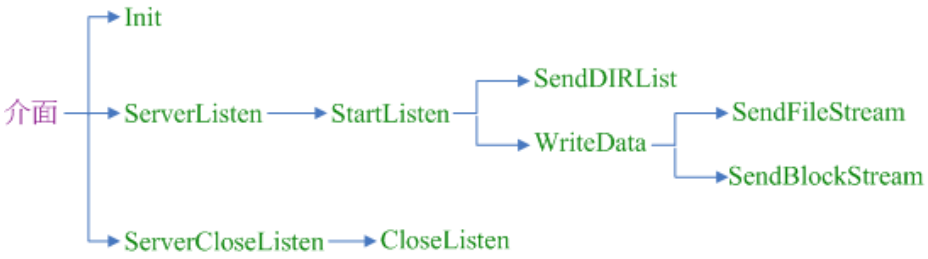
- Read_QuadTree
- Byte_To_Bit
- Eight_To_Seven
- QuadTree_Traversal
- Read_PreCondition
- Travel

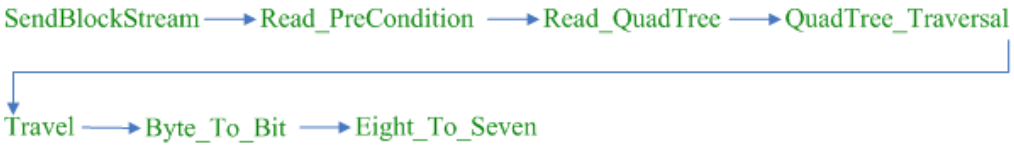
(3.1.2 Client 端)

- 介面:
  - Init
  - ServerConnect
  - CloseConnect
  - ImageDownload
  - ImageDelete
- FormTCPClient:
  - CreateSocket
  - GetFileList
  - CloseSocket
  - FileDownload
  - WriteData
- subTabPage:
  - OpenImageFile
  - AppendImage
  - SelectArea
- QuadTree
  - Read_PreCondition
  - Read_QuadTree
  - QuadTree_Traversal
  - Read_CodeBook
  - First_Compr
  - ReadIndex
  - Recover_Image
  - Recover_Block
  - Append_Image
  - Match
  - Travel

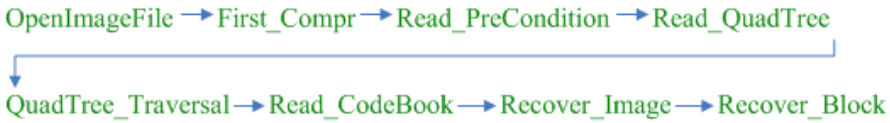
【3.2 類別模組關聯圖】:

(3.2.1 Server 端)





(3.2.2 Client 端)



【3.3 類別與模組說明】:

(3.3.1 Client 端):

類別：介面	
模組	說明
Init	將程式介面初始化
ServerConnect	取得與 Server 端的連線，並可開始傳送/接收資料
CloseConnect	關閉與 Server 端的連線
ImageDownload	可從 Server 端下載所需的圖片檔案
ImageDelete	刪除本機上該圖片的相關檔案

類別：FormTCPClient	
模組	說明
CreateSocket	開啓一個 TcpClient 物件，連線到 Server 端
GetFileList	取得 Server 端上的圖片目錄
CloseSocket	關閉 TcpClient 物件
FileDownload	從遠端目錄下載檔案
WriteData	傳送訊息字串

類別：subTabPage	
模組	說明
OpenImageFile	開啓使用者選取的圖片
AppendImage	從遠端下載檔案，將選擇區域或全部區域進行影像精鍊
SelectArea	選擇圖片上欲精鍊的區塊

類別：QuadTree	
模組	說明
Read_PreCondition	讀取影像壓縮檔的檔頭(如編碼簿名稱、四元樹序列長度等等…。)
Read_QuadTree	讀取四元樹內容資訊
QuadTree_Traversal	利用 DFS 走訪方式將圖檔的 x,y 座標以及區塊大小
Read_CodeBook	讀取編碼簿內容資訊
First_Compr	開啓檔案並且進行相關讀取檔案處理
ReadIndex	必須進行還原影像區塊時，從影像壓縮檔讀取一個編碼簿索引值。
Recover_Image	還原圖片
Recover_Block	還原圖片某區塊
Append_Image	進行圖片精鍊

Match	找到子節點的上一層父節點
Travel	走訪四元樹的內部及樹葉節點

(3.3.2 Server 端):

類別：介面	
模組	說明
Init	將程式介面初始化
ServerListen	處理連線動作，並開始監聽 Client 端的連線要求
ServerCloseListen	結束監聽 Client 端的連線要求

類別：**FormTCPListener**	
模組	說明
StartListen	開啓監聽 Client 端的連線要求
SendDIRList	傳送檔案列表
CloseListen	結束監聽並關閉 server 物件
SendFileStream	傳送四元樹完整資料檔案
SendBlockStream	傳送四元樹區塊資料檔案
WriteData	傳送訊息字串

類別：**DownSizeQuadTree**	
模組	說明
Read_QuadTree	讀取四元樹內容資訊
Byte_To_Bit	將每個 Byte 值化成 Bit 表示方式
Eight_To_Seven	
QuadTree_Traversal	利用壓縮四元樹的方式取得所需的區塊資料
Read_PreCondition	讀取影像壓縮檔的檔頭(如編碼簿名稱、四元樹序列長度等等…。)
Travel	走訪四元樹的內部及樹葉節點

【四、四元樹影像解壓縮暨漸近式傳輸】

【4.1 四元樹影像壓縮技術淺談】:

四元樹影像壓縮技術主要是以儲存**失真**的影像資訊，以得到節省儲存空間的目的; 而我們所儲存的失真影像資訊，主要是把我們的影像儲存格式，從二維 pixel 值的陣列結構，化爲四元樹樹狀結構:

186	190	182	77	85	87	80	78
182	88	80	78	80	88	81	82
76	71	70	62	60	62	64	63
74	71	68	66	67	65	60	59
64	68	77	68	74	72	78	80
70	72	68	72	66	66	78	76
60	66	57	59	63	66	62	68
61	63	58	56	60	68	65	66

摘錄自洪政霖學長的報告

例如上圖是一塊 8*8 的灰階圖，我們若以 BMP 點陣圖格式儲存該灰階圖所有 pixel 值，則我們的儲存格式是二維的。

四元樹影像壓縮技術是將影像分割成若干個區塊，每個區塊內的灰階值差異性不會很大; 一旦區塊被分割完成，我們將這些區塊，利用**區塊取樣技術**，歸納成許多相同尺寸的區塊，這些區塊會被紀錄在一個**編碼簿**(Codebook)的檔案中。然後在壓縮亦或是解壓縮的過程，我們會針對每一個被分割出來的區塊，給定一個編碼簿的索引值，表示我們原先這個區塊最類似編碼簿某個索引值指定的區塊，然後我們根據這個區塊進行壓縮或解壓縮的動作。

有關區塊取樣技術請參閱洪政霖學長的 Paper「植基於四元樹的低位元率漸進式影像傳輸技術」。

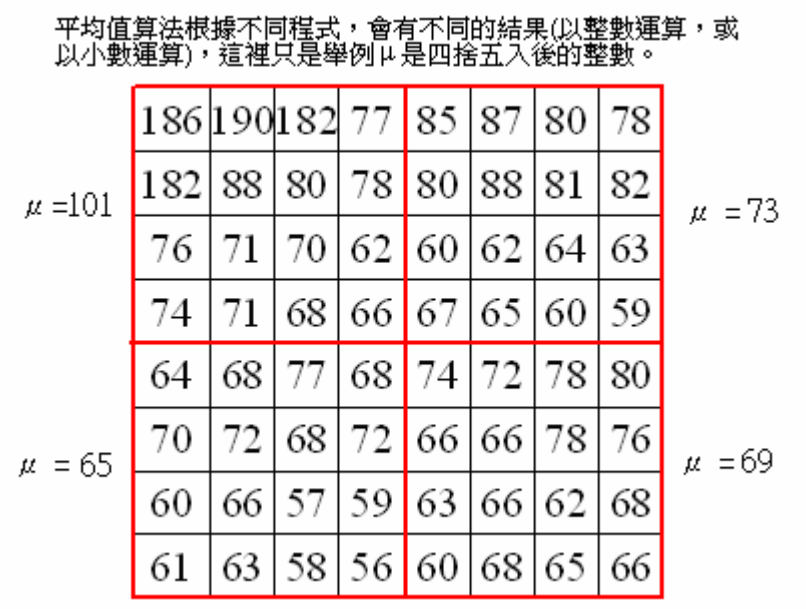
我們想讓一個區塊內的灰階值差異性不大，那麼我會給定一個**門檻值**(threshold)爲 ***th***，以及算出這個區塊內所有灰階值的平均數爲 μ 。若:

$|$*該區域的每個灰階值* $xi - \mu| \leq th$ *for all i,* *(式 1)*

則我們稱這個區塊的灰階值差異性在 th 之間。若存在至少一個點 x’，使得 $|x' - \mu| > th$，則我們認爲這個區塊灰階值差異性不夠小，因此我們會將它分割成四個小區域，然後再判別這四個小區域各自灰階值的差異性，直到所有被分割的區域皆滿足式 1 的不等式。

譬如說我們想分割上面那張 8*8 的影像，我們給定門檻值 th = 50，我們計算這 8*8 個灰階值的平均值 μ =76，但像第 1 個 pixel 值=186，$|186 - \mu| = |186 - 76|$

= 100 > th = 50。由於有些點分佈狀況不是我們所想要的，所以我們會將這 8*8 大小的區域分割成四個等分:

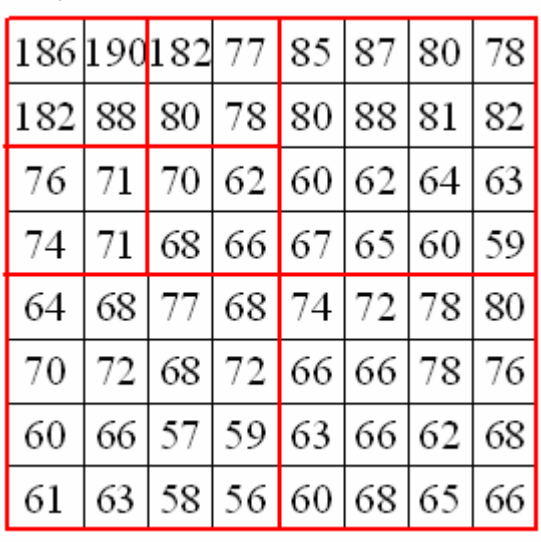


我們要求影像的大小爲 $2^n * 2^n$，因此若我們定義一個區塊的資訊爲 (x, y, size)，其中(x,y)表示區塊左上角的座標位置，而 size 表示該區塊寬以及高的大小。那麼我們所分割出來的四個小區塊，它們各自的區塊資訊爲:

(x,y, size/2), (x+size/2,y, size/2), (x,y+size/2, size/2), (x+size/2,y+size/2, size/2)

因此，我們根據所分割的區塊，再判斷各自區域內的灰階值和平均值的差是否小於等於門檻值，決定是否再向下切割。

最後我們根據門檻值 th = 50 所切割出來的區塊如下:

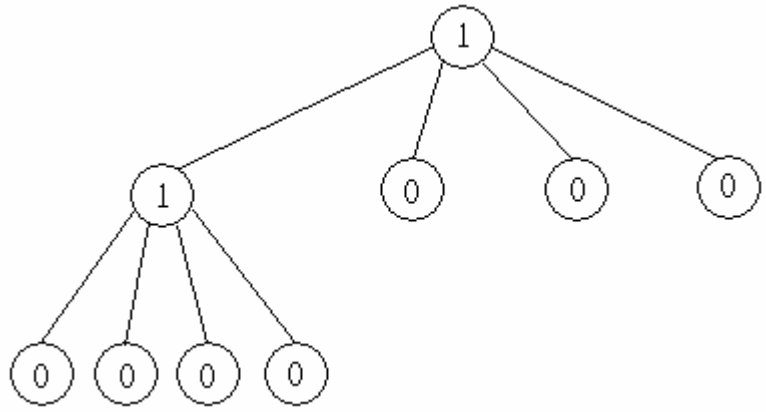


那麼我們如何將所切割出來的區塊化做四元樹樹狀結構呢?

想像我們有一個待檢查的區域 A，它在四元樹中表示爲一個節點，我們判斷區域 A 所有灰階值是否滿足我們門檻值的條件; 若是至少有一個灰階值不滿足條件，則我們會將這個區塊再分成四個小區塊。在樹狀結構中，這種現象就是區

域 A 表示的樹狀節點長出四個子節點。

我們若以 0 表示樹狀節點中的樹葉節點(leaf node)，以 1 表示內部節點(internal node); 至於擁有同樣的父節點(parent node)的四個子節點(child node)，它們在樹狀結構中的排列方式，是由左上、右上、左下、右下的順序存放。因此我們上面 8*8 灰階圖所分割出來的區塊的四元樹結構如下:



我們儲存四元樹樹狀結構時，是以 DFS 走訪這棵樹，然後將走訪到的每個節點的值(0 或 1)搜集起來; 因此上面的四元樹，我們存放它時，會存放:

"110000000"這個序列，每個 0 或 1 以一個 bit 來存放，以節省空間。

當然四元樹只能表示我們將影像分割成怎麼樣的情況，我們並不知道某個區塊的灰階值資訊。這些灰階值資訊會根據前述所說的，利用區塊取樣技術，將若干個區塊灰階值紀錄在**編碼簿**中。換句話說，編碼簿中有許多相同大小的區塊(每個區塊給予一個索引值編號)，我們最後每一個不能再分割的區塊(在樹狀結構中的樹葉節點)，給予一個索引值，這個索引值表示這個區塊和編碼簿中編號爲這個索引值的區塊的灰階值最接近。我們還原影像的時候，就根據這個索引值來還原影像。

當然每個區塊的大小或多或少不一致，我們編碼簿中所有的區塊大小應當是最小的。這樣一來，其他比它大的區塊想要根據編碼簿中某個區塊的灰階值來還原，那麼我們會將被參考的區塊大小擴大，以求填滿整個大區塊。

38	33	34	35
33	36	36	31
34	37	41	30
51	48	38	47

編碼簿某參照區塊

擴大

---------->

38	38	33	33	34	34	35	35
38	38	33	33	34	34	35	35
33	33	36	36	36	36	31	31
33	33	36	36	36	36	31	31
34	34	37	37	41	41	30	30
34	34	37	37	41	41	30	30
51	51	48	48	38	38	47	47
51	51	48	48	38	38	47	47

參照左邊編碼簿某區塊還原 8*8 影像

因爲是根據編碼簿來還原圖片，所以我們還原的影像會失真。

【4.2 影像解壓縮】

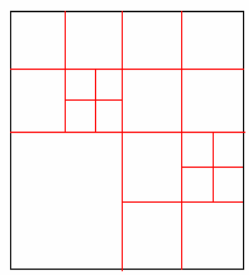
我們有四元樹的序列，以及每個樹葉節點所參照的索引值(想像四元樹序列和若干索引值依序存放在同樣一個檔案中)，那麼我們在影像解壓縮時的流程如下:

```
Quadtree_Decompress( ){
    //前置處理: 從壓縮檔讀取檔頭
    從壓縮檔讀取四元樹序列;
    DFS(四元樹的 root); //從四元樹的樹根開始走訪
}
DFS(現在走訪的節點 u){
    if(u 爲內部節點){ //走訪 u 的四個子節點
        DFS(u 的第一個子節點);
        DFS(u 的第二個子節點);
        DFS(u 的第三個子節點);
        DFS(u 的第四個子節點);
    }
    else{ //u 爲樹葉節點
        從壓縮檔讀取一個編碼簿索引值;
        根據這個索引值還原節點 u 所表示區域的影像;
    }
}
```

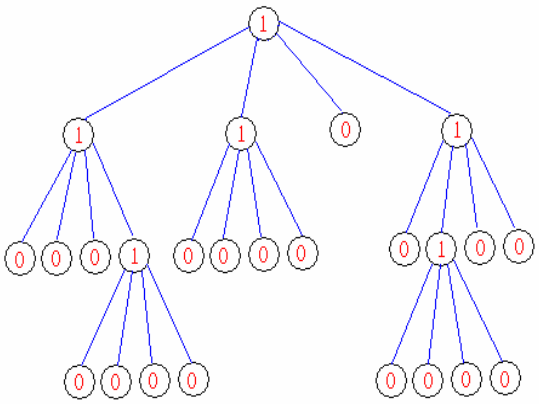
還原影像的區塊都是樹狀結構的樹葉節點，區塊間不會有交集，所以在還原影像的時候不會有某個區塊的還原影像蓋掉其他區塊的影像。

【4.3 影像精鍊】:

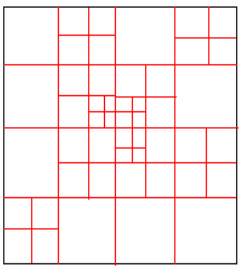
我們每一個區塊會根據它所有的灰階值和平均值的距離是否皆小於門檻值決定是否再分成四個小區塊。一般而言，門檻值愈小，需要被分割的區塊愈多，也就是說，我們的門檻值愈小，我們最後所得到的樹狀結構會愈大:



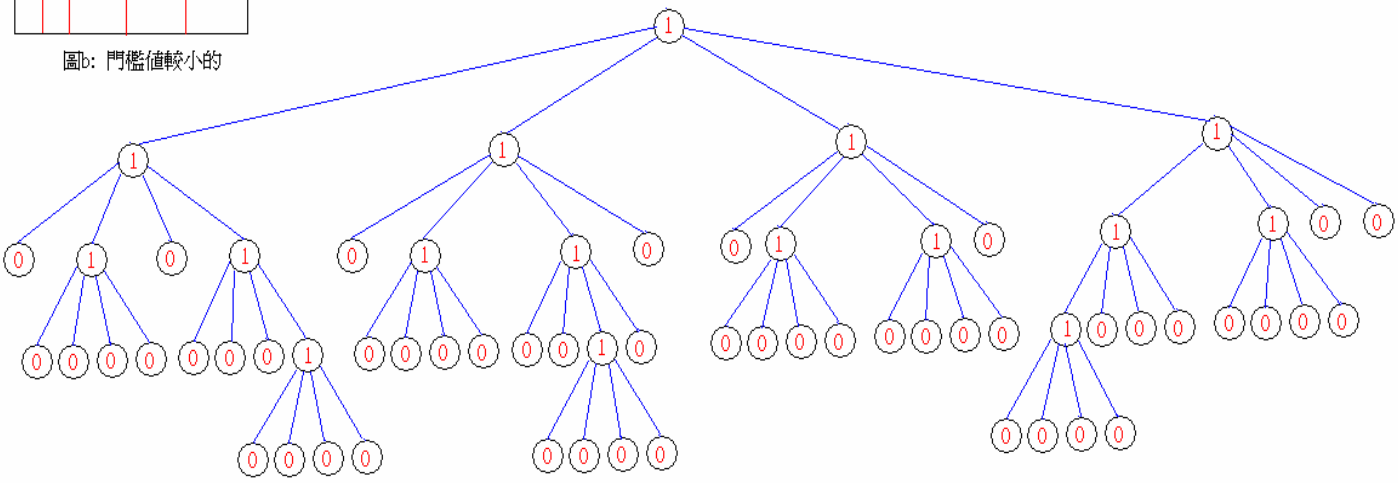
圖a: 門檻值較大的



四元樹序列:
1100010000100000101000000



圖b: 門檻值較小的



四元樹序列: 1101000001000100001010000100100000010100001000001010000100000111000000001000000

假設我們給定若干個門檻值 TSi, i=1, 2, 3, 4, 5，其中

TS1 = 90, TS2 = 70, TS3 = 50, TS4 = 30, TS5 = 10，根據每個門檻值，門檻值愈小，所需被分割的區塊會愈多; 同樣地因爲每個區塊內的灰階值誤差愈小，所以整張圖片也會變得更清晰。若以一張 128*128 的灰階影像 lena，依照這五個門檻值切割區域，最後所還原的影像如下所示:

lena 灰階圖原始影像如下:

門檻值 = 90
影像模模糊糊的，看不出是什麼?

門檻值 = 70
影像變清晰了，帽子? 眼睛?

門檻值 = 50
影像變更清晰了，是一個女人。

門檻值 = 30
影像變更清晰了

門檻值 = 10
影像變更清晰了

我們會發現雖然解回的影像和原圖比較，失真的情況很嚴重，但是大致上我們可以看出影像的輪廓是一個女人—lena。

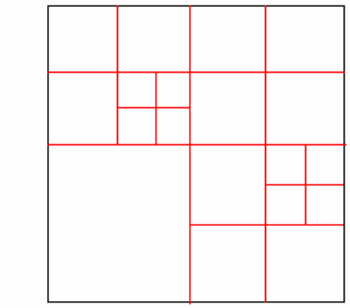
另外如果仔細看解回來的圖片，我們看到會有方塊效應，這是因爲我們解回原圖時，是以一個區塊一個區塊根據所賦予的編碼簿索引值的灰階值序列貼圖。

一般處理方塊效應的方式是將處理過的圖再進行優化: 利用像內插演算法，將區塊的每個 pixel 值，進行運算處理。

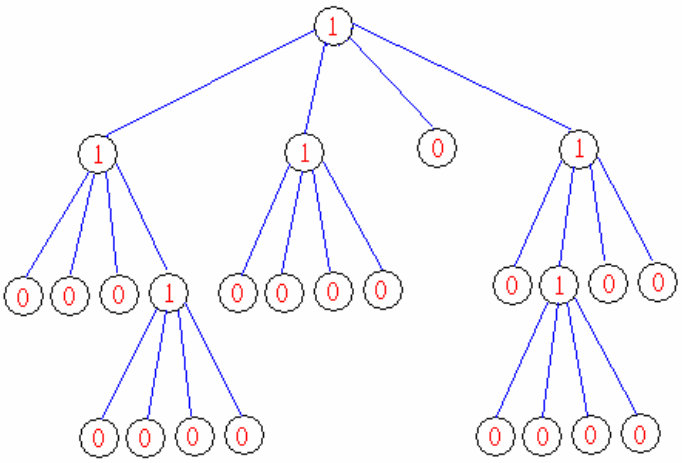
不過由於我們在解回圖檔時是在 PDA 上進行解壓縮，我們測試過如果利用內插演算法來優化影像，在執行效率上會很差，因此我們的系統就不實現影像優化的處理。

根據不同的門檻值，我們所得到的影像清晰度也不盡相同，在上例看下來，影像是愈來愈清晰的; 考慮一種情境: 一開始我們解回來的影像是門檻值 90 的那張圖片，由於我們不滿意影像的清晰度，所以我們進行**精鍊**的動作，也就是讓影像的門檻值變成 70，如果再不滿意就再變成 50、30、10。

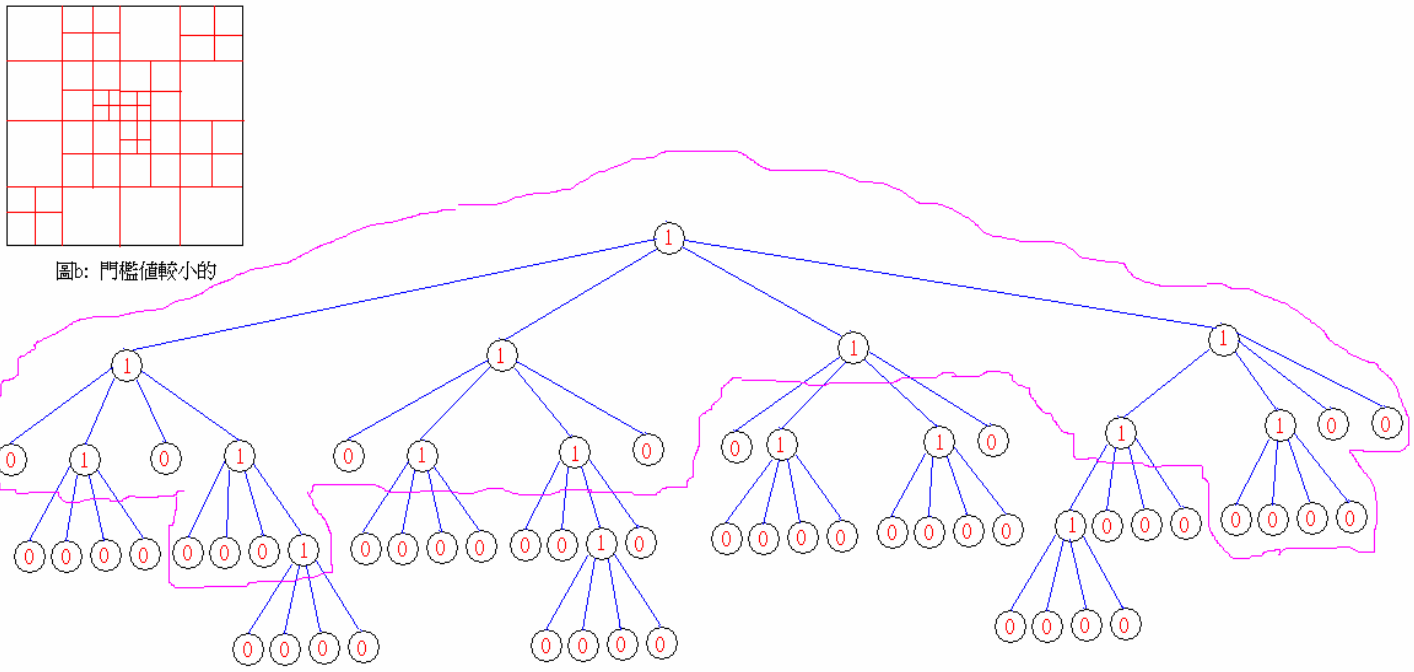
不同的門檻值，所切割出來的四元樹的大小也不盡相同，但它們之間是有關係的: 同樣的一張影像在切割區域時，以不同的門檻值進行切割。那麼切割出來的結果，門檻值較大的四元樹是門檻值較小的四元樹的部分樹(partial tree)。



圖a: 門檻值較大的



上圖是之前舉的例子，一開始可能在某個門檻值，所切割出來的四元樹如上所示，那麼當我們以門檻值較大的進行切割的動作，我們如得到之前的結果:



圖b: 門檻值較小的

四元樹序列: 11010000010001000010100001001000000101000010000001010000100000111000000001000000

圈選的部分就是以門檻值較大進行區塊分割的部分樹(partial tree)。

考慮一開始以門檻值較大進行區塊分割的影像還原，我們在遇到樹葉節點時(partial tree 的 leaf node)，我們會讀取相對應的編碼簿索引值，然後還原該區域的影像，在走訪完整棵樹後，我們的影像就會全部被還原了。

那麼在門檻值較小的四元樹，我們在什麼情況下必須還原影像? 在圈選出來的部分樹的樹葉節點，我們在之前就已經還原相對應的區塊影像，所以我們沒有必要再還原一次。因此我們只會在新產生的子樹的每一個樹葉節點(也就是沒圈選到的樹葉節點(值爲 0))，進行影像還原的動作—這個過程就叫做**影像精鍊**。

我們影像精鍊的流程如下:

```
/*註: 我們必須將上一次影像還原的四元樹存放下來(舊四元樹序列，才能做展開新子樹的判斷*/
Append_Image( )
{
    //前置處理: 讀取檔頭
    讀取精鍊壓縮檔的四元樹序列; //新四元樹序列
    Append_DFS(舊四元樹樹根，新四元樹樹根);
    儲存新四元樹序列，當作下一次精鍊的舊子樹;
}
```

```
Append_DFS(目前舊子樹走訪節點 u，目前新子樹走訪節點 v)
{
    if(u 節點值 = v 節點值){     //表示沒有產生新子樹
        if(u 爲內部節點){  //走訪四個子樹
            DFS(u 的第一個子節點，v 的第一個子節點);
            DFS(u 的第二個子節點，v 的第二個子節點);
            DFS(u 的第三個子節點，v 的第三個子節點);
            DFS(u 的第四個子節點，v 的第四個子節點);
        }
        else{       //舊的樹葉節點影像已還原過
            //do nothing
        }
    }
    else{       //表示有新子樹的產生(舊樹葉節點變新子樹出來)
        NewTree_DFS(v);  //有新子樹，其樹葉節點就要精鍊影像區塊
    }
}

NewTree_DFS(目前新子樹走訪節點 v)
{
    if(v 是內部節點){  //走訪 v 的四個子樹
        NewTree_DFS(v 的第一個子節點);
        NewTree_DFS(v 的第二個子節點);
        NewTree_DFS(v 的第三個子節點);
        NewTree_DFS(v 的第四個子節點);
    }
    else{       //是新子樹的樹葉節點，我們還原它的精鍊影像
        從壓縮檔讀取一個編碼簿索引值;
        根據這個索引值還原節點 v 所表示區域的影像;
    }
}
```

【4.4 漸近式影像傳輸】:

我們針對某張影像有了若干個不同清晰度的檔案後，我們在做漸近式影像傳輸時，先下載影像失真最嚴重的壓縮檔，然後在 PDA 上還原，而我們如果想要讓影像變得更清晰，於是我們再從 PC 上下載門檻值較小的影像壓縮檔，然後在還原影像的時候，是在之前較模糊的影像某些區塊(該次精鍊所新分割出來的區塊)，覆蓋一個新的區塊還原影像。

漸近式影像傳輸爲了滿足使用者這樣的需求，於是在和 PC 取得連線後，首

先會有一個列表，列出使用者在 PC 上所壓縮的影像壓縮檔。每一張灰階影像所壓縮出來的影像壓縮檔有若干個，依照門檻值由大至小，我們將由同樣灰階影像壓縮出來的壓縮檔給予編號。譬如說上面 lena 的灰階影像，依照五個不同的門檻值: 90、70、50、30、10 壓縮出來的壓縮檔，我們給定它的檔名分別爲:

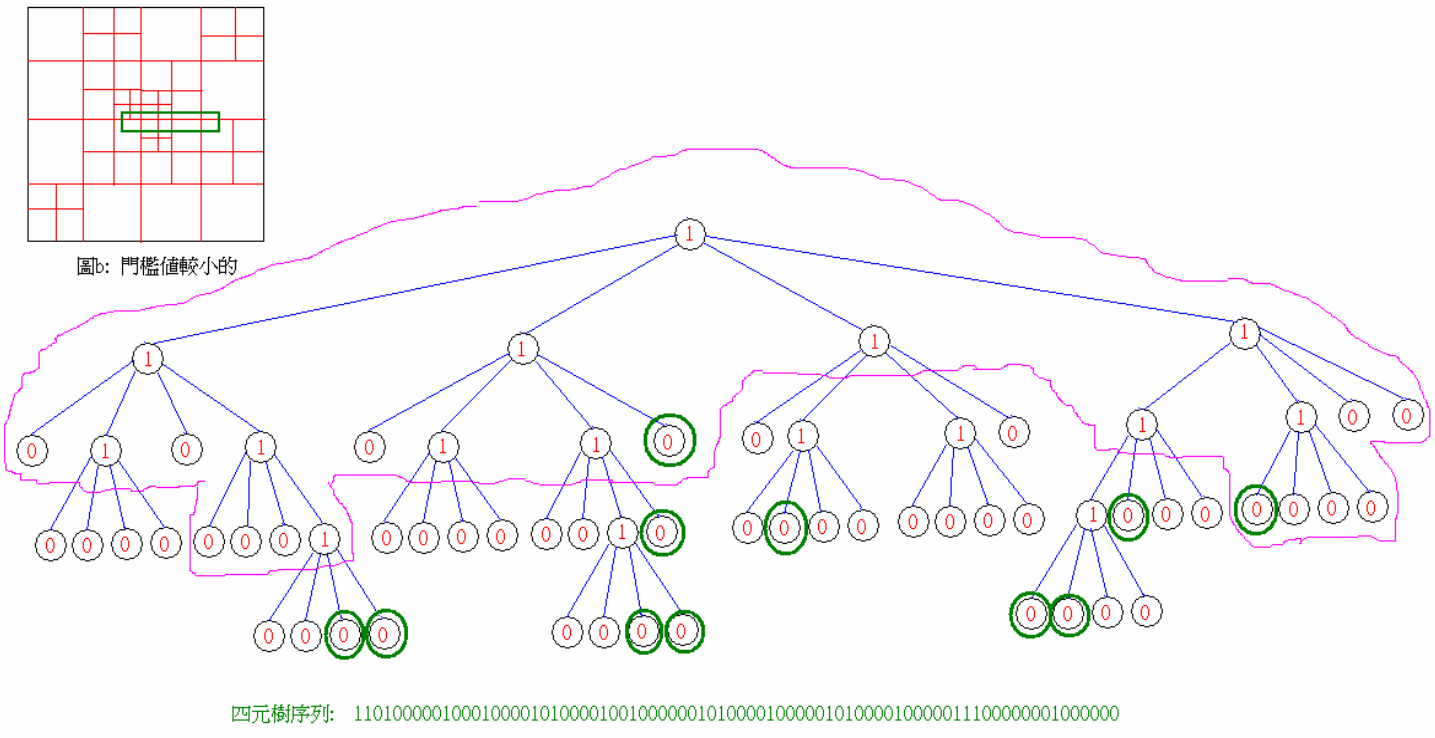
lena_1, lena_2, lena_3, lena_4, lena_5。副檔名型態是"byte"二進位檔案。

我們一開始從 PC 上下載的壓縮檔爲 lena_1.byte，然後在 PDA 上利用我們的系統還原影像。待使用者覺得影像不夠清晰時，便再下載 lena_2.byte，然後精鍊由 lena_1.byte 還原的影像，讓它變得更清晰。重覆這樣的步驟，直到使用者滿意現在的清晰度，亦或是已經精鍊完 lena_5.byte，也就是還原最清晰的影像。

「下載並且還原影像或精鍊影像」在使用者下達一個動作就完成的功能，便是我們漸近式影像傳輸所要實現的。

【4.5 影像區域精鍊】:

上頭所講的**影像精鍊**係以將上一層(門檻值較大的)所還原的影像，透過較小門檻值所分割出來的四元樹的新的樹葉節點，以提升影像的清晰度。由於是針對每個新樹葉節點進行精鍊的動作，但有時候如果我們只想要精鍊某區域的影像，而不是精鍊整張圖，我們要如何達成?



一樣從之前的例子開始，我們左上角是我們的影像以及它依照某個門檻值所分割出來的若干個區域，而下面是我們這些分割區域所組合出來的四元樹。

我們在左上影像圖，如果圈選一個綠色框框，方式我們只想要精鍊這個區域的影像(註，這個矩形由於不是正方形，如要儲存資訊，則必須同時儲存矩形區域的 width 以及 height)。從分割區域所對應的四元樹節點，我們可以知道由綠色圓圈圈選出來的

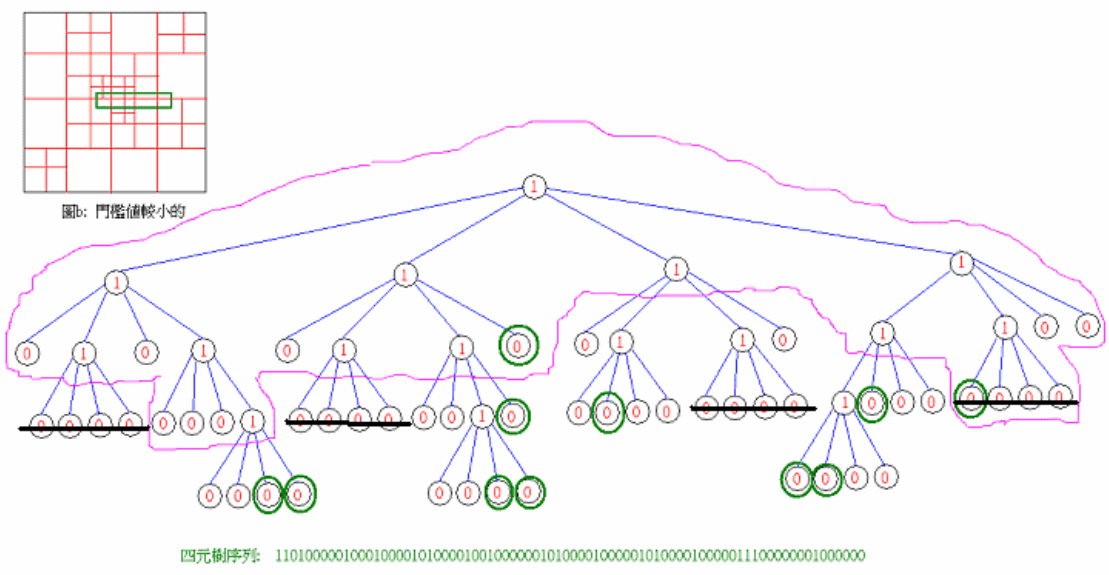
節點，會和我們選定的區域有交集。換句話說就是當我們走訪到這些節點時，我們才會做影像精鍊的動作。(註: 但如果我們在上一層就已經還原過的節點，在這一次精鍊就不會再還原了，如上圖有兩個點和該區域有交集，但我們不精鍊它)。

比較簡單解決區域精鍊的演算法是同樣地我們擁有像全部精鍊時所保有的四元樹，也就是上面的那棵四元樹; 然後我們會有所欲被精鍊的選定區域的(x, y, size)資訊，在走訪整棵樹的過程中，同樣地當我們遇到新產生的樹葉節點時，我們之前全部精鍊的方式是直接讀取索引值然後進行精鍊的動作; 在區域精鍊時，我們判斷該新樹葉節點是否和我們選定的區域(x, y, size)有無交集，如果有交集，我們才會進行讀取索引值然後精鍊的動作。

但是實際上，假設我們所選定的區域相對整張影像區域較小時，我們所浪費的儲存空間也會愈多; 像上面的例子，我們新產生的樹葉節點有 36 個，但我們實際上只還原 9 個節點。當四元樹愈大棵時，這種浪費空間的現象會愈明顯。

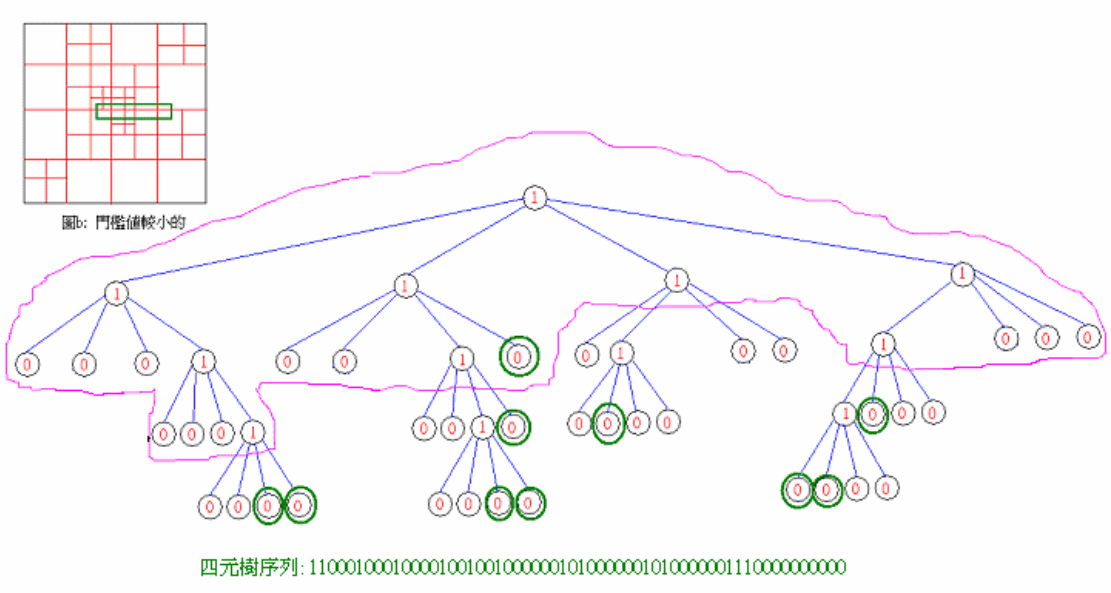
也因此我們會將我們精鍊的演算法做改良，我們儲放影像的四元樹序列，只會留下有必要的節點; **一個區域精鍊下的樹狀節點 i 如果稱做有必要，則它所對應的正方形區塊{xi, yi, sizei}勢必會和我們所圈選的矩形區域{x, y, width, height}有所交集。**像上面例子圖中由綠色圓圈圈選出來的，且是新產生的樹葉節點的這些節點，它們以及它們的祖先節點(這些樹葉節點一定是由較大的區塊分割成小區塊，所以這些區塊一定也會和選定的矩形區域有所交集)，都是必要的節點。

當然我們在儲存四元樹時是將這些必要節點的 silbing nodes 一起保留，因爲我們在 DFS 走訪時是走訪四元樹，一次伸展四個子樹。



上圖畫黑線的地方表示我們不將它保留，我們會發現它們的父親由於是內部節點，所以節點值爲 1，我們在走訪這棵樹後，會產生一棵新的且較小的四元樹，它會將這些所有子樹被刪除的父親節點的值設爲 0; 另外注意最後邊刪除節點的情況，雖然其中有一個節點和這個區域相關，但是我們在上一層就已經將它的影像還原過，所以我們也將之刪除。

因此我們最後保留下來的四元樹樹狀結構如下:



注意是一個節點 A 的四個子樹中至少有一個子樹和選定區域有關，或是節點 A 爲新產生的樹葉節點且和選定區域有關，我們就保留 A 以及 A 的 sibling node; 如果一個節點 A 爲內部節點，且它四個子樹皆和選定區域無相關或是之前已經影像還原過，那麼我們就把節點 A 的子樹砍掉，然後視節點 A 是我們這棵樹的某個樹葉節點。

我們區域精鍊的流程如下:

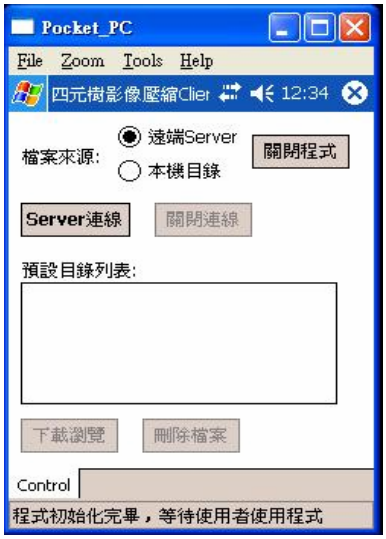
```
//註: 所欲精鍊的區塊資訊位元於影像壓縮檔的檔頭
Area_Append_Image( )
{
    //前置處理: 讀取檔頭; 讀取區塊資訊(x, y, size)
    讀取精鍊壓縮檔的四元樹序列; //新四元樹序列
    Area_Append_DFS(舊四元樹樹根，新四元樹樹根);
    儲存新四元樹序列，當作下一次精鍊的舊子樹;
}

Area_Append_DFS(目前舊子樹走訪節點 u，目前新子樹走訪節點 v)
{
    if(u 的節點值 = v 的節點值){ //表示沒有長出新子樹
        if(u 爲內部節點){ //走訪四個子樹
            DFS(u 的第一個子節點，v 的第一個子節點);
            DFS(u 的第二個子節點，v 的第二個子節點);
            DFS(u 的第三個子節點，v 的第三個子節點);
            DFS(u 的第四個子節點，v 的第四個子節點);
        }
```

```
            else{       //舊的樹葉節點影像已還原過
                //do nothing
            }
        }
        else if(u 爲樹葉節點但 v 爲內部節點){ //表示有新子樹生成
            Area_NewTree_DFS(v); //新子樹的樹葉節點必須做區塊精鍊
        }
        else{ //表示 u 原本的子樹因爲和區塊無關被縮減了
            Area_OldTree_DFS(u); //走訪舊四元樹節點 u 被縮減的子樹
        }
}

Area_NewTree_DFS(目前新子樹走訪節點 v)
{
    if(v 是內部節點){  //走訪 v 的四個子節點
        Area_NewTree_DFS(v 的第一個子節點);
        Area_NewTree_DFS(v 的第二個子節點);
        Area_NewTree_DFS(v 的第三個子節點);
        Area_NewTree_DFS(v 的第四個子節點);
    }
    else{       //若 v 是樹葉節點，進行區塊精鍊的動作
        從壓縮檔讀取一個編碼簿索引值;
        根據這個索引值還原在節點 v 中和我們選定區域有交集的影像;
    }
}

Area_OldTree_DFS(目前舊子樹走訪節點 u)
{
    if(u 是內部節點){  //走訪 u 的四個子節點
        Area_OldTree_DFS(u 的第一個子節點);
        Area_OldTree_DFS(u 的第二個子節點);
        Area_OldTree_DFS(u 的第三個子節點);
        Area_OldTree_DFS(u 的第四個子節點);
    }
    else{
        //do nothing
    }
}
```

【五、系統操作介面與使用】

5.1 連至 Server 端功能執行過程

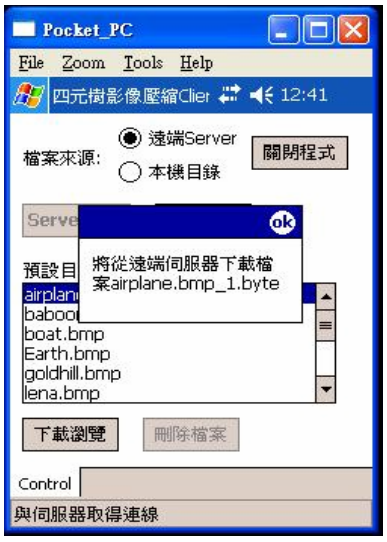
1.Menu



2.連線至 Server 端

3.連線成功秀出子目錄

4.下載瀏覽



5.Server 端傳送第一層狀態

6.圖檔保存至 Client 端

7.傳送完畢 Client 解圖顯示結果

8.漸進式傳輸影像精練 2 層~5 層 第二層下載狀態

9.第二層精練結果

10.簡略至第 5 層精練

11.第五層精練結果

12.超過第五層狀態

13. 精練已經達最深度

14. 區塊精練

15.下載第二層部分精練

16.第二層部分精練結果

17. 簡略至第 5 層部分精練

18. 第五層部分精練結果

18. (同上)部分精練已經達最深度

19.也可刪除檔案

5.2 本機端功能執行過程

本機端功能執行過程如同 **Server** 端執行過程，差別在於部分精練的有無，在本機端是無法進行部分精練的，而在本機端只能進行全精練，但是在進行全精練時要確保與 **Server** 端是連線的狀態，才能進行精練。

【六、四元樹影像壓縮率評估】

壓縮率: 影像壓縮檔加總檔案大小 ÷ 原圖檔案大小(*.bmp,(*.jpg，)) = %

全圖壓縮:

影像壓縮率評估:		
種類 / 比較項目	四元樹影像壓縮	**JPEG**
測試圖片	G_RGB-airplane.bmp	G_RGBairplane.bmp
影像寬、高	512x512	512x512
檔案大小	1.00 MB (1,048,632 位元組)	1.00 MB (1,048,632 位元組)
CodeBook	4128_G_400x300-Cat.jpg.txt.byte	
門檻值	90 70 50 30 10	
壓縮檔案大小	13.6KB (14,012 位元組)	31.7 KB (32,545 位元組)
壓縮率	13.6KB /1025KB=1.3%	31.7KB/1025KB=3%
Performance	最佳	次要
Utility	最佳	次要

影像壓縮率評估:		
種類 / 比較項目	四元樹影像壓縮	**JPEG**
測試圖片	G_RGB-scene.bmp	G_RGB-scene.bmp
影像寬、高	128x128	128x128
檔案大小	17.0 KB (17,462 位元組)	17.0 KB (17,462 位元組)
CodeBook	4256_sssss.bmp.txt_sort.txt.byte	
門檻值	90 70 50 30 10	
壓縮檔案大小	2.00 KB (2,057 位元組)	4.77 KB (4,885 位元組)
壓縮率	2.00KB/17.0KB=11.5%	4.77KB/17.0KB=28%
Performance	最佳	次要
Utility	最佳	次要

結論:四元樹影像壓縮不論是在壓縮率、Performance、Utility 都比傳統 JPEG 壓縮還要來的好，因此非常適合用在 MicroKernel 下的系統中來使用，可以盡量減輕系統的負擔。

影像區域精練 v.s 影像全部精鍊:

影像壓縮率評估:		
種類 / 比較項目	影像全精練	影像區域精練
測試圖片	G_RGB-airplane.bmp	G_RGBairplane.bmp
影像寬、高	512x512	145x91
檔案大小	1.00 MB (1,048,632 位元組)	1.00 MB (1,048,632 位元組)
CodeBook	4128_G_400x300-Cat.jpg.txt.byte	4128_G_400x300-Cat.jpg.txt.byte
門檻值	90 70 50 30 10	90 70 50 30 10
壓縮檔案大小	13.6KB (14,012 位元組)	1.37 KB (1,404 位元組)
壓縮率	13.6KB /1025KB=1.3%	1.37 KB/1025KB=0.1% 1.37 KB/13.6KB=10%

結論:影像大小根據區域精練來說，如果選擇的區域越小則壓縮出來的檔案會越小，反之則越大；影像全精鍊是根據整張圖來進行壓縮，而區域精練則是跟據所選擇的區塊來精練，因次壓縮率區域精練會比全精練還要來的小。

【七、結論】

系統本身著重在增進空間的利用率，透過四元樹影像壓縮技術將圖片壓縮成若干個影像壓縮檔，我們的使用者可以選擇想要的影像清晰度。不過爲了提升壓縮率的效果，我們犧牲了系統執行時間以及提高了影像的失真效果。比起其它的漸進式傳輸的演算法，我們的還原影像的失真率嚴重許多。是往後必須解決的問題。