一個使用群集化與最大誤差降低量技巧的快速向量量化碼簿設計演算法
An Efficient Algorithm for Vector Quantization Codebook Design Using Clustering and Maximum Descent Techniques

劉于禎 Yu-Chen Liu
王廷基 Ting-Chi Wang

中原大學資訊工程學系
Department of Information and Computer Engineering
Chung Yuan Christian University
Chungli, Taiwan

摘要

本論文針對向量量化碼簿設計問題提出一個快速演算法稱為 ClusterMD。首先，ClusterMD 演算法利用一個群集化的演算法[16]，將待訓練向量分割成數個互斥的群集，而每一個群集必須滿足使用者事先指定的誤差限制。然後，ClusterMD 演算法延伸 MD 演算法[17]的概念將這些群集分割成與碼簿中碼向量個數相同個數的集合，而每一個集合的質量中心即為碼簿中的一個碼向量。我們已經將 ClusterMD 演算法以 C 語言實現並與其他演算法比較，由實驗結果得知，ClusterMD 演算法在碼簿品質上與 ClusterLBG 演算法[16]、MD 演算法雖互有勝負但相差不多，但是執行時間卻是最短；整體而言，ClusterMD 演算法較 MD 演算法少 31~47%的執行時間，而較 ClusterLBG 演算法少 69~86%的執行時間。

關鍵字：向量量化、碼簿設計、群集化、最大誤差降低量、演算法

Abstract

We present in this paper an efficient algorithm, called ClusterMD, for vector quantization codebook design. First, the ClusterMD algorithm applies a clustering algorithm [16] to partition the set of training vectors into a disjoint set of clusters each of which must satisfy a user-specified distortion constraint. Then, the ClusterMD algorithm extends the idea of the MD algorithm [17] to further partition the set of clusters into a certain number of groups each of whose centroid becomes a final codevector. The ClusterMD algorithm has been implemented in C language and compared with several other codebook design algorithms. The experimental results indicate that when comparing the ClusterMD algorithm with the ClusterLBG algorithm [16] and the MD algorithm, the ClusterMD algorithm could generate comparable codebooks while running much faster. Overall, the run time of the ClusterMD algorithm is 31-47% and 69-86% less than the MD algorithm and the ClusterLBG algorithm, respectively.

Keywords: vector quantization, codebook design, clustering, maximum descent, algorithms.

1. 前言

向量量化(vector quantization)一直是一個有效且重要的影像與語音資料的壓縮技術[1,2]。向量量化的概念是將維度(dimension) k 的一個向量(training vector)的集合經一映射過程對應到另一個含有 N 個向量的集合，此集合被稱為碼簿(codebook)，而碼簿中的向量稱為碼字元(codeword)或碼向量(codevector)。

碼簿設計問題是向量量化的關鍵問題之一[1,2,3]。一種常被用來解決碼簿設計的方法是將待訓練向量分成 N 個互斥群組，然後求出每一個群組的質量中心(centroid)以做為碼簿中的一個碼向量。因此，隨著產生分割結果的方式不同，已經有許多演算法被提出。例如：著名的 LBG 演算法[4]就是從一個給定的初始碼簿開始，採用不斷反覆改進(iterative improvement)的技術，不斷地以前一次的分割結果做為基礎來產生下一次更好的分割結果，直到收斂條件滿足為止。儘管 LBG 演算法非常容易實現，但是執行時間太長且最終所得到的碼簿只趨近於區域最佳解(local optimum)。

為了得到較佳的碼簿品質，有人提出 SA(simulated annealing)的方法[5,6]，但其執行時間卻遠比 LBG 要來得久，同時在碼簿改良上也沒有重大突破。此外，為了減少計算時間，有人提出子空間失真(subspace distortion)的方法[7]，這種方法主要是降低向量的維度數以縮短執行時間，但其碼簿品質卻避於或相同於 LBG 演算法。另外，也有人提出 PNN(pairwise nearest neighbor)的演算法[8]，此方法主要是不斷地合併(merge)鄰近向量直到產生指定數目的碼向量為止，其所需的時間較 LBG 短，但在碼簿的品質上卻略微下降。另外，參考文獻[9,10,11,12,13,14,15]中所提出的數種演算法也能縮短執行時間但碼簿品質都不超過 LBG 演算法。

在參考文獻[16]中，作者提出了一個新的碼簿設計方法稱為 ClusterLBG，其做法是採用群集化(clustering)的技術來改進 LBG 演算法。此方法包含兩個步驟：第一個步驟是先將所有待訓練向量分割

成若干群集，而每個群集的平均誤差不能超過一指定值。第二個步驟則是將每個群集視作一向量而借用 LBG 的觀念來反覆不斷改善群集分割 (cluster partitioning)結果，其中還加上積分映射(integral projections)技巧[15]來加快編碼速度。由於，事先做群集化的步驟可將應該對應到相同碼向量的向量儘早放置在同一群集中並確保分割品質，因此不但縮短了 LBG 演算法(即第二步驟)所需的時間，同時亦改善 LBG 的碼簿品質。

此外，參考文獻[17]的作者捨棄以往以反覆不斷產生多個分割結果的做法(如 LBG 演算法)，而提出只會產生一種分割結果之由上往下分割法(top-down partitioning)稱為 MD 演算法。MD 演算法的概念是將所有待訓練向量視為一群集，反覆地從目前產生出來的群集中選擇一具有最大誤差降低量(maximum descent)的群集加以分割直到群集個數等於所需的碼字元數為止。其中，為了將一群集分割成兩子群集，作者發展出兩種超平面(hyperplane)分割方法和一種使用 LBG 演算法進行快速分割的方法，而實驗結果證明這三種分割方法產生的碼簿均比 LBG 演算法好並且所需之執行時間也較少。雖然，這三種分割方法的碼簿品質都差不多，但是採用快速 LBG 分割方法卻比其他兩種都快得多。為了方便說明，本論文往後凡是提到的 MD 演算法皆指使用快速 LBG 分割方法之 MD 演算法。

在本篇論文中我們提出一個快速的碼簿設計演算法稱為 ClusterMD。此方法採用 ClusterLBG 演算法中表現優異的 Clustering 演算法，先將待訓練向量有效且快速地分割成數個互斥群集，使整個問題的大小從原有向量個數降低為群集的個數；接下來，再延伸 MD 的概念進行群集分割。我們已將 ClusterMD 演算法以 C 語言實現並與其他演算法比較，而由實驗結果得知 ClusterMD 演算法在碼簿品質上與 ClusterLBG 演算法、 MD 演算法雖互有勝負但相差不多，但是執行時間卻是最短；整體而言，ClusterMD 演算法較 MD 演算法少 31~47%的執行時間，而較 ClusterLBG 演算法少 69~86%的執行時間。

本論文往後的章節內容將安排如下：第 2 節介紹向量量化碼簿設計問題。第 3 節就我們所提出的 ClusterMD 演算法做一詳細說明。第 4 節將提出實驗結果。最後，在第 5 節我們將對本論文下一些結論。

2. 問題描述

向量量化器可視為一映射函數 Q，它將維度 k 的實數空間內的一個子集合 X 映射在維度 k 的實數空間內的另一個子集合 Y，亦即 $Q: X \to Y$，此處 $X=\{x_i \mid i=1,2,...,M; x_i \in R^k\}$ 代表 M 個待訓練向量所成的集合，而 $Y=\{y_i \mid i=1,2,...,N; y_i \in R^k\}$ 代表由 N 個碼向量所成的碼簿。若令 $d(x,y)$ 代表兩向量 x 和 y 之間的誤差且定義成歐幾里得距離平方(squared Euclidean distance)，則向量量化器 Q 將任一訓練向量 x_i 編碼成碼向量 $Q(x_i)$，其定義為 $Q(x_i)=\arg\min_{y_j \in Y} d(x_i,y_j)$。

本論文所討論的碼簿設計問題定義如下：給定一個包含 M 個 k 維的待訓練向量之集合 X，此問題的目標是求出一包含 N 個 k 維的碼向量之碼簿 Y，使得以 Y 來對 X 編碼所造成的總誤差，即 $\sum_{i=1}^{M} d(x_i,Q(x_i))$，愈小愈好。在本論文中，將以 k 代表向量的維度。

3. ClusterMD 演算法

在詳述 ClusterMD 演算法之前，我們必須先簡要回顧 Clustering 演算法[16]以及 MD 演算法[17]。

3.1 Clustering 演算法

ClusterLBG 演算法[16]使用類似建立一棵 k-D tree 的方式來執行群集化，此方法稱為 Clustering 演算法。此方法將所有的待訓練向量視為一群集，然後再遞迴地將一個群集分割成兩子群集，直到所有群集的平均誤差(亦即群集的每個向量對其質量中心的距離平方和除以群集內向量個數)均滿足事先給定之誤差量為止。

每次在分割一個群集成兩子群集之前，必須先從該群集所含向量選出變異量(variance)最大的一個維度作為分割軸，然後再求出以所有向量在該分割軸上的平均值做為分割臨界值。一旦分割軸及分割臨界值求出之後，群集中的所有向量便依據其分割軸上的座標被分割成大於或等於分割臨界值及小於分割臨界值的兩子群集。為了快速完成群集化的動作，Clustering 演算法僅藉一個軸來區分多維資料，如此一來群集化的正確性便降低，為了克服這個問題，產生的群集個數通常是控制在訓練向量個數的四分之一左右。

3.2 MD 演算法

MD 演算法是屬於一種由上到下的分割方式，一開始將所有待訓練向量視為一群集並利用 LBG 演算法將其分割成兩群集及求出分割後總誤差之降低量。接著重覆執行下面分割步驟直到分割出來之群集個數達到所需之碼向量個數為止。在每一次的分割步驟中，從目前所產生之群集中，選擇一個在分割後具有最大誤差降低量(maximum descent)之群集(此選擇動作可以使用 binary heap 的資料結構來加快尋找速度)並利用事先已決定之分割方式將其實際分割為二，然後再對新產生之兩個群集分別利用 LBG 演算法將其分割並求出誤差降低量。在結束所有分割步驟之後，求出每一群集內所含向量之質量中心，並將其設定成所求之一個碼向量。

每個群集被分割後其誤差降低量被定義如下。令 S_i 代表任一群集，以 $D(S_i)$ 代表 S_i 中所有向量與其質量中心 v_i 的誤差總和，即 $D(S_i)=\sum_{x \in S_i} |x-v_i|^2$。假設 S_i 可被分割成 S_{ia} 和 S_{ib}，並分別以 n_i、n_{ia} 及 n_{ib} 代表群集 S_i、S_{ia} 及 S_{ib} 所含的向量個數及以 v_i、v_{ia} 及 v_{ib} 代表 S_i、S_{ia} 和 S_{ib} 的質量中心。則因分割群集 S_i 成子群集 S_{ia} 和 S_{ib} 後所造成的誤差降低量 $\Delta D(S_i)$ 被定義如下：

$$\Delta D(S_i)=D(S_i)-[D(S_{ia})+D(S_{ib})]$$

上式可化簡成[17]：

$$\Delta D(S_i)=n_i \frac{n_{ia}}{n_{ib}} |v_i-v_{ia}|$$

由於將 S_i 分割成 S_{ia} 和 S_{ib}，是利用 LBG 演算法求出，因此，S_{ia} 和 S_{ib} 在每次疊代之後皆會改變。而在重新分配 S_i 中每一個向量 x 到新的 S_{ia} 或 S_{ib} 之前，必須先計算 x 與前次 S_{ia} 及 S_{ib} 質量中心之距離平方，即 $d(x,v_{ia})=\Sigma_{j=1}^{k}|x_j-v_{ia_j}|^2$ 和 $d(x,v_{ib})=\Sigma_{j=1}^{k}|x_j-v_{ib_j}|^2$。如果 $d(x,v_{ia})<d(x,v_{ib})$ 成立的話，則向量 x 將被分到新的群集 S_{ia}，反之則分到新的群集 S_{ib}。按照 LBG 演算法的做法，在計算 $d(x,v_{ia})$ 及 $d(x,v_{ib})$ 總共需 $4k-2$ 個加法、$2k$ 個乘法及 1 次比較。但是為了節省計算時間，MD 演算法採用一個較快速的方式來判斷 x 應該被分到那一個群集。令 $d(x,v_{ia},v_{ib})=d(x,v_{ia})-d(x,v_{ib})$，則

$$\begin{aligned}d(x,v_{ia},v_{ib}) &= \Sigma_{j=1}^{k}\left|x_j-v_{ia_j}\right|^2-\Sigma_{j=1}^{k}\left|x_j-v_{ib_j}\right|^2\\ &=\Sigma_{j=1}^{k}\left[(x_j-v_{ia_j})^2-(x_j-v_{ib_j})^2\right]\\ &=-2\Sigma_{j=1}^{k}(v_{ia_j}-v_{ib_j})x_j+\Sigma_{j=1}^{k}({v_{ia}}_j^2-{v_{ib}}_j^2)\end{aligned}$$

如果 $d(x,v_{ia},v_{ib})$ 小於零，則向量 x 將被分到新的群集 S_{ia}，反之則分到新的群集 S_{ib}，因此，須 1 次比較。由於 $(v_{ia_j}-v_{ib_j})$ 和 $\Sigma_{j=1}^{k}({v_{ia}}_j^2-{v_{ib}}_j^2)$ 在同一次疊代過程中並不會改變，故只需計算一次；因此計算 $d(x,v_{ia},v_{ib})$ 只需 $k-1$ 個加法、k 個乘法。

3.3 ClusterMD 演算法

接下來將詳細介紹我們所提之 ClusterMD 演算法。ClusterMD 演算法包含兩個步驟，步驟一是向量的群集化，即是將待訓練向量分割成若干個群集，而每一個群集的平均誤差必須小於或等於一事先給定之指定值。如果群集化能夠快速且儘早將原本就會被對應到相同碼向量之訓練向量放置在同一群集中的話，則便能確保往後步驟二中群集分割的品質不致受到群集化的影響而仍保有或甚至超越原向量分割問題的品質。因此，一個快速且有效的群集化演算法便成 ClusterMD 演算法成功的最主要關鍵。在參考文獻[16]中作者提出了一個使用群集化技巧的碼簿設計演算法稱為 ClusterLBG 演算法，而從實驗數據得知 ClusterLBG 演算法確實能大幅改進 LBG 演算法的執行速度以及碼簿品質。由此可知其所使用之群集化演算法，稱為 Clsutering，能有效且快速地實現向量的群集化，故 ClusterMD 演算法的第一步驟亦採用 Clustering 演算法來進行向量的群集化。同時，在進行群集化的過程中，亦將每一群集的質量中心和所含向量個數記錄下來，供第二步驟使用。

至於 ClusterMD 演算法的第二步驟則是延伸 MD 演算法的觀念將每一群集視做向量而進行分割工作，並進而設計出碼簿來。步驟二首先將步驟一求出之所有群集放在一個集合並重覆執行類似 MD 演算法之分割步驟(需做修改並詳述於後)直到分割出來之集合個數達到所需之碼向量個數為止。需注意的是在分割的過程中，同一群集內之所有向量永遠不會在任何一次的分割步驟後被分到不同之集合。接下來，我們將詳述如何定義一個由群集所組成的集合在被分割成兩個子集合後所產生之誤差降低量、以及在 LBG 演算法的每一次疊代過程中如何快速判斷每個群集該分到那一個子集合。

令 C_i 代表任一含有數個群集的集合，以 cv_i 代表 C_i 中所有向量之質量中心，其可由下式快速求出：$cv_i=1/m_i\,\Sigma_{S\in C_i}(|S|\times cv(S))$，此處 S 代表 C_i 所含之任一群集、$|S|$代表 S 所含之向量個數、$cv(S)$代表 S 所含之向量之質量中心並已在步驟一求得、m_i 代表 C_i 所含之向量個數。令 $D(C_i)$代表 C_i 中所有向量與其質量中心的誤差總和，即 $D(C_i)=\Sigma_{S\in C_i}\Sigma_{x\in S}\|x-cv_i\|^2$，此處 x 代表群集 S 所含之任一向量。假設 C_i 可被分割成兩個子集合 C_{ia} 和 C_{ib}，而分別以 m_i、m_{ia} 及 m_{ib} 代表 C_i、C_{ia} 和 C_{ib} 所含向量個數且以 cv_i、cv_{ia} 及 cv_{ib} 代表 C_i、C_{ia} 和 C_{ib} 的質量中心。則將 C_i 分割成 C_{ia} 和 C_{ib} 後，其誤差降低量 $\Delta D(C_i)$ 被定義並計算如下：

$$\begin{aligned}\Delta D(C_i) &= D(C_i)-\left[D(C_{ia})+D(C_{ib})\right]\\ &=\Sigma_{S\in C_i}\Sigma_{x\in S}\|x-cv_i\|^2\\ &\quad-\left[\Sigma_{S\in C_{ia}}\Sigma_{x\in S}\|x-cv_{ia}\|^2+\Sigma_{S\in C_{ib}}\Sigma_{x\in S}\|x-cv_{ib}\|^2\right]\\ &=m_i\frac{m_{ia}}{m_{ib}}\|cv_i-cv_{ia}\|^2\end{aligned}$$

此外，為了使 LBG 演算法在每一次疊代過程中能快速判斷 C_i 中每個群集 S 該分到集合 C_{ia} 或是集合 C_{ib}，其做法如下。令 $D(S,cv_{ia},cv_{ib})=D(S,cv_{ia})-D(S,cv_{ib})$ (此處 $D(S,cv_{ia})=\Sigma_{x\in S}d(x,cv_{ia})$、$D(S,cv_{ib})=\Sigma_{x\in S}d(x,cv_{ib})$)，並以下列公式求出：

$$D(S,cv_{ia},cv_{ib})=|S|\left(-2\Sigma_{j=1}^{k}(cv_{ia_j}-cv_{ib_j})v_j+\Sigma_{j=1}^{k}({cv_{ia}}_j^2+{cv_{ib}}_j^2)\right)$$

此處的 $(v_1,v_2,...v_k)$ 代表 S 中所有向量的質量中心。若 $D(S,cv_{ia},cv_{ib})$ 為負值，則 S 歸集合 C_{ia}，反之，則歸集合 C_{ib}。在同一次疊代過程中，$(cv_{ia_j}-cv_{ib_j})$ 和 $\Sigma_{j=1}^{k}({cv_{ia}}_j^2+{cv_{ib}}_j^2)$ 不會改變故僅需計算一次即可。因此，對每個群集 S 該歸到集合 C_{ia} 或是集合 C_{ib} 的判斷，所需運算可降為 $k-1$ 次加法、k 次乘法及 1 次比較。

由 3.2 節可知，n 個向量以 MD 演算法來分割時，在 LBG 演算法的每次疊代中需要 $(n(k-1)+2k)$ 次加法、$(n+2)k$ 次乘法及 n 次比較；但是，若相同的向量以 ClusterMD 演算法先行群集化成 l 個群集再使用 LBG 演算法分割時，則每次疊代僅需 $(l(k-1)+2k)$ 次加法、$(l+2)k$ 次乘法及 l 次比較。由於 l 較 n 來得小，因此，ClusterMD 演算法執行時間預期會比 MD 演算法來得少。

4.實驗結果

我們已將 ClusterMD 演算法以及 FastLBG[15](使用積分映射技巧來改善 LBG 編碼速度)、ClusterLBG 及 MD 等其他三種碼簿設計演算法分別以 C 語言實作並且在 SUN SPARC 10 工作站上執行。採用 Lena、Peppers 和 F16(如圖 1~3 所示)三張 512×512 大小的影像作為測試資料，並設計下列兩種實驗模式來比較四種演算法在碼簿品質以及執行時間的表現。

在實驗模式一中，我們分別以這三張影像作為待訓練向量設計出個別的碼簿，然後再利用產生的碼簿對該影像做編碼以求出 PSNR 值。實驗模式二則是把 Peppers 和 F16 合併成為一張影像並當做待訓練向量資料來設計碼簿，然後再利用產生的碼簿分別對該影像及 Lena 做編碼以求得個別 PSNR 值。

在任一種實驗模式中，向量維度皆設為 16(即 4×4)，而碼簿所含之碼向量數則分別有 256、512 及 1024 三種不同大小。

圖 1：Lena 原圖。

圖 2：Peppers 原圖。

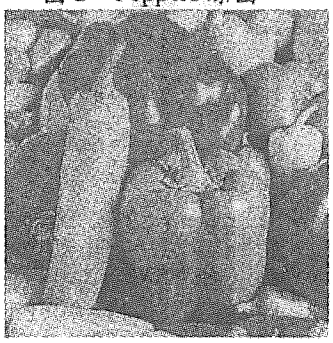

圖 3：F16 原圖。

此外， Clustering 演算法在執行群集化時需事先指定一個群集平均誤差的上限值，當此上限值越小則執行群集化所需之時間就越久，而品質也越好；反之，越小則時間越快，但品質較差。在實驗中，我們將其設為 200。

表 1~3 記錄實驗模式一的結果，其中 ClusterMD 演算法對 Lena、Peppers 及 F16 在進行 Clustering 動作分別需要 7.10、7.50 及 6.65 秒的時間(已包含於 ClusterMD 的執行時間)。從這些數據可以看出 ClusterMD 演算法在碼簿品質與速度上均明顯較 FastLBG 演算法好很多。另外，在碼簿大小為 256 時，ClusterMD 演算法的碼簿品質平均僅比 MD 演算法少 0.01db，但速度則快了約 33%；而當碼簿大小為 512 時，品質較 MD 演算法差 0.07db，但速度則快約 47%。因此，當碼簿較大時，可考慮將 Clustering 演算法中群集平均誤差的上限值降低，則可使品質再提升。此外，若與 ClusterLBG 演算法比較，則 ClusterMD 演算法平均可快 69~86%，但是在碼簿品質的比較上，當碼簿大時，ClusterMD 演算法較好，但當碼簿小時，則 ClusterLBG 演算法較好。整體而言， ClusterLBG、MD 以及 ClusterMD 演算法在碼簿品質上差不多，但 ClusterMD 演算法卻能保證執行時間為最短。

表 1： 實驗模式一中，碼簿大小為 256 時的結果。

		Lena	F16	Peppers	平均
FastLBG	PSNR	32.64	30.21	31.70	31.52
	Time(sec)	201.06	141.91	156.31	166.43
ClusterLBG	PSNR	32.82	31.08	32.65	32.18
	Time(sec)	43.91	48.86	46.83	46.53
MD	PSNR	32.70	31.14	32.62	32.15
	Time(sec)	21.57	20.62	21.80	21.33
ClusterMD	PSNR	32.70	31.15	32.57	32.14
	Time(sec)	13.83	14.38	14.93	14.38

表：2 實驗模式一中，碼簿大小為 512 時的結果。

		Lena	F16	Peppers	平均
FastLBG	PSNR	33.64	31.37	33.13	32.71
	Time(sec)	348.07	245.07	274.41	289.18
ClusterLBG	PSNR	33.88	32.25	33.56	33.23
	Time(sec)	61.45	80.78	66.95	69.73
MD	PSNR	33.91	32.37	33.79	33.36
	Time(sec)	23.38	23.23	25.00	23.87
ClusterMD	PSNR	33.85	32.29	33.73	33.29
	Time(sec)	14.90	15.33	15.77	15.33

表 3： 實驗模式一中，碼簿大小為 1024 時的結果。

		Lena	F16	Peppers	平均
FastLBG	PSNR	34.51	32.38	34.17	33.69
	Time(sec)	406.02	434.98	437.57	426.19
ClusterLBG	PSNR	35.08	33.55	34.64	34.42
	Time(sec)	96.41	107.63	101.03	101.69
MD	PSNR	35.35	33.98	35.17	34.83
	Time(sec)	25.52	25.83	26.90	26.08
ClusterMD	PSNR	35.24	33.91	35.09	34.75
	Time(sec)	15.63	16.40	16.03	16.02

表 4~6 則為實驗模式二的結果。對於內部編碼(對原影像編碼部份)而言，以 MD 演算法品質表現最佳，ClusterMD 演算法次之(平均僅差 0.05dB)，接下來才是 ClusterLBG、FastLBG 演算法。而在速度上，以 ClusterMD 演算法最快，MD 演算法次之，FastLBG 演算法最慢。至於外部編碼部份(對 Lena 影像編碼部份)，則表現出不穩定的情況，並不一定以 FastLBG 演算法為最差或以 MD 演算法為最好，如表 6 中 FastLBG 演算法以 Peppers+F16 為訓練資料產生碼簿再對 Lena 編碼後的品質為四者中最好。

綜合實驗模式一及模式二的結果來看，並無一演算法能保證在任何情況下所造出的碼簿品質皆為最佳，但是，毋庸置疑的是我們所提出來的 ClusterMD 演算法卻能保證其執行速度最快，而且其所造出的碼簿亦有相當好的品質。

表 4： 實驗模式二中，碼簿大小為 256 時的結果。

		Peppers+F16	Lena
FastLBG	PSNR	30.03	30.01
	Time(sec)	509.11	
ClusterLBG	PSNR	31.31	31.12
	Time(sec)	99.08	
MD	PSNR	32.01	30.88
	Time(sec)	43.55	
ClusterMD	PSNR	31.90	30.85
	Time(sec)	30.15	

表 5： 實驗模式二中，碼簿大小為 512 時的結果。

		Peppers+F16	Lena
FastLBG	PSNR	31.54	31.23
	Time(sec)	539.80	
ClusterLBG	PSNR	32.26	31.72
	Time(sec)	146.69	
MD	PSNR	32.86	32.13
	Time(sec)	48.165	
ClusterMD	PSNR	32.86	32.12
	Time(sec)	32.28	

表 6： 實驗模式二中，碼簿大小為 1024 時的結果。

		Peppers+F16	Lena
FastLBG	PSNR	32.64	31.80
	Time(sec)	824.35	
ClusterLBG	PSNR	33.26	31.72
	Time(sec)	232.29	
MD	PSNR	33.81	31.55
	Time(sec)	52.58	
ClusterMD	PSNR	33.76	31.57
	Time(sec)	33.38	

5.結論

本論文針對向量量化碼簿設計問題提出一個快速演算法稱為 ClusterMD。由實驗結果得知，ClusterMD 演算法在編碼品質上與 ClusterLBG 演算法及 MD 演算法雖互有勝負但相差不多，然而執行時間卻是最短；整體而言，ClusterMD 演算法較 MD 演算法少 31~47%的執行時間，而較 ClusterLBG 演算法少 69~86%的執行時間。

如何進一步針對 ClusterMD 演算法在加快執行速度的同時也能再提升碼簿品質將是我們未來的研究目標。

參考文獻

[1] Gresho and R. M. Gray, *Vector Quantization and Signal Compression*, 1992, Kluwer Academic Publiahers.

[2] N. M. Nasrabadi and R. A. King, "Image Coding Using Vector Quantization: a Review," *IEEE Trans. on Communications*, Vol. 36, No. 8, 1988, pp. 957-971.

[3] N. Akrout, R. Prost and G. Goutte, "Image Compression by Vector Quantization: a Review Focused on Codebook Generation," *Image and Vision Computing*, Vol. 12, No. 10, 1994, pp. 627-637.

[4] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communications*, January 1980, pp. 86-95.

[5] J. K. Flanagan, D. R. Morrell, R. L. Frost, C. J. Read and B. E. Nelson, "Vector Quantization Codebook Generation Using Simulated Annealing," *Proc. ICASSP*, 1989, pp. 1759-1762.

[6] N.-A. Lu and D. R. Morrell, "VQ Codebook Design Using Improved Simulated Annealing Algorithms," *Proc. ICASSP*, 1991, pp. 673-676.

[7] L. M. Po and C. K. Chan, "Subspace Distortion Measurement Techniques for Efficient Implementation of Image Vector Quantization," *Electronics Letters*, March 1990, pp. 480-482.

[8] W. H. Equitz, "A New Vector Quantization Clustering Algorithm," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1989, pp. 1568-575.

[9] C. D. Bei and R. M. Gray, "An Improvement of Minimum Distortion Encoding Algorithm for Vector Quantization," *IEEE Trans. on Communications*, Vol. Com-33, 1985, pp. 1132-1133.

[10] C. K. Chan and L. M. Po, "A Complexity Reduction Technique for Image Vector Quantization," *IEEE Trans. on Image Processing*, Vol. 1, No. 3, 1992, pp. 312-321.

[11] L. Guan and M. Kamel, "Equal-Average Hyperplane Partitioning Method for Vector Quantization of Image Data," *Pattern Recognition Letters*, 1992, pp. 693-699.

[12] K. K. Paliwal and V. Ramasubrammainian, "Effect of Ordering the Codebook on the Efficiency of the Partial Distance Search Algorithm for Vector Quantization," *IEEE Trans. on Communications*, Vol. 27, 1989, pp. 538-540.

[13] L. Torres and J. Huguest, "An Improvement on Codebook Search for Vector Quantization," *IEEE Trans. on Communications*, Vol. 42, No. 2/3/4, 1994, pp. 208-210.

[14] X. Wu and L. Guan, "Acceleration of the LBG Algorithm," *IEEE Trans. on Communications.*, February-April 1994, pp. 1518-1523.

[15] Y. C. Lin and S. C. Tai, "Fast Vector Quantization of Image Coding Using Integral Projections," *Proc. National Computer Symposium*, 1995, pp. 729-236.

[16] 黃鴻儒，*Efficient Algorithms for Vector Quantization Codebook Design Based on Clustering*，中原大學資訊工程學系碩士學位論文.

[17] C. K. Chan and C. K. Ma, "A Fast Method of Designing Better Codebooks for Image Vector Quantization," *IEEE Trans. on Communications*, Vol. 42 No. 2//3/4, 1994, pp. 237-243.