行政院國家科學委員會專題研究計畫 成果報告

快速 k 個最近鄰居搜尋之效能檢測資料庫與演算法開發研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 99-2221-E-343-006-
執 行 期 間 ：99 年 08 月 01 日至 100 年 08 月 31 日
執 行 單 位 ：南華大學資訊工程學系

計 畫 主 持 人：廖怡欽

計畫參與人員：大專生-兼任助理人員：陳君儀
大專生-兼任助理人員：吳松展
大專生-兼任助理人員：王柏堯
大專生-兼任助理人員：李宗耀
大專生-兼任助理人員：楊仁傑
大專生-兼任助理人員：尉遲仲涵
大專生-兼任助理人員：陳裕益
大專生-兼任助理人員：劉慶龍
大專生-兼任助理人員：尤薪禾

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 100 年 10 月 12 日

行政院國家科學委員會補助專題研究計畫 **☑ 成 果 報 告**
□期中進度報告

快速 k 個最近鄰居搜尋之效能檢測資料庫與演算法開發

計畫類別：☑ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 99－2221－E－343－006－
執行期間：99 年 8 月 1 日至 100 年 8 月 31 日

計畫主持人：廖怡欽
計畫參與人員：陳君儀、吳松展、王柏堯、尤薪禾、李宗耀、楊仁傑、陳裕益、尉遲仲涵、劉慶龍

成果報告類型(依經費核定清單規定繳交)：☑精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
☑出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：南華大學 資訊工程學系

中　華　民　國　100　年　10　月　1　日

摘要

k 鄰居搜尋方法是用來在一群資料點中找尋一給定查詢點 *k* 個最近鄰居的方法，*k* 鄰居搜尋方法非常耗時，為了加快 *k* 鄰居的搜尋速度，許多快速 *k* 鄰居搜尋方法已被提出。現有快速 *k* 鄰居搜尋方法的效能很容易受到資料集的維度、資料量、以及資料分佈情形影響，在極端的情況下，快速 *k* 鄰居搜尋方法的效能可能比完全搜尋方法還差。為了解決這個問題，本計劃使用多種不同的真實資料集，測試 5 種快速 *k* 鄰居搜尋方法。實驗結果顯示，我們所設計的方法在不同維度、不同資料量、以及不同資料分佈情形的資料集下，大部份情況下，表現均比現有快速 *k* 鄰居搜尋方法更好。本計劃的結果非常適合用來為未知資料集挑選快速 *k* 鄰居搜尋方法。

關鍵字： 快速 *k* 鄰居搜尋方法，正交搜尋樹，效能評估.

Abstract

The problem of k-nearest neighbors (*kNN*) search is to find nearest *k* neighbors from a given data set for a query point. The finding process of *kNN* is very time comsuming. To speed up the finding process of nearest *k* neighbors, many fast *kNN* search algorithms were proposed. The performance of fast *kNN* search algorithms is highly influenced by the number of dimensions, number of data points, and data distribution of a data set. In the extreme case, the performance of a fast *kNN* search algorithm may be poorer than the full search algorithm. To overcome this problem, five fast algorithms were tested using multiple real data sets. Experimental results show that the proposed methods have better performance than others under various numbers of dimensions, various numbers of data points, and different types of data sets in most cases. The result of the project will be very useful in choosing a fast *kNN* search algorithm for an unknown data set.

Keywords: Fast *kNN* Search Algorithm, Orthogonal Search Tree, Performance Evaluation.

1.前言

搜尋最接近的 k 個鄰居(k nearest neighbors; kNN)問題是由一組已知資料集中找出與查詢點最接近的 k 個鄰居出來，這個技術已廣泛應用在許多科學與工程應用中，主要應用領域有：樣型識別[1]、物件識別[2]、資料分群[3, 4]、函式近似[5]、向量量化[6, 7]、與樣型分類[8, 9]等，隨著網路的普及以及資訊產業的快速發展，資料搜尋的應用變得愈來愈普遍，快速資料搜尋技術也變得愈來愈重要。

針對一個查詢點 Q，要由一個內含 n 個資料點的資料集 $S=\{X_1, X_2, ..., X_n\}$ 中找到 k 個與 Q 最接近的鄰居，最直覺的作法是計算查詢點 Q 與資料集 S 中所有點的距離，這個方法稱為完全搜尋法(Full search algorithm; FSA)，針對一個查詢點，需要計算 n 個距離。距離計算通常使用平方歐基里德距離(squared Euclidean distance)，針對一個維度為 d 的查詢點 $Q=[q_1, q_2, ..., q_d]^T$，以及資料集 S 中的一個資料點 $X_i=[x_{i1}, x_{i2}, ..., x_{id}]^T$，兩點的平方歐基里德距離計算公式定義如下：

$$D(\boldsymbol{X}_i, \boldsymbol{Q})=\sum_{j=1}^{d}(x_{ij}-q_j)^2 . \qquad (1)$$

也就是說，要找到 k 個最接近鄰居，必須計算資料集內所有資料點與查詢點的距離，每個距離計算必須做 d 個減法，d 個乘法，與 $d-1$ 個加法，針對一內含 n 個資料點的資料集，其計算複雜度為 $\theta(d\times n)$。使用完全搜尋法尋找 k 個最接近鄰居需要大量的計算，而且耗時。為了降低尋找 k 個最接近鄰居所需的計算時間，目前已有許多快速 kNN 搜尋演算法[10-22]可用，各種方法的表現與資料集內資料點的數量、維度大小、以及資料分佈情形有密切關係，如何針對自己的需要找到適用的方法，並不是件容易的事。有些方法，雖然在某些情況（例如低維度）有不錯的表現，但是在其它情況（例如高維度）表現並不好，若選錯方法，則加速效果不佳。在現有文獻中，每種快速搜尋方法僅針對部份資料集做測試，並無法窺探每種方法在各種情況下的表現。如何快速依資料集特性選擇適用的快速 kNN 搜尋方法是個很大的問題。

2.研究目的

本計劃主要目的在透過足夠的真實資料，測試各種快速 kNN 搜尋方法在不同情況下的表現，藉以提供快速 kNN 搜尋方法的快速選擇依據，此外，藉由所產生的實驗結果，也可回頭設計適合各種情況的快速 kNN 搜尋方法。

3.文獻探討

現有的快速 kNN 搜尋方法大致上可概分為以下兩類：

- 具樹狀結構之快速 kNN 搜尋方法

  這類演算法[10-16]事先將資料集中的資料點依資料點的特徵做分類，並使用樹狀結構儲存資料點的分類資訊；針對一給定的查詢點，要尋找資料集中距離查詢點最近的 k 個鄰居時，則使用分枝以及限制(branch and bound)搜尋方法，過濾不可能的分枝，減少距離計算量。在這類方法中，Fukunaga 與 Narendra [10]使用 Ball Tree 將資料點分類，

作法是使用分群方法[23-25]將資料點分成幾個不同的群組，針對每個群組的資料，再細分為多個群組，同樣的過程重覆進行直到每個群組內的資料點距離小於容許值或是只剩下一個資料點；針對每個群組，須記錄群組中心、群組半徑、以及群組內每個資料點到群組中心的距離；搜尋資料時，依群組中心與查詢點距離由近至遠依序尋找，然後再以目前找到之第 k 個最近距離 D_k 與群組的半徑做為檢查依據，過濾群組。一般而言，Ball Tree 方法對高維度資料有良好的表現[11]，但需要計算查詢點與群組中心的距離以及將群組依與查詢點的距離排序，因此容易受到所使用之分群方法、群組數量、以及資料點數量的影響。Friedman [12]等人使用空間分解方法將資料點分類，這個方法產生一個平衡的 k 維度二元樹，又稱為 *k-d* tree；*k-d* tree 的建構方式是先找出各個資料維度中使得資料集分離度最大的維度，然後將資料集依該維度將資料均勻的分成兩個群組，同樣的過程可以對每個群組再次進行分群動作，直到群組中只剩一個資料點或群組中資料點的距離小於容許值為止；查詢資料時，則依 *k-d* tree 的分類方式找到查詢點所屬的樹葉群組，由該群組找起，然後再找兄弟群組，搜尋一個群組前，先檢查該群組是否可被過濾掉。由於 *k-d* tree 只需計算查詢點到群組邊界的距離，因此額外負擔較小，當資料集的維度不大時 *k-d* tree 通常比 Ball Tree 有更好的表現，資料點數量對 *k-d* tree 的影響也會比較小，相反的，因分群方法較簡略，當維度大時，效果較差。其它方法還有 Kim 與 Park [13]等人使用有序分割方法建立一個排序好的多分枝搜尋樹，方便依與查詢點距離的遠近來搜尋群組；Mico [14]等人則事先建立一個距離表及二元樹，然後使用不同的群組過濾方法來過濾不可能的群組；McNames [15]使用主軸分析方法找出資料集的主軸，然後使用資料點映射到主軸的映射值將資料點分成 n_c 個子群組，每個子群組也使用相同的方式再分成 n_c 個子群組，直到每個群組的數量小於 n_c 為止。為了方便依與查詢點的距離遠近搜尋群組，所產生的子群組必須依映射值排序，方便依與查詢點的距離遠近搜尋群組，所建立出來的樹稱為主軸搜尋樹(PAT)；Chen[16]等人則使用金字塔結構建構低限制樹(LBT)，金字塔最下層是個別資料點，往上一層則儲存下層所有資料點所成群組的中心點(維度大小減半)以及群組半徑，最上層則是所有資料點的平均值且維度為 1；搜尋資料時則使用贏者更新(winner update)搜尋方法來加速 *kNN* 的搜尋速度。

- 非樹狀結構之快速 *kNN* 搜尋方法

這類搜尋方法通常使用排序方式以及資料集的部份特徵來過濾資料點。這類方法中，Cheng[17]等人提出一個 min-max 方法，用來降低運算數量，另外，他們也提出一個部份搜尋方法(PDS)，該方法在距離計算過程中，一旦發現此距離不可能是 *kNN* 時，可提早結束距離計算。Bei and Gray [18]稍後也提出相同於 PDS 的詳細演算法以及實驗結果，證明 PDS 確實可有效降低計算量，PDS 方法通常也會被使用在其它快速 *kNN* 搜尋方法中。Ra and Kim [19]將資料點事先依資料點的平均值作排序，在 *kNN* 搜尋過程中，使用二元搜尋法找到資料集中平均值與查詢點平均值最相似的 k 個資料點，然後依平均值差異由小而大，逐次搜尋資料點，搜尋過程中，則利用資料點的平均值與查詢點的平均值差異程度以及目前第 k 個最近鄰居的距離來決定是否提早結束搜尋過程。Tai [20]等人應用資料點的多個映射值（projection values）來過濾不可能的資料點。Lu [21]等人應用資料點各維度的平均值、變異數、以及 norm 來過濾不可能的資料點；Lai [22] 等人則應用資料點的多個映射值與三角不等式來過濾不可能的資料點。

4.研究方法

為了改善現有 *kNN* 快速搜尋方法，我們提出兩種快速 *kNN* 搜尋演算法，分別說明如下：

- Modified Principal Axis Tree(MPAT)方法

在使用樹狀結構的方法中，PAT 是最好的方法之一，針對大部份的資料集都能提供良好的效能。在 PAT 方法中，必須先建構搜尋樹，樹狀結構的每個非樹葉節點均必須記錄一個主軸(Principal Axis)向量以及每個子節點的映射值範圍，且其子節點必須依對主軸的映射值排序。*kNN* 的搜尋過程，由根節點開始搜尋，搜尋過程必須計算查詢點對節點主軸的映射值，然後使用該映射值與子節點映射值的距離作為過濾子節點的依據，針對過濾不掉的節點，則使用深先搜尋方法繼續檢查該子節點，直到遇到樹葉節點為止，針對過濾不掉的樹葉節點，則必須計算樹葉節點內所有資料點與查詢點的距離。

為了進一步減少距離的計算量，針對樹葉節點，我們記錄樹葉節點中所有資料點對其父節點主軸的映射值(在 PAT 中樹葉節點沒有主軸)，在搜尋過程中，將過濾不掉的樹葉節點依其與查詢點的關係分成 L、R、M 三類，其中 L 類樹葉節點中所有資料點的映射值均小於查詢點的映射值，R 類樹葉節點中所有資料點的映射值均大於查詢點的映射值，M 類樹葉節點中資料點的映射值則可能大於或小於查詢點的映射值。然後針對不同種類的樹葉節點提出不同的資料點過濾方法。由於 MPAT 利用過濾不可能的資料點加速搜尋速度，因此使用 MPAT，必須依據資料量調整 n_c 的值，使得樹葉節點具有較多的資料點，才能得到較佳的效能表現。本計劃所提的 MPAT 方法已發表在 SCI 國際期刊 Digital Signal Processing[28]。

- Orthogonal Search Tree(OST)方法

在 PAT 方法中，必須為每個節點計算並記錄主軸，由於每個節點的主軸均不相同，因此在搜尋過程中，必須為每個過濾不掉的非樹葉節點計算查詢點的映射值以及節點的邊界，當樹狀結構龐大，且過濾不掉的非樹葉節點數量多時，會增加許多計算量。

為了解決這個問題，我們提出一個不同的樹狀結構，樹狀結構的建構方法跟 MPAT 方法類似，但是主軸則不需針對個別節點計算，而是事先針對整個資料集計算出一組固定的正交基底(orthonormal basis)，正交基底內含 d 個互相垂直的向量(d 是資料集的維度)。建立搜尋樹時，針對每個節點，不需計算主軸，而是由正交基底中挑出一個最適合的向量作為其主軸，挑法是尋找一個向量使得節點中所有資料點對該向量之映射值具有最大的標準差。所挑選出來的主軸，必須不同於其所有祖先節點所挑選的主軸。由於 orthonormal basis 內的向量有限，因此查詢點最多只需計算 d 個映射值，且每個路徑上節點的主軸均互相垂直，因此不需計算各節點的邊界。

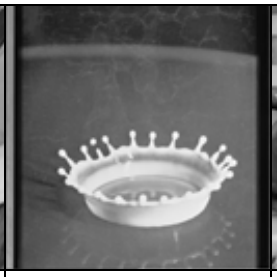
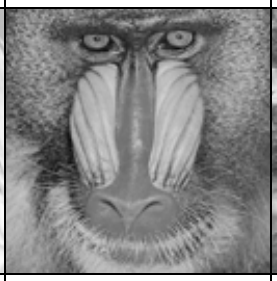
所提方法可有效減少映射值與邊界的計算量，雖然所使用的節點主軸並非最佳，但所建構出來之搜尋樹效果在大部份情況下跟 PAT 方法差不多，因此，搜尋速度明顯較 PAT 方法好，所提 OST 方法已發表在 SCI 國際期刊 Pattern Recognition 中[29]。

為了評估現有 *kNN* 方法在不同資料集下的效能表現，本計劃使用兩組資料集，這兩組資料集分別說明如下：

■ 真實影像資料集

為了測試資料維度與資料量對 *kNN* 搜尋方法的影響，本資料集使用 8 張 USC-SIPI 影像資料庫[30]所提供的影像作為實驗資料，每張影像大小為 512×512 像素。本實驗所使用的影像如表一所示：

表一：真實影像

			
Lena	Milk	Peppers	F16
			
Tiffany	Baboon	Bridge	Lake

本實驗共產生 9 個資料集，其中包括 5 個具有 16 個維度的資料集，其資料量分別是 1024, 2048, 4096, 8192, 與 16384，以及 4 個具有 1024 個資料點的資料集，維度分別是 4, 64, 256, 與 1024。每個資料集都是一個碼書(codebook)，碼書是使用 8 張影像中所有不重疊的影像區塊透過 LBG 演算法[23]產生而成。以一個 16 個維度且資料量為 1024 的資料集為例，要產生這個資料集，8 張影像中所有不重疊的 4×4 影像區塊 (131072 個影像區塊)均被用來做為資料點，所產生出來的碼書內含 1024 個碼字。為了測試搜尋速度，本實驗產生 10000 個查詢點，每個查詢點悉由 8 張影像中的影像區塊 (131072 個影像區塊)隨機選取而來。

■ 真實資料集

為了測試不同維度、不同資料量、以及不同種類之真實資料集對 *kNN* 搜尋方法的影響，我們由美國加州大學 Irvine 分校所建立的機器學習資料庫[31]中過濾挑選出沒有資料完整、純數值、且含有群組資訊的真實資料集，所挑選的資料集及其統計資訊如表二所列，其中資料分散程度的計算方式是取各群組"標準差/(最大值－最小值)"的平均值。針對每個資料集，分別產生 10000 個查詢點作為測試資料，每個查詢點則是隨機由資料集中挑選產生。

表二：真實資料集

資料集	維度	群數	資料點	資料分散程度
Iris	4	3	150	0.289
Abalone	8	3	4177	0.064
MAGIC Gamma Telescope	10	2	19020	0.077
Letter Recognition	16	26	20000	0.191
Ionosphere	34	2	351	0.293
Statlog (Landsat Satellite)	36	6	6435	0.163
Waveform Database Generator	40	3	5000	0.141
Optical Recognition	64	10	5620	0.377

為了方便表示，表二所列的資料集名稱分別簡化為 Iris、DL_Sensor、Abalone、Magic、Letter、Ionosphere、Statlog、Waveform、Optical、與 SECOM。

5.結果與討論

目前快速 *kNN* 搜尋方法中，表現較好的方法有 PAT[15]、LAI[22]、與 LBT[16]等方法，為了清楚了解本計劃設計的快速 *kNN* 搜尋方法與現有方法於不同維度、不同資料量、與不同種類之資料集下的表現，本計劃實作 FSA、PAT、LAI、LBT、MPAT、與 OST 等方法，並實際測試這些方法在不同資料集下的表現。

實驗中，針對 PAT、MPAT、OST 等方法，每個搜尋樹的子節點數目(n_c)均設為 20；針對 LBT 方法，我們使用作者網站[32]所提供的原始碼經過我們使用的編譯器重新編譯後的執行檔執行。所有程式均在 Microsoft Visual C++ 2008 Express 實作編譯完成，所有實驗均在一台配備有 Intel Core 2 Due P8600 2.4G Hz 與 4GB 記憶體以及 Windows Vista Home Premium 作業系統的個人電腦上完成。以下分別列出針對兩組不同種類資料集的實驗結果。

- 真實影像資料集

表三與表四分別列出 5 個快速 *kNN* 搜尋方法針對不同資料量之資料集與不同維度之資料集的前處理時間(單位：ms)。由表三與表四，我們可以看到 LBT 與 OST 方法在表三與表四分別需要較多的前處理時間。LBT 方法的前處理時間與資料集的資料量有關，資料量愈大，LBT 方法所需要的前處理時間愈多。OST 方法的前處理時間則與資料維度有關，維度愈大，OST 方法需要愈多前處理時間。

表三：針對 16 個維度不同資料量之資料集的前處理時間

資料量	PAT	LAI	LBT	MPAT	OST
1024	16	0	39	16	31
2048	16	0	164	16	78
4096	39	8	1,014	47	164
8192	117	47	6,755	133	375
16384	312	179	59,826	343	905

表四：針對 1024 個資料點不同維度之資料集的前處理時間

維度	PAT	LAI	LBT	MPAT	OST
4	0	0	24	0	8
16	16	0	39	16	31
64	16	0	39	32	718
256	78	0	55	109	15,140
1024	289	15	4,431	437	390,461

表五與表六分別列出5個快速*kNN*搜尋方法針對不同資料量之資料集與不同維度之資料集搜尋10000個查詢點3個最近鄰居的搜尋時間(單位：ms)。由表五的結果，我們可以發現，當資料量不大時，LAI方法效果最好，當資料量大時，OST方法可得到最佳的效果。由表六的結果，我們可以發現，當維度小時，LAI方法效果最好，當維度大時，LBT方法可得到最佳的效果。

表五：針對16個維度不同資料量之資料集的搜尋時間

資料量	FSA	PAT	LAI	LBT	MPAT	OST
1024	590	71	45	110	71	55
2048	1,175	98	73	172	98	78
4096	2,338	149	119	281	147	120
8192	4,738	288	196	476	290	182
16384	9,311	372	331	811	367	252

表六：針對1024個資料點不同維度之資料集的搜尋時間

維度	FSA	PAT	LAI	LBT	MPAT	OST
4	174	30	20	47	30	23
16	590	71	45	110	71	55
64	2,430	204	136	265	200	158
256	9,943	846	568	890	836	655
1024	41,355	3,987	2,757	2,442	3,931	3,076

由表五與表六發現，當資料集來自真實影像時，我們可以依據資料集的資料量與維度挑選適用的快速*kNN*搜尋方法。挑選建議如表七所示：

表七：針對真實影像資料集的快速*kNN*搜尋方法挑選建議

資料量 / 維度	小於8192	大於8192
小於1024	選擇LAI方法	選擇OST方法
大於1024	選擇LBT方法	選擇OST方法

■ 真實資料集

表八與表九分別列出快速*kNN*搜尋方法針對不同真實資料集的前處理時間與搜尋10000個查詢點3個最近鄰居的資料搜尋時間(單位：ms)。

表八：針對不同真實資料集各種快速*kNN*搜尋方法的前處理時間

資料集	PAT	LAI	LBT	MPAT	OST
Iris	0	0	0	0	0
Abalone	16	16	1,906	31	47
Magic	343	234	1,013,609	343	546
Letter	421	265	159,140	468	1,154
Ionosphere	0	0	15	0	63
Statlog	94	47	3,468	124	1,482
Waveform	78	16	2,109	93	1,950
Optical	140	31	2,110	156	3,682

表九：針對不同真實資料集各種 *kNN* 搜尋方法的搜尋時間

資料集	FSA	PAT	LAI	LBT	MPAT	OST
Iris	36	14	16	15	13	11
Abalone	1,223	37	34	110	33	27
Magic	6,900	220	939	907	217	131
Letter	11,323	772	1,733	1,250	780	541
Ionosphere	435	173	162	515	168	153
Statlog	8,274	516	518	1,500	499	451
Waveform	7,281	2,775	4,477	17,843	2,796	2,746
Optical	13,163	2,427	5,025	11,156	2,360	2,256

由表八可以發現，LBT 與 OST 方法分別在不同情況下花費較多的前處理時間，其中 LBT 針對資料量大的資料集花費較多前處理時間，例如： Magic 與 Letter 等資料集；而 OST 則針對維度較大的資料集花費較多前處理時間，例如：Waveform。由表九可見，OST 對所有資料集均有最佳的效果，MPAT 則排第二。

由表八與表九，我們可以總結針對大部份的真實資料集而言，OST 是最好的選擇，如果前處理時間相對重要且資料維度大，則 MPAT 是最佳選擇。當選擇 MPAT 作為快速 *kNN* 搜尋方法時，必須依據資料量調整子節點的數目，使得建構出來的搜尋樹之樹葉節點擁有較多的資料點，以提高資料點的過濾效果。

表十列出真實資料集的資料分散程度，FSA 與 OST 的執行時間，以及 OST 可降低的執行時間比例。由表十可見，快速 *kNN* 方法的效能改善與資料分散程度有明顯的相關性。

表十：真實資料集

資料集	資料分散程度	FSA	OST	OST 可降低的執行時間比例
Iris	0.289	36	11	69%
Abalone	0.064	1,223	27	98%
Magic	0.077	6,900	131	98%
Letter	0.191	11,323	541	95%
Ionosphere	0.293	435	153	65%
Statlog	0.163	8,274	451	95%
Waveform	0.141	7,281	2,746	62%
Optical	0.377	13,163	13,163	83%

kNN 搜尋方法是一個很重要的技術，為了降低 *kNN* 搜尋方法的計算量，本計劃提出兩個有效的快速 *kNN* 搜尋方法。快速 *kNN* 搜尋方法的效能很容易受到維度、資料量、與資料集種類的影響。為了幫助了解各種快速 *kNN* 搜尋方法的效能，本計劃分別使用一組真實影像資料集以及一組不同種類真實資料集測試快速 *kNN* 搜尋方法在不同資料量、不同維度、與不同資料集下的效能表現。實驗結果顯示，針對真實影像，LAI、LBT、與 OST 等方法分別在不同情況下有較好的表現。針對實驗中的所有真實資料集，OST 則具有最佳的表現。

參考文獻

[1] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2nd edition, 2003.

[2] H. Murase and S. K. Nayar, "Visual learning and recognition of 3D objects from appearance," International Journal of Computer Vision, vol. 14, no. 1, pp. 5-24, January 1995.

[3] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann, "Optimal cluster preserving embedding of nonmetric proximity data," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 12, pp. 1540-1551, December 2003.

[4] Y. C. Liaw, "Improvement of the fast exact pairwise-nearest-neighbor algorithm," Pattern Recognition, vol. 42, no. 5, pp. 867-870, May 2009.

[5] C. Y. Chen, C. C. Chang, and R. C. T. Lee, "A near pattern-matching scheme based on principal component analysis," Pattern Recognition Letters, vol. 16, pp. 339-345, April 1995.

[6] Y. C. Liaw, J. Z. C. Lai, and Winston Lo, "Image restoration of compressed image using classified vector quantization," Pattern Recognition, vol. 35, no. 2, pp. 329-340, February 2002.

[7] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston MA., 1991.

[8] T. M. Cover, P. E. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967.

[9] Nicolás García-Pedrajas and Domingo Ortiz-Boyer, "Boosting k-nearest neighbor classifier by means of input space projection," Expert Systems with Applications, vol. 36, issue 7, pp. 10570-10582, March, 2009.

[10] Keinosuke Fukunaga and Patrenahalli M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," vol. C-24, issue 7, IEEE Transactions on Computers, pp. 750-753, July 1975.

[11] Ting Liu, Andrew W. Moore, and Alexander Gray, "New algorithms for efficient high-dimensional nonparametric classification," Journal of Machine Learning Research, vol. 7, pp. 1135-1158, 2006.

[12] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," ACM Transaction on Mathematical Software, vol. 3, no. 3, pp.209–226, 1977.

[13] B. S. kim and S. B. Park, "A fast k nearest neighboring finding algorithm based on the ordered partition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 761-766, November 1986.

[14] L. Mico, J. Oncina, and R. C. Carrasco, "A fast branch and bound nearest neighbor classifier in metric spaces," Pattern Recognition Letters, vol. 17, no. 7, pp. 731-739, June 1996.

[15] J. McNames, "A fast nearest-neighbor algorithm based on a principal axis search tree," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 9, pp. 964-976, September 2001.

[16] Yong-Sheng Chen, Yi-Ping Hung, Ting-Fang Ten, Chiou-Shann Fuh, "Fast and versatile algorithm for nearest neighbor search based on a lower bound tree," Pattern Recognition, vol. 40, no. 2, pp. 360-375, February 2007.

[17] D.-Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, "Fast Search Algorithms for Vector Quantization and Pattern Matching," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 9.11.1-9.11.4, March 1984.

[18] C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," IEEE Transactions on Communications, vol. 33, no. 10, pp. 1132-1133, October 1985.

[19] S. W. Ra and J. K. Kim, "Fast mean-distance-ordered partial codebook search algorithm for image vector quantization," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 40, no. 9, pp. 576–579, September 1993.

[20] S. C. Tai, C. C. Lai, and Y. C. Lin, “Two fast nearest neighbor searching algorithms for image vector quantization,” *IEEE Trans. on Communications*, vol. 44, no. 12, pp.1623-1628, December 1996.
[21] Zhe-Ming Lu, Shu-Chuan Chu, and Kuang-Chih Huang, "Equal-average Equal-variance Equal-norm nearest neighbor codeword search algorithm based on ordered Hadamard transform," *International J. of Innovative Computing, Information and Control*, vol. 1, no. 1, pp. 35-41, March 2005.
[22] J. Z. C. Lai, Y. C. Liaw, and J. Liu, “Fast k-nearest-neighbor search based on projection and triangular Inequality,” Pattern Recognition, vol. 40, no.1, pp. 351-359, January 2007.
[23] Y. Linde, A. Buzo, and R. M. Gray, “An algorithm for vector quantizer design,” IEEE Transactions on Communication, vol. 28, no. 1, pp. 84-95, January 1980.
[24] Jim Z.C. Lai, Tsung-Jen Huang*, and Yi-Ching Liaw, "A fast k-means clustering algorithm using cluster center displacement," Pattern Recognition, Vol. 42, No. 11, November 2009, pp.2551-2556.
[25] Yi-Ching Liaw, "Improvement of the Fast Exact Pairwise-Nearest-Neighbor Algorithm," Pattern Recognition, Vol. 42, No. 5, May 2009, pp.867-870.
[26] http://archive.ics.uci.edu/ml.
[27] Z. Volkovicha,b, Z. Barzilyb, and L. Morozenskyb, "A statistical model of cluster stability," Pattern Recognition, Vol. 41, Issue 7, July 2008, pp. 2174–2188.
[28] Yi-Ching Liaw*, Chien-Min Wu, and Maw-Lin Leou, "Fast K-Nearest Neighbors Search Using Modified Principal Axis Search Tree," Digital Signal Processing, Vol. 20, Issue 5, pp. 1494-1501, September 2010. (SCI)
[29] Yi-Ching Liaw*, Maw-Lin Leou, and Chien-Min Wu, "Fast exact k nearest neighbors search using an orthogonal search tree," Pattern Recognition, Vol. 43, No. 6, June 2010, pp.2351-2358. (SCI;EI)
[30] http://sipi.usc.edu/database.
[31] http://archive.ics.uci.edu/ml/index.html.

計畫成果自評

■ **原計畫相符程度與達成預期目標情況**

本計劃預計完成一快速 *kNN* 搜尋方法的效能檢測資料庫並設計受資料量數目，資料維度，以及資料分佈情況影響較小的快速 *kNN* 搜尋方法。

計劃執行完畢，已完成開發兩個受資料量數目、資料維度、以及資料分佈情況影響較小的快速 *kNN* 搜尋方法以及整理出兩組可作為效能檢測的真實資料集，實驗結果顯示，本計劃所提方法在不同維度、不同資料量、以及不同資料分佈情形的資料集下，大部份情況下，表現均比現有快速 *k* 鄰居搜尋方法更好。本計劃的結果也非常適合用來為未知資料集挑選快速 *k* 鄰居搜尋方法。

■ **研究成果之學術或應用價值**

學術價值：本計劃已開發兩個快速 *kNN* 搜尋方法，成果發表在 SCI 國際期刊 Digital Signal Processing[28]與 Pattern Recognition [29]，實驗分析結果發表在國際研討會[32]中。

應用價值：本計劃所開發出來的快速 *kNN* 搜尋方法是目前已知受資料量、維度、與資料種類影響最小的方法，此方法可應用在樣型識別[1]、物件識別[2]、資料分群[3, 4]、函式近似[5]、向量量化[6, 7]、與樣型分類[8, 9]等各種領域。

■ **程式開發人才培育**

本計劃執行人員，包括多位大學部學生，協助閱讀整理文獻、搜集實驗資料、轉換實驗資料格式、分析實驗資料特性、以及實作快速搜尋方法等，參與本計劃，對學生英文文件閱讀能力、報告能力、程式實作能力的提昇、做事態度的養成、與實驗方法的熟悉均有很大的幫助。

可供推廣之研發成果資料表

☑ 可申請專利　☑ 可技術移轉　日期：100 年 9 月 30 日

國科會補助計畫	計畫名稱：快速 *k* 個最近鄰居搜尋之效能檢測資料庫與演算法開發 計畫主持人：廖怡欽 計畫編號：NSC 99-2221-E-343-006-　　學門領域：資訊學門二
技術/創作名稱	快速 *k* 個最近鄰居搜尋方法
發明人/創作人	廖怡欽
技術說明	中文：*k* 鄰居搜尋方法是用來在一群資料點中找尋一給定查詢點 *k* 個最近鄰居的方法，*k* 鄰居搜尋方法非常耗時，為了加快 *k* 鄰居的搜尋速度，許多快速 *k* 鄰居搜尋方法已被提出。現有快速 *k* 鄰居搜尋方法的效能很容易受到資料集的維度、資料量、以及資料分佈情形影響，在極端的情況下，快速 *k* 鄰居搜尋方法的效能可能比完全搜尋方法還差。為了解決這個問題，本計劃使用多種不同的真實資料集，測試 5 種快速 *k* 鄰居搜尋方法。實驗結果顯示，本計劃所提方法在不同維度、不同資料量、以及不同資料分佈情形的資料集下，大部份情況下，表現均比現有快速 *k* 鄰居搜尋方法更好。本計劃的結果非常適合用來為未知資料集挑選快速 *k* 鄰居搜尋方法。
	英文：The problem of k-nearest neighbors (*kNN*) search is to find nearest k neighbors from a given data set for a query point. The finding process of *kNN* is very time comsuming. To speed up the finding process of nearest *k* neighbors, many fast *kNN* search algorithms were proposed. The performance of fast *kNN* search algorithms is highly influenced by the number of dimensions, number of data points, and data distribution of a data set. In the extreme case, the performance of a fast *kNN* search algorithm may be poorer than the full search algorithm. To overcome this problem, five fast algorithms were tested using multiple real data sets. Experimental results show that the proposed methods have better performance than others under various numbers of dimensions, various numbers of data points, and different types of data sets in most cases. The result of the project will be very useful in choosing a fast *kNN* search algorithm for an unknown data set.
可利用之產業及可開發之產品	可利用之產業：機器學習、樣型識別、資料探勘與知識挖掘、行為識別等相關產業
技術特點	受資料量數目、資料維度、以及資料分佈情況影響較小 是目前最快速的 *kNN* 搜尋方法
推廣及運用的價值	可提昇機器學習、樣型識別、資料探勘與知識挖掘、行為識別速度。

※ 1. 每項研發成果請填寫一式二份，一份隨成果報告送繳本會，一份送　貴單位研發成果推廣單位（如技術移轉中心）。

※ **2. 本項研發成果若尚未申請專利，請勿揭露可申請專利之主要內容。**

※ 3. 本表若不敷使用，請自行影印使用。

出席國際學術會議心得報告

計畫編號	NSC 99-2221-E-343-006
計畫名稱	快速 k 個最近鄰居搜尋之效能檢測資料庫與演算法開發
出國人員姓名 服務機關及職稱	廖怡欽 南華大學副教授
會議時間地點	民國 100 年 7 月 18 日至民國 100 年 7 月 21 日
會議名稱	The 2011 International Conference on Image Processing, Computer Vision, and Pattern Recognition
發表論文題目	Evaluation of Fast K-Nearest Neighbors Search Methods using Real Data Sets

一、參加會議經過

此次出國主要目的是到美國拉斯維加斯參加 IPCV2011 會議發表論文(附件一),該會議屬於 WORLDCOMP 2011 的一個分會,WORLDCOMP 2011 共包括 12 個分會,是一個非常大的會議,共有 88 個國家超過一千人參加。會議時間 7 月 18 日至 7 月 21 日共四天,開會場地是 Monte Carlo 飯店。

此行搭乘華航 CI0006 班機於 7/17 下午 4 點 40 分起飛到美國洛杉機轉機,然後再搭乘美國航空到拉斯維加斯,到達拉斯維加斯 Excalibor 飯店時已是美國時間下午 6:00。我住的飯店離會議地點不遠,走路大約 20 分鐘,飯店 Check In 後,先到會議地點看看,順便參觀拉斯維加斯。

7 月 18 日早上 8:30 在 Monte Carlo 飯店的戲院開幕,接下來是一系列的演講及論文發表,7 月 18 日晚上 9:10 大會晚宴,晚宴期間認識了一些與會學者,有從台灣過去目前在美國北德州大學(UNT)Dallas 分校教書的楊逢仁教授,阿肯色中央大學(UCA)教書的陳啟天教授,有新加坡國立大學的黃建民教授,馬來西亞多媒體大學的 Sim Kok Swee 教授,馬來西亞 Universiti Kebangsaan 的 Nasharuddin Zainal 教授,...等。

7/19~7/20 挑選了幾篇有興趣的論文聽聽順便跟一些與會學者談談,發現美國許多學校也有招生壓力,而且也必須到高中宣導,也發現美國電腦相關科系招生不好,多數美國人不念電腦相關科系,東南亞狀況則較穩定。7/20 搭乘下午 6:35 的飛機到洛杉機,然後在洛杉機停留幾天,順道到本校的姊妹校西來大學與西來寺,洽談交換學生事宜,7/26 凌晨 1:35 分搭乘華航 CI0007 回國,7/27 早上 6:00 回到台灣。

二、與會心得

參加過許多次會議，這次會議跟以前參加的會議比較不同，參加會議的人數較多，共有 88 個國家 1 千多人參加，會議場次很多，只能精選幾場參加。這次會議認識一些人，也發現不同國家電腦相關科系的發展狀況有很大的差異，先進國家一般大學電腦相關科系招生不足，經營困難，東南亞大學教育供給不足，沒有招生的問題，是值得開發的地方。拉斯維加斯很熱，除了深夜及早上外，大部份時間均溫約 40 度，可是到處可以看到許多觀光客在路上走路，我發現許多觀光客都是來參加會議的學者或是其家屬，而且除了我參加的會議之外，還有許多其它會議，藉由舉辦會議促進觀光發展是相當聰明的作法。拉斯維加斯的飯店與商場集中，方便觀光客參觀及購物，部份距離較遠的飯店，也有提供免費的輕軌電車，為了吸引觀光客，部份飯店也會提供免費的表演，例如 Virage 提供的火山秀，以及 Bellagio 提供的大型水舞。台灣目前並沒有類似的場地，如果台灣可以建置一個類似功能的觀光場地，並鼓勵大型會議到該場地舉辦，相信對提昇台灣的觀光產業，就業機會，以及國際形象都有很大的幫助。

Administered by: UCMSS
13223-1 Black Mountain Road,
Suite 135
San Diego, CA 92129-2658

WORLDCOMP'11
July 18-21, 2011
Monte Carlo Resort
Las Vegas Nevada, USA

Official Contact: Prof. H. R. Arabnia
Phone: (706) 542-3480
Fax: (706) 542-2966
Email: hra@cs.uga.edu
www.world-academy-of-science.org

April 14, 2011

Prof. Yi-Ching Liaw
No.55, Sec.1, Nanhua Rd., Zhongkeng
Dalin Township,
Chiayi County 62248,
Taiwan

Dear Prof. Yi-Ching Liaw,

This notification letter is to inform you that the paper entitled "***EVALUATION OF FAST K-NEAREST NEIGHBORS SEARCH METHODS USING REAL DATA SETS***" which was submitted to The 2011 International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV'11) has been accepted for publication.

You have been invited by the members of the Organizing Committee of The 2011 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'11) to attend the annual summer conference series to be held in Las Vegas, Nevada, USA (July 18-21, 2011). Universal Conference Management Systems & Support (UCMSS), who is managing the operation of these conferences, wishes to congratulate you on behalf of the Conference Committee and extend this invitation to you for presentation of the above paper at this event. Enclosed, please find a letter addressed to the United States Consulate General for US VISA purposes.

We hope that you will take advantage of this exceptional opportunity to join us. We are confident that you will enjoy the scientific program that is being offered at this year's event.

Congratulations once again, and thank you for your contribution to the conference. We look forward to welcoming you at the conference in Las Vegas.

Sincerely,

Kaveh D. Arbtan
UCMSS
San Diego, California
U.S.A.

Administered by: UCMSS
13223-1 Black Mountain Road,
Suite 135
San Diego, CA 92129-2658

WORLDCOMP'11
July 18-21, 2011
Monte Carlo Resort
Las Vegas Nevada, USA

Official Contact: Prof. H. R. Arabnia
Phone: (706) 542-3480
Fax: (706) 542-2966
Email: hra@cs.uga.edu
www.world-academy-of-science.org

April 14, 2011

TO: US Embassy (Consulate General) and US Immigration and Naturalization Service Officer
RE: Request for U.S. Visa
Requested by WORLDCOMP'11 Organizing Committee
Requested for Prof. Yi-Ching Liaw
Duration of the Conference; July 18-21, 2011

Dear Honorable Consul,

Members of the Organizing Committee of The 2011 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'11) have invited Prof. Yi-Ching Liaw to present a research paper entitled "***EVALUATION OF FAST K-NEAREST NEIGHBORS SEARCH METHODS USING REAL DATA SETS***" at The 2011 International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV'11), to be held in Las Vegas, Nevada, USA, (July 18-21, 2011).

Universal Conference Management Systems & Support (UCMSS), who is managing the operation of these conferences, respectfully requests that Prof. Liaw be allowed entry into the United States on an appropriate Visa status for the term of the visit as stated above.

We are grateful for all due courtesy extended to this individual by your personnel. Your kind consideration to this request will be greatly appreciated.

Sincerely,

Kaveh D. Arbtan
UCMSS
San Diego, California
U.S.A.

EVALUATION OF FAST K-NEAREST NEIGHBORS SEARCH METHODS USING REAL DATA SETS

Yi-Ching Liaw
Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan

Abstract - *The problem of k-nearest neighbors (kNN) search is to find nearest k neighbors from a given data set for a query point. To speed up the finding process of nearest k neighbors, many fast kNN search algorithms were proposed. The performance of fast kNN search algorithms is highly influenced by the number of dimensions, number of data points, and data distribution of a data set. In the extreme case, the performance of a fast kNN search algorithm may be poorer than the full search algorithm. To help understand the performance of fast kNN search algorithms on data sets of different types, five fast algorithms were tested in this paper using multiple real data sets. The experimental results of the paper will be very useful in choosing a fast kNN search algorithm for an unknown data set.*

Keywords: Fast *kNN* search algorithm, performance evaluation.

1 Introduction

The problem of *k*-nearest neighbors (*kNN*) search is to find nearest *k* neighbors for a query point from a given data set. This problem occurs in many types of applications [1-3].

The intuitive method of finding nearest *k* neighbors for a query point $\boldsymbol{Q}$ from a data set $S=\{\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_n}\}$ of *n* data points is to compute *n* distances between the query point *Q* and all data points in the data set *S*. This method is called the full search algorithm (FSA). Generally, the squared Euclidean distance is applied to measure the distance between two points, for a query point $\boldsymbol{Q}=[q_1, q_2, ..., q_d]^T$ with dimension *d* and a data point $X_i=[x_{i1}, x_{i2}, ..., x_{id}]^T$ from data set *S*, the distance between these two points is defined as below :

$$d(X_i, Q) = \sum_{j=1}^{d} (x_{ij} - q_j)^2 \qquad (1)$$

The full search algorithm is easy to be implemented, but always takes a lot of computation time. To reduce the computation time of the *kNN* finding process, many fast *kNN* search algorithms were proposed [4-8]. Among available fast *kNN* search methods, algorithms proposed by McNames [4], Lai et al. [5], Chen et al. [6], and Liaw et al. [7][8] usually have better performance than others. In this paper, we refer to algorithms presented in [4], [5], [6], [7], and [8] as PAT, LAI, LBT, MPAT, and OST, respectively.

The performance of fast *kNN* search algorithms is highly influenced by the number of data dimensions, the number of data points, and the type of data distribution. For data sets of different types, different search algorithms should be chosen to achieve the best performance. For the case of a wrong fast algorithm is chosen for a data set, the performance of finding nearest *k* neighbors may be poor. In the extreme case, the performance of a fast *kNN* search algorithm may poorer than the FSA.

To help finding the best *kNN* search algorithm for an unknown data set, this paper evaluates the performance of several new and high performance fast *kNN* search algorithms [4-8] using real data sets of various dimensions, various numbers of data points, and various types of applications.

The rest of this paper is organized as follows. In section II, five fast *kNN* search algorithms [4-8] are briefly reviewed. The real data sets selected in this paper are described in section III. Experimental results and conclusions are given in section IV and section V, respectively.

2 Fast *kNN* Search Algorithms

2.1 The PAT Algorithm

The PAT algorithm [4] consists of two phases : the principal axis search tree construction phase and the nearest neighbors finding phase.

The principal axis search tree construction phase builts a principal axis search tree for a data set. In the beginning, there is only one node (the root node) is created for the tree and all data points in the data set are assigned to the root node. Next, data points in the root node are partitioned into n_c child nodes. The partition process of a node is started by evaluating the principal axis of data points in the node and followed by partitioning data points in the node into n_c child nodes according to projection values of data points onto the principal axis. The partition process is repeatedly applied to each child node until the number of data points in a node is less than n_c. After the principal axis search tree is built, every

internal node in the tree maintains n_c child nodes and a principal axis while a leaf node records a list of data points.

To find nearest k neighbors from a principal axis search tree for a query point, projection values of the query point onto principal axes of tree nodes are used as features to filter out impossible tree nodes by traversing the tree in depth-first search order and using a node elimination criterion.

During the tree nodes filtering process, boundary points and projection values of the query point onto principal axes of nodes must be evaluated. For a large data set or a search tree with high depth, the computation time for evaluating such information may be large.

2.2 The LAI Algorithm

This algorithm [5] uses projection values of a point onto three vectors (mean, horizontal, and vertical vectors) as features and four inequalities to reject impossible data points in the nearest neighbors finding process of a query point. Before this algorithm can be applied, projection values of data points should be evaluated and all data points are arranged in asscending order according to mean values of data points.

To find nearest neighbors from sorted data points for a query point, projection values of the query point onto three vectors are first evaluated and the searching order of data points is from the data point with the closest mean value to the query point to the one with the farthest mean value to the query point. During the searching process, four inequalities are used in sequence to filter out unlikely data points based on their projection values.

2.3 The LBT Algorithm

The LBT algorithm [6] uses a multilevel lower-bound tree and a winner update search method to speed up the nearest neighbors finding process.

To build the multilevel lower-bound tree for a data set, data points in the data set are first transformed using an orthonormal basis of d orthogonal vectors, where d is the number of dimensions for the data set. Then, data points in the data set are partitioned into clusters according to their projection values onto the first orthogonal vector and a given radius using an agglomerative clustering technique. Each cluster is then further partitioned into clusters according to their projection values onto the second orthogonal vector. This process will be repeated $L=\lceil log(d) \rceil$ times and generates a tree of $L+1$ levels. After the above tree is built, a multilevel lower-bound tree can be generated according to radiuses and centroids of nodes.

To find nearest neighbors for a query point using the LBT method, the projection values of the query point onto orthogonal vectors must be evaluated first and the finding process of the query point is performed by traversing the multilevel lower-bound tree in breadth-first search order and using a node elimination criterion to filter out impossible nodes.

2.4 The MPAT Algorithm

The *kNN* finding process for a query point using the PAT method is to reject impossible nodes from a principal axis search tree using a node elimination criterion. Once a leaf node cannot be rejected, distances between the query point and all data points in the leaf node must be computed.

To speed up the leaf node search process, the MPAT algorithm [7] stores projection values of all data points in leaf nodes, categorized leaf nodes into three types, and proposed a new leaf node search algorithm. For a leaf node in the tree cannot be rejected by the node elimination criterion, data points in the leaf node are further checked using their pre-stored projection values, type information of the leaf node, and a data point rejection inequality to reject more impossible data points.

2.5 The OST Algorithm

To find nearest neighbors from a principal axis search tree using the PAT and MPAT method, many projection values and boundary points must be evaluated for a query point. Numbers of projection values and boundary points to be evaluated for a query point are in proportion to the number of unrejected internal nodes. For a large tree, to evaluate projection values and boundary points for a query point may take much computation time.

To overcome this problem, the OST method creates an orthogonal search tree for a data set using an orthonormal basis evaluated from the data set and a similar search tree construction process as that presented in the PAT method. In the OST method, an orthogonal vector is chosen from the orthonormal basis for a node to partition data points of a node into child nodes. The orthogonal vector selected by a node must perpendicular to those orthogonal vectors chosen by ancestors of the node. That is, the level of an orthogonal search tree must less or equal to the number of dimemsions.

To find nearest neighbors for a query point using the OST method, projection values of the query point onto orthogonal vectors and a node elimination inequality are used to reject impossible nodes. For a node, which cannot be rejected by the node elimination inequality, a point elimination inequality is also applied to reject impossible data points.

Comparing to the PAT and MPAT algorithms, the OST algorithm requires no boundary points and only little computation time on evaluating projection values in the nearest neighbors finding process.

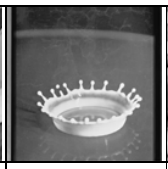
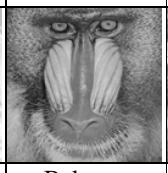
3 Real Data Sets

To evaluate the performance of fast *kNN* search algorithms, two sorts of real data sets were selected and described in the following sections.

3.1 Data Sets From Real Images

In this paper, 8 real images with size of 512×512 pixels from USC-SIPI Image Database [9] (see Table I) were selected to generate 9 data sets and 10000 query points.

Table I : Images from USC-SIPI Image Database

			
Lena	Milk	Peppers	F16
			
Tiffany	Baboon	Bridge	Lake

Each data set is a codebook generated using the LBG [10] algorithm and all non-overlapping 4×4 image blocks obtained from 8 images. Here, 9 data sets were generated. Five of them are with 16 dimensions and various numbers of data points (1024, 2048, 4096, 8192, and 16384) and the others are with various dimensions (4, 64, 256, 1024) and 1024 data points.

Each query point is a non-overlapping 4×4 image block randomly selected from 8 images.

3.2 Data Sets From UC Irvine Machine Learning Repository

UC Irvine machine learning repository [11] provides many useful real data sets. In this paper, 10 real data sets with numeric data and various numbers of dimensions as listed in Table II were selected in the paper. For each data set, 10000 query points were generated by randomly selecting data points from the data set.

In convinience, data sets listed in Table II are referred to as "Iris", "DL_Sensor", "Abalone", "Magic", "Letter", "Ionosphere", "Statlog", "Waveform", "Optical", and "SECOM", respectively.

Table II : Data Sets from UC Irvine machine learning repository

Name	Dimensions	Data Points
Iris	4	150
Dodgers Loop Sensor	6	50400
Abalone	8	4177
MAGIC Gamma Telescope	10	19020
Letter Recognition	16	20000
Ionosphere	34	351
Statlog (Landsat Satellite)	36	6435
Waveform Database Generator	40	5000
Optical Recognition	64	5620
SECOM	590	1567

4 Experimental Results

To evaluate the performance of fast *kNN* search algorithms, data sets and query points obtained in section 3 were used for testing the performance of six *kNN* search algorithms (FSA, PAT, LAI, LBT, MPAT, and OST).

In the experiment, all search trees were constructed with n_c=20 for the PAT, MPAT, and OST algorithms. For the LBT algorithm, source code downloaded from the author's web site [12] is used. All programs were implemented using Microsoft Visual C++ 2008 Express under Windows Vista Home Premium. All programs were executed on a personal computer with Intel Core 2 Due P8600 2.4G Hz and memory of 4 GB.

4.1 Experiment Using Data Sets From Real Images

Tables III and IV list the preprocessing time (in milliseconds) for five fast *kNN* search algorithms. From Tables III and IV, we can see that the LBT and OST algorithms take the most preprocessing time in different cases. When a data set with a large number of data points, the LBT algorithm will take a lot of preprocessing time while for a data set with a large number of dimensions, the OST algorithm has the most preprocessing time.

Table III : Preprocessing time for five *kNN* search algorithms were applied on data sets with 16 dimensions and various numbers of data points.

Number of Data Points	PAT	LAI	LBT	MPAT	OST
1024	16	0	39	16	31
2048	16	0	164	16	78
4096	39	8	1,014	47	164
8192	117	47	6,755	133	375
16384	312	179	59,826	343	905

Table IV : Preprocessing time for five *kNN* search algorithms were applied on data sets with 1024 data points and various numbers of dimensions.

Number of Dimensions	PAT	LAI	LBT	MPAT	OST
4	0	0	24	0	8
16	16	0	39	16	31
64	16	0	39	32	718
256	78	0	55	109	15,140
1024	289	15	4,431	437	390,461

Table V gives the total search time (in milliseconds) to find nearest 3 neighbors for 10000 query points from data sets of various numbers of data points using six *kNN* search algorithms. From Table V, we can find that the LAI algorithm is the best algorithm when the number of data points is small. For a large number of data points, the OST method becomes the best one.

Table V : Total search time for finding nearest 3 neighbors for 10000 query points from data sets of various numbers of data points.

Number of Data Points	FSA	PAT	LAI	LBT	MPAT	OST
1024	590	71	45	110	71	55
2048	1,175	98	73	172	98	78
4096	2,338	149	119	281	147	120
8192	4,738	288	196	476	290	182
16384	9,311	372	331	811	367	252

Table VI shows the total search time (in milliseconds) to find nearest 3 neighbors for 10000 query points from data sets with 1024 data points and various numbers of dimensions. From Table VI, we can find that the LAI algorithm is the best algorithm when the number of dimensions is small. For a large number of dimensions, the LBT method becomes the best one.

Table VI : Total search time for finding nearest 3 neighbors for 10000 query points from data sets of various numbers of dimensions.

Number of Dimensions	FSA	PAT	LAI	LBT	MPAT	OST
4	174	30	20	47	30	23
16	590	71	45	110	71	55
64	2,430	204	136	265	200	158
256	9,943	846	568	890	836	655
1024	41,355	3,987	2,757	2,442	3,931	3,076

From this experiment, we can say that the LAI and OST algorithms are good choices for data sets from real images and with numbers of dimensions less than 256. For the number of dimensions greater than 256, the LBT algorithm becomes the best choice.

4.2 Experiment Using Data Sets From UC Irvine Machine Learning Repository

Tables VII and VIII enumerate the preprocessing time (in milliseconds) and total search time (in milliseconds) for *kNN* search algorithms were applied on data sets from UC Irvine machine learning repository. Total search time for a data set and a *kNN* search algorithm in Table VIII is the total computation time to find nearest 3 neighbors for 10000 query points.

Table VII : Preprocessing time for five fast *kNN* search algorithms were applied on data sets from UC Irvine machine learning repository.

Data Set	PAT	LAI	LBT	MPAT	OST
Iris	0	0	0	0	0
DL_Sensor	1,872	1,669	1,001,890	1,965	2,106
Abalone	16	16	1,906	31	47
Magic	343	234	1,013,609	343	546
Letter	421	265	159,140	468	1,154
Ionosphere	0	0	15	0	63
Statlog	94	47	3,468	124	1,482
Waveform	78	16	2,109	93	1,950
Optical	140	31	2,110	156	3,682
SECOM	265	16	188	359	106,314

Table VIII : Total search time for finding nearest 3 neighbors for 10000 query points from different data sets.

Data Set	FSA	PAT	LAI	LBT	MPAT	OST
Iris	36	14	16	15	13	11
DL_Sensor	11,572	84	718	390	83	69
Abalone	1,223	37	34	110	33	27
Magic	6,900	220	939	907	217	131
Letter	11,323	772	1,733	1,250	780	541
Ionosphere	435	173	162	515	168	153
Statlog	8,274	516	518	1,500	499	451
Waveform	7,281	2,775	4,477	17,843	2,796	2,746
Optical	13,163	2,427	5,025	11,156	2,360	2,256
SECOM	36,361	2,105	3,518	16,531	2,072	1,752

From table VII, we can find that the LBT and OST algorithms take more preprocessing time than others. The preprocessing time of LBT algorithm is very sensitive to the number of data points. For a data set with a large number of data points, such as "DL_Sensor", "Magic" and "Letter" data sets, the LBT algorithm requires a long time to create a lower-bound tree. The preprocessing time of the OST is highly influenced by the number of dimensions. For a data set with a large number of dimensions, it takes a lot of computation time to evaluate the orthogonal basis for the data set. In table VIII, we can see that

the OST algorithm performs well for all data sets and the MPAT ranks second.

From this experiment, we can conclude that the OST algorithm is a good choice for most data sets. However, in the case of a data set with a huge number of dimensions and the preprocessing time is important, the MPAT algorithm is a better choice than the OST algorithm. From [7], we know that the performance of the MPAT algorithm is highly influenced by the number of data points in leaf nodes. In the case of the MPAT algorithm is applied, a suitable n_c must be chosen to ensure that the number of data points in leaf nodes is large to achieve a good performance.

5 Conclusions

The performance of fast *kNN* search algorithms is highly influenced by the number of dimensions, number of data points, and data distribution of a data set. To help understand the performance of fast *kNN* search algorithms on data sets of different types, five fast *kNN* search algorithms were tested in this paper using data sets from real images and UC Irvine machine learning repository. From the experimental results, we can find that the OST algorithm performs well for most cases. For a data set from real images, the LAI algorithm is also a good choice. If a data set with a huge number of dimensions and the preprocessing time is important, the MPAT method becomes the best choice.

6 Acknowledgement

This work is supported by a grant from the National Science Council (NSC-99-2221-E-343-006).

7 References

[1] S. Theodoridis and K. Koutroumbas, Pattern Recognition, Academic Press, 2nd edition, 2003.

[2] Yi-Ching Liaw, "Improvement of the fast exact pairwise-nearest-neighbor algorithm," Pattern Recognition, vol. 42, no. 5, pp. 867-870, May 2009.

[3] Chih-Tang Chang, Jim Z. C. Lai, and M. D. Jeng, "Fast agglomerative clustering using information of k-nearest neighbors," Pattern Recognition, vol.43, no. 12, pp. 3958-3968, December 2010.

[4] J. McNames, "A fast nearest-neighbor algorithm based on a principal axis search tree," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 9, pp. 964-976, September 2001.

[5] J. Z. C. Lai, Y. C. Liaw, and J. Liu, "Fast k-nearest-neighbor search based on projection and triangular Inequality," Pattern Recognition, vol. 40, no.1, pp. 351-359, January 2007.

[6] Yong-Sheng Chen, Yi-Ping Hung, Ting-Fang Ten, Chiou-Shann Fuh, "Fast and versatile algorithm for nearest neighbor search based on a lower bound tree," Pattern Recognition, vol. 40, no. 2, pp. 360-375, February 2007.

[7] Yi-Ching Liaw, Chien-Min Wu, and Maw-Lin Leou, "Fast K-Nearest Neighbors Search Using Modified Principal Axis Search Tree," Digital Signal Processing, Vol. 20, Issue 5, pp. 1494-1501, September 2010.

[8] Yi-Ching Liaw, Maw-Lin Leou, and Chien-Min Wu, "Fast exact k nearest neighbors search using an orthogonal search tree," Pattern Recognition, Vol. 43, No. 6, pp. 2351-2358, June 2010.

[9] http://sipi.usc.edu/database.

[10] Y. Linde, A. Buzo, and R. M. Gray, “An algorithm for vector quantizer design,” IEEE Trans. on Communications, vol. 28, no. 1, pp. 84-95, January 1980.

[11] http://archive.ics.uci.edu/ml/index.html.

[12] http://www.cs.nctu.edu.tw/~yschen/software.html.

國科會補助計畫衍生研發成果推廣資料表

日期:2011/09/30

國科會補助計畫	計畫名稱: 快速k個最近鄰居搜尋之效能檢測資料庫與演算法開發
	計畫主持人: 廖怡欽
	計畫編號: 99-2221-E-343-006- 學門領域: 影像處理

無研發成果推廣資料

99 年度專題研究計畫研究成果彙整表

計畫主持人：廖怡欽	計畫編號：99-2221-E-343-006-
計畫名稱：快速 k 個最近鄰居搜尋之效能檢測資料庫與演算法開發	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	2	0	50%	篇	本計劃相關結果發表在 Digital Siginal Processing and Pattern Recognition 期刊.
		研究報告/技術報告	0	0	100%		
		研討會論文	1	0	50%		kNN 快速搜尋方法的相關應用, 發表在 2011CVGIP.
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	9	0	100%		本系沒有研究生, 因此找大學部學生幫忙搜集整理及分析真實資料集, 以及幫忙寫計劃相關程式.
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	1	0	100%		kNN 快速搜尋方法效能分析結果發表在美國舉辦的 IPCV 2011 研討會.
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		

	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	1.2010~2011 擔任國際期刊論文審稿委員: Circuits, Systems &amp； Signal Processing, Computational Statistics and Data Analysis, Information- International Information Institute, IEEE Transactions on Circuits and Systems for Video Technology, Journal of Visual Communication and Image Representation, Microscopy Research and Technique, Pattern Recognition. 2.獲得獎項: 99 年度國科會’’’’ 補助大專校院獎勵特殊優秀人才措施案’’’’ 100 年度國科會’’’’ 補助大專校院獎勵特殊優秀人才措施案’’’’ Marquis Who’s Who in the Asia 2012 Marquis Who’s Who in the World 2012

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：■已發表 □未發表之文稿 □撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

k 鄰居搜尋方法是用來在一群資料點中找尋一給定查詢點 k 個最近鄰居的方法，k 鄰居搜尋方法非常耗時，為了加快 k 鄰居的搜尋速度，許多快速 k 鄰居搜尋方法已被提出。現有快速 k 鄰居搜尋方法的效能很容易受到資料集的維度、資料量、以及資料分佈情形影響，在極端的情況下，快速 k 鄰居搜尋方法的效能可能比完全搜尋方法還差。為了解決這個問題，本計劃提出兩個快速 k 鄰居搜尋方法。為了幫助了解在不同種類資料集下各種快速 k 鄰居搜尋方法的效能表現，本計劃使用多種不同的真實資料集，測試 5 種快速 k 鄰居搜尋方法。實驗結果顯示，本計劃所提方法在不同維度、不同資料量、以及不同資料分佈情形的資料集下，大部份情況下，表現均比現有快速 k 鄰居搜尋方法更好。本計劃的結果非常適合用來為未知資料集挑選快速 k 鄰居搜尋方法。