Retrieval of Online Handwriting by Synthesis and Matching

C. V. Jawahar, A. Balasubramanian, Million Meshesha and Anoop Namboodiri

Center for Visual Information Technology,
International Institute of Information Technology,
Gachibowli, Hyderabad - 500 032, India

Abstract

Search and retrieval is gaining importance in the ink domain due to the increase in the availability of online handwritten data. However, the problem is challenging due to variations in handwriting between various writers, digitizers and writing conditions. In this paper, we propose a retrieval mechanism for online handwriting, which can handle different writing styles, specifically for Indian languages. The proposed approach provides a keyboard-based search interface that enables to search handwritten data from any platform, in addition to pen-based and example-based queries. One of the major advantages of this framework is that information retrieval techniques such as ranking relevance, detecting stopwords and controlling word forms are extended to work with search and retrieval in the ink domain. The framework also allows cross-lingual document retrieval across Indian languages.

Key words: Search and Retrieval, Handwriting Synthesis, Online Handwriting, Indian Language Documents, Information Retrieval Measures

1 Introduction

Pen-based interfaces are gaining popularity due to the flexibility and convenience of pen as an interface. The compactness of online handwritten data also enables efficient storage and communication. Moreover, handwriting has more expressive power as compared to typed text due to the possibility of annotations and sketches, that makes it an effective medium of communication. The pen technologies have also matured over the last twenty years starting from touch screen based sensors with limited resolution to highly accurate and robust sensors based on electromagnetic and sonic technologies. Due to its capabilities, applications that treat handwriting or *ink* as the primary data

type are also on the rise. However, as the amount of data available in the form of handwriting increases, access to specific documents becomes an issue due to the inability of current indexing and retrieval algorithms to efficiently handle such data.

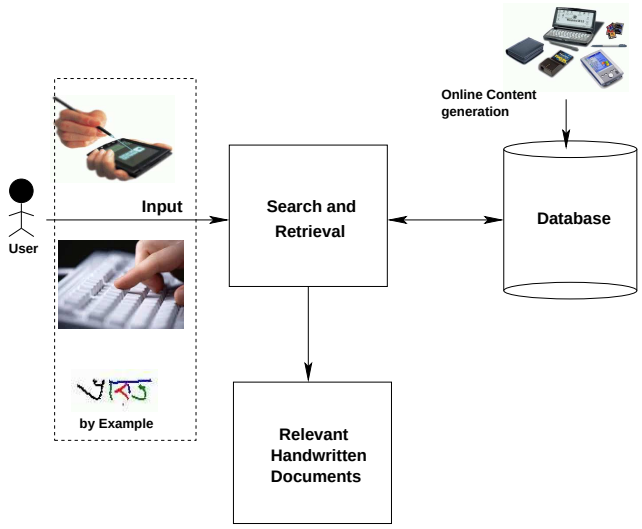


Fig. 1. An effective online handwritten database should accept queries based on Keyboard, Pen or a Sample Handwritten Word.

The primary solution to handle online handwritten data is to employ a *HWR* (handwriting recognizer) to convert the ink into text, and use the results to search and retrieve the documents [1,2]. However, this approach is suited only where the handwritten data is purely text and where a robust HWR is available for the language contained in the document. An alternate solution is to do matching and retrieval in the ink domain itself. The problem is very challenging due to the large amount of variations present in online handwriting. The sources of these variations include: i) differences in writing styles of various users, and inconsistency in writings of a single user, ii) differences in writing surfaces and capabilities of different digitizers, iii) noise introduced by the digitizers, representations, etc. In addition to the above, online handwriting also contain variations due to differences in writing speeds as the temporal information is also captured in the digitization process. Figure 1 shows search for relevant documents from a collection of handwritten documents. The input could be a handwriting using a pen/stylus, a sample word from a document, or even text typed from a keyboard. Users expect the most relevant documents to be retrieved from a database and presented to them in a ranked and meaningful way.

The level of complexity of the search increases as the diversity and size of the document collection increases. The range of applications also varies accordingly.

- **Single Writer Collections:** These are typical in scenarios like the archived

notes taken by the user of a pen-based device. The matching and retrieval of online documents are not easy even with such a collection of homogeneous writings [3]. Such collections usually contain a single script or language.

- **Multi-Writer Single Script:** As the number of writers increase, the variability of handwriting also increases dramatically. Such document collections are part of applications that communicate handwritten documents across users, such as a mailing application.
- **Multi-Writer Multi-Script:** As digital communications span across continents, the documents transmitted are likely to contain a large variety of languages and scripts. Dealing with such documents would be essential for email processing applications such as spam filters or searches in the archives of an organization. The search in such large collections have to be efficient in addition to being able to handle the different languages used in the documents. The problem is commonplace in a country like India, where there are 18 official languages, most of them having their own scripts.
- **Digital Libraries:** Applications such as digital libraries add one more level of complexity to the problem due to an increase in the order of magnitude of the database as well as the varieties in devices that are used for digitization of handwriting.

2 Retrieval of Handwritten Documents

Approaches used for searching handwritten documents can be classified into recognition-based and recognition-free techniques. Depending on the application and digitization process, the data could be either word images (offline) or traces of pen motion (online). Recognition of online data has the advantage of using the additional temporal information present in the data.

2.1 Overview of the Previous Work

Recognition of handwritten data has received a lot of research attention in the past. Initial attempts in this direction were to recognize scanned images of handwritten characters (offline handwriting recognition) and is useful in applications such as postal address recognition [8], handwritten form recognition [9], handwritten bank check recognition [10], etc. However, in the case of HCI for pen-based devices, one can utilize the additional writing order, direction and velocity information to aid the recognition process. This is referred to as online handwriting recognition. The strokes in the word could be modeled using statistical models such as HMMs [11]. Such models are often tuned for a particular writer's handwriting. On the other hand, writer independent handwriting recognition is an extremely difficult problem due to the extent

Work	Data	Approach	Pros	Cons	Applications
Rath and Manmatha [4]	Offline, Historic	Word Image Matching	Accuracy	Single Writer	Single writer document collections
Srihari and Shi [5]	Offline	Writer Matching	Multi-User	Lower accuracy	Forensic document retrieval
Russell *et al* [1]	Online	Recognition results	Multi-User, Fast	Needs Recognizer	Indexing and retrieval for multi-user and single script collections
Kamel [6]	Online	KLT-based, RTree	Multi-User, Fast	Limited Data size, Lower accuracy	Online document retrieval
Jain and Anoop[3]	Online	Ink Matching	Accurate	Single User	Search in single-writer document collections
Balasubramanian *et al* [7]	Offline, printed	Word Matching	Accurate, Robust	Slow to Index	Search in large printed document collections
Current Work	Online	Synthesis and Matching	Multi-user, Accurate	Slow to Index	Search, Index multi-user document collections

Table 1
Overview of existing and current work in the area of document retrieval.

of variation among writing styles of different writers. Such variations require the recognition engines to be trained to a particular user's handwriting style. Adaptation of a recognizer to a specific writers' handwriting [12] has been the most promising solution in this regard.

A second approach is to represent the handwritten words using a set of features and match two words by comparing the corresponding feature vectors. This approach is referred to as *word spotting*, and is quite effective when compared to recognition-based search for poor quality documents. Rows 1 and 5 of Table 1 show examples of the word spotting technique applied to offline and online handwritten documents.

2.2 Challenges in Recognition-free Handwritten Document Retrieval

Searching and retrieval of relevant documents from handwritten data collections is a challenging task, especially when there is no textual representation. To come up with a successful search engine in the ink domain, we need to address the following issues:

Variations in Handwritten Data: Search in handwritten documents requires appropriate representational schemes and similarity measures. This is because of the large amount of variations in handwriting style, speed and direction among various writers, besides variations and noises introduced by digitizers. We need to come up with robust feature extraction and matching schemes, which can represent the content well, while invariant to writing variations.

Scalability: Retrieval of relevant documents requires matching users query word with handwritten words in the collection. Doing this online is inefficient and time taking, which can possibly be solved by indexing documents offline. However, indexing multiple writers documents is a difficult task unless textual representation of index terms are available.

Segmentation: The existence of irregularities in handwriting, inconsistency of inter-word and inter-character spacing in handwritten data and the existence of non-textual objects in a page makes detection of words in handwritten document a challenging task. Robust segmentation tool needs to be designed to detect words from handwritten data.

Document Relevance Ranking: Existing information retrieval measures like TF/IDF are designed for ranking relevance of textual documents. To make sure that the retrieval system achieves users expectations there is a need to design suitable ranking methods in the ink domain.

Indian languages: Indian languages pose many additional challenges. Some of these are: (i) lack of standard representation for the fonts and encoding, (ii) lack of support from operating system, browsers and keyboard, (iii) lack of language processing routines, (iv) lack of robust HWRs, and (v) presence of multiple scripts. These issues add to the complexity of developing handwriting retrieval scheme.

2.3 The Proposed Approach

The approach proposed here consists of three major steps: i) Synthesis of handwritten data from the query, ii) Matching of the query to words in the

database, and iii) Computation of relevance scores of documents to order the results of matching. The synthesis is primarily aimed at Indian scripts, which are non-cursive in nature. The synthesized handwritten word is then matched with all the words in the database and those with scores below a threshold qualifies as matches. The matching documents are then ranked according to the frequency of the search term in the document (TF) and the inverse of the number of documents containing the word (IDF), and the result is presented to the user.

The following are the characteristic differences of the current work, when compared to other approaches to retrieval of online handwriting data.

(1) **Use of Synthesis:** Our approach employs a handwriting synthesis module to map the query text into the ink domain. Matching is done at the ink level rather than on the recognition output. This enables the use of a keyboard, pen or example word for query input.
(2) **Use of IR Measures:** The proposed approach extends the concepts of TF/IDF and word-form variations into the ink domain and integrates it with the search and retrieval module. This enables us to have better page rank computations. This part is an extension of our work that uses a similar approach for printed documents [7].
(3) **Indexing:** Indexing and clustering schemes are introduced to organize multi-user document collections so as to enhance efficiency of searching and retrieval from handwritten data.
(4) **Cross-Lingual Retrieval:** The synthesis module, when combined with the transliteration properties of Indian languages enables us to do cross-lingual retrievals for a single query word.
(5) **Indian Language Search:** The state of both OCR and handwriting recognition for Indian languages are still in its infancy and there are no commercial OCR or HWR programs available even on platforms like the Tablet PCs. A recognition-free approach is extremely useful in this context. To the best of our knowledge, this is the first attempt at the problem of document retrieval from Indian language online handwritten data.

The bottleneck of most recognition-free approaches is the fact that one is trying to match the content of the query word with that of the words in a document. The matching algorithm proposed by Jain and Namboodiri [3] achieves excellent results for word matching. However, the performance of the algorithm drops drastically when one tries to compare the words written by two different writers. In general, ink-matching based algorithms work well only for single user document collections. Moreover, the user interface for the method described in [3] is restricted to pen-based input and hence is not convenient in all settings. The user needs to provide an example query word, in place of the textual query that we employ in this work. We solve both of

the problems by generating a handwritten sample of the query word in the writing style used in the document. In other words, matching is done using a synthesized sample of the query in the style of the writer under consideration, which is generated from typed text.

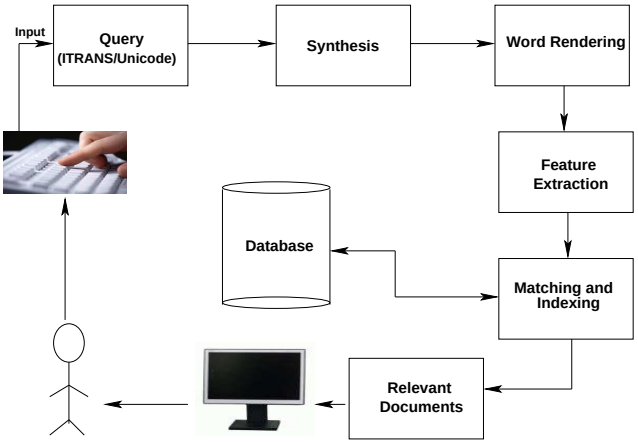


Fig. 2. The entire process of synthesis and retrieval of relevant documents.

The block diagram in Figure 2 describes the entire process of synthesis of the query text and retrieval from handwritten document collections. The input query is entered either in ITRANS or in Unicode. The query word is then synthesized and the corresponding handwritten word is rendered by the word rendering module. Features are then extracted from the rendered word and is matched against all the feature values of the handwritten words that are stored in the database. Then relevant handwritten documents are fetched based on their relevance scores for users view. Note that in the present work we assumed handwritten word segmentation is available and hence it is not part of this work.

3 Handwriting Synthesis and Modeling

Handwriting synthesis transforms the query text into the ink domain. To get better results during synthesis, models of handwritten data are learned from training samples. In doing so synthesis enables to perform matching at the ink level.

3.1 Online Handwriting Synthesis

Given an input text, the problem of handwriting synthesis is to generate data that is close to how a human would write the text. The characteristics of the generated data could be that of a specific writer or that from a generic model.

Even with a given model, the synthesis method should not be deterministic since the variations that are found in human handwriting are inherently stochastic. However, if we need to generate data that is similar to a particular writer's handwriting, we need to identify, model, and preserve the basic characteristics of his/her handwriting. The problem of maintaining the writing style while introducing variability [13] makes the problem of synthesis very difficult. A handwriting synthesis solution has a variety of applications including automatic creation of personalized documents [14], generation of large quantities of annotated handwritten data for training recognizers [15][16], and writer-independent matching and retrieval of handwritten documents.

Traditionally, handwriting synthesis has been dealt within the realm of offline handwriting [17], where the handwritten data is a scanned image of a paper document. Online handwriting is stored as a sequence of *strokes*, where each stroke is defined as the trace of the pen tip from a pen-down to the next pen-up. Devices with pen-based interfaces facilitate storing of the handwritten ink in the digital format and thus enabling a variety of applications such as search and retrieval of large sets of handwritten notes [18] as well as efficient communication via the Internet. In the context of digital ink, the technique of handwriting synthesis is extremely useful as it leads to applications that preserve the compactness of online data, while being natural.Additional details of the synthesis process could be found in [19].

(a) (b)

Fig. 3. Telugu word EdainA written by (a) Writer 1, and (b) Writer 2.

3.1.1 Characteristics of Indian Scripts

The problems associated with handwriting synthesis are different depending upon the nature of the script that one is trying to synthesize. Languages that use the Roman script contain a small set of symbols that are arranged in a linear fashion to create a word. The complexity of these scripts arises due to the cursive nature of the script, where the individual characters are connected together. In fact, real-world handwriting is a mixture of cursive and non-cursive parts, which makes the problems of recognition and synthesis, more difficult.

Indian language scripts are fundamentally non-cursive in nature, where the `aksharas` (equivalent to characters) are written independently, separated by space or pen-lifts. An `akshara` can be combination of one or more (up to 3) consonants with a vowel. However, these scripts often contain a large num-

ber of characters that have complex spatial layout of strokes. Indian scripts have compound characters, which are combinations of multiple consonants and vowels. The handwriting synthesis process should hence model all the possible variations of characters and their combinations to be able to generate any given text. This makes the problem of synthesis, extremely complex in the case of Indian language scripts.

There are many other properties of the Indian scripts that are not seen in Roman. This arises due to a variety of factors:

- Alphabets of Indian scripts have far more complex shapes and varied writing styles.
- The size of the alphabets is typically high. In addition, the presence of *samyuktaksharas* (compound characters) make modeling Indian scripts more difficult.
- The basic stroke shapes in Roman scripts are often unambiguous in their meaning. However, this is not the case with the Indian scripts, where a single stroke shape can acquire different meanings depending on its position and size.
- Indian scripts are non-cursive in nature and thus the available models for cursive scripts need not be the most suitable.
- The spatial location of an `akshara` is dependent on the previous `akshara` in some Indian language scripts.

Along with the spatial complexity, the variance in writing styles also increases for Indian scripts. For example, Figure 3 shows the handwritten Telugu word *EdainA* (means *anything*) written by two different writers. As it is evident from Figure 3, the second writer's handwriting (Figure 3 (b)) is readable and clean as compared to that of the first writer (Figure 3 (a)), whose handwriting sample is composed of several pen lifting movements, stroke discontinuities, slant and other deformities. Also note the fact that the stroke order is not unique for every writer for a given word and sometimes even the same writer has different stroke order when writing the same word.

As pointed out before, the scripts of Indian languages are more complex than Roman script. Following is a list of interesting characteristics of Indian scripts that are relevant to the synthesis problem. Words in many of the Indian language scripts (like Hindi, Bangla, Marathi, and Gurumukhi (Punjabi)) have *Shirorekha* (horizontal bar on the top). Figure 4(a) shows how characters of *Bangla* script are joined on the top with the *Shirorekha* to form a word. The other feature of Indian scripts is shown in Figure 4(b), where vowels often get converted as augmented shape modifiers to consonants in most Indian scripts. In *Hindi* when the consonant *ka* is combined with the vowel *e* it gives a compound `akshara` *ki*. Figure 4(c) depicts the third case. This is an example of *Samyuktakshara* which could contain multiple consonants and vowels. In *De-*

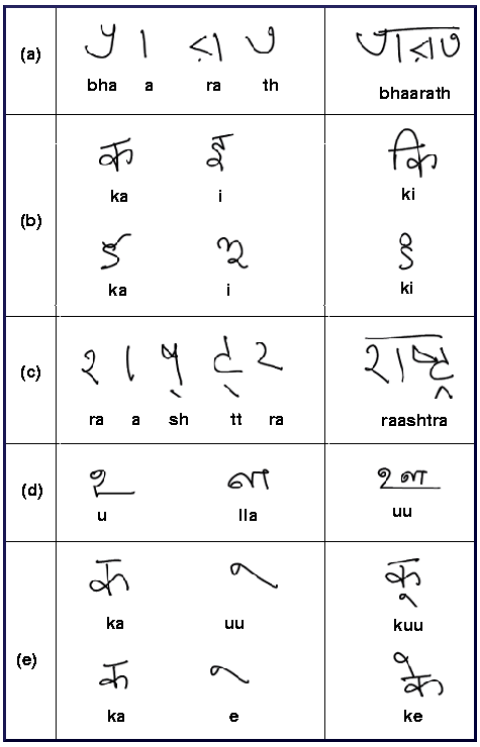


Fig. 4. Some of the special cases in Indian language scripts.

vanagari script the word *raashtra* is formed from the basic consonants and vowels listed. The fourth case (Figure 4(d)) shows an example from the *Tamil* script where the vowel sound *uu* is a concatenation of the symbols of *u* and *lla*. Finally, in Figure 4 (e) we have the consonant *ka* in *Hindi*, which when combined with the vowel *uu* results in *kuu* and when combined with the vowel *e* results in *ke*.

One may observe that for Indian scripts, spatial positioning of the strokes is equally important as their shapes.

3.1.2 Synthesis of Non-Roman Scripts

Roman script was the primary focus in the motor model based synthesis techniques [20][21]. In addition there have been attempts like modulation of oscillatory motions of the pen [22], the vector-matrix of successive strokes [23] and use allograph codes as input [24]. On the other hand there has been no concrete model available for the oriental languages. Some oriental characters need many pen-tip lifting steps due to the presence of large number of short segments. Scripts such as the Korean has been studied before for the purpose of synthesis [25,26]. The Beta-Velocity model was proposed for Kanji scripts [25] to simulate cursiveness with a letter or a word. This model was an improvised version of the Delta LogNormal model, the main difference being that the Beta-Velocity model uses asymmetric curves, whose skewness can be

controlled by variables.

In this work, we use a stroke shape and layout model for Indian language scripts that can be learned from labeled samples. Hence we can use the model for generation of a specific writer's handwriting or develop a generic writer model. This model captures both the shape and temporal aspects of the strokes as well as their order information. Hence we can generate either online or offline data using the learned parameters.

3.2 Modeling Handwritten Data

Figure 5 gives the outline of the learning and synthesis steps of our method. The handwriting model consists of two parts, a **stroke model**, which captures the shapes and variations in the basic strokes that form the characters, and a **layout model**, which controls the spatial layout of the individual strokes. The individual models can be learned from online handwritten data that is collected from a writer or multiple writers. The training data needs to be annotated manually. Examples of strokes corresponding to each stroke class is used to learn the stroke model. The spatial distribution of strokes are learned by the layout model.

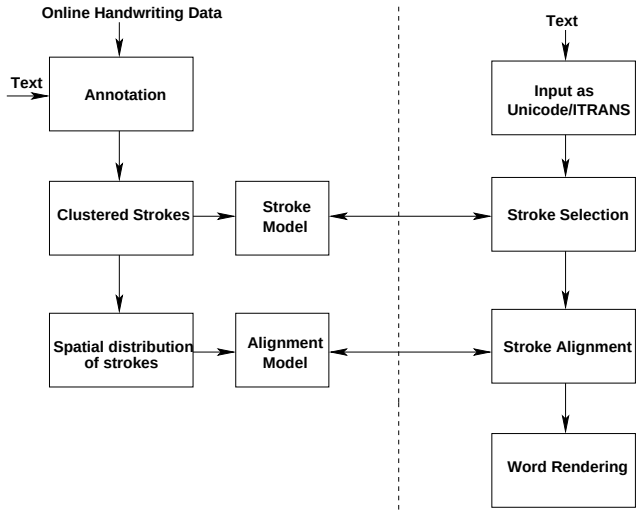


Fig. 5. Block diagram for handwriting synthesis. The modules to the left of the dotted line forms the training phase, while those to the right form the online synthesis.

3.2.1 Stroke Model

Each script contains a set of basic strokes that are used to form all the characters in the script. For example, Hindi consists of around 200 basic strokes, while Telugu consists of more than 300 basic strokes. The stroke model consists

of a representation for each of these basic strokes. We have used two different models for representation of the strokes: Normalized Template Model and Mean Trace Model.

Normalized template model represents each basic stroke class using a set of training strokes, that are normalized in size. The stroke selector will randomly select one of the samples for the required class and scale it to the appropriate size which is determined by the position of the stroke in the character. The model M, could be represented by a set of k stroke models, each containing a set of stroke samples:

$$M = \{S_1, S_2, .., S_k\}$$
$$S_i = \{s_1, s_2, .., s_{n_i}\}$$
$$s_i = [(x_1, y_1)(x_2, y_2)..(x_m, y_m)],$$

where S_i represents the model of the i^{th} stroke class containing n_i stroke samples, and each s_i is a sample that consists of a sequence of (x, y) coordinate pairs.

Mean trace model computes the mean of all the traces (or strokes) of each of the given stroke classes by normalizing the strokes and aligning the points using an elastic matching technique. The distribution of the samples of the strokes are estimated using the means and covariance matrices of the aligned sample points and are stored in their trace order. The stroke selector module will generate random samples from each of the distributions to create a new stroke of a given class, assuming a Gaussian distribution.

The mean model estimate is the maximum likelihood estimate of the sample sequence, assuming each sample comes from a multivariate Gaussian distribution.

$$M = [\bar{X}_1, \bar{X}_2, .., \bar{X}_k]^T$$
$$\bar{X}_{ip} = \frac{1}{n_i} \sum_{j=1}^{n_i} (x_{pj}, y_{pj}),$$

where $\bar{X}_i$ represents the mean of the p^{th} point of the i^{th} stroke class containing n_i stroke samples, and each sample point of the strokes model is the mean of the corresponding points ((x, y) coordinate pairs) after alignment and outliers removal.

Note that the stroke models that we employ are relatively simple compared to the motor model based approaches for Roman scripts. However, since the strokes themselves are only parts of characters in the Indian language scripts, their spatial distribution, captured by the layout model, can generate realistic renderings of handwriting.

3.2.2 Layout Model

The most important part of our synthesis model is the layout model that is capable of capturing the spatial relationship between stroke classes. Let ω_i and ω_j be two stroke classes that are modeled using the stroke model defined in Section 3.2.1. We describe the layout model as the spatial relationship between the two strokes ω_i and ω_j in terms of distance, r, and direction, θ. In this experiment the relative distance and direction are computed based on the bounding boxes.

Let $p(r, \theta|\omega_i\omega_j)$ represent the spatial distribution of the class ω_j with respect to the class ω_i, where r is the radial distance and the θ is the angle between the two classes. We represent the spatial layout $D_{\mathcal{L}}$ of a script/language $\mathcal{L}$, as a set of such pairwise spatial distributions of the strokes:

$$D_{\mathcal{L}} = \{p(r, \theta|\omega_i\omega_j)|\omega_i \text{ and } \omega_j \text{ are neighbors in } \mathcal{L}\}$$

Here we assume that the parameters r, θ form a distribution for each stroke class pair, which can be modeled as a multivariate Gaussian distribution. The mean and covariance matrices are estimated using MLE, similar to that in the stroke model. Once the model parameters are estimated from the annotated samples, we can synthesize any given text based on the synthesis procedure (shown to the right of the dotted line in Figure 5).

Each character in the Indian script could compose of multiple strokes and its number can differ based on the writing style. Hence, in addition to the spatial layout, we also need to learn the set of stroke classes that are used to write a particular character by specific writer. This information is identified and stored for each character class during the training phase. Detailed discussion of stroke and layout models could be found in [19].

4 Retrieval by Synthesis

We retrieve online handwriting by synthesizing the query word as online handwritten data. Thereafter we find the results by matching and ranking relevance of documents identified during searching.

4.1 Query Expansion

During searching if a relevant document does not contain the terms that are in the query, then that document will not be retrieved. Query expansion is

the process of reformulating a given query to reduce query-document mismatch [27]. The aim of query expansion is to improve retrieval performance (precision and/or recall) in information retrieval operations. This procedure have greater importance for cross-lingual retrieval of handwritten documents in Indian language, since relevant documents may exist in some other language. In text search engines the additional terms may be taken from a thesaurus. For example a search for 'car' may be expanded to 'car', 'cars', 'automobile', 'automobiles', etc. In this work, multi-script query expansion is done to allow cross-lingual search and document retrieval across Indian languages. Thus, given an input query we use transliteration to generate the corresponding terms in any of the Indian scripts.

4.2 Query Generation

Let $\omega_1, \omega_2, ..., \omega_k$ be the k primitive classes that compose the entire script. These primitive classes consist of strokes extracted from the stroke model. Let x_i be a particular stroke in the training samples of ω_j. Conventional algorithms model $p(x_i|\omega_j)$ (denoted as $p_{\omega_j}(x_i)$) based on a set of parameters that control the shape of x_i [28], and the primitives used are characters. In our model, $p_{\omega_j}(x_i)$ is controlled by the stroke model.

The word is synthesized using the individual primitive classes, and the distribution of samples within the classes. The synthesis proceeds as follows:

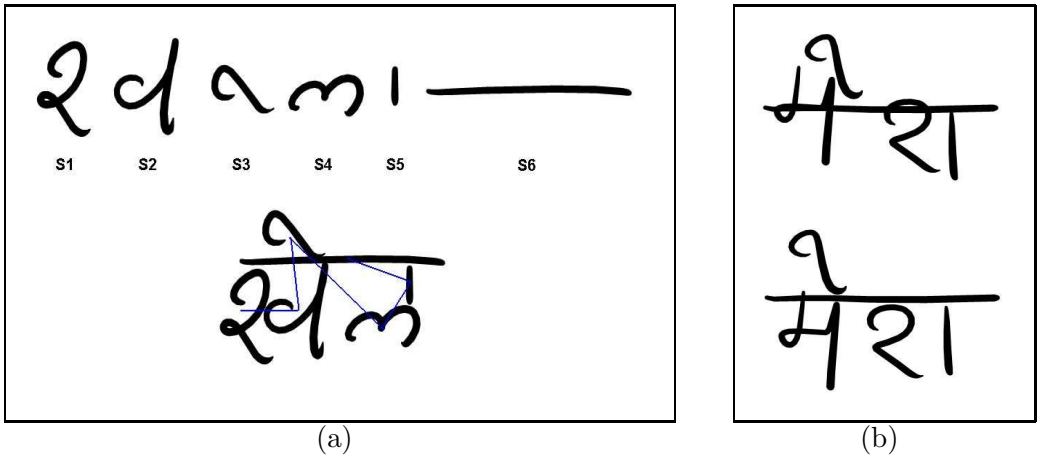
- Given a text word, create a sequence of stroke classes that constitute the handwriting equivalent of the input word. This information is learned during the training of the layout model.
- For each stroke class, we select/generate a sample stroke using the stroke model for the writer under consideration.
- The layout model is used to arrange the sample strokes to generate the final word.
- The word could be rendered in appropriate form, depending on the application or could be passed to the next phase in applications such as retrieval and training of recognizers and also can be used for other applications.

4.2.0.1 Stroke Selection: The stroke selection module selects a sample stroke from the set of training samples in the case of normalized template model. For the mean trace model, we generate a sample stroke based on the learned distribution of the sample points within the stroke. One can introduce variations in the synthesis by generating random samples from each sample distribution. Alternately, the mean of the sample points would give the most likely stroke sample for each stroke class.

4.2.0.2 Stroke Alignment: The sequence of primitives generated by the stroke selection algorithm should be aligned to generate the handwritten query word. We use the layout model, computed during the training phase, to determine the orientation between every pair of consecutive strokes. Given two consecutive strokes, belonging to classes ω_{i-1} and ω_i, we compute the most likely position of the second stroke with respect to the first stroke: $argmax_{r,\theta} p(r, \theta|\omega_{i-1}\omega_i)$, where r, θ is the distance and direction of the second stroke with respect to the first. To introduce writing variations, one could generate random samples of r and θ according to the density function, $p(r_i, \theta_i|\omega_{i-1}\omega_i)$. Note that position of the first stroke of the word is immaterial, and hence the total probability of the synthesized word with m stroke classes can be written as:

$$P(word) = p(x_1|\omega_1) \prod_{i=2}^{m} p(r_i, \theta_i|\omega_{i-1}\omega_i) p(x_i|\omega_i),$$

where $p(x_i|\omega_i)$ gives the probability generation of each stroke from the corresponding class, and $p(r_i, \theta_i|\omega_{i-1}\omega_i)$ gives the probability of observing the two stroke classes, ω_{i-1} and ω_i, in that particular relative spatial positions. Figure 6(a) illustrates the concept, where the lines indicate the direction and distance (r, θ) generated by the synthesis module.



(a) (b)

Fig. 6. (a) Generation of spatial layout from strokes and (b) an erroneous layout along with the ideal output.

The approach works well to align the generated sequence, except when an error or outliers is generated in some position, which gets propagated to the remaining strokes. Figure 6(b) shows such an example of error in alignment along with the ideal layout that should have been created from the strokes. We note that the second stroke was placed incorrectly above the first one, which led to incorrect placement of the subsequent strokes. The errors in synthesis could affect the retrieval performance. However, by using appropriate stroke distance measures (such as distance between bounding boxes) that suits a particular script, such errors can be minimized.

4.3 Matching Handwritten Words

Once a handwritten word is synthesized from the query word, we use it to search the database of handwritten documents using elastic matching. Since every online handwritten word is a collection of strokes, we need to define a matching technique to compare two strokes as well as two words. Distance or dissimilarity between two strokes is computed using a set of features extracted from the group of strokes. Each stroke consists of a sequence of sample points, (x, y) that describes the trace of the pen during writing. The strokes are first converted into a sequence of feature vectors, extracted from each of the sample points. The feature vector consists of:

(1) The direction, θ, of the tangent to the stroke curve
(2) The curvature, c, of the stroke at the sample point, and
(3) The height, h, of the sample point from the word baseline

Figure 7 (a) illustrates the computation of the three features. The angle, curvature and height of a point on the stroke completely characterizes the local neighborhood. Moreover, these features have been demonstrated to be effective for online word spotting for single user datasets [3].

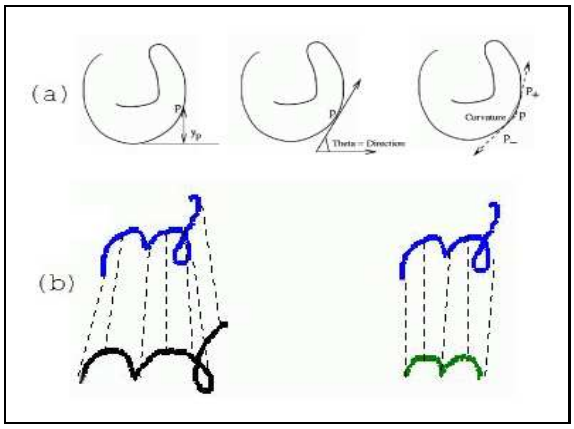


Fig. 7. The Matching process. (a) Features computed for stroke matching: direction, height and curvature, and (b) Matching using dynamic time warping.

The distance between two feature vectors $F_1 = \langle \theta_1, c_1, h_1 \rangle$ and $F_2 = \langle \theta_2, c_2, h_2 \rangle$ is defined as the weighted Euclidean distance between the two vectors:

$$D^2 = k_\theta * (\theta_1 - \theta_2)^2 + k_c * (c_1 - c_2)^2 + k_h * (h_1 - h_2)^2,$$

where ks are the weighting factors. A sequence alignment score $D(i, j)$ between two sequences $F = \mathrm{F}_1, \mathrm{F}_2, \ldots, \mathrm{F}_M$ and $G = \mathrm{G}_1, \mathrm{G}_2, \ldots, \mathrm{G}_N$ is then computed using a Dynamic Time Warping (DTW) procedure. The use of the total cost of Dynamic Time Warping as a similarity measure is helpful to group together strokes that are related to their root character by partial

match. Dynamic Time Warping is a dynamic programming based procedure to align two sequences of signals. This can also provide a similarity measure. Further details of DTW computation is available in [7].

The distance between two words are computed similar to the inter-stroke distance. However, in this case, the primitives becomes strokes (unlike feature vectors in stroke matching) and $D(i, j)$ is used as the distance measure between strokes. The DTW algorithm used at the word level allows us to handle spurious or missing strokes in a word. It also allows us to do partial matching of words as will be seen in the next section.

4.4 Information Retrieval Measures for Handwritten Document Retrieval

Most of the document retrieval algorithms mentioned in Section 2, focus on spotting of similar looking words and retrieving documents containing such instances. However, with widespread use of textual search engines along with powerful information retrieval techniques, one naturally look for mimicking these ideas, even in the absence of an explicit recognition engine. We had already presented a computationally efficient procedure for the indexing and retrieval of offline printed documents [7]. Here, we extend the same idea to the online handwritten data. Table 2 shows a comparative review of how IR principles in text-domain are effectively used for searching handwritten data.

	Algorithm(s) Used	
Modules	Text search engine	Current work
Query expansion	Thesaurus	Language modeling and transliteration
Matching	String matching	Elastic matching
Stemming	Language modeling (e.g. Porter algorithm)	Dynamic time warping
Stopword detection	Negative dictionary	Inverse document frequency and negative dictionary
Relevance ranking	Term frequency (TF)	Modified TF/IDF
Clustering	—	Minimum Spanning Tree
Indexing structure	Inverted index and signature file	Inverted index

Table 2
A comparative review of information retrieval measures used in text search engines vs. the present work.

In the case of ASCII encoded text documents, the retrieved documents are ranked according to a relevance score. The relevance scores are computed as follows: i) stop words in the query and documents are removed to avoid spurious documents from being retrieved, ii) morphological variations of the word are identified, and iii) relevance of the document to a query is ranked based on factors such as *term frequency* (TF) and *inverse document frequency* (IDF). An important process in speeding up the retrieval of documents is that of indexing. Indexing documents require identification of their representative index terms. To this end, we apply basic information retrieval measures in the ink domain for stemming word form variations, detecting stopwords and relevance measurement. During indexing, the unique words in a document are identified and a table is created with them, where each index term points to all the instances of that term in the document. We use inverted data structure to organize clustered documents around the words they contain. Thereafter one can search only in the index table to locate any document containing the query.

In our problem of recognition-free retrieval, the identification of instances of a particular word becomes the problem of unsupervised clustering of the words in a document collection. Once we identify clusters that correspond to unique words, we can compute measures such as TF and IDF without explicit recognition of the words, as they need only the number of instances of a particular word in the document. We use a Minimum Spanning Tree (MST) based clustering algorithm for this purpose.

To carry out the clustering, we construct a complete graph for each document, where the word instances are the nodes and the distance between any two nodes is their matching distance computed in Section 4.3. The edges between instances of the same word (or variants of it) are expected to be shorter than those between instances of different words. We then compute the minimum spanning tree of the graph. The clustering algorithm proceeds by removing the longer edges from the tree, thus separating clusters of same words from others. The process is done in a recursive fashion, where each edge, e_{ij} is inspected for its relative length with respect to the neighboring edges, incident of nodes i and j. An edge with a large ratio of length to its neighbors is considered an inter-cluster edge and is removed. The process is repeated until the largest ratio, remaining in the graph, is within a threshold. The threshold is often set experimentally, and is dependent on the stability of the distance score computed above. For multi-page documents, the words in a page are clustered and the clusters across pages are merged to form the index, for efficiency purposes.

The above indexing and clustering approach can be directly applied to single-writer handwritten document collections. However, in the multi-writer case there is a great variations in the writing style among writers. Hence applica-

bility of the indexing approach depends on the availability of a dictionary of words, which can be synthesized and compared to the words in the documents to form clusters.

The statistical distribution of the words across documents is used for detecting stop words and also for computing a pseudo-TF/IDF. Once similar words are clustered, we analyze the clusters for their relevance. A measure of the uniformity of the presence of similar words across the documents is computed. We define the uniformity measure (η), as μ/σ, where μ is the mean, and σ is the standard deviation of the number of instances of the word per page. Note that the value of η will be high for those words which appear frequently on most pages. The inverse of the uniformity measure ($1/\eta$) is used as inverse document frequency. If a word is common in all the documents, this word is less meaningful to characterize any of the documents. Given a query, the corresponding handwriting is synthesized and the cluster corresponding to the words are identified. In each cluster, documents with highest occurrence of similar words are ranked and listed. The process of computation of relevance of the word as well as the counting of word frequencies does help to index document images similar to TF/IDF (Term Frequency/Inverse Document Frequency) for text indexing.

One of the main problems in document search using keywords is that of word-form variations. The exact word that is present in the document being searched is a variant of the keyword. There are two types of variations that are possible for a word: i) variation of the form of the word and ii) alternate word with the same meaning. For example, let the keyword being searched is *compute*. Word-form variants include words such as *computer*, *computing*, *computation*, etc., where as, alternate words could include *calculate* and *determine* in specific contexts. Addressing the second type of variation needs knowledge of the meaning of the word and its context and is beyond the scope of this work. We address the problem of dealing with the first class of variations, namely word-form variations. To match variants of a word, we note that word form varies mostly as changes in the prefix and/or suffix of a word, both in English and in Indian languages. Figure 8 shows an example of partial matching for the Hindi words having root word "gyan" (meaning *knowledge*). As can be seen, the word "Agyanta" (on the top of Figure 8) is matched with the word "gyan" (to the left of Figure 8). We thus modify our matching algorithm to minimize the penalty added on the total matching score as a result of the initial and final parts of a variant word (that is not matching with its base word). The net cost is finally used as a distance measure to group together similar words during clustering. This makes it easier to determine the relevance of words for representing documents during indexing.

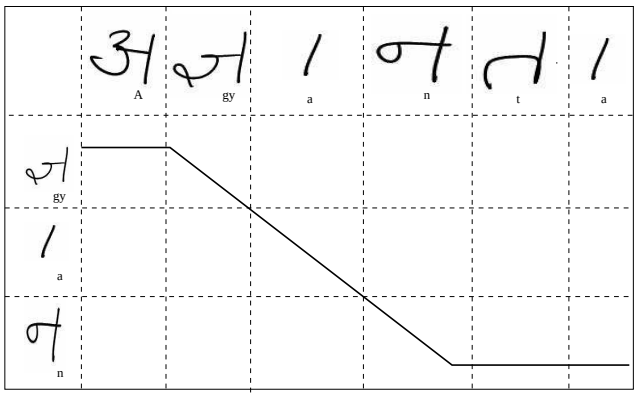


Fig. 8. Partial matching of the word 'gyan' with its variant 'Agyanta'.

5 Experimental Results

The proposed search and retrieval scheme is tested on online handwriting datasets in Indian scripts, primarily in Telugu. The data for the experiments were collected using an IBM Crosspad and a Tablet PC. For convenience of description, we divide our dataset into two: *Dataset 1*, consists of 100 pages of data from 20 different writers in Telugu script. The writers were chosen from varied backgrounds based on attributes such as script familiarity, educational qualification, age, and gender. The data contains over $12,000$ words. *Dataset 2*, consists of 15 pages each in Bangla, Hindi, Malayalam, Tamil, and Telugu scripts, collected from a total of 5 writers. The Telugu pages are common in both sets. The scripts mentioned above were selected for the following reasons: i) Telugu, which has one of the most complex layouts among all of the Indian languages, ii) Hindi, which has shirorekha, iii) Malayalam which has long and complex strokes, and iv) Bangla, which is similar to Hindi and v) Tamil, which has the smallest set of alphabets in Indian languages. A larger Telugu dataset (Dataset 1) was collected due to its complexity in spatial layout of characters.

5.1 Modeling and Synthesis of Handwriting:

The first set of experiments were to evaluate the performance of the synthesis algorithm. The experimental framework consists of i) an annotation tool that can annotate the handwritten data at the stroke level, ii) a handwriting model and synthesis module and iii) an ITRANS-based encoding module for processing the text input. The data was annotated manually using the annotation toolkit [29] in ITRANS [30]. In this toolkit, handwritten data can be displayed simultaneously with the annotation, which is dynamically updated as the user types the annotation.

Figure 9 shows two original samples of the Telugu word 'EdainA' from two

different writers and the corresponding synthesized words. As can be seen, the synthesized words for two different writers (Figure 9 (b) and (d)) are very similar to their original forms (Figure 9 (a) and (c)). As noted from the example, our model is able to generate natural and realistic words that are very close to the original ones. Also shown in Figure 9 (e)-(j) the word "manushya" in their synthesized and original forms in Malayalam, Bengali and Tamil scripts respectively.

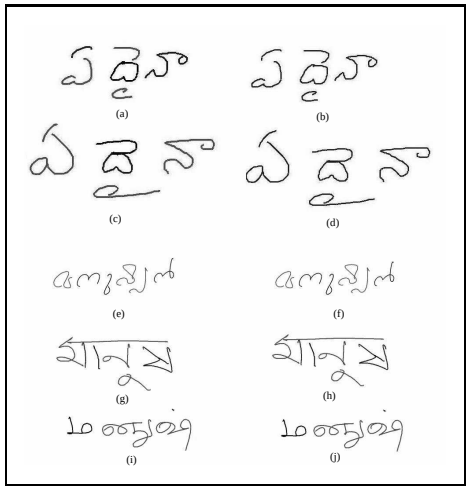


Fig. 9. Sample synthesized words and their original forms written by various writers in Telugu, Malayalam, Bengali and Tamil.

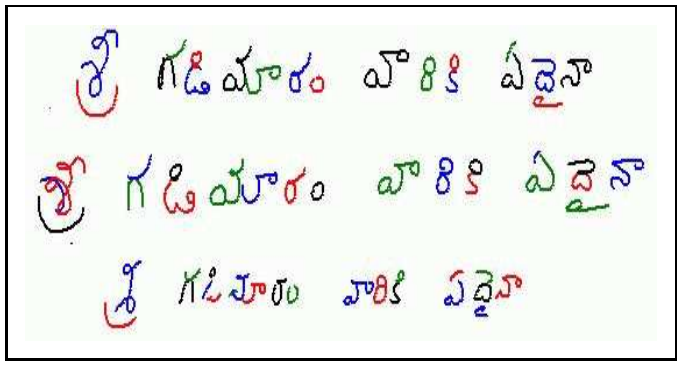


Fig. 10. Sample words synthesized for three different writers.

Figure 10 shows the synthesis results of the same set of words written by three different writers. Characteristics such as the inter-character spacing, relative size of loops and other matras (diacritical marks) of the three writers vary greatly. The mean trace model allows us to generate variations in handwriting of a specific writer, while maintaining the characteristics traits of the individual.

During synthesis the user input is typically in the form of either Unicode or ITRANS [30] encoded text, which are the two popular encodings for Indian language scripts. The input encoder converts the ITRANS or Unicode text string into a sequence of stroke classes. The specific stroke classes that are

generated depend on the character in the input as well as the writing style of the writer under consideration. The stroke selector module generates a sequence of strokes using the stroke model that are needed to generate the input text. Stroke alignment module is then used to compute the spatial positioning of the generated strokes based on the information from the alignment model and finally the word is rendered.

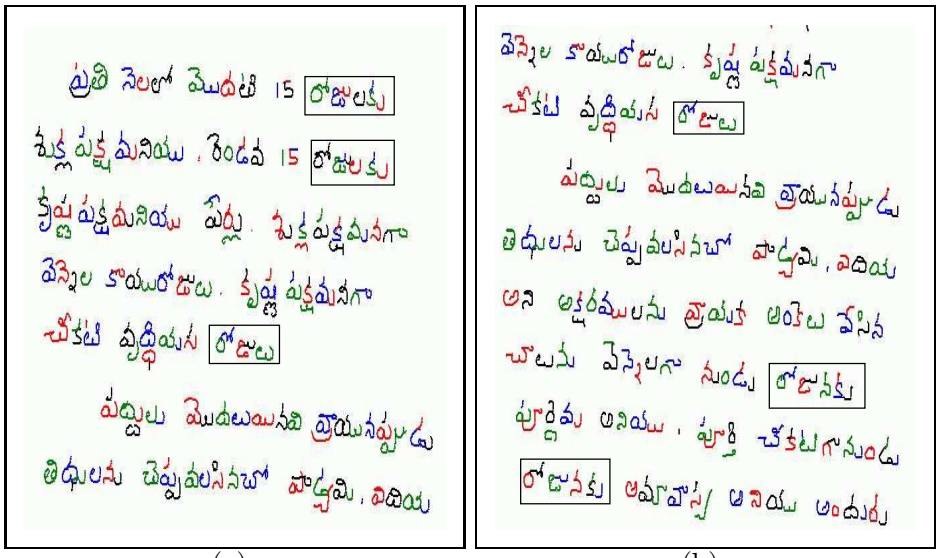
A quantitative evaluation of the synthesis algorithm is not feasible without a similarity measure between the generated word and the writing style of the writer under consideration. However, the writing style is a subjective quantity, that has not yet been quantified in the literature. Hence we use the effectiveness of the synthesis module in the retrieval experiments to measure the correctness of our synthesis algorithm.

To run the retrieval experiments, we identify a set of query words that are present in the documents in the database, and the corresponding online handwriting samples are synthesized. The features from the synthesized word are extracted as described before and compared against the words in the document. Those words with a matching (distance) score that is lower than a pre-specified threshold were assumed to match the query. As the threshold is relaxed, more documents are retrieved, which may contain documents that are less relevant, and as the threshold is tightened, the relevance of the retrieved documents increase, although the number of documents retrieved will decrease. Retrieved documents are then ranked based on the relevance to the query using TF/IDF scores.

Figure 11 shows the retrieved documents with the matching words placed inside the bounding box for the query in Figure 12. We note that all the documents that contained the word were retrieved and the top three matches were all relevant documents. Figure 11(a) has three matches where the first two matches are for the word *"roojulaku"* which is a variant of *"roojulu"*, while the last match is a direct match with the word *"roojulu"*. The match with the variant occurs due to the fact that the first sequence of strokes in all the three matches are similar and constitute the root word *"rooju"* (meaning *"day"*). Figure 11 (b) also has similar results for the word *"roojulu"* written by a different writer. In general, as can be observed from the retrieval result the search is able to retrieve words written by different writers based on the synthesized words, which shows the effectiveness of the synthesis based retrieval scheme.

Figure 12 presents a closer look at the retrieved words, which contain both writer differences and word form variations. The results clearly indicate that the partial matching scheme can effectively handle word form variations.

We also quantified the overall performance of the system on the complete database using measures such as precision and recall. The precision refers to



(a) (b)

Fig. 11. Search result for the input word "roojulaku" in Telugu written by (a) Writer 1, (b) Writer 2.

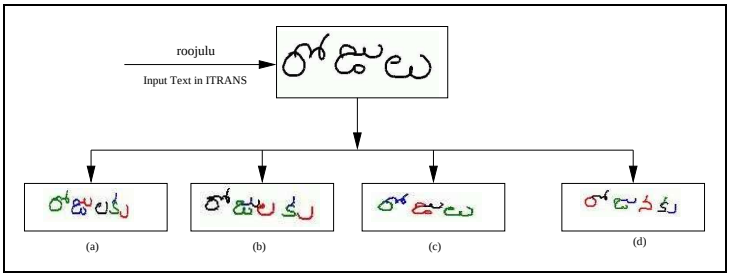


Fig. 12. Synthesized Telugu word *"roojulu"* and retrieved words (a,b) *"roojulaku"* by Writer 1, (c) *"roojulu"* by Writer 2, and (d) *"roojulaku"* by Writer 2.

the proportion of the retrieved entities that are relevant, while recall refers to the proportion of relevant entities in the database that were retrieved. For objective evaluation and computation of the precision-recall curve, we look at word level precision, and define a document as relevant if the word is matched correctly as per the ground truth. By varying the threshold used for matching, one gets various values for precision and recall. Precision and recall are basically inversely related such that as the threshold is made tighter, the precision increases, while the recall decreases, and vice versa.

Figure 13 shows performance results of word matching as a precision-recall curve. It is clear that even in a multi-writer databases, our algorithm is able to achieve very good retrieval performances. The only report in the literature that do recognition-free retrieval of handwritten documents are those by Lopresti and Tompkins [18] and Jain and Namboodiri [3]. The focus of both of them is single-writer document collections. For cross-writer retrieval, Lopresti and Tompkins [18] reports a precision of 2.8% at 50% recall and 1.6% at 100% recall on a dataset of 6240 words and an equal error rate of around 45% for single-

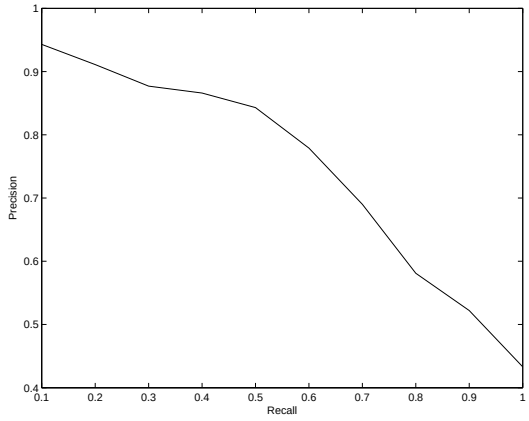


Fig. 13. Precision-vs-Recall plot for the experiment.

writer datasets. Clearly the algorithm works poorly on cross-writer retrieval. Jain and Namboodiri [3] achieves an equal error rate of around 10%, again for singe-writer dataset of $3,800$ words. In contrast, our algorithm achieves en equal error rate of 25.2% on a dataset of around $12,000$ words written by 20 different writers. A direct comparison of the error rates is not meaningful due to differences in the dataset used. In addition, our approach is able to accept textual queries instead of handwritten samples.

5.2 Multi-Script Synthesis and Retrieval

In addition to the multi-writer case, our algorithm is also able to handle multi-script databases using a single query, especially in Indian scripts. For multi-script synthesis, the input data is either translated or transliterated into the target script. Transliteration is possible for proper nouns in Indian languages since all the scripts share a common alphabets although the shape of scripts themselves are very different (see Figure 14), which is used for cross-lingual search as shown in Figure 14.

Once the query word is converted to the representation in the required script, a synthesizer in the corresponding script is employed to generate the corresponding handwritten data (see Figure 15).

Given an input word in ITRANS, one can generate the corresponding strokes in any of the Indian scripts and use the corresponding language model to generate the handwriting for a particular user. This capability is essential for applications such as cross-lingual search, where one would like to search for a word in different scripts, simultaneously.

Each of the generated query string is then compared with the key words to retrieve the corresponding documents. The results are identical to providing

Roman	Bengali	Hindi	Gujarati	Kannada	Tamil	Telugu	Malayalam
A	অ	अ	અ	ಅ	அ	అ	അ
AA	আ	आ	આ	ಆ	ஆ	ఆ	ആ
I	ই	इ	ઇ	ಇ	இ	ఇ	ഇ
II	ঈ	ई	ઈ	ಈ	ஈ	ఈ	ഈ
U	উ	उ	ઉ	ಉ	உ	ఉ	ഉ
UU	ঊ	ऊ	ઊ	ಊ	ஊ	ఊ	ഊ
R	ঋ	ऋ	ઋ	ಋ	ஏ	ఋ	ഋ
L	ঌ	ऌ		ಌ	ஏ	ఌ	ഌ
E	এ	ए	એ	ಎ	ஐ	ఎ	എ
AI	ঐ	ऐ	ઐ	ಏ	ஓ	ఐ	ഐ
O	ও	ओ	ઓ	ಐ	ஓ	ఒ	ഒ
AU	ঔ	औ	ઔ	ಔ	ஔ	ఔ	ഔ

Fig. 14. Shared alphabets of multiple scripts.

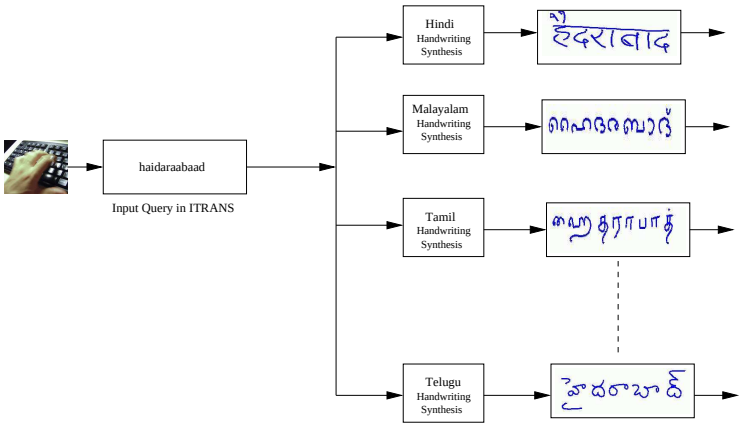
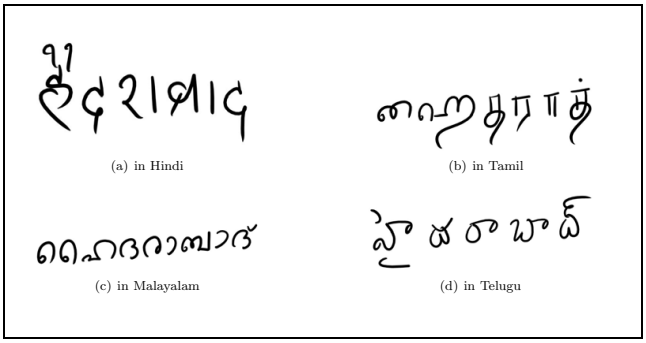


Fig. 15. Multi-script query expansion.

separate queries for each script, and hence is very effective. The query results on the multi-script database yields a precision of 94.3% at a recall of 89.1%. The results are comparable to the single script case and hence shows the effectiveness of the framework.

6 Conclusions

We have proposed a writer-independent recognition-free approach for retrieval of handwritten data in Indian language scripts. The proposed approach uses



(a) in Hindi (b) in Tamil (c) in Malayalam (d) in Telugu

Fig. 16. The word *haidaraabaad* synthesized in various Indian languages.

handwriting synthesis to do matching in the ink domain as opposed to the use of a recognizer. The framework also incorporates information retrieval measures such as TF/IDF to rank relevance of the retrieved documents. The approach also supports multi-lingual queries, which is especially useful for Indian languages. On the other hand, there is a need to further investigate on writer independence, stroke ordering, a complex stroke model and a way to quantify performance. We are currently working on modeling the handwritten data as a mixture distribution, which will be able to incorporate more writing variations within the data of a single user. Another interesting directions that we are currently pursuing is the study of stroke shape variations when it is in the proximity of other strokes or when the position of the stroke in the word changes.

References

[1] G. Russell, M. P. Perrone, Y. M. Chee and A. Ziq, Handwritten document retrieval, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, Korea, 2002, pp. 233–238.

[2] E. H. Ratzlaff, Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database, in: Proceedings of the International Conference on Document Analysis and Recognition, 2003, pp. 623–628.

[3] A. K. Jain and A. M. Namboodiri, Indexing and retrieval of on-line handwritten documents, in: Proceedings of the International Conference on Document Analysis and Recognition, Edinburgh, Scotland, 2003, pp. 655–659.

[4] T. Rath and R. Manmatha, Word image matching using dynamic time warping, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2003, pp. 521–527.

[5] S. N. Srihari and Z. Shi, Forensic handwritten document retrieval system, in: Proceedings of the International Workshop on Document Image Analysis for Digital Libraries, Palo Alto, CA, 2004, pp. 188–192.

[6] I. Kamel, Fast retrieval of cursive handwriting, in: Proceedings of the 5th International Conference on Information and Knowledge Management, Rockville, MD, 1996, pp. 91–98.

[7] A. Balasubramanian, Million Meshesha and C. V. Jawahar, Retrieval from document image collections, in: International Workshop on Document Analysis Systems, Nelson, NewZealand, 2006, pp. 1–12.

[8] S. N. Srihari, Recognition of handwritten and machine-printed text for postal address interpretation, Pattern Recognition Letters 14 (1993) 291–302.

[9] C. Cracknell and A. C. Downton, A handwritten form reader architecture, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, Korea, 1998, pp. 67–76.

[10] Rafael Palacios, Amar Gupta, Patrick Shen-Pei Wang, Handwritten bank check recognition of courtesy amounts, International Journal of Image and Graphics 4 (2) (2004) 203–222.

[11] J. Hu, M. K. Brown and W. Turin, HMM based online handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (10) (1996) 1039–1045.

[12] S. D. Connell and A. K. Jain, Writer adaptation for online handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 329–346.

[13] S. N. Srihari, S. H. Cha, H. Arora and S. Lee, Individuality of handwriting, Journal of Forensic Sciences 47 (4) (2002) 1–17.

[14] I. Guyon, Handwriting synthesis from handwritten glyphs, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, Colchester, England, 1996, pp. 140–153.

[15] T. Varga and H. Bunke, Generation of synthetic training data for an HMM-based handwriting recognition system, in: Proceedings of the International Conference on Document Analysis and Recognition, Edinbugh, Scotland, 2003, pp. 618–622.

[16] M. Helmers and H. Bunke, Generation and use of synthetic training data in cursive handwriting recognition, in: Proc. 1st Iberian Conference on Pattern Recognition and Image Analysis, 2003, pp. 336–345.

[17] Y. Zheng and D. Doermann, Handwriting matching and its application to handwriting synthesis, in: Proceedings of the International Conference on Document Analysis and Recognition, 2005, pp. 861–865.

[18] D. Lopresti and A. Tomkins, On the searchability of electronic ink, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, Taipei, 1994, pp. 156–165.

[19] C. V. Jawahar and A. Balasubramanian, Synthesis of online handwriting in indian languages, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2006.

[20] W. Guerfali and R. Plamondon, The Delta LogNormal theory for the generation and modeling of cursive characters, in: Proceedings of the International Conference on Document Analysis and Recognition, 1995, pp. 495–498.

[21] R. Plamondon, A Kinematic theory of rapid human movements, Part I. Movement representation and generation, in: Biological Cybernetics, Vol. 72, 1995, pp. 295–307.

[22] Y. Singer and N. Tishby, Dynamical encoding of cursive handwriting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1993, pp. 341–346.

[23] S. Kondo, A model of handwriting process and stroke structure of character figures, in: Computer Recognition and Human Production of Handwriting, 1989, pp. 103–118.

[24] L. R. B. Schomaker, A. J. W. M. Thomassen and H. L. Teulings, A computational model of cursive handwriting, in: Computer Recognition and Human Production of Handwriting, 1989, pp. 119–130.

[25] D. H. Lee and H. G. Cho, A new synthesizing method for handwriting Korean scripts., International Journal of Pattern Recognition and Artificial Intelligence 12 (1) (1998) 46–61.

[26] O. Velek, C.L. Liu and M. Nakagawa, Generating realistic kanji character images from on-line patterns, in: Proceedings of the International Conference on Document Analysis and Recognition, 2001, pp. 556–560.

[27] O. Vectomova and W. Ying, A study of the effect of term proximity on query expansion, Journal of Information Science 32 (4) (2006) 324–333.

[28] J. Wang, C. Wu, Y. Q. Xu, H. Y. Shum and L. Ji, Learning-based cursive handwriting synthesis, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2002, pp. 157–162.

[29] A. Bhaskarbhatla, S. Madhavanath, M. Pavan Kumar, A. Balasubramanian and C. V. Jawahar, Representation and Annotation of Online Handwritten Data, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2004, pp. 136–141.

[30] ITRANS, Indian language transliteration package, at: `http://www.aczoom.com/itrans/` (2001).