開發應用程式以保護社群網站影像之浮水印機制

廖惠雯[1]　張士勳[2]　孫全澍[3]
[1,3]嶺東科技大學資訊科技應用所、[2]嶺東科技大學資訊網路系
[1]hwliao@teamail.ltu.edu.tw; [2]sschang@teamail.ltu.edu.tw; [3]s39676536@gmail.com

摘要

現今過於開放便利的社群網站，社群朋友日益複雜，使用者常常上傳生活中點點滴滴的相片或者是個人創作的圖片分享給社群朋友，而這些影像並無良好的保護機制，使網路上竊取及濫用的行為屢見不鮮。本文利用 Java 程式開發基於小波轉換之浮水印技術，應用於 Android 作業系統，使用者可以簡易將浮水印嵌入圖片後再上傳至社群網站，以保護智慧財產權；當浮水印需要驗證時，亦可簡易萃取浮水印辨識著作權。本研究目的在於使即時分享的影像建立安全的保護機制。

關鍵詞：社群網站、智慧財產權、浮水印。

Abstract

The overly open and convenient online social networks for friends to socialize over the internet are increasingly complicated. Users often share their photographs that show every detail of their lives or users' own creations are often shared with their social friends. These images do not have good protection schemes; therefore, the data thefts and misuses often occur. This paper utilizes Java program to develop a watermarking technique based on the wavelet transform for Android operating system. With this program, it's very easy for users to embed watermarking into pictures; then upload to social network sites to protect intellectual property rights. When verifying the watermarking, it can be extracted easily from images to reveal who has copyright. The purpose of this study is to establish the secure protection scheme for real-time shared images.

Keywords: Social Network Sites, Intellectual Property Rights, Watermarking.

1. 前言

社群網站目前已成為現代人與人交流不可或缺的一部份，從最早的無名小站、痞客邦，到現在的 Facebook、Twitter、Plurk，愈來愈便利的操作與簡潔漂亮的介面，讓使用者更樂於使用，然而，如此便利的分享也帶來了許多問題，例如：利用訊息流動快速的特性發表不實消息來操弄群體意識及不法人士盜用帳號對朋友進行詐騙和隨意下載個人創作圖片做不當的使用等，都透露出社群網站非常的不安全。本研究針對影像安全議題，思考現今攝影與繪圖創作豐富的時代，如何保障個人的創作不被濫用與盜用，提出社群網站浮水印機制，以解決社群網站影像安全問題。

2. 文獻探討

浮水印技術眾多，但將之應用於社群網站上卻是罕見，Liao 和 Hwang[1]於 2011 年提出多重浮水印機制，結合小波轉換[2]、直方圖修改[3]及小波樹[4]等技術，本文採用 Liao 和 Hwang 嵌入技術開發 Android 應用程式(Application, App)，提供即時嵌入浮水印及萃取浮水印，以保護圖像分享至社群網站的及時保護機制。

2.1 小波轉換(Wavelet Transform)

小波轉換的頻率域技術是最早由 Haar[2]所提出，是一種由空間域轉頻率域的方法，其方法為在二維影像上將影像分為高頻及低頻兩個頻帶，高頻區塊如影像邊界部份，此頻段像素變動不易察覺，低頻區塊為影像基礎組成，人類視覺系統對於此頻段區塊之影像異動較為敏感，小波轉換可將各頻段區隔開，利於各種影像嵌入或壓縮設計，如圖 1 為 3 階小波轉換後子頻帶示意圖。

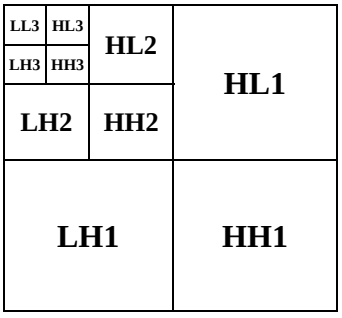


圖 1 三階小波轉換處理後之子頻帶

2.2 直方圖修改(Histogram Modification)

在進行小波轉換前，需透過直方圖調整修改媒體影像，避免浮水印嵌入小波係數與反小波轉換之

後有些像素可能產生溢位的情況。圖 2 為以像素(pixel)大小為 8 bits 的直方圖調整公式，其目的是避免嵌入資訊後轉換回空間域時，數值產生負數或大於 255 之情況，需將影像參數接近 0 或 255 之數值做調整，其中 X_i 為影像像素；G 為調整參數，本研究 G 設定為 30。

$$
\begin{aligned}
&If \quad X_i \geq 255 - G/2 \quad Then \\
&\quad X_i = X_i - G/2 \\
&Else \quad If \quad X_i \leq G/2 \quad Then \\
&\quad X_i = X_i + G/2 \\
&Else \\
&\quad X_i = X_i
\end{aligned}
$$

圖 2 直方圖修改公式

2.3 小波樹(Wavelet Tree)

小波樹概念由 Shapiro[4]所提出，Wang 和 Lin[5]提出一種用於保護版權浮水印的小波樹量化技術，在小波階層式架構，各波段尺寸大小依階層的不同而有所改變，以三階小波為例，第一階為第二階的四倍大，第二階與第三階也相差四倍；為使不同階層有空間區段上的對應關係，以第三階波段為基礎，該階層之每一係數皆可對應到第二階的四個係數，而第二階小波係數亦可對應第一階的四個係數，即可建立起小波樹結構，如圖 3 所示。

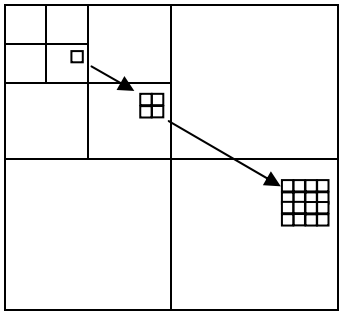


圖 3 三階小波樹對應圖

2.4 嵌入演算法

嵌入演算法[1]如圖 4 所示，首先，將浮水印中的每個座標取二元值 w，然後，一個二元值位置對應於經過二階小波轉換影像的五個座標係數值 d，之後，二元值分別與對應的五個係數值經過嵌入演算法公式處理後，會得到新的係數值 v'，其相對應位置如圖 5 所示，最後修改對應的五個座標的係數值，重覆步驟直到浮水印中的每個座標的二元值皆經處理，即完成浮水印的藏入，此演算法能夠藏入的浮水印長寬限制為原始影像的 1/4，色彩為黑白的浮水印。

$$
\begin{aligned}
&r = |d| \bmod \alpha \\
&\Delta(v) = d - \frac{d}{|d|} \cdot r \\
&If \quad w = 1 \quad Then \\
&\quad n = (\alpha - 1) \cdot \beta \\
&else \\
&\quad n = (\alpha - 1) \cdot (1 - \beta) \\
&v' = \Delta(v) + \frac{d}{|d|} \cdot n
\end{aligned}
$$

d：小波係數
w：浮水印二元值
α：模數
β：差距參數
v'：調整後之小波係數

圖 4 嵌入演算法公式

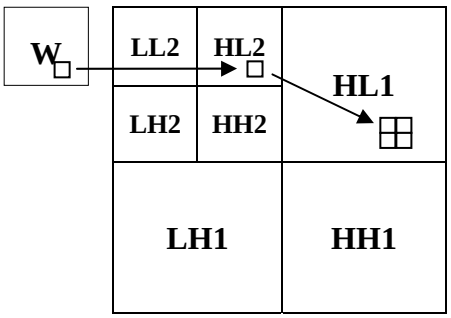


圖 5 浮水印 W 對應至 HL2 與 HL1 位置示意圖

2.5 萃取演算法

萃取演算法[1]如圖 6 所示，原始影像經二階小波轉換處理藏入浮水印相對應的五個座標係數值為 a、b、c、d、e，將係數取出後，經過萃取演算法判斷浮水印對應座標的二元值 $W_{i,j}$。

$$
\begin{aligned}
&a = |HL2_{i,j}| \bmod \alpha \\
&b = |HL1_{i*2-1,j*2-1}| \bmod \alpha \\
&c = |HL1_{i*2-1,j*2}| \bmod \alpha \\
&d = |HL1_{i*2-1,j*2}| \bmod \alpha \\
&e = |HL1_{i*2,j*2}| \bmod \alpha \\
&m = \frac{(a+b+c+d+e)}{5} \\
&If \quad m \geq (\alpha - 1) \cdot (1/2) \quad Then \\
&\quad W_{i,j} = 1 \\
&Else \\
&\quad W_{i,j} = 0
\end{aligned}
$$

圖 6 萃取演算法公式

2.6 影像品質鑑定

在影像品質的鑑定上，本文採用影像尖峰訊號對雜訊比值(Peak Signal to Noise Ratio, PSNR)，浮水印辨識採用竄改評估函數(Tamper Assessment Function, TAF)，PSNR 是衡量兩張影像的影像品質程度，一般來說，PSNR 值達 28dB 以上，影像已無法以視覺來區別，影像品質良好；TAF 用於二元形式的數位浮水印，採用兩張浮水印之間相異部份來做為衡量的標準，統計浮水印位元錯誤之機率，當 TAF 值愈接近 0，表示原始浮水印與取出浮水印的相似性越高，反之，則表示兩者的相似性愈低。

3. 影像浮水印 App 軟體分析與設計

因為本研究應用於行動裝置上，行動裝置的硬體資源不如桌上型電腦來得充裕，因此在 App 軟體的設計相對來講會有更多的細節要去注意，例如記憶體空間的使用、CPU 的負載、電源的消耗、網路的穩定性及作業系統本身的限制等；另外社群網站對於影像會有特別的處理方式，因此需要良好的設計萃取出的浮水印才能達到可辨識的程度。

3.1 社群網站會將上傳的影像做壓縮處理

一般社群網站會將上傳的影像做壓縮處理後儲存，以節省不必要的資源浪費，而 Flickr 是攝影界中較推崇的社群網站，但 Flickr 並非無壓縮處理，而是在壓縮處理的情況下盡可能的保有影像的細節資訊；而 Facebook 則是以節省資源為目的處理圖片壓縮，如圖 7，這些壓縮圖片的動作都會對浮水印造成破壞，導致浮水印辨識不易。

圖 7 原始圖(左)與傳上至 Facebook 圖片(右)比較

3.2 App 軟體架構

本文的 App 軟體架構依功能分為兩個部份，影像嵌入浮水印與影像取出浮水印，如圖 8 所示，在影像嵌入浮水印的功能中，需要選擇影像與要嵌入的版權浮水印，即可完成操作，操作處理完成的影像會存放於要處理的影像相同目錄下，檔名為“原始影像名稱”加上“_Embedding.png”，例如原始影像名稱為 Lina.png，處理完後的影像就為 Lina_Embedding.png，而影像萃取浮水印的部份，需要選擇影像即可取出浮水印，處理完成的浮水印會放置與要處理的影像相同目錄下，名為“要處理影像名稱”加上“_Extracting.png”。影像浮水印嵌入與萃取的處理流程如圖 9 及圖 10 所示。

3.3 App 軟體運行時記憶體不足的問題發生

記憶體不足(Out of Memory，OOM)問題的產生，是基於 Android 對系統資源的保護，App 使用超過限度的記憶體空間而導致，一般最少一個 App 可使用的記憶體空間為 16MB，會因每種設備性能不同，而對可使用的記憶體空間會有所調整，在本文實作中，為了避免過度使用記憶體空間，在使用 BitmapFactory 類別載入影像時，會將影像的長寬經過處理，長寬最大為 1024 像素，以避免 OOM 的產生。

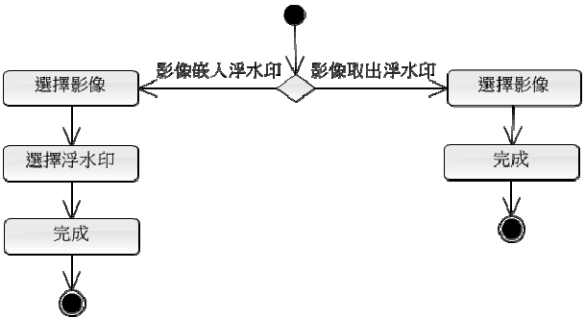


圖 8 App 軟體使用流程圖

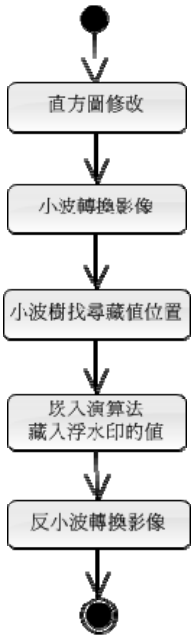


圖 9 影像嵌入浮水印演算法使用流程圖

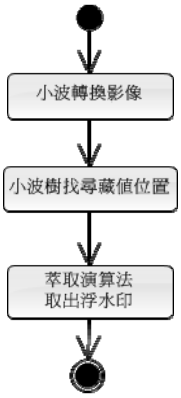


圖 10 影像取出浮水印演算法使用流程圖

4. 實驗結果

本研究開發的 App 軟體，軟體實際的操作畫面如圖 11、圖 12 及圖 13。圖 11 為浮水印 App 軟體的圖示(ICON)，圖 12 為軟體功能清單，主要功能為將圖片嵌入浮水印及取出浮水印，圖 13 為 App 軟體影像浮水印處理畫面。

圖 11 App 軟體 ICON

圖 12 App 軟體功能清單

圖 13 App 軟體影像浮水印處理畫面

本研究選擇 Facebook 社群網站進行實驗，主要原因是 Facebook 與 Twitter 和 Plurk 的影像壓縮破壞比較中，Plurk 的影像壓縮是破壞程度較大，從上傳過後的影像與原影像比較計算後，在未嵌入浮水印時，PSNR 值雖有 30.22dB，如表 1 所示，但在使用 α=50 與 β=5/6 之參數嵌入浮水印測試時發現 TAF 值高達 0.69，所以 Plurk 對於浮水印的破壞程度相當嚴重，造成浮水印辨識效果困難，如表 2 所示；而 Twitter 則完全保有影像的資訊，上傳後的影像與上傳前影像之 PSNR 並無差異，因此浮水印得以完整的保留，因此本研究實驗針對 Facebook 社群網站為實驗對象。。

表 1 各社群網站影像品質鑑定

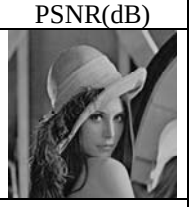
社群網站	浮水印
	PSNR(dB)
Facebook	
	50.35
Twitter	
	與原影像相同
Plurk	
	30.22

表 2 各社群網站對於浮水印之破壞測試
(α=50、β=5/6)

社群網站	浮水印
	TAF
Facebook	
	0.13
Twitter	
	0
Plurk	
	0.69

在實驗過程中，影響浮水印可辨識性的因素在

於嵌入演算法公式所用到的模數α與距離參數β的設定，所以測試不同的模數α與距離參數β，再將影像上傳至 Facebook，實驗步驟流程如圖 14；實驗所使用的影像與浮水印如圖 15。

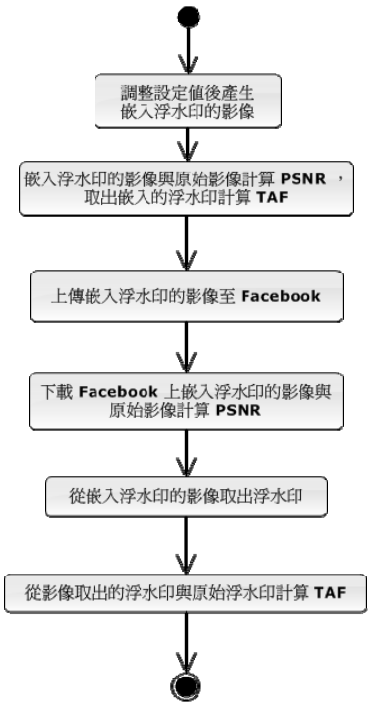


圖 14 實驗步驟流程

圖 15 原始影像(左)與嵌入的浮水印(右)

首先，固定模數α=20，調整不同距離參數β，將浮水印嵌入原始影像，然後將嵌入浮水印影像上傳至 Facebook 後下載，計算嵌入浮水印之 PSNR 及上傳 Facebook 後影像之 PSNR，由表 3 發現：上傳前之 PSNR 在 44dB 左右，上傳後之 PSNR 在 30dB 左右。由表 3 可得知：當α值固定，β值愈高，PSNR 值愈低。表 4 為固定模數α=20，調整不同距離參數β，上傳 Facebook 前及上傳後萃取出浮水印之 TAF 值，由實驗發現，未經上傳 Facebook 之 TAF 值均接近 0，辨識度良好，經上傳 Facebook 後，辨識度下降，當α值固定，β值愈高，TAF 值有降下降趨勢。

表 3 調整距離參數β的影像比較(α=20)

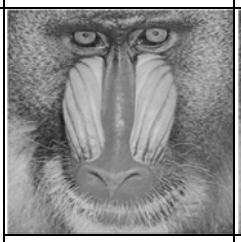
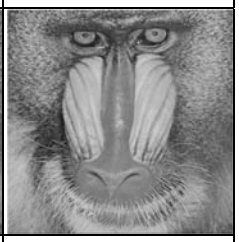
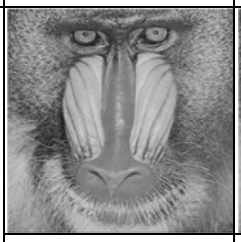
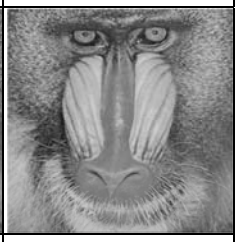
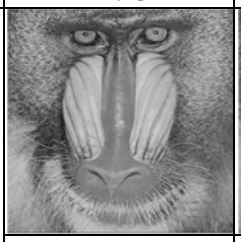
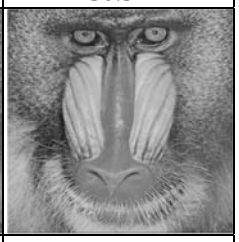
β	PSNR(dB)	
	上傳前	上傳後
3/4		
	44.96	30.97
5/6		
	44.29	30.97
7/8		
	44.02	30.97

表 4 調整距離參數β的浮水印比較(α=20)

β	TAF	
	上傳前	上傳後
3/4	***aloha~~ very happy~~***	***aloha~~ very happy~~***
	0.00	0.62
5/6	***aloha~~ very happy~~***	***aloha~~ very happy~~***
	0.00	0.59
7/8	***aloha~~ very happy~~***	***aloha~~ very happy~~***
	0.00	0.59

表 5 為固定距離參數β=5/6，調整模數α，α值為設為 10、20、30、40 及 50，比較不同α值時，上傳前及上傳後之 PSNR 之變化，由表 5 發現：當β值固定，α值愈高，PSNR 值愈低，上傳 Facebook 後，PSNR 值下降為 30dB 左右。表 6 為固定距離參數β=5/6 及調整不同模數β，上傳前及上傳後萃取

出浮水印之 TAF 值，由實驗發現，未經上傳 Facebook 之 TAF 值均接近 0，經上傳 Facebook 後，當β值固定，值α愈高，TAF 值愈低。

表 5 調整模數α影像比較(β=5/6)

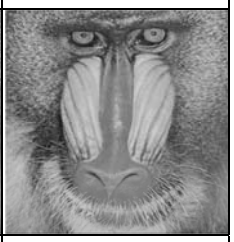
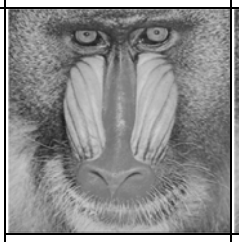
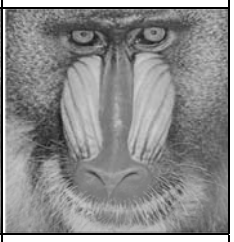
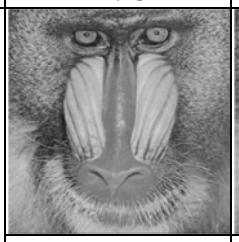
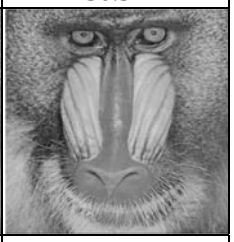
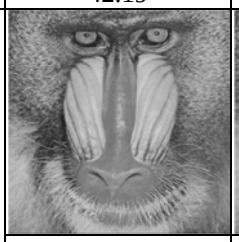
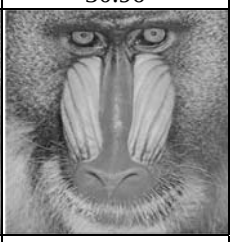
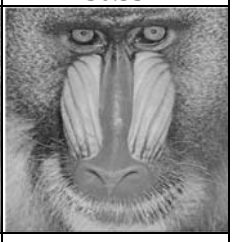
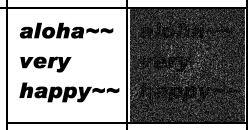
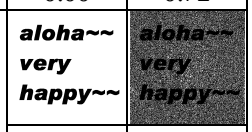
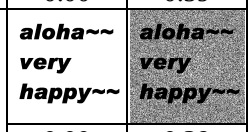
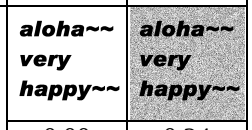
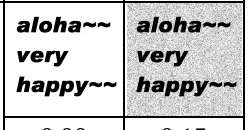
α	PSNR(dB)	
	上傳前	上傳後
10		
	47.21	30.98
20		
	44.29	30.97
30		
	42.19	30.96
40		
	40.60	30.95
50		
	39.47	30.93

表 6 調整模數α浮水印比較(β=5/6)

α	TAF	
	上傳前	上傳後
10		
	0.00	0.72
20		
	0.00	0.59
30		
	0.00	0.39
40		
	0.00	0.24
50		
	0.00	0.15

5. 結論

本研究提出了分享於社群網站的影像之保護機制，開發 App 應用程式，保護日益盛行的社群網站生活。實驗發現：為了保護浮水印影像因上傳至社群網被破壞而造成不可辨識的程度，建議將嵌入演算法之距離參數β值設定在 5/6，模數α值建議設定在 50 以上，可得較佳的浮水印辨識度。

未來研究將全面考量保護彩色影像之社群網站浮水印機制，還有類似 Plurk 嚴重破壞影像的壓縮技術該如何應對，均是進一步研究的目標。

參考文獻

[1] H.W. Liao and H.W. Huang, “A Multiple Watermarking Scheme for Gray-Level Imagines Using Visual Cryptography and Integer Wavelet Transform,” Journal of Computers, Vol.22, No.1,pp.18 36,Apr. 2011.

[2] A. Haar, “Zur Theorie der Orthogonalen Funkionen Systeme, (Erste Mitteilung),” Mathematische Annalen, Vol. 69, pp. 331-371, 1910.

[3] E. Khan, M. Ghanbari, “Wavelet-based video coding with early predicted zerotrees,” Image Processing, IET, Vol 1, pp. 95-102, 2007.

[4] J. M. Shapiro, “Embedded Image Coding using Zerotrees of Wavelet Coefficients,” IEEE Transactions on Signal Processing, Vol. 41, No. 12, pp. 3445-3462, 1993.

[5] S.H. Wang and Y.P. Lin, “Wavelet Tree Quantization for Copyright Protection Watermarking,” IEEE Transactions on Image Processing, Vol. 13, No. 2, pp. 154-165, 2004.