國立臺灣師範大學
資訊工程研究所碩士論文

指導教授：　紀博文　博士

使用兩階式降低差值及調整參考像素值的灰階圖像隱寫術

An Steganographic Approach Using Two Reducing Phases and Adaptive Reference Pixel in Grayscale Images

研究生：　　方友群　　撰

中華民國　一零九　年　二　月

摘要

在網路方便又快速的時代，許多人可自由上傳、下載想要的多媒體檔案，但同時造成數位盜版等問題。為解決這些問題，有許多學者提出了資訊安全的方法來保護原作者的檔案，藉此保障他們的權利。幾年前，有幾位學著更提出了隱寫術(Steganography)等資料隱藏演算法，嵌入數位簽章或其他檔案，以保護作者的所有權。另外同時兼顧輸出後檔案的品質以及將嵌入能力提升，是在設計一個隱寫術演算法時所要面臨的一個極大挑戰。

差值擴張(Difference Expansion)是在隱寫術界中應用較廣泛演算法之一，其方法是透過圖像像素之間的差來儲存隱藏資料，以達到隱寫的效果。本文提出了一個灰階圖像隱寫術演算法，它使用了改良過的差值擴張演算法，並從方法內部做一些優化改良。此演算法可以進一步地縮小圖片破損並使得輸出後圖片的像素更接近於原始像素，進而達到提高圖片品質的效果。研究結果證明我們修改的方法確實可以再降低像素的變動幅度，能夠比前人方法輸出更高品質的圖片。

關鍵詞：圖像隱寫術、資料隱藏、差值擴張、降低差值擴張

致謝

首先我要感謝在研究撰寫論文的期間中一路指導我、輔導我的指導老師－紀博文老師。從碩一下正式進入老師的實驗室起，我對碩士論文的研究概念還算矇矇糊糊，是這位老師主動開導我、校正我、教會我一些碩士研究所隱含的意義，讓我可以漸漸地進入情況，逐步地將我的研究給完成。雖然在我所研究的領域上，仍然跟老師他的拿手領域有些出入，但在我撰寫論文研究中提供了我一些建議，讓我有了想法將我的論文成果達到最好。在寫作文法、簡報撰寫上他也給了我一些糾正修改建議，讓我可以盡可能地將研究內容描述可以讓人看得懂，這使得原本中文文法本身就不太好的我得到了一些救濟。在這兩年的漫長歲月，紀博文老師帶領我、幫助我解決了許多我在研究生涯上所遇到的種種問題，也督促我使我成為了一個好的研究生。老師，辛苦您了！

最後我要感謝鼓勵我完成論文的家人們，適時給了我有關製作論文的意見。有了他們的支持、鼓勵與督促，我才能建立起自己的信心，並順利地靠我自己的力量完成我的研究，一路順利地取得碩士學位。

方友群 謹誌

國立臺灣師範大學 資訊工程研究所

目錄

附表目錄

附圖目錄

第一章　緒論

第一節　研究背景

拜高科技所賜，近代的網路發展日漸快速。有許多人藉由高速網路將自己喜愛的多媒體檔案上傳至雲端服務並加以共享。但在現代網路普及的同時，也因為有人的貪圖方便，藉由網路擅自分享他人的創作物，衍生出數位盜版、抄襲等一些不合法的舉動出現，導致創作者的數位版權無意間地被侵犯，造成創作者的權益損失。因此，數位版權管理是一個相當重要的議題。

為解決這些問題，學者使用資料隱藏(Information Hiding)技術來保護資料。資料隱藏技術是在幾十年前由 D. Parnas 所提出的概念 [1]，將訊息或數位簽章嵌入在創作物上，其外表用肉眼看起來跟原本的創作物沒什麼不同，但再次使用演算法反過來提取資料，以聲明此創作物的所有權，這能有效防止第三方非法竊取、擅自分享原作等問題，藉以保障創作者的權益。

第二節　隱寫術概觀

如圖 1 [2]所示，資料隱藏技術之下被分為幾個分支技術，如：浮水印(Watermarking)、隱寫術(Steganography)、匿名(Anonymity)等，隱寫術即是其中之一。隱寫術的概念是由「眼見為憑(What You See Is What You Get)」 [3]這個觀念引申而來。因為有的時候只憑靠事情的表面也不能完全知道真正的實情，而隱寫術即是利用這一點，使用「以假亂真」的手法蒙騙了常人的感官，並藉此達到隱藏資料的效果。

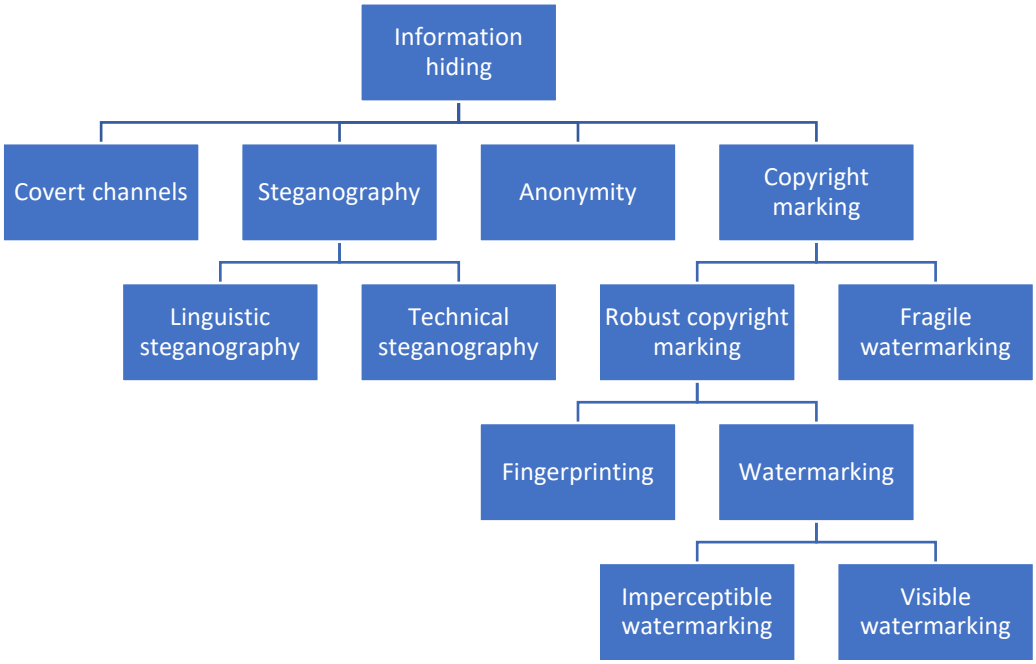


圖 1 資料隱藏技術及其衍生分支技術

隱寫術這一詞源自於兩個希臘詞：「封面(stegos)」以及「寫作(grafia)」 [3]，所以它也可以另稱為「封面寫入演算法」。就如同名稱上的意思一樣，隱寫術即是將要隱藏的訊息寫入在某一個物體表面上，使訊息與之完全融入。加入訊息後該物體表面上用人類的感官不易辨認出隱藏訊息的存在，但只要拿出補助工具或是解碼線索，就可以從該物體表面得知當初寫入的隱藏訊息 [4]。隱藏訊息在隱寫術領域上非常的重要，所以隱寫術必定要做到取出來的隱藏訊息要跟原本的訊息完全一模一樣 [4]。

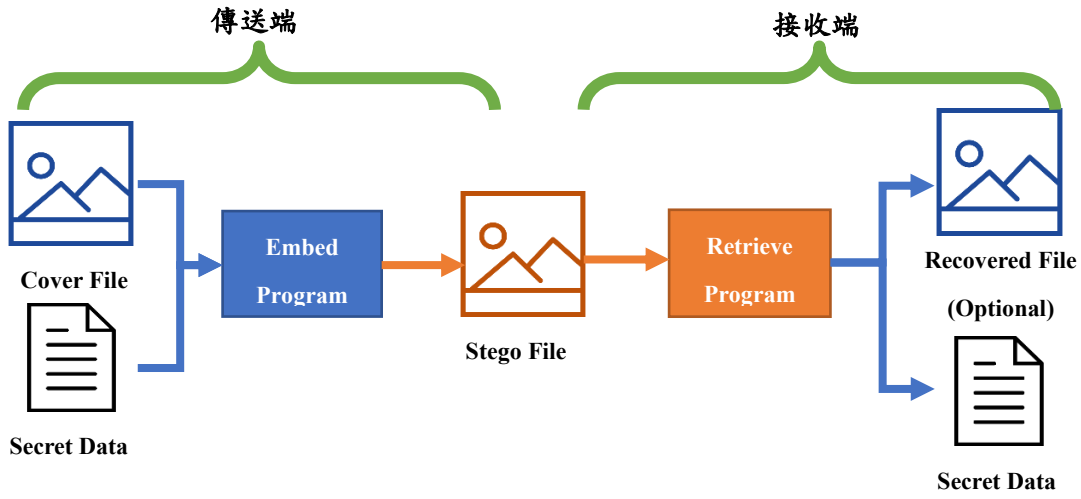


圖 2 隱寫術資料嵌入及解嵌入步驟

隱寫術通常做法如圖 2 所示 [3]，傳送端將隱藏資料(Secret Data)及準備好的多媒體載體(Cover File)，透過嵌入演算法將資料嵌入載體，使其變成新的檔案(Stego File)。接收端要取回原隱藏資料時再使用解嵌入演算法取回即可。

可逆式隱寫術(Reversible Steganography)是一個比較特殊的隱寫術。在可逆式隱寫術的場合下，接收端使用解嵌入演算法取回資料的同時也可以恢復原本嵌入前的檔案載體(Recovered File)。這類技術有著讓載體完全無損(lossless)地還原的特性，非常適合用在醫療用圖片上，可確保醫療上的正確性，免得造成醫生的誤判而導致不可彌補的後果 [5] [6]。

對於隱寫術演算法的設計，學者們歸納了一套隱寫術的設計特性規則，我們會在下一節去做說明。

第三節　隱寫術特性

隱寫術有三大特性 [4] [7]：

1. 隱蔽性(Imperceptibility)：嵌入隱藏資料之後，載體本質難免會有一些失真。隱寫術最主要的特性即是盡量讓輸出載體的失真程度降低，做到在外觀上「幾乎」看不出甚麼破綻，以防被第三方察覺。
2. 穩健性(Robustness)：檔案在傳輸的過程中，有可能會造成接收到的檔案有缺失不完整或被惡意竄改的情形。穩健性就是在於在檔案被破壞的情況下也能把資料提取出來，這有助於降低惡意攻擊的影響。
3. 容量(Capacity)：在現代隱寫術，載體高容量性是必須考慮的要素。因為演算法主要注重於內部的隱藏資料，載體必須有辦法騰出如一整個要隱藏資料的大小甚至更大的空間來寫入資料。

但是對於任何隱寫術(或是跟任何資料隱藏相關的技術)而言，這三種特性是無法同時並存的。

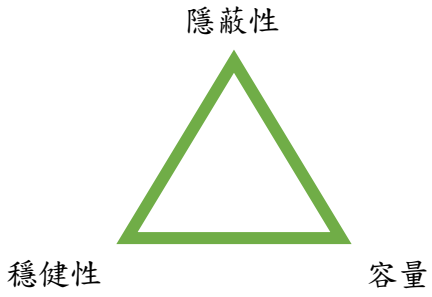


圖 3 隱寫術三大特性矛盾關係示意圖

如圖 3 所示，隱寫術的三大特性實際上是處於互相矛盾的狀態 [8]。若是要讓其中一個特性效益最大化時，必定會干擾到其餘的兩個特性。同時將其中兩個特性增強時，很有可能會導致剩餘的一個特性無效化。譬如在提升載體品質的同時，很有可能導致載體可塞的資料量變少。或者若將穩健性提升，相對的可能要浪費更多載體的空間或要素拿去設定並加強資料的穩健度等。在現代要製作有效率的隱寫術，必須同時兼顧載體隱蔽性及容量是設計隱寫術演算法的一大課題。所以在本論文的隱寫術將只針對其中兩大特性：隱蔽性、容量去做設計。

第四節　研究目標及計劃

隱寫術應時代變遷而推陳出新，在現代的隱寫術領域中流行使用多個既有的隱寫術或演算法進行組合，為的就是追求良好的嵌入品質以及特性。本論文主要以研究圖像隱寫術為主，是以圖像裡的像素值做為載體來嵌入資料。

我們使用了由 J. Tian 提出的差值擴張演算法(Difference Expansion，簡稱 DE) [9]以及從其衍生出來的改良演算法如：降低差值擴張演算法(Reduced Difference Expansion，簡稱 RDE) [10]、DE of quads [11]以及其它衍生的隱寫術文獻 [12] [13] [14]等作為研究對象。並從 DE 演算法領域中挑出其中一篇隱寫術論文 [14]，再從其演算法進行改良。在隱寫術的三大特性中，我們以隱蔽性及容量作為改善對象，以盡量提升圖片載體品質為主要目標。最後的實驗部分將我們的方法以及前人的方法 [14]做測試，並在品質與容量上做比較。

第五節　章節說明

在接下來的幾個章節，我們會在第二章回顧一下前人所提出的 DE 演算法及其相關的隱寫術演算法，第三章講述我們的方法，第四章將會列出實驗比較結果，並以第五章總結。

第二章　文獻回顧

在這一章下面會一一介紹前人所提過的隱寫術文獻。以下介紹的前人方法全部都是以圖像像素作為載體的可逆式隱寫術，並跟 DE 演算法相互關聯，且有幾個方法之間有被引用的關聯性。在此章的最後一節，我們會再度探討前人方法的優缺點，以及解釋我們想改良前人演算法的原因。

第一節　差值擴張相關文獻

據演算法做法，隱寫術可分為空間域(spatial domain)及頻域(frequency domain)兩大做法 [15]。其空間域是直接在圖片本身的「長度」要素上(如：像素等)做一些調整，以達到藏匿資料的目的 [16]。而頻域則可視需要利用圖片上的像素作為函數，並使用傅立葉變換(Fourier transform)、小波轉換(Wavelet transform)等頻率轉換運算式，將頻率可視化至像素上，並再用空間域作法藏匿資料即可 [16]。而從第二章第二節介紹的差值擴張(DE)即是空間域的做法。

在比較上，空間域的做法比較直接、容易理解且運作較快，而頻域的演算法雖然穩健性強，但演算法較複雜、需要花額外的運作時間在頻域的轉換上 [16]。以下我們舉了以 DE 演算法做為參考的前人文獻作舉例：

1. Firmansyah 等人提出了使用從圖片的 LSB 上執行 DE 隱藏資料的空間域隱寫術 [17]。其作法是將圖片上所有像素的 LSB 另外拼湊成一張矩陣，並在該矩陣上執行 DE 藏匿資料，其後把新算好的矩陣算回至圖片上，以達到隱寫的效果。雖然不適合儲存較大的檔案，但步驟及演算法非常明瞭。
2. Angreni 等人提出了使用共同數值的 DE 改良空間域隱寫術 [18]。其作法是提出一個隨機的共同數值，使所有圖片中的像素值跟該共同數值之間求出差值，並用這些差值套用 DE 演算法的嵌入資料方式。這個方法聲稱可做到將資料完全塞滿至整張圖片，同時也可增加圖片品質，但圖片品質、容量的多寡仍必須視使用者訂出的共同數值而定。

3. 葉氏提出了搭配小波轉換的 DE 改良頻域隱寫術 [19]。其作法是先將圖片作小波轉換，並為每個像素作類似差值擴張的隱寫方法。雖然隱蔽性提高，但因為該文獻的差值擴張算法稍微跟原 DE 演算法不同，且嵌入完成後使用反小波轉換時可能會有像素值溢位的情形，必須額外空出圖片空間去紀錄溢位還原資訊，工程相對比較複雜，演算法較不穩定。
4. Gao 等人提出了搭配離散餘弦變換(Discrete cosine transform)的 DE 改良頻域浮水印演算法 [20]。其作法是先在圖片上各像素組上做離散餘弦變換，再用 DE 的方式在像素組其中兩個像素嵌入資料。雖然增強了隱蔽性，也可保有高品質的圖片，不過因為僅取少量的像素來藏匿資料，所以不支援高容量，同時工程也比較複雜，容量甚至取決於使用者設定的門檻值而定。

而在第二章第三節要介紹的降低差值擴張(RDE)也是屬於空間域隱寫術，所以演算法也是相對比較容易理解、速度也比較快的那一型，同時也具備了比 DE 還要高的圖片品質、容量等優點。另外 RDE 演算法的構成是以步驟行的方法所組成，其相鄰步驟之間的互相影響力並不會太重，改良演算法時可以直接修改其中的步驟或添加穿插新步驟即可。這也是我們採用了以 RDE 演算法為基底的隱寫術作為研究改善對象的原因。

第二節　差值擴張 (Difference Expansion)

差值擴張(DE)是由 J. Tian 在 2003 年提出的一項灰階圖像隱寫術演算法 [9]，它是基於圖像中的像素差分(pixel value differencing，簡稱 PVD) [3]的方法，在一張圖片的兩個像素的差值中修改、嵌入要隱藏的訊息。它也是一個可逆式隱寫術，所以取出原本的隱藏資料的同時也可以恢復原本的圖像。DE 演算法的好處在於可以放寬容量限制以及降低演算法複雜度。

就因為具有低複雜、高容量性等優點，DE 演算法在隱寫術界上引用次數甚廣，可以算是隱寫術的主流之一。DE 演算法的大略運作示意圖如圖 4。

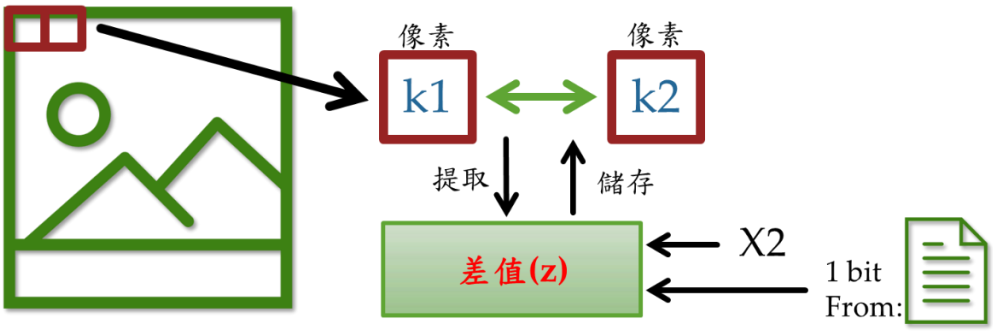


圖 4 DE 演算法運作示意圖

下面介紹 DE 演算法的主要作法：首先在圖片載體上分割成若干個不相重疊的像素組，其每個像素組皆由相鄰的兩個像素構成，如圖 5。

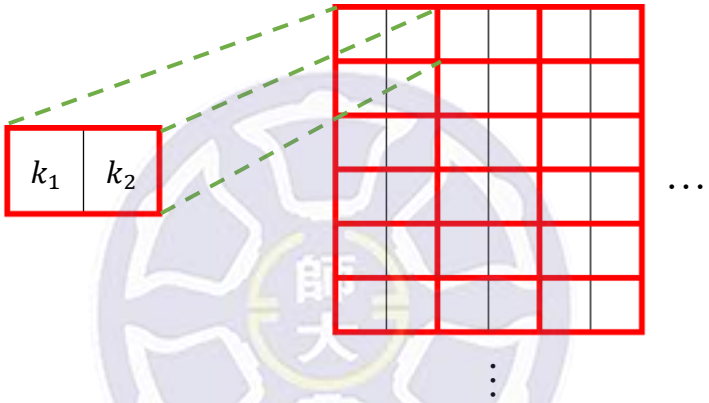


圖 5 像素組在圖片分割示意圖

在像素組中索取兩個像素的數值k_1 & k_2，並從中計算兩個像素之間的平均值 w 跟差值 z：

$$w = \left\lfloor \frac{k_1 + k_2}{2} \right\rfloor, \qquad z = k_1 - k_2 \tag{2.1}$$

其中平均值 w 是作為在差值 z 儲存好資料後回復成像素的必備數據。算出來的差值 z 之後要擴張至 2 倍，並將隱藏資料的其中一個位元($b = [0,1]$)嵌入，規劃出新的差值z'：

$$z' = 2 \times z + b \tag{2.2}$$

再計算新的像素值k'_1 & k'_2來取代原本的兩個像素值：

$$k'_1 = w + \left\lfloor \frac{z' + 1}{2} \right\rfloor, \qquad k'_2 = w - \left\lfloor \frac{z'}{2} \right\rfloor \tag{2.3}$$

但要特別注意的是，新的像素值不能超過色階數值的範圍(灰階色階數值定在0~255 之間)，所以在儲存像素值之前，會先使用以下方程式檢查新的像素值是否會溢位：

$$0 \le w + \left\lfloor \frac{z' + 1}{2} \right\rfloor \le 255, \qquad 0 \le w - \left\lfloor \frac{z'}{2} \right\rfloor \le 255 \tag{2.4}$$

透過(2.4)這方程式，我們可以再簡寫成(2.5)，將新的差值z'及平均 w 直接套用，檢查(2.2)所製造的新差值z'是否會造成新像素值溢位，若通過了即可用(2.3)寫回像素內：

$$|z'| \le \min(2(255 - w), 2w + 1) \tag{2.5}$$

若要將隱藏資料提取出來時，可以再使用(2.1)重新提取兩個像素的平均值w'及差值z'，從差值取出最低有效位(least significant bit，簡稱 LSB)即可取得隱藏資料 b，之後再將差值z'縮減成一半，就能回復成原來的差值 z，並使用(2.3)恢復原圖的像素值：

$$b = \mathrm{LSB}(z'), \qquad z = \left\lfloor \frac{z'}{2} \right\rfloor \tag{2.6}$$

以上的擴張並嵌入差值方式我們簡稱為「可擴張(Expandable)」，新的差值z'若符合(2.5)的條件時，可以稱z'或該像素組為 expandable。

不過，expandable 方法在多數的情況都會造成差值變大導致溢位，間接導致圖片載體空間不足，所以在 DE 演算法內加入了「多重嵌入」機制，以預防因 expandable 的缺點造成空間不足的缺陷。從 DE 演算法開始，就採用了一個像素組可以採用不同模式來嵌入隱藏資料的機制，有一些引用此演算法的隱寫術文獻，也有將這個機制再度引用或做一些改良。在 DE 演算法中，一個像素組有三種可套用的嵌入模式：

1. 可擴張(Expandable)：使用 DE 演算法的擴展方程式(2.2)擴張差值並嵌入資料的方式，即以上我們講過的 DE 嵌入方法跟解嵌入方法。
2. 可變動(Changeable)：同樣基於 PVD 的方法，在差值上的 LSB 上直接儲存資料。

3. 不可變動(Non-changeable)：不儲存資料、保持像素組原封不動。

其中 changeable 的嵌入跟解嵌入方法跟 expandable 類似，只不過在嵌入資料之前會先將原差值 z 的 LSB 部分提取並儲存一個額外的位元(bs)，以便在恢復原圖像時可以回復原圖的差值：

$$bs = \text{LSB}(z) \text{ where } bs = [0,1] \tag{2.7}$$

在嵌入方法的(2.2)換成此方程式，直接將差值 z 的 LSB 替換成隱藏資料 b：

$$z' = 2 \times \left\lfloor \frac{z}{2} \right\rfloor + b \tag{2.8}$$

在 changeable 同樣也要注意寫回的像素不可以超越色階數值範圍，只要經過(2.5)方程式的檢查通過，新差值z'或該像素組就可以被稱為 changeable。

在解嵌入演算法中，回復差值的(2.6)將會以下列方程式代替，用原本用(2.7)提取的 bs 替換回原本的差值：

$$b = \text{LSB}(z'), \qquad z = 2 \times \left\lfloor \frac{z'}{2} \right\rfloor + bs \tag{2.9}$$

從這裡再度回到多重嵌入機制的話題：在 DE 演算法設計多重嵌入機制的功能主要是為了若 expandable 嵌入資料發生數值溢位等問題時，還有其他的方法可以走，changeable 同時也避免了差值過大而導致圖片過度失真的問題，進而提升圖片載體品質，也有助於增加圖片載體容量。

另外在多重嵌入機制下，儲存各個像素組的嵌入模式是使用位置地圖(Location map)的方式，在每個位置地圖單位資料(嵌入模式)上都各自對應所屬的像素組。由 DE 演算法自行判斷該像素組適合使用哪一種嵌入模式，同時採用的模式將以代號的方式記錄在位置地圖上，以供使用解嵌入演算法時，即可呼叫這些位置地圖的代號記錄來使用對應的解嵌入方程式，補助取回隱藏資料及還原圖像。此外，使用 changeable 模式的(2.7)所紀錄的額外資料 bs 也可以使用位置地圖的方式儲存，在各自對應的 changeable 像素組使用對應的 bs 值來還原原本的差值。

透過以上對 expandable、changeable 跟多重嵌入機制的說明，我們可以將整

個嵌入模式判斷系統簡化成如圖 6 描述的步驟結構。

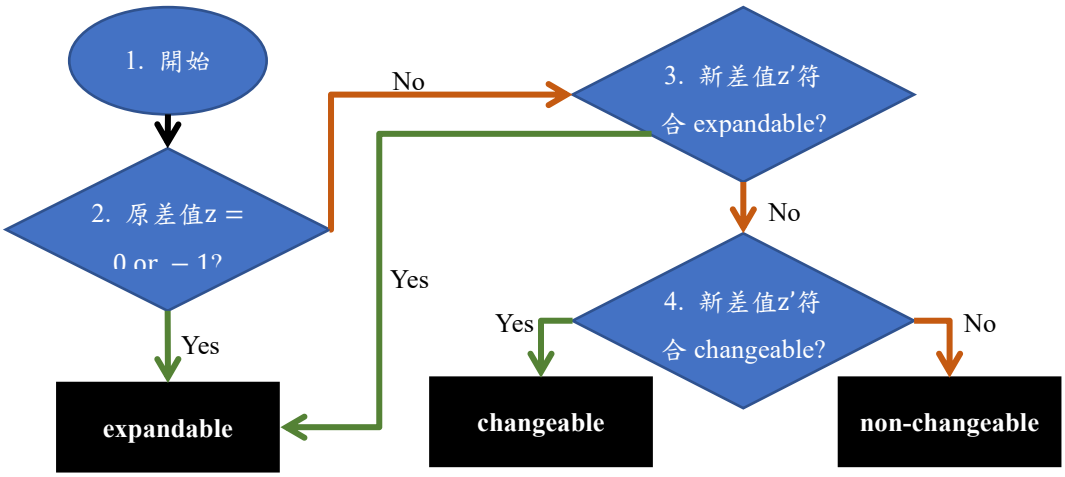


圖 6 DE 簡易多重嵌入機制判斷架構

其中步驟 3 及步驟 4 即是利用(2.5)檢查在 expandable 及 changeable 方程式下是否不會導致溢位。步驟 2 是因為在原差值為 0 或-1 的情況下，expandable 及 changeable 都會求到相同的新差值，為降低演算法複雜度，遇到這個情況一律把該像素組視為 expandable。

總結以上 DE 演算法的差值嵌入計算過程，我們將它畫成簡易的有限順序架構圖：

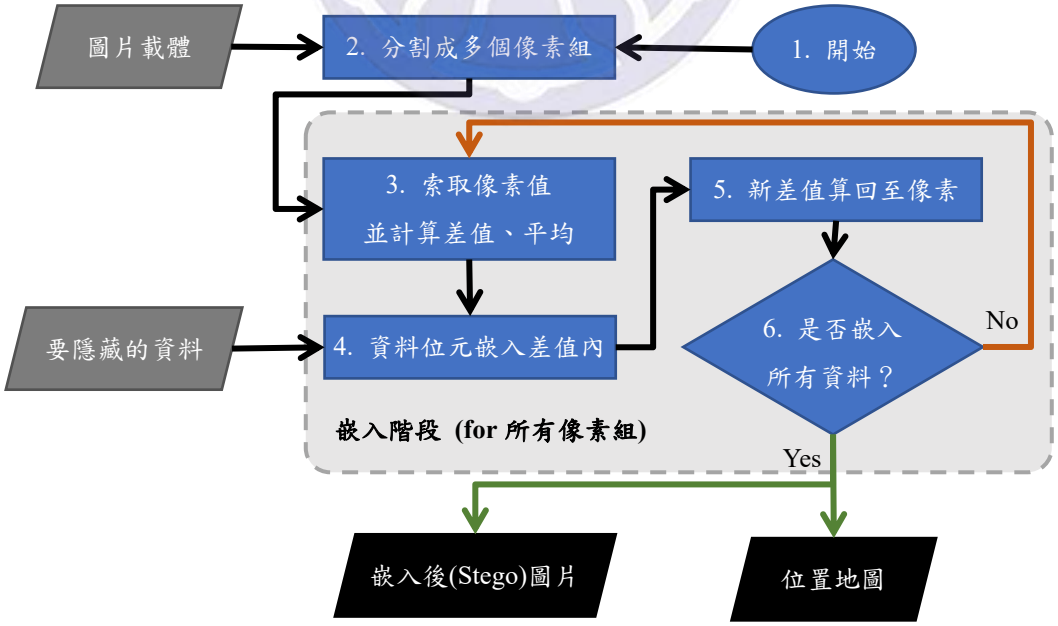


圖 7 DE 隱寫演算法簡易架構(嵌入側)

圖 7 是 DE 隱寫術中的嵌入步驟圖，將圖片分成數個像素組後，每個像素組

都將做一次嵌入差值的一系列步驟，直到所有資料被寫進去為止。至於在 DE 演算法所介紹的多重嵌入判斷機制，因為其中的溢位判斷機制(2.5)是透過 expandable 及 changeable 的嵌入方程式(2.2)/(2.8)衍生而來，此判斷機制可以在步驟 3 以後就判斷差值是否使用哪一個機制，或者是跟步驟 4 一起配合使用(在原 DE 隱寫術文獻是直接從原差值做判斷)。另外步驟 4 會根據該像素組套用了哪種嵌入模式(non-changeable 除外)來使用不同的嵌入方程式來寫入隱藏資料。

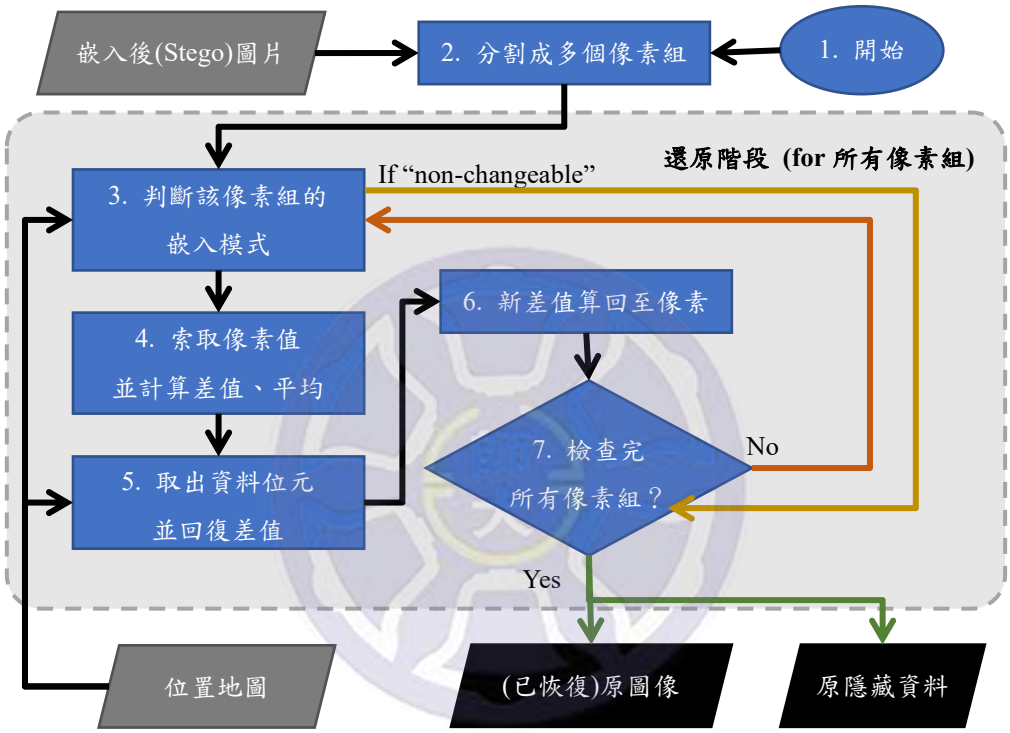


圖 8 DE 隱寫演算法簡易架構(解嵌入側)

圖 8 是 DE 隱寫術中的解嵌入步驟圖，跟圖 7 的嵌入側不一樣的是，除分割及輸出圖像及資料外，像素組解嵌入側動作是跟嵌入側採用相反順序的：嵌入側是將資料寫入、改變差值，而解嵌入側是將資料提出、還原差值。步驟 3 的地方即是將像素組解嵌入前，就得透過位置地圖判斷其嵌入模式了。另外，若該像素組是 non-changeable 的話，該像素組會直接略過，直接檢測下一組像素組，因為基本上該像素組沒有做甚麼變動，也沒有塞入隱藏資料。

DE 演算法同時也採用了「多層嵌入」的方式，用來應付塞入更多隱藏資料。對於已經嵌入資料完的圖片，或者是已檢查所有像素組並全部嵌入完成的圖

片，可以再針對這張圖片作為載體繼續塞資料做為第二層，塞完後可再拿此圖片繼續塞資料做為第三層，以此類推。但是原則上已嵌入的圖片再拿來做載體時，因為該圖片像素之間的差值曾被擴張過，所剩空間可能會隨著嵌入的層數增加而減少。

第三節　降低差值擴張 (Reduced Difference Expansion)

2008 年，D. C. Lou 等人對 J. Tian 的 DE 演算法進行改良，提出了降低差值擴張(RDE)演算法 [10]。

圖 9 RDE 演算法差值變更步驟示意圖

修改的地方在於如圖 9 的差值變更步驟部分上，比 DE 演算法多了減少差值的動作(如紅框部分)，在資料嵌入差值之前會先將差值減少，其目的是減少使用擴張差值的方法後隨時會造成新像素值溢位的機率。因為依序減少再擴張差值的關係，新計算差值會比較接近原圖像差值，這同時也改善了因差值過度擴張而導致圖片載體像素值相差過大等問題，進而提升圖片品質。

Expandable 的方法上，在經過使用(2.1)將差值及平均算出來後，原差值 z 會被方程式(2.10)降低至約原差值 1/2 倍的值，製作出中間差值z'，之後才利用(2.2)將差值擴張及嵌入隱藏資料位元($z' \rightarrow z''$)，製造新的差值z''：

$$z' = \begin{cases} z & ,\text{if } z < 2 \\ z - 2^{\lfloor \log_2 z \rfloor - 1} & ,\text{otherwise} \end{cases} \tag{2.10}$$

為了成功恢復差值，在 RDE 演算法另外新增了一個位置地圖，專門記錄恢復中間差值z'到原差值所用到的額外資訊位元($LM = [0,1]$)：

$$LM = \begin{cases} 0 & ,\text{if } 2^{\lfloor \log_2 z' \rfloor} = 2^{\lfloor \log_2 z \rfloor} \; or \; z' = z \\ 1 & ,\text{if } 2^{\lfloor \log_2 z' \rfloor} \neq 2^{\lfloor \log_2 z \rfloor} \end{cases} \tag{2.11}$$

在解嵌入演算法上，從更改後的圖片載體提取出來的差值z''，則是反過來先用原 DE 演算法的方程式(2.6)提取資料及恢復成中間差值($z'' \rightarrow z'$)，之後用 RDE 的方法將原本縮減的中間差值z'恢復成原差值($z' \rightarrow z$)。如下圖(紅框是增加的步驟)：

圖 10 RDE 演算法差值還原步驟示意圖

在中間差值z'還原成原差值z的步驟上是以方程式(2.12)來還原，其中有使用在 RDE 嵌入演算法使用(2.11)所記錄的位置地圖資訊(LM)來補助還原差值：

$$z = \begin{cases} z' + 2^{\lfloor \log_2 z' \rfloor - 1} & ,if \; LM = 0 \\ z' + 2^{\lfloor \log_2 z' \rfloor} & ,if \; LM = 1 \end{cases} \tag{2.12}$$

在原 RDE 文獻有提到：DE 演算法通常會讓像素變更幅度到$|z|/2$，而在 RDE 演算法上最壞狀況下也頂多讓像素的變更幅度到$|z|/4$，最好情況甚至做到完全沒有變更像素值。使用 RDE 演算法後差值擴張的狀況推估會比 DE 演算法還要減少至少1/2~1倍。從此得知 RDE 演算法對差值擴張的抑制是挺有幫助的。

第四節　4 個像素為一組的差值擴張 (DE of Quads)

2004 年，A.M. Alattar 基於 DE 演算法做了修改，提供了 DE of quads 方法[11]。其作法是將原本的 2 個像素為一組變成 4 個像素作為一組，藉此增加單一圖片可以塞進的位元數，有助於提升載體容量。在 DE of quads 方法中，一個像素組可以在 4 個像素之間算出 3 個差值，並能夠嵌入 3 個隱藏資料位元，如圖 11。在一個圖片載體上最大的像素平均容量為0.75 $\left(=\frac{3}{4}\right)$ bpp (bit per pixel)，比 DE 演算法的0.5 (2 個像素可允許塞入 1bit 資料) bpp還要大。

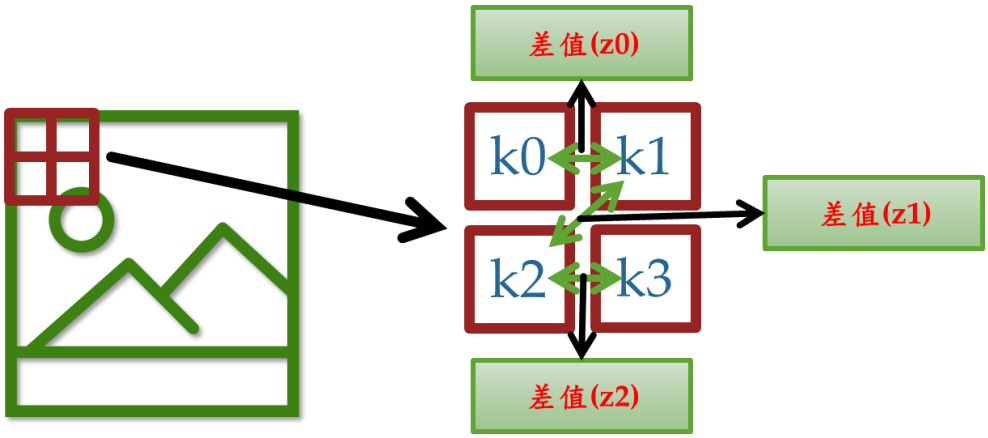


圖 11 DE of quads 演算法示意圖

DE of quads 在 DE 演算法主要修改的地方是在計算差值及恢復至像素這兩個地方上修改。提供一組向量數據$\boldsymbol{z} = (z_0, z_1, z_2, z_3)$，其中$z_0, z_1, z_2$分別儲存三個差值，$z_3$則是記錄平均，計算方程式如(2.13)所示。

$$\begin{cases} z_0 = k_1 - k_0 \\ z_1 = k_2 - k_1 \\ z_2 = k_3 - k_2 \\ z_3 = \left\lfloor \dfrac{k_0 + k_1 + k_2 + k_3}{4} \right\rfloor \end{cases} \tag{2.13}$$

差值嵌入方式是引用原本 DE 演算法所用過的(2.2)跟(2.8)，但因為一組像素組允許使用 3 個差值儲存資料，所以各差值將各自嵌入隱藏資料 3 個位元$\boldsymbol{b} = (b_0, b_1, b_2)$中的一個位元，嵌入方程式將被整理成如(2.14)(2.15)所示(各自代表 expandable 跟 changeable 的方式)：

$$\begin{cases} z'_0 = 2 \times z_0 + b_0 \\ z'_1 = 2 \times z_1 + b_1 \\ z'_2 = 2 \times z_2 + b_2 \end{cases} \tag{2.14}$$

$$\begin{cases} z'_0 = 2 \times \left\lfloor \dfrac{z_0}{2} \right\rfloor + b_0 \\ z'_1 = 2 \times \left\lfloor \dfrac{z_1}{2} \right\rfloor + b_1 \\ z'_2 = 2 \times \left\lfloor \dfrac{z_2}{2} \right\rfloor + b_2 \end{cases} \tag{2.15}$$

寫回像素的方程式有一些改變，使用的是跟(2.13)對應的反轉換方程式，如(2.16)。

$$\begin{cases} k'_0 = z'_3 - \left\lfloor \dfrac{3z'_0 + 2z'_1 + z'_2}{4} \right\rfloor \\ k'_1 = z'_0 + k'_0 \\ k'_2 = z'_1 + k'_1 \\ k'_3 = z'_2 + k'_2 \end{cases} \tag{2.16}$$

同樣的，在 DE of quads 演算法中，新的像素組中所有像素值不能超過灰階像素值所定義的範圍。一旦有其中一個以上的像素值不符合，整個像素組就會被判定有溢位問題，不能取代原本的像素組的數值。

同時 DE of quads 也支援在 DE 演算法上的多重嵌入模式的機制，可搭配圖 6 的順序圖表使用來做嵌入模式的判斷。只不過要注意的是：一個像素組僅支援一種嵌入模式，像素組中的所有像素值都必須同時支援同一組模式(expandable 或 changeable)，該像素組才可以使用該模式並記錄在位置地圖上。

解嵌入演算法也跟 DE 演算法大同小異，搭配圖 8 的步驟確認嵌入模式後，使用(2.13)提取差值再引用 DE 的方法(2.6)或(2.9)提取隱藏資料並回復至原差值，隨後用(2.16)恢復成原像素值即可。

第五節　RDE 跟 Quads 結合的降低差值擴張 (RDE of quads)

2013 年，T. Ahmad 等人結合了 D. C. Lou 的 RDE 及 A.M. Alattar 的 DE of quads 這兩種方法，提出了新的改良隱寫術 [12]，透過結合 RDE 的降低差值及 DE of quads 的增加可使用差值數這兩種優點，可同時增加載體容量以及提高圖片品質。為方便起見，在本文獻上我們簡稱此隱寫術為「RDE of quads」。

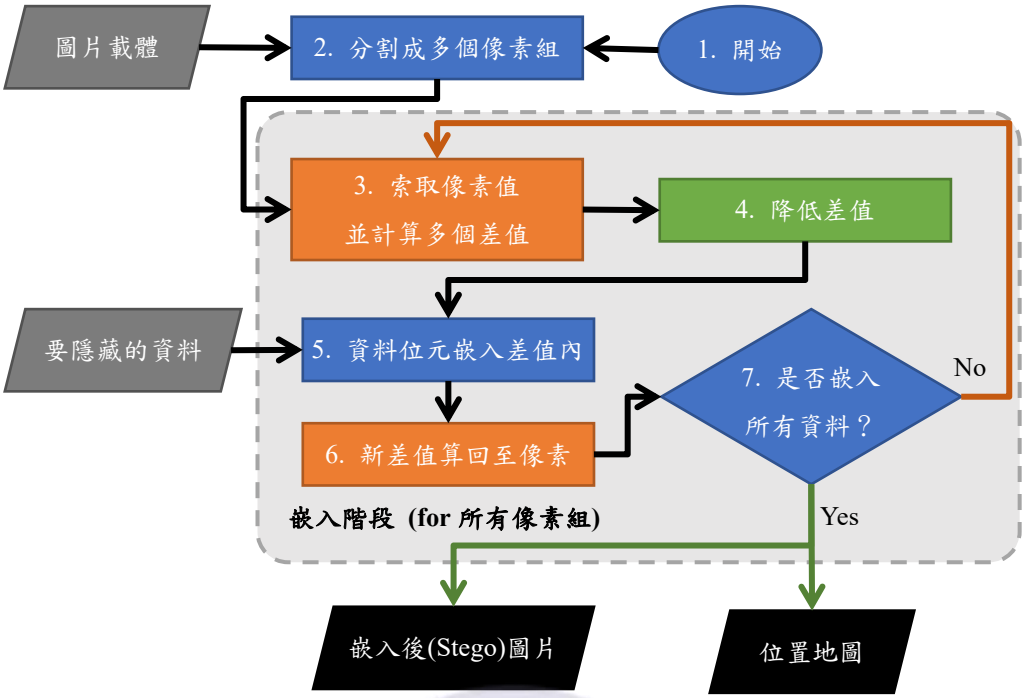


圖 12 RDE of quads 簡易步驟構造圖

如圖 12 所示，RDE of quads 是同時結合了 RDE 及 DE of quads 的技術所構成的新隱寫術。其中步驟 3 及步驟 6(橘色框部分)是取自 DE of quads 採用 4 個像素組成一組的機制，但在 RDE of quads 的文獻中，步驟 3 只使用了 DE of quads 演算法中(2.13)計算三個差值部分，不計算平均：

$$\begin{cases} z_0 = k_1 - k_0 \\ z_1 = k_2 - k_1 \\ z_2 = k_3 - k_2 \end{cases} \tag{2.17}$$

並在步驟 6 上改成使用下列方程式來恢復像素值，其中一個新像素值以原像素值替換：

$$\begin{cases} k'_0 = k_0 \\ k'_1 = z'_0 + k'_0 \\ k'_2 = z'_1 + k'_1 \\ k'_3 = z'_2 + k'_2 \end{cases} \tag{2.18}$$

步驟 4(綠框部分)即是 RDE 中 expandable 方法上的降低差值機制，一個像素組的 3 個差值都要套用 RDE 的方法(2.10)。相對地，每個像素組在位置地圖上都需要使用(2.11)儲存 3 位元恢復降低差值的額外資料($\boldsymbol{LM} = (LM_0, LM_1, LM_2)$ $where\ LM_n = [0,1]$)，以用來個別恢復 3 個差值(2.12)。

順帶一提，RDE of quads 演算法因為使用了 RDE 降低差值的方法，減少了差值會發生溢位的機率，間接提高可嵌入資料的差值數量，跟 DE of quads 比起來，此演算法更能提高載體的容量。

RDE of quads 被提出之後，T. Ahmad 等人又在後來的幾年又提出了幾篇以 RDE of quads 為基礎優化的隱寫術，我們索取了其中的兩篇在以下兩節做說明：

第六節　16 個像素為一組的降低差值擴張 (RDE of block 16)

2015 年，T. Ahmad 等人以 RDE of quads 演算法為基礎提出了一個改良的隱寫術 [13]，其作法是從原本只有 4 個像素的 quads 擴張成 16 個像素為一組，並在降低差值的方法上下功夫，目的是為了進一步地增加載體容量，同時提高圖片的質量。在本文獻我們簡稱此隱寫術為「RDE of block 16」。

圖 13 RDE of quads(左)跟 RDE of block 16(右)之間的像素組差異

如圖 13 所示，RDE of block 16 演算法擴大了像素組的大小，每個像素組包含了 16(4×4)個像素，可允許儲存隱藏資料的差值就有 15 個，相當於可以嵌入 15 位元的資料，最大像素平均容量也被擴增到0.9375 $\left(=\frac{15}{16}\right)$ bpp，比 RDE of quads 的 0.75 bpp 還要大。

計算差值及差值算回像素的方程式皆繼續採用 RDE of quads 的方式，不計算平均值，且其中一個新像素值皆以原像素值代替，如(2.19)及(2.20)：

$$\begin{cases} z_0 = k_1 - k_0 \\ z_1 = k_2 - k_1 \\ \quad\vdots \\ z_{14} = k_{15} - k_{14} \end{cases} \tag{2.19}$$

$$\begin{cases} k'_0 = k_0 \\ k'_1 = z'_0 + k'_0 \\ k'_2 = z'_1 + k'_1 \\ \vdots \\ k'_{15} = z'_{14} + k'_{14} \end{cases} \tag{2.20}$$

跟 RDE 一樣，在嵌入演算法的 expandable 方法中，擴增差值前會先將差值減少，但在 RDE of block 16 的降低差值方法有些變更。為防止圖片便失真幅度變大，降低差值方程式不僅引用原 RDE 演算法的(2.10)，還多減掉了$[\log_2 z_n]$(z_n代表第 n 個差值，$n = 0,1,2,\dots,14$)，如下列方程式所示：

$$z'_n = \begin{cases} z_n - \left(2^{\lfloor \log_2 z_n \rfloor} + [\log_2 z_n]\right), \text{if } z_n > 1 \\ z_n + \left(2^{\lfloor \log_2 z_n \rfloor}\right) + ([\log_2 z_n]), \text{if } z_n < -1 \end{cases} \tag{2.21}$$

一個像素組總共要記錄 15 個恢復降低差值的額外資料($\boldsymbol{LM} = (LM_0, LM_1, LM_2, \dots, LM_{14})$)，在這裡也同樣使用位置地圖的方式記錄，如下列方程式所示：

$$LM_n = \begin{cases} 0 & \text{if } z'_n \pm \left(2^{log_2(\lfloor z'_n \rfloor)-1}\right) = z_n \\ 1 & \text{if } z'_n \pm \left(2^{log_2(\lfloor z'_n \rfloor)-1}\right) \neq z_n \end{cases} \tag{2.22}$$

在解嵌入演算法恢復降低差值至原差值時，改用(2.23)方程式，這同時搭配從(2.22)計算而來的位置地圖(LM_n)來補助恢復差值((2.23)中的±符號是在z'_n分別為>1或<-1的情況下，就分別使用+或−)：

$$z_n = \begin{cases} z'_n \pm \left(2^{\log_2(\lfloor z'_n \rfloor)-1} + \log_2(\lfloor z'_n \rfloor)\right) - 1 & \text{if } LM_n = 0 \\ z'_n \pm \left(2^{\log_2(\lfloor z'_n \rfloor)} + \log_2(\lfloor z'_n \rfloor)\right) & \text{if } LM_n = 1 \end{cases} \tag{2.23}$$

在該文獻上結果顯示：RDE of block 16 雖然可以擴增圖片載體容量，但因為單位像素組擴大至 16 個像素的關係，增加了被覆寫的像素數量，導致平均每個像素值的變更量比 RDE of quads 更大，圖片載體的失真率變得更嚴重。

第七節　4×1 新降低差值擴張 (Enhance-RDE of quads)

2017 年，同一個作者又在 RDE of quads 演算法做了一些優化修正，提出了另一個改良隱寫術 [14]。他們提議了類似垂直掃描的機制，在每一像素組的長寬設定被改成4×1的方式，並從降低差值及計算差值的方法修改，為的就是盡可能地

縮小差值以及將資料嵌入至更小的差值內，以提高圖像的品質、容量。在本文獻我們簡稱此隱寫術為「Enhance-RDE of quads」。

在計算差值的部分上，改成每個差值都是以最後一個像素為準，如方程式(2.24)；將差值寫回像素的方法則改成對應(2.24)的反轉換方程式，如(2.25)：

$$\begin{cases} z_0 = k_0 - k_3 \\ z_1 = k_1 - k_3 \\ z_2 = k_2 - k_3 \end{cases} \tag{2.24}$$

$$\begin{cases} k'_0 = z'_1 + k_3 \\ k'_1 = z'_1 + k_3 \\ k'_2 = z'_2 + k_3 \\ k'_3 = k_3 \end{cases} \tag{2.25}$$

改成上列兩個方程式的目的可能是為了修正在差值被放大的情況下，原本RDE of block 16 方法中(2.19)與(2.20)因像素之間的牽連而導致像素變動幅度一個比一個大等問題。使用(2.24)與(2.25)的話，因為各個差值是從一個像素值與指定的像素值之間計算出來的，變動差值只會影響一個像素值，不會影響到其他像素，在一個圖片中的像素值平均變動幅度就會少一些。

Expandable 方法中的降低差值方法參考了從 RDE of block 16 的方程式(2.21)並經過一些修改，可以進一步將差值縮小，也能將隱藏資料位元嵌入至更小的差值中，如(2.26)所示。

$$z'_n = \begin{cases} z_n & ,if\ -1 \le z_n \le 1 \\ z_n - \left(2^{\lfloor \log_2 z_n - 1 \rfloor} + \lfloor \log_2 z_n - 1 \rfloor\right) & ,if\ z_n > 2 \\ z_n + \left(2^{\lfloor \log_2 z_n \rfloor} + \lfloor \log_2 z_n \rfloor\right) & ,if\ z_n < -2 \end{cases} \tag{2.26}$$

恢復降低差值的額外資訊(LM_n where $n = 0,1,2$)改使用(2.27)此方程式計算並記錄在位置地圖上：

$$LM_n = \begin{cases} 0 & \text{if } z'_n \pm \left(2^{\log_2(\lfloor z'_n \rfloor)-1}\right) + 2(\lfloor \log_2 z'_n \rfloor) = z_n \\ 1 & \text{if } z'_n \pm \left(2^{\log_2(\lfloor z'_n \rfloor)-1}\right) + 2(\lfloor \log_2 z'_n \rfloor) \neq z_n \end{cases} \tag{2.27}$$

恢復降低差值至原差值的方程式改用下列兩個方程式：

1. 若$z'_n \ge 2$的話：

$$z_n = \begin{cases} z'_n + \left(2^{\lfloor \log_2 |z'_n| \rfloor - 1} + \lfloor \log_2 2z'_n \rfloor - 1\right) & \text{if } LM_n = 1 \\ z'_n + \left(2^{\lfloor \log_2 |z'_n| \rfloor} + \lfloor \log_2 2z'_n \rfloor\right) & \text{if } LM_n = 0 \end{cases} \quad (2.28)$$

2. 若$z'_n \leq 2$的話：

$$z_n = \begin{cases} z'_n + \left(2^{\lfloor \log_2 |z'_n| \rfloor} + \lfloor \log_2 2z'_n \rfloor\right) & \text{if } LM_n = 0 \\ z'_n - \left(2^{\lfloor \log_2 |z'_n| \rfloor} + \lfloor \log_2 2z'_n \rfloor\right) & \text{if } LM_n = 1 \end{cases} \quad (2.29)$$

該文獻宣稱 Enhance-RDE of quads 會比 RDE of block 16 的嵌入效果更好，不僅在緩和差值擴張方面上有所幫助，提升嵌入資料後圖片的品質，同時也間接提升圖片載體容量。

第八節　文獻總結

我們總結前人的成果特點如下：

1. 差值擴張(DE) [9]：雖然 DE 演算法，在當代的隱寫術領域中是被聲稱有複雜度低、高容量等優點，但在擴張差值的特性上，有些比較大的差值經擴張後會造成對應的像素值變動幅度過大，甚至造成溢位等問題，對比現代的演算法仍然沒那麼出色。
2. 降低差值擴張(RDE) [10]：RDE 改善了原 DE 所存在的像素值變動幅度大的缺點，提升了圖片品質及容量。但相對的，它比 DE 演算法要多做降低差值的步驟，需額外儲存位置地圖數據作為恢復降低差值的額外資訊。
3. DE of Quads [11]：它比傳統 DE 演算法增加圖片載體平均一個像素可儲存的最大容量。但是一個像素組僅能使用一個嵌入模式，並且所有像素值必須符合該嵌入模式的條件，自由度比 DE 的還要低。
4. RDE of Quads [12]：它是結合了 RDE 及 DE of Quads 兩大演算法做出來的隱寫術，所以同時兼具了高品質、高容量兩大優點。但我們認為其演算法輸出圖片的載體容量、品質仍然還有待改進的空間。
5. RDE of Block 16 [13]：使用了 RDE of Quads 為底製作的隱寫術，比

RDE of Quads 更能儲存更多資料容量。但因像素組的過度放大，造成演算結果顯示像素平均變動量比 RDE of Quads 增得更大，導致圖片品質下降。

6. Enhance-RDE of Quads [14]：從 RDE of Quads 演算法調整改良計算差值的方式，使得在計算差值時能有助於緩和初始差值的大小，改善圖片品質。

不過，在 RDE of Block 16 及 Enhance-RDE of Quads 這兩個演算法文獻中我們發現：雙方的降低差值演算法基本上都是採用同一種降低模式，都是搭載 RDE 的降低方程式(2.10)再多降低一些數值的手法，而兩邊演算法上的降低差值方程式在式子構成上有一些問題，經過我們的驗算，有部分差值會無法恢復原差值，很有可能導致無法將嵌入後的圖片恢復成原圖片。

比如 Enhance-RDE of Quads 上的(2.26)及(2.28)(2.29)：對可逆式演算法而言，解嵌入演算法必須有辦法將嵌入演算法製造出來的新圖片還原成原本載體圖片的本質，所以解嵌入演算法相當於嵌入演算法的反函數。作為恢復降低差值演算法的(2.28)跟(2.29)照理來說應該跟(2.26)有所相關，其方程式應該大致寫成$z_n = z'_n + \left(2^{\lfloor \log_2 z'_n \rfloor - 1} + \lfloor \log_2 z'_n \rfloor - 1\right)$及$z_n = z'_n - \left(2^{\lfloor \log_2 z'_n \rfloor} + \lfloor \log_2 z'_n \rfloor\right)$才對。

還有一個例子，就是在恢復降低差值用的位置地圖(LM_n)參數部分：通常在 RDE of Block 16 的(2.22)及 Enhance-RDE of Quads 的(2.27)是用來儲存在降低演算法((2.21) & (2.26))上降低的差值幅度方便地將幅度再加回去((2.23) & (2.28) & (2.29))的補助資訊，其方程式必定跟原降低差值方程式相關。在 RDE of Block 16 及 Enhance-RDE of Quads 的場合下，計算恢復降低差值額外資訊的方程式大致上應為(2.30)才對。

$$LM_n = \begin{cases} 0 & \text{if } z'_n \pm \left(2^{\lfloor \log_2 z'_n \rfloor - 1} + \lfloor \log_2 z'_n \rfloor - 1\right) = z_n \\ 1 & \text{if } z'_n \pm \left(2^{\lfloor \log_2 z'_n \rfloor - 1} + \lfloor \log_2 z'_n \rfloor - 1\right) \neq z_n \end{cases} \tag{2.30}$$

就算在 RDE of Block 16 及 Enhance-RDE of Quads 的降低差值部分沒有式子構成上的問題，降低差值演算法仍然還是無法恢復部分差值至原差值，尤其是在

原差值為2^x (where $x \in$ N)的情況下：因為計算恢復降低差值額外資訊的方程式(2.30)允許恢復的差值變更幅度介於0~$\frac{1}{2}z_n$之間，而使用降低差值方程式((2.21)或(2.26))會使得計算出來的差值小於$2^{(x-1)}$，相對地差值變更幅度會大於$2^{(x-1)}$(=$\frac{1}{2}z_n$)，這已經超過了(2.30)允許的差值變更幅度，表示該差值若使用在 RDE of Block 16 或 Enhance-RDE of Quads 的降低差值模式時會無法恢復原差值。

另外，我們認為使用 RDE of Block 16 或 Enhance-RDE of Quads 的降低差值演算法有可能造成降低的差值不夠收斂，即是沒有將降低過後的差值接近原差值的一半。以上這些問題都是我們想要修正及改良前人演算法的原因。

第三章　改良式圖像隱寫術

本論文提出了一種演算法，它是基於 Enhance-RDE of Quads [14]隱寫術方法做一部份的演算法內部改良，主要目的是以改善輸出後圖片品質最為優先。在該演算法我們使用特殊方法來優化降低差值演算法的收斂度。

另外，在上一章我們提到 Enhance-RDE of Quads [14]原本的演算法有部分方程式有問題，導致其演算法是無法在所有差值的情況下，都能經過嵌入跟解嵌入步驟後，完全恢復為原本差值的。為此，我們也在此章節提出了我們的修正方案，用來解決我們對 Enhance-RDE of Quads 所提過的問題。

以下是我們對 Enhance-RDE of Quads 提出的修正方案及我們提出的改良方案：

第一節　針對 Enhance-RDE of Quads 的修正方式

在 Enhance-RDE of Quads 的降低差值演算法中，我們將替換成如圖 14 所表示的方案：

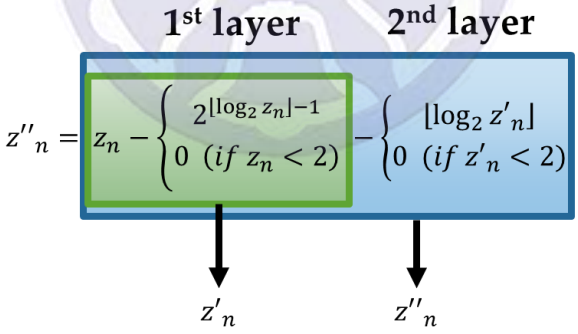


圖 14 Enhance-RDE of Quads 降低差值的修正方案結構

同樣是先沿用了(2.10)再降一點數值的手法，在我們的修正方案上改成了雙階段的方式，將$2^{\lfloor \log_2 z_n \rfloor - 1}$與$\lfloor \log_2 z'_n \rfloor$這兩個部分分開並在各自的階段分別計算，以便容易處理。同時將位置地圖(*LM*)的數量改為每個差值各有兩個($LM2_n$, $LM3_n$)，一個像素組就需要儲存 6 個位置地圖數值：$\boldsymbol{LM2} = (LM2_0, LM2_1, LM2_2)$、$\boldsymbol{LM3} = (LM3_0, LM3_1, LM3_2)$。這麼做是為了避免 *LM* 只有一個的情況下造成有一些差值跟降低差值演算法不相容的狀況。雖然這個修正方案跟原本 Enhance-RDE of

Quads 的文獻 [14]上所使用的降低差值做法(2.26)仍然有形式上的差異，但在功能及成效上都差不多。所以我們認為，我們的修正方案確實可以套用在 Enhance-RDE of Quads 上。為方便稱呼，我們把 Enhance-RDE of Quads 套用過我們降低差值做法的新修正演算法為「Fixed Enhance-RDE of Quads」。

在 Fixed Enhance-RDE of Quads 嵌入側的降低差值演算法架構可以用圖 15 來顯示(紅框為變動部分)：

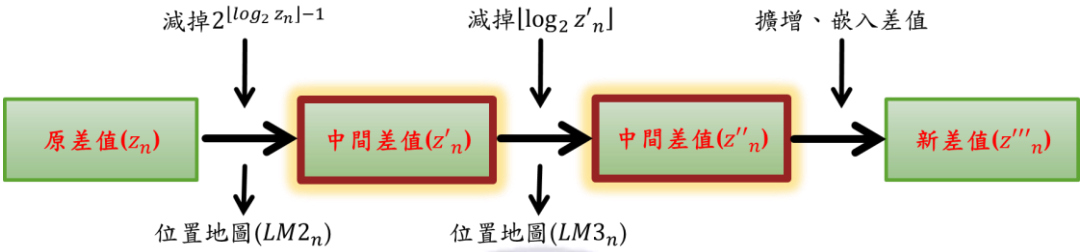


圖 15 Fixed Enhance-RDE of Quads 降低差值演算法架構

其中z'''_n是經過 Fixed Enhance-RDE of Quads 雙階段手法降低差值，並經過擴張差值並嵌入資料之後的新差值。z'''_n其功能跟 RDE 的z''_n一樣，都是經嵌入處理過後準備存入至像素的預備數值。

透過圖 14 的這一整串的方程式，我們可以再轉換為下列兩個算式：

$$z'_n = z_n - \begin{cases} 2^{\lfloor \log_2 z_n \rfloor -1}, \text{if } z_n \geq 2 \\ 0 \quad\quad\quad\quad, \text{if } z_n < 2 \end{cases} \tag{3.1}$$

$$z''_n = z'_n - \begin{cases} \lfloor \log_2 z'_n \rfloor, \text{if } z'_n \geq 2 \\ 0 \quad\quad\quad\quad, \text{if } z'_n < 2 \end{cases} \tag{3.2}$$

將像素組中求出來的差值依序用(3.1)跟(3.2)來減少差值，此後再用下列方程式計算恢復降低差值的額外資訊$LM2_n, LM3_n$並儲存至位置地圖內，才能繼續執行接下來的嵌入差值動作(2.14)($z''_n \rightarrow z'''_n$)：

$$LM2_n = \lfloor \log_2 z_n \rfloor - \lfloor \log_2 z'_n \rfloor, LM3_n = \lfloor \log_2 z'_n \rfloor - \lfloor \log_2 z''_n \rfloor \tag{3.3}$$

在解嵌入演算法上的恢復降低差值演算法中，則是採用原圖 15 降低差值演算法的相反步驟。在經過(2.6)取出隱藏資料、還原差值($z'''_n \rightarrow z''_n$)之後，搭配從(3.3)算出來的位置地圖資訊$LM2_n, LM3_n$，先後使用(3.4)與(3.5)來還原至原本的差值：

$$z'_n = z''_n + (\lfloor \log_2 z''_n \rfloor + LM3_n) \tag{3.4}$$

$$z_n = z'_n + 2^{\lfloor \log_2 z'_n \rfloor - 1 + LM2_n} \tag{3.5}$$

另外，在我們提出的修正演算法中，降低及還原差值的方程式(3.1)、(3.2)、(3.4)跟(3.5)只適用於正數的差值，所以使用該方程式之前，先記錄原差值是正數還是負數，並將原差值一律轉為正數：

$$isN, z_n = \begin{cases} 0, z_n & \text{if } z_n \geq 0 \\ 1, -z_n & \text{if } z_n < 0 \end{cases} \tag{3.6}$$

待計算完成後，再次將原本的差值轉回原本的正數或負數：

$$z_n = \begin{cases} z_n & \text{if } isN = 0 \\ -z_n & \text{if } isN = 1 \end{cases} \tag{3.7}$$

Fixed Enhance-RDE of Quads 使用了跟圖 6 類似的像素組嵌入模式判斷架構，但判斷方式有些不同，如圖 16：

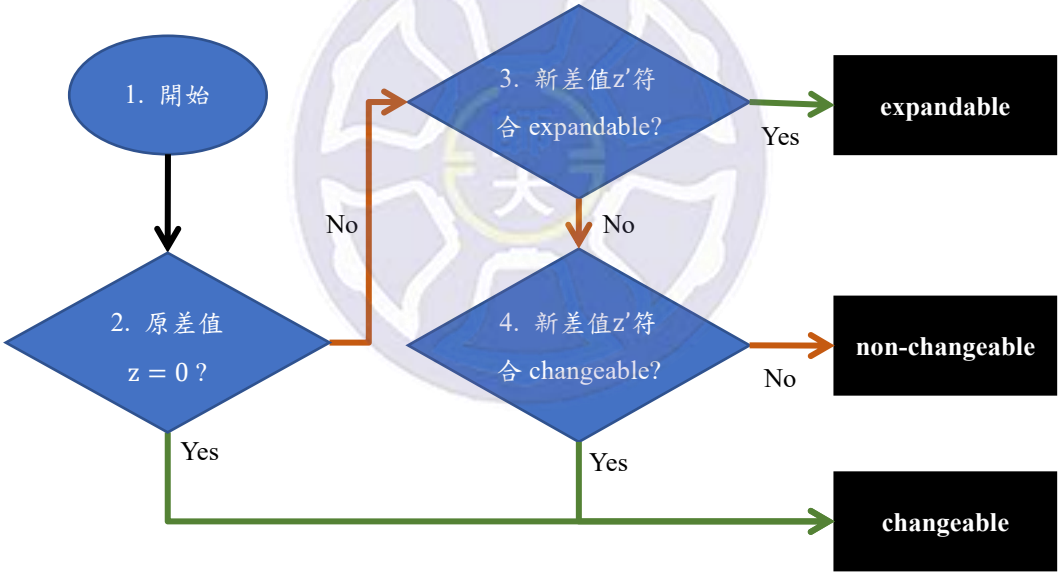


圖 16 Fixed Enhance-RDE of Quads 嵌入模式判斷架構

其中 expandable 即是我們在這一節所說的兩階段式降低差值方法以及擴張嵌入差值演算法的所屬模式。Changeable、non-changeable 繼續保有原本 DE 演算法所擁有的 Changeable、non-changeable 步驟。另外，步驟 2 則改為差值 z 為 0 的時候，直接判定成 changeable。雖然說差值為 0 時，使用 expandable 仍然可以使嵌入並還原的差值變成 0，但是考慮到在降低差值部分只使用第一階段方程式(3.1)時，原差值 0 經嵌入並還原的差值會變成$-\infty$，怕很有可能造成還原差值不

穩定的問題而特地設了這個條件。

另外，在 changeable 模式下，同樣是套用原本 DE 演算法所提過的 changeable 步驟，只不過因為在一個像素組中，總共要使用三個差值嵌入資料，所以在嵌入資料時，每個差值都要套用此方程式(2.8)各自嵌入要隱藏的資料其中一個位元，同時也套用了(2.7)在嵌入差值之前先將原差值的 LSB 提取的機制。在一個像素組中，總共需要計算三個 *BS* 參數($\boldsymbol{BS} = (bs_0, bs_1, bs_2)$)並儲存至位置地圖內，每一個 *BS* 參數各自對應像素組其中一個差值。在解嵌入演算法同樣在像素組三個差值中，搭配從位置地圖儲存下來的三個 *BS* 參數，使用(2.9)在每個差值上提取資料、回復原差值即可。

順帶一提，在嵌入演算法中判斷符合嵌入模式，我們是採用先計算再判定的模式的方法來做判斷檢查，如圖 17。我們使用了如圖 16 所提及的檢查順序：expandable → changeable → non-changeable 如同多重關卡般那樣地檢查，為該像素組尋找適合地嵌入模式。在每個關卡上，先將新的像素值算出來，事後再檢查新的像素值是否有溢位。若沒有溢位，就適用該嵌入模式，同時也將原像素值覆寫成新的像素值，並儲存該模式所用到的位置地圖資料；若有像素值溢位則跳到下一關卡，以此類推，直到 non-changeable 為止。

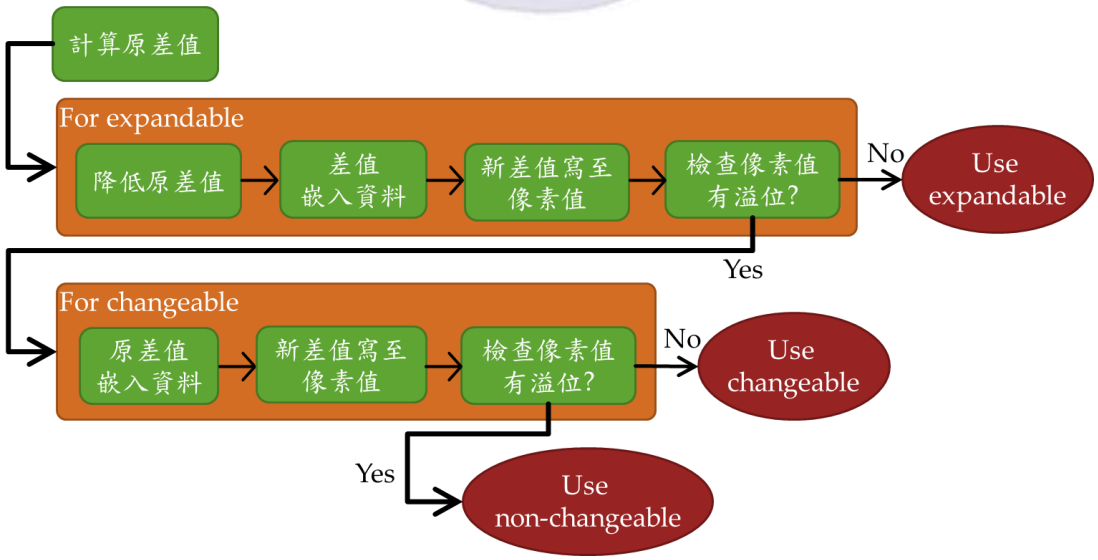


圖 17 Fixed Enhance-RDE of Quads 嵌入模式判斷作法

像素組嵌入模式的紀錄上，我們使用 *CH*、*LM1* (= [0,1])參數作為像素組嵌

入模式的判斷標籤，儲存在位置地圖內，其個別模式中標籤標示如表 1 所示：

表 1 各嵌入模式標籤定義

模式種類	CH	LM1
non-changeable	0	0
changeable	1	0
expandable	1	1

在嵌入演算法上可用表 1 上的標籤定義儲存該像素組的嵌入模式，解嵌入演算法則是使用同樣的表來參考，判定該像素組用了甚麼嵌入模式。

在 Fixed Enhance-RDE of Quads 上用到的位置地圖資料參數總共有五種，各參數功能如表 2 所示：

表 2 Fixed Enhance-RDE of Quads 各位置地圖參數功能定義

參數	容量 (每個像素組)	功能
$CH, LM1$	各1 bit	紀錄像素組的嵌入模式，標籤定義請參考表 1。
$\boldsymbol{LM2}$	1 bit × 3	紀錄 expandable 模式第一階段用位置地圖(LM)參數
$\boldsymbol{LM3}$	1 bit × 3	紀錄 expandable 模式第二階段用位置地圖(LM)參數
$\boldsymbol{BS}$	1 bit × 3	紀錄 changeable 模式時各差值的 LSB

其中$\boldsymbol{LM2}$、$\boldsymbol{LM3}$跟$\boldsymbol{BS}$皆是由 3 個數值所組成的向量，每一個數值各自對應像素組的每一個差值。另外$\boldsymbol{LM2}$、$\boldsymbol{LM3}$跟$\boldsymbol{BS}$在一個像素組中不會同時存在：使用 expandable 模式只儲存$\boldsymbol{LM2}$、$\boldsymbol{LM3}$這兩組參數，而 changeable 只要儲存$\boldsymbol{BS}$這一組即可。下面我們再整理出了每個嵌入模式所用到的位置地圖參數，以及其所需要的儲存空間，如表 3：

表 3 Fixed Enhance-RDE of Quads 各嵌入模式所需位置地圖參數及其使用空間

參數名稱 / 嵌入模式	CH	$LM1$	$\boldsymbol{LM2}$	$\boldsymbol{LM3}$	$\boldsymbol{BS}$	所需容量

Expandable	✔	✔	✔	✔		8 bits
Changeable	✔	✔			✔	5 bits
Non-changeable	✔	✔				2 bits

總結以上講過的修正方案、原 Enhance-RDE of Quads 演算法及其它我們所提過的一些細節，我們可以統整出 Fixed Enhance-RDE of Quads 演算法的構造。

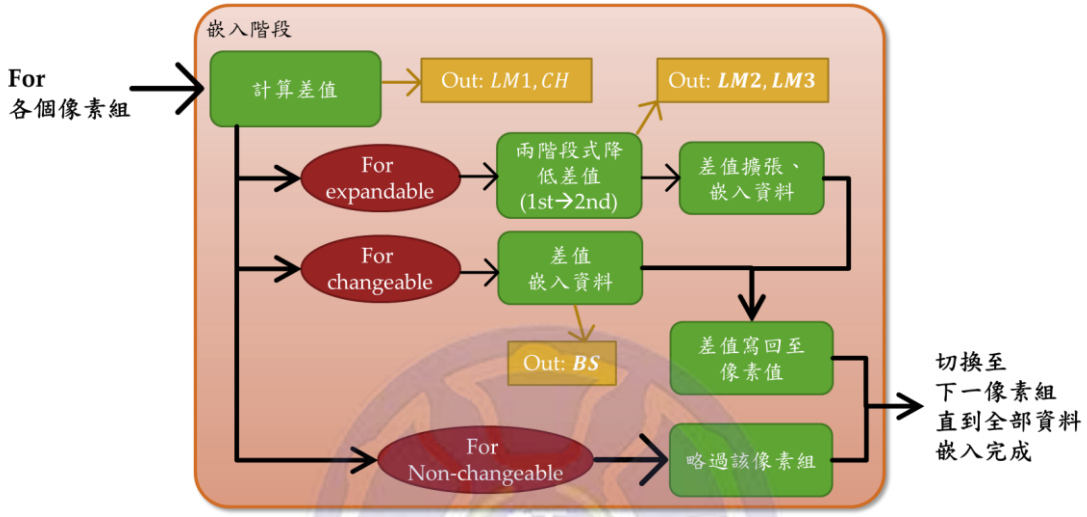


圖 18 Fixed Enhance-RDE of Quads 嵌入資料側構造

在嵌入資料側，先將一張灰階圖片分成分成若干組4×1的像素組，並將每個像素組都做出如圖 18 嵌入資料的動作，全部像素組嵌入後若資料還有剩，就再用此圖片繼續嵌入，直到所有資料嵌入為止。其中圖 18 的嵌入階段部分，除了「計算差值」及「略過該像素組」這兩個步驟，整體演算法是採用圖 17 使用的嵌入模式檢查手法，邊計算新像素值邊檢查有無溢位，確定套用某特定嵌入模式就直接儲存運算過的新像素值及位置地圖資料，不必再計算第二次。而「兩階段式降低差值」即是使用我們在這一節提過的降低差值修正方案。

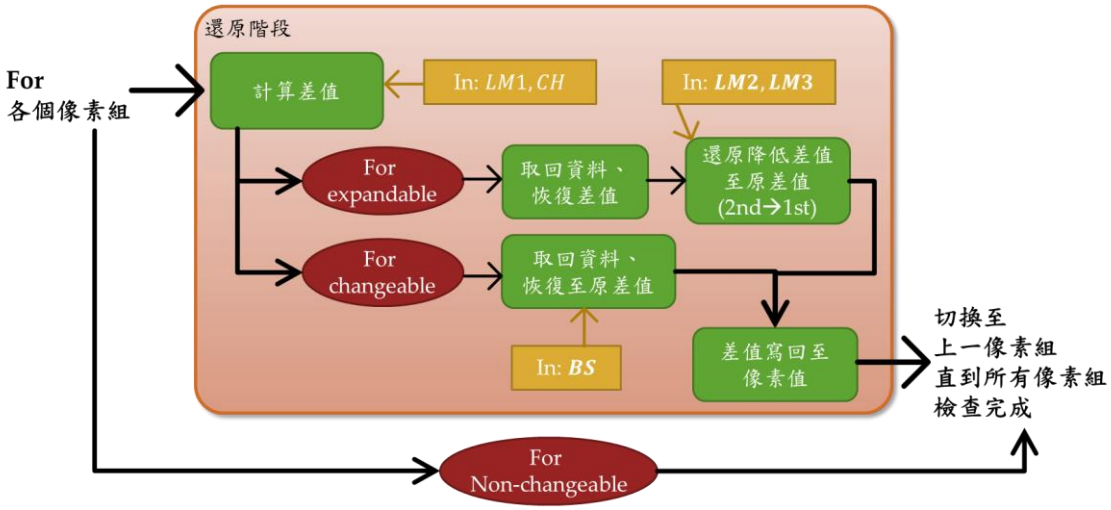


圖 19 Fixed Enhance-RDE of Quads 提取資料側構造

提取資料側的作法是採用嵌入資料側完全反方向的步驟。同樣先把一張已嵌入資料的灰階圖片分成若干組4 × 1的像素組，並從上一次儲存的像素組位置開始如圖 19 做提取資料的動作，若還沒取完所有隱藏資料就再拿此圖片從最後一個像素組開始檢查，直到檢查過所有像素組為止。其中「還原降低差值至原差值」即是使用我們在這一節提過的兩階段式降低差值修正方案中的還原差值方式，即(3.4)跟(3.5)。另外在該像素組為 Non-changeable 的情況下就直接略過該像素組，跳到上一個像素組繼續。

接下來介紹的是我們的改良演算法，它是針對我們從 Enhance-RDE of Quads 演算法修正而來的 Fixed Enhance-RDE of Quads 作為基礎，在內部改寫其部分演算法來加強差值的收斂，以提升輸出圖片的品質。要記住：差值是由兩個像素值之間求出來的差距大小，只要隱寫演算法有辦法將新差值收斂到一定的程度，使得新的差值近似於原差值，那麼之後計算出來的新像素值相對於原像素值的變更幅度就不會太大。

為達成目的，在下面我們介紹兩個我們提出的改良方案：

1. 加強兩階段式降低差值收斂
2. 交換像素值以降低原差值

在此論文，我們將我們所提出的改良演算法稱為「Improved Two-Steps-RDE of Quads」。

第二節　加強兩階段式降低差值收斂

透過圖 15 我們可以知道，Fixed Enhance-RDE of Quads 的降低差值方式是使用兩個階段方式將差值降低再降低，但對我們而言，這樣的方法仍然不夠收斂。所以在 Improved Two-Steps-RDE of Quads 上，我們再次運用 Fixed Enhance-RDE of Quads 降低差值的方法，並嘗試在其中一個降低階段做改良，使用了類似除法的商數餘數概念方法來加強差值的收斂度，使降低過後的差值更接近原差值的一半。

改良過的降低差值演算法如圖 20 所示：

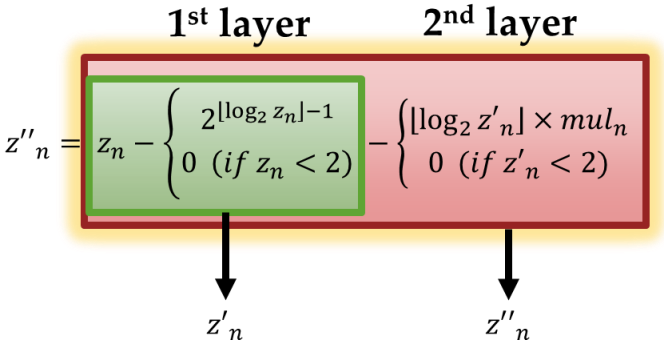


圖 20 Improved Two-Steps-RDE of Quads 降低差值演算法結構

其中圖 20 上的第二降低差值階段(2nd layer，即紅框部分)是在原本圖 14 中的第二階段再多加了乘積參數(mul_n)。這個點子是透過原 Fixed Enhance-RDE of Quads 的第二降低差值階段(3.2)只降低一點點差值的特性而想到的點子，目的是為了藉由多次使用第二階段降低差值，使降低後差值z''_n漸漸趨近原差值的一半(亦即$z''_n \approx \frac{1}{2}z_n$)，製造出來的新差值$z'''_n$才會更接近原差值$z_n$。

透過圖 20，我們可以在區分為下列兩個方程式：

$$z'_n = z_n - \begin{cases} 2^{\lfloor \log_2 z_n \rfloor - 1}, \text{if } z_n \geq 2 \\ 0 \quad\quad\quad, \text{if } z_n < 2 \end{cases} \tag{3.8}$$

$$z''_n = z'_n - \begin{cases} \lfloor \log_2 z'_n \rfloor \times mul_n, \text{if } z'_n \geq 2 \\ 0 \quad\quad\quad\quad\quad, \text{if } z'_n < 2 \end{cases} \tag{3.9}$$

在 Improved Two-Steps-RDE of Quads 的降低差值演算法中，乘積參數(mul_n)成為了收斂差值的關鍵，因為它決定了第二降低差值階段的使用次數。要達成收斂的目的，需要使用一系列演算法來計算mul_n。經過第一降低差值階段(1st layer)(3.8)計算出中間差值z'_n，並使用(3.3)計算出位置地圖$LM2_n$後，繼續依照以下步驟執行第二降低差值階段，同時計算mul_n參數：

1. 計算中間差值z'_n與原差值一半($\frac{1}{2}z_n$)的差距(在這裡我們稱此參數為 *offset*)：

$$offset = z'_n - \left\lfloor \frac{z_n}{2} \right\rfloor \tag{3.10}$$

2. 若$offset > 0$，且中間差值$z'_n \geq 2$，即可繼續下一步。否則的話mul_n跟$LM3_n$直接為 0，並略過其他步驟。

3. 暫時求出mul_n參數：

$$mul_n = \left\lfloor \frac{offset}{\lfloor \log_2 z'_n \rfloor} \right\rfloor \tag{3.11}$$

4. 為了要讓(3.9)計算過後的差值z''_n更接近$\frac{1}{2}z_n$，mul_n會被做調整。先求出目前用(3.9)計算過後的差值z''_n跟$\frac{1}{2}z_n$的距離，也就是餘數(*remainder*)參數：

$$remainder = offset - mul_n \times \lfloor \log_2 z'_n \rfloor \tag{3.12}$$

5. 假使餘數比$\lfloor \lfloor \log_2 z'_n \rfloor / 2 \rfloor$還要小的話(如圖 21 所說的狀況)，$mul_n$參數就要+1，如方程式(3.13)。此時$mul_n$才算是正式計算完成。

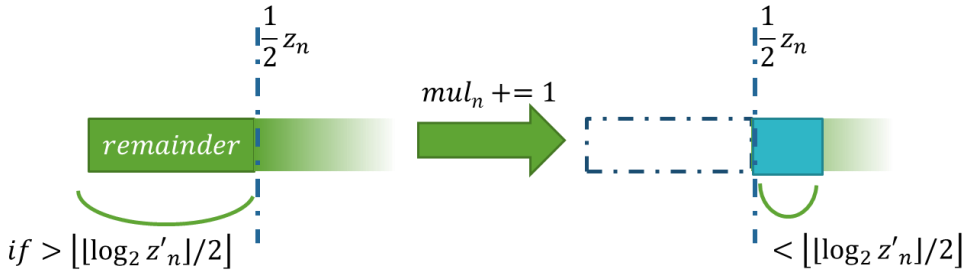


圖 21 餘數微調方法示意圖

$$mul_n = mul_n + \begin{cases} 1 & ,\text{if } remainder > \left\lfloor \frac{\lfloor \log_2 z'_n \rfloor}{2} \right\rfloor \\ 0 & ,\text{otherwise} \end{cases} \tag{3.13}$$

6. 計算完乘數參數mul_n之後，再使用(3.9)配合mul_n參數降低差值，並使用(3.3)計算位置地圖$LM3_n$即可。

另外，mul_n參數也需要透過位置地圖的方式儲存。在一個像素組中，需要儲存 3 個乘積參數，每一個參數對應一個差值(即$\boldsymbol{mul} = (mul_0, mul_1, mul_2)$)。等第二降低差值階段完成之後，$\boldsymbol{mul}$參數也會連同其他位置地圖參數一樣儲存於位置地圖內。還有，在執行 Improved Two-Steps-RDE of Quads 的第二降低差值階段(圖 20 上的 2nd layer)步驟之前，也要跟之前我們講過的 Fixed Enhance-RDE of Quads 一樣，使用(3.6)將差值轉成正數，待第二階段做完後再用(3.7)將差值轉回原本的正數或負數。

在解嵌入演算法也是同樣套用 Fixed Enhance-RDE of Quads 的恢復降低差值演算法(即(3.4)→(3.5))，但其中一個方程式(3.4)因應第二降低差值階段的方式的修改而被替換成(3.14)，這個需要搭配從第二降低差值階段計算而來的位置地圖參數$\boldsymbol{LM3}$跟$\boldsymbol{mul}$。

$$z'_n = z''_n + (\lfloor \log_2 z''_n \rfloor + LM3_n) \times mul_n \tag{3.14}$$

為證明該改良過後的兩階式降低差值演算法的成效，在做正式實驗之前，我們做了一個廣域測試，計算在各種可能差值的場合上($z = 1 \sim 255, -1 \sim -255$)，

比較 Fixed Enhance-RDE of Quads 跟 Improved Two-Steps-RDE of Quads 套用各自的 expandable 的嵌入演算法之後，差值的變更幅度有多少(亦即原差值z_n減掉新差值z'''_n後的絕對差距)：

圖 22 Fixed Enhance-RDE of Quads 跟 Improved Two-Steps-RDE of Quads 的差值變更幅度比較($z = 1 \sim 255$，$b = 1$)

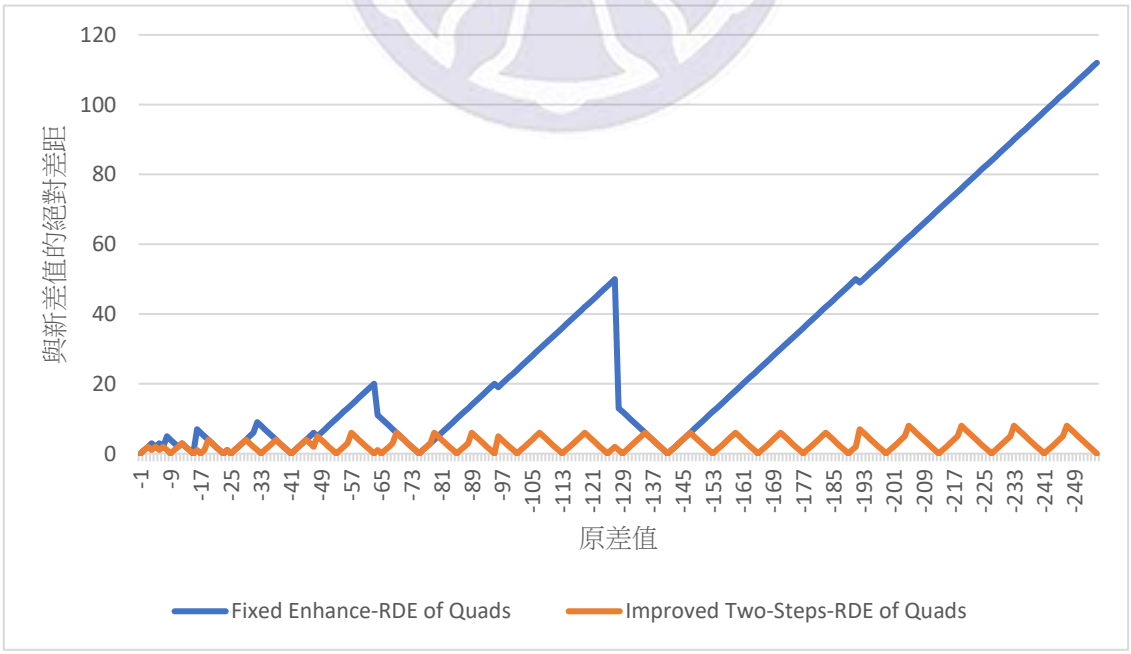


圖 23 Fixed Enhance-RDE of Quads 跟 Improved Two-Steps-RDE of Quads 的差值變更幅度比較($z = -1 \sim -255$，$b = 1$)

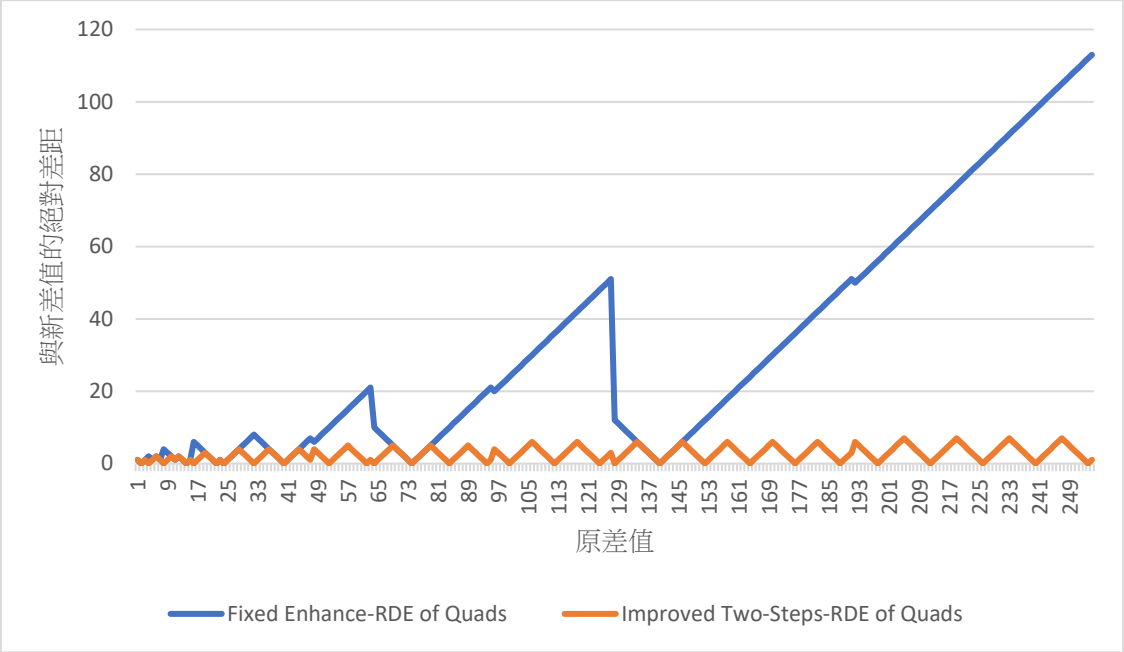


圖 24 Fixed Enhance-RDE of Quads 跟 Improved Two-Steps-RDE of Quads 的差值變更幅度比較($z = 1\sim255$，$b = 0$)

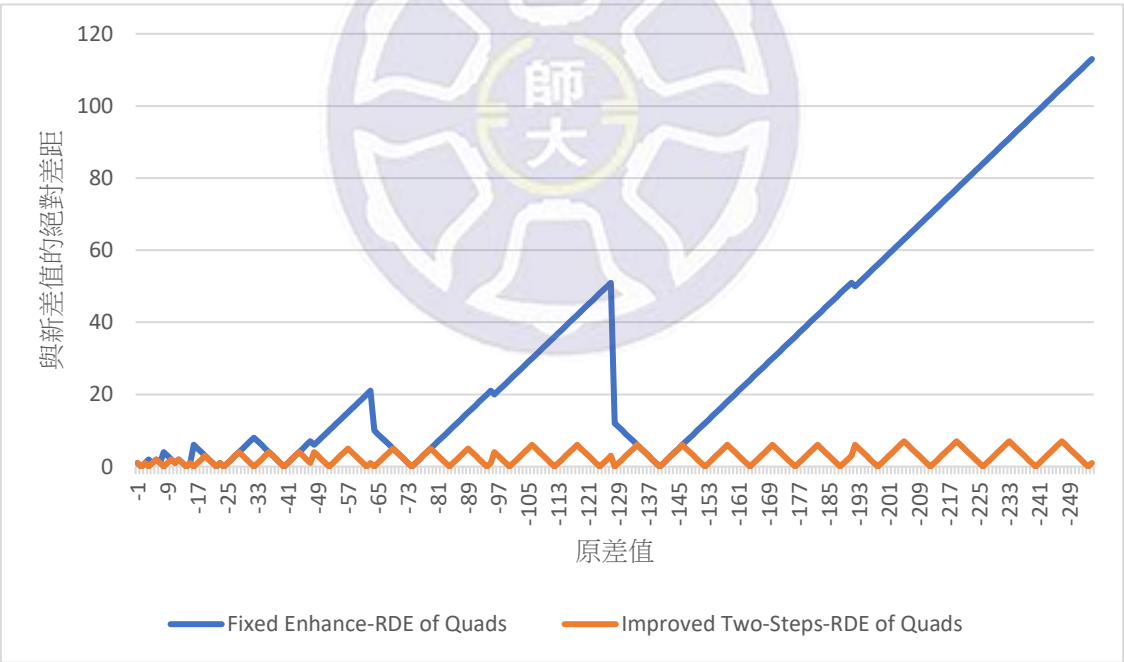


圖 25 Fixed Enhance-RDE of Quads 跟 Improved Two-Steps-RDE of Quads 的差值變更幅度比較($z = -1\sim-255$，$b = 0$)

從以上圖 22～圖 25 的測試結果可以得知：使用 Fixed Enhance-RDE of Quads 的 expandable 嵌入方法，新差值跟原差值的最大差距雖然是比原差值一半還要減少一些，但仍然接近原差值的一半。尤其是當原差值越大時，新差值所造

成的變動幅度就越大。反之若使用 Improved Two-Steps-RDE of Quads 的 expandable 嵌入方法，搭配在此章節所介紹的改良降低差值演算法，在我們限制的原差值範圍內，其新差值跟原差值的最大差距只介於十位數以下。就結果的成效來說：在加入改良版的降低差值法之後，Improved Two-Steps-RDE of Quads 的 expandable 嵌入方法比 Fixed Enhance-RDE of Quads 的更能有效縮減原差值與新差值之間的變化量，其變化量約可以縮減至幾倍甚至到十幾倍的程度，這證明了改良演算法更能有效減少新差值與原差值之間的差異擴大程度。透過這次證明，我們可以確定我們的方法能有效達成差值的收斂，使新差值更接近原差值，進而降低輸出圖片像素值的失真程度。

第三節　交換像素值以降低原差值

在改良雙階段降低差值演算法過後，我們從圖 22～圖 25 發現 Improved Two-Steps-RDE of Quads 仍然繼續保留若原差值絕對大小增加，新差值幅度也會跟著增加的特性。我們另外從趨勢線的角度來看(如圖 26)，可以得知差值變更幅度會隨著原差值減少而變小的趨勢：

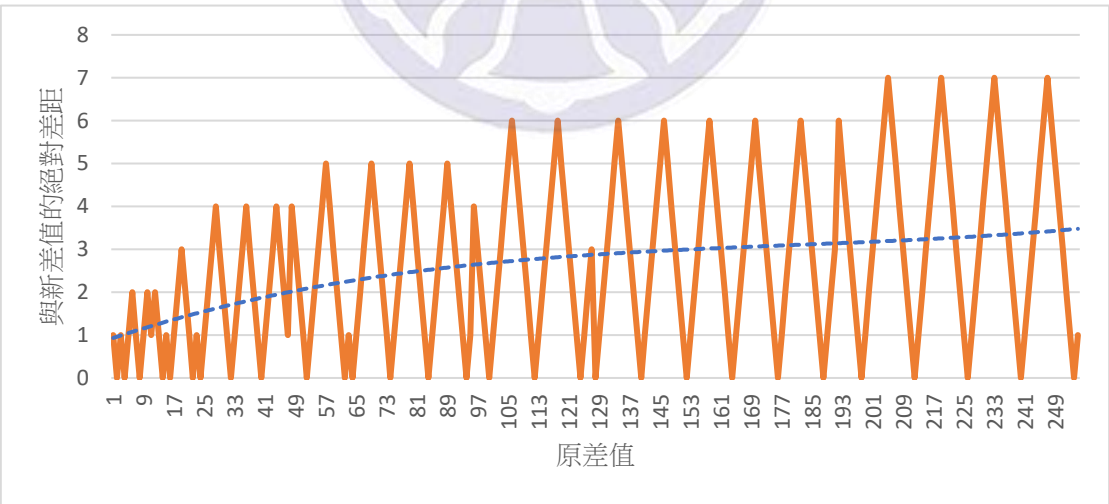


圖 26 Improved Two-Steps-RDE of Quads 差值變更幅度及其趨勢線的走向 ($z = 1$~255，$b = 0$，橘色實線為實際的差值變更幅度、藍色虛線為趨勢線走向)

在 Fixed Enhance-RDE of Quads 的製造差值的方程式(2.24) (也同時是原本 Enhance-RDE of Quads 的製造差值方程式)上，求取差值皆由像素組中最後一個像

素(k_3)為準，所求到的差都是k_3跟其他像素之間的差值，所以我們特地提出了「交換像素值」這個優化機制。

設計此機制為的就是在計算差值前先將兩個像素互換在像素組中的位置，這樣在為像素組做一次計算差值的動作後，該像素組的差值平均會比沒有加入交換像素值機制的還要來得更小。同時因為從圖 26 知道原差值越小、新差值變更幅度也會跟著變小的原理，只要藉由這個原理，加上交換像素值的機制，可以降低大部分的原差值，來減少輸出後新差值的整體平均，進而達成減少圖片的失真程度。

我們設定一個主導像素(dominant pixel)機制，此機制是類似在一像素組中找一個像素代替(2.24)裡的k_3來做k_3的工作。找到的主導像素必須跟其它像素值之間的差值和達到最小，如圖 27 所表示的情況。

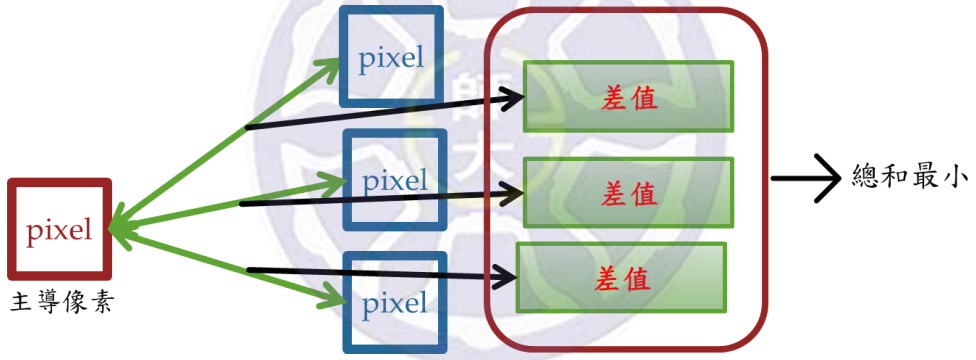


圖 27 交換像素值方法原理

研究發現如圖 28，在理想情況下，假使主導像素越處於中間值，也就是在該像素值最接近像素組中最大像素值(Max)及最小像素值(Min)其兩個數值之間的中間值的時候，所有求出來的差值和將會達到最小。這樣的話，使用 Improved Two-Steps-RDE of Quads 的嵌入演算法後，新差值的平均變更幅度將會被再度降低。

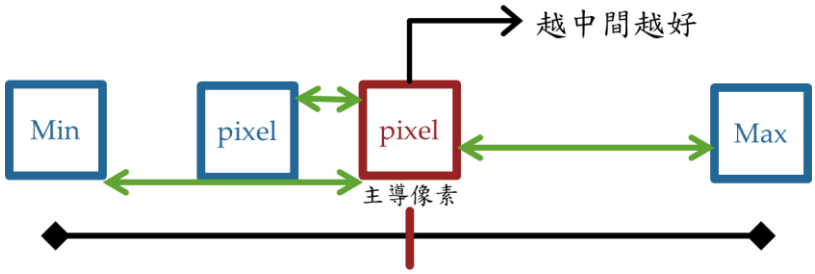


圖 28 主導像素的理想狀況

在此演算法，我們設定讓演算法來尋找最靠近該像素組中間值的像素，作為我們需要的主導像素。在嵌入演算法中，首先我們定義距離平方和公式，計算每個像素的距離參數($distance_n$)：

$$distance_n = (Min - k_n)^2 + (Max - k_n)^2$$
$$\text{where } Min = \min(k_0, k_1, k_2, k_3), Max = \max(k_0, k_1, k_2, k_3) \tag{3.15}$$

有最小距離($distance_n$)的像素將被定義為主導像素，該主導像素的編號將會被另外記錄成參數($dPixel$)，如下列方程式：

$$dPixel = \underset{n=[0,3]}{\operatorname{argmin}}(distance_n) \tag{3.16}$$

接下來在計算差值開始之前，先把主導像素的值(k_{dPixel})跟最後一個像素的值(k_3)做交換，等嵌入資料並算回至像素值之後再交換回來，並同時將主導像素的編號($dPixel$)儲存在位置地圖上。此時一個像素組中整個嵌入演算法結構將會變成如圖 29 所示：

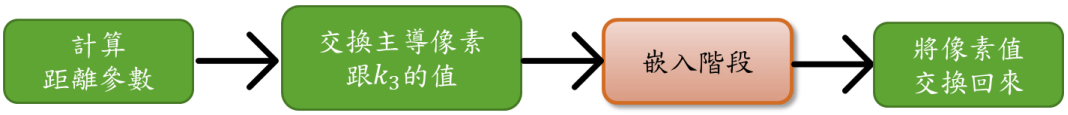


圖 29 交換像素值流程圖(嵌入側)

在解嵌入演算法也同樣，先將原本用(3.15)(3.16)所算出來的主導像素編號($dPixel$)從位置地圖取回，並在計算差值前，先把主導像素的值跟最後一個像素的值做交換，等取出資料、將差值回復完成並算回至像素值之後再交換回來。如圖 30 所示：

圖 30 交換像素值流程圖(解嵌入側)

第四節 其他細節

我們所提出的改良演算法－Improved Two-Steps-RDE of Quads，基本上除了增加上述我們所講過的兩個改正方案之外，整體演算法架構直接使用了跟 Fixed Enhance-RDE of Quads 一樣的演算法架構。在像素組的配置，我們依舊保持 Enhance-RDE of Quads 的 4×1 像素為一組的方式。同時計算差值的演算法依舊採用 Enhance-RDE of Quads 的方式。

像素組嵌入模式仍舊採用跟 Fixed Enhance-RDE of Quads 一樣的判定順序(即圖 16)，但仍舊使用步驟 2 原本的判定條件(若$z_n = 0$，直接跳至 changeable)則是考慮到若在差值z_n為 0 的情況下使用 expandable 嵌入模式，會導致$\log_2 z_n$變成$-\infty$，此時再使用 Improved Two-Steps-RDE of Quads 的降低差值演算法就沒意義可言了，所以繼續保留步驟 2 的條件以預防降低差值時有錯誤發生。而像素組嵌入模式判定方式也同樣採用 Fixed Enhance-RDE of Quads 所使用的圖 17 那樣，邊計算新差值邊檢測像素是否會溢位的手法。

在 Improved Two-Steps-RDE of Quads 所用到的位置地圖參數總共有七種，各嵌入模式如表 4 所示：

表 4 Improved Two-Steps-RDE of Quads 各位置地圖參數功能定義

參數	容量 (每個像素組)	功能
$dPixel$	2 bits	紀錄主導像素的編號
$CH, LM1$	各1 bit	紀錄像素組的嵌入模式，標籤定義請參考表 1。
$\boldsymbol{LM2}$	1 bit × 3	紀錄 expandable 模式第一階段用位置地圖(LM)參數
$\boldsymbol{LM3}$	1 bit × 3	紀錄 expandable 模式第二階段用位置地圖(LM)參數

$\boldsymbol{mul}$	4 bits × 3	紀錄 expandable 模式第二階段中的乘積(mul)參數
$\boldsymbol{BS}$	1 bit × 3	紀錄 changeable 模式時各差值的 LSB

其中CH跟$LM1$同樣使用了表 1 所標示的標籤定義參考。$dPixel$因為是從一個像素組內的四個像素中決定哪一個可以作為主導像素，所以僅需$\sqrt{4} = 2$ bits 即可記錄。mul_n參數則是因為在最大差值(255或−255)之下仍不會超過十位數，所以$\boldsymbol{mul}$最少只要用 4 bits × 3，以自然數的方式紀錄即可。

當然，在 Improved Two-Steps-RDE of Quads 上，各個嵌入模式不一定用到所有位置地圖參數。在這裡，我們整理出了每個嵌入模式所用到的位置地圖參數，以及其所需要的儲存空間，如表 5：

表 5 Improved Two-Steps-RDE of Quads

各嵌入模式所需位置地圖參數及使用空間

參數名稱 / 嵌入模式	$dPixel$	CH	$LM1$	$\boldsymbol{LM2}$	$\boldsymbol{LM3}$	$\boldsymbol{mul}$	$\boldsymbol{BS}$	所需容量
Expandable	✔	✔	✔	✔	✔	✔		22 bits
Changeable	✔	✔	✔				✔	7 bits
Non-changeable		✔	✔					2 bits

其中 expandable 因為加入了$dPixel$跟$\boldsymbol{mul}$參數的關係，所需位置地圖容量會被增大，但因為有了這些參數，才可以增加差值的收斂度，使像素值的變質程度不會太大。因此對我們而言，在位置地圖上多加一點容量對我們的論文是有幫助的。

而$dPixel$只有在像素組有被更動的時候才用到。在像素組為 Non-changeable 模式時，不會用上主導像素，所以在嵌入階段前後的交換像素步驟將不會被用到，同時$dPixel$參數也不會被製造出來。解嵌入演算法也一樣，像素組為 Non-changeable 時，將忽略交換像素以及還原階段等動作，直接切換至下一個像素組。

另外，Improved Two-Steps-RDE of Quads 有套用在 DE 演算法就提過的「多層嵌入」的機制，可以透過反覆儲存圖片像素的方式來塞入檔案較大的隱藏資料。為了檢測 Improved Two-Steps-RDE of Quads 的容量特性，在正式實驗上，當演算法每嵌入一層資料時，將會計算該層所用到的資料容量。在此同時，上面第一節所提出的修正演算法 Fixed Enhance-RDE of Quads 也會加上「多層嵌入」的機制，並跟 Improved Two-Steps-RDE of Quads 一樣在正式實驗上逐層檢測每層的資料容量。實驗結果將會顯示 Fixed Enhance-RDE of Quads 跟 Improved Two-Steps-RDE of Quads 雙方的資料容量檢測數據以供比較。

總和我們在改良演算法加入的兩個改良方案、上述講過的其他細節，以及作為演算法基底的 Fixed Enhance-RDE of Quads，我們可以整理出 Improved Two-Steps-RDE of Quads 演算法的構造。

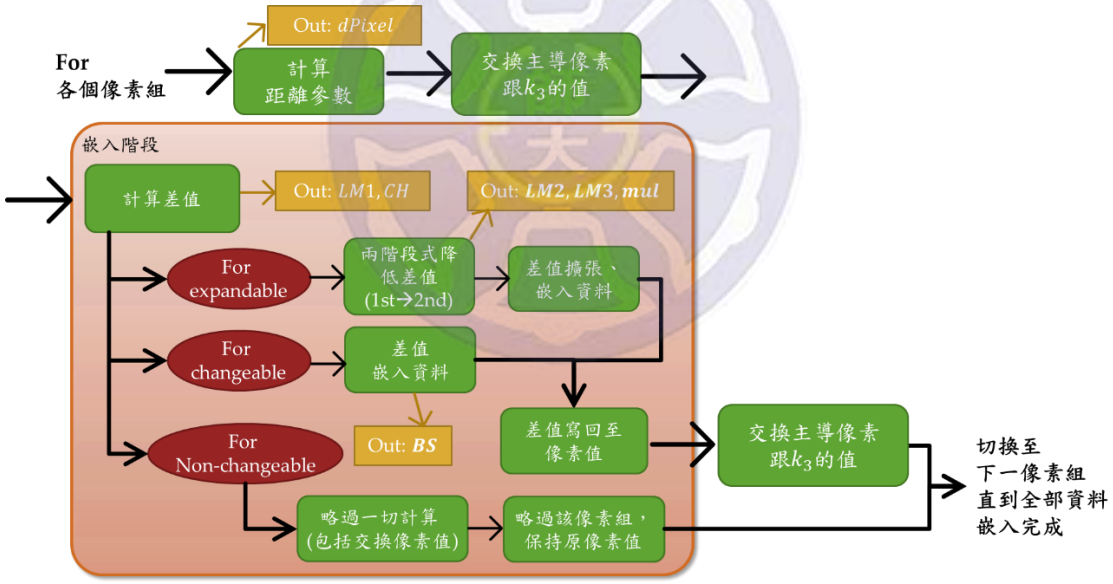


圖 31 Improved Two-Steps-RDE of Quads 嵌入資料側構造

在嵌入資料側，我們的方法會將一張灰階圖片分成若干組4×1的像素組，並將每個像素組都做出如圖 31 嵌入資料的動作，全部像素組嵌入後若資還有剩，就再用此圖片繼續嵌入，直到所有資料嵌入為止。其中「兩階段式降低差值」即是使用 Improved Two-Steps-RDE of Quads 所使用的改良版兩階段式降低差值方法。在整個嵌入資料側構造上，雖然在嵌入階段的前後都加入了交換像素值的動

作，但若是在檢測到該像素組是 Non-changeable 的情況下，除了CH跟$LM1$之外，原本在嵌入階段所計算的所有資訊連同交換像素值所用到的$dPixel$參數將會被全部忽略，不儲存任何資料，同時保有該像素組原本的像素值。

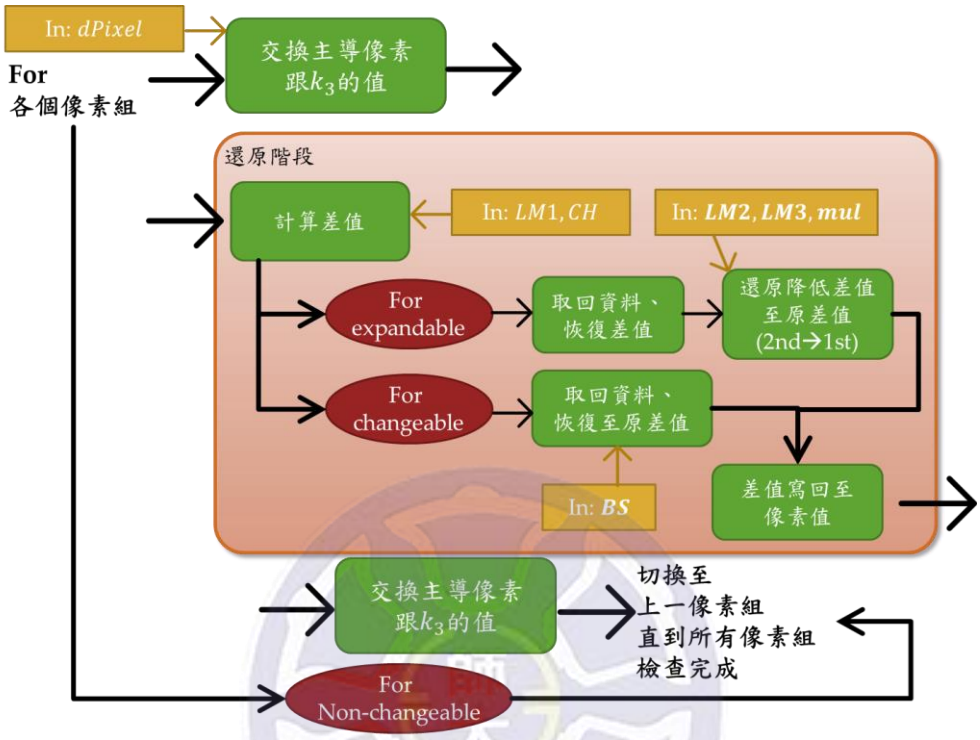


圖 32 Improved Two-Steps-RDE of Quads 提取資料側構造

提取資料側的作法是採用嵌入資料側完全反方向的步驟。同樣先把一張已嵌入資料的圖片分成若干組4×1的像素組，並從上一次儲存的像素組位置開始如圖 32 做提取資料的動作，若還沒取完所有隱藏資料就再拿此圖片從最後一個像素組開始檢查，直到檢查過所有像素組為止。其中「還原降低差值至原差值」即是使用 Improved Two-Steps-RDE of Quads 所使用的改良版兩階段式降低差值修正方案中的還原差值方式，即(3.14)跟(3.5)。在 Improved Two-Steps-RDE of Quads 提取資料側也同樣，當該像素組為 Non-changeable 時，將直接略過該像素組，跳到上一個像素組繼續。

第四章　實驗結果

在本論文我們原本打算使用我們所提出的改良演算法－Improved Two-Steps-RDE of Quads 與前人的方法－Enhance-RDE of Quads [14]做實驗，並在輸出圖片的品質以及容量上做比較。不過在第二章第八節有提到 Enhance-RDE of Quads 因為某些方程式寫法上的問題，有時候沒有辦法將建好的新圖像完全轉換成原圖像。所以我們將使用先前在第三章第一節我們針對 Enhance-RDE of Quads 進行修改、功能也相近的修正演算法－Fixed Enhance-RDE of Quads 來代替前人的方法進行實驗。

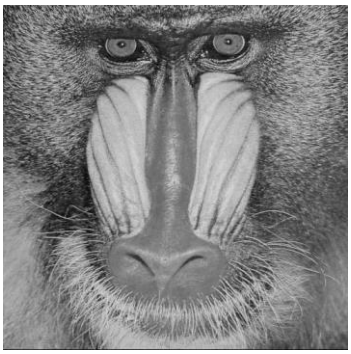
本章的實驗將以 Improved Two-Steps-RDE of Quads 以及 Fixed Enhance-RDE of Quads 這兩個演算法來做實驗(圖表上顯示的結果分別以 ITS-RDE 跟 FE-RDE 來區分)，其目標是要比較使用該演算法輸出圖片的品質以及容量狀況為何。我們將執行兩個不同的實驗：

1. 在只嵌入一層的情況下，兩個演算法的圖片品質為何？
2. 在多層嵌入的情況下，兩個演算法輸出的圖片在各層的品質及容量為何？

在執行實驗的時候，也同時記錄輸出圖片每一層的每種像素組嵌入模式各使用多少次，以及每一層所使用的位置地圖容量有多大等其他統計資料。

第一節　實驗環境

在演算法實作上，我們使用搭載 Windows 10 作業系統的筆記型電腦作為實作演算法程式碼的環境，編寫我們要做 Improved Two-Steps-RDE of Quads 及 Fixed Enhance-RDE of Quads 雙方的程式碼，之後編譯成程式檔並運行比較兩個演算法程式實際輸出圖片的狀況。程式開發是使用 Code::Blocks 16.01 軟體，並外部嵌入 OpenCV 函式庫用來讀取及儲存圖片。正式實驗我們將會用到下面 6 張大小都是512×512的灰階圖片作為實驗用載體圖片，如圖 33：



(a) baboon

(b) car

(c) elaine

(d) flower

(e) fruits

(f) lena

圖 33 實驗用載體圖片

我們使用了 BMP (Bit map)圖片格式作實驗。BMP 在儲存圖片時因為沒有壓縮而變質的特性，所以可以精確地儲存圖片上每一個像素值，使用 BMP 不用考慮圖片儲存後會有像素值變質，並導致提取的資料不正確的問題。

在品質的探測上，直接使用人眼偵測難免會因主觀認知上的不同而使得檢測的結果有所差異。如圖 34 及圖 35，以 baboon 跟 elaine 為例，從原圖輸出的新圖像乍看之下跟原圖外觀幾乎沒什麼差別，除非把圖片放大並仔細看像素狀況，否則就連憑肉眼也很難看出實驗用的兩個演算法所輸出的圖片品質到底互相差在哪裡。

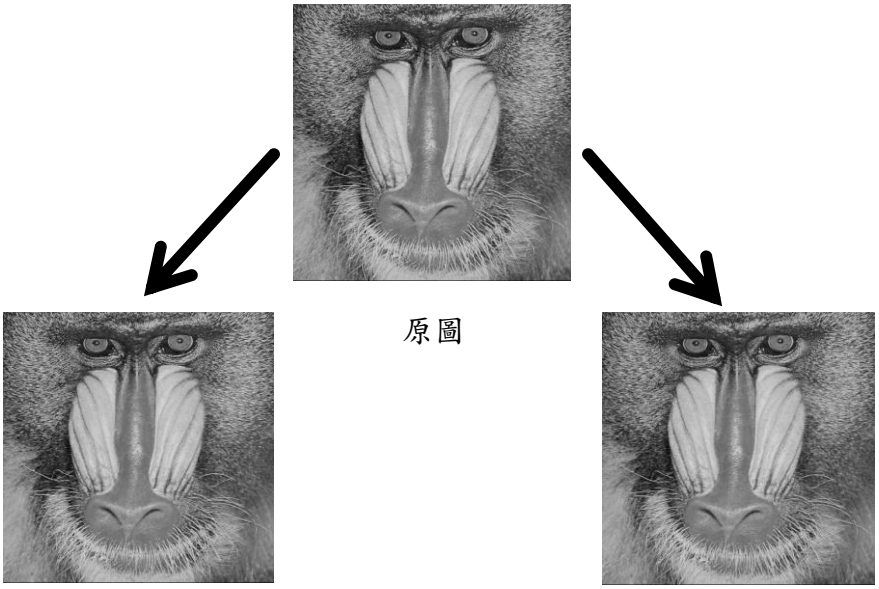


圖 34 原圖與雙方演算法輸出圖片比較(以 baboon 為例)

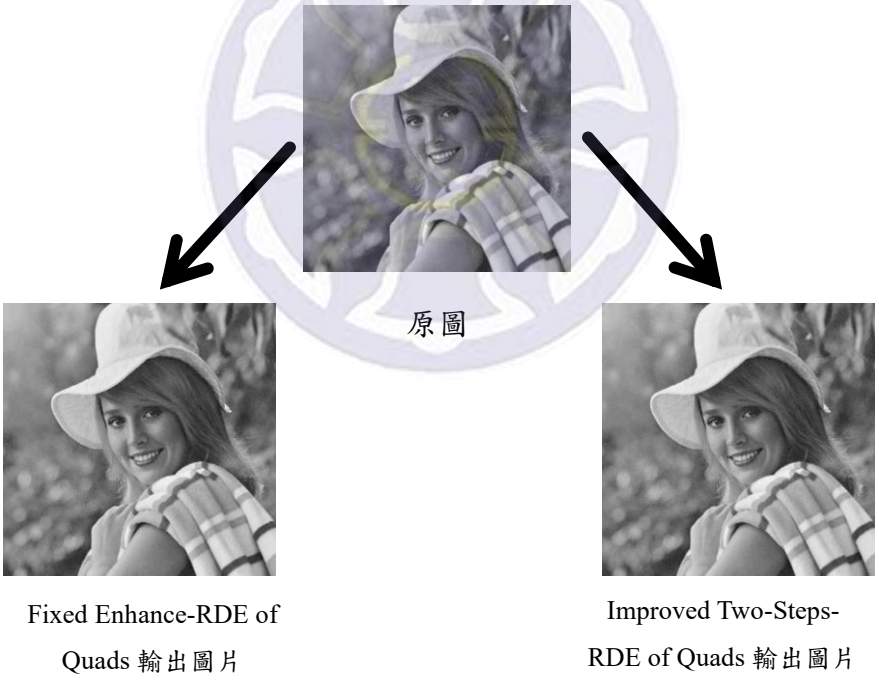


圖 35 原圖與雙方演算法輸出圖片比較(以 elaine 為例)

所以在這實驗上，我們使用了峰值信噪比(Peak signal-to-noise ratio，簡稱 PSNR)公式作為判定圖片品質的標準，以利檢查輸出圖片實質上的品質有多少：

$$PSNR = 10\log_{10}\frac{(255)^2}{MSE} \tag{4.1}$$

$$MSE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} [A(i,j) - A'(i,j)]^2$$

其中 *MSE* 指的是平方誤差(Mean Square Error)，$A(i,j)$跟$A'(i,j)$分別代表原圖片跟嵌入資料過後新圖片的像素值，H 跟 W 則代表圖片的長度跟寬度。PSNR 通常是用來檢測圖片經改動後，跟原圖片的實質相似程度。若新圖像越接近原圖像的外觀，PSNR 將會越高。

第二節　嵌入一層資料的比較

在該實驗中，我們隨機使用了一個檔案作為該實驗用的隱藏資料，套用在實驗用的兩個演算法上做計算，分別在六張圖片上做 PSNR、嵌入模式個數以及位置地圖容量大小做比較。該資料大小為 22,985 位元組(相當於 183,880 bits)，實驗結果將如表 6 所示。

另外，在只有嵌入一層隱藏資料的情況下，所有實驗(包括圖 33 所使用的所有載體圖片及實驗使用的兩個演算法)場合寫進去的 bits 都是 183,880 bits，單一像素儲存資料平均嵌入密度都為 0.701447 bpp，因此對嵌入容量將不進行比較。

表 6 嵌入一層資料的場合上，兩個方法製造出來的圖片各種數值上的比較

圖片載體	使用演算法	PSNR (dB)	各嵌入模式使用個數			位置地圖容量 (bits)
			Non-changeable	Changeable	Expandable	
baboon	FE-RDE	33.2255	0	5054	56240	475190
	ITS-RDE	44.7759	0	5651	55643	1263703
car	FE-RDE	38.2289	278	21564	39730	426216
	ITS-RDE	47.7	355	24798	36496	977208
elaine	FE-RDE	41.6003	0	11039	50255	457235
	ITS-RDE	47.9416	0	12544	48750	1160308
flower	FE-RDE	39.9029	0	11913	49381	454613
	ITS-RDE	47.8142	0	14511	46783	1130803
fruits	FE-RDE	38.562	0	17884	43410	436700
	ITS-RDE	47.3724	0	19481	41813	1056253
lena	FE-RDE	41.3439	0	13880	47414	448712
	ITS-RDE	48.2524	0	15759	45535	1112083

根據表 6 上的實驗結果可得知：Fixed Enhance-RDE of Quads 製造的圖片其 PSNR 能夠到約 30~40dB 的程度，但 Improved Two-Steps-RDE of Quads 卻可以普遍到達 40dB 以上，而且比 Fixed Enhance-RDE of Quads 還要好上約 1.2~1.3 倍的程度，這顯示我們的改良演算法能有效提高輸出圖片的品質。

但就嵌入模式個數的統計上，似乎不符合我們的理想。雖然大部分的狀況下，嵌入資料都會用到全部的像素組，但在 Expandable 嵌入模式的個數上，Improved Two-Steps-RDE of Quads 相比 Fixed Enhance-RDE of Quads 就少用了一些。我們最希望的就是：經過我們的改良演算法再度降低差值後，可以增加 Expandable 的使用機會，阻止更多的像素組因使用擴張差值後差值過大而使得新像素變異更多甚至溢位、導致資料無法嵌入的狀況。但沒想到就表 6 嵌入一層資料的實驗結果來看，反而帶來了反效果。

另外，在位置地圖容量方面，Improved Two-Steps-RDE of Quads 要儲存的位置地圖容量大小至少需要比 Fixed Enhance-RDE of Quads 還要儲存約 2~3 倍大的位置地圖數據。

第三節　嵌入多層資料的比較

在該實驗，我們做了另外一項實驗測試，改成使用嵌入多層資料的方式，同樣分別在六張圖片上檢測並計算在圖片每次嵌入一層隱藏資料時，其目前圖片品質(PSNR)，以及該層可儲存的資料容量、嵌入模式個數跟位置地圖容量大小等諸多數據。我們使用比上一次實驗用的隱藏資料還要大的隨機檔案做為要隱藏的資料，該檔案大小為 165,765 位元組(相當於 1,326,120 bits)，其可以使所有實驗場合下都能讓整個隱藏資料在一張圖片上分成 7 層儲存在圖片像素的差值內。使用各種載體圖片做成的結果如以下圖表所示(要記住：每一層嵌入動作都是由前一層所動過的圖片拿來作為載體繼續嵌入資料，其 PSNR 數值會呈現遞減現象。而其餘數值則是檢測從該層所計算出來的數值，而非全部層所統計出來的。另外，因為在最後一層僅儲存所剩不多的隱藏資料，所以不會檢查到所有像素組，最後一

層的嵌入資料容量相對於其他層的還要少，同時該層的嵌入模式個數以及位置地圖容量也比其他層的少，這都屬於正常現象)：

(a) baboon：

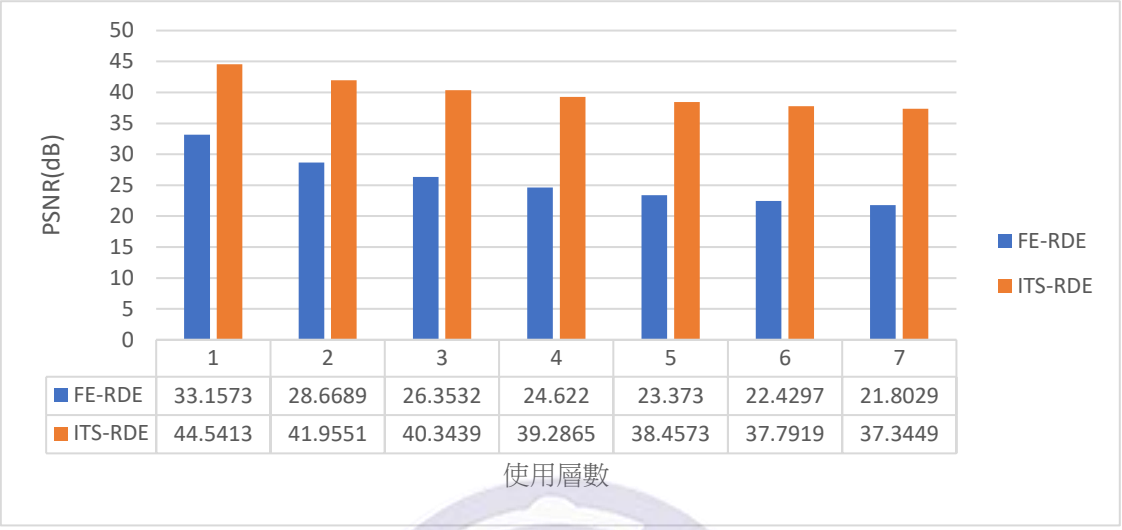


圖 36 以 baboon 為例，嵌入多層資料時 PSNR 的比較

表 7 以 baboon 為例，嵌入多層資料時各層各數值的比較

層級	使用演算法	該層資料容量 (bits, bpp)	各嵌入模式使用個數			位置地圖容量(bits)
			Non-changeable	Changeable	Expandable	
1	FE-RDE	196608 (0.750000bpp)	0	5792	59744	506912
	ITS-RDE	196602 (0.749977bpp)	2	6105	59429	1350177
2	FE-RDE	196587 (0.749920bpp)	7	4695	60834	510161
	ITS-RDE	196599 (0.749966bpp)	3	4123	61410	1379887
3	FE-RDE	196542 (0.749748bpp)	22	5119	60395	508799
	ITS-RDE	196599 (0.749966bpp)	3	2884	62649	1398472
4	FE-RDE	196452 (0.749405bpp)	52	6529	58955	504389
	ITS-RDE	196605 (0.749989bpp)	1	2175	63360	1409147
5	FE-RDE	196314 (0.748878bpp)	98	8342	57096	498674

	ITS-RDE	196599 (0.749966bpp)	3	1754	63779	1415422
6	FE-RDE	196182 (0.748375bpp)	142	10297	55097	492545
	ITS-RDE	196596 (0.749954bpp)	4	1440	64092	1420112
7	FE-RDE	147435 (0.562420bpp)	167	10135	39010	363089
	ITS-RDE	146520 (0.558929bpp)	0	821	48019	1062165

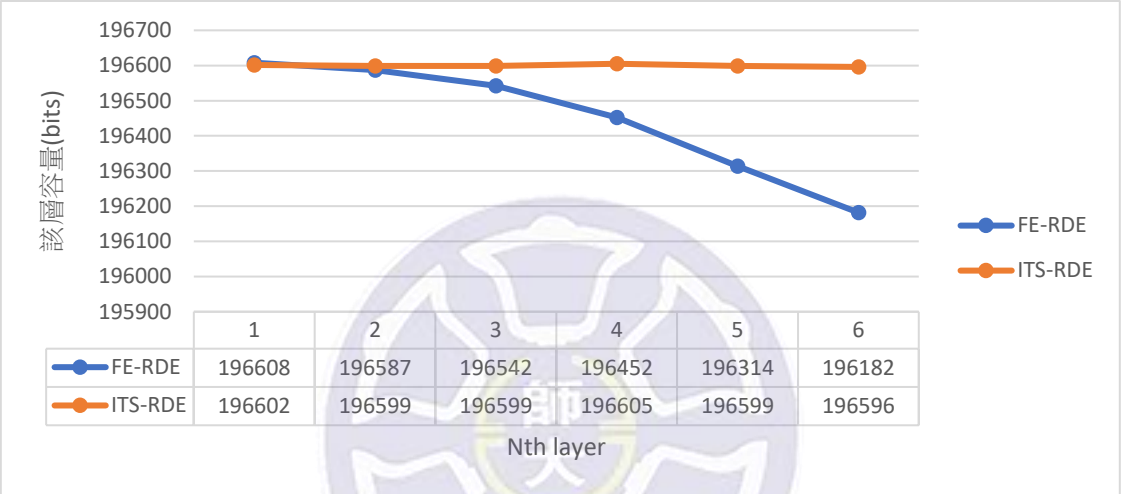


圖 37 以 baboon 為例，嵌入多層資料時各層資料容量比較(最後一層除外)

(b) car：

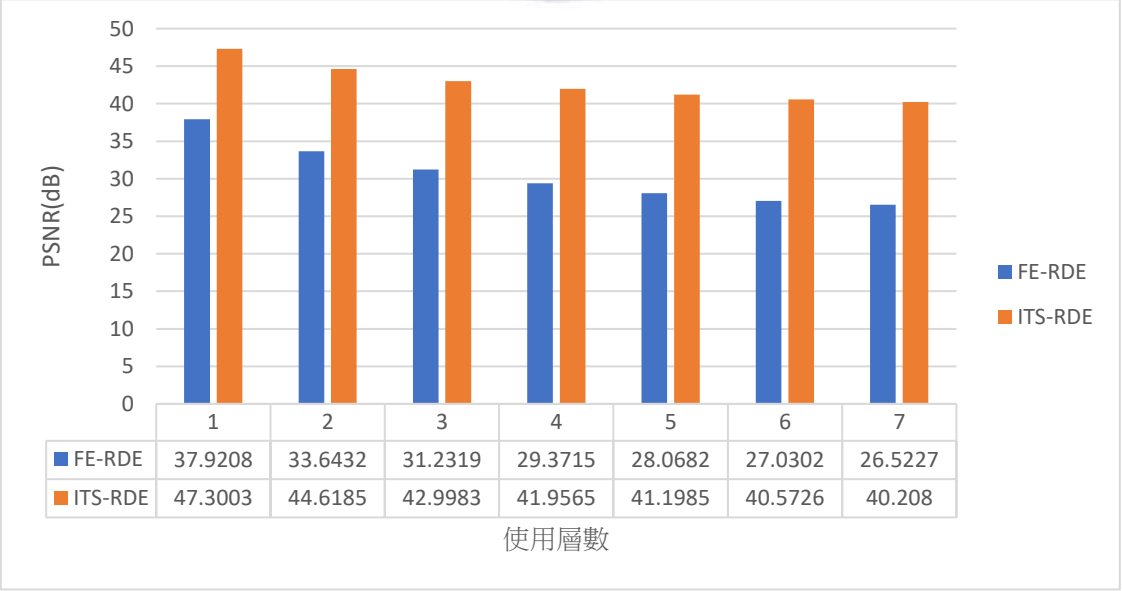
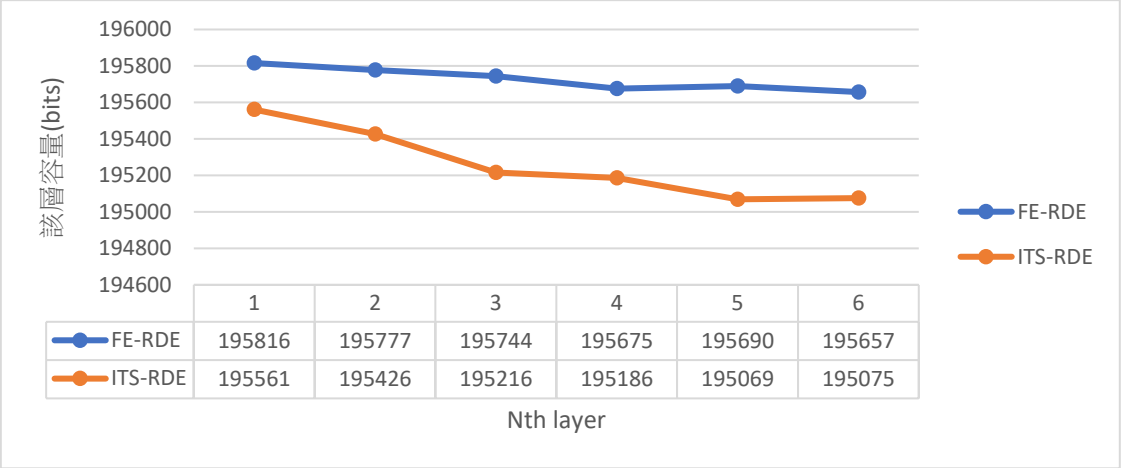


圖 38 以 car 為例，嵌入多層資料時 PSNR 的比較

表 8 以 car 為例，嵌入多層資料時各層各數值的比較

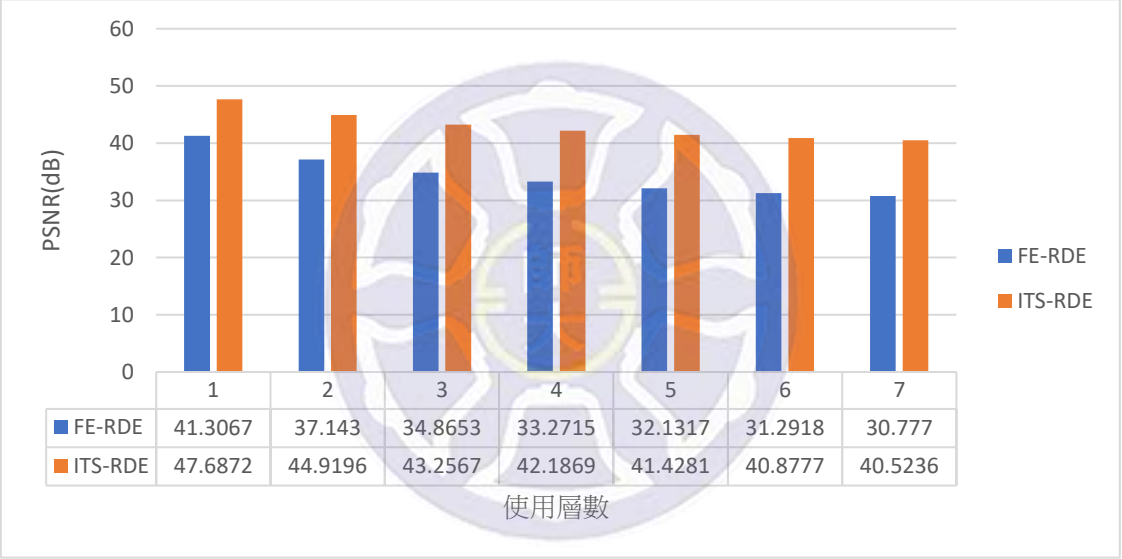
層級	使用演算法	該層資料容量 (bits, bpp)	各嵌入模式使用個數			位置地圖容量(bits)
			Non-changeable	Changeable	Expandable	
1	FE-RDE	195816 (0.746979bpp)	264	21939	43333	456887
	ITS-RDE	195561 (0.746006bpp)	349	25164	40023	1057352
2	FE-RDE	195777 (0.746830bpp)	277	16996	48263	471638
	ITS-RDE	195426 (0.745491bpp)	394	19100	46042	1147412
3	FE-RDE	195744 (0.746704bpp)	288	13674	51574	481538
	ITS-RDE	195216 (0.744690bpp)	464	14832	50240	1210032
4	FE-RDE	195675 (0.746441bpp)	311	11517	53708	487871
	ITS-RDE	195186 (0.744576bpp)	474	11961	53101	1252897
5	FE-RDE	195690 (0.746498bpp)	306	10234	54996	491750
	ITS-RDE	195069 (0.744129bpp)	513	9874	55149	1283422
6	FE-RDE	195657 (0.746372bpp)	317	9309	55910	494459
	ITS-RDE	195075 (0.744152bpp)	511	8374	56651	1305962
7	FE-RDE	151761 (0.578922bpp)	329	7778	42809	382020
	ITS-RDE	154587 (0.589703bpp)	499	7041	44488	1029021



	1	2	3	4	5	6
FE-RDE	195816	195777	195744	195675	195690	195657
ITS-RDE	195561	195426	195216	195186	195069	195075

圖 39 以 car 為例，嵌入多層資料時各層資料容量比較(最後一層除外)

(c) elaine：



	1	2	3	4	5	6	7
FE-RDE	41.3067	37.143	34.8653	33.2715	32.1317	31.2918	30.777
ITS-RDE	47.6872	44.9196	43.2567	42.1869	41.4281	40.8777	40.5236

圖 40 以 elaine 為例，嵌入多層資料時 PSNR 的比較

表 9 以 elaine 為例，嵌入多層資料時各層各數值的比較

層級	使用演算法	該層資料容量 (bits, bpp)	各嵌入模式使用個數			位置地圖容量(bits)
			Non-changeable	Changeable	Expandable	
1	FE-RDE	196608 (0.750000bpp)	0	11814	53722	488846
	ITS-RDE	196608 (0.750000bpp)	0	13489	52047	1239457
2	FE-RDE	196602 (0.749977bpp)	2	7481	58053	501833
	ITS-RDE	196608 (0.750000bpp)	0	9409	56127	1300657

3	FE-RDE	196605 (0.749989bpp)	1	5018	60517	509228
	ITS-RDE	196608 (0.750000bpp)	0	6750	58786	1340542
4	FE-RDE	196599 (0.749966bpp)	3	3652	61881	513314
	ITS-RDE	196608 (0.750000bpp)	0	5316	60220	1362052
5	FE-RDE	196593 (0.749943bpp)	5	2893	62638	515579
	ITS-RDE	196608 (0.750000bpp)	0	4341	61195	1376677
6	FE-RDE	196581 (0.749897bpp)	9	2523	63004	516665
	ITS-RDE	196608 (0.750000bpp)	0	3621	61915	1387477
7	FE-RDE	146532 (0.558975bpp)	6	1699	47145	385667
	ITS-RDE	146472 (0.558746bpp)	0	2314	46510	1039418

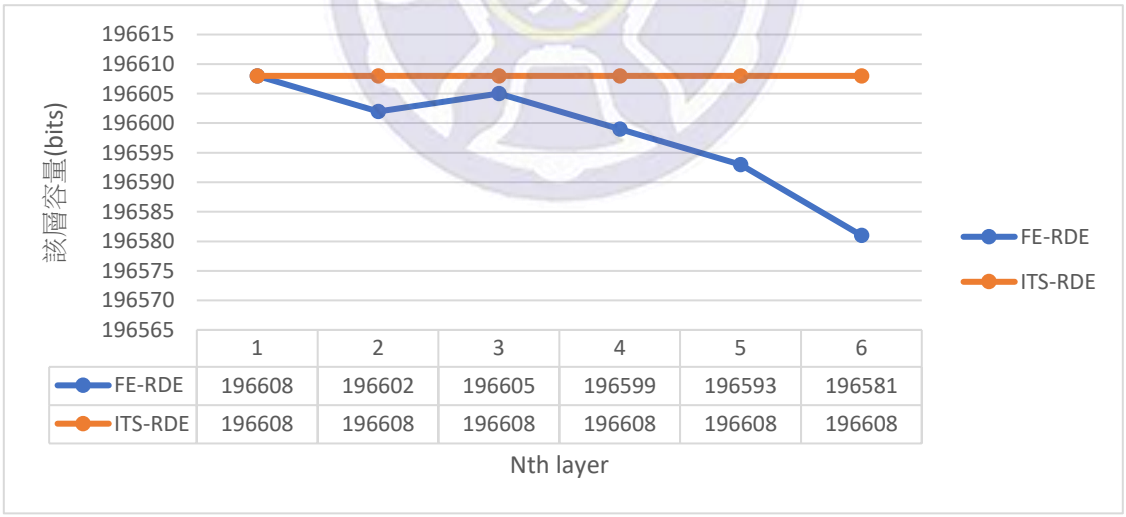


圖 41 以 elaine 為例，嵌入多層資料時各層資料容量比較(最後一層除外)

(d) flower：

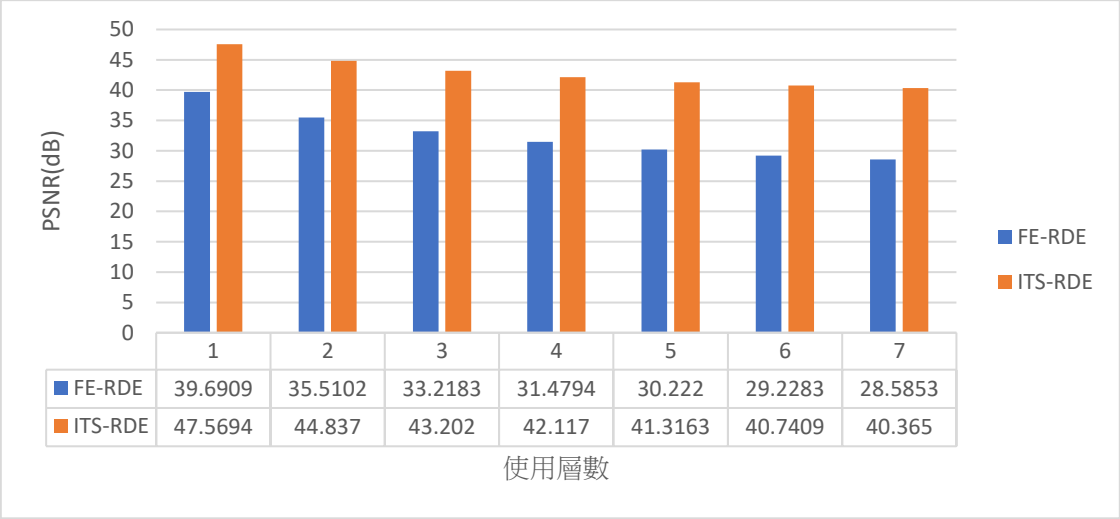


圖 42 以 flower 為例，嵌入多層資料時 PSNR 的比較

表 10 以 flower 為例，嵌入多層資料時各層各數值的比較

層級	使用演算法	該層資料容量 (bits, bpp)	各嵌入模式使用個數			位置地圖容量(bits)
			Non-changeable	Changeable	Expandable	
1	FE-RDE	196608 (0.750000bpp)	0	12833	52703	485789
	ITS-RDE	196608 (0.750000bpp)	0	15693	49843	1206397
2	FE-RDE	196605 (0.749989bpp)	1	8236	57299	499574
	ITS-RDE	196608 (0.750000bpp)	0	11188	54348	1273972
3	FE-RDE	196602 (0.749977bpp)	2	5516	60018	507728
	ITS-RDE	196608 (0.750000bpp)	0	8190	57346	1318942
4	FE-RDE	196593 (0.749943bpp)	5	4086	61445	512000
	ITS-RDE	196608 (0.750000bpp)	0	6251	59285	1348027
5	FE-RDE	196584 (0.749908bpp)	8	3398	62130	514046
	ITS-RDE	196608 (0.750000bpp)	0	5022	60514	1366462
6	FE-RDE	196557 (0.749805bpp)	17	3098	62421	514892

	ITS-RDE	196608 (0.750000bpp)	0	4179	61357	1379107
7	FE-RDE	146571 (0.559124bpp)	20	2461	46396	383513
	ITS-RDE	146472 (0.558746bpp)	0	2745	46079	1032953

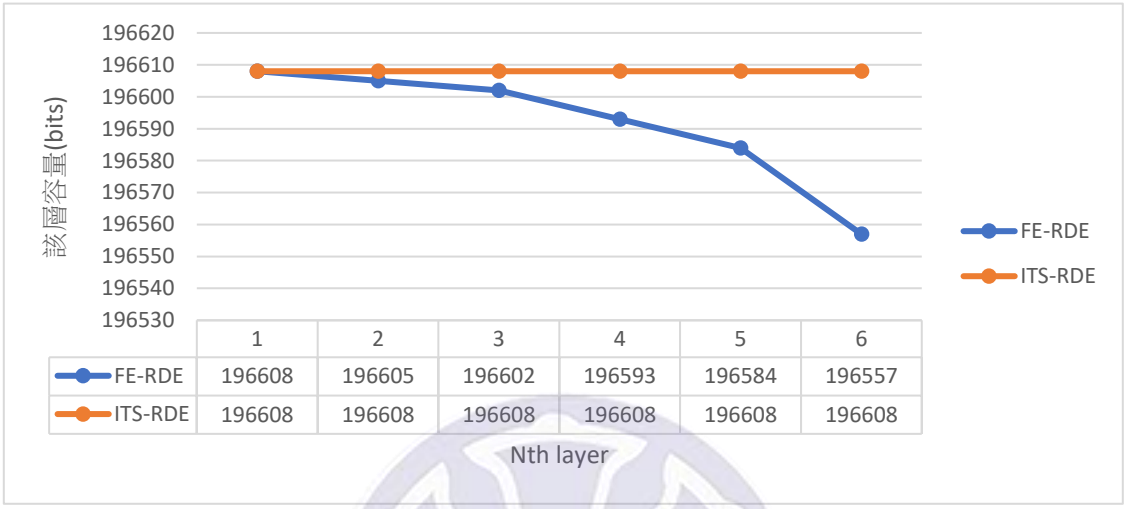


圖 43 以 flower 為例，嵌入多層資料時各層資料容量比較(最後一層除外)

(e) fruits：

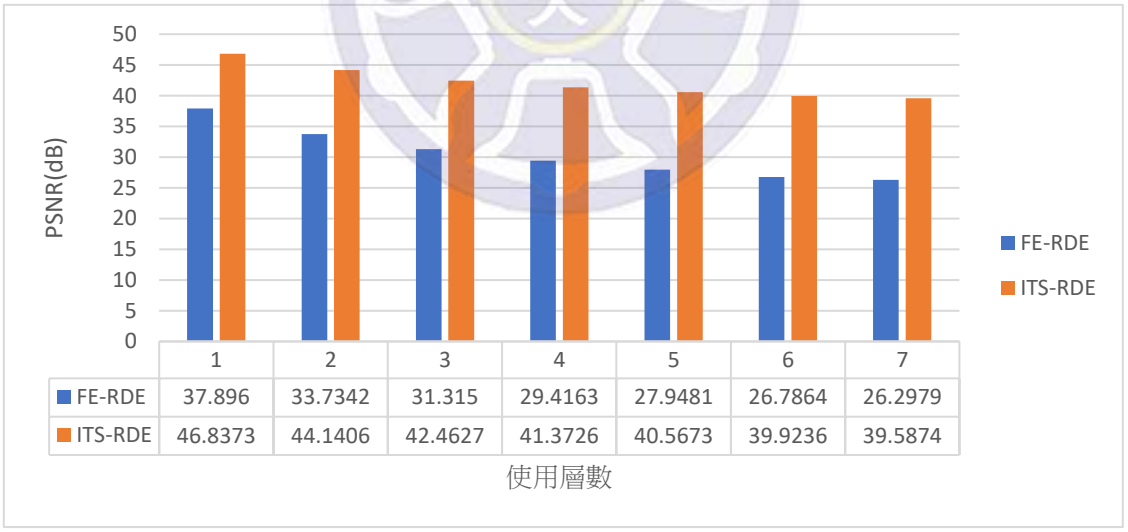


圖 44 以 fruits 為例，嵌入多層資料時 PSNR 的比較

表 11 以 fruits 為例，嵌入多層資料時各層各數值的比較

層級	使用演算法	該層資料容量 (bits, bpp)	各嵌入模式使用個數			位置地圖容量(bits)
			Non-changeable	Changeable	Expandable	
1	FE-RDE	196608 (0.750000bpp)	0	18099	47437	469991

	ITS-RDE	196608 (0.750000bpp)	0	19705	45831	1146217
2	FE-RDE	196581 (0.749897bpp)	9	14074	51453	482012
	ITS-RDE	196608 (0.750000bpp)	0	15140	50396	1214692
3	FE-RDE	196533 (0.749714bpp)	25	11436	54075	489830
	ITS-RDE	196608 (0.750000bpp)	0	12224	53312	1258432
4	FE-RDE	196494 (0.749565bpp)	38	9828	55670	494576
	ITS-RDE	196605 (0.749989bpp)	1	10256	55279	1287932
5	FE-RDE	196428 (0.749313bpp)	60	8943	56533	497099
	ITS-RDE	196605 (0.749989bpp)	1	8844	56691	1309112
6	FE-RDE	196389 (0.749165bpp)	73	8464	56999	498458
	ITS-RDE	196590 (0.749931bpp)	6	7612	57918	1327492
7	FE-RDE	147087 (0.561092bpp)	48	6603	42426	372519
	ITS-RDE	146496 (0.558838bpp)	5	6323	42509	979469

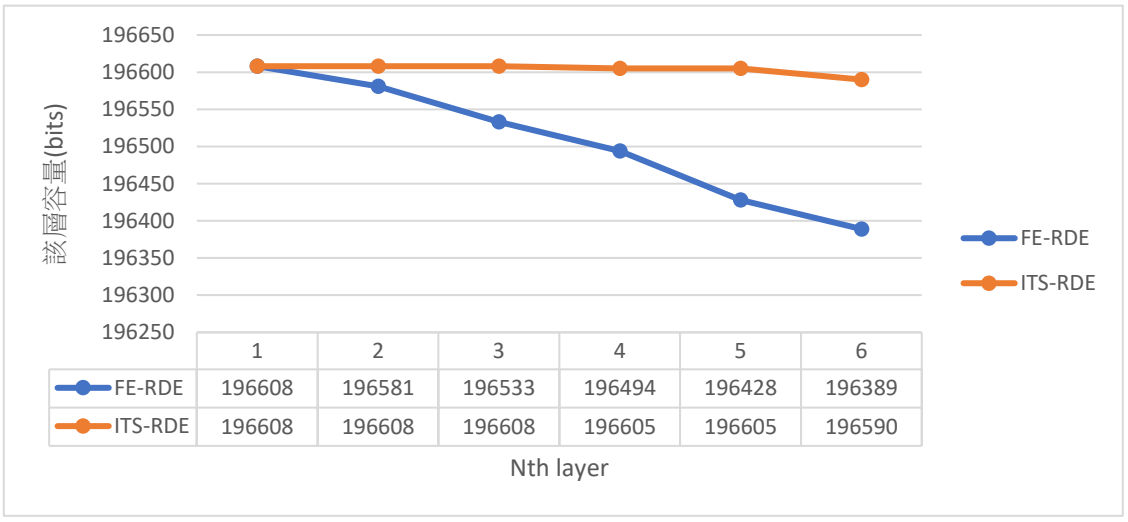
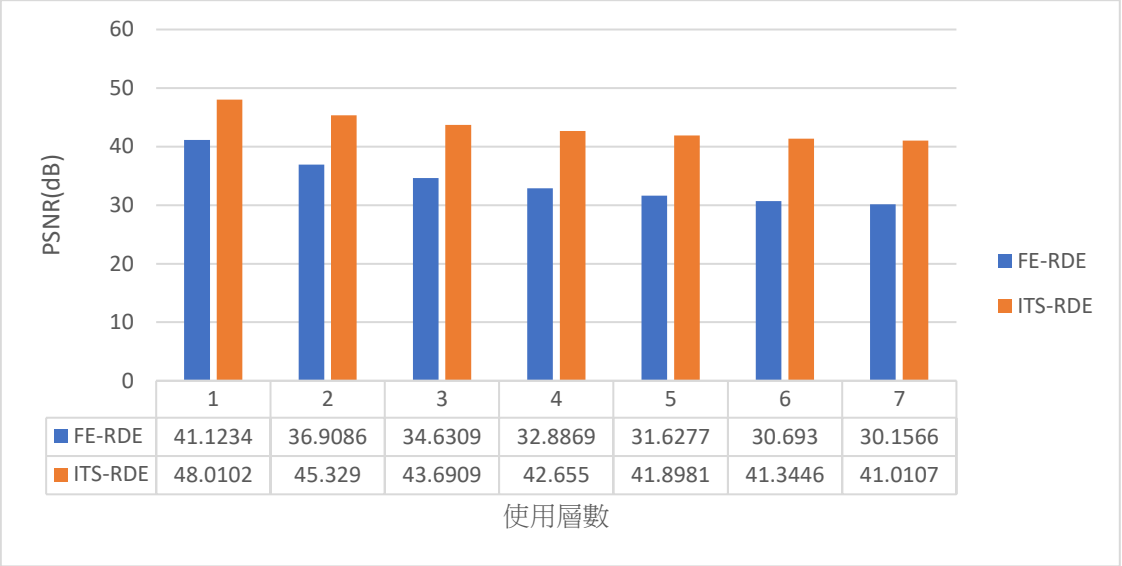


圖 45 以 fruits 為例，嵌入多層資料時各層資料容量比較(最後一層除外)

(f) lena：



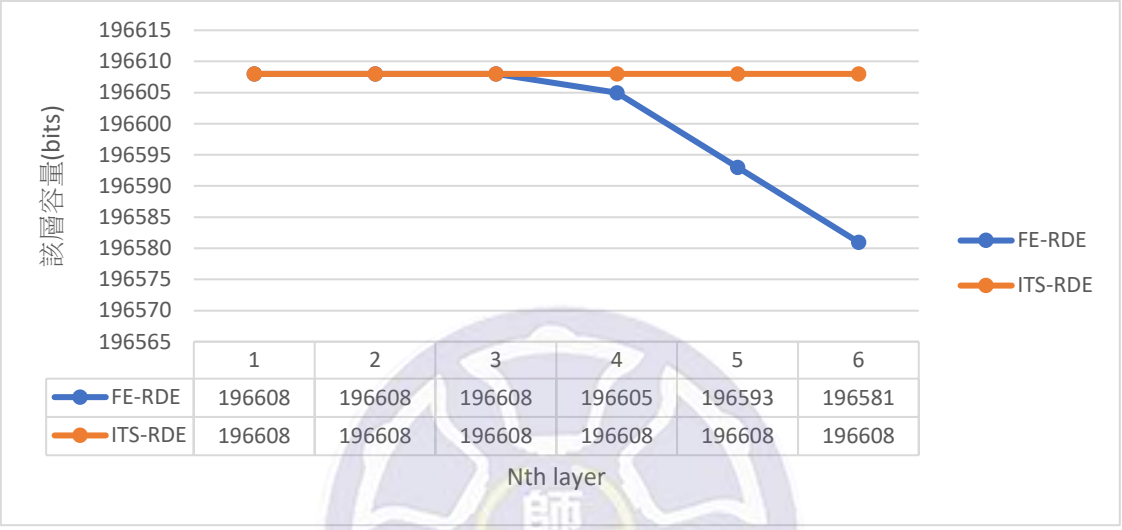
	1	2	3	4	5	6	7
FE-RDE	41.1234	36.9086	34.6309	32.8869	31.6277	30.693	30.1566
ITS-RDE	48.0102	45.329	43.6909	42.655	41.8981	41.3446	41.0107

圖 46 以 lena 為例，嵌入多層資料時 PSNR 的比較

表 12 以 lena 為例，嵌入多層資料時各層各數值的比較

層級	使用演算法	該層資料容量 (bits, bpp)	各嵌入模式使用個數			位置地圖容量(bits)
			Non-changeable	Changeable	Expandable	
1	FE-RDE	196608 (0.750000bpp)	0	14753	50783	480029
	ITS-RDE	196608 (0.750000bpp)	0	17172	48364	1184212
2	FE-RDE	196608 (0.750000bpp)	0	9222	56314	496622
	ITS-RDE	196608 (0.750000bpp)	0	11439	54097	1270207
3	FE-RDE	196608 (0.750000bpp)	0	5952	59584	506432
	ITS-RDE	196608 (0.750000bpp)	0	7900	57636	1323292
4	FE-RDE	196605 (0.749989bpp)	1	4278	61257	511448
	ITS-RDE	196608 (0.750000bpp)	0	5986	59550	1352002
5	FE-RDE	196593 (0.749943bpp)	5	3366	62165	514160
	ITS-RDE	196608 (0.750000bpp)	0	4763	60773	1370347

6	FE-RDE	196581 (0.749897bpp)	9	2873	62654	515615
	ITS-RDE	196608 (0.750000bpp)	0	3953	61583	1382497
7	FE-RDE	146517 (0.558918bpp)	15	2008	46831	384718
	ITS-RDE	146472 (0.558746bpp)	0	2534	46290	1036118



	1	2	3	4	5	6
FE-RDE	196608	196608	196608	196605	196593	196581
ITS-RDE	196608	196608	196608	196608	196608	196608

圖 47 以 lena 為例，嵌入多層資料時各層資料容量比較(最後一層除外)

在以上圖 36～圖 47 以及表 7～表 12 所檢測出的結果上我們可以看出：

1. 在品質檢測方面，從嵌入第一層資料時，Improved Two-Steps-RDE of Quads 輸出圖片的 PSNR 值會比 Fixed Enhance-RDE of Quads 的多約 1.2 倍。但隨著嵌入的層數越高，兩個演算法算出來的的 PSNR 相差會越大。到嵌入最後一層(第七層)的時候，Improved Two-Steps-RDE of Quads 的圖片大約可以掌控到約 40dB 左右的程度，且比 Fixed Enhance-RDE of Quads 的還要多約 1.4~1.5 倍。由此可知我們的改良演算法對降低圖片失真度。
2. 在每層資料容量的檢測下，原本在嵌入第一層時，雙方演算法的嵌入容量幾乎是同樣的大小，甚至在某些圖片的場合下，Improved Two-Steps-RDE of Quads 在第一層儲存的容量會比 Fixed Enhance-RDE of Quads 還要低。但在大部分的圖片載體的場合上，在層數增高的時候，Improved

Two-Steps-RDE of Quads 可嵌入容量反而繼續維持高容量，甚至沒有降低的現象。反觀 Fixed Enhance-RDE of Quads 因為擴張差值越來越大的關係，隨著層數增加，可用的容量就因為大多數像素可能幾乎造成溢位而跟著下降一些。雖然看似我們的改良演算法對圖片每層容量的提升有幫助，但因為雙方演算法在每層的容量落差也沒很大，有些圖片在 Fixed Enhance-RDE of Quads 上，第六層資料容量也頂多只下降了近百個位元的程度，對於本身可以儲存近十萬多個位元的圖片來說，我們的改良演算法對改善圖片容量的幫助其實並不大。

3. 從嵌入一層的比較得知，Improved Two-Steps-RDE of Quads 的 Expandable 在第一層上的使用個數是會比 Fixed Enhance-RDE of Quads 的還要少，但是在嵌入多層比較就不同了，隨著層數增加，使用 Improved Two-Steps-RDE of Quads 往後的層級 Expandable 的使用次數就跟著增加。在部分圖片的場合上，Expandable 使用次數隨層數增加而增加的趨勢上，Improved Two-Steps-RDE of Quads 逐層增加頻度甚至追過了 Fixed Enhance-RDE of Quads。這稍微符合了我們對改良演算法的期望。

4. 製造新圖片所製造的位置地圖容量大小，Improved Two-Steps-RDE of Quads 最少儲存大小仍舊比 Fixed Enhance-RDE of Quads 多存了約 2~3 倍的容量，而且有時會因為 Expandable 使用次數隨層級越大，使用次數就越多，該層的位置地圖容量大小也有呈現增加的趨勢。

至於 car 是一個比較特別的例子：在每層的容量比較上，Improved Two-Steps-RDE of Quads 的容量大小仍然處於比 Fixed Enhance-RDE of Quads 的還要小的劣勢，且隨著層數越多，越後面層級的容量大小也就跟著變小。其可能的原因是因為在 car 圖片上，其顏色相近或相同的區域會比較大，導致某些像素組在掃到相近顏色的區塊時，其算出的差值就會變的非常小，此時會造成嵌入模式變成 changeable 或 non-changeable 的可能性會很高。又因為 car 圖片上過白及過黑的區

域也很大，這可能造成某些像素組在擴增差值的同時，很有可能不小心造成像素的溢位，而必須強制變成 non-changeable，禁止在該像素組上嵌入資料，導致圖片可以嵌入的一層資料量變少。另外，隨著嵌入層數增多，原本圖片上過白及過黑的地方可能會有被增大的現象，致使 non-changeable 發生的機率會被增高。

第四節　實驗數據跟圖片的關係

從以上嵌入一層以及多層資料後統計出來的實驗數據來看，使用作為載體的圖片其圖片內容會影響整體實驗結果，有些圖片會導致其結果不符合理想，有些則對整體嵌入效果有顯著的提升。

嵌入多層資料的 baboon 即是一個嵌入效果提升最好的一個例子：baboon 圖片的內容上，極端值的像素(亦指灰階度接近極白或極黑的像素)是呈平均分散的程度，各個像素組的平均像素值差異較大。Fixed Enhance-RDE of Quads 上，expandable 最大仍然還是會使得新差值被擴張了約原差值的 1/2 倍，加上平均像素差距大的加劇，很有可能會讓像素值容易發生溢位，從而逐層減少容量。反之 Improved Two-Steps-RDE of Quads 中 expandable 可以將新差值收斂至原差值的大小，像素值溢位的機率相對比較少，使得每一層都可保有高容量。所以我們認為我們的方法可以改善極端值像素分散的圖片嵌入像素後容量較低的問題。

嵌入多層資料的 car 則是嵌入效果同時有優缺點的例子：缺點是在於如第四章第三節所說的：car 圖片上過白及過黑的區域很大，造成像素組變成 non-changeable 的機率更多，而且 Improved Two-Steps-RDE of Quads 在容量方面跟 Fixed Enhance-RDE of Quads 相比仍處於劣勢。不過也因為有著大範圍的顏色相近或相同區域的關係，相鄰像素之間的平均差距較小，所製造的新差值平均也比較小，致使像素的變動量縮小，PSNR 數值也跟著增高。我們認為對於有大範圍過白及過黑區域的圖片，雖然我們的方法無法對它嵌入比較大的資料，但至少更能提高圖片的品質。

對於嵌入一層資料的比較：嵌入品質(PSNR)仍得視原圖片上鄰近像素之間的

差距而定：像說 baboon 圖片中極端值的像素呈平均分散，相鄰像素之間的平均差距會比較大，所以相反地，PSNR 數值自然沒有其他圖片例子來得那麼高。圖片現況有時也會反映在嵌入模式的個數上，例如：如 baboon 般極端值像素分散的圖片，它使用的 expandable 個數比其他嵌入模式的個數還要多許多、如 car 那樣有大範圍過白及過黑區域的圖片，它使用的 expandable 個數會比較少，而 non-changeable 的個數會被增多…等。

但是在本論文上的實驗都還是在僅限於圖片中大部分的像素值都在非極度白或極度黑的狀況下。對於幾乎全部像素值都是極度白或極度黑的圖片可能就要當心了，像這樣的圖片使用 Improved Two-Steps-RDE of Quads 的方法後，可能不僅無法輸出高品質圖片，也很難有效地提升圖片載體每層資料的容量。

根據以上推論，我們認為我們的改良隱寫術演算法還不見得應付得了各種圖片情況，所以在此我們供給要使用我們的演算法的人一些選圖做載體的建議：若是想要有比較高品質的圖片，建議選擇平均像素灰度值基於中間灰度，且相鄰近像素值的差距不要過大的圖片尤佳。若是想要圖片有比較高的容量，則建議選擇平均像素灰度值基於中間灰度，且相近或相同顏色的像素盡量不要聚在一起，或盡量有一些雜訊細節的圖片尤佳。

第五章　結論及未來展望

本論文提出的改良灰階圖像隱寫術－Improved Two-Steps-RDE of Quads，是針對 T. Ahmad 等人在 2017 年所提出的灰階圖像隱寫術 Enhance-RDE of Quads [14]下去做改良。其作法是基於降低差值擴張(RDE)的方法再做一些改良，致使從原差值嵌入資料之後的新差值能夠更趨近於原差值，同時增加了交換像素值的機制，使得製造的原差值降低，並間接降低嵌入資料後的新差值。我們希望透過這些改良方法，降低差值的變更幅度，可以達到改善輸出圖片的品質，以及提升圖片對隱藏資料的可容納性等目的。

經實驗證實：我們的改良演算法能夠有效降低輸出圖片後平均像素值的變更幅度，進而提升輸出圖片的品質。在大部分的圖片的場合下，可以稍微增加使用多層式嵌入資料的方式時，其圖片上每層的資料容量大小。

在隱寫術的範疇中，有些隱寫術會添加非公開金鑰機制，在傳送端與接收端都各有一份相同數值的非公開金鑰，在使用嵌入/解嵌入演算法時就配合此金鑰補助做嵌入/提取還原的動作，以加強資料傳輸上的安全 [4]。在未來的目標，我們可能會嘗試為我們的改良演算法添加非公開金鑰的要素，譬如：計算差值可加入方程式的權重調整、可自由改變像素組大小形狀、或者是在兩階段式降低差值下新增第二階段乘數的門檻值等。

參考著作

[1] "Information hiding - Wikipedia," 28 November 2018. [Online]. Available: https://en.wikipedia.org/wiki/Information_hiding.

[2] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Information Hiding - A Survey," *Proceedings of the IEEE,* vol. 87, no. 7, pp. 1062-1078, 1999.

[3] M. S. Subhedar and V. H. Mankar, "Current status and key issues in image steganography: A survey," *Computer Science Review,* vol. 13–14, pp. 95-113, 2014.

[4] A. Cheddad, J. Condell, K. Curran and P. McKevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing,* vol. 90, no. 3, pp. 727-752, 2010.

[5] O. Koval, S. Voloshynovskiy, T. Holotyak and T. Pun, "Information-theoretic analysis of steganalysis in real images," *Proceedings of the 8th workshop on Multimedia & Security (MM&Sec),* pp. 11-16, September 2006.

[6] A. K. Pal, N. Dey, S. Samanta, A. Das and S. S. Chaudhuri, "A hybrid reversible watermarking technique for color biomedical images," *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC),* pp. 1-6, 2013.

[7] M. Mishra, P. Mishra and M. Adhikary, "Digital Image Data Hiding Techniques: A Comparative Study," *ANSVESA,* vol. 7, no. 2, pp. 105-115, 2012.

[8] A. Kaur and M. K. Dutta, "A hybrid approach for optimizing transparency, robustness and capacity of an audio watermarking algorithm," *Ninth International Conference on Contemporary Computing (IC3),* pp. 1-6, 2016.

[9] J. Tian, "Reversible Data Embedding Using a Difference Expansion," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, no. 8, pp.

890-896, 2003.

[10] D. C. Lou, M. C. Hu and J. L. Liu, "Multiple layer data hiding scheme for medical images," *Computer Standards & Interfaces,* vol. 31, no. 2, pp. 329-335, 2009.

[11] A. M. Alattar, "Reversible watermark using difference expansion of quads," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* vol. 3, pp. 377-380, 2004.

[12] T. Ahmad, M. Holil, W. Wibisono and I. R. Muslim, "An improved Quad and RDE-based medical data hiding method," *International Conference on Computational Intelligence and Cybernetics (CYBERNETICSCOM),* pp. 141-145, 2013.

[13] M. H. A. AL_Huti, T. Ahmad and S. Djanali, "Increasing the capacity of the secret data using DE pixels blocks and adjusted RDE-based on Grayscale Images," *IEEE-International Conference on Information Communication Technology and System (ICTS),* pp. 225-230, 2015.

[14] P. Maniriho and T. Ahmad, "A data hiding approach using enhanced-RDE in grayscale images," *IEEE-International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA),* pp. 35- 40, 2017.

[15] C.-H. Chuang, C.-W. Cheng and Z.-Y. Yen, "Reversible data hiding with affine invariance for 3D models," *IET International Conference on Frontier Computing. Theory, Technologies and Applications,* pp. 77-81, 2010.

[16] J. Singh, G. Kaur and M. K. Garcha, "Review of Spatial and Frequency Domain Steganographic Approaches," *International Journal of Engineering Research & Technology (IJERT),* vol. 4, no. 6, pp. 1122-1124, 2015.

[17] T. Ahmad, D. M. Firmansyah and D. S. Angreni, "Difference Expansion-Based Data Hiding Method by Changing Expansion Media," *The 2nd International Conference on Soft Computing and Data Mining,* vol. 549 AISC, pp. 414-423, 2016.

[18] D. S. Angreni and T. Ahmad, "Enhancing DE-based data hiding method by controlling the expansion," *2016 4th International Conference on Cyber and IT Service Management,* pp. 1-6, 2016.

[19] 葉佳憲, “基於小波轉換及差值擴張之可逆式資訊隱藏,” 國立屏東教育大學資訊科學系碩士班碩士論文, 2014.

[20] L. Gao, T. Gao, G. Sheng, Y. Cao and L. Fan, "A new reversible watermarking scheme based on Integer DCT for medical images," *2012 International Conference on Wavelet Analysis and Pattern Recognition,* pp. 33-37, 2012.