植基於複雜度分析之無失真資訊隱藏技術

呂慈純
朝陽科技資訊管理系
e-mail：
tclu@cyut.edu.tw

黃政鈞
朝陽科技大學碩士生
e-mail：
s9514644@cyut.edu.tw

摘要

本篇論文提出一個有效的無失真資訊隱藏技術，利用像素其鄰近 8 個像素來計算複雜度，用以預測像素的藏入位元數。此外，為了達成無失真還原原始隱蔽影像，我們將原圖和量化圖的差異值保存起來，並且嵌入隱蔽影像中，以達到無失真還原的目的。

所提的方法主要是改進 Maniccam 和 Bourbakis 所提之資訊隱藏技術，對於一張標準的 Lena 而言，所提方法其較藏入量較 Maniccam 和 Bourbakis 大幅提昇。此外，所提方法不僅可以取出機密資訊，同時還可以從偽裝影像中還原出原始圖形。

關鍵詞：無失真資訊隱藏、算數編碼法。

1. 前言

資訊隱藏依照處理方式的不同，主要可以分成空間域(Spatial Domain)和轉換域(Transform Domain)[2]，空間域的處理方式主要是改變掩護影像(Cover Image)像素，然後將機密資訊(Secret Data)直接藏入，達到資訊藏入的目的。常見的空間域資訊隱藏方法有補丁法(Patchwork)[3]、紋理區塊編碼法(Texture Block)[4]、向量量化編碼法(Vector Quantization)[5]、最低位元取代法(Least Significant Bit, LSB)[6]等，其中最低位元取代法運算容易且資訊藏入量大，具有不可察覺性等優點，所以為空間域中最常使用的一種演算法。

轉換域的技術主要是把空間域的像素值，經過轉換成頻率域的係數值，接著，再將其機密資訊藏到特定或是指定的係數之中。以網路上最流行的 JPEG 圖像為例，它所使用的方法就是將機密資訊藏入到 JPEG 量化後的離散餘弦轉換(Discrete Cosine Transform, DCT)係數中。

資訊藏入量用來衡量資訊隱藏技術的一項重要指標，如何提高藏入量，而且提昇偽裝影像品質，是大家所追求的目標，本篇論文提出一個無失真資訊隱藏技術，其資訊藏入量不但較先前方法多，並且能夠達到無失真回復。此外偽裝影像的影像品質也在人眼可接受範圍內。

2. 相關文獻探討

Maniccam 和 Bourbakis 於 2004 提出一無失真壓縮及資訊隱藏技術[1]，他們的方法一開始先找出特定的藏入點，之後以藏入點鄰近的 8 個鄰居做預測，判斷每個藏入點可以藏入多少位元個數，再利用最低位元取代法的方法嵌入資訊。

Maniccam 和 Bourbakis 的方法中可藏入的像素如圖 1 所示，圖中 x 表示可藏入的點。每一個 x 利用鄰近 8 個像素計算出能夠嵌入的位元個數。以圖 2 為例，假設中心點為 x，其鄰近 8 個像素值為 $NB_1(x)$、$NB_2(x)$，...，$NB_8(x)$。

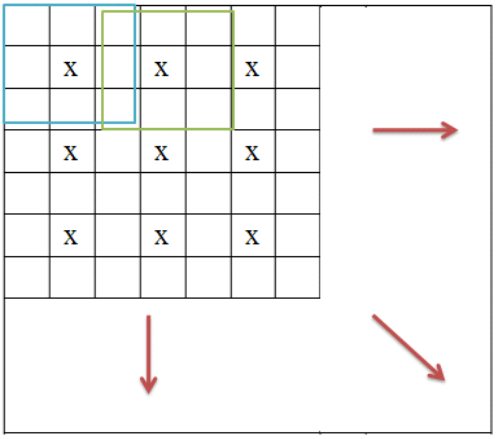


圖 1　x 為能嵌入資訊的位置

$NB_3(x)$	$NB_2(x)$	$NB_1(x)$
$NB_4(x)$	x	$NB_8(x)$
$NB_5(x)$	$NB_6(x)$	$NB_7(x)$

圖 2 像素 x 值的 8 個鄰近像素

接著算出鄰近點的複雜度程度，其計算式為

$$d = \frac{|NB_1(x) - NB_2(x)| + |NB_2(x) - NB_3(x)| + \ldots + |NB_8(x) - NB_1(x)|}{8} \quad (1)$$

。

例如，圖 3 的複雜度為

$$d = \frac{|10-20| + |20-30| + |30-40| + \ldots + |80-10|}{8} = 17.5 \text{ ，}$$

接著將所有可能的複雜度切割成 4 等份，如圖 4 所示。其中，T_1, T_2, T_3, T_4 為門檻值，用來界定不同的複雜度下，藏入點的藏入的位元數。一般而言，門檻值設的越小，其嵌入量越高；反之，門檻值設的越大，所能嵌入的量就越低。假設所設定的門檻值為 2、4、8、16，上面例子中 d = 17.5，其界大於 T_4，故可藏入的位元數為 4。反之，如果 d 小於 T_1，其藏入位元數為 0，則不做任何的嵌入動作。

30	20	10
40	3	80
50	60	70

圖 3 案例圖形

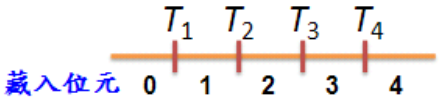


圖 4 門檻值與藏入位元

當找出圖中每個 x 點的藏入位元數後，接下來我們利用最低位元取代法嵌入資訊。由於灰階影像的像素值是介於 0 到 255，每個像素值是由 8 個位元所組成的。若可藏入的位元數為 2 則以 LSB 取代法取代最右邊的二個位元。Maniccam 和 Bourbakis 其藏入演算法如 Algorithm 1 所示。

Algorithm 1：

Inputs: Cover image $I = \{I_1, I_2, \ldots, I_N\}$
Secret data bit stream M
Output: Stego image I'
Let p = length(M)
For (i = 1 to N)
If p equals 0
Stop embedding
Else
k = minimum{ $F(I_i), p$}
Replace k LSBs of I_i
with k bits from M to generate I_i'
$p = p - k$
End If
Next

演算法中，F 為回傳每個藏入點可藏入位元數的函式。輸入值為 I , M，I 為隱蔽影像，I_i 為第 i 個像素，M 為機密訊息以二進制表示。接著對每個像素值利用 F 函式算出可藏入位元數，如果 $F(I_i)$ 等於 0，將不嵌入任何訊息，否則，M 依序的嵌入到訊息 I_i 中形成 I_i'。M 載入 I 的結果即為偽裝影像 I_i'。

當收方收到一張嵌入資訊的 I_i' 後。首先，必須要使用複雜度分析計算出每個點的複雜度，並且利用 F 函式找出每個點所對應出來的藏入位元數。之後把每個像素值轉換成二進制，根據每個像素值所對應的藏入位元數，依序取出。換言之，如果藏入位元素個數為 1，則取出最後一個位元數。將這些資訊串接起來，就可得到原先所藏入的機密資訊。而下方 Algorithm 2 演算法即為資訊取出過程。

Algorithm 2：

Inputs: Stego image $I' = \{I_1', I_2', \ldots, I_N'\}$
Output: Secret data M
For (i = 1 to N)
k = $F(I_i')$
Extract k LSBs of I_i' and append to M
Next

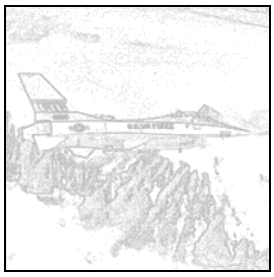
以 Lena，Pepper，Airplane 和 Baboon 這四張圖進行分析，如圖 5，可以發現 Baboon 這張圖黑色部分的點特別多也特別明顯，這代表著能藏入的資訊量也越多；反之，越接近白色點的地方則越平滑，能藏入的量越少，而完全是白點的部分表示不能藏入任何資訊。另外，Maniccam 和 Bourbakis 針對 Lena 圖做能藏入 1 到 4 位元個數做分析，如圖 6，發現能藏 4 的位元的部分個數比較少，也就是說對於這四張圖，我們可以很明顯的知道能一次藏入 4 個位元數的位置其實是很少的。

此外，Maniccam 和 Bourbakis 的方法是

會失真的藏入法，即原始影像會被破壞且無法還原，為了改進 Maniccam 和 Bourbakis 的方法，本篇論文提出一個能夠藏入大量資訊且又能同時回復原始影像的方法。


Lena


Pepper


Airplane


Baboon

圖 5 複雜度分析


1-bit


2-bit


3-bit


4-bit

圖 6 不同位元數嵌入的位置

3. 植基於複雜度分析之無失真資訊隱藏技術

所提方法與 Maniccam 和 Bourbakis 提出方法不同之處在於我們先將一張圖經過量化之後，才開始做複雜度分析，所提方法中，資訊藏入點，即 x 其鄰近的 8 個像素都必須要做嵌入的動作。也就是說，如果算出的複雜度為 1 的話，x 附近的 8 個像素複雜度也將取代成為 1，如圖 7，x 值附近的 8 個像素都將使用 LSB 的方式嵌入 1 個位元。此外，Maniccam 和 Bourbakis 的藏入點是以 2 為一個單位，例如，假設目前的藏入點為(2,2)則下次的藏入點為(2,4)，也就是說他下次要藏入的點為(2,4)這個點。而我們則是以 3 個點為單位嵌入，例如，(2,2)的下一個藏入點為(2,5)。

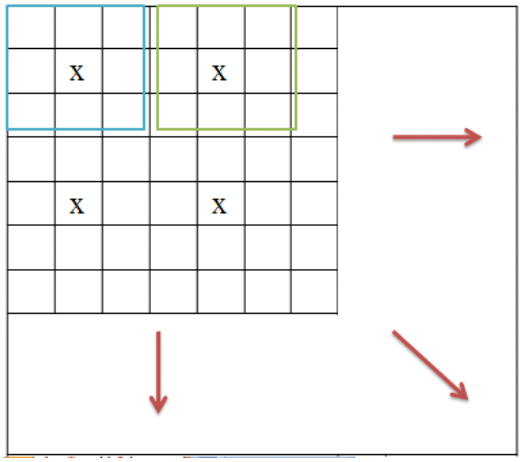


圖 7、藏入點位置

3.1 嵌入過程

主要的嵌入過程為：

1. 先將隱蔽影像做量化的動作，求量化後的影像 V，接著計算 V 中每個 x 值之複雜度，其複雜度算法如公式(1)所示，接著，根據所算出之複雜度決定 x 和附近的像素值可以嵌入的位元數。
2. 將隱蔽影像與量化後的影像做相減的動作，求得差異量表，r，再將 r 進行算數編碼的動作，得到二進制字串，稱為 W。
3. 設定檔頭大小並且固定為 16，所提的方法共需要嵌入 3 個資訊，第一個為檔頭，用來顯示 W 的大小，第二個為 W，第三個為機密訊息 M。
4. 將隱蔽影像依照複雜度依序嵌入，上面三個資訊即是將來要嵌入隱蔽影像中的資訊，其格式如表一所示，即可得到嵌入後的偽裝影像。

3.1.1

一開始我們先對隱蔽影像做量化的動作，為什麼要量化呢?因為量化是失真壓縮的實現方法，而量化的範圍我們設成 16，用來將隱蔽影像中每個像素值的最後 4 個位元清空。

因為一張灰階影像，每個像素是由 8 個位元表示的，通常越靠近左邊的位元越重要，越靠近右邊的位元越不重要。為了不造成太大的失真，我們最多改變右邊 4 個位元，所以將量化的大小設定為 16。一開始先將像素值除以 16，再與 16 做相乘的動作。例如，假設像素值為 165，量化後的值為 $160=\left\lfloor \frac{165}{16} \right\rfloor \times 16$。量化過後的圖稱為 V。

接下來算出 V 中每個 x 點的複雜度，如公式(1)，利用 x 值鄰近 8 個像素計算對應到所設的門檻值，看是屬於哪個範圍，來決定這個 x 值最多可以嵌入多少個位元，同時將其 x 值附近的 8 個像素也嵌入與 x 相同的位元數。

3.1.2

將經過量化後的圖 V 與原始隱蔽圖 I 做相減的動作，得到一個 r 表，這個 r 表就是原圖與經過量化後圖的差異表。我們將這個 r 表進行算數編碼的壓縮得到一二進制字串的資訊稱為 W。

3.1.3

接著，設定檔頭的大小，用來指示 W 的長度當收方接收資訊的時候，可以知道那些部份是嵌入壓縮過後的編碼表，那些部份是真正藏入的機密資訊。因此，檔頭大小的設定是必要的，下方表一即為要嵌入隱蔽影像的資訊 M'。

表一 嵌入資訊的格式(M')

檔頭	壓縮結果(W)	機密資訊

3.1.4

最後，將隱蔽影像配合之前所算出來的複雜度的大小，利用最低位元取代法依序嵌入，如果藏入位元數為 0，就不嵌入任何位元，反之，如果藏入位元數為 1，就將所對應的像素值轉換成二進制，然後嵌入到對應的像素值最右邊，直到要嵌入的資訊嵌入完畢，即可得到一張已嵌入資訊的偽裝影像 I'。

Algorithm 3：

Inputs: Cover image $I=\{I_1, I_2, \ldots, I_N\}$

Secret data bit stream M'

Output: Stego image I'

Let p = length(M')

For (i = 1 to N)

If p equals 0

Stop embedding

Else

k = minimum{ $F(I_i), p$}

Replace k LSBs of I_i

with next k bits from M' to generate I_i'

$p = p - k$

End If

Next

3.2 取出過程

當接收方接收到偽裝影像 I'，第一個步驟就是先對這張偽裝影像做量化的動作，然後計算出之前每個嵌入點 x 值的複雜度。如此可以知道先前每個 x 值及其附近的 8 個像素值各嵌入多少個位元，取出前 16 個位元，轉換成二進制，即為壓縮資訊 W 的長度，接著讀取 W 及機密訊息，同時將 W 的長度作算數解碼的動作還原出原先的 r 表，在與量化後的圖做相加的動作，還原出原始隱蔽影像。其步驟如下：

1.先將收到的偽裝影像 I' 做量化，並且計算出每個 x 值的複雜度，可以得到每個 x 值及其附近的 8 個像素值所嵌入的位元數。
2.取出前 16 個位元，並將它轉換成十進制，為 W 的長度。
3.將 W 做算數解碼，並與量化後的圖做相加的動作以還原原先的隱蔽影像，並取出機密資訊。

各步驟詳細說明如下：

3.2.1

將偽裝影像做量化的動作得到 V，在這裡量化的範圍也設成 16，與之前嵌入演算法設定的一樣。接著，計算每個 x 值的複雜度藏入位元數，計算出原先每個 x 值像素值的值，依據 x 的複雜度取出像素 LSB 藏有資訊的位元形成一個位元字串。

3.2.2

當我們算出每個像素的複雜度大小後，配合最低位元取代法的方法依序先取出前 16 個位元的資訊，接著，將這 16 個二進制的資訊轉換成十進制。例如轉換後的十進制為 48592，則後面的 48592 個位元即為先前壓縮過後的 r 表，即 W，而第 48608(16 + 48592)開始為真正嵌入的機密資訊 M。

3.2.3

接下來，將剛才取出來的壓縮資訊 W 做算數解碼的動作，將之還原成原先的 r 表，將 r 表與量化後的圖 V 做相加的動作，即可還原原始的隱蔽影像 I。取出演算法如 Algorithm 4 所示。

Algorithm 4：
Inputs: Stego image $I' = \{I'_1, I'_2, \ldots, I'_N\}$
Output: Secret data M, Cover Image I
For (i = 1 to N)
V = quantizes I
If n equals 0
Stop extracting
Else
$k = F(I'_i)$
Extract k LSBs of I'_i and append to M'
End If
Next
$\#\_W$ = first 16 bits of M'
W = Next $\#\_W$ of M'
$M = M' - \#\_W - W$
$I = V + AC^{-1}(W)$
$AC^{-1}(W)$表示算術解碼。

4. 實驗結果

我們以 Lena、Pepper、Airplane 和 Baboon 四張大小為 512×512 的影像做實驗。首先做量化的動作，接著算出各像素的複雜度，將複雜度加總起來，扣除檔頭大小和差異圖以算數編碼壓縮後的位元數，剩下的即為經算數編碼真正嵌入量。表二為不同影像的差異量表壓縮後的結果。

我們以亂數產生器產生機密資訊，並算出各自的 PSNR 值，表三為 Maniccam 和 Bourbakis 及所提方法的比較。從表中，可以發現所提出的方法能嵌入的資訊量明顯的比 Maniccam 和 Bourbakis 所提出的資訊量大很多，而且影像品質也不算太差，最低都大於 30。另外，門檻值設越低時，藏入量也會越來越高，當然我們也針對了 Maniccam 和 Bourbakis 所提出的三種門檻值的設定，分別做實驗。

第二個實驗，針對 Lena、Pepper、Airplane、Baboon 這四張影像分別算出各自的複雜度，如圖 6，此四張圖的複雜度為 0, 1, 2, 3, 4，分別對應灰階值 255(white), 180, 120, 60, 0(black)，黑色的部份代表著能嵌入的位元個數為 4，而白色的部份代表複雜度越小，也就是不能嵌入任何位元個數。圖 7 所使用的門檻值為{2, 4, 8, 16}，以 Lena 實驗圖而言，分別找出能嵌入 1 到 4 位元的位置，很明顯的發現對於 Lena 而言，能嵌入 1 個位元的數量為 0，而能嵌入 2 個位元的位置佔大多數，能嵌入 4 個位元的位置大部份集中在邊緣的部分。

表二 四張壓縮過後的位元

512×512	Lena	Pepper	Airplane	Baboon
差異圖壓縮結果(bits)	48592	48760	48680	49032

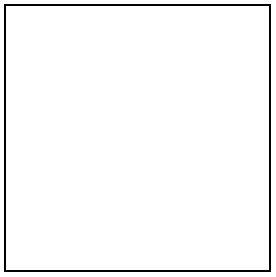

Lena


Pepper


Airplane


Baboon

圖 6 複雜度分析


1-bit


2-bit


3-bit


4-bit

圖 7 不同位元個數所嵌入的位置

表三 Maniccam 和 Bourbakis 提出方法的藏入量及所提方法之比較

(a)

檔名	Lena		Pepper	
	Maniccam 和 Bourbakis	Proposed	Maniccam 和 Bourbakis	Proposed
{2, 4, 8, 16}				
Bits	100368	378764	120208	367823
Bpp	0.3829	1.4449	0.4586	1.4031
PSNR	45.33	38.04	47.15	39.89
{1, 3, 6, 12}				
Bits	128896	394874	146848	381260
Bpp	0.4917	1.5063	0.5602	1.4544
PSNR	44.27	37.96	45.39	38.46
{1, 2, 4, 8}				
Bits	160720	526679	181656	517187
Bpp	0.6131	2.0091	0.6929	1.9729
PSNR	43.47	34.82	42.92	35.84

(b)

檔名	Airplane		Baboon	
	Maniccam 和 Bourbakis	Proposed	Maniccam 和 Bourbakis	Proposed
{2, 4, 8, 16}				
Bits	82560	330706	201464	744876
Bpp	0.3149	1.2615	0.7685	2.8415
PSNR	46.99	39.79	40.67	34.51
{1, 3, 6, 12}				
Bits	112064	344494	218680	781047
Bpp	0.4275	1.3141	0.8342	2.9795
PSNR	45.46	38.75	39.93	33.91
{1, 2, 4, 8}				
Bits	134920	446077	237392	878841
Bpp	0.5147	1.7016	0.9056	3.3525
PSNR	44.10	36.13	39.13	30.34

除此之外，我們將複雜度的門檻值設為{1, 2, 4, 8 }，並嵌入一張大小 220×220 的 Tiffany 機密資訊，如圖 8，到 Lena、Pepper、Airplane、Baboon 中，圖 9 為最後所得到的結果，左邊為原圖，而右邊為嵌入過後的影像，可以發現很難用肉眼的分辨這兩張影像的不同，因為嵌入後的 PSNR 值都大於 30，尚在容許範圍之內，所以我們可以說這個實驗的結果是令人滿意的。

圖 8 要嵌入的 secret image

(a)

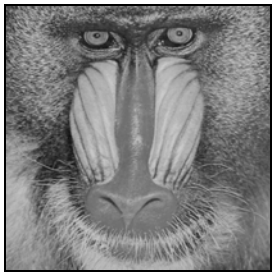
(b) PSNR = 37.22 dB

(c)

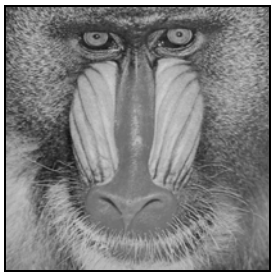
(d) PSNR = 37.24 dB

(e)

(f) PSNR = 34.83 dB



(g)



(h) PSNR = 35.57 dB

圖 9 原圖和嵌入後的影像

5. 結論與未來工作

根據實驗結果，可以得到二個結論，第一個就是提出的方法藏入量很明顯的改善了許多，PSNR 值也不會太差還在容許範圍之內，第二個就是收方不僅可以取出機密資訊，同時也可以利用算數解碼過後的差異表，還原原來的隱蔽影像。

對於未來，我們將試著用霍夫曼編碼去對差異表壓縮，看壓縮的效果跟算術編碼來做比較，那個壓縮效果比較好，以提升資訊的藏入量。另外，還可以對偽裝影像做加密及打散的動作，以增加影像嵌入的安全性。

參考文獻

[1] Maniccam S. S. and Bourbakis N., "Lossless Compression and Information Hiding in Images," ***Pattern Recognition***, Vol. 37, pp. 475-486, 2004.

[2] Cacciaguerra, S. and Ferretti S., "***Data Hiding: Steganography and Copyright Marking***," http://www.cs.unibo.it/~scacciag, Sep. 2002.

[3] Kim, Y. H., Kang H., Kim K., and Han S. S., "A Digital Audio Watermarking Using Two Masking Effects," ***Proceedings of the 3rd IEEE Pacific-Rim Conference on Multimedia 2002, LNCS 2532***, Springer-Verlag Berlin Heidelberg, pp. 655-662**,** 2002.

[4] Bender, W., Gruhl D., Morimoto N., and A. Lu, "Techniques for Data Hiding," ***IBM Systems Journal***, Vol. 35, No. 3-4, pp. 313-336, 1996.

[5] Chen, T. S., Chang C. C., and Hwang M. S., "A Virtual Image Cryptosystem Based upon Vector Quantization," ***IEEE Transactions on Image Processing***, Vol. 7, No. 10, pp.1485-1488, 1998.

[6] Mielikainen, J., "LSB Matching Revisited," ***IEEE Signal Processing Letters***, Vol. 13, No. 5, pp. 285-287, 2006.