利用材質合成演算法之快速物體摘除系統
Fast Object Removal System Using Texture Synthesis Algorithm

李俊賢[1]　陳英豪[2]　何佩璇[2]
Jiunn-Shyan Lee　Ying-Hao Chen　Pei-Syuan He
修平技術學院　[1]資訊傳播系　[2]資訊管理系
jslee@mail.hit.edu.tw　BF92035@mail.hit.edu.tw　BF92020@mail.hit.edu.tw

摘要

本研究主旨在發展一套有效的、快速的、自動化物體摘除系統，搭配現今主流市場上最多人使用的視訊剪輯軟體 VirtualDub，設計成一個專門給自由軟體使用的外掛模組，不但可強化應用系統之附加價值，更可滿足一般使用者對於物體摘除工具的迫切需求。我們打算利用材質合成技術來做為物體摘除的解決工具。首先我們採用以區塊為基礎的材質合成法，根據一張較小的範本材質，將每一塊被挑選出的來源區塊以予部份覆疊，進而產生出較大張且具有相同紋理特性的新材質。然後我們擴大材質合成的用途實施到限制性材質合成，在我們所設計的雛型系統裡，使用者只需提供來源與遮罩輸入影像，然後系統便可自動利用相鄰的有效紋理進行填補遮罩所指示的空洞。

關鍵詞：範本為基礎的、限制性材質合成、視訊剪輯與處理。

1. 前言

在影像處理的編輯工具裡或是影片的後製剪輯作業中，想辦法去摘除影像的部份缺陷是一項非常重要的工作，例如：老舊的相片可能會有刮痕或斑駁的現象；掃描進來的影像可能會有摺痕出現；拍攝的照片也可能會出現其他遊客。一般人會藉由影像處理軟體所提供的編輯工具，如雲彩筆(clone brush)，先選擇相近的色塊，再去不斷的塗抹於要移除的區域。這種擾人、重覆性與容易出錯的工作，通常需要經驗老道有技巧的設計師，不斷操作才能達成令人滿意的修正效果。

材質影像一般都可由真實世界中透過數位化方式擷取出來，然而從真實世界中所擷取到的材質影像往往具有若干缺陷，例如被前景物體所遮掩或陽光投射後，所產生之高亮點與陰影等瑕疵。材質合成演算法[6, 9]已將這些缺陷予以消除或使之視覺上很難察覺。因此我們將利用材質合成技術來自動填補移除前景物體時所遺留下之空隙。

綜觀國內外當前視訊剪輯或影像處理軟體的主要功能，當下之急似乎需要一套能有效提供物體摘除或雜訊修復方面的工具。雖然現今相關影像修補的技術已被廣泛的研究，但通常還停留在影像(image)而非影片(video)，同時開放源碼做為免費使用且效果令人滿意的系統，似乎付之闕如，於是興起我們去設計發展一套簡單、好用又有效的物體摘除應用軟體之念頭。我們的實作系統將設計成一個專門給 VirtualDub [14]使用的外掛工具，如此可發揮此實作系統的最大成效，並擴充自由軟體 VirtualDub 的強大附加價值。

影像補繪(Image Inpainting)[2, 3]技術能夠還原裂痕，修復小塊的缺陷，就像是被劃傷的舊相片或沾有污漬等瑕疵；材質合成(Texture Synthesis)[4, 6, 9]技術能夠填補較大塊的缺陷，適合內容為一致性、規律性與結構性的材質紋理。通常被前景物體所遮掩的未知背景，其影像內容千變萬化，任何情況都有可能[8]，只能利用有限的已知線索來臆測未知背景，所以物體摘除技術的發展仍是一個值得深入研究的挑戰課題。

2. 背景知識

電腦圖學應用領域中，將一張材質影像貼附於三度空間物體的表面，是一項被廣泛採用的技巧，此種技術被統稱為材質貼圖(Texture Mapping)。材

*本研究承蒙國科會工程處專題研究計畫之經費補助(NSC94-2218-E-164-001)，謹向國科會工程處致謝。

質貼圖不僅可提升成圖後圖像之視覺效果，亦可降低成圖所需之龐大計算時間。然而，由於場景中各物體表面大小不一，但材質影像卻為固定大小，故雖然我們可以利用重複連續貼圖的方式來覆蓋物體的表面；但是，此舉也導致物體表面的材質具有太多的重複性，相對的也產生不佳的視覺效果。有鑑於此，研究學者乃轉而研究材質合成演算法[12]。此類演算法主要探討如何利用一張固定大小、低解析度的輸入材質圖像，來產生出具有任意解析度、類似於原始輸入材質之圖像。

材質合成演算法只需輸入一張範本材質，就能夠依據此材質紋理內容來輸出任意大小的新材質，並且對人類的直接觀察而言是類似的、相差無幾的。此演算法是建築在馬可夫隨機場(Markov Random Fields)的模型上，可以不必去建構完整或明確的機率函數，而改採用一個較為決定性的來源上面做搜尋。文獻研究已從像素為基礎(pixel-based)的合成[6, 9, 12]轉移到區塊為基礎(patch-based)的合成方法[7, 10]，主要原因為區塊為主的合成法較像素為主的合成法快上許多倍，並且可成功的套用到多種類材質，包含隨機性紋理(如礫石、草地)與規則性紋理(如磚牆、門窗)。

經過實驗證實，材質合成演算法也能用來修護圖片瑕疵，特別是毀壞的區域原本內容就是屬於結構性的紋理；而影像補繪演算法則互補了材質合成法，可以拿來解決顏色或強度漸層的問題。Bertalmio [2]模擬專業修復者的技法，從外圍邊界逐漸向內進行平滑的資訊傳遞，利用此影像補繪法來填補影像的遺漏區域，此法比較適合面積相對小、內容平順與非紋理的區域。最近，Bertalmio [3]結合影像補繪與材質合成之技術，把影像分解成兩種不同的組成成份，影像補繪方法被套用到影像的基本結構，而材質合成方法則被實施到影像的細節，最後再合成兩種組成。針對遺漏區域面積相對較大、包含紋理與大型結構體的填補，Drori [5]提出了影像完成(Image Completion)的全新技術，疊代方式處理要摘除的物體，其使用簡單的平滑內插來近似估計遺漏區域，並且根據最頻繁的與最相似的範本來逐步加入細節。

3. 系統架構

在本章節中首先我們將描述材質合成演算法的基礎作法，材質合成意謂根據一張較小的範本材質，進而產生出較大張且具有相同紋理特性的新材質。材質合成的用途之一為可進一步擴展成具限制性材質合成的應用，例如空洞填補(hole filling)與物體摘除(object removal)等。我們所提出的新方法是採用以區塊為基礎(patch-based)的合成方式，不管在計算速度或輸出品質上均佔有優勢，詳細系統實作架構依序敘述於以下各小節。

3.1 材質擴張

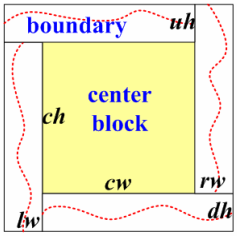
我們所提出的材質合成方法係啟發自 Efros [7]與 Liang [10]的研究論文，兩者皆使用以區塊為基礎的合成取樣法，其為將每一塊被挑選出的來源區塊以予部份覆疊，以形成另一個新的大張輸出材質，如此作法浮現出兩個問題，一個是如何挑選下一區塊，另一個是如何處理覆疊(overlap)邊界之問題。與他們方法較大不同點在於我們提出一個整合的覆疊技術，此新技術整合動態規化(dynamic programming)與加權羽化融合(weighted feathering)的優點來處理覆疊邊界，促使達成平滑過渡。

首先，輸入的範本材質本身成就了一系列的候選區塊 B_s，我們的演算方法係採用下到上、左到右的掃描線處理方式，對於每一個要被合成的目的區塊 $B_t(i)$，它的覆疊邊界區域其像素值早已得知，會被拿來跟所有的候選區塊 B_s 的相同位置像素進行比對，通常 L2 norm 距離會被採用以作為評估兩個相互比較區塊的相似度，如方程式(1)所示為計算兩重疊邊界區域S與T的距離，然後一個(或多個)最佳配對(best-matched)的來源區塊 $B_s(j)$，其擁有最短距離因而被挑選中，會被貼附於目的區塊 $B_t(i)$ 的原始位置，其中中心區域直接複製，而邊界區域因有重疊需要進一步處理。

$$D(S,T)=\sum_{k\in O}[\,(r_S(k)-r_T(k))^2+(g_S(k)-g_T(k))^2+(b_S(k)-b_T(k))^2\,] \quad (1)$$

從上述的演算流程中得知其最主要的步驟為覆疊邊界的比較搜尋，於是我們必需在所有的來源候選區塊中找出與目的區塊擁有最小差距的區塊，我們將此動作歸類為高維度(high-dimensional)的最接近鄰居搜尋(nearest neighbors search)問題。當有 n 個要被比對的來源候選區塊，且每一個含有 m 個向量成份，其暴力(brute force)搜尋法的複雜度將高達 $O(mn)$，假若改由 kd-tree 的技術輔助，其平均搜尋與最差搜尋時間之複雜度將可有效降低到 $O(\log n)$ [1]，但是建構 kd-tree 的前處理時間與空間是必需花費的成本。所以在進行合成的搜尋前，我們會將所有來源候選區塊的邊界區域組成一個個特徵向量，依此特徵向量建築一個 kd-tree，以利後續的最佳配對搜尋操作。

要是只針對材質擴張的訴求而言，使用 L-shaped 重疊邊界也就足夠，然而對於較特定的應用如可貼磚材質(tiling texture)與限制性合成(constrained synthesis)等，就要再進一步的設計，在 Liang [7]的系統中其建立了四種不同型的 kd-tree 以合乎各種邊界情況要求，而我們是採用如下圖 1 所示的區塊架構，其包含一個中心區域與包圍它的四個邊界區域，對於一般的材質合成採用簡化的 L-shaped 即可，而針對具有限制性的材質合成我們將採用 O-shaped 區塊，而去避免建置多棵 kd-tree。



configuration for O-shaped patch
dotted line : minimum cost path

source resolution : sw×sh
target resolution : tw×th
patch width : pw = lw+cw+rw
patch height : ph = dh+ch+uh
#candidate = (sw-pw+1)*(sh-ph+1)
for non-tiling source
#candidate = sw*sh
for tiling source
boundary_size = pw*ph-cw*ch
feature_vector_size = boundary_size*3
for RGB match
feature_vector_size = boundary_size
for luminance match
#task = ⌈tw/cw⌉*⌈th/ch⌉

圖 1　O-shaped 區塊的設計架構圖

此外，對於合成結果輸出品質最有顯著影響的是重疊邊界的處理方式，如圖 2 所示為我們新提出的整合技術說明，以求有效達成兩相鄰區塊在重疊邊界上的一致性與平滑連貫性，第一步驟我們先採用動態規劃演算法求得一條最短路徑，第二步驟採行加權的羽化融合技術來合併兩重疊像素。

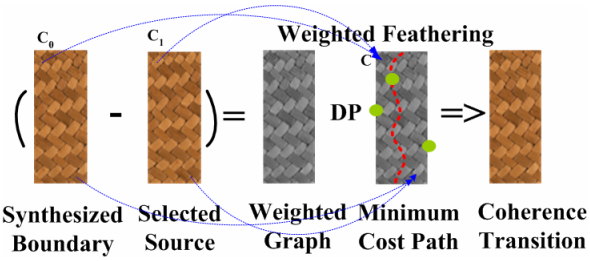


圖 2　整合的覆疊邊界處理技術

讓 B1 與 B2 表兩個有相同 mxn 大小的覆疊邊界區域，對於每個位於位置(i, j)的距離權重(weight)為 $e_{i,j}$，如方程式(2)所示，然後我們將找到一條最短路徑從列 0 出發走到列 m-1，而方程式(3)為從列 i=m-1 遊走到列 i=0 時的計算過程，$E_{i,j}$ 表示從位置(i, j)到終點列 m-1 為止所發現的最短距離值。

$$e_{i,j} = (B1_{i,j} - B2_{i,j})^2 = (r1_{i,j} - r2_{i,j})^2 + (g1_{i,j} - g2_{i,j})^2 + (b1_{i,j} - b2_{i,j})^2 \quad (2)$$

$$E_{i,j} = \begin{cases} e_{i,j} & for\ i = m-1 \\ e_{i,j} + min(E_{i+1,j-1}, E_{i+1,j}, E_{i+1,j+1}) & for\ i = 0..m-2 \end{cases} \quad (3)$$

一旦求出最短路徑我們接著套用混合(blending)的技術，也就是線性內插的方式融合重疊在一起的兩像素點，讓 C_0 與 C_1 表示兩個重疊在一起即將要被混合的兩像素點，其線性內插所求得的輸出 C，式子為 $C = \alpha C_1 + (1-\alpha)C_0$。對於最短路徑通過的位置其權重 α 是 0.5，並且兩左右邊界端點分別為最低 0 與最高 1，其權重計算函數如方程式(4)所示，其中 p 表於該列最短路徑所經過位置。

$$for\ each\ pixel\ k\ in\ column\ 0..n-1\ in\ the\ same\ row: \quad \alpha = \begin{cases} 1 & if\ \ p = 0 \\ 0 & if\ \ p = n-1 \\ k/(2p) & if\ \ k \le p \\ (k-p)/2(n-1-p) + 1/2 & if\ \ k > p \end{cases} \quad (4)$$

3.2 物體摘除

材質合成技術也可被用來成為解決某些特定問題的工具，舉例來說，老舊的相片可能會有刮痕或斑駁的現象；掃描進來的影像可能會有摺痕出現；拍攝的照片也可能會出現其他遊客，因此我們提出的技術能用來掩蓋回復這些另人不悅的缺陷。

在我們所設計的系統裡，使用者只需事前準備好四張輸入影像，其分別為：來源(source)影像、來源遮罩(source mask)影像、目的(target)影像與目的遮罩(target mask)影像。然後系統即可自動利用相鄰的有效材質進行填補遮罩所指示的空洞，並可在極短的時間內計算合成完畢。在來源遮罩影像的白色像素區域代表了可作為使用的有效候選區塊，而在目的遮罩影像的黑色像素區域則指示了那些區塊需要進一步被重新合成，所以兩者放在相同一張並不起衝突。利用來源與目的遮罩影像，我們雛型系統將可快速的合成限制性材質(constrained texture)，而完全不需要藉助任何的人工干涉。

3.3 軟體之外掛工具設計

VirtualDub 在 Free Software 與 Open Source 社群是一套享有盛名的專業等級軟體，簡單來說它是一個高效能、高品質的非線性剪輯軟體工具。其支援現今所有主流影片與圖片的壓縮格式(如 MPEG-2、MPEG-4)，於是它常被用為不同壓縮格式影片的轉換工具。此外它也提供各式各樣的影像處理濾鏡功能(如色彩、亮度、對比調整、平滑、銳化、浮雕等)，當添加濾鏡效果輸出後可視為一般的影像處理軟體之用，而且就算是影像處理濾鏡用途不符合需求，更可自行發展或修改外掛濾鏡程式，可說是直逼商業化高價位影片剪輯軟體水準。

在本系統發展考量上，優先考慮各種可能影響系統架構的可能因素，考量是否採用自行研發的封閉系統或開放軟體的外掛工具設計架構。

- 方案一：自行研發的封閉系統

通常要完成一個高效能、高品質的非線性剪輯軟體工具，所花的時間與人力是龐大的，在有限的時間與人力的研究與實作過程當中，通常只能完成初步的材質合成模組與簡易的實驗執行界面。此外對於使用者的操作訓練學習過程也是耗時辛苦的。

- 方案二：開放軟體的外掛工具

外掛(plug-in)程式通常是指針對某一套應用軟體，設計出可整合加入的程式模組，可進一步提升程式附加價值與彌補原應用軟體之不足，於是我們可考慮採用 VirtualDub 軟體的外掛方式發展系統，如此可發揮此實作系統的最大成效。

由各項比較得知，為發揮實作系統的最大效用與實踐自由軟體之精神與美意，我們偏向採用目前較多人充份使用的主流軟體 VirtualDub，利用其外掛程式庫來整合發展物體摘除系統，以完成其外掛工具的方式做為最終目標。是故系統設計將採用[方案二：開放軟體的外掛工具]為解決方案。目前系統執行可快速得到如圖 3 所示之合成計算結果。

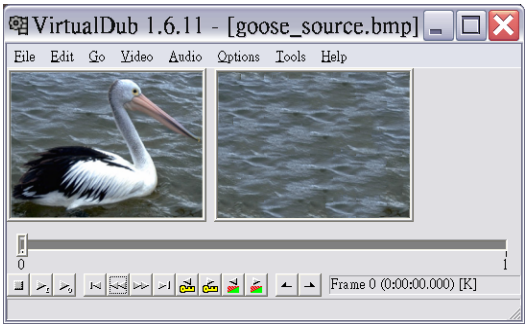


圖 3　系統執行結果之畫面

4. 結果與討論

4.1 實驗結果

如圖 4 所示為我們的實驗結果，從圖中可以顯示出系統有不錯的初步成效，的確可成功的實現物體摘除。系統是執行在 Pentium 4 - 2.6 GHz 及 512M RAM 的個人電腦上，其總共合成時間包含訓練與合成階段如表 1 所示都能夠在很短的時間內完成，結果堪稱理想，但是虛擬記憶體的需求用量較大，這是未來可加以改進之處。

表 1　物體摘除之執行時間統計(單位：秒)

	訓練階段	合成階段	總合時間
Hollywood	22.662	2.524	25.186
Pigeon	9.347	2.034	11.381
Railroad	0.054	0.592	0.646
Caption	0.009	0.034	0.043
Lincon	5.454	2.421	7.875

經由實驗結果顯示出我們的初步系統可達成令人滿意的合成結果，但仍有其進一步修正處與擴

充性。材質合成填補技術適合較大塊、結構性材質紋理的填補(如 Hollywood, Pigeon, Railroad)，但卻不易消除較小塊之影像雜訊與裂痕(如 Lincon)，相對的影像補繪填補技術則無法對付結構性的材質紋理，卻可成功修補小塊之影像雜訊與裂痕，於是未來可考量設計材質合成與影像補繪兩種修補技術相互結合的新修補方法。

4.2 分析討論

如圖 1 所示我們設計了一個 O-shaped 的區塊(patch)，其包含四個將作為比較的邊界，該區塊的所有參數設定是可彈性調整的。當系統在作配對搜尋的階段，這些重疊的邊界被視為特徵向量(feature vector)處理，成為比對搜尋上的統計限制(statistical constraint)。區塊中心區域的大小(cw×ch)選擇是使用者所指定的，通常仰賴所給定材質的特性，其大小必需滿足材質所展現的紋理結構，也就是要比材質的內容元素大。一個較小區塊中心的使用會導致較弱的統計限制，相反的一個較大區塊中心的使用會造成龐大的計算量與 kd-tree 的空間花費。特別值得一提，只要經過適當的設定，要將 O-shaped 區塊轉型成 L-shaped 區塊也輕而易舉，此外要是不打算實施區塊合成而要實行像素合成，也不必大費周張重修改程式。

對於邊界大小(lw-dh-rw-uh)的選擇也是一種交換取捨(tradeoff)，通常是要足夠大才能避免流失紋理特性或形成邊界不連續，一個較大的邊界選擇會有較佳的統計限制，但同時也造成了合成時的 kd-tree 空間與時間的大量花費。我們還採用了一個簡單的加速策略，對於連續相臨的 R 個候選區塊不必全部加入到 kd-tree 中，我們只挑選其中一個成為候選區塊，如此以降低來源區塊的數量，但 R 值的挑選也是一種對於速度與品質間的交換取捨。

5. 結論與未來工作

5.1 結論

本研究主旨在發展一套有效的、快速的、自動化物體摘除系統，實用上可作為影片上商標與字幕的摘除工具，我們的實作系統被設計成一個專門給自由軟體 VirtualDub 使用的外掛模組，可當成影像濾鏡來用，主要功能為摘除任何不要的物體，例如在影片中隨手可見的電臺標誌、字幕、跑馬燈與時間郵戳等。

對於靜態影像由於沒有任何資訊，所以不知道這些不透明物件背後所隱藏的真正影像內容為何，我們只好根據材質合成的技術原理來重繪物體被拿掉後所遺留的空缺。為了達成自動化合成，我們用可見的部份當成訓練資料集合去填補未知的區域，產生出一個完整、效果佳與一致性的填補結果，例如利用周圍相鄰的區域像素值與紋理內容。

系統設計目標為半自動化工具，不需要煩人的操作過程，只要簡單的幾個步驟就能自動完成物體摘除與填補修復。例如，給定一張輸入來源影像與其對應的填補遮罩，其用於定義摘除區域，系統會根據現有的已知區域去完成填補未知區域。

5.2 未來展望

影像補繪是一種以內插手法，利用遺失區塊邊界資訊，以由外往內方式平滑地填滿遺失的影像資訊。此類技術適用於較小、較平滑且非材質類背景之區域或影像雜訊。一般而言，進行修補的操作，除了顏色的傳遞或擴充外，物件線條的回復也是影像修補技術中需要被考慮的[11]。當要被修補的區域很大時，無論是形狀上或顏色的分布上都很難從周圍取得足夠的資訊來做漸進式的填補，例如當磚瓦的一角被污損一大塊時，影像補繪之修補技術都無法有效解決。此時可改用材質合成之填補技術，但是相反的材質合成技術則不易消除較細微之影像雜訊，於是新的修補方法將是著重在於材質合成與影像補繪兩種修補技術的結合[3, 5, 13]，這是我們未來進一步設計改良時的另一思考方向。

摘除物體的修補可利用與填補區域有相關的資訊來進行修補，其資訊來源可能是鄰近區域的像素、影像內的全域資料、或是連續影片中相鄰的畫面，皆可供作搜尋使用的實用資訊。若是針對多張相似影像或動態影片進行修補的行為，我們就可以從其他影像或是連續的影片畫面裡，找尋一樣的物

體，然後以它為參考資訊，找尋可作為來源區塊的資訊，進而進行填補的合成計算。

參考文獻

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," *Journal of the ACM*, 45(6), pp. 891-923, 1998.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," in *Proceedings of SIGGRAPH 2000*, pp. 417-424, 2000.

[3] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous Structure and Texture Image Inpainting," *IEEE Transactions on Image Processing*, 12(8), pp. 882-889, 2003.

[4] A. Criminisi, P. Perez, and K. Toyama, "Object Removal by Exemplar-Based Inpainting," in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 721-728, 2003.

[5] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-Based Image Completion," in *Proceedings of SIGGRAPH 2003*, pp. 303-312, 2003.

[6] A. Efros and T. Leung, "Texture Synthesis by Non-Parametric Sampling," in *Proceedings of International Conference on Computer Vision*, pp. 1033-1038, 1999.

[7] A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer," in *Proceedings of SIGGRAPH 2001*, pp. 341-346, 2001.

[8] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications*, 22(2), pp. 56-65, 2002.

[9] P. Harrison, "A Non-hierarchical Procedure for Re-synthesis of Complex Textures," in *Proceedings of Central Europe on Computer Graphics, Visualization and Computer Vision*, pp. 190-197, 2001.

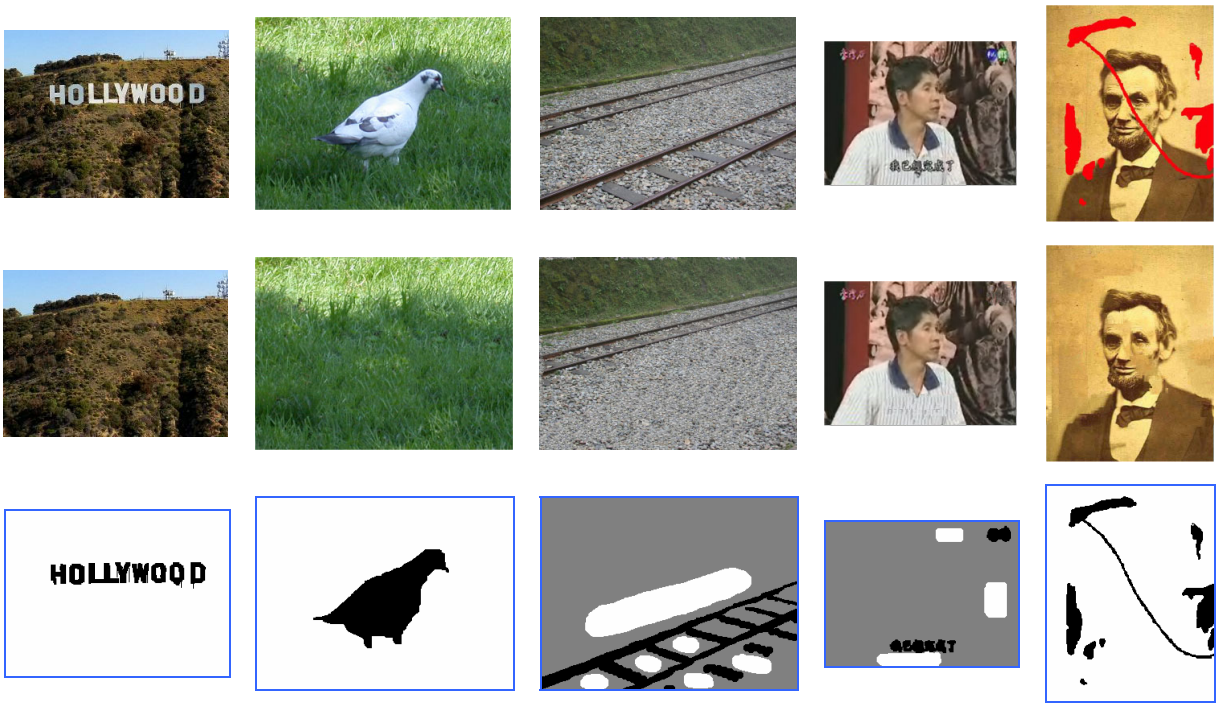
[10] L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum, "Real-Time Texture Synthesis by Patch-Based Sampling," *ACM Transactions on Graphics*, 20(3), pp. 127-150, 2001.

[11] J. Sun, L. Yuan, J. Jia, and H. Y. Shum, "Image Completion with Structure Propagation," in *Proceedings of SIGGRAPH 2005*, pp. 861-868, 2005.

[12] L. Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-structured Vector Quantization," in *Proceedings of SIGGRAPH 2000*, pp. 479-488, 2000.

[13] H. Yamauchi, J. Haber, and H. Seidel, "Image Restoration using Multiresolution Texture Synthesis and Image Inpainting," in *Proceedings of Computer Graphics International*, pp. 120-125, 2003.

[14] "VirtualDub.org," http://www.virtualdub.org.



Hollywood　　Pigeon　　Railroad　　Caption　　Lincon

圖 4　物體摘除之執行結果。第一列為來源影像，第二列為合成結果影像，第三列為遮罩影像。