Probabilistic and Interactive Retrieval of Chinese Calligraphic Character Images Based on Multiple Features*

Yi Zhuang[1], Nan Jiang[2], Hua Hu[3], Haiyang Hu[3], Guochang Jiang[4], and Chengxiang Yuan[1]

[1] College of Computer & Information Engineering, Zhejiang Gongshang University, P.R. China
[2] Hangzhou No.1 People's Hospital, P.R. China
[3] School of Computer, Hangzhou Dianzi University, P.R. China
[4] The Second Institute of Oceanography, SOA, P.R. China
zhuang@zjgsu.edu.cn

Abstract. This paper proposes an efficient probabilistic indexing scheme called Probabilistic Multiple-Feature-Tree(PMF-Tree) to facilitate an interactive retrieval of Chinese calligraphic manuscript images based on multiple features such as contour points, character styles and types of character. Different from conventional character retrieval and indexing methods [18] which only adopts shape similarity as a query metric, our proposed indexing algorithm allows user to choose the above three kinds of features they prefer to as query elements. Moreover, a probabilistic modal is introduced to refine the retrieval result. Comprehensive experiments are conducted to testify the effectiveness and efficiency of our proposed retrieval and indexing methods respectively.

Keywords: Chinese calligraphic character, high-dimensional indexing, probabilistic retrieval.

1 Introduction

Chinese historical calligraphy work is a valuable part of the Chinese cultural heritage. To effectively protect these works from ruining or damage, they have been digitized to store permanently. The issue of retrieval and indexing of such digital works becomes new challenges. As shown in Fig. 1, the state-of-the-art character retrieval and indexing methods only use contour points extracted from a character as a feature in the similarity retrieval. The styles, types, even number of strokes of a character, however, have not been adopted to facilitate the character retrieval, which can be also as two effective features to prune search region in the retrieval processing.

* This paper is partially supported by the Program of National Natural Science Foundation of China under Grant No. 60003047, No.60873022, No.60903053; The Program of Natural Science Foundation of Zhejiang Province under Grant No. Z1100822, No.Y1080148, No.Y1090165; The Science Fund for Young Scholars of Zhejiang Gongshang University under Grant No. G09-7. The Science & Technology Planning Project of Wenzhou under Grant No. 2010G0066.

© Springer-Verlag Berlin Heidelberg 2011

In essence, the efficient retrieval of Chinese calligraphic characters directly relates to the category of high-dimensional data indexing with multiple features. Although considerable research efforts have been done on the high-dimensional indexing issue [10], unfortunately, the existing high-dimensional indexing methods can not be directly applied to Chinese calligraphic characters [18]. In our previous work [18], we studied the character retrieval and its indexing algorithms only based on their contour similarity measure. However, in Figs. 2 and 3, for a same character "书", there are two different styles(i.e., *Yan Ti* and *Mi Ti*, etc) and two different types (i.e., *Kai Su* and *Li Su*, etc). In most cases, people would like to get some result characters with specific style or type they prefer to, which can also be regarded as effective pruning criteria to reduce a search region.

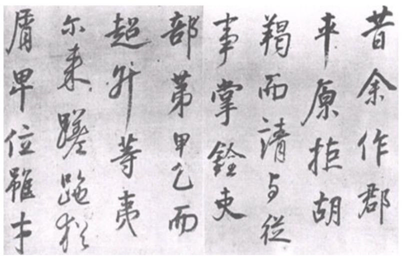


Fig. 1. A calligraphic character images

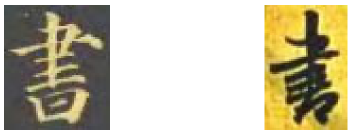

(a). Yan Ti (b). Mi Ti

Fig. 2. Two different styles of a same character

(a). Kai Su (b). Li Su

Fig. 3. Two different types of a same character

In addition, due to the shape complexity of the Chinese calligraphic character, personal knowledge level and errors occurred in the feature extraction, as shown in Tables 1-3, for the character "书", the identifications of its corresponding features such as character type, style, even number of strokes are not trivial, and the precision ratio of character retrieval is not so accurate. For example, for the character in Table 1, the probabilities of the character style are *Yan Ti*, *Mi Ti* and *Liu Ti* are 10%, 85% and 5%, respectively. So the introduction of probability modal can further refine the retrieval results.

Table 1. The probability of style of"书"

	Style	***Probability***
	Yan Ti	10%
	Mi Ti	85%
	Liu Ti	5%

Table 2. The probability of type of"书"

	Type	***Probability***
	Kai Su	5%
	Li Su	5%
	Cao Su	85%
	Xing Su	5%

Table 3. The probability of number of strokes of"书"

	Number of strokes	***Probability***
	9	15%
	10	80%
	11	5%

From the above discussion, in this paper, based on the shape-similarity-based retrieval algorithm for Chinese calligraphic character in our previous work [18], we propose a probabilistic and composite high-dimensional indexing scheme based on multiple features, called PMF-Tree, which is specifically designed for indexing the large Chinese calligraphic characters. With the aid of the PMF-Tree index, a probabilistic k-nearest neighbor query of character λ_q in high-dimensional spaces is transformed into a range query in the single dimensional space.
The primary contributions of this paper are as follows:

1. We propose a novel probabilistic interactive retrieval method to effectively support the Chinese calligraphic characters retrieval by choosing multiple features of character.
2. We introduce a *Probabilistic Multiple-Feature-Tree*(*PMF-Tree*)-based indexing method to facilitate the interactive and efficient Chinese calligraphic characters retrieval with multiple features.

The remainder of this paper is organized as follows. In Sections 2, we provide background of our work. Then in Section 3, we propose a *Probabilistic Multiple-Feature-Tree*(PMF-Tree)- based high-dimensional indexing scheme to dramatically speed up the retrieval efficiency. In Section 4, we report the results of extensive experiments which are designed to evaluate the efficiency and effectiveness of the proposed approach. Finally, we conclude in the final section.

2 Background

Numerous promising research works have been done on the handwriting recognition [5]. For instance, a word-matching technology is used to recognize George Washington's manuscripts [3], and the historical Hebrew manuscripts were identified in [4]. However, no published research work has been done successfully on Chinese calligraphic character retrieval because it differs from other languages by its enormous numbers and complex structure of ideographs. In [6] Shi et al. have shown a content-based retrieval method for antique books, however it is unknown how well this rigid visual similarity-based method works on the Chinese calligraphic characters retrieval with different styles of handwritings in different dynasties. Belongie et al. [8] have proposed an inspirational and similar approach to ours, yet it is much more complex at least for calligraphic character retrieval. Our earlier work includes applying the Projecting method [2] and the Earth Movers' Distance (EMD) method [9] to the Chinese calligraphic character retrieval. However, these two recognition techniques are too rigid to be applied to the retrieval process. For calligraphic character retrieval, shape is a promising feature to model a character. So in this paper, we adopt a shape-similarity(*SS*)-based retrieval method proposed in our previous work [18] as a similarity measure between two characters.

There is a long stream of research on solving the high-dimensional indexing problems [10]. The R-tree [11] and its variants [12], etc are based on data & space partitioning, hierarchical tree index structure, however their performance deteriorates rapidly as dimensionality increases due to the "*dimensionality curse*" [10]. Another category is to represent original feature vectors using smaller, approximate representations, e.g., VA-file [13] and IQ-tree [14], etc. Although these methods accelerate the sequential scan by using data compression, they incur higher computational cost to decode the bit-string. The above two categories of indexing approaches, however, are only suitable for indexing the multi-dimensional data with the fixed dimensionality and does not fit for indexing the character features since almost every character has different number of contour points (dimensionalities). The distance-based approach (e.g., iDistance [15]) may be a promising scheme to indexing them since it does not heavily depend on the dimensionalities of the characters.

3 The PMF-Tree Index

In order to improve the probabilistic retrieval efficiency, in this section, we develop a novel probabilistic high-dimensional indexing technique, called the *Probabilistic Multiple-Feature Tree*(*PMF-Tree* for short), to accelerate the retrieval process.

3.1 Preliminaries

The design of the *PMF-Tree* is motivated by the following key observations. First, conventional character retrieval and indexing methods [18] are only based on similarity of two characters without considering the other factors such as number of strokes, styles, etc. So the effectiveness of these methods are not satisfactory and their query performances are not well scaled for large dataset due to the CPU-intensive distance computation in the retrieval process [18]. Second, the feature uncertainty has not been studied in the state-of-the-art character retrieval techniques, which may enable the query results to be more accurate and objective.

First we briefly introduce the notations that will be used in the rest of paper.

Table 4. Meaning of Symbols Used

Symbols	Meaning
Ω	a set of Chinese calligraphic character
λ_i	the i-th character and $\lambda_i \in \Omega$
M	the number of contour points from a character
N	the number of characters in Ω
λ_q	a query character user submits
$\Theta(\lambda_q,r)$	a query sphere with centre λ_q and radius r
$Sim(\lambda_i,\lambda_j)$	the distance between two characters defined in [18]
ε	Threshold value

As a preliminary step, before constructing the PMF-Tree, all characters in Ω are first grouped into T clusters by an AP-Cluster algorithm [16]. Each cluster is denoted as C_j, in which the centroid character(O_j) in C_j can be adaptively selected by the algorithm [16], where $j \in [1,T]$. So we can model a cluster as a tightly bounded sphere described by its *centroid* and *radius*, which is saved in a class information file.

Definition 1 (Cluster Radius). *Given a cluster Cj, the distance between Oj and the character which has the longest distance to Oj is defined as the cluster radius of Cj, denoted as CRj.*

Given a cluster *Cj*, the cluster sphere of it is denoted as $\Theta(Oj,CRj)$, where *Oj* is the centroid character of cluster *Cj*, *CRj* is the cluster radius.

Definition 2 (Centroid Distance). *Given a character λi, its centroid distance is defined as the distance between λi and Oj, the centre of cluster that λi belongs to:*

$$CD(\lambda_i)=Sim(\lambda_i,O_j) \tag{1}$$

where $j\in[1,T]$, $i\in[1,\delta]$, *and δ is the number of characters in C_j.*

Once *T* clusters are obtained, then the centroid distance and the number of strokes of each character are computed. Moreover, its style and the type are identified at the same time. Finally, a uniform index key of a character is obtained, which is inserted by a B$^+$-Tree.

As we know, for a same character, there are a number of different styles and types, respectively (see Figs. 2 and 3). To embed these two information into the unified index key that will be discussed in Section 3.2, two encoding schemes of the style and the type are needed which is shown in the following tables.

Table 5. Style of character

Style Name	Yan Ti	Liu Ti	Cai Ti	Su Ti	…
Style ID	1	2	3	4	…

Table 6. Type of character

Type Name	Li Su	Kai Su	Cao Su	…
Type ID	1	2	3	…

3.2 The Data Structure

In order to effectively prune the search region, we propose the *PMF-Tree*, an probabilistic multiple-feature indexing scheme in which the high-dimensional index for contour points is based on the iDistance[15]. As mentioned before, all characters are first grouped into *T* clusters using an AP-Cluster algorithm [16], then the *centroid-distance* and the number of strokes of each character are computed, the style and type of each character can be identified by user in preprocessing step. Thus the character λi can be modeled by a six-tuple:

$$\lambda_i ::= < i,\ CID,\ CD,\ Style,\ Type,\ Num > \tag{2}$$

where

- *i* refers to the *i*-th character in Ω;
- *CID* is the ID number of the cluster λi belongs to;
- *CD* is the centroid distance of λi;
- *Style* $=\{StyID, P_s\}$, where *StyID* is the style ID of λi, and P_s=**Prob**(the style ID of λi is *StyID*);
- *Type* $=\{TyID, P_t\}$, where *TyID* is the type ID of λi, and P_t=**Prob**(the type ID of λi is *TyID*)
- *Num* $=\{NumS, Pn\}$, where *NumS* is the number of strokes of λi, and Pn=**Prob**(the number of strokes of λ_i is *Num*)

For each character λ_i in a cluster sphere, its index key can be defined as:

$$key(\lambda_i)=CD(\lambda_i) \tag{3}$$

Since the characters are grouped into T clusters, to get a uniform index key of image in different clusters, the index key in Eq. (8) can be rewritten by Eq. (9):

$$key(\lambda_i)=CID+{CD(\lambda_i)}/{MAX} \tag{4}$$

where the CID is the ID number of cluster λ_i falls in.

Note that since $CD(\lambda_i)$ may be larger than one, the value of $CD(\lambda_i)$ should be normalized into the range of [0,1] by being divided a large constant MAX. Thus, it is guaranteed that the search range of centroid distance of each character can not be overlapped.

To facilitate retrieving characters via submitting an auxiliary information (e.g, the style name, type name or number of strokes) of λ_i, its index keys can be derived in Eqs. (5) and (7), respectively:

$$KEY(\lambda_i)=\alpha*StyID(\lambda_i)+P_S \tag{5}$$

$$KEY(\lambda_i)=\beta*TyID(\lambda_i)+P_T \tag{6}$$

$$KEY(\lambda_i)=\gamma*NumS(\lambda_i)+P_N \tag{7}$$

Where α, β and γ are three stretch constants which are set 10, 10^2 and 10^3 respectively.

In the above, we suppose that user submits two query elements (e.g., (a) *StyID* and λ_i, (b) *TyID* and λ_i, or (c) *NumS* and λ_i). If user submits three query elements (*TyID*, *StyID* and λ_i,), then a uniform index key for λ_i can be rewritten below:

$$KEY(\lambda_i)=\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+P_T*P_S \tag{8}$$

$$KEY(\lambda_i)=\alpha*StyID(\lambda_i)+\gamma*NumS(\lambda_i)+P_S*P_N \tag{9}$$

$$KEY(\lambda_i)=\beta*TyID(\lambda_i)+\gamma*NumS(\lambda_i)+P_T*P_N \tag{10}$$

Similarly, for four query elements (*TyID*, *StyID, NumS* and λ_i,) submitted by a user, a uniform index key for λ_i can be derived as follows:

$$KEY(\lambda_i)=\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+\gamma*NumS(\lambda_i)+P_T*P_S*P_N \tag{11}$$

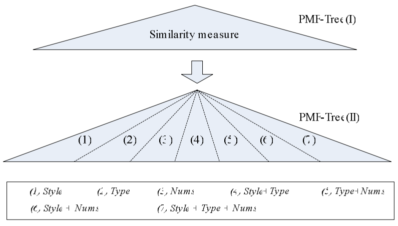


Fig. 4. The PMF-Tree index structures

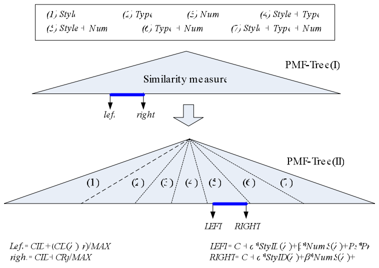


Fig. 5. The search range in B+-Tree

Eqs. (5-11) represent the index keys of character respectively, which correspond to seven independent indexes. In order to incorporate them into an integral index, we derive a new uniform index key expression by adding seven stretch constants(i.e., C_1 to C_7), which is shown in Eq. (12):

$$KEY(\lambda_i)=\begin{cases} C_1+\alpha*StyID(\lambda_i)+P_S & \text{(a)} \\ C_2+\beta*TyID(\lambda_i)+P_T & \text{(b)} \\ C_3+\gamma*NumS(\lambda_i)+P_N & \text{(c)} \\ C_4+\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+P_T*P_S & \text{(d)} \\ C_5+\alpha*StyID(\lambda_i)+\gamma*NumS(\lambda_i)+P_S*P_N & \text{(e)} \\ C_6+\beta*TyID(\lambda_i)+\gamma*NumS(\lambda_i)+P_T*P_N & \text{(f)} \\ C_7+\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+\gamma*NumS(\lambda_i)+P_T*P_S*P_N & \text{(g)} \end{cases} \tag{12}$$

where C_1=0, C_2=1×10^4, C_3=1.5×10^4, C_4=2×10^4, C_5=2.5×10^4, C_6=3×10^4, and C_7=3.5×10^4.The above seven constants should be set large enough to stretch the value ranges of the index keys so that they do not overlap with each other.

3.3 Building PMF-Tree

For a character, its four values of *CD, NumS, styID* and *tyID* are recorded in the corresponding index key of PMF-Tree whose basic structure is the B$^+$-tree, which is shown in Fig. 4. Fig. 6 shows the detail steps of constructing a PMF-Tree. Note that the routines ***TransDis***(λ_i) and ***TransDis*1**(λ_i) are two distance transformation function in Eq.(4) and Eq. (12) respectively, and ***BInsert***(*key*,*bt*) is a B$^+$-tree insert procedure.

```
Algorithm 1. PMF-Tree Index Construction
Input: Ω: the character set;
Output: bt and bt': the index for PMF-Tree(I) and (II);
1.  The characters in Ω are grouped into T clusters using the AP cluster algorithm
2.  bt←newFile(), bt'←newFile(); /* create index header file for PMF-Tree(I),(II)*/
3.  for each character $\lambda_i \in \Omega$ do
4.      The CD of $\lambda_i$ are computed;
5.      The style, type and stroke number of the character are identified by user with probabilities;
6.      key($\lambda_i$)=TransDis($\lambda_i$);     /* Function TransDis() is shown in Eq. (4) */
7.      KEY($\lambda_i$)=TransDis1($\lambda_i$);    /* Function TransDis1() is shown in Eq. (12) */
8.      BInsert(key($\lambda_i$), bt);          /* insert it to B+-tree */
9.      BInsert(KEY($\lambda_i$), bt');         /* insert it to B+-tree */
10. return bt and bt'
```

Fig. 6. The index construction algorithm for PMF-Tree

3.4 Probabilistic *k*-NN Search Algorithm

For *n* high-dimensional characters, a probabilistic *k*-Nearest-Neighbor(P*k*-NN) search is a most frequently used search operation which retrieves the *k* most similar characters that are closest in distance to a given character with a probabilistic threshold. In this section, we will focus on P*k*-NN searches of Chinese calligraphic character. For example, when user submits a query character “国” and a threshold ε, its type name and the style name of the result characters are *Kai Su* and *Song Ti* respectively. Then as shown in Figure 5, the retrieval process is composed of two steps: 1). Candidate characters returned by retrieving over the PMF-Tree(I) in which the range is [*left*, *right*], where $left=CID+(CD(\lambda_i)-r)/MAX$, $right=CID+CR_j/MAX$; 2). Retrieval over the PMF-Tree(II), whose range is [*LEFT*, *RIGHT*], where $LEFT=C_5+\alpha*tyID(\lambda_i)+\beta*StyID(\lambda_i)+P_t*P_s$, $RIGHT=C_5+\alpha*tyID(\lambda_i)+\beta*StyID(\lambda_i)+1$;

```
Algorithm 2. PkNN Search
Input: query character $\lambda_q$, k, StyID or TyID or NumS, $\varepsilon$
Output: query results S
1.  $r \leftarrow 0, S \leftarrow \Phi$;          // initialization
2.  while (|S|<k)          // |S| refers to the number of candidate characters in S
3.      $r \leftarrow r+\Delta r$;
4.      $S \leftarrow$ RSearch($\lambda_q$,r);
5.      if (|S|>k) then
6.          for i:=1 to |S|-k do
7.              $\lambda_{far} \leftarrow$ Farthest(S, $\lambda_q$);
8.              $S \leftarrow S-\lambda_{far}$;
9.  return S;
RSearch($\lambda_q$,r)
10.  $S1 \leftarrow \Phi, S2 \leftarrow \Phi$;
11.  for each cluster sphere $\Theta(O_j,CR_j)$ and $j \in [1, T]$
12.      if $\Theta(O_j,CR_j)$ contains $\Theta(\lambda_q,r)$ then
13.          $S1 \leftarrow S1 \cup$ Search($\lambda_q$,r, j);
14.          end loop
15.      else if $\Theta(O_j, CR_j)$ intersects $\Theta(\lambda_q,r)$ then
16.          $S1 \leftarrow S1 \cup$ Search($\lambda_q$,r, j);
17.  $S5 \leftarrow$ Search1(StyID, TyID, NumS and i);
18.  for each character $\lambda_i \in S1$ do
19.      if $\lambda_i \in$ S5 then $S1 \leftarrow S1 \cup \lambda_i$
20.  return S1;          // return candidate characters
Search($\lambda_q$,r, i)
21.  $left \leftarrow i+(CD(\lambda_q)-r)/MAX$,   $right \leftarrow i+CR_j/MAX$;
22.  $S3 \leftarrow$ BRSearch[left, right];   // the filtering step
23.  for each character $\lambda_j \in S3$ do
24.      if $Sim(\lambda_q, \lambda_j)>r$ then $S3 \leftarrow S3-\lambda_j$;   // the refinement stage
25.  return S3;          // return the candidate character set
Search1(StyID, TyID, NumS and i)
26.  if user submits a $\lambda_q$ and its style then
27.      $LEFT \leftarrow C_1+\alpha*StyID(\lambda_i)+Ps$, $RIGHT \leftarrow C_1+\alpha*StyID(\lambda_i)+1$;
28.  else if user submits a $\lambda_q$ and its type then
29.      $LEFT \leftarrow C_2+\alpha*TyID(\lambda_i)+Pt$; $RIGHT \leftarrow C_2+\alpha*TyID(\lambda_i)+1$;
30.  else if user submits a $\lambda_q$ and the number of strokes then
31.      $LEFT \leftarrow C_3+\alpha*NumS(\lambda_i)+Pn$, $RIGHT \leftarrow C_3+\alpha*NumS(\lambda_i)+1$;
32.  else if user submits a $\lambda_q$ , its style and the number of strokes then
33.      $LEFT \leftarrow C_4+\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+P_s*P_n$, $RIGHT \leftarrow C_4+\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+1$;
34.  else if user submits a $\lambda_q$ , its type and the number of strokes then
35.      $LEFT \leftarrow C_5+\alpha*StyID(\lambda_i)+\beta*NumS(\lambda_i)+P_t*P_n$, $RIGHT \leftarrow C_5+\alpha*StyID(\lambda_i)+\beta*NumS(\lambda_i)+1$;
36.  else if user submits a $\lambda_q$ , its type and style then
37.      $LEFT \leftarrow C_6+\alpha*tyID(\lambda_i)+\beta*NumS(\lambda_i)+Pt*Ps$, $RIGHT \leftarrow C_6+\alpha*tyID(\lambda_i)+\beta*NumS(\lambda_i)+1$;
38.  else if user submits a $\lambda_q$, its type, style and the number of strokes then
39.      $LEFT \leftarrow C_7+\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+\gamma*NumS(\lambda_i)+Pt*Ps*Pn$;
40.      $RIGHT \leftarrow C_7+\alpha*StyID(\lambda_i)+\beta*TyID(\lambda_i)+\gamma*NumS(\lambda_i)+1$;
41.  $S4 \leftarrow$ BRSearch[LEFT, RIGHT];   // the filtering step
42.  return S4;          // return the candidate character set
```

Fig. 7. P*k*-NN search algorithm

Figure 7 details the whole search process. Routine ***RSearch***() is the main range search function which returns the candidate characters of range search with centre λq and radius r with probability larger than ε, ***Search*()** and ***Search*1()** are the implementation of the range search. ***Farthest***() returns the character which is the longest from λq in S. ***BRSearch***() is a B+-tree range search function.

4 Experimental Results

In this section, we present an extensive performance study to evaluate the effectiveness and efficiency of our proposed retrieval and indexing method. The Chinese Calligraphic characters image data we used are from *CADAL Project* [17] which contains a set of contour point features extracted from the 12,000 character images in which each feature point is composed of a pair of coordinates <*x axis*, *y axis*>. We implemented the shape-similarity-based retrieval approach and the PMF-Tree index in *C* language in which a B+-tree is as the single dimensional index structure. The index page size of B+-tree is set to 4096 Bytes. All the experiments are run on a Pentium IV CPU at 2.0GHz with 1G Mbytes memory. In our evaluation, we use the number of page accesses and the total response time as the performance metric.

4.1 Effectiveness of the Retrieval Method

In the first experiment, we have implemented an online interactive retrieval system for Chinese calligraphic characters to testify the effectiveness of our proposed retrieval method comparing with the conventional one [18]. As shown in the right part of Figure 8, when user submits an example Chinese calligraphic character by drawing a character "天" and the number of strokes (e.g., 4) as well, the query radius and a threshold value are set 0.8, 60% respectively, the candidate characters are quickly retrieved by the system with the aid of the PMF-Tree. The left part of the figure is the query result in which the similarity and confidence values of the answer character images are given with respective to the query one.

Figure 9 illustrates a *Recall-Precision* curve for the performance comparisons of the *shape- based* method [18] and our proposed composite search. It compares the average retrieval result (the average precision rate under the average recall rate) of 20 characters queries randomly chosen from the database. Each of them has more than 4 different calligraphic styles and types. The figure shows that the retrieval performance of the composite search is better than that of the shape-based one by a large margin.

4.2 Efficiency of PMF-Tree Index

In the following, we test the performance of our proposed indexing method—PMF-Tree under different sizes of databases and different selectivity.

4.2.1 Effect of Data Size

In this experiment, we measured the performance behavior with varying number of characters. Figure 10a shows the performance of query processing in terms of CPU cost. It is evident that PMF-Tree outperforms sequential scan significantly. The CPU cost of PMF-Tree increases slowly as the data size grows. It's worth mentioning that the CPU cost of sequential scan is ignored since the computation cost of it is very expensive. In Figure 10b, the experimental result reveals that the I/O cost of PMF-Tree is superior to sequential scan.

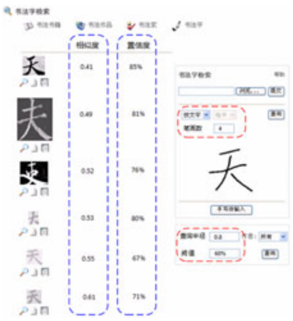


Fig. 8. One retrieval example

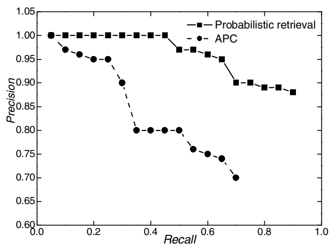
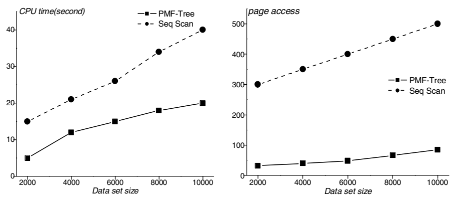
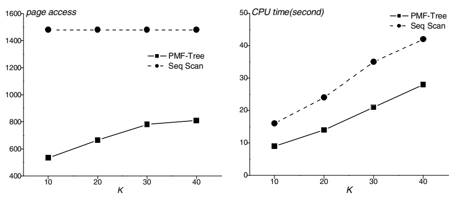


Fig. 9. Recall vs. precision



(a) CPU cost *vs.* Data size (b) I/O cost *vs.* Data size

Fig. 10. Effect of data size



(a) *K vs.* Page Access (b) *K vs.* CPU Cost

Fig. 11. Effect of *k*

4.2.2 Performance Behavior with k(Selectivity)

In this section, we proceed to evaluate the effect of *k* (selectivity) on the performance of a P*k*-NN retrieval by using the PMF-Tree. Figures 11a and 11b both indicate that when *k* ranges from 10 to 40, the PMF-Tree is superior to other methods in terms of page access and the CPU cost. The results conform to our expectation that the search region of PMF-Tree is significantly reduced and the comparison between any two characters is a CPU-intensive task. The CPU cost of sequential scan is ignored due to the expensive computation cost of it.

5 Conclusions

In this paper, we proposed a novel probabilistic and interactive multiple-feature-based indexing scheme to support large-scale historical Chinese calligraphic character images retrieval. Two main components are included, such as 1). an effective approach to probabilistic retrieving Chinese calligraphic characters by choosing three kinds of the features is introduced; 2). a novel multiple-feature-tree(*PMF-Tree*)-based probabilistic high-dimensional indexing scheme is then proposed to boost the retrieval performance of the large Chinese calligraphic characters. The prototype retrieval system is implemented to demonstrate the applicability and effectiveness of our new approach to Chinese calligraphic character retrieval.

References

[1] Zhang, X.-Z.: Chinese Character Recognition Techniques. Tsinghua University Press, Beijing (1992)

[2] Wu, Y.-S., Ding, X.-Q.: Chinese character recognition: the principles and the implementations. High Education Press, Beijing (1992)

[3] Rath, T.M., Manmatha, R., Lavrenko, V.: A search engine for historical manuscript images. In: SIGIR, pp. 369–376 (2004)

[4] Yosef, I.B., Kedem, K., Dinstein, I., Beit-Arie, M., Engel, E.: Classification of Hebrew Calligraphic Handwriting Styles: Preliminary Results. In: DIAL 2004, pp. 299–305 (2004)

[5] Palmondon, R., Srihari, S.N.: On-Line and Off-Line hand-writing Recognition: A Comprehensive Survey. IEEE Trans. on PAMI 22(1), 63–84 (2000)

[6] Shi, B.-l., Zhang, L., Wang, Y., Chen, Z.-F.: Content Based Chinese Script Retrieval Through Visual Similarity Criteria. Chinese Journal of Software 12(9), 1336–1342 (2001)

[7] Chui, H.-l., Rangarajan, A.: A new point matching algorithm for non-rigid registration. CVIU 89(2-3), 114–141 (2003)

[8] Belongie, S., Malik, J., Puzicha, J.: Shape Matching and Object Recognition Using Shape Contexts. IEEE Trans. on PAMI 24(4), 509–522 (2002)

[9] Cohen, S., Guibas, L.: The Earth Mover's Distance under Transformation Sets. In: ICCV, Corfu, Greece, pp. 173–187 (September 1999)

[10] Böhm, C., Berchtold, S., Keim, D.: Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. ACM Computing Surveys 33(3) (2001)

[11] Guttman, A.: R-tree: A dynamic index structure for spatial searching. In: SIGMOD, pp. 47–54 (1984)

[12] Berchtold, S., Keim, D.A., Kriegel, H.P.: The X-tree: An index structure for high-dimensional data. In: VLDB, pp. 28–37 (1996)

[13] Weber, R., Schek, H., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB, pp. 194–205 (1998)

[14] Berchtold, S., Bohm, C., Kriegel, H.P., Sander, J., Jagadish, H.V.: Independent quantization: An index compression technique for high-dimensional data spaces. In: ICDE, pp. 577–588 (2000)

[15] Jagadish, H.V., Ooi, B.C., Tan, K.L., Yu, C., Zhang, R.: iDistance: An Adaptive B+-tree Based Indexing Method for Nearest Neighbor Search. ACM Trans. on Database Systems 2(30), 364–397 (2005)

[16] Frey, B.J., Dueck, D.: Clustering by Passing Messages Between Data Points. Science 315, 972–976

[17] 2010, http://www.cadal.zju.edu.cn

[18] Zhuang, Y., Zhuang, Y.-T., Li, Q., Chen, L.: Interactive high-dimensional index for large Chinese calligraphic character databases. ACM Trans. Asian Lang. Inf. Process. 6(2) (2007)