# WORD HASHING FOR EFFICIENT SEARCH IN DOCUMENT IMAGE COLLECTIONS

Submitted in partial fulfillment of
the requirements for the degree of

Master of Science (by Research)
*in*
Computer Science

*by*

Anand Kumar
200507002
<anandkumar@research.iiit.ac.in>

Faculty Advisors
Prof. C. V. Jawahar
Prof. R. Manmatha

International Institute of Information Technology
Hyderabad, INDIA
June 2008

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
Hyderabad, India

# CERTIFICATE

It is certified that the work contained in this thesis, titled " *Word Hashing for Efficient Search In Document Image Collections* " by Anand Kumar, has been carried out under our supervision and is not submitted elsewhere for a degree.

Date

Prof. C. V. Jawahar (Advisor)
IIIT, Hyderabad, INDIA

Date

Prof. R. Manmatha (Advisor)
University of Massachusetts, Amherst, USA

# Acknowledgements

I would like to thank my adviser Dr. C. V. Jawahar for his guidance and support during my research. He provided me with the opportunities to pursue my research interests and helped in developing a thinking processes required for the research. I would also like to thank my co-adviser Dr. R. Manmatha for his guidance and help in different stages of my research work. His advise and constant feedback helped me in achieving this milestone.

Dr. Anoop M. Namboodiri had been providing his constructive suggestions in research paper writings. I was able to get my first publication during MS with his support. I am also grateful to Dr. P. J. Narayanan for the intense and stimulating research ambiance provided at Center for Visual Information Technology (CVIT).

My acknowledgment wouldn't be complete without the mention of MS friends especially Yokesh, Pawan Harish, Jyotirmoy, Ananda Swarup Das, Jagdeesh, Praveen Dasigi, Sanjeev, Bhavani Shankar and Akash Agrawal. I would like to thank my CVIT friends for providing joyous company.

# Abstract

A large numbers of 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. 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 for large collection of books. Direct matching of images is inefficient due to the complexity of matching and thus impractical for large databases. In general, indexing and retrieval methods for document images cluster similar words and build indexes with the representatives of the clusters. The time required for building such a clustering based index is very high. Such indexing methods are time inefficient with the use of complex image matching procedures required in the clustering step. This problem is solved by directly hashing word image representations.

An efficient mechanism for indexing and retrieval in large document image collections is presented in this thesis. First, document images are segmented to get words. Then features are computed at word level and indexed. Word retrieval is done very efficiently with *content-sensitive hashing* (CSH), which uses an approximate nearest neighbor search technique called locality sensitive hashing (LSH). The word images are hashed into hash tables using 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 CSH scheme makes the search very fast without degrading accuracy. Experiments on a collection of Kalidasa's - the classical Indian poet of antiquity - books in Telugu demonstrate that the word images may be searched in a few milliseconds. The approach thus makes searching document image collections practical. The search time is reduced significantly by hashing the words.

The proposed method of efficient access to document images at word level allows us to group words with similar content. Thus, clusters of similar words can be generated using the proposed retrieval method. These clusters can be labeled with text to represent the images. The text representative is obtained from the text output by a recognizer for all the words of the cluster. With this method the whole collection of word images can be annotated. The text can be used for searching in the document images using text search methods. This method is called "*annotation based indexing*". An annotated corpus is critical to the development of robust OCRs. Manual annotation of document images is a labor and time intensive task, especially for large collections. Most of the time is consumed in the annotation of the textual content of document images. With the proposed annotation procedure, text content can be annotated automatically and efficiently.

In this thesis, an efficient indexing and retrieval scheme for searching in large document image databases is presented. The main component is the efficient, sub-linear time, search method in large collection of word images. The search technique applied is similar for word clustering and for the annotation of the clusters. Along with the previous work related to the proposed methods, the features employed are also discussed in the thesis. The effect of each feature and their combinations on the performance of the proposed search method is analyzed. Experimental results are presented to show that the proposed indexing and retrieval scheme is time efficient and makes searching large document image collections practical.

# Contents


**1 Introduction** **1**

1.1 Document image collections and access . . . . . 3

1.1.1 General framework . . . . . 3

1.1.2 Challenges in document representation . . . . . 4

1.2 Techniques for document image retrieval . . . . . 6

1.2.1 Searching by image matching . . . . . 6

1.2.2 Recognition for retrieval . . . . . 6

1.2.3 Annotation and retrieval technique . . . . . 7

1.3 Challenges of searching in document collections . . . . . 8

1.3.1 Matching and Retrieval . . . . . 8

1.3.2 Time efficient search in document collection . . . . . 8

1.3.3 Auto annotation with erroneous OCR . . . . . 9

1.4 Overview of this work . . . . . 10

1.4.1 Problem statement . . . . . 10

1.4.2 Contributions of this work . . . . . 11

1.4.3 Organization of thesis . . . . . 12

**2 Related Work** **13**

2.1 Searching in document image collections . . . . . 13

2.1.1 Recognition based search . . . . . 14

2.1.2 Recognition free search . . . . . 15

2.1.3 Other search techniques in document images . . . . . 17

2.2 Similarity search in large databases . . . . . 20

2.3 Nearest neighbor search . . . . . 20

2.3.1 $k$-d trees . . . 21
2.3.2 $R$-trees . . . 22
2.3.3 Other structures . . . 22
2.3.4 Discussion . . . 23
2.4 Approximate nearest neighbor (ANN) search . . . 24
2.4.1 BBD-trees . . . 25
2.4.2 Locality sensitive hashing . . . 26
2.5 Applications of similarity search . . . 26
2.5.1 Search in multimedia databases . . . 26
2.5.2 Pattern recognition and classification . . . 27
2.5.3 Document retrieval . . . 28
2.6 Summary and Comments . . . 29

**3 Approximate Nearest Neighbor Search in Document Databases 31**
3.1 ANN search for CBIR . . . 31
3.2 Locality sensitive hashing (LSH) and application . . . 33
3.2.1 A simple example of LSH . . . 35
3.3 Indexing document image databases using LSH . . . 36
3.3.1 Representation of word images . . . 36
3.3.2 The basic steps . . . 40
3.4 Experiments and results . . . 41
3.4.1 Data set generation . . . 41
3.4.2 Data sets . . . 42
3.4.3 Results and discussion . . . 43
3.5 Summary and comments . . . 47

**4 Search in Collection of Kalidasa Books 49**
4.1 Document search and retrieval systems . . . 49
4.1.1 Search engine for document images . . . 50
4.1.2 Indian language documents . . . 50
4.1.3 Problem statement and contribution . . . 52
4.2 Efficient search in large collections . . . 53
4.2.1 Content sensitive hashing . . . 53
4.3 Hashing parameters and analysis . . . 57

4.4 Cross-lingual search . . . 58
4.5 Experiments and results . . . 60
4.5.1 Search performance in Kalidasa collection . . . 61
4.5.2 Time efficiency of hashing . . . 64
4.5.3 Effect of query distance . . . 65
4.5.4 Comparison with DTW based search . . . 65
4.6 Summary and Comments . . . 66

**5 Word Annotation for Retrieval 68**
5.1 Annotation of document images . . . 68
5.2 Auto annotation: State of the art . . . 69
5.2.1 Traditional methods . . . 69
5.2.2 Reverse annotation . . . 70
5.3 Applications of annotation . . . 71
5.3.1 Annotation for archival . . . 71
5.3.2 Annotation for retrieval . . . 72
5.3.3 Annotation for recognition . . . 72
5.4 Overview of annotation based retrieval . . . 73
5.5 Word image clustering . . . 74
5.6 Annotation of word clusters . . . 77
5.6.1 Word error correction . . . 78
5.7 Application to retrieval . . . 81
5.8 Experiments and Results . . . 83
5.8.1 Word annotation results . . . 84
5.8.2 Search results . . . 85
5.9 Future Work . . . 86

**6 Conclusions 88**
6.1 Summary and conclusion . . . 88
6.2 Future work . . . 89

**Bibliography 90**

**A Related Publications 101**

# List of Tables


2.1 Summary of the work done in the area of document image retrieval. The work is done in both handwritten and printed document images. . . . . 19

3.1 Search performance on generated data sets of Telugu and English language word images. . . . . 44
3.2 Font specific performance of retrieval technique on a collection of word images 45
3.3 Feature specific performance of retrieval technique . . . . . 47
3.4 Retrieval performance with combination of different features . . . . . 47

4.1 Search Performance: Precision, recall and F-score values for retrieval experiments conducted on each book from Kalidasa collection. . . . . 62
4.2 Performance: DTW based exhaustive search is much slower. Accuracy of the proposed method similar to the DTW matching. . . . . 66

5.1 Comparing accuracies of different methods on English language data sets. WAc = Word Accuracy, CAc = Character Accuracy . . . . . 84

# List of Figures


1.1 Searching in a database of word image collection: A conceptual diagram . . 1
1.2 A simplified overview of document image indexing and retrieval technique . 4
1.3 Some artifacts (or degradations) appearing in scanned document images. (a) Noisy image. (b) Blobs and joining components. (c) Cuts/Breaks in components. (d) Processed clean image. . . . . 6

2.1 Building $k$-d tree for nearest neighbor search . . . . . 21
2.2 Illustration of ANN search using BBD tree . . . . . 25

3.1 Representing word images using features for indexing and retrieval . . . . . 38
3.2 Data sets (English): Sample word images from the English data sets. EG_Data_1 contains word image in only one of these fonts. EG_Data_2 contains word images in "Arial" and "Times" fonts (3.2(a) and 3.2(b)). EG_Data_3 contains images in all three fonts. . . . . 42
3.3 Results: Example words retrieved from querying in English data sets. . . . 44
3.4 Search result: Some example word images obtained as a result of querying the word image with content "ineluctable" in the hash table. . . . . 45
3.5 Effect of query radius on the Precision and Recall values in retrieving the word images. . . . . 46

4.1 Sample document images from Kalidasa's books in Telugu. . . . . 51
4.2 Examples word images, selected from Kalidasa's collection of books. . . . . 52
4.3 Conceptual Diagram of the Efficient Searching Procedure from Document Image Database. Showing offline preprocessing and on-line query processing stages. . . . . 54

4.4 Cross lingual search in document images. Transliteration and dictionary based approach. A Conceptual diagram that shows document searching in multiple languages using transliteration and dictionary based approach. . . 59
4.5 Cross lingual search in document images. Building transliteration map. Sample Entries of the Transliteration Map Built for Cross-lingual Retrieval in English, Devanagari and Telugu. . . . . 59
4.6 Data sets (Telugu): Sample word images from the Telugu data set TG_Data. Images are generated in a single font. . . . . 61
4.7 Results: Example (Telugu) words searched for input queries. . . . . 62
4.8 Results: Words with small variations in style and size are retrieved. . . . . 63
4.9 Results: Words with small form variations are retrieved as relevant. . . . . 63
4.10 Results: Examples of results obtained by searching in Hindi document images. 64
4.11 Results: Examples of words obtained by cross-lingual search in the collection. 65
4.12 Performance comparison: Effect of data size (number of words) on Hashing and exhaustive nearest neighbor search. . . . . 66
4.13 Effect of distance: Precision and recall change with the query radius. . . . . 67

5.1 Annotation based search: Annotation of word images for efficient access to document image collections. . . . . 74
5.2 Frequency of words in a collection of document images. . . . . 76
5.3 Word error correction (Majority voting): An example of correcting word errors using cluster information. . . . . 79
5.4 Figure (Matrix) Illustrating the word DTW. Horizontal continuous line segments show the components in columns grouped together to match with a character in a row. Continuous line segments across rows show the place of character boundaries. . . . . 81
5.5 Word error correction (String alignment): An example of correcting word errors using cluster information. . . . . 82
5.6 Example word images obtained by clustering using hashing method. . . . . 85
5.7 Effect of cluster size on the performance of search . . . . . 86

# Chapter 1

# Introduction

The evolution of the world wide web (WWW) over the past decade has made access to relevant information easy and effective. This has been powered by the technological advancement in information retrieval. Information from all the domains of daily life, technical or non-technical, is available over the Internet with the click of a button. Often, the Internet is searched for documents containing required information. With the emergence of digital libraries, a very large number of documents are stored electronically [65]. These documents need not be stored as text always, which makes the search in them more challenging. This thesis focuses on the situation when the documents are scanned or digitized and stored as images. To be more precise, digitized content is stored as images corresponding to pages in books.

Often, digital libraries store digital images of documents. These document images are

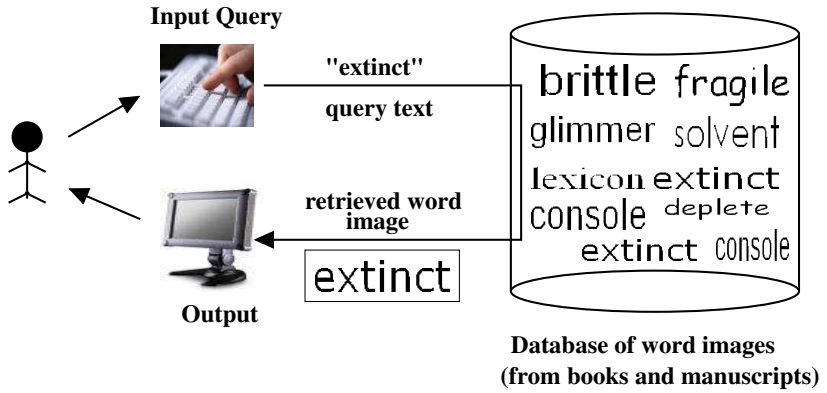



Figure 1.1: Searching in a database of word image collection: A conceptual diagram

rich in information content [9]. However, these images can not be searched like the textual documents. This makes the access to such documents challenging. The basic problem of the search in a database of document images is shown pictorially in Figure 1.1. Given a textual query, we are interested in retrieving word images (images of single word) with the queried text. We assume that indexing is done at word level and document images are segmented into words. The problem is challenging due to the following facts:

- Since the representation is non-textual, traditional indexing schemes are non-applicable.
- Even if we want to solve the problem in the image domain, the similarity computation is difficult and time consuming.
- Scalability of the methods to a real life situation is challenging.

The textual content in printed book documents is type-set in a specific font style and font size with some level of uniformity across all pages. These printed documents are scanned/digitized to obtain images. These are typically referred to as *document images*. Handwritten or printed text along with pictures may be present in these images. We restrict our attention to printed document images like those from a book. Such a collection is also typically typeset in a single font or style. A direct method to access these documents[1] is by converting them to textual form by recognizing text from images. Optical character recognizers (OCRs) are required to obtain the textual content from these documents [78, 8]. Books, journals, articles, newspapers, magazines are some examples of *Printed* documents. Popular digitizers used for creating document image include digital cameras, and hand-held and flat-bed scanners.

Documents are typically available in very large numbers. Indexing is a pre-requisite for efficient retrieval [74]. Manually grouping and filing these documents for easy access is a very tedious procedure. At the same time it is of paramount importance that these documents are made accessible to the users who would in fact like to search them with relative ease. Printed documents do not contain searchable text as is, but contain words as images which cannot be searched by existing text search engines. Conventional text search is based on matching or comparison of textual description (say in ASCII/UNICODE). These techniques can not be used to access content at the image/ink level, where text is represented as pixels but not as ASCII/UNICODE.

[1] A document in the context of this thesis is an image of the paper obtained after scanning.

## 1.1 Document image collections and access

Digital libraries [81] have received wide attention in recent years allowing access to digital information from anywhere in the world [75]. They have become widely accepted and even preferred information sources in the areas such as science and education [8]. The storage of scanned documents in electronic form has created a huge amount of data in digital libraries. The rapid growth of the Internet and the increasing interest in development of digital technologies and document collections [65, 74] has helped to accelerate the digitization of printed documents in the past few years. Archives of books and various collections of document images enable the need for easy and rapid access. They have a number of advantages in terms of storage, access, sharing etc. Indexing and retrieval are critical for success of any electronic library [65].

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 OCRs. The precision and recall of statistical text retrieval methods ("bag-of-words" [33]) is not seriously affected by the typical error rates of commercial OCR. However, some textual analysis tasks (e.g. those depending on syntactic analysis), whether modeled statistically or symbolically, can be derailed by low OCR error rates. Since OCRs are not well developed for many of the non-European languages [84], we need to explore search algorithms in the image domain.

### 1.1.1 General framework

Document image indexing schemes represent the information in a meaningful way. These also provide a mechanism to efficiently retrieve, in rank order, most relevant documents for a given query. In text retrieval systems, documents are indexed as words. The most common method of characterizing the content of text document during indexing is to perform document processing for the extraction of representative terms [2]. This is achieved through applying a series of document processing techniques that identify words, extract the relationship between words and measure their relevance. The extracted index terms are grouped together based on their similarity measurement and are indexed using a selected indexing structure [7]. Once the index is built, given a query, the relevant documents are located from the index.

The process of building an index for document images is different from that of text documents. Text processing is replaced by a series of document image processing procedures.

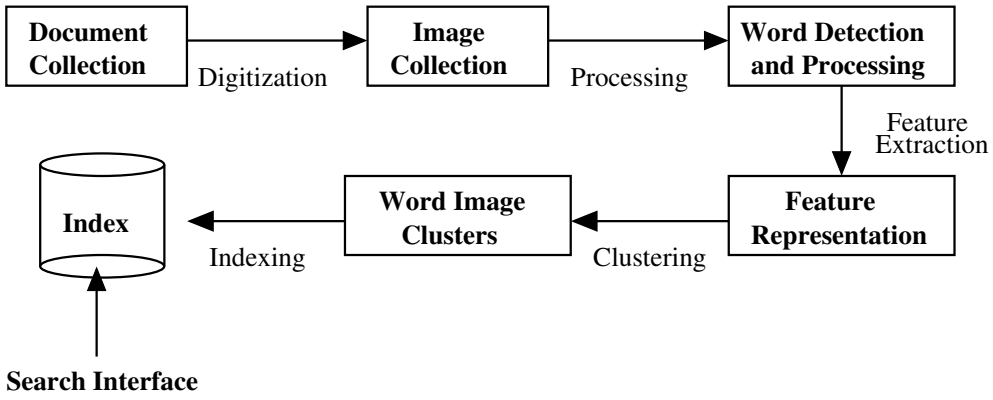



Figure 1.2: A simplified overview of document image indexing and retrieval technique

Extracting index terms is different from that used in the text domain. The relationship between words is measured by complex image matching techniques. In general, the framework of document image indexing and retrieval appears as shown in Figure 1.2 [51, 105]. The system accepts a collection of document images as input. They are preprocessed to generate a content-bearing list of index terms (word images in our case). These terms are then clustered based on their similarity measures. A representative word, called keyword, is found and assigned to each cluster. A set of clusters are finally organized in an indexing data structure using cluster keywords. Keywords could also be annotated to ease searching from large collection of document images. The indexing process finally constructs index lists that are used during the search process.

The indexing scheme first segments the pages into words and then matches the actual word images against each other to create clusters [75]. Each cluster consists of multiple instances of the same word. For each cluster, the number of words in it are counted. Those clusters with the highest number of elements are stop-words and, hence they are eliminated from further consideration. A list is then made of the remaining clusters and ordered according to the number of words contained in each of the classes. The user then assigns for each member of cluster its ASCII interpretation [10, 53]. The words in these clusters are then indexed and ready for searching and retrieval.

### 1.1.2 Challenges in document representation

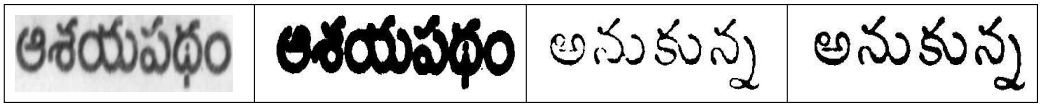
Words are the meaningful entities that are searched in document collections. All words in a page represent the complete document. Words are easily available in electronic text

documents, which are relatively difficult to obtain in document images. The document image has to be processed in a number of stages to obtain the word images. Some of the challenges faced in obtaining the words from document images are mentioned below.

- **Preprocessing:** The document images obtained after scanning are preprocessed to obtain words. The document has to go through different processing stages before words are extracted. Any noise or degradation induced as a result of the scanning environment has to be eliminated. It may involve noise removal, rotation (or skew correction) and filtering operations. When the alignment of the paper with the edge of the scanning surface changes, there are chances of getting skew in the image. In such situations, the image has to be processed for deskewing. The clean document is now split into words using document segmentation techniques [61]. The complexity of the segmentation depends on the layout complexity of the document images. However, a segmentation algorithm can be designed for specific type of document collections. In this work we do not focus on the issues of document image segmentation.

- **Representation:** We observe that the words in document images are represented by a set of features. Usually, the features are vectors of numeric values. The features used for representing the objects in different applications may differ for similar objects. The change in the feature values would be due to a change in the appearance of the objects or the viewing angle. Even in document images, the appearance of words may change due to a number of factors that arise at the time of digitization. The print style of the documents and the type of font also affect the representation. The representation of similar words printed in different fonts vary considerably. The other factors that affect the appearance and representation of words are (see Figure 1.3):

  - Excessive dusty noise on the paper.
  - Large ink-blobs joining disjoint characters or components of characters.
  - Vertical cuts due to folding of the paper.
  - Cuts in arbitrary direction due to paper quality or foreign material.
  - Degradation of printed text due to the poor quality of paper and ink.
  - Floating ink from facing pages.



(a) (b) (c) (d)

Figure 1.3: Some artifacts (or degradations) appearing in scanned document images. (a) Noisy image. (b) Blobs and joining components. (c) Cuts/Breaks in components. (d) Processed clean image.

## 1.2 Techniques for document image retrieval

The document images can be searched in text or image domain using different approaches. These approaches are essentially classified into three classes, as enumerated below. We explore them in detail in Chapter 2.

### 1.2.1 Searching by image matching

The search in a collection of words involves looking for similar words [58, 98]. The simple solution is to match the query word image with all the words in the collection and report the similar ones. Direct matching of image pixels may fail even in presence of small changes in sizes of images [89]. Moreover, there will be a number of variations occurring in words due to artifacts, digitization process, size changes etc [10, 91].

The task of looking for similar words in a collection is challenging. However, many methods have overcome this problem with the invention of new techniques. The word images are represented by a set of features [89], usually vectors or sequences of numeric values. These representations are used for word image matching. Thus, similar words can be searched using the matching the technique. A similarity measure that looks for approximate matches, to some extent, may be one of the methods for image matching. This method of searching similar words by image matching technique is expensive. As a result of matching query with every word of the collection the search may take a long time to generate the results. Hence, exhaustive search techniques are not suitable for search in word image collections.

### 1.2.2 Recognition for retrieval

This method is based on using a recognizer to convert an image to text and then searching the results using a text search engine. Text word similarity search is pretty straight forward

in these methods. It involves looking for some simple measures, like edit distance, and deciding whether to produce that word in the result. In the text domain grouping or clustering of text words to build index structures is simple. This simplicity tempts us to convert the document images into text by using recognizers. An example of recognition based techniques is the gHMM approach of Chan *et al.* [21], suggested for printed and handwritten Arabic documents.

The performance of recognition based search techniques depends on the accuracy of the recognizer. Words that do not exist in the language are generated by the results of erroneous recognizers. This makes the search engines consider these words as special. As a result, the index size is increased by words that do not exist in the language and which are never searched. Search in document images converted into text is challenging, as there are no robust OCRs available [84] for most Indian languages.

### 1.2.3 Annotation and retrieval technique

Groups of similar words can be generated by efficiently indexing the document images at word level. Thus, clusters of similar words can be generated. To enable search over a large collection of document images, the groups or clusters of words generated by the indexing process have to be labeled with representative content. The representatives can then be recognized using OCR and the whole cluster can be annotated using the recognized text. With this method the whole collection of word images can be annotated. These annotations are used for text based indexing of images. This method of indexing and retrieval is called *annotation based retrieval* [94]. A similar approach, proposed by Rath and Manmatha [90, 91] for handwriting, 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. The annotations are used to search the documents.

A text index serves as an interface to the document image collections for faster access. Annotation becomes very important if there are no retrieval methods available. The annotation is used for indexing, which can be used for answering user queries. After the annotation process, text based retrieval methods can be applied for the search. The annotation based search is expensive due the cost of labor and time required for manual annotation. Most of the time is consumed in the annotation of the textual content of document images.

## 1.3 Challenges of searching in document collections

Response time is an important parameter of an information retrieval system. The information to be accessed should be made available as quickly as possible to the user. A search technique taking an enormous amount of time may not be useful for real time applications. Search time increases primarily because of the fact that image comparison is computationally complex. As the size of the database to be searched increases, the response time of systems built on such techniques also increases. In such cases alternative methods for search have to be explored. The challenges involved in the indexing process are discussed below.

### 1.3.1 Matching and Retrieval

Searching for similar words involves matching the query word image with all the images of the database. The matching is done at the feature level. An effective representation will have to take care of the artifacts [11]. The search for similar word image is equivalent to the task of searching for similar or closest points in high-dimensional feature space. Thus the problem of word similarity search can be transformed into a nearest neighbor search [85] problem.

The similarity search is popular in a number of problem related to computer vision. Image search, video indexing and retrieval, mining and searching in large multimedia databases are some of the examples. Based on the applications and the objects, their representation and matching techniques change. However, the values of a feature would still be similar for two similar objects (words) chosen from different sources. Hence, the similarity search should be capable of capturing such objects (words) in the results of search. Similarity search can be done efficiently using an approximate nearest neighbor search [48, 62, 50] technique like locality sensitive hashing.

### 1.3.2 Time efficient search in document collection

The problem of searching in the image domain can be avoided if annotations are available to the entire data set. An annotation based system would work well if an index is built offline by first clustering and there after annotating the clusters. The access to an index is much faster than search through images by matching. However, such systems are not scalable to large collections of document images. The techniques available to search in document images are limited by lack of efficiency.

In general, databases are searched linearly comparing every item with the query. This is a very costly procedure if the database size is large. This is worse if the data is of very high dimension and the similarity measure is complex. For example, assume that matching two word images takes 1ms time (good techniques may take even more) and the database consists of 100,000 words (from a few books). Searching for similar words through all these images takes 100 seconds. The user has to wait for more than a minute to search in a small collection. A text based search methods can search in millions of words in less than 1ms.

Obviously the problem of searching in high-dimensional feature points of word representation can be solved through simple brute force search. The traditional data structures, built for such approaches, will not perform well for dimensions more than 3 [5]. A number of methods have been proposed which provide relatively modest constant factor improvements (e.g., through partial distance computation [22], or by projecting points onto a single line [41]). However, there is a need for approximate nearest neighbor search techniques that can work fast enough and return results to the user quickly. In any fixed dimension greater than 2, many known methods do not achieve the simultaneous goals of roughly linear space and logarithmic query time.

### 1.3.3 Auto annotation with erroneous OCR

Retrieval of document images can be efficient if annotations are available for words. Text retrieval techniques can be applied on the annotations for efficient access. However, generation of annotations is also a challenging task. Manual annotation is difficult and time consuming process. Another method is to annotate the words automatically by recognition. As mentioned earlier, the recognizers are not accurate and robust enough to obtain acceptable annotation accuracy.

The accuracy of annotation is lowered due to errors in the recognition of characters in word images. Since we are searching similar words it is possible to cluster them. The recognized text of all the words of the cluster can be used to identify the correct representative word for the cluster. Some of these recognized words of a cluster may be erroneous. But the information from the correctly recognized words can be used to find the representative word for the cluster. The method of obtaining the correct word from a cluster of recognized words is effective to generate annotations. Thus, auto annotation can be used for enabling annotation based search over the document image collections.

## 1.4 Overview of this work

The thesis mainly aims at addressing the problems of effective and efficient retrieval in large document image collections with effective representation of the word images. The approaches proposed in the literature are not scalable to large collections of documents. In the following sections we discuss the problem, contributions of this work and organization of the thesis.

### 1.4.1 Problem statement

The document image indexing and retrieval techniques available are not feasible for collections of document images. As the number of words present in the collection increases, the search time also increases. Most of the existing search techniques based on complex image matching methods are not scalable to even a moderate collection of document images. The major hurdles in achieving efficient search are the processing time and the word image matching techniques [88]. A number of scans made on the words during clustering for building the index contribute to the large amount of time required [10]. The search based on word image matching also contributes to the retrieval time. The word matching methods are expensive and become more expensive when words are represented by high dimensional feature vectors [89]. Hence, the problem is to search efficiently in a collection of word images represented using high-dimensional feature vectors.

A number of possible solution are present in the literature such as nearest neighbor search can be applied to this situation. But the goal is to retrieve the maximum number of relevant words from the collection. There is a need for high precision and recall along with the search speed. The representations for similar words may vary due to variations in image sizes, degradations like noise etc. The search technique has to take care of such variations also to achieve good performance. Therefore, there is a need to explore the characteristics of the representations and choose the right ones. We choose a group of well defining representations and examine their expressive power through a number experimental results.

Text search techniques can be applied if annotated clusters of word images are available. The problems of old search techniques and the unavailability of robust recognizers limit us from accomplishing this task. However, the annotation accuracy can be improved by making use of the words present in every cluster. Thus, annotation based retrieval technique enables

faster search in the collections of document images.

### 1.4.2 Contributions of this work

Retrieval is straight forward when the image collection is annotated. However, annotation itself is time consuming process. In this thesis, an efficient indexing and retrieval scheme for searching in large document image databases is presented. The major contributions are listed below.

1. The data is preprocessed fast, in linear time, to build the index structure. Efficient, sub-linear time search in large collection of document images is made possible by the content sensitive hashing method presented.

2. The effect of various types of data and word representation on the retrieval performance is analyzed by conducting many experiments on different data sets.

3. The scalability of the proposed search method is demonstrated on a set of Indian language document images and Kalidasa's books.

4. The word image clusters obtained by preprocessing are used for annotation and recognition. The accuracy of annotation can be improved by using these clusters.

5. Annotation of clusters of word images makes retrieval faster. The proposed method is applied for annotation based indexing and retrieval of document image collections.

In this work we focus on efficient retrieval of documents from a collection. Previous methods of generating an index based on text retrieval methods are time expensive. We propose a method for efficient data processing to build the index. This method helps in reducing the retrieval time and hence efficient retrieval. The scalability of the technique is demonstrated on a collection of 7 books of Kalidasa. We conducted a number experiments to find out the performance of the method using different combination of features. Expensive clustering methods are avoided to annotate the group of similar words. The clustering speed is improved significantly. Recognizers can make use of such clusters to find exact words representing each group. An experimental study of recognizer improvement is also presented in this work.

### 1.4.3 Organization of thesis

A brief survey of the search and retrieval techniques available for document image collections is presented in Chapter 2. The different recognition based and recognition free retrieval methods are discussed and the state-of-the art is presented. The drawbacks of the methods and alternatives to be explored are also discussed briefly. The different indexing structures, for near and approximate neighbor search, that can be employed in the search are also reviewed. The computational complexities of these techniques and feasibility to the document image collections are discussed. Some applications of these nearest neighbor techniques in different domains are discussed to explore the possibility of usage in document images.

The approximate nearest neighbor searching (ANNS) technique based word indexing and search is explored in detail in Chapter 3. The features used for representing word images are discussed in detail. The procedure of hashing, based on locality sensitive hashing, for facilitating efficient search are discussed in detail. The chapter also discusses about the application of ANNS in content based image retrieval (CBIR). The performance of the hashing method for word image collections is demonstrated with illustrations of results obtained from different experiments conducted. The technique to achieve scalability on large databases of document images is presented in Chapter 4. The algorithmic steps of the technique for indexing and retrieval are presented in detail. The scalability and efficiency of the methods are proved with a number of experiments conducted on a collection of Kalidasa's books.

The annotation based document retrieval technique is presented in Chapter 5. An idea for using the clusters for improving accuracy of annotation is discussed in this chapter. Finally, Chapter 6 presents the summary of contributions of this work, possible directions of the work in future and the final conclusions.

# Chapter 2

# Related Work

In this chapter, we look at the literature of indexing and retrieval techniques used for search in large image databases. The topic of interest overlaps with databases, pattern recognition, content based image retrieval, digital libraries, document image processing and information retrieval. A subset of the various methods are reviewed here. We first explore the methods adopted for search in document image databases from a computer vision perspective. We discuss some of the popularly used techniques, available in the literature, in text and image retrieval. Most of the image retrieval problems are modeled as similarity search. We look at the similarity search techniques and their applications to computer vision problems in this chapter. A brief review of the nearest neighbor and approximate nearest neighbor techniques is also provided.

## 2.1 Searching in document image collections

A number of approaches have been proposed in recent years for efficient search and retrieval of document images. There are essentially two classes of techniques to search document image collections. The first approach is to convert the images into text and then apply a search engine. Character recognizers are used for obtaining the text from images [78, 82]. The second technique is to search directly in the images bypassing the recognition task. In the following subsections we will briefly discuss about each of these techniques and the state of the art.

### 2.1.1 Recognition based search

In recognition based search and retrieval techniques, the document images are passed through an optical character recognizer (OCR) to obtain text documents. The text documents are then processed by a text search engine to build the index. The text index makes the document retrieval efficient. An example is the gHMM approach of Chan *et al.* [21], suggested for printed and handwritten Arabic documents. It uses gHMMs with a bi-gram letter transition model, and kernel principal component analysis (KPCA) / linear discriminant analysis (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 [70] 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 performance of recognition based search depends on the accuracy of the recognizer [33, 76]. In practice, specifically for Indian languages, no recognizer is perfect. When the recognized text is used for indexing, incorrectly recognized words are also indexed. Since the recognizer makes similar mistakes for a particular word, the word's frequency would be high and it appears in the index. In practice, a user gives the correct word as a query. When the index is searched, the actual word is never retrieved.

One solution to the problem of search in the presence of errors in text output by recognizers is to correct the errors by spell checking using a dictionary. The spell checkers are mainly built to correct errors that are caused by either the conventional keyboard arrangement (i.e. "a" may be mistyped as "s") or to correct common spelling errors caused by doubling, undoubling, or transposition of characters. Taghva *et al.* [102] built a search engine for documents obtained after recognition of images. Searching is done based on the results of similarity calculation between the query words and the database words. The indexed terms are divided into two groups of correctly recognized and incorrectly recognized words based on frequency calculations using a dictionary. Similar words are identified from the correct terms by applying mutual information measure. There have been attempts to retrieve complete documents (rather than searching words) by considering the information from word neighborhood (like n-grams [46]) to improve the search in presence of OCR errors [56].

To make the documents ready for indexing, a number of steps have to be undertaken.

Most of these processes take a huge amount of time. This becomes more significant when a large number of documents, say books, are to be indexed. After the text is obtained, again it has to be checked for errors. A major assumption behind these methods is the availability of robust recognizers with reasonably high recognition rates. The number of errors made by recognizer depends on the quality of the document and characteristics of the recognizer. These affect the process of indexing documents.

In summary, success of the recognition based search and retrieval techniques depends on the performance of the OCRs. Recognizers with good performance are not available for Indian languages [100, 84] or in general for many non-European languages. Accuracy of the available recognizers also falls below acceptable levels on many practical documents.

### 2.1.2 Recognition free search

An alternative to recognition based search method is to search directly in the images. A method used by Rath *et al.* [92] for document search 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 based retrieval approach is then used to search the documents. The technique was successfully used to build a 1000 page demonstration system for George Washington's handwritten manuscripts.

#### Word spotting

Word spotting is a method of searching and locating words in document images by treating a collection of documents as a collection of word images. The words are clustered and the clusters are annotated for enabling indexing and searching over the documents. It involves segmentation of each document into its corresponding lines and then into words. Each document is indexed by the visual image features of the words present in it. Word level matching has been attempted for printed [23] as well as online [51] and offline [89] handwriting documents. They are useful for locating similar occurrences of the query word. There have been successful attempts on locating a specific word in a handwritten text by matching image features for historical documents [89]. The word spotting approach [89, 90] has been extended to searching queried words from printed document images of newspapers and books. Dynamic time warping (DTW) based word-spotting algorithm for indexing and retrieval of online documents is also reported in [51]. Features for matching words are

computed from the constituent strokes. The DTW, a dynamic programming technique used to align sequences, is applied for matching the features of the word images.

In the word spotting approach proposed by Rath and Manmatha [90, 91], word images are matched with each other and then clustered . Each cluster is then annotated by a person. Alternatively, Jawahar *et al.* [53, 10] 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 [90] 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 also employed by Jawahar *et al.* [53, 10] 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 [6] tried word spotting for handwritten Ottoman documents where they use successive pruning stages to eliminate irrelevant words. Gatos *et al.* [58] 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. The traditional OCRs do not take advantage of such facts. In addition, techniques that work at the symbol level of word images, like [21], 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.

Word spotting in different scripts like Devanagari, Arabic and Latin using the shape features has also been demonstrated [98]. Global word shape features are used for finding word similarities during search. The word image features are, gradient, structural and concavity (GSC) features which measure the image characteristics at local, intermediate and large scales and hence approximate a heterogeneous multi resolution paradigm to feature extraction. The features are extracted by dividing each image into 4 x 8 rectangles and contain 384 bits of gradient feature, 384 bits of structural features and 256 bits of concavity features, giving a binary feature vector of length 1024. The distance between a word to be spotted and all the other words in the documents being matched against is computed using a normalized correlation similarity measure.

As discussed, the performance of some image matching techniques is acceptable in finding similar words. These methods use paradigms like dynamic programing, correlation etc. to achieve effective word matching. Computational complexity is the major problem when the data set to be processed increases. The performance degrades drastically with the data set size. Use of these measures for clustering of the data set adds to the time overhead involved in processing the data. Therefore, the retrieval time is high and the approaches may not be feasible for rapid indexing of a large amount of data.

### 2.1.3 Other search techniques in document images

Indexing and retrieval of words in old documents is discussed in [72]. This approach is based on two main principles: unsupervised prototype clustering and string encoding for efficient string matching. Character Objects (CO) are extracted from the word images by locating connected components and grouping together the overlapping components. Each CO is described with a code or feature vector representation. Character coding is based on grouping of similar characters with a Self Organizing Map (SOM) based clustering approach. Before encoding and indexing of the words, the SOM is trained so as to cluster the similar COs. After the COs of a word image are labeled, the word image is partitioned into fixed number of vertical slices. Each slice is assigned the label of the CO with the largest overlap with it. By using the trained SOM, the words in the whole collection are stored and represented with a fixed-length description. This description is used for comparison in order to score most similar words in response to a query. When retrieving, each word image is mapped to a fixed-size vector and compared with vectors in the database of the same length. Words with different aspect ratios are not considered for comparison

Word based indexing of Indian language document images has been attempted by S. Chaudhury *et al.* [23], where the indexing scheme exploits typical structural characteristics of Indian scripts. For the purpose of indexing, word images are represented in the form of geometric feature graphs (CFG). CFGs indicate basic connectivity relation between the shape primitives which form the word. The CFG is traversed in depth first fashion to obtain an indexing string in the form of symbolic descriptors. Hence the indexing problem is reduced to string based indexing. A suffix tree was used for indexing. A pattern is partitioned into sub-patterns, which are searched for. All occurrences of the sub-patterns are later verified for a complete match. When querying, verification is done by calculating edit distance between the pattern and each of the strings selected for verification. All

the strings with edit distance within a given threshold are considered for indexing of the documents.

Many of these techniques require a lot of computationally expensive processing. Dynamic time warping (DTW) based searching is one such example. In spite of this Sankar *et al.*[94] successfully indexed 500 books in Indian languages using this approach by doing virtually all the computation offline over a large period of time. Avoiding DTW, Rath *et al.* [91] demonstrated the use of direct clustering of word image features on historical handwritten manuscripts. However, clustering is itself an expensive operation (though not as expensive as computing DTW) when there are many clusters. In fact, many of the previous approaches to document image indexing and retrieval involve large amounts of offline pre-processing, and are not easily scalable to large collections.

The previous approaches to document image indexing and retrieval involve large offline preprocessing. Application of methods, like [53, 10] for retrieval requires identification of stop words and clustering for relevance of words. Clustering takes a large amount of time. Searching relevant words in the index also takes large amount of time. Search involves matching query word with all the entries of the index table using a complex matching algorithm, whose complexity depends on the dimensionality of the data. The search time increases linearly with increase in the size of the index. In other words, the preprocessing and search time depends on the size of the data set. The techniques that work at symbol level of word images, like [21], are very sensitive to segmentation errors. The segmentation of Indian language document images at symbol level is very difficult due to complexity of the scripts. Hence, most of the techniques available in literature have scalability and efficiency problems.

Various document image retrieval techniques present in the past literature are summarized in Table 2.1. Many of these techniques make use of different representations and word matching techniques to build index. These techniques perform well on very small data set of document images. Efficiency problems arise when we try to scale these techniques for considerably large data set. The representations are generally of high dimensions and the similarity measures/techniques should be efficient for building the index quickly. The search time increases with dimensionality and complexity of the matching procedure. Therefore, our focus is on an indexing technique that can build the index in a single scan of the data set and answer queries quickly without examining all the words (or points) in the data set.

| Reference Work | Approach Followed | Application Area |
|---|---|---|
| *Printed document retrieval* | | |
| Chaudhury *et al.* [23] | Feature graph matching | Indexing to access Indian language printed document images |
| Marinai *et al.* [71] | Prototype clustering and string matching | Search in modern printed documents |
| Ataer and Duygulu [6] | Word image matching | Retrieval of historical Ottoman documents |
| Konidaris *et al.* [58] | Word image matching | Word spotting in historical documents |
| Trenkle and Vgot [105] | Word image matching | Information retrieval from document images |
| Tan *et al.* [103] | Dot product | Retrieval from document images |
| Lu and Tan [69] | String matching | Retrieval from document image databases |
| Balasubramanian *et al.* [10, 53] | Word matching | Searching in large document image collections |
| Pramod and Jawahar [94] | Reverse annotation of word images | Search over collections of document images in Indian languages |
| Anand Kumar *et al.* [60] | Content sensitive hashing of words | Searching in large collections of document images |
| *Handwritten document retrieval* | | |
| Rath *et al.* [92] | Word image matching using relevance models | Search engine for George Washington historical manuscripts |
| Srihari *et al.* [99] | Writer matching | Forensic document retrieval |
| Russel *et al.* [93] | Recognition results | Retrieval from multi-user single script collections |
| Kamel [55] | KLT based, RTree | Online document retrieval |
| Jain and Anoop [51] | Ink Matching | Search in single writer document collections |
| Rath and Manmatha [90, 91] | Image matching and Word spotting | Retrieval from historical document collection |
| Balasubramanian *et al.* [11] | Synthesis and matching | Indexing and search in multi-user document collections |

Table 2.1: Summary of the work done in the area of document image retrieval. The work is done in both handwritten and printed document images.

## 2.2 Similarity search in large databases

A large number of objects stored in databases are characterized by a collection of relevant features and may be represented as points in high-dimensional attribute space. Often, given a query object, it is required to search for similar objects in the database. The similarity search problem is defined as follows: given a collection of $n$ points, build a data structure which, given any query point, reports the data points that are closest to the query. A particularly interesting and well-studied instance is where the data points live in a $d$-dimensional space under some (e.g., Euclidean) distance function. This problem is of major importance in several areas. High-dimensional similarity search problems arise naturally when complex objects are represented by vectors of $d$ numeric features.

Similarity searching is an important problem in a variety of applications, including knowledge discovery and data mining [36] pattern recognition and classification [28, 34], machine learning [27], data compression [43], multimedia databases [39] and statistics [32]. Nearest or near-neighbor query problems arise in the context of similarity search. Image matching for searching in document image collections is one such example that often involves nearest neighbor computations and storage for efficient access.

## 2.3 Nearest neighbor search

The nearest neighbor (NN) problem involves determining the point in a dataset that is nearest to a given query point. There are several efficient algorithms available when the dimension $d$ is low (e.g., up to 10 or 20). The first such data structure, the $k$-d tree was introduced in 1975 by Jon Bentley [14] and remains one of the most popular data structures used for searching in multidimensional spaces. It is frequently used in Geographical Information Systems (GIS), where points are associated with some geographical location (e.g., cities). Many other multidimensional data structures are also known (see [45] and [95]).

A number of indexing techniques are available in the literatures for answering nearest neighbor queries. These techniques are mainly designed for search in large databases. In this section we discuss the popular indexing techniques for nearest neighbor search.

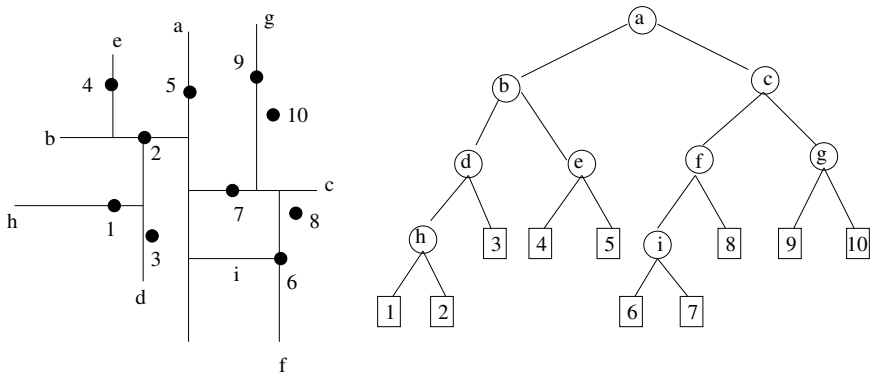



Figure 2.1: Building $k$-d tree for nearest neighbor search

### 2.3.1 $k$-d trees

The $k$-d tree is a generalization of the binary search tree used for sorting and searching. This data structure was introduced by Bentley [14]. The $k$-d tree partitions space for organizing points in a $k$ dimensional space. It is similar to a binary tree in which the root of the tree represents the entire collection of points. Each node represents both a sub-collection of the points in the space and a partitioning of that sub-collection. The collection is partitioned by splitting planes that are perpendicular to one of the coordinate system axes. Each nonterminal node has two children; these child nodes represent the two sub-collections defined by the partitioning. The terminal nodes represent mutually exclusive small subsets of the points, which collectively form a partition of $k$-space. These terminal subsets are called buckets (Figure 2.1).

Range searching with $k$-d trees is straightforward. To perform the NN calculation, the tree is searched in a depth-first fashion and at each stage it makes an approximation to the nearest distance. Starting at the root the $k$-d tree is recursively searched. When visiting a node which discriminates by the $j$-th key (called a $j$-discriminator) one compares the $j$-th range of the query with the $j$-th key of the node. If the query range is totally above (or below) the key's value then one need only search the right subtree (respectively left) of that node; the other child can be pruned from the search because any node it contains does not satisfy the query in that particular key.

The $k$-d tree has been applied to a number of database problems. The applications are discussed by Bentley in [15]. Another variant of the $k$-d trees called adaptive $k$-d trees introduced by Friedman *et al.* is presented in [40]. This structure was designed keeping the

need for effective for nearest neighbor searching. However, Sproull [97] observed that the empirically measured running time of $k$-d trees does increase quite rapidly with dimension and in practice is limited to problems with ten or fewer dimensions.

### 2.3.2 $R$-trees

An $R$-tree is a height-balanced tree similar to a $B$-tree [12, 25] with index records in its leaf nodes containing pointers to data objects. The data structure splits the space with hierarchically nested, and possibly overlapping, minimum bounding rectangles. Nodes correspond to disk pages if the index is disk-resident, and the structure is designed so that a spatial search requires visiting only a small number of nodes. Each node of an $R$-tree has a variable number of entries. Each entry within a non-leaf node stores two pieces of data: a method of identifying a child node, and the bounding box of all entries within this child node. The index is completely dynamic; inserts and deletes can be intermixed with searches and no periodic reorganization is required.

The search algorithm descends the tree from the root in a manner similar to a B-tree. Every internal node contains a set of rectangles and pointers to the corresponding child node. Every leaf node contains the rectangles of spatial objects. For every rectangle in a node, it has to be decided if it overlaps the search rectangle or not. If yes, the corresponding child node has to be searched. Searching is done in a recursive manner until all overlapping nodes have been traversed.

However, more than one subtree under a node visited may need to be searched, hence it is not possible to guarantee good worst-case performance. Nevertheless with most kinds of data the update algorithms will maintain the tree in a form that allows the search algorithm to eliminate irrelevant regions of the indexed space, and examine only data near the search area.

### 2.3.3 Other structures

In this section we briefly mention several structures that we feel are no longer competitive with the best ones. We include them for completeness.

Finkel and Bentley [38] describe a structure called the quad tree. It is a generalization of a binary tree in which every node has $2^k$ children. Bentley and Stanat [16] analyzed the performance of quad trees for "square" range searches in uniform planar point sets. The

quad trees (called "search-sort $k$ trees") have advantages over binary trees when used in a synchronized multiprocessor system. This application aside, however, the quad tree seems to be dominated by its historical successor, the $k$-d tree.

Bentley and Shamos [13] describe a data structure (the ECDF tree) for finding the empirical cumulative distribution of a point (in k-dimensional space) among a collection of points. If only a count of the number of points in the query hyper-rectangle is required and, not a listing of the points, then several ECDF searches can be used to obtain that count. This structure has very desirable worst-case query performance but requires storage and preprocessing requirements similar to the range trees.

However, all the methods discussed above suffer as dimension increases. From the perspective of worst-case performance, an ideal nearest neighbor solution would be to preprocess the points in $O(n \log n)$ time, into a data structure requiring $O(n)$ space so that queries can be answered in $O(\log n)$ time. In dimension 1, this is possible by sorting the points, and then using binary search to answer queries. In dimension 2, this is also possible by computing the Voronoi diagram for the point set and then using any fast planar point location algorithm to locate the cell containing the query point [30, 86]. However, in dimensions larger than 2, the worst-case complexity of these techniques grows rapidly. Higher dimensional solutions with sublinear worst-case performance were considered by Yao and Yao [109]. Later Clarkson [24] showed that queries could be answered in $O(\log n)$ time with $O(n^{\lceil d/2 \rceil})$ space. The $O$-notation hides constant factors that are exponential in $d$.

### 2.3.4 Discussion

The general approach to search in documents is to convert them into text and then apply a search engine on the entire data. The text search engines are scalable to very large data sets and are very efficient in retrieval. Word morphology and form variations are handled very effectively by the text search engines. However, application of text search engines is limited due to unavailability of robust OCRs for document images. Then, the search is done in the image domain by representing the documents by $d$ dimensional feature vectors. Searching in such databases is simple and can be achieved by nearest neighbor search methods. Representations of two similar objects/words may have feature vectors that may differ a little in their values. In such representation of points/objects, the nearest neighbor techniques may not be able to retrieve many similar looking objects. In that case, approximate nearest neighbor solutions are necessary.

Apart from the requirement of search in approximate representation of words, there are difficulties caused by data size and the dimensionality of the representation. In any fixed dimension greater that 2, the traditional methods do not achieve the simultaneous goals of roughly linear space and logarithmic query time. The constant factors hidden in the asymptotic running time grow at least as fast as $2^d$ (depending on the metric). Arya et al. [4] showed that if data size $n$ is not significantly larger than $2^d$, as arises in some applications, then boundary effects mildly decrease this exponential dimensional dependence. In document databases the methods discussed above may not be much effective. Therefore, we explore the efficient approximate nearest neighbor search methods for handling the document image databases in retrieval task.

## 2.4 Approximate nearest neighbor (ANN) search

The apparent difficulty of obtaining algorithms that are efficient in the worst case with respect to both space and query time for dimensions higher than 2, suggest that the alternative approach of finding approximate nearest neighbors is worth considering [5]. Consider a set $S$ of data points in $R^d$ and a query point $q \in R^d$. Given $\epsilon > 0$, we say that point $p \in S$ is a $(1+\epsilon)$-approximate nearest neighbor of q if

$$dist(p,q) \leq (1+\epsilon)dist(p^*,q), \tag{2.1}$$

where $p^*$ is the true nearest neighbor to q. In other words, $p$ is within relative error $\epsilon$ of the true nearest neighbor.

Nearest or near-neighbor query problems arise in the context of similarity searching. Image indexing often involves offline nearest neighbor computations and storage for efficient access. These nearest neighbor techniques for image matching are expensive in high dimensions even when computed offline. Feature based image matching, searching and retrieval could fall prey to the "curse of dimensionality". That is the data structures scale poorly with data dimensionality. In recent years, several researchers have proposed methods for overcoming the running time bottleneck by using approximation. In this section we discuss some of the promising approximate nearest neighbor search techniques.

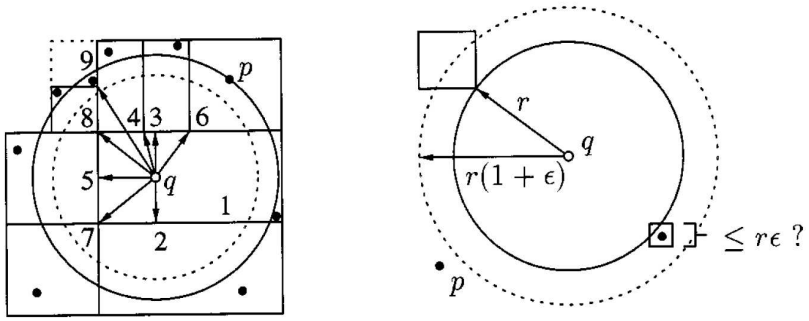



Figure 2.2: Illustration of ANN search using BBD tree

### 2.4.1 BBD-trees

Bern [17] proposed a data structure for approximate nearest neighbor problem. A data structure based on quad trees, which uses linear space provides logarithmic query time. A fixed function of the dimension was the approximate error factor for the algorithm. A randomized data structure that achieves polylogarithmic query time in the expected case and nearly linear space was proposed by Arya and Mount[3]. In this algorithm, the approximation error factor is an arbitrary positive constant, fixed at preprocessing time.

A data structure based on a hierarchical decomposition of space, called *balanced box decomposition (BBD)* tree is presented in [5]. The BBD tree has $O(\log n)$ height, and subdivides space into regions of $O(d)$ complexity. The regions are defined by axis-aligned hyper-rectangles that are fat, meaning that the ratio between the longest and shortest sides is bounded.

This data structure is similar to balanced structures based on box-decomposition, with few new elements included for the purposes of nearest neighbor searching and practical efficiency. Space is recursively subdivided into a collection of cells, each of which is either a $d$-dimensional rectangle or the set-theoretic difference of two rectangles, one enclosed within the other. Each node of the tree is associated with a cell, and hence it is implicitly associated with the set of data points lying within this cell. Each leaf cell is associated with a single point lying within the bounding rectangle for the cell. The leaves of the tree define a subdivision of space.

### 2.4.2 Locality sensitive hashing

Locality sensitive hashing(LSH) is a very efficient approximate nearest neighbor search technique. Indyk and Motwani [44, 50] proposed LSH to achieve sub-linear search time in a large databases of objects. The main idea of LSH is to hash points from database, using several hash functions, so as to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far apart. Thus, similar objects are mapped to the same index of the hash table. Given a query, hash the query point and retrieve elements stored in buckets containing that point. This speeds up the process of searching objects that are similar or nearest to the query objects.

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.* [96]. Matei *et al.* [73] 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 and also for finding documents from a set of camera-based document images [101, 79].

In particular all indexing techniques degrade to linear search for sufficiently high dimensions. The amount of data to be inspected for searching similarity of objects is large. In document images, represented by features of high dimension, the techniques degrade quickly even for moderate size data sets. Therefore, there is a need for searching only part of the data that is approximately relevant to the query. This requirement is accomplished by the LSH technique. LSH inspects only small portion of the data set in the process of search.

## 2.5 Applications of similarity search

Nearest neighbor searching has been used in a number of problems related to computer vision. In this section we discuss some of the nearest neighbor search techniques that have been used in solving problems such as image matching and object recognition.

### 2.5.1 Search in multimedia databases

An important research issue in the field of multimedia databases is the content based retrieval of similar multimedia objects such as images, text, and videos. However, in contrast to searching data in a relational database, a content based retrieval requires the search of

similar objects as a basic functionality of the database system. Most of the approaches addressing similarity search use a so-called feature transformation which transforms important properties of the multimedia objects into high-dimensional points (feature vectors). Thus, the similarity search is transformed into a search of points in the feature space which are close to a given query point in the high-dimensional feature space. A detailed survey of the indexing structures used for similarity search in multimedia databases is provided in [19].

Naively, a nearest neighbor query could be answered by examining the entire representative of every data object. Two general classes of possible approaches for the approximate searching problem naturally arise [37]:

- `Retrieved set reduction:` In this approach the data is organized to examine a small subset of the objects to answer a nearest neighbor query. The search technique focuses on the data most relevant, instead of the entire data set, to the given query. One effective and well-known solution is to index the multi-dimensional data [42]. The goal of all such index structures is to reduce the size of the data set which is retrieved as a result of a query.

- `Representative size reduction:` The representatives of the data set are organized to examine only a partial representation (e.g., 2 out of $d$ dimensions) of each object. Using multi-dimensional trees for indexing such data degrades the performance significantly [18, 107]. Considering all the dimensions at the same time dramatically degrades the efficiency of high dimensional query processing. Hence, high dimensional data is first reduced to lower dimensions and the search is conducted in the lower dimensional space [1].

The retrieved set reduction approach is important especially for scalability in large data sets. An approximate nearest neighbor (ANN) search technique which retrieves the feature vectors related to a subset of the data set clearly outperforms a technique that retrieves the feature vectors for the entire data set. Similarly, the representative size reduction approach is crucial for high dimensional data. Considering all the dimensions at the same time dramatically degrades the efficiency of high dimensional query processing.

### 2.5.2 Pattern recognition and classification

Nearest neighbor techniques are applied in a number of pattern classification problems. Searching near points in multi-dimensional data points is one of the earlier applications of

nearest neighbor search in pattern classification problems [28, 34]. The $k$-nearest neighbor technique is one of the most used technique in many applications such as text categorization, protein structure prediction, character recognition.

In recognizing objects, geometric hashing [63] and indexing methods are efficient and can easily be made parallel. These methods are especially attractive in model-based schemes, but they also hold significant advantages in pairwise object-scene comparisons because of their ability to also handle partially occluded objects. However, this is difficult because it is not known which data-base objects will appear and what their pose will be. The model information is encoded in a pre-processing step and stored in a large memory, in this case a hash table [108]. The contents of the hash table are independent of the scene and can thus be computed offline, not affecting the recognition time. Access to the memory is based on geometric information that is invariant of the object's pose and computed directly from the scene. During the recognition phase, and when presented with a scene, the method accesses the previously constructed hash table, indexing geometric properties of features extracted from the scene for matching with candidate models. A search of all scene features is still required, but geometric hashing obviates a search of all models and their features.

The ANN search technique is applied in $3D$ object indexing [73] and pose estimation problems. An example-based algorithm for fast parameter estimation using local models, which are dynamically built for each new input image is presented in [96]. The problem of computational complexity is overcome by employing LSH. The training examples are indexed by a number of hash tables for the parameters. The parameter sensitive hashing uses hash functions sensitive to the similarity in the parameter space, and retrieves in sublinear time approximate nearest neighbors of the input with respect to parameter values as well as the features. The key construction is a new binary feature space that is learned from examples in order to more accurately reflect the proximity in parameter space. The objective of parameter sensitivity is formulated in terms of a classification problem, and a simple and efficient algorithm is presented for evaluating this objective and selecting parameter-sensitive hash functions.

### 2.5.3 Document retrieval

The word wide web (WWW) contains information spread across very large number of huge databases. Searching for similar or related pages is one of the great challenges on WWW.

Traditional web search engines take a query as input and produce a set of (hopefully) relevant pages that match the query terms. While useful in many circumstances, search engines have the disadvantage that users have to formulate queries that specify their information need, which is prone to errors. A related web page is one that addresses the same topic as the query, but is not necessarily semantically identical. A number of algorithms have been proposed for searching related pages [31]. Evaluation and selection of a proper similarity search strategy is another important factor in finding related pages over the Internet. The quality of these measures mainly depends on the user feedback [47].

The methods available for searching in text documents may not be applied for searching in document image collections. The major challenge is the conversion of images into text [33, 76]. Hence, there is a need for mechanism to search in document image collections. In indexing and retrieval of document images, a large amount of time is spent in matching and indexing building. The time efficiency can be gained by eliminating inefficient preprocessing and applying techniques like hashing. Geometric hashing [63] has been successfully demonstrated for retrieval from a set of camera-based document images [101, 49]. In [101] feature based document image retrieval using invariants and hashing is presented. Indexing is done based on local combination of projective invariants calculated from feature points. For retrieval, a voting technique is employed for efficiency and robustness against erasure of feature points. A content-based image retrieval technique based on interest points matching and geometric hashing was presented in [49]. Features are extracted to describe images patches, small regions around interest points. To provide matching scheme invariant to local and global geometric transforms, the image patches are indexed into a 2-D hash table by geometric hashing. Here, the emphasis was on addressing the imaging related issues rather than scalability (e.g. perspectivity of the images) and word level document access. In these techniques, the features are extracted at document level to generate the index. A document image is required for querying the index. Therefore, content level access is not possible in such retrieval methods.

## 2.6 Summary and Comments

Our objective is to query a collection of document images. The document images are represented and accessed at word level. The words of the documents are represented using some features. The features are, generally, vectors containing numeric values of high-dimensions.

Similar words are to be searched to retrieve documents relevant to the queries. This can be done using nearest neighbor search techniques.

The representations of similar words may differ. In order to search in such small variations of features, approximate matches have to be determined. The nearest neighbor methods can give the most similar words. However, with some modifications to the data structures, $K$ nearest neighbors can be obtained. As the dimensionality of the data is very high, these structures may not be feasible. Therefore, we go for approximate nearest neighbor methods.

The approximate nearest neighbor search should also be acceptably efficient in searching the words. When the database is very large, the methods may perform very badly. The preprocessing time could also be high. There is a need for adopting methods that can preprocess in linear time and answer the queries in sub-linear time. This makes the searching in document images efficient and scalable to large collections.

We discussed about some popular data structures and methods for nearest neighbor (NN) search. Most of the NN search methods are feasible for low dimensional data only. Their query time increases with the dimensionality of the data sets. There is extensive need for similarity search in databases of multimedia, document databases and various problem of pattern recognition. Data organization and effective representation techniques have been adopted to achieve effective search in databases. However, as the data size increases the similarity search becomes time inefficient. Most practical applications of similarity search require approximate answers to queries rather than exact. In such cases, approximate nearest neighbor search methods are adopted. Locality sensitive hashing was found to be one of the best approximate nearest neighbor search technique for high dimensional data sets Other methods suffer from the computationally costly initial processing of the data.

We also discussed other methods of document search techniques. These include the recognition based, recognition free and annotation based methods. The traditional methods are also not feasible for search in high-dimensional representations of large collection of document images. Therefore, we resort to the approximate nearest neighbor search methods. In the next chapter, we explore the feasibility of locality sensitive hashing technique for indexing collection of words images. We determine effective word image representation based on a number of experimental results. Different data sets are used to determine the performance and effectiveness of the similarity search method.

# Chapter 3

# Approximate Nearest Neighbor Search in Document Databases

Searching in document images involves looking for similar words in the whole collection. Words inn the collection are represented by high-dimensional features. Due to variation in images of similar content, these representations may also vary. The problem of interest is to search in the collection and look for words which are similar to the query in this representation. The methods available in the literature for computing similarity are often computationally expensive and can not be scaled to large databases of high-dimensions. These methods require large amount of time to carry out the computations for preprocessing and search. Thus, we need an approximate nearest neighbor search (ANNS) technique that (i) can retrieve similar words efficiently, (ii) is scalable to large collections and (iii) takes less preprocessing time.

In this chapter we look at the ANNS applied for content-based image retrieval from a database of word images, along with a time efficient locality sensitive hashing (LSH) technique. We present various methods of representing word images using suitable features. The performance of LSH on different datasets of word image collections is measured and the results are discussed in detail.

## 3.1 ANN search for CBIR

Nearest neighbor searching is an important problem in a variety of applications, including knowledge discovery, data mining, pattern recognition, machine learning, data compression,

multimedia databases, information retrieval and statistics. The image and video databases contain a collection of objects that are characterized by a set of relevant features and represented as points in high-dimensional vector space. Given a query in the form of points in this space, a similarity search problem involves finding the nearest objects to the query.

Content-based image retrieval (CBIR) (i.e. content-based visual information retrieval (CBVIR) ) is the application of computer vision to the image retrieval problem, that is, the problem of searching for relevant images in large image databases [66]. Here, "content-based" means that the search will analyze the actual contents of the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image and represented in a feature space. In the context of document images, the content refers to the words in the images. Without the ability to examine image content, search may rely on meta data such as captions or keywords, which may be laborious or expensive to produce in all situations.

Two key characteristics of CBIR are (1) its use of image and video content—computable properties such as color, texture, shape, and motion from images and videos and (2) its graphical query process in which queries are posed by an example. To achieve this functionality, query by image content (QBIC) [39] has two main components: database population (the process of creating an image database) and database query. During the population, images and videos are processed to extract features describing their content—colors, textures, shapes, motion—and the features are stored in a database. During the query, the user composes a query graphically. Features are generated from the graphical query and then input to a matching engine that finds images or videos from the database with similar features. In the general framework of search in document images, a text query is given from which an image is generated. The image is represented using features for search in the collection of words.

Indexing tabular data for exact matching or range searches in traditional databases is a well-understood problem and structures like B-trees provide efficient access mechanisms. For queries in which similarity is defined as a distance metric in high dimensional feature spaces (for example, color histogram queries), indexing involves clustering and indexable representations of the clusters. In the case of queries that combine similarity matching with spatial constraints on objects, the problem is more involved. Data structures for fast access of high dimensional features for spatial relationships need to be invented. The color and texture features are the most commonly used representations. These representations may

not apply to databases of general imagery [106]. In the absence of such features, retrieval systems must resort to different features [80].

In a query, features from the database are compared to corresponding features from the query specification to determine which images are a good match. In an object shape query, the query specification is the drawn shape. Area, circularity, eccentricity, major-axis direction, features derived from the object moments, and a set of tangent angles around the object perimeter are the features used to characterize and match shapes. In a multi object query the features are standard color and texture. The matching is done by combining the color and texture distances. For a small database, a sequential scanning of the features followed by straightforward similarity computations is adequate.

## 3.2 Locality sensitive hashing (LSH) and application

A collection of documents is often characterized by a collection of relevant features. Typically, the features of the document words are represented as points in $\mathbb{R}^d$ and a distance metric is used to measure the similarity of the words. Our basic problem is to perform indexing or similarity searching for query words in such data sets which are points in a $d$ dimensional space. In searching for relevant words, it is not necessary to insist on the exact answer, instead, determining an approximate answer should suffice.

In this section, we describe locality sensitive hashing (LSH). This technique was originally introduced in [50] for the purpose of $\epsilon$-NNS problem. An enhanced version of this technique is used in chapter 4 for search in large databases with improved search time. Let $p = (x_1, \ldots, x_d)$ be a point representing a set of features of a word image and $P$ be set of all such points representing the collection of word images. Each point is transformed into a binary vector using equation 3.1. This embeds $P$ into the Hamming cube $H^{d'}$ with $d' = Cd$, where $C$ is the largest coordinate in $P$.

$$v(p) = Unary_C(x_1) \ldots Unary_C(x_d), \tag{3.1}$$

where $Unary_C(x)$ represents the unary representation of $x$, i.e., is a sequence of $x$ ones followed by $C - x$ zeroes. For example, if $x = \{1, 3, 2\}, d = 3$ and $C = 3$, then $v(x) =$ 100 111 110. The distance between any pair of points $p, q$ with coordinates in the set $1 \ldots C$ is given by,

$$d_1(p, q) = d_H(v(p), v(q)) \tag{3.2}$$

That is, the embedding preserves the distance between the points. The understanding of the hashing algorithm is made simple by using the embedding. In practice, the embedding becomes expensive when $C$ is large. A hash function $g(p) = p_{|I}$ is defined by projecting vector $p$ on a coordinate subset $I$ of $1, \ldots, d'$ ($1, 2, \ldots, 9$ when $d = C = 3$). That is, compute $p_{|I}$ by selecting the coordinate positions as per $I$ and concatenating the bits in those positions. Such $l$ (specified later) subsets $I_1, \ldots I_l$ of $1, \ldots, d'$ are chosen to denote $l$ hash functions $g_j(p) = p_{|I}$. For example, if $p = \{1, 3, 2\}$, and $I = \{2, 3, 7, 9\}$ then $g(p) = 0010$ ($value2indecimal$).

The hashing is done in two levels. In the first level, the functions map each point $p$ to bucket $g_j(p)$. In the second level, the contents of these buckets are mapped into a hash table of size $M$ by a standard hash function. The number of points $n$, the hash table size $L$ and the maximum bucket size $B$ are related by the following equation:

$$M = \alpha \frac{n}{B}, \tag{3.3}$$

where $\alpha$ is the memory utilization parameter i.e., the ratio of the memory allocated for the index to the size of the data set.

Let $D(\cdot, \cdot)$ be a distance function of elements from a set $S$, and for any points $p \in S$ let $B(p, r)$ denote the set of elements from $S$ within the distance $r$ from $p$. If $D(\cdot, \cdot)$ is the Hamming distance, then the family of projections on one coordinate is locality-sensitive. A family $H$ of functions from $S$ to $U$ is locality-sensitive if the following conditions hold.

$p \in B(q, r_1)$ then $Pr_H[h(q) = h(p)] \geq p_1$
$p \notin B(q, r_2)$ then $Pr_H[h(q) = h(p)] \leq p_2$
where the probabilities are $p_1 = p(r_1)$ and $p_2 = p(r_2)$

To process a query $q$, all the indices $g_1(1), \ldots, g_l(q)$ are searched until either at least $c \cdot l$ points are encountered or all $l$ indices are used. The set $I$ consists $k$ elements sample uniformly at random with replacement from $1, \ldots, d'$. An optimal value of $k$ is chosen to maximize the probability of two points $p$ and $q$ to fall into the same bucket if they are close to each other. Also, the probability of $p$ and $q$ falling into same bucket is minimized when they are far apart. The values of $k$ and $l$ are given by,

$$k = \log_{1/p_2}(n/B) \tag{3.4}$$

$$l = \left(\frac{n}{B}\right)^{\rho}, \ where\ \rho = \frac{ln(1/p_1)}{ln(1/p_2)} \tag{3.5}$$

There is need to compute $p'_{|I}$ for an image $p'$ of point $p$ from the data set, where $I$ is the set of coordinates as mentioned earlier. For $i = 1, \ldots, d$, let $I_{|i}$ denote, in sorted order, the coordinates in $I$ which correspond to the $i$th coordinate of $p$. It is sufficient to compute $o_i$ for $i = 1, \ldots, d$ in representing $p'_{|I}$. Because, it results in a sequence of bits which is monotone, i.e., consists of a number, say $o_i$, of ones followed by zeros. This task is equivalent to finding the number of elements in the sorted array $I_{|i}$ which are smaller than a given values, i.e., the $i$th coordinate of $p$. This can be done via binary search in $\log(C)$ time, or even in constant time using precomputed array of $C$ bits. Thus, the total time needed to compute the function is either $O(d\log(C))$ or $O(d)$, depending on resources used.

### 3.2.1 A simple example of LSH

The hashing technique explained above can use a number of different hash functions for indexing purpose. The values of these functions can be results of simple mathematical operations or chosen from a distributions. We demonstrate the operation of above mentioned technique with the help of a simple example. We use three dimensional data to demonstrate the LSH technique.

Let $p = \{1, 3, 2\}, q = \{1, 2, 3\}, r = \{3, 1, 1\}$ are the three dimensional ($d = 3$) data points. Now we embed the points into Hamming space. The maximum value in the dimensions is $C = 3$. We assume the number of hash tables is $L = 2$. After embedding the points into Hamming space the new dimension obtained is $d' = Cd = 9$ and $I_1 = \{1, 5, 6\}$ and $I_2 = \{2, 3, 7, 9\}$ are L subsets from $d'$. Now, we apply the operation of equation 3.1 for the data points as follows,

$v(p) = 100\ 111\ 110$
$v(q) = 100\ 110\ 111$
$v(r) = 111\ 100\ 100$

The next step is to compute the hash values of these points for each of the $L$ hash tables. We have defined the hash functions as:

$g_L(p) = v_1(p) \dots v_L(p)$
$v_i(p) \Rightarrow$ select $I_i$ bits from $v(p), i = 1 \dots L$
$g_1(p) = 111,\ g_2(p) = 0010.\ (7, 2)$
$g_1(q) = 110,\ g_2(q) = 0011.\ (6, 3)$
$g_1(r) = 100,\ g_2(r) = 1110.\ (4, 13)$

Let the query is $s = \{2, 1, 1\}$ and by following the above conversions and applying the hash functions

$v(s) = 110\ 100\ 100$
$g_1(s)$ =110 and $g_2(s)$ =1010. (4, 10)

we obtain $r$ as the resulting near point. The example is just an illustration of the operation of LSH. The LSH is designed for large data bases of high dimensionality. Lower dimension data bases can be searched easily using $B$-tree or similar other data structures.

## 3.3 Indexing document image databases using LSH

Digital libraries are digitizing a large number of documents through scanning generating a very large data bases of document images. Searching in these document images for specific content is a difficult task. We formulate this problem as content based image retrieval. Any text search engine uses words as the basic components for searching. Similarly, the search in document images also should be at the word level. Words are extracted from the images by applying a method called segmentation. The segmentation splits the document image into words and returns images of words. However, these words are still images and can not be used like the text words. Hence, we represent these words with the help of special features. Search for any document at content level requires searching through this database of word features for features similar to the query words. The query also has to be in the form of features from the word image.

### 3.3.1 Representation of word images

Two images of the same words may differ in a number of ways due to pixel variations, noise etc. A description or representation is required for the words of the documents which will allow matching inspite of these differences. Building an appropriate description is critical

to the robustness of the system against signal noise. In general, color, shape or/and texture features are used for characterizing the content in content-based image retrieval systems. More specific features are required for word representation in document images. These features can be more specific to the domain as they contain an image-description of the textual content in it. It is observed that many of the popular structural features work well for good quality documents. Word images, particularly from newspapers and old books, are of extremely poor quality. Common problems in such document databases will have to be analyzed before identifying the relevant features.

Some of the popular artifacts in printed document images include [75] (a) Excessive dusty noise, (b) Large ink blobs joining disjoint characters or components, (c) Vertical cuts due to folding of the paper, (d) Cuts in arbitrary directions due to paper quality or foreign material, (e) Degradation of printed text due to the poor quality of paper and ink, (f) Floating ink from facing pages. An effective representation of the word images will have to take care of these artifacts for successful indexing and retrieval.

We found that three categories of features are effective for addressing these artifacts. The features could be a sequential representation of the word shape, content or a structural representation of the word image. We employ a combination of scalar, profile, structural and transform domain feature extraction methods as used in [10, 90, 91]. The images we operate on are all binary with two intensity levels $[0, 255]$. We refer to an image of height $h$ as $I(r, c)$, where $r$ and $c$ indicate the row and column, respectively, index of the pixel.

**Scalar Features:** Scalar features include an estimate of the number of ascenders and descenders in the word and the aspect ratio of the image.

**Profile Features:** The profile features include projection profiles, background to ink transitions, upper and lower word profiles.

- `Projection profile:` It is a measure of ink distribution of the word image. Each value in the profile is calculated by summing over the pixel values in the corresponding image column (Eq. 3.6). Figure 3.1(f) shows the plot of a typical projection profile. We invert the image before the calculation, causing concentrations of ink to create peaks, because we would like to measure the ink contained in each column. The descriptive power of the modified profile remains unchanged.

$$pp(I,c) = \sum_{r=1}^{h}(255 - I(r,c)) \tag{3.6}$$

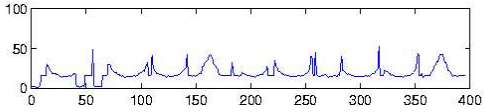
- `Ink transition:` Ink transition represent the internal shape of the image, computed by counting the number of transitions $ntrans(I,c)$ from background (paper) to "ink" for every column of the image (see Figure 3.1(g)).

$$it(I,c) = ntrans(I,c) \tag{3.7}$$

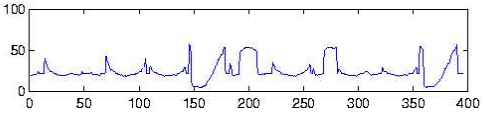

(a) Word image to be represented


(b) Projection from upper boundary of the word

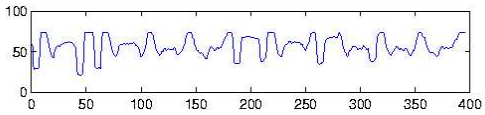

(c) Projection from lower boundary of the word



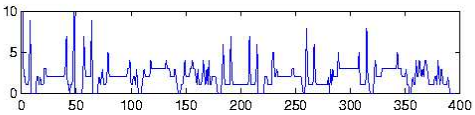

(d) Upper word profile




(e) Lower word profile




(f) Projection profile




(g) Background to ink transition

Figure 3.1: Representing word images using features for indexing and retrieval

- `Upper and lower word profile:` The outline shape of the word is captured by upper and lower word profiles. Upper (lower) word profile features are computed by recording – for each image column – the distance from the upper (lower) boundary of the word image to the closest "ink" pixel. Ink pixels are determined by a thresholding algorithm that classifies pixels into the categories ink and paper. If an image column does not contain ink, the feature value is taken as 0. Figures 3.1(d) and 3.1(e) shows two typical profiles (feature values are inverted).

$$up(I,c) = \begin{cases} \text{undefined}, & \text{if } \forall r \ (is\_ink(I,r,c) = 0) \\ argmin_{r=1\ldots h}(is_ink(I,r,c) = 1), & \text{otherwise} \end{cases} \tag{3.8}$$

$$lp(I,c) = \begin{cases} \text{undefined}, & \text{if } \forall r \ (is\_ink(I,r,c) = 0) \\ argmax_{r=1\ldots h}(is_ink(I,r,c) = 1), & \text{otherwise} \end{cases} \tag{3.9}$$

where is_ink(I,r,c) returns 1 if I(r,c) is foreground pixel, and 0 otherwise.

**Structural Features:** The structure of the word image is described by statistical moments and region-based moments. Each moment order carries different structural information about the same image. Many of the pattern recognition problems are addressed used different moment orders. Normalized moments, such as first-order moments ($M_{00}$, $M_{01}$), central moments ($CM_{pq}$), and statistical moments (mean, standard deviation) are employed in this work for describing the structure of the word. Structural features are also robust for representing images containing various artifacts and noise like, salt and pepper noise. Moments of order $(p+q)$ for image $I$ are given by the following equations.

$$M_{pq} = \sum_r \sum_c r^p c^q I(r,c) \tag{3.10}$$

$$CM_{pq} = \sum_r \sum_c (r-\overline{r})(c-\overline{c}) I(r,c) \tag{3.11}$$

where, the region-based moments along the row and column of the image are given by:

$$\overline{r} = \frac{M_{10}}{M_{00}}, \overline{c} = \frac{M_{01}}{M_{00}} \tag{3.12}$$

**Transformed Domain Representations:** A compact representation of a series of observations (such as profiles) is based on the Fourier Transform. A few coefficients are enough to represent a word robustly in a transformed domain, and these coefficients are matched at a coarse level for recognition.

A DFT is performed on feature vector $\mathbf{f} = f_0, \ldots, f_{n-1}$ to obtain its frequency space representation $\mathbf{F} = F_0, \ldots, F_{n-1}$:

$$F_k = \sum_{l=0}^{n-1} f_l \cdot e^{-2\pi ilk/n}, \quad 0 \leq k \leq n-1 \tag{3.13}$$

Then the first $c$ real components and $c-1$ imaginary components are extracted from the DFT representation for use as scalar features. The word feature is split vertically into four parts of equal width. Then $2 \cdot c - 1$ Fourier features are obtained from each of these parts. In total we use 84 Fourier coefficients (with $c = 4$) of the segmented profiles and ink transition features to represent the word images.

### 3.3.2 The basic steps

The problem of searching similar words is solved using the LSH technique explained in Section 3.2. The points, which represent words, are preprocessed by hashing for answering the near neighbor queries. The steps involved in the preprocessing are enumerated below. The preprocessing step give a number of hash tables which contain the data points.

1. `Input:` A set of points $P$, which are representations of word images from document images. The number of hash tables $l$.
2. `Output:` Hash tables $T_i, i = 1 \ldots l$
3. Initialize each hash table $T_i, i = 1 \ldots l$ by generating a random hash function $g_i(\cdot)$
4. Store point $p_j$ on bucket $g_i(p_j)$ of hash table $T_i, i = 1 \ldots l$. Repeat this step for all the $l$ hash tables.

Searching for similar words involves similarity search in the database of $P$ points. Therefore, we have to query the hash tables built in the preprocessing step. The method of querying for approximate nearest points in the database is as enumerated below.

1. `Input:` A query point $q$ and $K$ (number of approximate nearest neighbors).
2. Access to hash tables $T_i, i = 1 \ldots l$ generated by the preprocessing algorithm. $S \leftarrow \emptyset$
3. `Output:` $K$ (or less) approximate nearest neighbors.
4. $S \leftarrow S \cup \{\text{points found in } g_i(q) \text{ bucket of table } T_i, i = 1 \ldots l\}$. Repeat the step for each hash table $l$

5. Return the $K$ nearest neighbors of $q$ found in set $S$.

Thus, the problem of searching for similar word images in a large database of words from document images is solved using approximate near neighbor search technique. Given a query word image, the relevant documents can retrieved by searching for similar words in sublinear time.

## 3.4 Experiments and results

The proposed method of indexing and retrieval is tested on a collection of word images from the English language. The main focus of the experiments is on the accuracy values obtained by querying the hash table of word image feature points from the collection.

We employ the hashing method as explained in this chapter. Computation of hash functions is different from the way it was explained in section 3.2. 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$[0, w]$ ($w = 4$ provides good results). For a fixed $a$, $b$ the hash function $h_{a,b}$ is given by equation 3.14. Details about the computation of other parameters of the hashing technique are presented in Chapter 4.

$$h_{a,b}(x) = \left\lfloor \frac{a \cdot x + b}{w} \right\rfloor \tag{3.14}$$

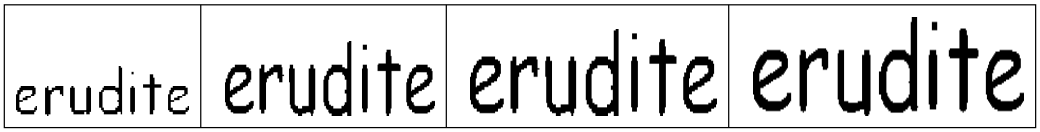
### 3.4.1 Data set generation

We need word images for testing the LSH method. The word images can be generated from text or obtained by segmenting the document images. Getting word images from real document images involves a few steps of processing. It requires scanning document images, processing for noise and skew removal etc., and then segmentation. We generate images, using an image generation program, from a text corpus obtained from online newspapers and dictionaries.

Given a text word, the program generates and returns an image of the text. Most of the image processing libraries are capable of doing this operation. ImageMagick [68] is one such library that we made use of in generation of word images for the experiments. Given the font information, it has support for generating images of different styles, sizes and shapes.


(a) Images generated using "Arial" font


(b) Images generated using "Times" font




(c) Images generated using "Comic" font

Figure 3.2: Data sets (English): Sample word images from the English data sets. EG_Data_1 contains word image in only one of these fonts. EG_Data_2 contains word images in "Arial" and "Times" fonts (3.2(a) and 3.2(b)). EG_Data_3 contains images in all three fonts.

The library supports other image processing operations also. In this chapter we present experiments conducted on the data sets to test the performance of hashing technique.

### 3.4.2 Data sets

The data sets of English language are used for evaluating the performance of hashing technique. The size and characteristics of the data in these data sets is discussed below.

**Dataset EG_Data_1:** English word images of around 2200 words. The images were in a single font with varying images sizes. The image heights varied from 35 to 75 pixels (see Figure 3.2).

**Dataset EG_Data_2:** Around 3520 word images using two similar looking fonts (e.g. Arial and Times, see Figure 3.2(a) and 3.2(b)). The image heights in this case also varied from 35 to 75 pixels.

**Dataset EG_Data_3:** A collection of 7920 word images in three different looking fonts (e.g. Arial, Comic and Times). The image heights in this case also varied from 35 to 75 pixels (see Figure 3.2(a), 3.2(b) and 3.2(c)).

### 3.4.3 Results and discussion

The search technique discussed above is evaluated by experimenting with the different data sets. The generated words in the above mentioned data sets contain annotations that are used for performance evaluation. Different experiments are conducted for testing and evaluation of the hashing method. We use precision and recall along with F-score for measuring the performance. We analyze and discuss the results obtained in the experiments in the rest of the chapter.

#### Performance measure

The performance of the search results is measured in terms of precision and recall values. Precision is defined as the number of relevant words retrieved by a search divided by the total number of words retrieved by that search. Recall is defined as the number of relevant words retrieved by a search divided by the total number of existing relevant words in the database, which should have been retrieved). A popular measure that combines Precision and Recall is the weighted harmonic mean of precision and recall, called F-score or F-measure.

$$Precision = \frac{|\{\text{Relevant Words in Database}\} \cap \{\text{Retrieved Words}\}|}{|\{\text{Retrieved Words}\}|} \tag{3.15}$$

$$Recall = \frac{|\{\text{Relevant Words in Database}\} \cap \{\text{Retrieved Words}\}|}{|\{\text{Relevant Words in Database}\}|} \tag{3.16}$$

$$Fscore = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.17}$$

**Performance on different data sets:** The data sets explained in the previous section are used for testing the performance of hashing technique. The precision and recall values along with the F-score measure are shown in table 3.1. It can be observed from the results that the performance is acceptable in terms of the number of words that are searched in this method. Some examples of the retrieved set of word images obtained by querying are shown in Figure 3.3. It is clearly observed that the retrieved results are all similar in content.

An example of search results obtained by querying the hash tables on an English data set is shown in Figure 3.4. The word image in Figure 3.4(a) is the query and rest of the images are the retrieved images. The projection profile and ink transition features are also shown along with the word images to demonstrate the representation using features. It is observed

| Data Set | Precision | Recall | F-score |
|---|---|---|---|
| English EG_Data_1 | 97.23 | 98.00 | 97.61 |
| English EG_Data_2 | 94.45 | 96.80 | 95.61 |
| English EG_Data_3 | 63.70 | 56.47 | 59.87 |

Table 3.1: Search performance on generated data sets of Telugu and English language word images.

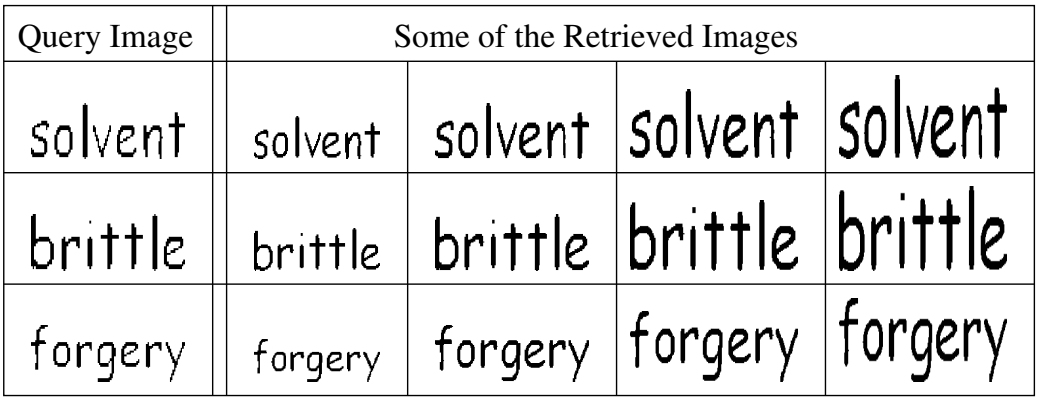
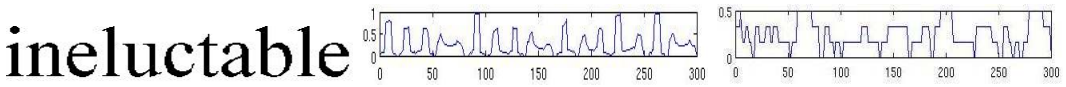



Figure 3.3: Results: Example words retrieved from querying in English data sets.

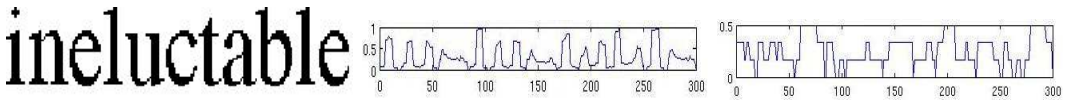
from the result in Figure 3.4(d), that the wrongly retrieved word has a representation similar to the correctly retrieved words. There is only a small variation in the projection profile, which is approximated in the search. Hence, small variations in the words are ignored by the retrieval technique.

**Effect of query radius:** The LSH is an approximate near neighbor search technique for large databases. To obtain the best precision and recall values the query distance determines the nearest points that are to be specified during the query process. Getting the right query radius that gives good performance values is very important. The right query radius value is determined experimentally by querying the index with different radius values. The plot shown in Figure 3.5 portrays the effect of query radius on the precision and recall values. The plot is obtained on the English data set EG_Data_2. In the experiments conducted, the number of hash tables required was $L = 6$ for obtaining the desired performance values.

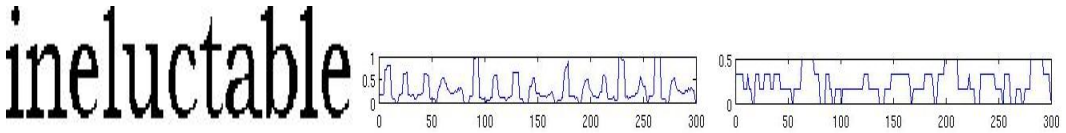
**Effect of fonts:** In an another experiment, data set EG_Data_3 was used to test the effect of different fonts on the performance of the proposed technique. Table 3.2 shows the



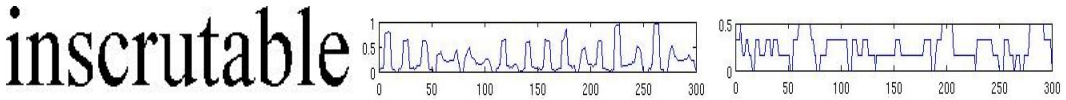

(a) Input query word image




(b) Example of correct retrieved image




(c) Example of correct retrieved image




(d) Example of wrong image retrieved

Figure 3.4: Search result: Some example word images obtained as a result of querying the word image with content "ineluctable" in the hash table.

changes observed in the precision and recall values with change in query radius. It is clearly observed from the table that the technique works very well for data sets of single font word images (see columns 4 to 9). When words of different fonts are combined together, the performance degrades immediately (see columns 2 and 3).

| **Query** | **Overall Performance** | | **Font - Arial** | | **Font - Times** | | **Font - Comic** | |
|---|---|---|---|---|---|---|---|---|
| **Radius** | **Precision** | **Recall** | **Prec.** | **Recall** | **Prec.** | **Recall** | **Prec.** | **Recall** |
| 0.15 | 85.41 | 19.79 | 84.80 | 74.40 | 79.27 | 59.90 | 84.00 | 57.50 |
| 0.20 | 96.36 | 32.62 | 92.40 | 87.60 | 96.54 | 88.73 | 98.66 | 86.00 |
| 0.24 | 96.51 | 39.88 | **95.12** | **92.80** | **97.26** | **96.39** | **96.80** | **96.50** |
| 0.28 | 93.87 | 44.23 | **97.22** | **98.00** | **91.37** | **98.64** | **90.49** | **99.00** |
| 0.34 | 69.23 | 54.09 | 84.21 | 100.00 | 58.17 | 100.00 | 53.83 | 100.00 |
| 0.38 | 46.02 | 63.83 | 63.20 | 100.00 | 33.89 | 100.00 | 30.74 | 100.00 |
| 0.40 | 35.24 | 68.52 | 52.60 | 100.00 | 21.68 | 100.00 | 20.21 | 100.00 |

Table 3.2: Font specific performance of retrieval technique on a collection of word images

The performance of the hashing technique is well acceptable for features obtained from single font word images (data set EG_Data_1). It can be observed from the results shown

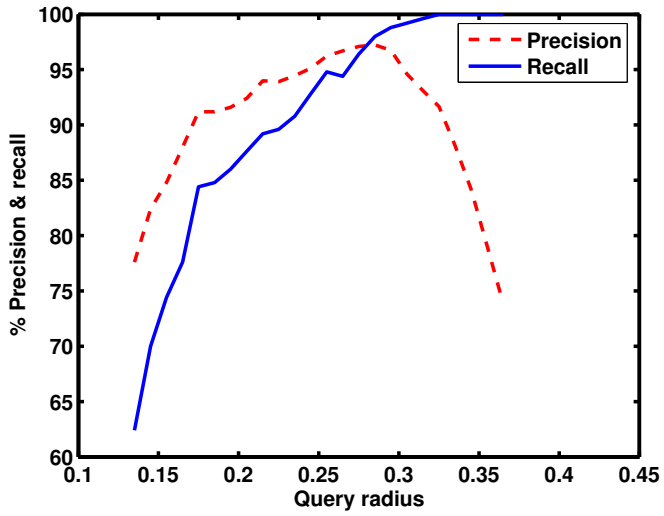



Figure 3.5: Effect of query radius on the Precision and Recall values in retrieving the word images.

in Table 3.1 also that the performance degrades (data set EG_Data_3) if word images of different font types are hashed together. The performance is acceptable with a data set obtained from word images of similar looking fonts (data set EG_Data_2). Therefore, it is concluded that the method of indexing using LSH is effective for the feature points extracted from collection of word images that are in single or similar looking font type.

**Selection of features:** A feature may not be good by itself performance-wise. However it can provide a significant performance improvement when combined with others. This requires a method to combine them. The performance of individual feature is shown in Table 3.3. The variation of performance with combination of some of these feature is shown in Table 3.4. It can be seen that features projection, lower and upper word profiles perform well individually. Their relative good level of performance is attributed to their representation of the word image using its internal characteristics, which further indicates the relative high level of information contained by them.

Out of the combined runs of each of the four features, background to ink transition and profiles register the highest F-score of 94.34%. This indicates that combining these features result in a better representation of a word image for the retrieval task.

| Features | Precision | Recall | F-score |
|---|---|---|---|
| Projection profile (*pp*) | 91.93 | 80.64 | 85.91 |
| Upper word profile (*up*) | 87.63 | 74.19 | 80.35 |
| Lower word profile (*lp*) | 93.22 | 88.70 | 90.90 |
| Ink transition (*it*) | 80.64 | 59.67 | 68.58 |
| Moments | 22.58 | 25.75 | 24.06 |

Table 3.3: Feature specific performance of retrieval technique

| Features | Precision | Recall | F-score |
|---|---|---|---|
| *pp* and *it* | 95.69 | 87.09 | 91.18 |
| *lp* and *up* | 92.90 | 91.93 | 92.41 |
| *lp* and *it* | 93.54 | 80.64 | 86.61 |
| *up* and *it* | 95.69 | 87.09 | 91.18 |
| All profiles | 93.01 | 93.54 | 93.27 |
| All profiles and ink transition | 95.16 | 93.54 | 94.34 |

Table 3.4: Retrieval performance with combination of different features

## 3.5 Summary and comments

Search in document image databases is similar to the content based image retrieval (CBIR) problem. The content of images here is the text. Given a query, we need to search for word images of similar textual content. This problem is addressed by applying a hash based indexing and search technique. The images were represented by a set of features that are used for indexing and search. The experimental results clearly suggest that the technique is efficient unlike the word spotting, direct image matching or other traditional approaches.

The word similarity search is possible in time much lower, as compared to other methods available for document images. Experimental results have been presented on different data sets to test the time efficiency. The proposed search method is easily scalable to large homogeneous (images in similar looking fonts) collections of document images. We demonstrate the scalability of the technique on document images obtained from a collection of books in next chapter. The effect of data size on the query time and more experiments are conducted to evaluate the technique.

It can be inferred from the experimental results that the features are good enough for representing the word images. This makes the features as possible candidates to represent

word images of any other language. However, the feasibility of these features have to be explored. As there are no efficient search techniques for Indian language documents, we explore the possibility of applying our method on them. In the next chapter we present results of search in a collection of documents from an Indian language and discuss the performance in comparison with an other popular method.

# Chapter 4

# Search in Collection of Kalidasa Books

Retrieval is primarily about searching in a collection of items for items that are similar to a query. It is about matching some user stated query against useful parts of records and presenting the matched results to the user. The records could be any type of structured text, such as bibliographic records, newspaper articles, or paragraphs in a user manual, multimedia records like, audio and video data. The queries can be few sentence descriptions or words of an information to be known.

## 4.1 Document search and retrieval systems

A search engine allows one to ask for content meeting specific criteria (typically those containing a given word or phrase) and retrieves a list of references that match those criteria. A regularly updated index is used by the search engine for efficient search and retrieval. Typically search engines are designed for textual documents and need special care to adapt to document images. Scalability of these search engines to adapt to the ever growing size of digital libraries, adaptation of language processing modules in the context of document images and many other search areas still remains as an open area. 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 (OCRs) and direct matching of images is inefficient due to the complexity of matching and thus impractical for large databases.

### 4.1.1 Search engine for document images

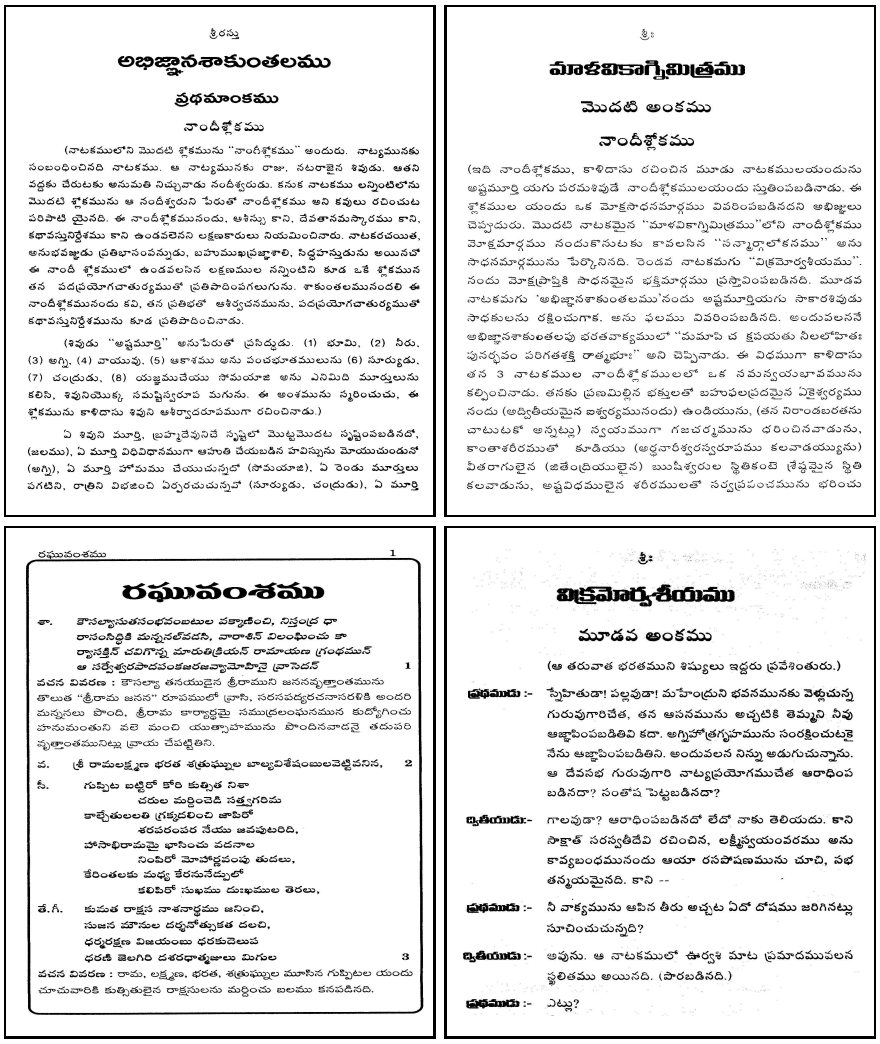
A large collection of printed and handwritten documents are being archived in digital libraries such as the Digital Library of India (DLI) [81]. The DLI aims at digitizing all literary, artistic, and scientific works of mankind so as to create better access to traditional materials, easier preservation, and making documents freely accessible to the global society. A Number of scanning centers all over India are working on the digitization of books and manuscripts. Each of these scanning centers has a number of scanners each one of them capable of scanning approximately 5000 pages in 8 hours. As on April 2008, close to 93,000 books were digitized and made available online by DLI (*http://dli.iiit.ac.in*) as document images.

Effective access to these document image collections requires designing and building a mechanism for effective search and retrieval of textual data from document image collections. Document image indexing and retrieval were studied with limited scope in the literature [33]. The success of these procedures mainly depends on the performance of the OCRs, which convert the document images into text. Much of the data in DLI are in Indian languages. Searching in these document image collections based on content, is not presently possible. This is because OCRs are not yet able to successfully recognize printed texts in Indian languages. We need an alternate approach to access the content of these documents. A promising alternate direction is to search for relevant documents in image space without any explicit recognition.

### 4.1.2 Indian language documents

A number of collections of historical prints, writings, manuscripts and books exist in Indian languages that need search options in images. The document images of such collections can not be recognized accurately. The image domain search systems also are not scalable to such large collections. Kalidasa's collection is one such collection, which consists of seven books.

Kalidasa was a *Sanskrit* poet and dramatist of India. His collection of writings comprise three plays and four poetries. The three famous plays written by Kalidasa are Malavikagnimitram (Malavika and Agnimitra), Vikramorvasiya (pertaining to Vikrama and Urvashi)



శ్రీరస్తు

## అభిజ్ఞానశాకుంతలము

### ప్రథమాంకము

#### నాందీశ్లోకము

(నాటకములోని మొదటి శ్లోకమును "నాందీశ్లోకము" అందురు. నాట్యమునకు సంబంధించినది నాటకము. ఆ నాట్యమునకు రాజు, నటరాజైన శివుడు. ఆతని వద్దకు వేరుటకు అనుమతి నిచ్చువాడు నందీశ్వరుడు. కనుక నాటకము లన్నింటిలోను మొదటి శ్లోకమును ఆ నందీశ్వరుని పేరుతో నాందీశ్లోకము అని కవులు రచించుట పరిపాటి యైనది. ఈ నాందీశ్లోకమునందు, ఆశిస్సు కాని, దేవతానమస్కారము కాని, కథావస్తునిర్దేశము కాని ఉండవలెనని లక్షణకారులు నియమించిరి. నాటకరచయిత, అనుభవజ్ఞుడు ప్రతిభానంపన్నుడు, బహుముఖప్రజ్ఞాశాలి, సిద్ధహస్తుడును అయినచో ఈ నాందీ శ్లోకములో ఉండవలసిన లక్షణములు నన్నింటిని కూడ ఒకే శ్లోకమున తన పదప్రయోగచాతుర్యముతో ప్రతిపాదింపగలుగును. శాకుంతలమునందలి ఈ నాందీశ్లోకమునందు కవి, తన ప్రతిభతో ఆశీర్వచనమును, పదప్రయోగచాతుర్యముతో కథావస్తునిర్దేశమును కూడ ప్రతిపాదించినాడు.

(శివుడు "అష్టమూర్తి" అనుపేరుతో ప్రసిద్ధుడు. (1) భూమి, (2) నీరు, (3) అగ్ని, (4) వాయువు, (5) ఆకాశము అను పంచభూతములును (6) సూర్యుడు, (7) చంద్రుడు, (8) యజ్ఞముచేయు సోమయాజి అను ఎనిమిది మూర్తులును కలిసి, శివునియొక్క సమష్టిస్వరూప మగును. ఈ అంశమును స్మరించుచు, ఈ శ్లోకమును కాళిదాసు శివుని ఆశీర్వాదరూపముగా రచించినాడు.)

ఏ శివుని మూర్తి, బ్రహ్మదేవునిచే సృష్టిలో మొట్టమొదట సృష్టింపబడినదో, (జలము), ఏ మూర్తి విధివిధానముగా ఆహుతి చేయబడిన హవిస్సును మోయుచుండునో (అగ్ని), ఏ మూర్తి హోమము చేయుచున్నదో (సోమయాజి), ఏ రెండు మూర్తులు పగటిని, రాత్రిని విభజించి ఏర్పరచుచున్నవో (సూర్యుడు, చంద్రుడు), ఏ మూర్తి

శ్రీః

## మాళవికాగ్నిమిత్రము

### మొదటి అంకము

#### నాందీశ్లోకము

(ఇది నాందీశ్లోకము, కాళిదాసు రచించిన మూడు నాటకములయందును అష్టమూర్తి యగు పరమశివుడే నాందీశ్లోకములయందు స్తుతింపబడినాడు. ఈ శ్లోకముల యందు ఒక మోక్షసాధనమార్గము వివరింపబడినదని అభిజ్ఞులు చెప్పుదురు. మొదటి నాటకమైన "మాళవికాగ్నిమిత్రము"లోని నాందీశ్లోకము మోక్షమార్గము నందుకొనుటకు కావలసిన "సన్మార్గాలోకనము" అను సాధనమార్గమును సేర్కొనినది. రెండవ నాటకమగు "విక్రమోర్వశీయము". నందు మోక్షప్రాప్తికి సాధనమైన భక్తిమార్గము ప్రస్తావింపబడినది. మూడవ నాటకమగు 'అభిజ్ఞానశాకుంతలము'నందు అష్టమూర్తియగు సాకారశివుడు సాధకులను రక్షించుగాక, అను ఫలము వివరింపబడినది. అందుపలననే అభిజ్ఞానశాకుంతలపు భరతవాక్యములో "మమాపి చ క్షపయతు నీలలోహితః పునర్భవం పరిగతశక్తి రాత్మభూః" అని చెప్పినాడు. ఈ విధముగా కాళిదాసు తన 3 నాటకముల నాందీశ్లోకములలో ఒక సమన్వయభావమును కల్పించినాడు. తనకు ప్రణమిల్లిన భక్తులతో బహుఫలప్రదమైన ఏకైశ్వర్యము నందు (అద్వితీయమైన ఐశ్వర్యమునందు) ఉండియును, (తన నిరాండబరతను చాటుటకో అన్నట్లు) స్వయముగా గజచర్మమును ధరించినవాడును, కాంతాశరీరముతో కూడియు (అర్ధనారీశ్వరస్వరూపము కలవాడయ్యును) వీతరాగులైన (జితేంద్రియులైన) ఋషీశ్వరుల స్థితికంటె (శ్రేష్ఠమైన స్థితి కలవాడును, అష్టవిధములైన శరీరములతో సర్వప్రపంచమును భరించు

రఘువంశము 1

## రఘువంశము

శా. కౌసల్యాసుతనంభవంబటుల వక్కాణించి, నిస్తంద్ర ధా
రానంసిద్ధికి మన్ననల్బడసి, వారాశిన్ విలంఘించు కా
ర్యానక్తిన్ చవిగొన్న మారుతత్రియన్ రామాయణ గ్రంథమున్
ఆ నర్వేశ్వరపాదపంకజరజవ్యామోహినై ప్రాసెదన్ 1

వచన వివరణ : కౌసల్యా తనయుడైన శ్రీరాముని జననవృత్తాంతమును తొలుత "శ్రీరాము జనన" రూపములో వ్రాసి, సరసపద్యరచనానరభికి అందరి మన్ననలు పొంది, శ్రీరామ కార్యార్థమై సముద్రలంఘనమున కుద్యోగించు హనుమంతుని వలె మంచి యుద్యోగమును పొందినవాడనై తదుపరి వృత్తాంతమునిట్లు వ్రాయ చేపట్టితిని.

వ. శ్రీ రామలక్ష్మణ భరత శత్రుఘ్నుల బాల్యవిశేషంబులవెట్టివనిన, 2

సీ. గుప్పిట బట్టిరో కోరి కుత్తుక నిశా
చరుల మర్దించెది సత్త్వగరిమ
కాల్చేతులలతి గ్రక్కదలించి జాపిరో
శరపరంపర నేయు జవవుటరిది,
హాసాభిరామమై భాసించు వదనాల
నించిరో మోహోద్దవంపు తుదలు,
కేరింతలకు మధ్య కేరనునేడ్పులో
కలిపిరో సుఖము దుఃఖముల తెరలు,

తే.గీ. కుమత రాక్షస నాశనార్థము జనించి,
సుజన మౌనుల దర్శనోత్సుకత దలచి,
ధర్మరక్షణ విజయంబు భరకుదెలుప
భరతి జెలంగిరి దశరథాత్మజులు మింగల 3

వచన వివరణ : రాము, లక్ష్మణ, భరత, శత్రుఘ్నుల మూసిన గుప్పిటల యందు చూచువారికి కుత్తుకలైన రాక్షసులను మర్దించు బలము కనపడినది.

శ్రీః

## విక్రమోర్వశీయము

### మూడవ అంకము

(ఆ తరువాత భరతముని శిష్యులు ఇద్దరు ప్రవేశింతురు.)

**ప్రథముడు :-** స్నేహితుడా! పల్లవుడా! మహేంద్రుని భవనమునకు వెళ్లుచున్న గురువుగారిచేత, తన ఆసనమును అచ్చటికి తెమ్మని నీవు ఆజ్ఞాపింపబడితివి కదా. అగ్నిహోత్రగృహమును సంరక్షించుటకై నేను ఆజ్ఞాపింపబడితిని. అందువలన నిన్ను అడుగుచున్నాను. ఆ దేవసభ గురువుగారి నాట్యప్రయోగముచేత ఆరాధింప బడినదా? సంతోష పెట్టబడినదా?

**ద్వితీయుడు:-** గాలవుడా? ఆరాధింపబడినదో లేదో నాకు తెలియదు. కాని సాక్షాత్ సరస్వతీదేవి రచించిన, లక్ష్మీస్వయంవరము అను కావ్యబంధమునందు ఆయా రసపోషణమును చూచి, పభ తన్మయమైనది. కాని --

**ప్రథముడు :-** నీ వాక్యమును అపిన తీరు అచ్చట ఏదో దోషము జరిగినట్లు సూచించుచున్నది?

**ద్వితీయుడు :-** అవును. ఆ నాటకములో ఊర్వశి మాట ప్రమాదముపలన స్ఖలితము అయినది. (పారబడినది.)

**ప్రథముడు :-** ఎట్లు?

Figure 4.1: Sample document images from Kalidasa's books in Telugu.

and Abhijnanasakuntalam (The Recognition of Sakuntala). In addition to his plays, Kalidasa wrote two surviving epic poems Raghuvamsha (Dynasty of Raghu) and Kumarasambhava (Birth of Kumar Kartikeya), as well as the lyrical Meghaduta (Cloud Messenger) and Ritusamhara (The Exposition on the Seasons). All the writings have been translated into many Indian languages. Sample pages and words (in Telugu language) from the Kalidasa collection are shown in Figure 4.1 and Figure 4.2.

Such collections can be made available to large communities through electronic media. There is a need for easy and efficient access to such collections. The search procedures available for text domain can be applied, if these document images are converted into

textual representations using recognizers. However, it is an infeasible solution due to the unavailability of efficient and robust OCRs for Indian languages [84].

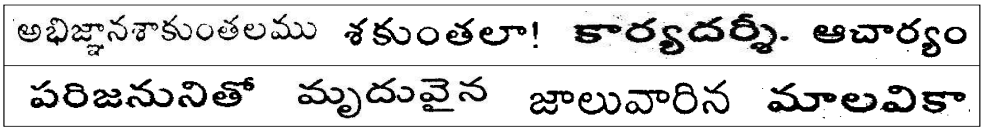



Figure 4.2: Examples word images, selected from Kalidasa's collection of books.

### 4.1.3 Problem statement and contribution

The size of the information available on the world wide web is growing tremendously fast. Most of the information is in the form of text, images and video. Present day search engines are designed for search and retrieval in text databases. However, image databases are also growing especially with images of printed and handwritten historical documents. Access techniques to such databases are studied by many researchers across the world on different languages. Methods of accessing document images presented in the literature are good enough for small data sizes. The access times are also acceptable for such datasets. With the fast increase in data over the WWW, digital libraries and similar sources, these methods are not scalable. Hence, there is a need for techniques for efficient search in large collections of document images.

This chapter describes a solution to the problem associated with the implementation of a scalable system for Indian language document images. A conceptual block diagram of our prototype system is shown in Figure 4.3. Our system accepts a textual query from users. The textual query is first converted to an image by rendering, features are extracted from these images and then a search is carried out for retrieval of relevant documents. Results of the search are pages from document image collections containing the retrieved words sorted based on their relevance to the query.

This 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.

## 4.2 Efficient search in large collections

We present an efficient mechanism for indexing and retrieval in large document image collections. Access to documents is possible only if they are represented as words, which are meaningful units of the content. The words are extracted from the document through a technique called segmentation. 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. It captures the structure and content of the word images through features and searches similar words efficiently with the help of hashing technique.

### 4.2.1 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 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 4.3. 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.

#### Hashing technique

Retrieval of document images, with words represented in terms of features, requires finding words with similar feature vector representation as the queries. A technique that computes

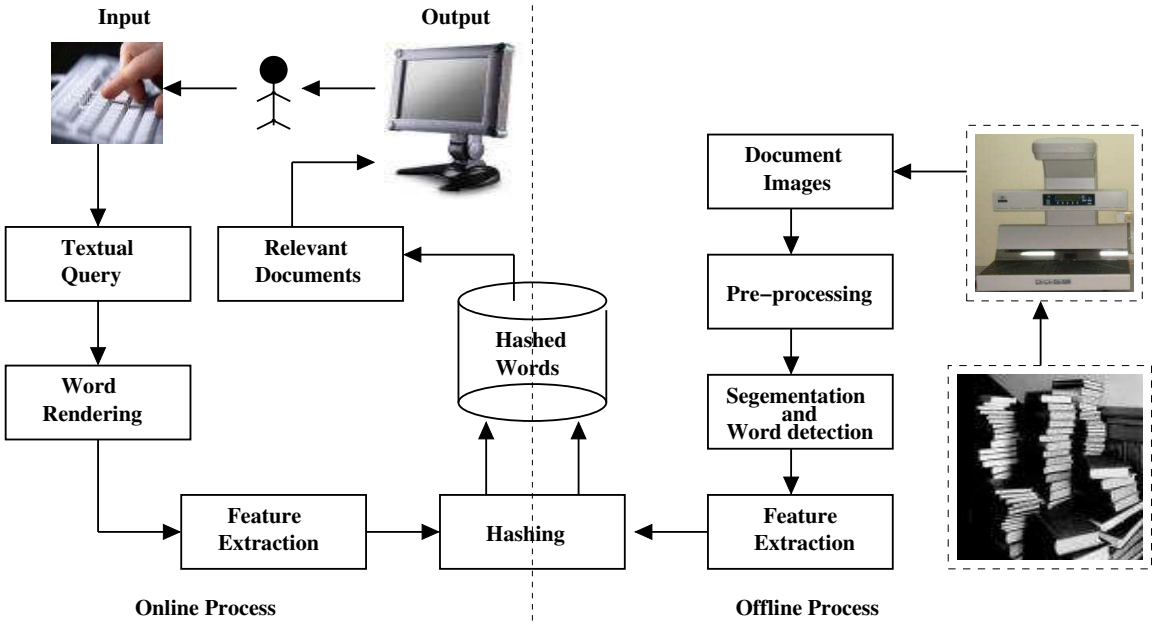



Figure 4.3: Conceptual Diagram of the Efficient Searching Procedure from Document Image Database. Showing offline preprocessing and on-line query processing stages.

simple distance measures, like Euclidean, can be applied to search the entire database for similar words. The exhaustive search method may not be applied to large databases. Since a query may comprise of few words to a sentence of many words, searching the complete database is time expensive process. This becomes even inefficient in the case of large collection of words. Also, the representation of two similar words may not have exact same feature values. Therefore, searching for similar words can be modeled as a nearest neighbor search problem.

In nearest neighbor search, given a query $q$, it is required to report a point in data set $P$ that is closest to $q$. In the case of word search using a feature representation, the requirement is to search for words with features approximately matching to that of the query word feature. Therefore, it is sufficient to report all feature points within a distance of at most $R$ from query $q$.

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.

With word image representations available, finding similar word images is now equivalent to the nearest neighbor search (NNS) problem: Given a set $P$ of $n$ points 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 [50] 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 [44, 50].

In order to find similar words, we can hash points from $P$ into some domain $U$, and then at the query time compute the hash function of $q$ and consider the points with which $q$ collides. That is, consider points $q$, $u$, $v$ with $v$ in a ball of radius $R$ centered at $q$, denoted by $v \in B(q, R)$ and $u \notin B(q, R)$. Then, we have that the probability $p(\parallel q - v \parallel) > p(\parallel q - u \parallel)$.

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

$$p(t) = Pr_H[h(q) = h(x) : ||q - v|| = t] \tag{4.1}$$

is strictly decreasing with $t$. 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 $G = \{g : S \rightarrow U^k\}$ 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 $T_i, i = 1, \ldots, l$

```
for each i = 1, ..., l do
    Initialize hash table T_i with hash functions g_i
end for
for each i = 1, ..., l do
    for each j = 1, ..., n do
        Pre-process word image W_j (noise removal etc).
        Extract features F_j of word image W_j.
        Compute hash bucket I = g_i(F_j)
        Store word image W_j on bucket I of hash table T_i
    end for
end for
```

### Hash function family

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 [111] 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,

$$h_{a,b}(x) = \left\lfloor \frac{a \cdot x + b}{w} \right\rfloor \tag{4.2}$$

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 optimal value for $w$ depends on the data set and the query points. As suggested in [29] $w = 4$ provides good results. Algorithm 1 summarizes

the major steps of content-sensitive hashing.

**Searching the query**

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.

**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: $O \leftarrow O \cup \{\text{points found in index } I \text{ of } T_i\}$
7: **end for**
8: Return similar words $O$ by linear search.

## 4.3 Hashing parameters and analysis

In the hashing scheme, the goal is that the nearest neighbor is reported with a probability at least $1 - \delta$. To achieve this probability of finding nearest neighbors efficiently, we need to specify the parameters $k$ and $L$. Let $p_1 = p(1) = p(R)$ and consider a query point $q$ with near neighbor $v \in B(q, R)$. Then, $Pr_{g \in G}[g(q) = g(v)] \geq p_1^k$. Thus, $q$ and $v$ fail to collide for all $L$ functions $g_i$ with probability at most $(1 - p_1^k)^L$. Requiring that the point q collides with $v$ on some function $g_i$ is equivalent to the inequality,

$$1 - (1 - p_1^k)^L \geq 1 - \delta \Rightarrow L \geq \frac{\log \delta}{\log(1 - p_1^k)} \tag{4.3}$$

The value of $k$ is chosen as a function of the data set to minimize the query time. The query time is decomposed into two terms for fixed values of $k$ and $L$. The first term is

$T_g = O(dkL)$ for computing the $L$ functions $g_i$ for the query point $q$ as well as retrieving the buckets $g_i(q)$ from hash tables. The second term is $T_c = O(dn_c)$ for computing the distance to all points encountered in the retrieved buckets. $n_c$ is the number of points encountered in the buckets $g_1(q), \ldots g_L(q)$.

Higher values of $k$ magnify the gap between the collision probabilities of close and far points, which (for proper values of L) decreases the probability of collision of far points. Hence, $T_c$ decreases as a function of $k$. $T_g$ increases as a function of $k$. Thus, typically there exists an optimal value of $k$ that minimizes the sum $T_g + T_c$ (for a given query point $q$). There might be different optimal $k$'s for different query points, therefore the goal would be optimize the mean query time for all query points. The value of $k$ is determined by estimating the times on a sample data set $S \subseteq P$.

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

## 4.4 Cross-lingual search

Many historical works of Indian origin are available in multiple languages. These documents have become part of the digital libraries like the Digital Library of India. Access to these collection is available only through the meta information created during digitization. A search system created for document collections of one language can not be searched using another language. Consider the case of searching collections over the Internet with the help of web browsers. The browsers do not support all the encodings and there are no standard keyboard layouts for Indian languages. In such situations, text typed in English or any other language encoding has to be used for the search. However, to search in the collections the language of the query and documents should be the same. To support the query in other languages, we make use of cross-lingual search.

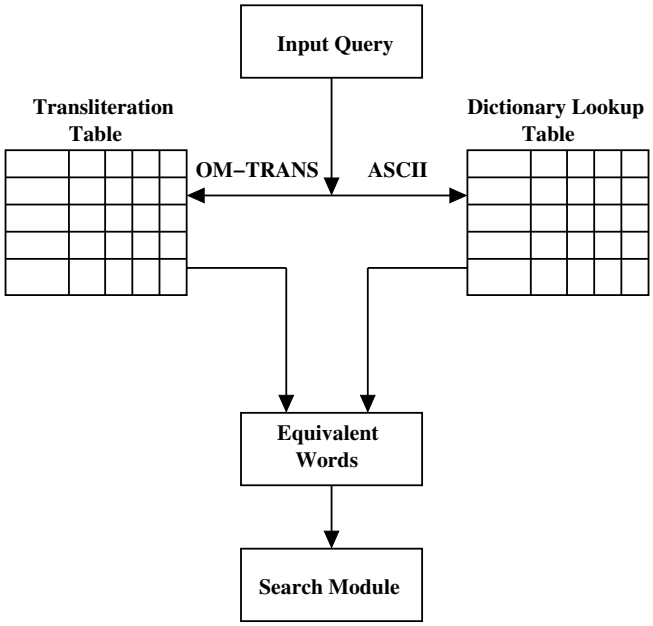



Figure 4.4: Cross lingual search in document images. Transliteration and dictionary based approach. A Conceptual diagram that shows document searching in multiple languages using transliteration and dictionary based approach.

Kalidasa books are also part of such collections which are translated to many Indian languages. There is need to design a mechanism that allows users to search all documents related to their queries in any of the Indian languages. As shown in figure 4.4 this can be achieved in two ways: transliteration and dictionary-based approaches. Since Indian scripts share a common alphabet (derived from *Brahmi*), we can transliterate the words across languages.

| Alphabet | a | aa | i | ii | u | ..... |
|---|---|---|---|---|---|---|
| Hindi | अ | आ | इ | ई | उ | ..... |
| Telugu | అ | ఆ | ఇ | ఈ | ఉ | ..... |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | |

Figure 4.5: Cross lingual search in document images. Building transliteration map. Sample Entries of the Transliteration Map Built for Cross-lingual Retrieval in English, Devanagari and Telugu.

In the OM-Trans scheme [83], there is a Roman equivalent for all the basic Indian

language characters. Figure 4.5 shows a sample transliteration map built for this purpose. For Example, "Bhim" can be typed in its Roman equivalent using OM-trans as "Bhima". Then the transliteration table is looked up for searching in Hindi (*Devanagari* script) and Telugu pages. Their cumulative result is finally displayed back to the user.

In dictionary-based translation approach, every English word has an equivalent word in the corresponding Indian and other oriental languages. If a user queries for a English word, the dictionary lookup points to a corresponding word in Telugu(say), an Indian language, for searching relevant words across languages. This table is also extended for other Indian languages.

The search on different language document image collections are carried out by creating separate hash tables. The cross-lingual conversion is required only for generating the query words. We employ the transliteration scheme to generate the queries for searching in collections of Hindi and Telugu language document images. Rest of the search procedure is language specific.

## 4.5 Experiments and results

Generation of real data is a challenging task involving a number of difficulties. Identification of appropriate source for proper demonstration is one of such issues. We selected a collection of 7 books of Kalidasa in Telugu language for this purpose. The collection consists of three famous plays and two surviving epic poems written by the great Indian *Sanskrit* writer Kalidasa. We also used a data set of generated word images of Telugu language for initial testing of the proposed search method. Some characteristics and details of the data sets used in the experiments are provided below.

**Dataset TG:** Telugu word images were generated from text data of around 6000 words. The images were generated in single font with varying image sizes. The image heights varied from 35 to 75 pixels. The word content repeated from 4 to 10 times for each word in the generated image collection (see Figure 4.6).

**Dataset TR:** Segmented words from document images. The document images were obtained from 7 Kalidasa books. Number of words ranged, in each book, from 11,000 to 56,000. The word images were in only one font for poetry books. The prose books consisted

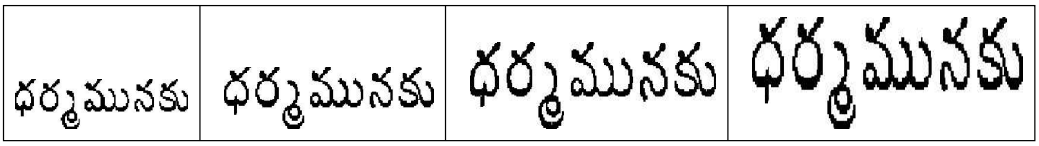


Figure 4.6: Data sets (Telugu): Sample word images from the Telugu data set TG_Data. Images are generated in a single font.

of words in two font types. This is the primary data set on which the performance of the proposed methods is tested.

**Dataset HR:** A Hindi book from Kalidasa collection was selected for evaluating the performance of the proposed technique on other Indian language document images. The book, *Vikramaditya*, was printed in single font and has around 25000 words.

The document images were preprocessed to eliminate any noise or degradations. Any accidental skew introduced during scanning process is also removed from the images. Some of the image processing operations and other algorithms like, segmentation require binary images to operate on. We binarize the images using standard image processing tools to obtain good quality binary images. The images are then segmented using standard document segmentation technique. The segmentation is an automatic process to identify the words, lines and other components in the document image. Most of the segmentation techniques perform well on the selected collection, as the collection has mostly plain text without any complex layout structure. The segmented words are used for feature extraction and then indexing for search and retrieval.

### 4.5.1 Search performance in Kalidasa collection

The words obtained from each book in TR and other data sets, after segmentation, were annotated for experimentation and performance evaluation purpose. The features extracted from the words are indexed using the hashing method. The features extracted from the query images are also hashed to retrieve the similar words. Textual query can also be supported by generating images from the text and then extracting features for hashing and retrieval. The book-wise performance measured using precision, recall and F-score values on the data set TR are shown in Table 4.1. Some experiments were conducted on data set TG before applying on the large collection. The search results were good on the data set TG (last row of Table 4.1).

| Book / Data set | # 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 |
| | | | | | |
| Data set TG | – | 6000 | 94.00 | 93.00 | 93.49 |
| | | | | | |
| Vikramaditya (Hindi) | 146 | 24622 | 92.23 | 90.36 | 91.29 |

Table 4.1: Search Performance: Precision, recall and F-score values for retrieval experiments conducted on each book from Kalidasa collection.

The query image and example search results are shown in Figure 4.7. The first two rows show results in which correct matching words are retrieved. Sometimes other words that are somewhat visually similar may also appear in the search results. The last two words of third row look very similar with only first character being different. But their meaning is different. The last column in the last two rows of the figure shows examples of such words being retrieved.

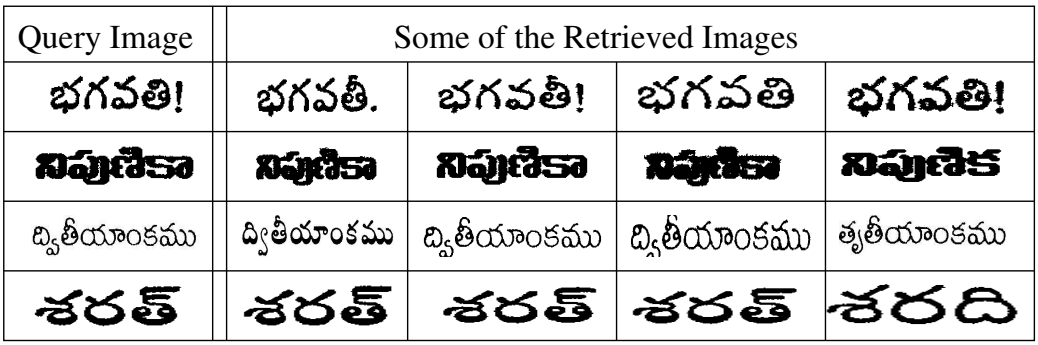



Figure 4.7: Results: Example (Telugu) words searched for input queries.

Searching efficiently in a collection of books is made possible by hashing each books separately. All books can not be hashed together if the fonts in them are different. The main problem with heterogeneous font image collection is in representation. The representation of word features change a lot due to variation in font and style. The hashing system can

handle only very small changes in font and style. Different words appear in the query results due to improper representations obtained with different font images. As a result the performance of the retrieval system goes down. However, search in multiple books can be done with a small change in hashing. We hash and query books that differ a lot in font style separately. A word that appears in more than one book is queried by making small changes in the query procedure. The query images are generated in different fonts for search in hash tables and based on font properties the query is fired. Examples of queries containing words of different sizes and style types are shown in Figure 4.8. Using the same query across two different books of the collection retrieves words which are content-wise similar.

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

Figure 4.8: Results: Words with small variations in style and size are retrieved.

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 4.9 (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.

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

Figure 4.9: Results: Words with small form variations are retrieved as relevant.

The proposed CSH technique was tested on Hindi language document images also (data

set HR). The Kalidasa's book *Vikramaditya* was selected in this experiments. The images were segmented to get words and features extracted in the same way as done with the Telugu language collection. The features appeared to be effective for Hindi language documents also. The results of the search experiment on this books are presented in Figure 4.10.

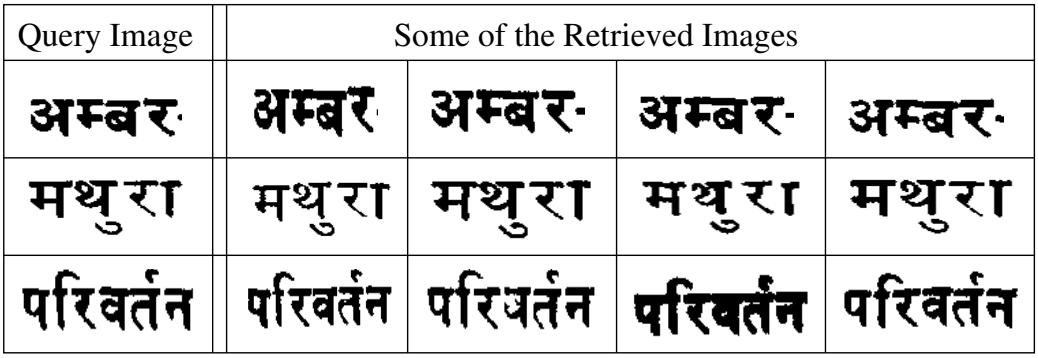



Figure 4.10: Results: Examples of results obtained by searching in Hindi document images.

The book selected in Hindi was also present in the Telugu collection. These two books are used for checking the cross-lingual search method. Some of the example results of the search are shown in Figure 4.11. To search in two different languages using single textual query is a little difficult. We followed the model explained in section 4.4 for this search. The major task in this search is the generation of images of query in two languages, each in a correct font. We annotated the words from both the books to select query words and measure the retrieval performance. If the font is not available, these annotation can be used to get query word images from the document images.

### 4.5.2 Time efficiency of hashing

The proposed hashed based search is sub-linear and much faster than exhaustive nearest neighbor search. The plot in Figure 4.12 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.

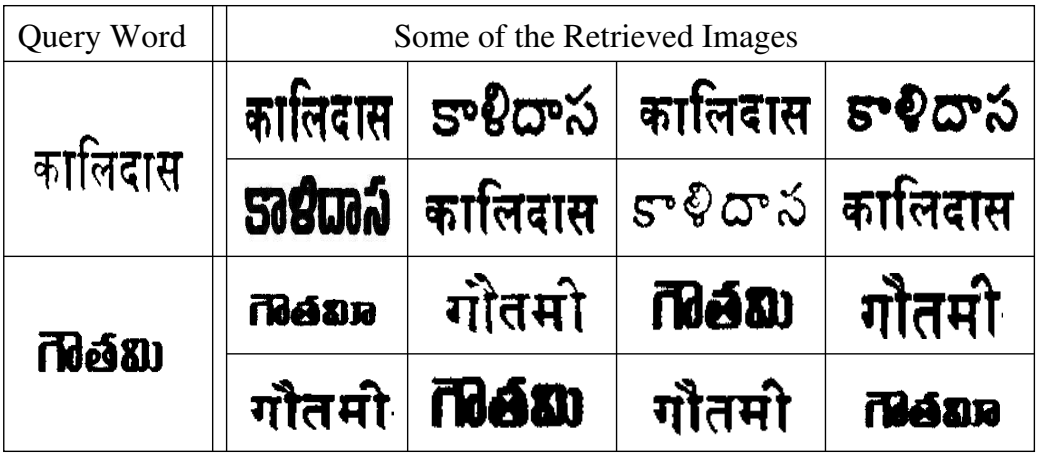



Figure 4.11: Results: Examples of words obtained by cross-lingual search in the collection.

### 4.5.3 Effect of query distance

The precision and recall values are controlled by the query radius (distance) value. Experiments were conducted on the synthetic images data set TG in the Telugu language to see the effect of radius on the performance. Around 6,000 synthetic word images of Telugu language were used for the experiments. Figure 4.13 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.

### 4.5.4 Comparison with DTW based search

Performance of the proposed hash based search technique is compared with the dynamic time warping (DTW) based word matching and retrieval technique [90, 91]. The features obtained from query images are matched with every other word of the collection for similarity search. The best matching words are retrieved as the search results. Table 4.2 compares our 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.

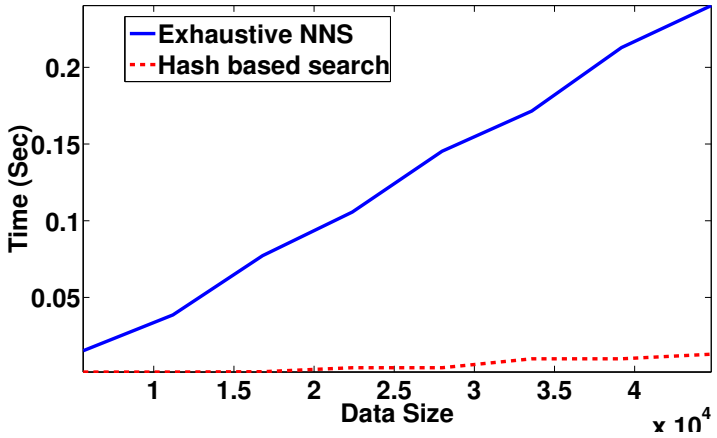



Figure 4.12: Performance comparison: Effect of data size (number of words) on Hashing and exhaustive nearest neighbor search.

| 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 |

Table 4.2: Performance: DTW based exhaustive search is much slower. Accuracy of the proposed method similar to the DTW matching.

## 4.6 Summary and Comments

The mechanism for search in document images is efficient and scalable to a large collection. A number of experiments were conducted on different data sets of English, Hindi and Telugu language document images. The index building time is linear and the data is scanned only once. Search for relevant words in the documents is very efficient. Search times of the proposed mechanism are better than other techniques available in literature. Table 4.2 shows the significant improvement in search by using the proposed method of search.

A mechanism is also presented to search in document image collections of different languages. A single query given in a specific format is converted into queries of different languages and the search is done on individual collection. Thus, cross-lingual retrieval is made possible. Search in a collection allows us to group together similar word images for a given query thus, generating a cluster, images of which can be annotated collectively. Annotation of document images can be carried out efficiently with the help of this method

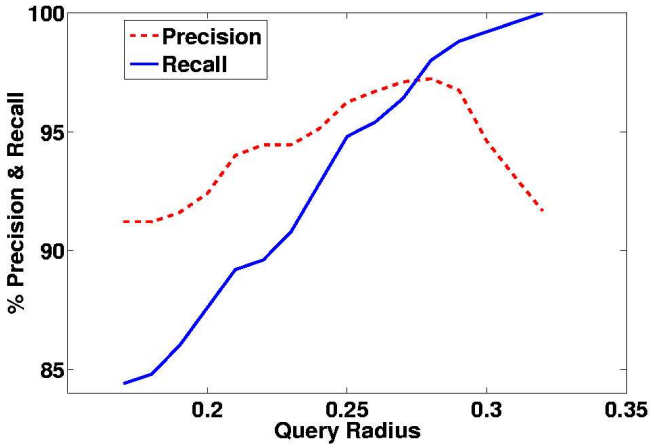



Figure 4.13: Effect of distance: Precision and recall change with the query radius.

and using a recognizer to generate the annotation text. In the next chapter we explore the method of annotation document images for retrieval. We also analyze the accuracy of annotation and its improvement with the use of grouping technique.

# Chapter 5

# Word Annotation for Retrieval

There are a number of search techniques available in text search and information retrieval. These techniques are robust and efficient for retrieval from very large collections of text documents. The process of index building and search are very simple and efficient. These methods can not be applied to documents that are available in image form. Conversion of these document images into text for efficient search, is not possible due to unavailability of robust recognizers [84]. However, the speed of search in document images can be improved if text annotations are available for the word images. The text words may be used for searching in the index and the image information for retrieving the corresponding document pages from the collection.

The proposed hash based efficient search in document image collections can also be used to annotate groups of similar words. The annotation information is a basic need for text based search in the document images. In this chapter we present a procedure for annotation of word groups from document images for building a text index. The search time is reduced significantly with the use of a text index.

## 5.1 Annotation of document images

Annotation is the process of identifying objects in images and labeling them with textual description that explains the elements of the objects precisely. Annotated data sets of document images have become important for research in document analysis systems [82]. Annotated data sets are a pre-requisite for a number of problems related to segmentation, recognition and retrieval algorithms.

A large number of document image collections in different languages are available over the Internet. Digital libraries (DLs) contain such information from different sources like, historical documents. The rapid growth of content and unavailability of robust recognizers poses many new challenges for easy access and for document image analysis research and development [9, 81]. Often, such collections are accessed with the help of meta data level annotations. However, content level access is not possible due to limitations of the annotations. Content level annotated datasets are prerequisites for the development, evaluation, performance enhancement and benchmarking of data driven document analysis and retrieval systems [52]. Lack of linguistic resources in the form of annotated datasets has been one of the hurdles in building robust document understanding systems for Indian languages.

The generation of large database of annotated document images involves well structured processing, labeling procedures and data storage for easy access and use. Documents need to be annotated at the structural, functional and content-level for building a variety of document understanding systems. Hence, content-level annotation is critical for developing OCRs. However, in this chapter we look at the task of annotation for the purpose of retrieval. We explore how the data can be automatically annotated, so that text labels can be assigned to the words. This helps in making use of text search systems for even more efficient retrieval in document image collections. This eliminates the step of searching in large databases of high-dimensional features vectors that represent the document images.

## 5.2 Auto annotation: State of the art

Annotation is a way to assign keywords to objects in an image. The assignment can be manual or automatic. The manual methods are costly and time inefficient. In automatic annotation, an image is matched with keywords models and the most relevant words are assigned to the image. There have been a number of traditional methods for annotation. A new technique of annotation called, reverse annotation, has emerged into the field. We discuss these two methods in brief.

### 5.2.1 Traditional methods

In automatic annotation, a model of the keywords that occurs in the collection is learned. A set of training images, that are already annotated, are selected. The image is segmented into different regions and features are extracted from them. Then, a relationship between

the features and annotations is learned from the training data set. The relationship model could be a *co-occurrence* model, a *translational* model of probabilities [35], a generative probabilistic *cross media relevance* model [54], statistical linguistic indexing [67] or a *latent semantic* model [77]. The choice of model depends on the ambiguity level in the annotations. Such methods are also applied to other type of image databases. The database of Corel is one such example. News photographs with caption or hand annotated data sets or any web pages that have some surrounding text can be chosen as the training data set.

The keyword models learned using the feature space of the annotated training images is used for annotation of the test images. To annotate the test image, it is segmented into regions and features are extracted from these regions. The features are compared against the keyword models and the labels are assigned using a suitable classifier, such as nearest neighbor. Rath *et al.* [92] successfully demonstrated annotation propagation across images using a collection 1000 handwritten document images.

### 5.2.2 Reverse annotation

In any retrieval system, the number of images to be annotated is much higher than the items to be indexed. With careful selection of the keywords for annotation, a large percentage of the documents can be indexed. In traditional annotation, keywords are identified for a given image. This is like a *classification* problem. Where as in reverse annotation, the relevant images are identified for each keyword. This is similar to the process of *verification*, in which we are trying to check if the keyword is the appropriate match for the given image.

In reverse annotation, the number of keywords is much less compared to the number of images to be annotated. An image of every keyword is generated to match with the words in the document images. When a keyword matches with a word, it is labeled with the keyword. Other keywords need not be matched with this word. This reduces the annotation time. If a cluster of word images of same type are available, the annotation time can be reduced considerably. Sankar *et al.* [94] presented a probabilistic reverse annotation technique, in which the probability that each cluster of images belongs to a keyword is estimated. This is done for each region of the segmented image. Based on the cumulative probability of a cluster belonging to a keyword the annotation label is determined. The annotation framework is demonstrated on a collection of 500 books from the Telugu language.

## 5.3 Applications of annotation

Large collections of images and other multimedia content are now available and shared online. Such data can be accessed easily if annotation are available. Some of the application of annotations are discussed in the following subsections.

### 5.3.1 Annotation for archival

Annotation for archival is one of the major application areas for annotation. Digital libraries maintain a huge number of documents. The documents thus are not only to be archived based on their content but in fact for quick indexing must be archived at the meta content level. Meta information could be, the genre of the document, its author, publisher, ISBN number if it is a digitized book, year of publication, edition etc. One needs to annotate information at the meta level, the ontology information pertaining to that document or sets of documents that share a common meta annotation.

Most of the times users search digital libraries with information which is stored in the meta information. The search through a huge database is made simple by storing meta information through annotation. This is an ideal way of indexing and archiving documents for much more efficient search in digital library application. The Digital Library of India [81] uses a similar mechanism to produce meta information during its archival activity where librarians manually enter/annotate meta information.

This sort of annotation is not limited to only digital libraries. People have been using this archival infrastructure for video and other multimedia data. The creation of meta information can be manual or automatic. Automatic annotation can be generated using OCR. Typically document images are converted to text using an OCR, and this in turn can be used as a meta information or an automatic form of annotation for archival. The challenge of proper retrieval become more difficult for archives containing documents and data that are semantically mixed and search terms are ambiguous. This can be solved by adding more fields in the meta information section that can reduce the ambiguity.

Annotation has been explored in the field of Biometrics, Medical Imaging, Multimedia databases and Forensic applications. Annotation information is extremely helpful in situations when the amount of data runs typically into large number, and retrieving becomes a tedious and time consuming task.

### 5.3.2 Annotation for retrieval

Traditionally, libraries have used manual image annotation for indexing and retrieval for image collections. There have been automatic image annotation and retrieval methods [54] that have been successfully used. Rath *et al.* [92] built a search engine for images of historical manuscripts. The retrieval system was trained using an annotated set of 100 pages of George Washington's manuscripts and is used to query a dataset containing images from the same collection. The automatic approaches to searching and accessing the content are very attractive. In such approaches, the manually created meta information is used for searching large collections efficiently. Another approach is to use recognizers followed by a text search engine. However, in real time the documents, especially the historical documents, are of poor quality which makes the recognizers vulnerable to poor results. In [92] alternative approach is used, bypassing explicit recognition.

In general, annotation based retrieval techniques have been widely applied for many multimedia data. Laverko *et al.* [64] used annotation for retrieval of images to video data. The video data primarily consisted of news videos, which needed to be annotated. After the annotation process, similar approaches of using text based retrieval methods were applied as described earlier. A method similar to searching in handwriting documents is applied to retrieve printed document images in [53]. In this case, the documents were chosen from the Digital Library of India [81], and were subjected to an off-line activity that pre-processed these images, segmented them into word images, and their features were extracted. Then all these segmented word images were clustered based on their similarity, and the clustered word images were annotated manually giving it a text representation and thereby indexing the whole document image collection for the printed texts. We follow the similar approach of annotation but, with an effective reduction in offline processing time of cluster generation [60].

### 5.3.3 Annotation for recognition

Annotation has been used widely for archival and retrieval purposes. Availability of large collections of annotated data also plays a vital role in recognition based techniques. The annotation data available is called ground truth. Ground truth data is used for training and testing the classifiers in OCR applications. The ground truth data consists of data for all classes of the classifier that is used for training. A large amount of data is necessary

for building OCR with degradations and performance evaluation of the recognizers, using confusion matrices and similar methods.

The ground truth data can generated in many ways providing annotation data with details of different level granularity. The annotation data comes with the document images that are used for OCR development. During the annotation phase, different level of hierarchies can be generated in the data set. That is, we can have corresponding text associated at the page level, the paragraph level, the sentence level, the word level, and up until the character or stroke level. Typically this annotation information is also very useful for segmentation based routines that can also build up on their segmentation results so that they can further accuracies.

Annotation has widely been used in the area of speech recognition [20] as well. During the training phase various phonetic sounds are tagged using various tagging techniques that enhance the recognition of a speech recognition system. Sphinx [87] is a popular speech recognition engine from the Carnegie Mellon University, while Festival [104] from University Edinburgh is a popular speech synthesis system.

## 5.4 Overview of annotation based retrieval

Access to document images is made efficient with the help of annotations. The annotations can be used for efficient access to documents by applying a text search engine. Since the text search methods work on words, it is necessary to annotate the words in document images. The document images can be annotated at different hierarchical levels [52]. The content level access is possible only if the word level annotations are available. Generation of word annotations for collections of document images is a tedious task. It is not feasible to carry annotations manually. One method is to use a recognizer to convert images into text and for generating the annotation. Annotation becomes challenging when the documents are degraded. Even the best recognizer fails to convert the all images into correct text. Therefore, searching in incorrect document text becomes difficult.

A number of similar words occur in a collection of document images. We can make use of this fact to improve the annotation using the recognizer. The idea is to group similar word images before recognition and annotation. Since all words in a group are similar, this information can be used to identify correct text word for the entire group. Once a correct word representing the group is identified, all word images of the group are annotated with

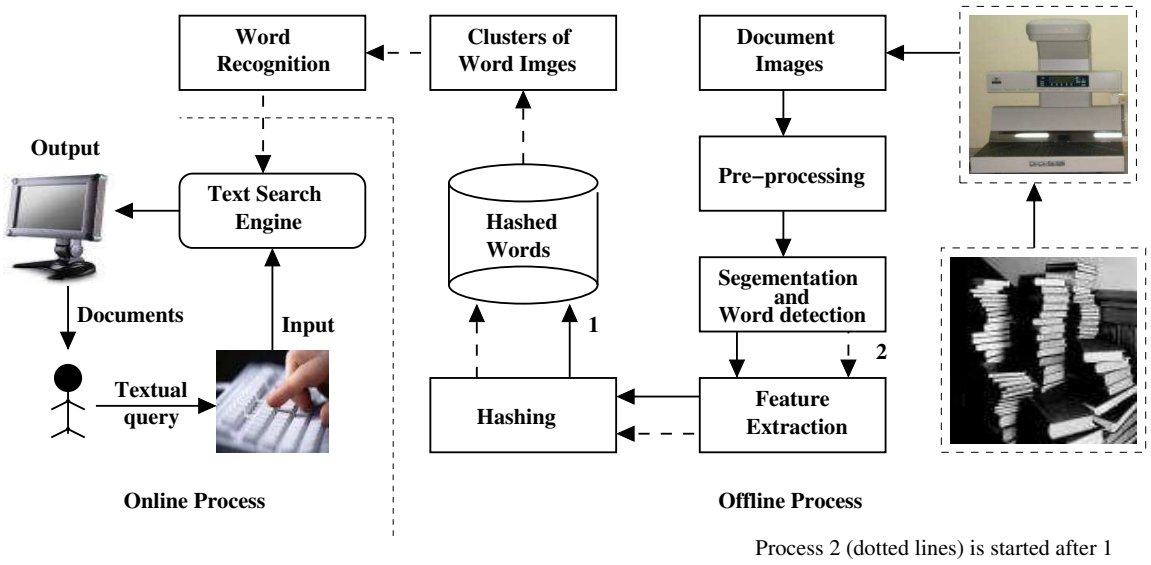



Figure 5.1: Annotation based search: Annotation of word images for efficient access to document image collections.

it. Then these annotations are used for text based search in the documents.

A block diagram of the proposed annotation based retrieval method is shown in Figure 5.1. The document images are preprocessed and then segmented to get words. An index of the word images is built using the content sensitive hashing method. Clusters of similar word images are generated by querying the index using all words of the collection. The results of every query are marked in the collection to eliminate any reoccurring queries (images with already queried content). Every word of the cluster is recognized to obtain text. These text clusters are processed further to generate representative words of the clusters. These representative words are used as annotations of the words. The annotation are used for search using text search methods. Any search engine can be used for indexing and retrieval of the document images. The rest of the chapter discusses the proposed annotation based retrieval technique.

## 5.5 Word image clustering

The proposed document level annotation scheme annotates words using information from a cluster of similar words. The annotation helps in the retrieval of document images using the text search engines. The document images are to be preprocessed and segmented at word

level to generate the clusters. The process of clustering word images using image matching techniques is time expensive. General clustering techniques can not be applied as the time required for cluster generation is high. The major challenge in efficient clustering is the computation of similarity measure for word matching. We overcome these challenges with the use of word hashing technique. We employ the time efficient content sensitive hashing [60, 29], technique to get a cluster for every word from the documents images. A similar technique is presented in chapter 4 for document image retrieval.

Clusters of similar words are obtained by querying the hash index with words from document images. Given a query word feature, the $L$ first level $d'$ dimensional hash codes are determined and all the words within the corresponding second-level hash tables are retrieved. The words obtained in a cluster are marked in the documents with unique cluster numbers. The query process is repeated for every unmarked word from the document images. Thus, all words from the document images are clustered into groups of similar words. Algorithm 3 shows the major steps in the process of generating word clusters.

**Algorithm 3** Word Image Clustering

**Input:** Word Images $W_j$ and Features $F_j, j = 1, \ldots, n$
**Output:** Word Image Clusters **O**

**for** each $i = 1, \ldots, l$ **do**
  **for** each $j = 1, \ldots, n$ **do**
    Compute hash bucket $I = g_i(F_j)$
    Store word image $W_j$ on bucket $I$ of hash table $T_i$
  **end for**
**end for**
$k = 1$
**for** each $i = 1, \ldots, n$ and $W_i$ unmarked **do**
  Query hash table for word $W_i$ to get cluster $O_k$
  Mark word $W_i$ with $k$
  $k = k + 1$
**end for**

The content of the words obtained in clusters differ very little. In other terms, the words with the same stem and little form variations are grouped together in single cluster. However, all word form variations are not grouped together. Words with same root but varying much in content are not grouped in one cluster. Words with extra character/symbols are also grouped if the stem content is same. The representation of such words produces similar features for words containing very little variations.

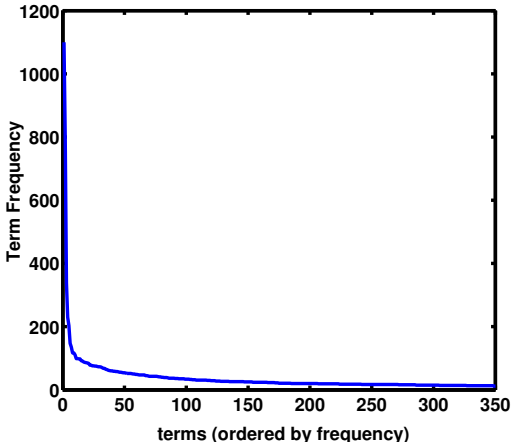



Figure 5.2: Frequency of words in a collection of document images.

Zipf predicted a distribution of term frequencies in a collection. A plot of term frequencies, where terms are ordered by decreasing frequency of occurrence, exhibits a distribution known as Zipf's law [110]. That is, the frequency of the $k$-th most frequent term has a frequency that is proportional to $f_0/k$, where $f_0$ is the frequency of the most frequent term. The terms occurring in the middle of the distribution should be taken for indexing. The number of words present in a collection of document images from a book are shown in Figure 5.2. It is observed, from the plot, that the actual distribution of term frequencies is similar to Zipfs predictions based on the actual frequency of the most frequent terms. Most of the terms in the collection are concentrated in high-frequency region and the long tail of the distribution to the right. In practice, it would be desirable to index all terms that occur in a collection, as many modern information retrieval systems do. However, most users are interested in terms that occur in the middle of the distribution.

Often stop words, in the left side of the plot, such as and/the/..., are high-frequency words. As these words occur in virtually all documents, they have no discriminating power. Terms with low frequencies are often not correlated to the topic of the text they occur in. Such terms may be removed from the index as they are not descriptive of the content in the collection. The repeated, but not excessive, use of the the terms that occur in the middle of the plot are descriptive of the content. The essential terms describing the content of the collection, should consequently be part of the index.

This method of annotating the word images is totally different from the document

annotation for document analysis systems. The goal of annotation of word clusters is to facilitate search in document images with the use of text. The annotated text is used for index building using a practical search engine. This makes the content level access to document images easier and more efficient.

## 5.6 Annotation of word clusters

The digitization technology has created an opportunity to save documents, as images, in electronic form. These documents are not accessible at content level, as there are no good methods that can provide the speed and precision as good as a text document retrieval system. To make this happen, the documents have to be converted into text with the help of a recognizer (optical character recognizer (OCR)). There are some recognizers for English language that can provide acceptable accuracies. But, for many other languages, there are no good OCRs available. Therefore, the task of editing these documents is not possible.

Most of the recognizers work at component level. A component is the smallest unit of connected pixels that are recognized in an OCR. The maximum number of components in English language characters are only two (namely, i and j). As the number of components increases, as in Indian languages, the complexity of the classifier increases and recognition becomes difficult. The complexity is further increased if there are degradations in the document images from which the character components are identified. Some of the degradation and their affects are:

- The cuts in character components may split a single component into multiple. These, many individual components are very hard to classify and recognize a character.
- Merging in multiple components may create such a component, which may not be part of the class of components that the recognizer is capable of recognizing.
- Noise and other kind of degradation affect the correct components. Sometimes these introduce components that are not part of the class. In other cases new unwanted characters are introduced due to the degradations. As an example if the component to be classified as “l” has very small dots due to noise around it, the classification may be “i” or a number “1” may be classified as character “l”.

The component level accuracies, even if high, affect the word level recognition accuracies

of the recognizers. Every component of the word is recognized and the results are concatenated to generate the word text. The word recognizer's accuracy is directly dependent on the component level accuracy, as they make use of the component recognizers. Therefore, the document level recognition accuracies of such systems are not acceptable in practical situations.

### 5.6.1 Word error correction

A number of words repeat several times in a collection of documents, say a book. Each of the repeating word image may have different properties such as size, appearance and quality due to degradations. As a result of such properties they form a special group of similar words. When these words are recognized, the text may have different results. The recognition results may have errors occurring at different positions in words. But in a group all the words are same and the text is expected to be same. The erroneous output of recognizers produces erroneous annotations of the words.

When a cluster of word images with the same content is available, their annotations should also be the same. If not, is there any possibility of making use of the cluster information to correct the errors? One way to correct the errors is to use the dictionary based correction methods. The dictionary based methods assume common spelling errors caused by doubling, undoubling or transposition of characters. As discussed above, the errors caused by the OCRs are not similar to the spelling errors. Now, we present two methods for correcting the recognition errors in the following subsections. The detailed analysis of the methods is presented in experimental section.

#### Correction by majority voting

Words present in a cluster are recognized to get text for annotating the cluster. There could be a number of errors in the words recognized. However, all words may not be recognized incorrectly. Some of the words will be correctly recognized and some with errors in one or two characters. Such words can be used to identify the correct word that represents the whole cluster. Hence, the problem is to find the representative word from words that have no errors or have errors in very few (one or two) characters.

The representative of the cluster can be identified by majority voting technique on the correct words. Let $W_i, i = 1, \ldots, n$ be the words present in cluster $C$ of the recognized words. We sort them to obtain a list of strings $S$ in decreasing order of their length. Now

**Algorithm 4** Word correction by majority voting

**Input:** Cluster **C** of words $W_i, i = 1, \ldots, n$
**Output:** Representative word $W_R$ for $C$

1: Find $S = \{$Sort $C$ based on string length$\}$
2: Obtain group $M \subseteq C$ such that $\forall A, B \in M$ edit distance between $A$ and $B$ is less than half of $minimum$(length of $A$, length of $B$)
3: Let $l$ be the maximum of lengths of strings in $M$
4: The length of $W_R$ is $l$
5: **for** each $i = 1, \ldots, l$ **do**
6: Get all $k$ words from $M$
7: Find majority of characters that stand for place $i$ of $W_R$
8: **end for**

we pick a group of words $M \subseteq C$ such that for any two words $A, B \in M$ number of correction to be made is less than half of their lengths. This subset $M$ of $k$ words, $M_i \in S, i = 1, \ldots k$ and $k \leq n$, is used to find the correct word to represent the cluster. When the majority of the $k$ words are similar then we make that word as the representative. Otherwise, the characters that need to be replaced are found by majority voting for that position. The algorithmic steps for the majority voting are shown in Algorithm 4. An illustration of the technique is shown in Figure 5.3

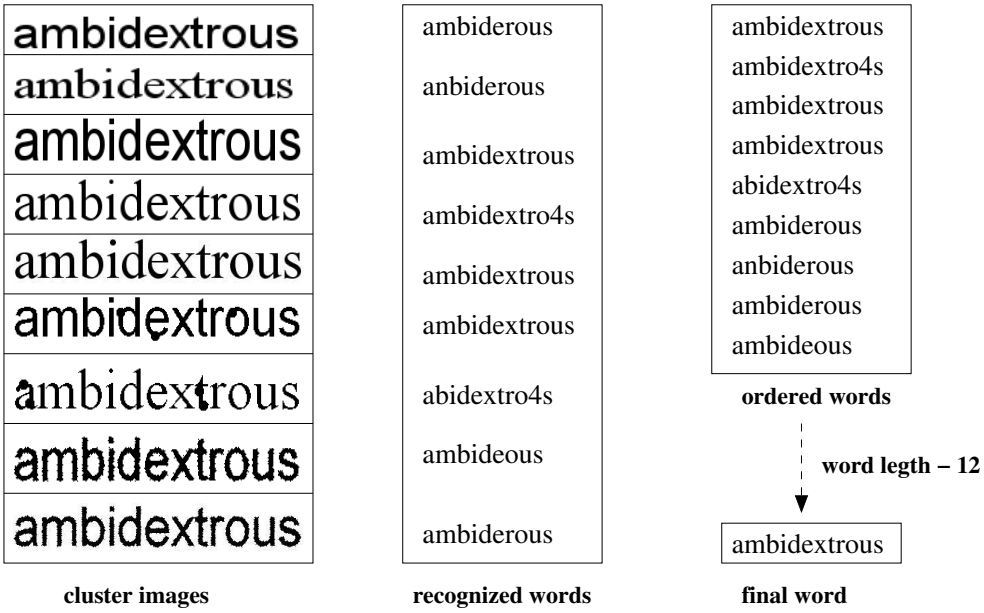



Figure 5.3: Word error correction (Majority voting): An example of correcting word errors using cluster information.

**Correction by string alignment**

Let $\mathbf{C}$ be a cluster of $n$ similar words obtained from the recognizer. Each word $w \in \mathbf{C}$ is aligned with other words $\mathbf{C} - w$ of the cluster. The alignment is obtained by a dynamic programming technique. All matching characters of two words are aligned and the unmatched characters are identified as possible errors in word $w$ of the recognizer. The unmatched characters are candidates to replace the erroneous characters in word $w$. Alignment of $w$ with all other words of the cluster gives a group $G_p, p = 1, \ldots, q$ of possible character replacements for errors. Hence, for every erroneous character $E_k, k = 1, \ldots m$ of $w$ there are a maximum of $n - 1$ possible corrections. The possible correction $C_h$ for $E_k$ is selected from $G_p$ by majority voting. A wrong character is replaced by a new character $C_h$ for which the count is maximum in $G$. The steps of this process are outlined in error correction algorithm 5. This procedure is repeated for every word of cluster $\mathbf{C}$ and correct words are obtained.

**Algorithm 5** Sequence alignment for word correction

**Input:** Cluster $\mathbf{C}$ of words $W_i, i = 1, \ldots, n$
**Output:** Clusters $\mathbf{O}$ of correct words

```
for each i = 1, ..., n do
    for each j = 1, ..., n do
        if j ≠ i then
            Align word W_i and W_j
            Record errors E_k, k = 1, ..., m in W_i
            Record possible corrections G_p, p = 1, ..., q for E_k from W_j
        end if
    end for
    Find correction C_h = G_p, if occurrences of G_p is maximum in G
    Correct E_k with C_h
    O ← O ∪ W_i
end for
```

The final word to be selected as representative is the one which occurs maximum number of times in the corrected cluster $O$. That is, the majority voting technique is applied on the words of the corrected cluster to find the representative word. The clusters obtained by the hashing method contain words mostly similar in content. Query parameters of hashing technique can be adjusted to cluster images with same content together. After the words are recognized and error corrections are made, the final word obtained will be same. A single word represents content of images from a cluster. Thus the final word is used for

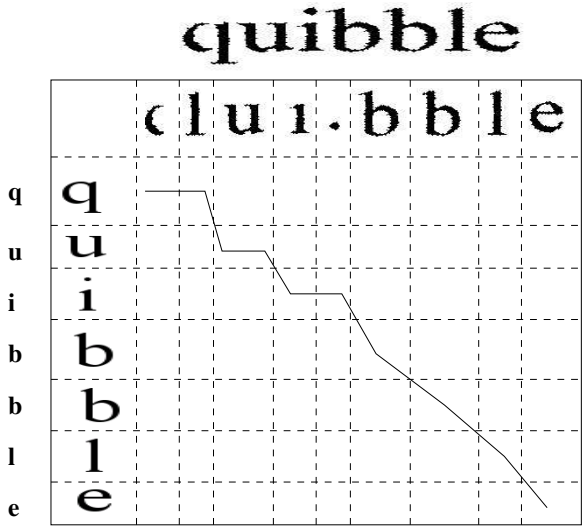



Figure 5.4: Figure (Matrix) Illustrating the word DTW. Horizontal continuous line segments show the components in columns grouped together to match with a character in a row. Continuous line segments across rows show the place of character boundaries.

annotation of clusters. The same word can be used for indexing and retrieval applications of document images.

The alignment of words in the algorithm is based on a dynamic programming technique [26]. It is similar to a dynamic programming method applied for the alignment of words to annotate characters in printed and handwriting documents. Given a word annotation, characters are annotated by matching character images with components of the word image [52] (Figure 5.4). Similarly, synthesized handwriting character strokes are matched with strokes of word to annotate handwriting at character level [59]. Our alignment method also attempts to find correct characters in two strings so that wrong ones can be corrected. it uses the edit distance [34] to find the alignment of characters. The alignment based method, unlike the majority voting method, tries to corrects errors present in every word of the cluster. An example of the string alignment based word error correction technique is shown Figure 5.5.

## 5.7 Application to retrieval

Annotation of any data set is important to the research and development of a technology. The annotations in document images are very useful for the development of document

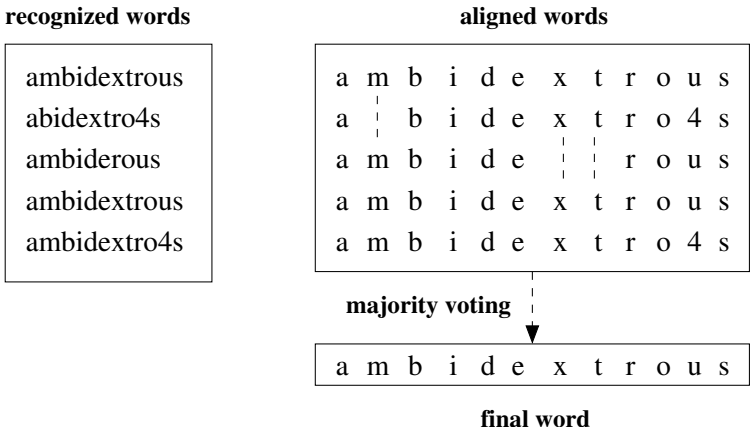



Figure 5.5: Word error correction (String alignment): An example of correcting word errors using cluster information.

analysis and retrieval systems. Annotations can be generated in a number of ways. The choice of each method depends on the type of application for which the data is to be generated. In this section we briefly discuss about the application of annotation for retrieval systems.

The annotation for document images are generated by clustering the words obtained from them. The clusters are annotated with a representative word from it. Then, these clusters are labeled with text. Manual annotation is preferred in this case. A recognizer, as explained above can be used to generate the annotations efficiently. However, traditional clustering techniques are not feasible in word image clustering. In such situations our method is the best way to generate clusters efficiently.

The annotations of document images stand as a base for building a very efficient retrieval system. The main idea is to use the text labels assigned to words of the documents, during annotation, as the index terms. A general text search engine can be used to generate the index structure. These index terms store location information of the word image in document images. When a term or word is searched, the index structure looks for the text word and then returns the location information of the actual word along with the document in the collection. This method of using a text search technique for image search is very efficient. It has a number of advantages.

- Complex image matching techniques are totally avoided.
- Text search is much faster than image search. Hence time efficient search is possible.

- The space utilization is very less as compared to the image search technique.
- Scalability is much more better than the search techniques for images.

The quality of search can be improved with text based methods. The text search engines make use of pruning, obtaining basic form of words. By pruning, word for variations can be eliminated from the recognized words that are used for indexing. The task of pruning in word images is tedious and not so accurate. Determining the word form variations involves partial matching of images. There could be errors in partial matching that can introduce unwanted words into the clusters. Hence, use of text search methods would eliminate some words by pruning, which would help in reducing size of the index. Thus annotation of word clusters can be used effectively for very efficient search in the document image collections.

## 5.8 Experiments and Results

The proposed word annotation techniques are tested on a number of different data sets of English language. The data sets and their properties are listed below. Experiments are conducted on each of the data sets. The words are clustered by content sensitive hashing (CSH) using word level features as representatives. The clusters obtained are used for recognition of words by applying the proposed technique.

- **Dataset EGA_Data:** Data set of around 2500 word images obtained from text words. The words vary in different sizes, quality/degradations etc. Some of the words were degraded using Kanungo's local degradation model [57] to make them more closer to real data. The images were grouped into around 100 manually annotated clusters for the experiments.
- **Dataset ARE_Data:** Word images obtained from document images of an English book. The document were preprocessed to remove degradation before segmentation and selection of the words. Data set contained around 3600 word images from the book. The images were obtained from 100 clusters that were annotated manually for the experiments.

| Method Applied | EGA_Data | | ARE_Data | |
|---|---|---|---|---|
| | WAc (%) | CAc (%) | WAc (%) | CAc (%) |
| Raw OCR | 32.47 | 69.59 | 82.71 | 93.78 |
| Majority voting | 58.65 | 92.92 | 83.26 | 94.31 |
| String alignment | 78.85 | 94.5 | 86.45 | 95.98 |

Table 5.1: Comparing accuracies of different methods on English language data sets. WAc = Word Accuracy, CAc = Character Accuracy

### 5.8.1 Word annotation results

The clustered word images are recognized with the help of a recognizer, which are then passed to the correction scheme. The corrected words are used as annotations of the clusters. The word level accuracies of the annotation techniques are presented in Table 5.1. Word clusters were annotated manually to measure and compare the performance of the method. It can be inferred that incorporation of the proposed correction methods in the annotation increases the performance significantly. Since, the proposed method corrects characters of recognized words, the accuracy of annotation is improved by the use of clusters.

Word images are generated to test the performance of the proposed technique. This data set EGA_Data consists of word images generated from text. The words can be generated from the given text using any image processing library like ImageMagick. The advantage in such generation is that the word segments are readily available for processing and the annotations are readily available. The word segments are processed to obtain features that are passed for hash based clustering. Figure 5.6 shows an example of such clustering obtained from the data set. It is observed that all of the words of a group are same.

The performance of majority voting technique is compared with the normal annotation method. In the majority voting method, the average length of the word is computed from the length of all words existing in the cluster. It is assumed that all words in the cluster are same. Once the word length is determined, the candidates for each character position are chosen. These candidates are chosen from the words of cluster based on number of characters that match the desired position. The results of majority voting are also presented in Table 5.1.

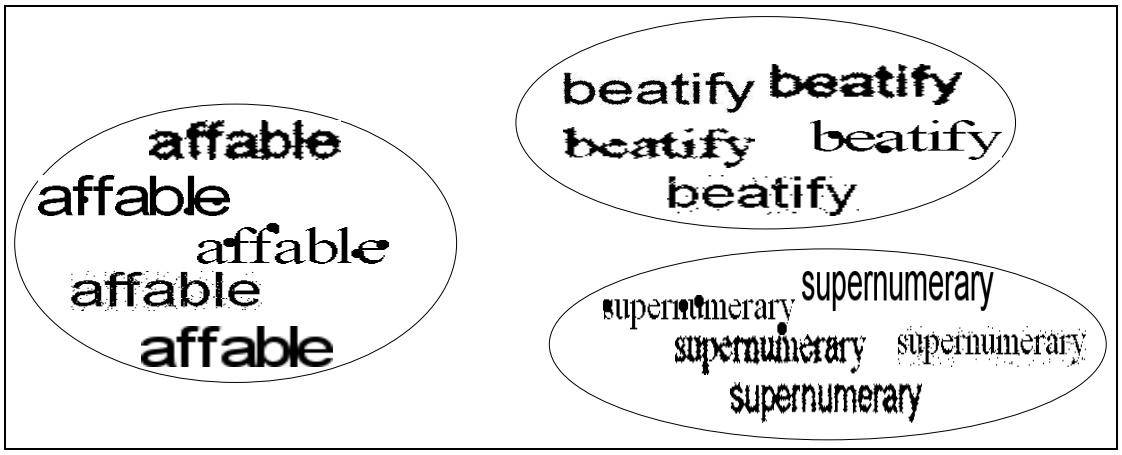



Figure 5.6: Example word images obtained by clustering using hashing method.

### 5.8.2 Search results

The performance of the search applied on the collection of words is measured on the corrected data set. We used the data EGA_Data for the experiments. Given a query word, the search is performed on the clusters obtained after correcting the errors in annotation. The complete cluster is presented as results of a query if there is a match. Occurrence of such representative words helps in retrieving all the words. Thus, producing 100% precision for almost all the queries.

An experiment is conducted to measure the effect of the number of words in a cluster on the recall of the proposed method. The result shown in Figure 5.7 is from the experiment conducted on the word images of data set EGA_Data. The effect of the number of words in a cluster on the recall of the search results is presented in the figure. It is observed that the number of words found correctly increases as the number of similar words in the cluster increase. The number of words in each cluster is varied from 2 to 20.

The precision and recall values of the search are bound by the performance of the hash based cluster generation process and the accuracy of annotation. If a cluster is wrongly annotated, the performance goes down drastically as all the words of that cluster will not be retrieved. Hence, performance of search in text is highly dependent on the accuracy of annotation/recognition of words. The major advantage of converting images to text is that the complex word image representation and the hash index can be avoided. Building text search engines is simple as compared to the indexes built for images. Also, the search is much more faster than the image domain search.

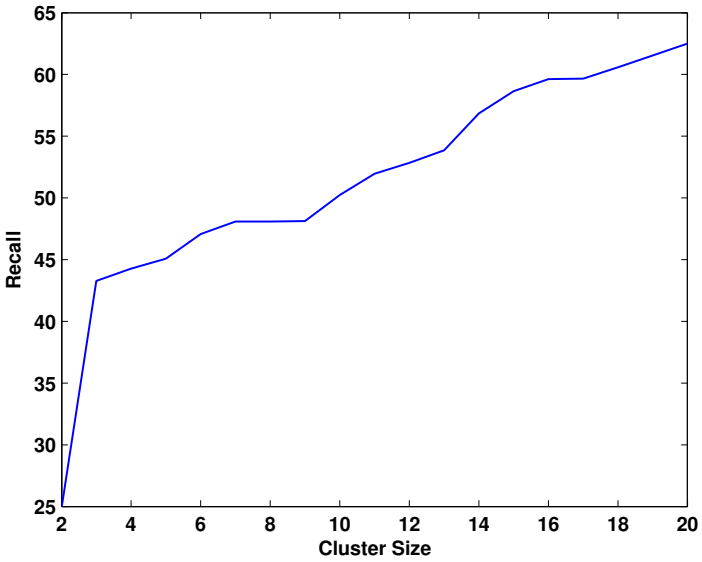



Figure 5.7: Effect of cluster size on the performance of search

## 5.9 Future Work

In this chapter we presented a mechanism for annotating words from document images. The proposed technique makes use of content sensitive hashing, which is based on locality sensitive hashing, to search in collection of words and obtain clusters of similar words. Clusters of similar words are generated efficiently by processing the words in linear time and searching in sub-linear time. Two word annotation techniques using clusters are also presented. The word annotation are further propagated down to characters to obtain annotation of characters. A dynamic programming based automatic character annotation technique was presented.

The hash based clustering technique produces clusters of words that are similar in content. The similarity is very close in all the words of a cluster. The words in a cluster may not vary considerably. That is, the word form variations are not handled by the clustering technique. As an example, the words “go”, “gone” and “going” may not fall into same cluster. Instead separate clusters are formed for words with content “gone” and “going”. Most of the text search engines take care of such word form variations and group such words together. This is a difficult task with the use of the proposed hashing technique. To accomplish this task, a possible direction is to look for different hash functions that can

consider the word form variations. Another possible direction is to look for functions that can learn from the data set.

The words are annotated by finding a representative of every cluster. The same annotation is propagated to all words of the cluster. This generalization of representative words as annotation for all words of the cluster may not be appropriate. Some times it is possible that two words, say "Tony" and "Tone" may fall into the same cluster due to very small variations in the representation (using features). Therefore, the words may be annotated incorrectly. The annotation may be propagated down and character annotation may also be wrong. This leads to wrong annotations. The wrong annotations would lead to time consuming and laborious process of manual verification of annotated data and correction. A method to detect such errors automatically would be very useful in speeding up of the annotation.

The accuracy of annotating words and then the characters depends on the accuracy of the recognizer used for obtaining text of the cluster words. If all the words in a cluster are recognized incorrectly, then the majority voting method fails completely. This may be due to frequent errors made by the recognizer for same character of every word. As an example, if the recognizer gives "h" as an output for "ff" in words then, words like "different" will be recognized as "diherent". Such errors can be rectified using a dictionary. However, as mentioned before, the dictionary based approaches also fail in achieving successful correction. The problem of identifying correct the representative of a cluster may not be difficult for English language, as there are may commercial recognizers that can provide good and acceptable recognition accuracies. The problem would be more serious for Indian languages and other non-European languages. Hence, the automatic annotation of document images at word and character level is an interesting and challenging problem to consider.

# Chapter 6

# Conclusions

## 6.1 Summary and conclusion

We have presented a mechanism for indexing and retrieval in large document image collections. A sub-linear time hashing technique, based on locality sensitive hashing, for search in large collections of document word images is applied to obtain good performance. The use of text search methods requires efficient and robust optical character recognizers(OCR), which are presently unavailable for Indian languages. The approaches used for word spotting so far, dynamic time warping and/or nearest neighbor search tend to be slow for large collection of books. Direct matching of images is inefficient due to the complexity of matching and thus impractical for large databases. The time required for building a clustering based index is very high. Such indexing methods are time inefficient with the use of complex image matching procedures in clustering step. This problem is solved by directly hashing word image representations.

The performance of the method is presented with the help of experiments on real data sets from Telugu language documents. The efficiency is demonstrated by conducting experiments on a collection of 7 Kalidasa's books in Telugu language. The feasibility of the proposed method is also tested on words from English language. It can be concluded from the experiments that, the method is efficient in retrieval from large document image collections in few milliseconds.

The efficient access to document images at word level allows us to group words with similar content. Thus, clusters of similar words are generated using the proposed retrieval

method. These clusters are labeled with text of the representative word images. The representatives are used for finding the annotation of the cluster. With this method the whole collection of word images is annotated. This method is called "*annotation based indexing*". An annotated corpus is critical to the development of robust OCRs. Manual annotation of document images is a labor and time intensive task, especially for large collections. Most of the time is consumed in the annotation of the textual content of document images. With the proposed annotation procedure, text content is annotated efficiently. With the application of the proposed automatic annotation procedure, the accuracy is improved significantly. These annotations can be used for efficient searching in document images by applying text search engine. Hence, search and retrieval of document image collection is made very efficient.

## 6.2 Future work

The method presented here works with documents printed in similar font styles. This method is also used for annotation based search in document images. Some of the possible directions in which the future work can be carried out are as below.

- The effect of combination of different fonts in a single collection can be one possible direction for exploring the feasibility of the proposed technique and improving it.
- Multi-lingual document images are not handled in the proposed technique. Many Indian language historical books are being translated to other languages of the world. The option of search in multi-lingual documents could be another interesting direction for the research.
- The clusters generated during the process of annotation, can be used effectively for improving the OCR performance. The clusters are helpful in training the classifier to improve the recognition performance.
- Document images with degradations of all types are challenging to recognize. This problem becomes even more challenging for non-English languages like, Asian languages. Other techniques of cluster annotation may have to be considered if the recognizers perform badly. In such documents, recognizers may make mistakes in every word of the cluster. Therefore, there is a need to explore new techniques of annotation.

# Bibliography


[1] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Efficient Similarity Search In Sequence Databases. In *FODO '93: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pages 69–84, London, UK, 1993. Springer-Verlag.

[2] J. F. Allen. *Natural Language Understanding*. Addison-Wesley Publishing, Reading Massachusetts, 1995.

[3] Sunil Arya and David M. Mount. Approximate Nearest Neighbor Queries in Fixed Dimensions. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 271–280, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[4] Sunil Arya, David M. Mount, and Onuttom Narayan. Accounting for Boundary Effects in Nearest Neighbor Searching. In *Proceedings of the eleventh annual Symposium on Computational Geometry (SCG)*, pages 336–344, New York, NY, USA, 1995. ACM.

[5] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[6] Esra Ataer and Pinar Duygulu. Retrieval of Ottoman documents. In *Multimedia Information Retrieval (MIR) workshop*, pages 155–162, 2006.

[7] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing, 1999.

[8] David Bainbridge, John Thompson, and Ian H. Witten. Assembling and Enriching Digital Library Collections. In *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS*

*joint conference on Digital libraries*, pages 323–334, Washington, DC, USA, 2003. IEEE Computer Society.

[9] Henry Baird. Digital Libraries and Document Image Analysis. In *Proc. 7th International Conference on Document Analysis and Recognition (ICDAR)*, pages 2–14, 2003.

[10] A. Balasubramanian, M. Meshesha, and C. V. Jawahar. Retrieval from Document Image Collections. In *International Workshop on Document Analysis Systems (DAS)*, pages 1–12, 2006.

[11] Anand Balasubramanian. *Document Annotation and Retreival Systems*. M.S. Thesis. International Institute of Information Technology (IIIT), Hyderabad, India, 2006.

[12] Rudolf Bayer and E. McCreight. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica*, 1(3):173–189, 1972.

[13] J. L. Bentley and M. I. Shamos. A problem in multivariate statistics: algorithm, data structure, and applications. In *Proceedings of the 15th Allerton Conference on Communication, Control and Computing*, pages 193–201, 1977.

[14] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[15] Jon Louis Bentley. Multidimensional Binary Search Trees in Database Applications. *IEEE Transactions on Software Engineering*, SE-5(4):333–340, July 1979.

[16] Jon Louis Bentley and D. F. Stanat. Analysis of Range Searches in Quad Trees. *Information Processing Letters*, 3(6):170–173, July 1975.

[17] Marshall Bern. Approximate Closest-Point Queries in High Dimensions. *Information Processing Letters*, 45(2):95–99, 1993.

[18] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When Is Nearest Neighbor Meaningful? In *ICDT '99: Proceedings of the 7th International Conference on Database Theory*, pages 217–235, London, UK, 1999. Springer-Verlag.

[19] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in High-Dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[20] Thorsten Brants. Tnt: A Statistical Part-of-speech Tagger. In *Proc. of the 6th Conference on Applied Natural Language Processing*, pages 224–231, 2000.

[21] Jim Chan, Celal Ziftci, and David A. Forsyth. Searching Off-line Arabic Documents. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR) (2)*, pages 1455–1462, 2006.

[22] Bei Chang-Da and R. Gray. An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization. *IEEE Transactions on Communications*, 33(10):1132–1133, 1985.

[23] Santanu Chaudhury, Geetika Sethi, Anand Vyas, and Gaurav Harit. Devising Interactive Access Techniques for Indian Language Document Images. In *Proc. of International Conference on Document Analysis and Recognition (ICDAR)*, pages 885–889, 2003.

[24] Kenneth L. Clarkson. A Randomized Algorithm for Closest-Point Queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

[25] Douglas Comer. Ubiquitous B-Tree. *ACM Computing Surveys*, 11(2):121–137, 1979.

[26] T. H. Corman, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1994.

[27] Scott Cost and Steven Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10(1):57–78, 1993.

[28] T. Cover and P. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1):21–27, Jan 1967.

[29] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *ACM Symposium on Computational Geometry (SOCG)*, pages 253–262, 2004.

[30] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, New York, USA, 1997.

[31] Jeffrey Dean and Monika R. Henzinger. Finding Related Pages in the World Wide Web. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 31(11-16):1467–1479, 1999.

[32] L. Devroye and T.J. Wagner. Nearest Neighbor Methods in Discrimination. *Handbook of Statistics, ed. P.R. Krishnaiah and L. Kanal*, 2:193–197, 1982.

[33] David Doermann. The Indexing and Retrieval of Document Images: A Survey. *Computer Vision and Image Understanding (CVIU)*, 70(3):287–298, 1998.

[34] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley & Sons, New York, 2001.

[35] P. Duygulu, Kobus Barnard, J. F. G. de Freitas, and David A. Forsyth. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 97–112, London, UK, 2002. Springer-Verlag.

[36] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining.* American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[37] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. Approximate Nearest Neighbor Searching in Multimedia Databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 503–511, Washington, DC, USA, 2001. IEEE Computer Society.

[38] R. A. Finkel and J. L. Bentley. Quad Trees A Data Structure For Retrieval on Composite Keys. *Acta Informatica*, 4(1):1–9, March 1974.

[39] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by Image and Video Content: The QBIC System. *Computer*, 28(9):23–32, 1995.

[40] Jerome H. Freidman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[41] J. H. Friedman, F. Baskett, and L. J. Shustek. An Algorithm for Finding Nearest Neighbors. *IEEE Transactions on Computers*, 24(10):1000–1006, 1975.

[42] Volker Gaede and Oliver Günther. Multidimensional Access Methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[43] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[44] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proc. of the 25th VLDB conference*, pages 518–529, 1999.

[45] Jacob E. Goodman and Joseph O'Rourke. *Nearest Neighbors in High-dimensional Spaces*. CRC Press, Inc., Boca Raton, FL, USA, 1997.

[46] Stephen M. Harding, W. Bruce Croft, and C. Weir. Probabilistic Retrieval of OCR Degraded Text Using N-Grams. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 345–359, 1997.

[47] Taher H. Haveliwala, Aristides Gionis, Dan Klein, and Piotr Indyk. Evaluating Strategies for Similarity Search on the Web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 432–442, New York, USA, 2002. ACM.

[48] Michael E. Houle and Jun Sakuma. Fast Approximate Similarity Search in Extremely High-Dimensional Data Sets. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 619–630, Washington, DC, USA, 2005. IEEE Computer Society.

[49] Chiou-Ting Hsu and Ming-Chou Shih. Content-Based Image Retrieval by Interest Points Matching and Geometric Hashing. In *International Workshop on Camera-Based Document Analysis and Recognition*, 2005.

[50] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing (SOTC)*, pages 604–613, 1998.

[51] A. Jain and A. M. Namboodiri. Indexing and retrieval of on-line handwritten documents. In *Proc. of the Seventh International Conference on Document Analysis and Recognition (ICDAR)*, page 655659, 2003.

[52] C. V. Jawahar and Anand Kumar. Content Level Annotation of Large Collection of Printed Document Images. In *Proc. of International Conference on Document Analysis and Recognition (ICDAR)*, pages 799–803, 2007.

[53] C. V. Jawahar, Million Mesha, and A. Balasubramanian. Searching in Document Images. In *Proc. of the Indian Conference on Vision, Graphics and Image Processing (ICVGIP)*, pages 622–627, 2004.

[54] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic Image Annotation and Retrieval using Cross-media Relevance Models. In *Proc of the 26th International ACM SIGIR Conference*, pages 119–126, 2003.

[55] Ibrahim Kamel. Fast Retrieval of Cursive Handwriting. In *CIKM '96: Proceedings of the fifth international conference on Information and knowledge management*, pages 91–98. ACM, 1996.

[56] P. Kantor and E. Voorhes. Report on the TREC-5 Confusion Track. In *In Online proceedings of TREC-5 (1996), NIST special publication 500–238*, pages 65–74, 1997.

[57] T. Kanungo, R. Haralick, and I. Phillips. Global and Local Document Degradation Models. pages 730–734, 1993.

[58] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal of Document Analysis and Recognition (IJDAR)*, 9(2):167–177, 2007.

[59] Anand Kumar, A. Balasubramanian, Anoop M. Namboodiri, and C. V. Jawahar. Model-Based Annotation of Online Handwritten Datasets. In *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2006.

[60] Anand Kumar, C. V. Jawahar, and R. Manmatha. Efficient Search in Document Image Collections. In *Asian Conference on Computer Vision (ACCV)*, pages 586–595, 2007.

[61] K. S. Kumar, S. Kumar, and C. Jawahar. On Segmentation of Documents in Complex Scripts. In *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR) Vol 2*, pages 1243–1247, Washington, DC, USA, 2007.

[62] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 614–623. ACM, 1998.

[63] Y. Lamdan and H. Wolfson. Geometric hashing: A General and Efficient Model-Based Recognition Scheme. In *Proc. of the International Conference of Computer Vision (ICCV)*, pages 238–249, 1988.

[64] V. Lavrenko, S. L. Feng, and R. Manmatha. Statistical Models for Automatic Video Annotation and retrieval. In *Proc. of IEEE International Conferene on Acoustics, Speech and Signal Processing (ICASSP)*, pages 17–21, 2003.

[65] Michael Lesk. *Practical Digital Libraries: Books, Bytes, and Bucks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[66] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-Based Multimedia Information Retrieval: State of The Art and Challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

[67] Jia Li and James Z. Wang. Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1075–1088, 2003.

[68] An Image Processing Library. http://www.imagemagick.org/.

[69] Yue Lu. Information Retrieval in Document Image Databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1398–1410, 2004. Senior Member-Chew Lim Tan.

[70] Zhidong Lu, Richard Schwartz, Premkumar Natarajan, Issam Bazzi, and John Makhoul. Advances in the BBN BYBLOS OCR System. In *Proc. of International Conference on Document Analysis and Recognition (ICDAR)*, pages 337–340, 1999.

[71] Simone Marinai and Emanuele Marino. Font Adaptive Word Indexing of Modern Printed Documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1187–1199, 2006. Member-Giovanni Soda.

[72] Simone Marinai, Emanuele Marino, and Giovanni Soda. Indexing and retrieval of words in old documents. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 223–227, 2003.

[73] B. Matei, Ying Shan, H.S. Sawhney, Yi Tan, R. Kumar, Daniel Huber, and Martial Hebert. Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(7):1111–1126, July 2006.

[74] Alexa T. McCray and Marie E. Gallagher. Principles for Digital Library Development. *Communications of the ACM*, 44(5):48–54, 2001.

[75] Million Meshesha. *Recognition and Retreival from Document Image Collections*. P.hD. Thesis. International Institute of Information Technology (IIIT), Hyderabad, India, 2008.

[76] M. Mitra and B. B. Chaudhuri. Information Retrieval from Documents: A Survey. *Information Retrieval (IR)*, 2(2-3):141–163, 2000.

[77] Florent Monay and Daniel Gatica-Perez. On image auto-annotation with latent space models. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 275–278, New York, USA, 2003. ACM.

[78] George Nagy. Twenty Years of Document Image Analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(1):38–62, 2000.

[79] T. Nakai, K. Kise, and M. Iwamura. Use of Affine Invariants in Locally Likely Arrangement Hashing for Camera-Based Document Image Retrieval. In *International Workshop on Document Analysis Systems (DAS)*, pages 541–552, 2006.

[80] Carlton W. Niblack, Ron Barber, Will Equitz, Myron D. Flickner, Eduardo H. Glasman, Dragutin Petkovic, Peter Yanker, Christos Faloutsos, and Gabriel Taubin. QBIC

Project: Querying Images by Content, Using Color, Texture, and Shape. In *Proceedings of the Conference on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–187, 1993.

[81] Digital Library of India (DLI). http://www.dli.gov.in.

[82] Lawrence O'Gorman and Rangachar Kasturi. *Document Image Analysis*. IEEE Computer Society Press, 1995.

[83] OMTRNAS - Indian Language Transliteration. http://www.cs.cmu.edu/~madhavi/Om/.

[84] U. Pal and B.B. Chaudhuri. Indian Script Character Recognition: A Survey. *Pattern Recognition*, 37:1887–1899, 2004.

[85] Apostolos N. Papadopoulos. *Nearest Neighbor Search: A Database Perspective*. Springer-Verlag Telos, 2004.

[86] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, USA, 1985.

[87] The Carnegie Mellon Sphinx Project. http://cmusphinx.sourceforge.net.

[88] T. Rath and R. Manmatha. Word Image Matching using Dynamic Time Warping. In *Technical Report MM-38, Center for Intelligent Information Retrieval, University of Massachusetts Amherst*, 2002.

[89] Toni M. Rath and R. Manmatha. Features for Word Spotting in Historical Manuscripts. In *Proc. of the Seventh International Conference on Document Analysis and Recognition (ICDAR)*, page 218222, 2003.

[90] Toni M. Rath and R. Manmatha. Word Image Matching Using Dynamic Time Warping. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR) (2)*, pages 521–527, 2003.

[91] Toni M. Rath and R. Manmatha. Word Spotting for Historical Documents. *International Journal of Document Analysis and Recognition (IJDAR)*, 9(2):139–152, 2007.

[92] Toni M. Rath, R. Manmatha, and Victor Lavrenko. A Search Engine for Historical Manuscript Images. In *ACM SIGIR*, pages 369–376, 2004.

[93] Gregory Russell, Michael P. Perrone, Yi min Chee, and Aiman Ziq. Handwritten Document Retrieval. In *IWFHR '02: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, pages 233–238, 2002.

[94] K. Pramod Sankar and C. V. Jawahar. Probabilistic Reverse Annotation for Large Scale Image Retrieval. In *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[95] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing).* The MIT Press, 2006.

[96] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast Pose Estimation with Parameter-Sensitive Hashing. In *Proc. of International Conference on Computer Vision (ICCV)*, pages 750–757, 2003.

[97] Robert F. Sproull. Refinements to Nearest-Neighbor Searching in k-Dimensional Trees. *Algorithmica*, 6(1):579–589, December, 1991.

[98] S. N. Srihari, H. Srinivasan, C. Huang, and S. Shetty. Spotting Words in Latin, Devanagari and Arabic Scripts. *Vivek: Indian Journal of Artificial Intelligence*, 16(3):2–9, 2006.

[99] Sargur Srihari, Harish Srinivasan, Pavithra Babu, and Chetan Bhole. Spotting Words in Handwritten Arabic Documents. *Document Recognition and Retrieval XIII*, 6067(1):1–12, 2006.

[100] Ching Y. Suen, Shunji Mori, Soo-Hyung Kim, and Cheung H. Leung. Analysis and Recognition of Asian Scripts - the State of the Art. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 866–879, 2003.

[101] K. Kise T. Nakai and M. Iwamura. Hashing with Local Combinations of Feature Points and Its Application to Camera-Based Document Image Retrieval. In *First International Workshop on Camera-Based Document Analysis and Recognition (CBDAR2005)*, Seoul, Korea, 2005.

[102] Kazem Taghva and Jeffrey Coombs. Hairetes: A Search Engine for OCR Documents. In *Proc. of 5th International Workshop on Document Analysis Systems (DAS)*, pages 157–168, 2002.

[103] Chew Lim Tan, Weihua Huang, Zhaohui Yu, and Yi Xu. Imaged Document Text Retrieval Without OCR. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(6):838–844, 2002.

[104] The Festival Speech Synthesis System. http://www.cstr.ed.ac.uk/projects/festival/.

[105] J. M. Trenkle and R. C. Vgot. Word Recognition for Information Retrieval in the image domain. In *Proc. of Document Analysis and Information Retrieval*, pages 105–122, 1993.

[106] N. Vasconcelos and A. Lippman. A Probabilistic Architecture For Content-Based Image Retrieval. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 216–221, 2000.

[107] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *VLDB '98: Proceedings of the 24th International Conference on Very Large Data Bases*, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[108] H. Wolfson and I. Rigoutsos. Geometric Hashing: An Overview. *IEEE Computational Science and Engineering*, 4(4):10–21, 1997.

[109] A. C. Yao and F. F. Yao. A General Approach to d-Dimensional Geometric Queries. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 163–168, New York, NY, USA, 1985. ACM.

[110] G. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.

[111] V. M. Zolotarev. *One-Dimensional Stable Distributions (Translations of Mathematical Monographs)*. Amer Mathematical Society, 1992.

# Appendix A

# Related Publications

- Anand Kumar, C. V. Jawahar and R. Manmatha. "Efficient Search in Document Image Collections". *Asian Conference on Computer Vision (ACCV)*, pages 586–595, November 18-22, 2007, Tokyo, Japan.
- C. V. Jawahar and Anand Kumar. "Content Level Annotation of Large Collection of Printed Document Images". *International Conference on Document Analysis and Recognition (ICDAR)*, pages 799–803, September 23-26, 2007, Brazil.
- Anand Kumar, A. Balasubramanian, Anoop M. Namboodiri and C.V. Jawahar. "Model-Based Annotation of Online Handwritten Datasets". *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, October 23-26, 2006, La Baule, France.