Efficient Search in Document Image Collections

Anand Kumar[1], C.V. Jawahar[1], and R. Manmatha[2]

[1] Center for Visual Information Technology
International Institute of Information Technology
Hyderabad, India - 500032
anandkumar@research.iiit.ac.in, jawahar@iiit.ac.in

[2] Department of Computer Science
University of Massachusetts Amherst, MA 01003, USA
manmatha@cs.umass.edu

Abstract. This paper presents an efficient indexing and retrieval scheme for searching in document image databases. In many non-European languages, optical character recognizers are not very accurate. Word spotting - word image matching - may instead be used to retrieve word images in response to a word image query. The approaches used for word spotting so far, dynamic time warping and/or nearest neighbor search, tend to be slow. Here indexing is done using locality sensitive hashing (LSH) - a technique which computes multiple hashes - using word image features computed at word level. Efficiency and scalability is achieved by content-sensitive hashing implemented through approximate nearest neighbor computation. We demonstrate that the technique achieves high precision and recall (in the 90% range), using a large image corpus consisting of seven Kalidasa's (a well known Indian poet of antiquity) books in the Telugu language. The accuracy is comparable to using dynamic time warping and nearest neighbor search while the speed is orders of magnitude better - 20000 word images can be searched in milliseconds.

1 Introduction

Many document image collections are now being scanned and made available over the Internet or in digital libraries. Effective access to such information sources is limited by the lack of efficient retrieval schemes. The use of text search methods requires efficient and robust optical character recognizers(OCR), which are presently unavailable for Indian languages [1]. Another possibility is to search in the image domain using word spotting [2,3,4]. Direct matching of images is inefficient due to the complexity of matching and thus impractical for large databases. We solve this problem by directly hashing word image representations.

We present an efficient mechanism for indexing and retrieval in large document image collections. First, words are automatically segmented. Then features are computed at word level and indexed. Word retrieval is done very efficiently by using an approximate nearest neighbor retrieval technique called locality sensitive hashing (LSH). Word images are hashed into multiple tables with features computed at word level. Content-sensitive hash functions are used to hash words such that the probability of grouping similar words in the same index of the hash table is high. The sub-linear time content-sensitive hashing scheme makes the search very fast without degrading the accuracy. Experiments on a collection of Kalidasa's - the classical Indian poet of antiquity - books in Telugu demonstrate that 20,000 word images may be searched in a few milliseconds. The approach thus makes searching large document image collections practical.

There are essentially three classes of techniques to search document image collections. The first is based on using a recognizer to convert an image to text and then searching the results using a text search engine. An example is the gHMM approach of Chan *et al.* [5], suggested for printed and handwritten Arabic documents. It uses gHMMs with a bi-gram letter transition model, and KPCA/LDA for letter discrimination. In this approach segmentation and recognition go hand in hand. The words are modeled at letter level, where the likelihood of a word given a segment is used for discriminating words. The Byblos system [6] also uses a similar approach to recognize documents where a line is first segmented out and then divided in to image strips. Each line is then recognized using an HMM and a bi-gram letter transition model. The second used by Rath *et al.* [7], involves the automatic annotation of word images with a lexicon and probabilities using a relevance-based language model. Here, words are segmented out and then each word image is annotated using a statistical model with the entire lexicon and probabilities. A language model retrieval approach is then used to search the documents. The technique was successfully used to build a 1000 page demonstration for George Washington's handwritten manuscripts. The third approach proposed by Rath and Manmatha [2,3] involves using what is called word spotting, where word images are matched with each other and then clustered. Each cluster is then annotated by a person. Alternatively, Jawahar *et al.* [4] showed that in the case of printed books one can synthesize the query image from a textual query for making the system more usable.

Word spotting has been tried for many different kinds of documents both handwritten and print. Rath and Manmatha [2] used dynamic time warping (DTW) to compute image similarities for handwriting. The word similarities are then used for clustering using K-means or agglomerative clustering techniques. This approach was adopted in Jawahar *et al.* [4] for printed Indian language document images. To simplify the process of querying, a word image is generated for each query and the cluster corresponding to this word is identified. In such methods, efficiency is achieved by significant offline computation. Ataer and Duygulu [8] tried word spotting for handwritten Ottoman documents where they use successive pruning stages to eliminate irrelevant words. Gatos *et al.* [9] used word spotting for old Greek typewritten manuscripts for which OCRs did not work. One advantage of word spotting over traditional OCR methods is that they take advantage of the fact that within corpora such as books the word images are likely to be much more similar, which traditional OCRs do not do. In addition, techniques that work at the symbol level of word images, like [5], are very sensitive to segmentation errors. Segmentation of Indian language document images at symbol level is very difficult due to the complexity of the scripts.

Many of these techniques (for example DTW) are computationally expensive and do not scale very well. Inspite of this, Sankar *et al.* [10] successfully indexed 500 books in Indian languages using this approach by doing virtually all the computation off-line. Avoiding DTW, Rath *et al.* [3] demonstrated the use of direct clustering of word image features on historical handwritten manuscripts. However, clustering is itself an expensive operation.

Image matching often involves offline nearest neighbor computations and storage for efficient access. These nearest neighbor techniques are expensive in high dimensions even when computed off-line. Indyk and Motwani [11] proposed an approximate nearest neighbor search technique called locality sensitive hashing (LSH) which is much more efficient. LSH has been applied to a number of problems including some in computer vision. For example, LSH is used to efficiently index high dimensional pose examples by Shakhnarovich *et al.* [12]. Matei *et al.* [13] use LSH for 3D object indexing. LSH is different from the geometric hashing approaches used in model based recognition of 3-D objects in occluded scenes from 2-D gray scale images [14] and also for finding documents from a set of camera-based document images [15].

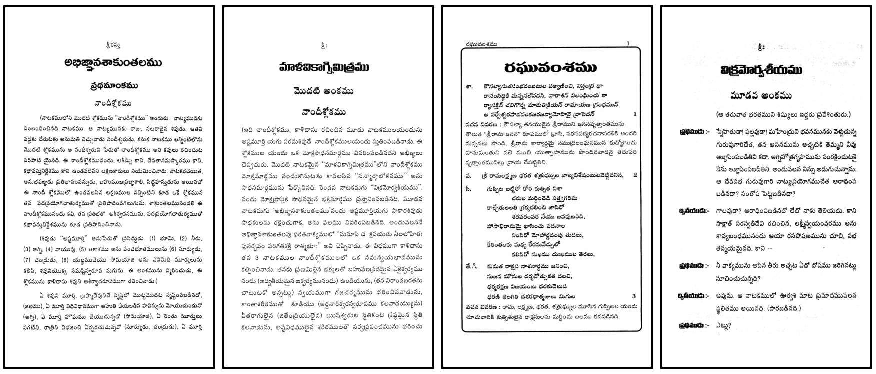


Fig. 1. Sample document images from Kalidasa's books in Telugu

Our work mainly aims at addressing some of the issues involved in effective and efficient retrieval in document images with effective representations of the word images. We demonstrate efficient retrieval through content-sensitive hashing on a collection of Kalidasa's writings. Sample pages from the Kalidasa collection are shown in Figure 1.

2 Content Sensitive Hashing

In the proposed retrieval technique, the index is built by hashing word level features of document images. The features are hashed using content sensitive hash functions, such that the probability of finding words with similar content in the same bucket is high. The same content sensitive hash functions are used to

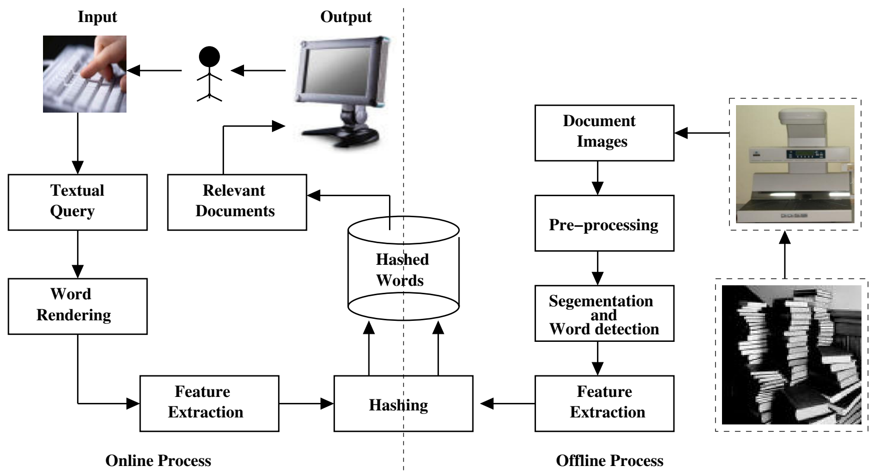


Fig. 2. Hashing Method: Word image hashing for efficient search. Showing offline pre-processing and on-line query processing stages.

query similar words during the search. The major challenges in efficient indexing and retrieval are the preprocessing and word matching times. We overcome these challenges with the use of hashing.

A conceptual block diagram of the technique is shown in Figure 2. Books are scanned and processed to index the document pages. The textual word query is first converted to an image by rendering, features are extracted from these images and then search is carried out to retrieve relevant word images. To facilitate searching, scanned document images are preprocessed and segmented at word level. A set of features are extracted as representatives of word images to be indexed. Content-sensitive hash functions are used to hash the features such that similar word images are grouped in the same index of the hash table.

2.1 Word Image Representation

We employ a combination of scalar, profile, structural and transform domain feature extraction methods as used in [2,3,4]. Scalar features include the number of ascenders, descenders and the aspect ratio. The profile and structural features include: projection profiles, background to ink transitions and upper and lower word profiles. Fixed length description of the features are obtained by computing lower order coefficients of a DFT (Discrete Fourier Transform) - discarding noisy high order coefficients makes the representation more robust. We use 84 Fourier coefficients of the segmented profiles and ink transition features to represent the word images.

Finding similar word images is now equivalent to the nearest neighbor search (NNS) problem: Given a set of n points $P = p_1, \ldots, p_n$ in some metric space X, we preprocess P so as to efficiently answer queries, which require finding the point in P closest to a query point $q \in X$. Traditional data structures for similarity search suffer from the curse of dimensionality. Locality sensitive hashing (LSH) is a state-of-the-art technique introduced by Indyk and Motwani [11] to alleviate the problem of high dimensional similarity search in large databases. The main idea in LSH is to hash points into bins based on a probability of collision. Thus, points that are far in the parameter space will have a high probability of landing into different bins, while close points will go into the same bucket. It has been shown that LSH out-performs tree-based structures such as the Sphere/Rectangle-tree (SR-tree) by at least an order of magnitude.

2.2 Hashing Technique

Let $P = \{x_1, x_2, \ldots, x_n\}$ be the words in the document image collection. A word is represented by a feature vector $x = \{f_1, \ldots, f_D\}$, represented as a point $x \in \mathbb{R}^D$ in feature space, where f_j is computed by extracting features that describe the content of the word images. The extracted features satisfy the following assumptions.

1. A distance function d is given which measures the content level similarity of the words, and a radius R in the feature space is given such that x_1, x_2 are considered similar iff $d(x_1, x_2) < R$.
2. For a randomly chosen word image, there exists a word image with high probability and similar feature values in the collection.
3. There are no significant variations in feature vectors of the words with similar content, or the feature extraction process is unbiased.

The distance function and the similarity threshold are dependent on the particular task, and often reflect perceptual similarities between the words. The last assumption implies that there are no significant sources of variation in the word features for words that are similar in content. The content similarity search is done by efficient nearest neighbor searching with content-sensitive hashing algorithm.

The content-sensitive hashing is achieved by hashing words using a number of hash functions from a family $H = \{h : S \to U\}$ of functions. H is called content-sensitive if for any q, the function

$$p(t) = Pr_H[h(q) = h(x) : ||q - v|| = t] \tag{1}$$

is strictly decreasing in t. That is, the probability of collision of points q and x is decreasing with content dissimilarity (distance) between them. We concatenate several hash functions $h \in H$. In particular define a function family $G = \{g : S \to U^k\}$ such that, $g(x) = (h_1(x), \ldots, h_k(x))$. For an integer L, the algorithm chooses L functions $g_1, \ldots, g_L$ from G, independently and uniformly at random. During preprocessing, the algorithm stores each input point in buckets $g_j(x)$, for all $j = 1, \ldots, L$. Since the total number of buckets may be large, the algorithm retains only the non-empty buckets by resorting to hashing.

Algorithm 1. Content Sensitive Hashing

```
Input: Word Images $W_j, j = 1, \ldots, n$
Output: Hash Tables $T_i, i = 1, \ldots, l$
 1: for each $i = 1, \ldots, l$ do
 2:   Initialize hash table $T_i$ with hash functions $g_i$
 3: end for
 4: for each $i = 1, \ldots, l$ do
 5:   for each $j = 1, \ldots, n$ do
 6:     Pre-process word image $W_j$ (noise removal etc).
 7:     Extract features $F_j$ of word image $W_j$.
 8:     Compute hash bucket $I = g_i(F_j)$
 9:     Store word image $W_j$ on bucket $I$ of hash table $T_i$
10:   end for
11: end for
```

A D dimensional word feature x is mapped onto a set of integers by each hash function $h_{a,b}(x)$. Each hash function in the family is indexed by a choice of random a and b, where a is a D dimensional vector with entries chosen independently from a p-stable distribution and b is a real number chosen uniformly from the range$[0, w]$. For a fixed a, b the hash function $h_{a,b}$ is given by,

$$h_{a,b}(x) = \left\lfloor \frac{a \cdot x + b}{w} \right\rfloor \tag{2}$$

Generally $w = 4$. The dot product $a \cdot x$ projects each vector onto a real line. The real line is chopped into equi-width segments of appropriate size w and hash values are assigned to vectors based on which segment they project onto. The value of k is chosen such that $t_c + t_g$ is minimal, where t_c is the mean query time and t_g is the time to compute the hash functions in L hash tables. The values of k is determined by estimating the times on a sample data set $S \subseteq P$. The details about such parameter settings and the hash functions are presented in [11,16]. Algorithm 1 summarizes the major steps of content-sensitive hashing.

Given a query word image, it is represented with the set of features q. The first level k hash functions are calculated and concatenated to get bucket id's $g_i(q)$, $i = 1, \ldots, L$ in L hash tables. Then all the features, and the corresponding words, in the buckets of L tables are retrieved as the query results. Thus the problem of finding nearest neighbor boils down to searching only the vectors in the bucket that have the same hash index value as the query. Algorithm 2 summarizes the major steps of querying.

The hash based search in a collection of document images is faster as compared to other approaches, like exhaustive search with DTW and nearest neighbor techniques. Approaches presented in the literature take a long time in building the index and retrieval due to preprocessing and complex matching procedures. This computational time can be reduced with the elimination of costly processes, like clustering. We achieve this by employing faster content-sensitive hashing technique. We achieve interactive retrieval with retrieval speed in milliseconds. Time

Algorithm 2. Word Retrieval

```
Input: Query Word Image $w$
Output: Similar word images
1: $O \leftarrow \phi$
2: for each $i = 1, \ldots, l$ do
3:     Pre-process word image $w$ (noise removal etc).
4:     Extract features $F_w$ of word image $w$.
5:     Compute hash function $I = g_i(F_w)$
6:     $O \leftarrow O \cup \{$points found in index $I$ of $T_i\}$
7: end for
8: Return similar words $O$ by linear search.
```

inefficient offline processing of the data is not required for creating the index. The hashing technique avoids complex image matching methods and searches in sub-linear time.

3 Results and Discussions

We evaluated the proposed hash based retrieval scheme on word image data sets obtained from a collection of 7 Kalidasa books. The books are printed in Telugu, an Indian language. The document images were scanned and preprocessed to get segmented words with little manual effort to remove segmentation errors. Then, the words were represented by a set of features. Around 20 words were annotated in each book for experimentation and performance evaluation purpose.

Given a textual query word, an image is rendered (generated). Features are extracted from the query image and hashed to search and retrieve the relevant words. The book-wise performance measured using precision, recall and F-score values are shown in Table 1.

The query image and example search results are shown in Figure 3. The first two rows show correct results. Sometimes other words may also appear somewhat visually similar and the last column in the last two rows shows examples of such words being retrieved.

Table 1. Search Performance: Precision, recall and F-score values for retrieval experiments conducted on each book from Kalidasa collection

Book	# Pages	# Words	Precision	Recall	F-score
Maalavikaagnimitra	292	22,500	100.00	91.72	95.68
Vikramuurvashiyam	286	23, 600	100.00	95.58	97.74
Abhijnanasakuntalam	312	22,500	96.79	91.27	93.96
Ritusamhara	142	11,000	94.65	93.67	94.16
Kumarasambhava	282	56,100	92.37	90.21	91.27
Raghuvamsha	300	36,000	93.23	92.6	92.91
Meghaduta	238	44,000	96.15	93.53	94.82

Query Image	Some of the Retrieved Images			
భగవతి!	భగవతీ.	భగవతీ!	భగవతి	భగవతి!
నిపుణికా	నిపుణికా	నిపుణికా	నిపుణికా	నిపుణిక
ద్వితీయాంకము	ద్వితీయాంకము	ద్వితీయాంకము	ద్వితీయాంకము	తృతీయాంకము
శరత్	శరత్	శరత్	శరత్	శరది

Fig. 3. Results: Example (Telugu) words searched for input queries

Query Image	Some of the Retrieved Images			
ఋతుసంహారమ్	ఋతుసంహారం	ఋతుసంహారము	ఋతుసంహారమ్	ఋతుసంహారము
విదూషకుడు	విదూషకుడు	విదూషకుడు	విదూషకుడు	విదూషకుడు
అంకము	అంకము	అంకము	అంకము	అంకము

Fig. 4. Results: Words with small variations in style and size are retrieved

Examples of queries containing words of different sizes and style types are shown in Figure 4. Such results are obtained by querying the same word in multiple books of the collection. Using the same query across two different books of the collection retrieves words which are content-wise similar.

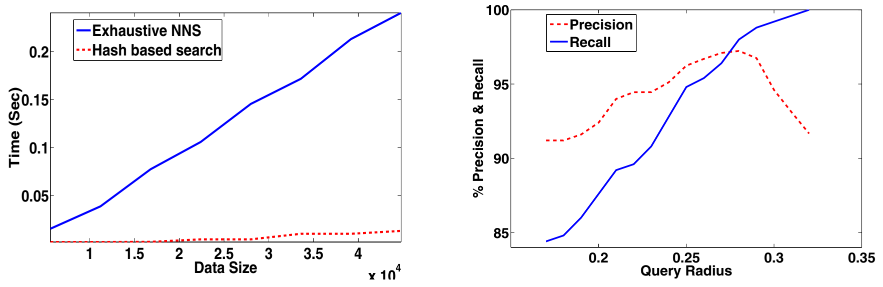
Indian language words have small form variations. For example, the same word may have different case endings. Such words are also searched correctly using the proposed solution. Example results of such queries are shown in Figure 5 (row 2). The retrieved words have the same stem, which is due to the similarity in image content. There are limits to the font variations that can be handled by the proposed retrieval technique. Experiments show that we cannot use combinations of different font words but such combinations are very unlikely to occur in books.

The proposed hashed based search is sub-linear and much faster than exhaustive nearest neighbor search. The plot in Figure 6(a) shows the time efficiency of our hash based search versus nearest neighbor search. The experiments were conducted on data sets of increasing size (by 5,000 words) in each iteration. The maximum number of words used were around 45,000. With the use of the maximum size data set, the maximum time to search relevant words was of the order of milliseconds. The experiments were conducted on an AMD Athlon 64 bit processor using 512 MB memory.

The precision and recall values are controlled by the query radius (distance) value. Experiments were conducted on synthetic images of Telugu language to see

Query Image	Some of the Retrieved Images			
ప్రియురాలా!	ప్రియురాలా!	ప్రియురాలా!	ప్రియురాలికి	ప్రియురాలిని
చంద్రుని	చంద్రుని	(చంద్రుని)	చంద్రుడే	చంద్రుల
శ్లోకములు	శ్లోకములు	శ్లోకములు	శ్లోకము	శోకమును

Fig. 5. Results: Words with small form variations are retrieved as relevant



(a) (b)

Fig. 6. Performance comparison: (a) Hashing and exhaustive nearest neighbor search. (b) Effect of distance: Precision and recall change with the query radius.

the effect of radius on the performance. Around 6,000 synthetic word images of Telugu language were used for the experiments. Each word was repeated around 4-10 times in the whole collection. Figure 6(b) shows the change in precision and recall values with the radius. The degradation in performance with the increase in radius indicates that many irrelevant words are added to group of similar words. Similar results were obtained with different font datasets for Telugu language. Therefore query distance has to be determined experimentally.

Table 2 compares this approach to one based on using the DTW score as a similarity measure. It shows that our method is much faster than the DTW based exhaustive matching and search procedure while the accuracy is similar.

Table 2. Performance: DTW based exhaustive search is much slower. Accuracy of the proposed method similar to the DTW matching.

Book	Hash Based Search			DTW Based NNS		
	Precision	Recall	Time(sec)	Precision	Recall	Time(sec)
Abhijnanasakuntalam	96.79	91.27	0.005	95.27	93.71	650
Ritusamhara	94.65	93.67	0.003	93.33	96.63	216

4 Conclusion and Future Work

We presented an efficient indexing and retrieval scheme for searching in large document image databases. Efficiency and scalability along with high precision and recall values are achieved by content-sensitive hashing. The retrieval speed is orders of magnitude better - the technique can search 20,000 word images in milliseconds. We have demonstrated that this technique is practical for searching printed documents rapidly. Future improvements could include feature selection using machine learning techniques to include multiple fonts and styles.

References

1. Pal, U., Chaudhuri, B.: Indian script character recognition: A survey. Pattern Recognition 37, 1887–1899 (2004)
2. Rath, T.M., Manmatha, R.: Word image matching using dynamic time warping. In: Conference on Computer Vision and Pattern Recognition, vol. (2), pp. 521–527 (2003)
3. Rath, T.M., Manmatha, R.: Word spotting for historical documents. IJDAR 9(2), 139–152 (2007)
4. Balasubramanian, A., Meshesha, M., Jawahar, C.V.: Retrieval from document image collections. In: Bunke, H., Spitz, A.L. (eds.) DAS 2006. LNCS, vol. 3872, pp. 1–12. Springer, Heidelberg (2006)
5. Chan, J., Ziftci, C., Forsyth, D.A.: Searching off-line arabic documents. In: CVPR. Conference on Computer Vision and Pattern Recognition, vol. (2), pp. 1455–1462 (2006)
6. Lu, Z., Schwartz, R., Natarajan, P., Bazzi, I., Makhoul, J.: Advances in the bbn byblos ocr system. In: ICDAR, pp. 337–340 (1999)
7. Rath, T.M., Manmatha, R., Lavrenko, V.: A search engine for historical manuscript images. In: SIGIR, pp. 369–376 (2004)
8. Ataer, E., Duygulu, P.: Retrieval of ottoman documents. In: Multimedia Information Retrieval (MIR) workshop, pp. 155–162 (2006)
9. Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., Perantonis, S.J.: Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. IJDAR 9(2), 167–177 (2007)
10. Sankar, K.P., Jawahar, C.V.: Probabilistic reverse annotation for large scale image retrieval. In: Conference on Computer Vision and Pattern Recognition, pp. 1–6 (2007)
11. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: SOTC, pp. 604–613 (1998)
12. Shakhnarovich, G., Viola, P., Darrell, T.: Fast pose estimation with parameter-sensitive hashing. In: ICCV, pp. 750–757 (2003)
13. Matei, B., Shan, Y., Sawhney, H., Tan, Y., Kumar, R., Huber, D., Hebert, M.: Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation. IEEE Trans. PAMI 28(7), 1111–1126 (2006)
14. Lamdan, Y., Wolfson, H.: Geometric hashing: A general and efficient model-based recognition scheme. In: ICCV, pp. 238–249 (1988)
15. Nakai, T., Kise, K., Iwamura, M.: Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In: Bunke, H., Spitz, A.L. (eds.) DAS 2006. LNCS, vol. 3872, pp. 541–552. Springer, Heidelberg (2006)
16. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th VLDB conference, pp. 518–529 (1999)