新的動態材質合成演算法

王宗銘
國立中興大學資訊科學研究所
cmwang@cs.nchu.edu.tw

賴昭宏
國立南投高商
jhlay@msl.pntcv.ntct.edu.tw

摘要

本篇論文中我們提出了一個新的演算法來合成動態材質。以往動態材質的合成皆需要以錄影片段為來源，且需要較長的計算時間。有別於之前的方法，我們的演算法只需一張材質，且所需計算時間較少，就能有效的合成不限時間及不限大小的動態材質。我們以材質合成的方法為基礎，以維持連續性的方式來合成所需材質，再以一近似值演算法巧妙的重排各張材質的順序，能排出接近自然動態材質的片段。我們也能有效的計算重播機率，且利用 cross-fading 的方法，可播放出任意時間長度的動態畫面。我們展示了多種材質的執行結果，如波浪、海水、火焰及雲霧等。結果顯示我們的方法能更方便、更快且有效的合成動態材質。

關鍵詞：材質合成、近似值演算法

一、簡介

動態材質合成在電腦繪圖中是一項很有趣且重要的研究。在電腦動畫、電腦遊戲、電腦繪圖藝術方面皆有許多應用。近年來已有許多研究工作能發展出各種動態材質，但仍有許多困難需要解決。

目前動態材質的合成方法可分為兩種：一種是直接模擬(direct simulation) [9]；另一種是使用材質合成(texture synthesis)的方法[12][6][7]。直接模擬的方式可產生很漂亮、逼真的畫面，也可產生一般環境中看不到的場景。此種方法不需要輸入材質，能以計算的方式產生輸出材質。但直接模擬的每個畫面往往需要大量的計算，且欲產生不同的材質需有不同的演算法。另一種使用材質合成的方法，可以產生和自然環境中幾乎一樣的場景。一般說來，利用材質合成的方法，使用者較能自己決定材質的種類，且合成的結果較接近現實畫面。此類方法能以一種演算法而能應用於多種動態材質的合成，且播放時不需再做大量的計算。但此類方法需要以錄影片段為輸入材質，來源取得較不容易且需較大的儲存空間，其合成所需片段時也需較多的計算時間。我們希望能較容易取得輸入材質，且能以一種演算法快速、有效的合成動態材質。

本論文基於上述論點，提出一種新的演算法，只利用一張材質，能很快的產生不限時間且不限大小的動態材質。我們有許多難題需要解決：(1).要如何由一張材質合成出所需張數的材質？。(2).如何維持各合成材質間的連續性(continuity)？(3).如何無限時間的播放？在第三節中我們將詳細介紹解決方法，我們以有限張數的合成材質，經過巧妙的重排順序，再加上能欺騙視覺的播放方法，能呈現出接近自然動態的效果。

本文計分六節，其架構如下：第二節探討相關工作；第三節詳述我們所提出的演算法；第四節敘述並分析實驗結果；第五節提出結論與未來工作；第六節為參考文獻列表。

二、相關工作

目前所知動態材質合成皆是以錄影片段為輸入材質，分析其結構，藉以合成其他片段。Wei et al. [12] 將其材質合成的方法由 2D 擴充為 3D。每合成一個新的畫面(frame)時，該畫面每個像素的周圍環境是由之前合成過的數張畫面的相同區域所組成。此周圍環境再與原暫時性材質中相對應的範圍比較，取最相似者。並使用 TSVQ(Tree-Structured Vector Quantization)加速搜尋的動作。但相對於合成所需時間的片段，仍需要不少時間。

Bar-Joseph et al. [6] 在 Texture Mixing 有很好的效果。在合成動態材質部分，使用隨時間變動材質(Time-Varying Textures)建立 MRA(Multi-Resolution Analysis)結構。使用統計式學習演算法(statistical learning algorithm)產生新的亂數的 MRA tree，以此建立新的動態材質。速度比 Wei 快。但其需要大量記憶體，且每次只能產生與原片段相同長度的片段。

Kwatra et al. [7] 利用圖像切割(graph cut)的方法產生不規則形狀的區塊，能與合成影像的其他部分有最佳的縫合(seam)狀態。不論在

二維或三維材質的合成皆有很好的效果。且能截取多張影像的某部分，有效的合成一張新影像。於動態材質合成方面可有效合成無限時間且無限大小的材質片段。但仍需以錄影片段為輸入資料，且需較長時間計算。其計算時間視輸入片段的大小而定，需從 5 分鐘至 1 小時不等。

至今動態材質合成皆需要以錄影片段為資料來源，且需長時間的計算或搜尋過程。我們所提出的新演算法能只以一張材質為來源，很快且有效的合成不限時間及不限大小的動態材質。且只需一般工作平台即可操作。

三、演算法

我們觀察到動態材質會呈現出暫時的規律性(temporal regularity)，其中各單張材質間呈現相似卻又不同的圖像。我們想到材質合成的方法正可以合成其他不同卻又看得出是相同紋理的材質。我們大膽推論：每張真實片段中的材質皆可在很多張合成的材質中找到極相似者。這是理想狀態。事實上受限於時間及電腦記憶空間，我們無法合成那麼多張材質。所以我們以有限張數的合成材質，經過巧妙的重排順序，再加上能欺騙視的播放方法，能呈現出接近自然動態的效果。

我們的方法主要分成四個部分：1.系列材質合成。2.重排順序。3.計算重播機率。4.播放。在決定合成所需張數的材質時，我們希望至少有 1 秒鐘的片段，以 10 frames/sec 來算，至少合成十張。以我們實驗的結果，合成二十張即可有不錯的效果。這部分可用參數控制。合成所需張數後，我們以一近似值演算法(Approximation algorithm)來加以重排，排成接近真實拍攝的畫面順序。在計算重播機率的部分，我們修改[10]中的計算方式，改用加權式的機率表，以配合之前重排的順序。最後以 cross-fading 的方式播放各畫面，其中各畫面的加權值部分，我們用一 gaussian kernel 來取代。

(一) 系列材質合成

請此節中我們要解決合成所需系列材質的問題。我們要以一張輸入材質合成所需系列材質。而在合成各張材質時，我們希望每一張材質不會和前一張材質有太大的差異(difference)。材質合成的方法有很多，我們找到速度較快的有[7][8][13]。其中我們發現以固定大小區塊為主(patch-based)的方法較容易掌握。因此我們參考[8]的演算法，再加以修改。為了加速，我們也用了[8]所提到的 QTP(Quadtree Pyramid) + PCA(Principal Components Analysis) + KD-Tree 的方法。

我們的方法如下：

0. 以原方法合成第一張材質。
1. 從輸入材質中，亂數選一區塊(patch)，此區塊不與之前已合成材質的第一個區塊重複，將此區塊貼至下一張輸出材質的左下角。
2. 從輸入材質中找出一群與目前輸出材質內重疊邊界(overlapped edge)符合相似條件的區塊，這些為候選區塊，設有 k 個。
3. 於前一張合成材質中取出一比對區塊，此區塊的位置與目前欲貼上輸出材質中的位置相同，如圖 1 中的區塊 P。由步驟 2.所得的 k 個候選區塊中，挑出與上述比對區塊符合相似條件者，設有 k’個。
4. 從步驟 3.所得的 k’個區塊中，亂數選一個貼至目前輸出材質中。若 k’為 0，則找最相似者。
5. 重複步驟 2. 3. 4. 直到貼滿為止。
6. 若未達合成張數，則再回到步驟 1.繼續合成下一張。

在亂數取第一個區塊時，我們刻意避免和之前所用過的第一個區塊相同。以避免產生相同的輸出材質，且能維持各材質間一定程度的差異性。我們的合成結果如圖 7 所示，其中各畫面是以圖 6 為輸入材質合成的。

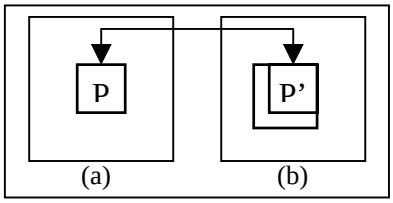


圖 1 (a)為前一張已合成的畫面，(b)為目前合成中的畫面。(b)中的 P’為目前欲貼入的區塊，(a)中的 P 為與(b)之 P’同位置的區塊。P’與 P 滿足 L2Norm(P, P’) < δ*d 或 P’是候選區塊中與 P 最近者，其中δ*d 為距離容忍度(Distance Tolerance)。

(二) 重排順序

此節中我們要將所得的合成材質的前後順序重排，以降低連續兩張畫面間的差異。在(一)小節的合成步驟時，我們已注意到連續性(continuity)的維持問題，即每張合成材質和前一張不致於有太大差異。但我們要先考慮到區塊在輸出材質中重疊邊界吻合的條件，因此連續兩張輸出材質間的連續性未必是好的。

我們希望在以(一)小節的方法所合成的材質中再找出具較佳連續性的順序，即每秀一張畫面時和前一張畫面不致於有太大差異。此差異值可用 L2Norm 距離來表示。因此我們希望

在播放順序中每張畫面與其下一張畫面有較近的距離。我們發現可以借用近似值演算法(Approximation Algorithm)中巡迴銷售員問題(Traveling-Salesman Problem)的解法。因為該方法可以在一個加權的完全圖(weighted complete graph)中找出最短的漢米敦路徑(minimum cost Hamilton path)。我們把每個畫面當成該完全圖中的頂點(vertex)，而各畫面間的距離以 L2Norm 的方式計算。所找到的漢米敦路徑正好就是我們所想排列的畫面順序。

我們重新排列畫面順序的方法如下：

1. 找到具有最短距離的兩個畫面，以原順序中較前者為起點。
2. 找下一個距離最近且未拜訪過的畫面，再以該畫面為下一個起點。
3. 重複步驟 2，直到連接所有畫面為止。

整體來說原演算法所找到的路徑未必是最短的。另我們所合成的各張材質中不一定每張都是好的，即不一定每張和其他各張都具有良好的連續性。我們針對上述兩個問題加以改進。設所需合成張數為 n。我們先以(一)小節的方法合成 m 張材質，m > n。以每張材質為起點，以原來的近似值演算法，各別找出長度為 n-1，即連接 n 張材質的 m 條最短路徑。在這 m 條路徑中選出距離總和最小者即為所選。我們可取各材質的 gaussian pyramid 以加速計算，因我們不需要知道實際距離，而只是要比較距離大小。

改進之後，如圖 2 示，經篩選後 RMS(Root Mean Square)的變化較小，其平均值也降低了。圖 2(a)為直接合成 10 張再重排的情形，其重排後 RMS 變化平均為 70.736。圖 2(b)為合成 20 張、篩選 10 張再重排的情形，其重排後 RMS 變化平均為 70.339。以此所得到的畫面順序即為原始播放順序(original rendering sequence)。如圖 7 及圖 8 所示，經我們重排畫面順序後，連續兩畫面間的 RMS 值明顯降低了。此方法確實可以減少連續兩張畫面間的差異。

(三) 計算重播機率

本節中我們要介紹計算重播條件的方法以能任意延伸播放時間。利用(二)小節中的方法，我們已可產生原始播放順序，為了要任意延伸播放時間，我們要在此播放順序中找到某些適當的重播點。如圖 3 所示，其中 f0, f1, … ,fj, fj+1, … fk, … 為原始播放順序為，在適當條件下播放完畫面 fk 之後可跳回畫面 fj，再由此往後重播。我們模仿[10]的方法，並加以修改以配合我們的原始播放順序的特性。

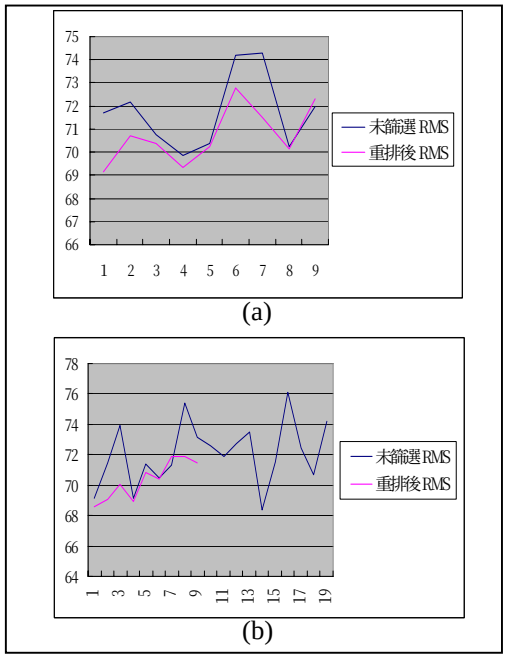


圖 2 以圖 6 中的材質合成所需材質的 RMS 變化折線圖，其中藍色線條表示未篩選前的情形，紫色線條表示重排後的情形。(a)為合成 10 張直接重排的情形，其重排後 RMS 平均為 70.736。(b)為合成 20 張，篩選 10 張再重排的情形，其重排後 RMS 平均為 70.339。經篩選後 RMS 的變化較小，其平均值也降低了。

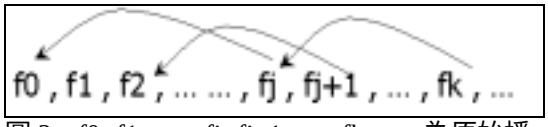


圖 3 f0, f1, … ,fj, fj+1, … fk, … 為原始播放順序。在適當條件下，播放完 fk 之後，順序可跳回 fj，j < k，於是可由 fj 起重播。

我們模仿[10]的方法計算機率矩陣。如(1)所示，其中 P_{ij} 為由畫面 fi 跳至畫面 fj 的機率，D_{ij} 則為畫面 fi 及畫面 fj 之間的距離。

$$P_{ij} \propto e^{\frac{-Dij}{\alpha \bullet \overline{Dij}}} \quad (1)$$

$$P_{ij} \propto e^{\frac{-Di+1,j}{\alpha \bullet \overline{Dij}}} \quad (2)$$

而原[10]中機率的計算公式如(2)所示。[10]文中認為若畫面 i+1 與畫面 j 相似，則可由畫面 i 跳回畫面 j。原來的順序是由畫面 i 接著畫面 i+1。我們之所以和[10]不同，是因為我們以(二)小節的方法所得的原始播放順序不是真實的錄影片段。我們的畫面 fi 及畫面 fi+1 不一定具有較好的連續性。

如此計算所得的機率值需再加以修改，因有時某 fi 及 fi+1 距離太近，以致於時常發生這兩張畫面來回重播的情形。為避免這種來回

振盪的情形發生，我們在機率矩陣中加強 $P_{i,i+1}$ 的機率成分，將其乘上一個加權值 W_i。修改之後，設由畫面 fi 跳至畫面 fi+1 的機率為 $P'_{i,i+1}$，其計算公式如(3)所示，其中 W_i 越大則 $P'_{i,i+1}$ 的值越接近 1。我們將新的機率矩陣中各列的機率值正規化，使得各列 $\sum_j P'_{i,j} = 1$。

$$P'_{i,i+1} = \frac{W_i * P_{i,i+1}}{W_i * P_{i,i+1} + \sum_{j \neq i+1} P_{i,j}} \qquad (3)$$

我們必須謹慎考慮如何決定公式(3)中 W_i 的值。我們可以看到在圖 8 中，以(二)小節的方法重排所得的原始播放順序中，越後面的連續兩張畫面的差異越大。因此播放的順序越往後時，我們希望跳回前面重播的機率越大，以避免有太大的差異出現。原始播放順序中較前面的連續兩張畫面間的差異較小，則越需要避免來回振盪的情形。所以我們希望越前面的順序中越要加強往後連續播放的機率，即該 $P'_{i,i+1}$ 要越大；而越後面的順序中其往後連續播放的機率則要越小。我們可以取 W_i 為一由大至小的曲線。而一次直線或二次曲線的值下滑的太快，因此我們取三次曲線，如圖 4 所示。最後 $W_{n-1} = 1$，即等於沒有加權。

我們取 W_i 為一三次函數 F(i)，如(4)所示，其中 n 為原始播放順序的畫面張數，w 為參數，可由使用者設定。其函數圖形如圖 4 所示。故 $W_0 = F(0) = w$，$W_{n-1} = F(n-1) = 1$。所以 $P'_{0,1} = \frac{w * P_{0,1}}{w * P_{0,1} + \sum_{j \neq 1} P_{0,j}}$，w 越大則 $P'_{0,1}$ 越接近 1，即由畫面 f0 接著播放畫面 f1 的機率就越高。

$$F(i) = A * i^3 + w \quad , \quad A = \frac{1-w}{(n-1)^3} \qquad (4)$$

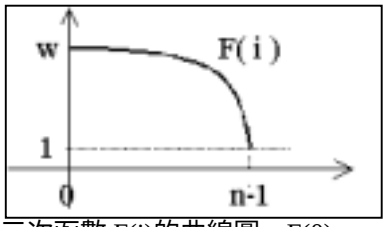


圖 4　三次函數 F(i)的曲線圖，F(0) = w，F(n-1) = 1

如圖 5 所示，以圖 7 所得的順序播放時的畫面編號變化情形。圖 5 中可看到隨著時間的進行播放順序的變化少有連續兩張畫面來回重複播放的情形發生。且如圖 10 所示，我們的播放過程中能將 RMS 的變化程度維持在一定範圍內，即播放時畫面很平穩的變化，少有較不連續的跳動情形。如此能任意延伸播放時間，且能維持動態材質半規律的特性(semi-regularity)，看不出來有固定的變化動作出現。

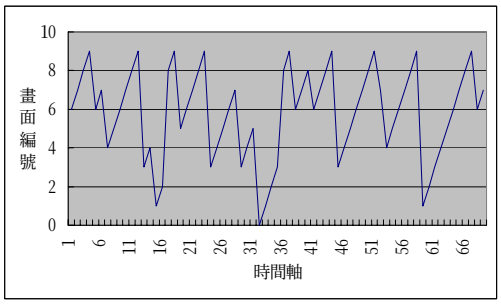


圖 5　以圖 7 所得順序播放時的畫面編號變化情形。縱軸刻度表示原始播放順序中的畫面編號，橫軸表示時間軸。在以公式(4)計算時，n = 10，我們取 w = 5。

四、播放

本節中要介紹能欺騙視覺的 cross-fading 播放方法。我們能減少連續兩張播放畫面間的差異，且有效的計算重播機率以能任意延伸播放時間。但，以各單張畫面播放時，仍然時常有明顯不連續變化的情形發生，即某畫面與前一張畫面差異太大。因此需另有方法再降低該差異性。

我們模仿[10]中的 cross-fading 方法，每次播放的畫面，是由前後多張畫面重疊組合而成。每個播放畫面的計算方式如(5)所示，其中 S_i 為一播放畫面顯示結果，T_i 為目前依 3.3 節所述的機率值求得的播放順序中的單張畫面。我們取 g(j)為一維的 gaussian kernel，如(6)所示。經 cross-fading 處理之後，我們有效的降低了連續播放畫面間的差異。如圖 10 所示，經 cross-fading 處理後，平均 RMS 值約降低了一半。

$$S_i(x,y) = \sum_j g(j) * T_{i+j}(x,y) \qquad (5)$$

$$g(j) \propto e^{\frac{-j^2}{2\sigma^2}} \quad , \quad \sum_j g(j) = 1 \qquad (6)$$

四、執行結果

本篇論文所使用的實驗平台為 AMD k7 850MHz、128MB RAM、作業系統為 Windows 98、程式撰寫工具為 Visual C++。我們以 RMS(Root Mean Square)來表示畫面差異的情形。RMS 定義如(7)所示，公式中 $G_i(x,y)$代表一畫面中一元素的色值。

$$RMS = \sqrt{\frac{1}{M \times N} \sum_{y=1}^{M} \sum_{x=1}^{N} \left(G_i(x,y) - G_j(x,y)\right)^2} \qquad (7)$$

我們以圖 6 中材質說明執行結果。我們將列出原始合成材質的畫面，並以紅色框線及數字標識出經篩選處理後所得的畫面及重排順序。接著展示原始合成的順序、篩選後及重排順序後的畫面差異變化情形。我們也列出執行時間，及從播放過程中所截取的一段畫面及其 RMS 變化情形。然後展示播放時的執行視窗畫面。

(一) 波浪材質測試

我們使用圖 6 為輸入材質，先以**三**.(一)節所述方法合成二十張輸出材質，大小皆為 288x288，如圖 7 所示。以三.(二)節的方法篩選及重排順序後，如圖 7 中以紅色框線框起來的圖即為所選，其左下角紅色數字表示重排後的順序。圖 8 說明原始合成順序及篩選前後的 RMS 變化情形。未篩選前之原始合成順序的 RMS 平均為 72.073，篩選後為 72.231，再重排後為 70.339。重排後，RMS 的平均有顯著的降低。

執行時間：合成原始二十張輸出材質所需時間為 102 秒，篩選及重排時間約 5 秒。播放時為即時顯示。

我們播放 10 秒鐘，此 10 秒片段中播放順序連續材質的 RMS 變化平均為 70.383，而經 cross-fading 處理後為 31.74。我們取出其中一段播放順序：9, 5, 6, 7, 8, 9, 3, 4, 5, 6, 7, 3, 4, 5, 0, 1，如圖 9 所示。

圖 10 顯示播放時原材質間的 RMS 的變化及經使用 cross-fading 處理之後的 RMS 的變化情形。使用 cross-fading 之後，RMS 平均約降低了一半，使連續畫面的差異降低，增加連續性但仍保有一定的差異程度以顯示動態的變化。

我們的播放介面如圖 11 所示。播放時可即時觀看 RMS 變化情形及播放順序的變化。由圖 10 中可看出我們能將 RMS 變化維持在一定範圍內。

(二) 其他材質執行結果

圖 12 中顯示其他各種材質的播放視窗畫面，各材質皆播放 10 秒。其各材質在執行過程中的 RMS 變化平均如表 1 及表 2 所示，而各執行時間如表 3 及表 4 所示。

我們可看到表 1 及表 2 中 RMS 變化情形，在合成材質的篩選前後變化不大。因在合成步驟已注意到維持連續兩張輸出材質間的相似性，故篩選前後、未重排前 RMS 平均變化不大。而在重排後，RMS 平均即有明顯降低。

而在播放階段，我們並不強制每個重播點一定要選最好的，因為我們要避免動態材質的重複動作頻率太高。但我們仍有加強連續順序的機率值，以避免忽前忽後的跳動情形。另，我們以 cross-fading 降低不連續的情形。我們可看到表 1 及表 2 中，使用 cross-fading 處理後，RMS 平均約降低一半。

在表 3 及表 4 所示的執行時間中，我們可看到主要時間花費在合成步驟。其中材質 (c)smoke 所花的合成時間明顯比其他材質多出很多，因為該材質本身色值分部較接近，色值相近的區塊較多，比對候選區塊的次數比其他材質多。

表 1 各階段 RMS 變化的平均值。①為合成二十張後。②為篩選十張後。③為重排順序後。④為播放時的連續單張材質間的 RMS 平均情形⑤為使用 cross-fading 處理後的 RMS 平均情形。

材質種類	①	②	③	④	⑤
(a)ocean	44.54	44.02	42.94	46.15	19.64
(b)fire	62.50	62.83	62.12	55.49	23.83
(c)smoke	29.77	30.20	28.94	33.29	11.53

表 2 各階段 RMS 變化的平均值。①為合成四十張後。②為篩選二十張後。③為重排順序後。④為播放時的連續單張材質間的 RMS 平均情形⑤為使用 cross-fading 處理後的 RMS 平均情形。

材質種類	①	②	③	④	⑤
(d)pond	34.94	34.47	33.80	36.09	15.77

表 3 輸入材質大小及各階段處理所需時間。①為合成二十張。②為篩選十張③為重排順序及計算重播機率。

材質種類	輸入材質大小	①	②	③
(a)ocean	200x200	82 秒	1 秒	4 秒
(b)fire	128x128	107 秒	1 秒	4 秒
(c)smoke	128x128	434 秒	1 秒	4 秒

表 4 輸入材質大小及各階段處理所需時間。①為合成四十張。②為篩選二十張③為重排順序及計算重播機率。

材質種類	輸入材質大小	①	②	③
(d)pond	128x128	204 秒	4 秒	17 秒

五、結論與未來工作

我們提出了一新的演算法能只利用一張材質來合成動態材質。我們成功的利用動態材質本身具有的半規律性，隨著時間的變化有相同的動作出現又有隨機的性質，不會連續重複相同的動作。我們示範了將一張波浪的材質以維持前後連續性的方式合成二十張擴大的材

質，篩選出十張，然後將其順序加以巧妙的重排。以此重排後的十張合成材質能產生無限時間的動態材質。我們也展示了其他材質的執行結果。我們列出一與其他動態材質合成演算法的比較表，如表 5 所示。一般來說，相信我們的方法是動態材質合成的一大進步。我們的方法有以下優點：1.只需要一張材質為來源，有別於以往需要錄影片段的情形。2.可很快的合成所需畫面，以大小 200x200 以內的輸入材質，輸出 288x288 大小的所需材質只需約 2~8 分鐘。如表 5 所示，我們合成較大的輸出畫面，而所需的平均時間最少。3.只需使用一般速度的 CPU 及一般容量的記憶體，我們使用 AMD k7 850MHz 及 128M RAM 的實驗平台。4.可合成不限大小的畫面，以符合貼圖所需。

限制：由於 cross-fading 的效果，會使得播放的結果具一定程度的模糊性。且我們的方法只適用於隨機性材質，且材質內容沒有固形狀者。

未來工作：1.加入方向性：使動態材質看起來是朝某方向移動，且可由使用者控制方向。例如河水除了水波浮動之外也會朝著某方向流動。2.增加即時性：雖然在播放時是即時的。但我們在合成各張材質、篩選、重排順序及計算機率時都要耗費一些時間。我們希望能直接即時的產生一張畫面，能與前一張有連續的效果。此時即可直接播放，而無需中間處理的過程。

表 5　與其他動態材質演算法的比較

比較項目	我們的演算法	Wei [12]	Bar-Joseph [6]	Kwatra [7]
材質來源	一張材質	錄影片錄	錄影片段	錄影片段
輸出畫面大小及合成時間	288 x 288 2 至 8 分鐘	150 x 112 超過 30 分鐘	256 x 256 約 10 分鐘	210 x 160 5 分鐘至 1 小時
所需記憶體	一般	一般	大量	一般
連續性	可	優	可	優
合成不限時間片段	可	無	無	可
合成不限大小片段	可	可	可	可

六、參考文獻

[1] S. Arya, D. M. Mount, N. S. Netanyhu, R. Silverman, and A. Y. Wu, “An Optimal Algorithm for Approximate Nearest Neighbor Searching,” *Journal of ACM*, Vol. 45, No. 6, pp. 891-923, 1998.

[2] M. Ashikhmin, “Synthesizing Natural Textures,” *In Proceedings of the ACM Symposium on Interactive 3D Graphics*, (March), pp. 217-226, 2001.

[3] N. Campbell, C. Dalton, D. Gibson and B. Thomas, “Pratical Generation of Video Textures Using the Auto-Regressive Process,” *British Machive Vision Conference*, pp. 434-443, Cardiff, UK. September 2002.

[4] G. Doretto, A. Chiuso, Y. N. Wu and S. Soatto, “Dynamic Textures,” *International Journal of Computer Vision*, 51(2), pp. 91-109, 2003.

[5] A. A. Efros and W. T. Freeman, “Image Quilting for Texture Synthesis and Transfer,” In *Proceedings of SIGGRAPH 2001*, pp. 341-346, August 2001.

[6] Z. B. Joseph, R. E. Yaniv, D. Lischinski, and M. Werman, “Texture Mixing and Texture Movie Synthesis Using Statistical Learning,” In *Proceeding of IEEE Transaction On Visualization And Computer Graphics*, Vol. 7, No. 2, pp. 120-135, June 2001.

[7] V. Kwatra, A. Schodl, I. Essa, G. Turk and A.Bobicks, “Graphcut Textures: Image and Video Synthesis Using Graph Cuts,” *ACM Transactions On Graphics*, Vol. 22, No. 3, pp. 277-286, July 2003.

[8] 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*, Vol. 20, No. 3, pp. 127-150, July 2001.

[9] K. Perlin, “Improving Noise,” *In Proceedings of SIGGRAPH 2002*, pp. 681-682, 2002.

[10] A. Schodl, R. Szeliski, D. H. Salesin and I. Essa, “Video Textures,” *In Proceedings of SIGGRAPH 2000*, pp. 489-498, July 2000.

[11] S. Soatto, G. Doretto, and Y. N. Wu, "Dynamic Textures," In *Proceedings of IEEE International Conference on Computer Vision 2001 (ICCV 2001)*, Vol. 2, pp. 439-446, July 2001.

[12] L. Y. Wei and M. Levoy, “Fast Texture Synthesis Using Tree-Structured Vector Quantization,” In *Proceedings of SIGGRAPH 2000*, pp. 479-488, July 2000.

[13] S. Zelinka and M. Garland, “Towards Real-Time Texture Synthesis with the Jump Map,” In *Proceedings of Eurographics Workshop on Rendering (EWR 2002)*, pp. 99-104, 2002.

圖 6　波浪，材質大小為 192x192

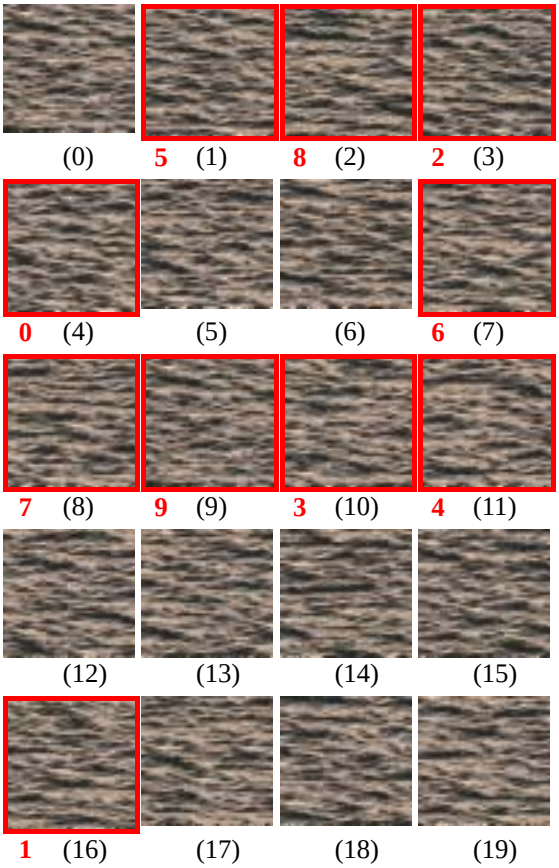


圖 7　以圖 6 為輸入材質，合成二十張大小為 288x288 的輸出材質。各畫面下方的黑色數字表示原始合成順序。以紅色框線框起來的為篩選所得的畫面，其左下方的紅色數字表示重排後的順序。

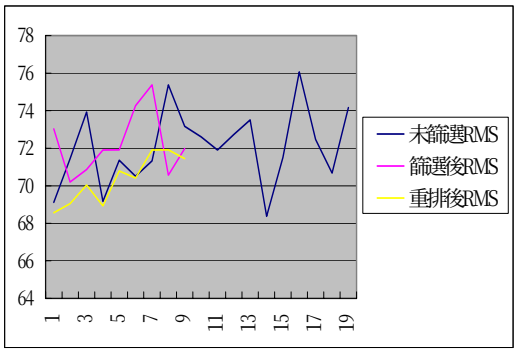


圖 8　藍色折線顯示二十張原始合成順序的 RMS 變化情形，紫色折線顯示篩選後 RMS 變化情形，黃色折線顯示重排順序後的 RMS 變化情形。

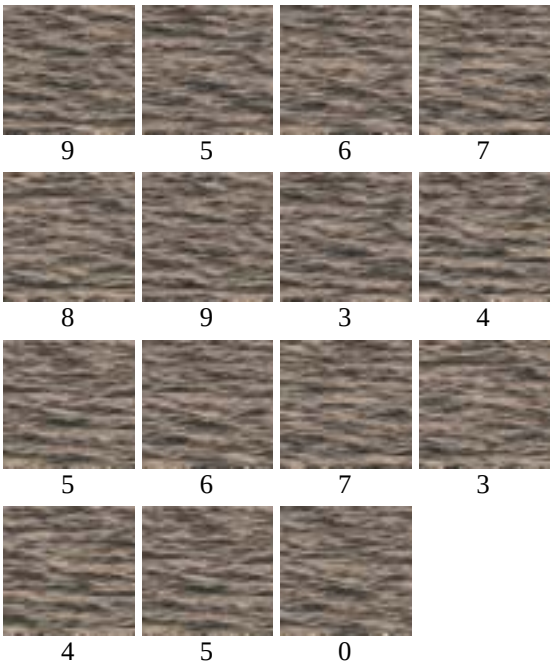


圖 9　我們以圖 7 中重排過的材質順序播放，截取其中連續顯示的十五張畫面。畫面已使用 cross-fading 處理過。

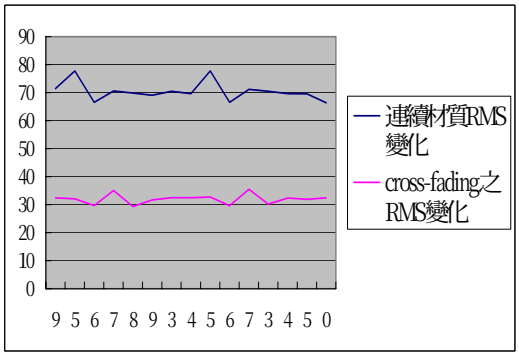


圖 10　顯示圖 9 中播放片段連續材質的 RMS 變化及經 cross-fading 處理後的 RMS 變化情形。經 cross-fading 處理之後，平均 RMS 約降低一半。

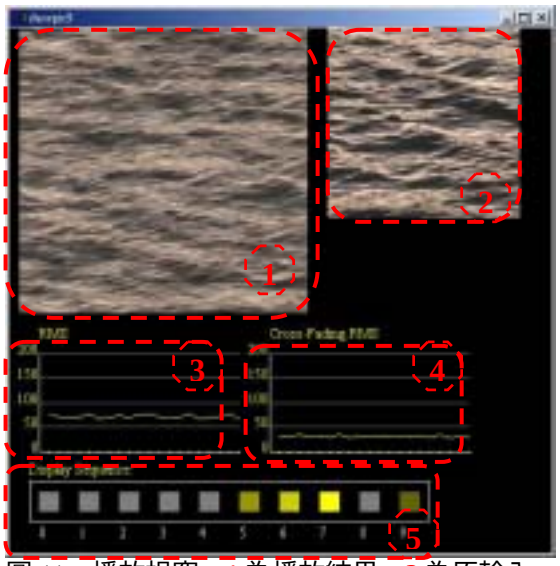
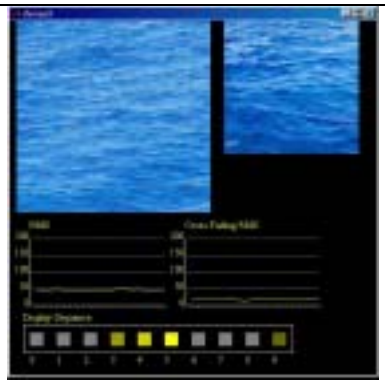
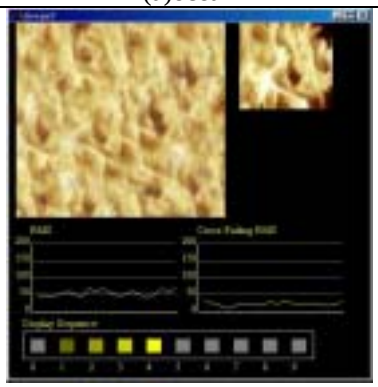


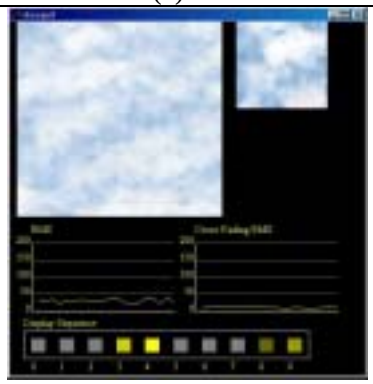
圖 11　播放視窗。**1.**為播放結果。**2.**為原輸入材質。**3.**為按播放順序的原始材質間的 RMS 變化折線圖。**4.**為播放時經 cross-fading 處理過的 RMS 變化折線圖。**5.**顯示目前播放順序，最亮的黃色方塊表示最近播放的序號。**1.3.4.5.**可即時顯示各種變化情形。



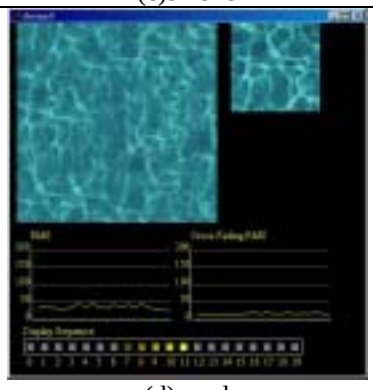
(a)ocean



(b)fire



(c)smoke



(d)pond

圖 12　其他材質播放視窗。輸出材質大小皆為 288x288。(a)(b)(c)各合成二十張，篩選十張。(d)合成四十張，篩選二十張。