Nearest Neighbor based Collection OCR

Pramod Sankar K.
Center for Visual Information Technology
IIIT-Hyderabad, India
pramod_sankar@research.iiit.ac.in

C. V. Jawahar
Center for Visual Information Technology
IIIT-Hyderabad, India
jawahar@iiit.ac.in

R. Manmatha
Dept. of Computer Science,
University of Massachusetts,
Amherst, MA-01003
manmatha@cs.umass.edu

ABSTRACT

Conventional optical character recognition (OCR) systems operate on individual characters and words, and do not normally exploit document or collection context. We describe a *Collection OCR* which takes advantage of the fact that multiple examples of the same word (often in the same font) may occur in a document or collection. The idea here is that an OCR or a reCAPTCHA like process generates a partial set of recognized words. In the second stage, a nearest neighbor algorithm compares the remaining word-images to those already recognized and propagates labels from the nearest neighbors. It is shown that by using an approximate fast nearest neighbor algorithm based on Hierarchical K-Means (HKM), we can do this accurately and efficiently. It is also shown that profile based features perform much better than SIFT and Pyramid Histogram of Gradient (PHOG) features. We believe that this is because profile features are more robust to word degradations (common in our documents). This approach is applied to a collection of Telugu books - a language for which no commercial OCR exists. We show from a selection of 33 Telugu books that starting with OCR labels for only 30% of the collection we can recognize the remaining 70% of the words in the collection with 70% accuracy using this approach. Since the approach makes no language specific assumptions, it should be applicable to a large number of languages. In particular we are interested in its applicability to Indic languages and scripts.

1. INTRODUCTION

There are a number of projects creating digital libraries from scanned books. These include Google Books, the Universal Digital Library (UDL) [1] and the Digital Library of India (DLI) [2, 25]. A large percentage of these collections are in non-Latin scripts. In DLI alone, 50% of the content comes from Indian language books, for which recognition technology has not fully matured. Though a few research prototype OCRs are available for Indic scripts, there is no end-to-end system that can recognize any given document image [10].

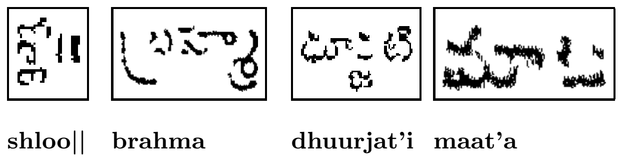

Figure 1: Examples of words that are very hard to recognize with a conventional OCR. Inspite of heavy degradations, our framework correctly predicts the recognized text for these words.

Our collection is a challenging set of scanned Telugu books. Figure 1 shows some of the more difficult examples of words present in the collection. The characters are broken, or smeared and show a number of artifacts. Traditional OCR systems recognize individual characters or words. They do not recognize documents or collections. The fonts in a document are often very similar. Hence, recognizing a word would benefit from its reappearance in other parts of the document. This is especially true if the document contains many pages. Books are one example of documents containing a large number of pages with text in mostly the same font.

We propose a novel and fast *Collection OCR* which applies a retrieval based approach to recognition. We assume that a part of the collection has been initially recognized using an OCR or human annotations. The system then segments and computes features over all word-images. A fast approximate nearest neighbor technique based on Hierarchical K-Means is used to determine, for each test image which training image is closest. The label is propagated from that training image.

We demonstrate for a set of 33K words which occur multiple times from a collection of Telugu books that given a training set of 30% of the data, then 70% of the remaining data can be recognized with an accuracy of 70% using a single pass of nearest neighbor recognition. The kinds of images recognized include those shown in Figure 1. We are unaware of any conventional OCRs which can recognize such images. Experiments are also performed on the entire 33 book Telugu dataset. Since no assumptions are made about languages, the technique should be applicable to a wide variety of different languages. In particular, our interest is in applying it to Indic languages for which commercial OCRs do not exist.

We also show that robustness and accuracy depend on appropriate features. Specifically, using discrete Fourier representations of profile features works better than gradient representations such as SIFT and Pyramid Histogram of Gradients (PHOG). This is surprising given the success of gradient features in recent image recognition and retrieval work [23] and the reported advantages of gradient features over profile features for handwritten word spotting [32].

The *Collection OCR* approach is fast. For example, for 3000 pages it takes 5.5 hrs to segment the word-images, 66 hours to extract the profile features, and about 9 min to do the recognition. In comparison a conventional OCR will take about 50 hrs (if we assume 1 min per page). The segmentation and feature extraction may be optimized for better performance. So, even when combined with a conventional OCR for recognizing an initial training set, it is practical to apply it to large collections.

2. RECOGNIZING DOCUMENT COLLECTIONS

A major hurdle in recognizing printed Indian language document images, is the difficulty encountered in accurate segmentation of components [10, 24]. In particular for the Telugu script, the vowel modifier is attached to the consonant. The segmentation of a character into its constituent consonant and vowel is quite tricky. Degradations in document images often result in multiple segments for a single character. Since neither of these segments appears like a proper character, the classifier rejects them all. Even modern English OCR systems fail when the assumptions they make are violated. For example, crossed out words, touching lines and broken characters cause OCR failures (see [31] for many examples of OCR failures). Post-processing modules that correct OCR errors rely heavily on language models, which are hard to build for Indian languages [5].

Given these challenges, it is not surprising that Telugu and other Indic languages and scripts have no commercial OCR systems. Experimental work on recognizing Telugu script was presented in [12, 22] and the word recognition accuracy is about 70% on much cleaner data than that presented here. Building recognizers for the kinds of degraded data present in our collections (see Figure 2), thus requires thinking out of the box.

Our approach has been motivated by work in word spotting in handwriting [28, 30] and print [4, 13, 26]. Word spotting approaches usually match entire word-images. Word-images contain a lot of context, and hence it is easier to match them than character images. This is also advantageous as difficult segmentation decisions are avoided. Even in situations where some of the characters in the word are degraded, matching to other instances of the same word may be accomplished. While whole word recognition schemes also try to recognize the entire word [17, 15] [1], there are important differences. These schemes would have to recognize the actual degraded word, while we can transform it to the easier problem of matching the degraded word with another instance of the same word. There is also an important distinction between conventional OCR training and the notion of a document or collection OCR. In conventional OCR training, all the training examples are distinct from the test data and may hence be different in appearance from it. In our case the training and test examples on the other hand are both likely to be similar (modulo noise). What both word spotting and whole word recognition share is the avoidance of character segmentation.

[1]Both papers cited used it for handwriting not print

Traditionally, most word spotting approaches match pairs of words and then cluster them. Each cluster is then labeled. A common approach to image matching is to use dynamic time warping which is accurate but very expensive. Some word spotting approaches use dynamic time warping and skip the clustering stage altogether. However, this does not change the essentially slow nature of word spotting for large datasets. Our work takes advantage of recent advances in fast approximate nearest neighbor techniques for image recognition and retrieval. In particular, we use an approach called Hierarchical K-Means (HKM) [23] which was shown to have good performance in other image retrieval tasks. Our experiments show that given the word segments and features, the actual indexing and nearest neighbor operations required for recognition only take about 9 min for 3000 pages.

Our approach requires that a portion of the collection has been labeled in some way. There are a number of ways this can be done. For example, there may be a "crude" OCR which does this. The OCR may use a rejection strategy so that it recognizes some words accurately. Whenever it has some doubts, it decides not to recognize. The word accuracy of such an OCR may be poor - we assume 50% or less.

An alternative to an OCR is to manually label some portion of the collection. Normally, manual labeling is expensive, but reCAPTCHAs [35] provide one approach to generating labels for training examples on the cheap. In a reCAPTCHA, pairs of word-images are presented for recognition by humans. The label for one of the words is known while that of the other is not. If the human recognizes the known image, it is assumed that their judgement about the other image is also valid. Judgements by multiple humans are integrated to decide the correct label for the word. reCAPTCHAs are normally used after the OCR process to improve accuracy. It is claimed that a high recognition rate can be achieved. The labor provided by humans is essentially free as reCAPTCHAs are required by many websites for sign on. While they have been used for improving OCR accuracy, the same idea may be applied to bootstrap the recognition process to provide some number of recognized outputs. In this paper, we are agnostic to what process is used to bootstrap and provide the initial recognition results.

2.1 Overview of the Algorithm

We now briefly describe the procedure for recognizing book collections (although the procedure may also be followed at book level). First, all the books are automatically segmented into words and features are computed over them (see later sections). A random set of word-images is collected for which text-labels are obtained.

Given the feature representation, we build an HKM (see Section 4.2) over the labeled dataset. Each of the unlabeled

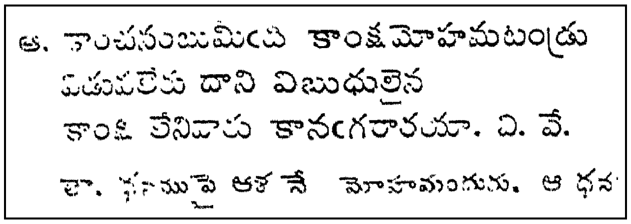
ఉ. [illegible] కాంక్షమోహమటంచ్రు
పడుపరేపు దాని విబుధుత్రైన
కాంక్ష తేనివాపు కానంగరాదయా. ఐ. వే.
తా. [illegible]ముపై ఆశ నే మోహమాంగును. ఆ ధన

Figure 2: Example document images. Notice the considerable degradations, cuts etc. in the passage. An OCR would not be able to recognize the characters in this situation. We propose the matching of word-images in lieu of recognizing the characters.

word-image's features is looked up in the HKM tree and is assigned to the nearest cluster. The label of the cluster centroid is assigned to the given test word-image. In cases where the labeled dataset does not contain the exemplar for a given word, such a word would still be assigned to a cluster. But, the distance between this unlabeled word and the cluster centroid would ideally be large. To handle such cases, we follow a refuse-to-label scheme, whenever the distance between the test word and the cluster centroid is greater than a certain threshold, τ. The overall algorithm followed is described below: Let us call the books $B_1, B_2, ..., B_{33}$, each containing N_i number of pages.

1. From each book B_i, choose a random set of pages P_i that shall be labeled. The remaining pages in the book are called T_i, the test data.
2. Obtain labels for words in every P_i, either manually or through an OCR.
3. Build a HKM over $\forall_i W_i$, where W_i is the list of words in pages P_i.
4. Lookup each word $W' \in T_i$ against the HKM; obtain the Nearest Neighbor $NN_{W'}$ and distance to NN, $d_{W'}$
5. If $d_{W'} < \tau$, label the word with the text equivalent of $NN_{W'}$; ignore otherwise

3. PREVIOUS WORK

Building OCRs for Indian languages is a very challenging task, owing to the large number of classes and the complexity of scripts [10, 24]. OCRs for Telugu script were previously presented in [12, 22]. Word based recognition has been suggested for handwritten documents [15, 17], but is not commonly used for print. Natarajan *et al.* [20] use a HMM to model characters, which are concatenated to form a word model, which is then recognised as a whole.

On the other hand, recognition-free approaches such as *Word Spotting* were proposed to enable search over handwritten [3, 28, 30] and printed documents [4, 13, 26]. Pramod and Jawahar [26] use an automatic annotation approach to assign text labels to a large collection of document images. However, they annotate for only those words that are considered useful for retrieval.

There is much less research focused on OCR at the book level. Rasagna *et al.* [27] cluster word images in an entire Telugu book using locality sensitive hashing and use this to correct character labels based on majority voting. Xiu and Baird [36], measure the disagreements between OCR results and language models using mutual entropy across a passage. This measure is then used to correct frequent OCR errors, or to add new words to the language model. The procedure was demonstrated on about 50 pages. Neeba and Jawahar [21] use a word-image matching scheme to verify OCR outputs and in turn improve the character recogniser.

Adapting the OCR to the data has been an important approach to OCR document with novel styles. Unsupervised learning of character appearances was demonstrated in Ho and Nagy [11]. Style adaptation of OCRs has been suggested by Sarkar and Nagy [33], and Mathis and Breuel [18]. This essentially consists of modifying the parameters of the OCR (for example the mixture of Gaussians used in the generative models) based on the data. We are not aware of any specific applications to book level collections.

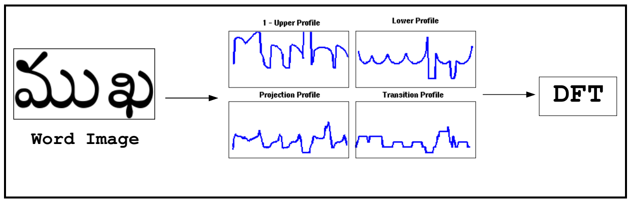

Figure 3: Extraction of Profile-Features and their fixed length versions. For each word image, Upper, Lower, Projection and Transition Profiles are computed. These features are run through a DFT, and the top 84 coefficients chosen for each.

4. WORD MATCHING

In this work, our aim is to label word-images by matching them with a set of labeled examples. We shall focus on matching word-images as a whole. Toward this end, appropriate features and matching schemes need to be identified to maximise performance.

4.1 Word-Image Representation

We examine three types of features. Profile features have been previously used for representing word-images [27, 28]. Gradient based features have been claimed to be better than Profile features for handwritten word-images [32]. So we use two popular features SIFT and dense PHOG features. In the case of SIFT and PHOG, we shall use a Bag-of-Words [34] model representation to reduce the time required to match word-images.

Profile Features. Profile features were popularised by Rath and Manmatha [28], where these features were used to represent words from handwritten documents. The profile features we extract include (see [28] for more details):

- The Projection Profile which counts the number of ink pixels in each column.
- Upper and Lower Profile measures the number of background pixels between the word and the word-boundary

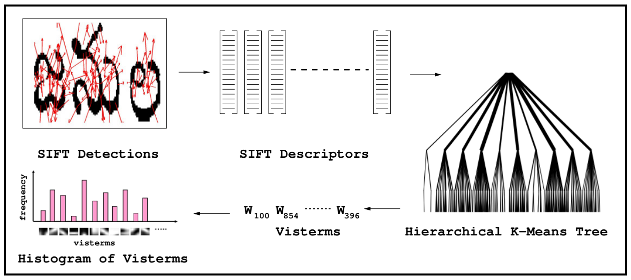

Figure 4: Representation of SIFT features in the Bag-of-Words model. Extracted SIFT features from a word image are assigned to the closest *visterm*. The word is then represented as a histogram of its constituent visterm occurrences

- Transition Profile is calculated as the number of ink-background transitions per column.

Profiles for an example word are shown in Figure 3. Each profile is a vector whose size is the same as the width of the word. The dimensionality of the feature, therefore, varies with the word used. One approach to matching features with different feature-lengths, uses Dynamic Time Warping (DTW) [4, 28].

Profile + DFT. Fixed length descriptions for the Profile features can be obtained by computing a Discrete Fourier Transform (DFT) of the profiles [14, 29]. The noisy higher order coefficients of the DFT are discarded, resulting in a robust representation for the word-images. We use 84 Fourier coefficients for each of the profile feature. With this representation, word-images can be matched by comparing feature vectors using a distance metric, such as the Euclidean distance.

SIFT + BoW. Another type of features use a point-based representation for the word-images. SIFT [16] (Scale Invariant Feature Transform) has proved to be a very robust interest point detector and feature descriptor for many computer vision tasks. The interest point detector is based on the difference between multi-scale Gaussians of the given image. The scale-invariance property of SIFT and its high repeatability across various affine transformations makes it a good candidate to be used in document images [3]. SIFT interest points and features are detected from each word-image - an example is shown in Figure 4.

Following the Bag-of-Words (BoW) [34] approach, these features are vector quantized using K-Means clustering over a large set of SIFT features. Each feature in a word-image is represented as the index of the cluster center it is closest to, known as the *visterm*. The word-image is then represented as a histogram of the occurrences of each visterm. For example, if an image has $F_1, F_2, ..., F_{300}$ features, and say the visterm size is 1000. Each feature in the image is assigned to the closest visterm, say $W_{100}, W_{200}, W_{150}, W_{240}, ..., W_{100}$. The number of times a visterm occurs in a word, is stored in a histogram of length 1000. The histograms are then normalised by number of features in the word image. Given this representation, two words are compared by finding the Euclidean distance between the corresponding histogram feature vectors. Since the geometric configuration of the features is ignored and only their incidence is taken into account, it is called a Bag-of-Words representation of the image akin to the representation in text retrieval. This process is depicted in Figure 4. Such a representation reduces the time required for explicit match of SIFT features, when matching two images.

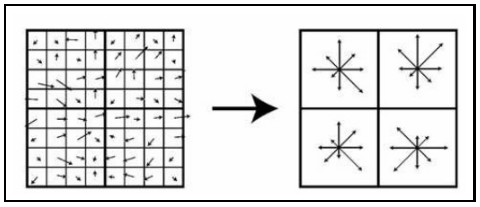

Figure 5: A depiction of HoG feature extraction. A 2x2 grid is shown for clarity. In the implementation, a 4x4 grid is used.

PHOG + BoW. Unlike SIFT, which is a sparse detector, one could use a dense representation with similar descriptors. This is performed using the Pyramid Histogram of Oriented Gradients (PHOG) [6]. A single level HoG [7] was recently used for handwritten documents by Rodriguez and Perronnin [32]. However, a pyramidal HoG at multiple scales was shown to be much more effective in object recognition tasks [6].

In this technique, a fixed size window is moved across the word-image. At each instance of the sliding window, a HoG descriptor is computed similar to the one shown in Figure 5. Once the entire word-image is scanned by the window, the window size is doubled and the process repeated. The process is stopped when the size of the window exceeds the smaller dimension of the word-image. The set of features thus extracted are vector quantized, and the word-image is represented as a histogram of the occurrences of each visterm.

In detail, the initial window size is 16×16 pixels, which is moved across the image by 8 pixels. The 16×16 window is divided into a grid of size 16 with cells of size 4×4 pixels each. The gradients at each pixel are quantized into one of 8 orientation bins. The oriented gradients in each grid are accumulated, and the histograms for all the grids are concatenated together. With this setting, we obtain a 128 dimension representation for each window. In the next iteration, the window size is doubled to 32×32, and moved across by 16 pixels and so on. To ensure that the feature length for PHOG is fixed, all the word-images are initially scaled to a standard size.

4.2 Efficient Nearest-Neighbour Search

Traditional nearest neighbor algorithms are slow, especially when the datasets are large. However, recent work has led to fast approximate nearest neighbor algorithms. It is known that in the limit, the error rate for NN algorithms can never

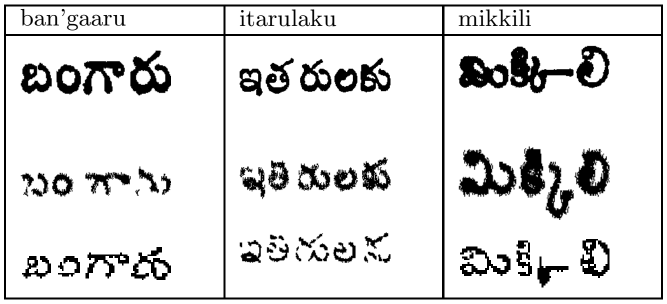

Figure 6: Examples from the groundtruth dataset. The text label for the words is given on top.

exceed twice the Bayes error [8].

In an NN classifier, whenever a word-image matches a particular labeled example better than any other labeled example, there is a high probability that they share the same label. Suppose we have M labeled images and N unlabeled images, the NN classifier has a time complexity of $O(N \cdot M)$. In many cases, such as when the feature length is large or the distance computation is expensive, the running time of this algorithm may be large. This process is speeded up by several orders of magnitude using approximate nearest neighbor algorithms. We will look at one such algorithm based on Hierarchical K-Means (HKM) [23].

Let us assume that we were able to cluster all the word-images using K-Means (with a very large K - say 100,000). Then, any two word-images which are part of the same cluster are more likely to have the same label. We can also do this for word-images which were not part of the original clustering process. We just find out which cluster they belong to and the label of that cluster is the label of the test word. Note, that we need a lot of clusters since we want to have a single label per cluster. It is fine for two clusters to share a label but not for the same cluster to have different labels. It would take a long time for K-Means to cluster, when there are many clusters. For example, it would take 31 yrs to do K-Means clustering for a million words.

Hierarchical K-Means is a way to quickly approximate K-Means. The idea [23] is that a small number of clusters - say B - are created at the top level. B is known as the branching factor. Then each of these B clusters is expanded to B more clusters giving B^2 clusters at the second level. This process is repeated up to a certain depth D so that there are B^D clusters or leaf nodes at depth D. If we take a new point, to find out which cluster it belongs to, it takes $B \cdot D$ comparisons unlike traditional K-Means which would take B^D comparisons. For example, if $B = 10$, $D = 6$ then there are a million leaf nodes. K-Means requires 10^6 (a million) comparisons while HKM only needs 60 comparisons. To build the entire HKM tree for a dataset of size N requires $O(N \cdot B \cdot D)$ time while K-Means would require $O(N \cdot B^D)$. If $N =$ a million word image features, this means K-Means would require $O(10^{12})$ and HKM $O(600)$. On a modern desktop processor the difference is 31 yrs for K-Means vs 13 hrs for HKM. For our experiments here we usually chose $D = log_B(N)$.

Feature	Distance Measure	Accuracy
Profile Features	DTW + Backtrack	81%
Profiles (Scaled)	Euclidean	68.8%
Profiles + DFT	Euclidean	79%
SIFT (1K visterms)	Euclidean	26.4%
SIFT (100K visterms)	Euclidean	8%
PHOG (100 visterms)	Euclidean	20.8%
PHOG (1K visterms)	Euclidean	41.8%
PHOG (100K visterms)	Euclidean	14.12%

Table 1: Word Recognition accuracy across various features and matching schemes

5. RESULTS AND DISCUSSION

5.1 Dataset

The recognition system is built using a collection of 33 Telugu books. These books are obtained from the Digital Library of India [2]. The collection consists of 3269 pages, with more than 269,000 words. All the images are provided to us in bitonal format. A segment of an example image is shown in Figure 2 and as can be seen there are many cuts and degradations of the characters. To begin with, the document images are segmented into words using a profile-based segmentation algorithm. The segmented words are stored separately, which we shall henceforth refer as a word-image.

We built a groundtruth dataset of 33,000 word-images. Each of these word-images is labeled with the corresponding text label. The label set consists of 1000 Telugu words represented in the Latin script using the OmTrans transcription scheme. The number of occurrences of each label in the groundtruth dataset varies from 5 to 500. The size of the word-images range from 30×30 to 500×300 pixels. The dataset contains degraded images, and considerable variation in font and print style. There is also a certain amount of noise in the word-images. Examples of such words in the dataset are shown in Figure 6. We shall use this dataset to evaluate the various design choices in the rest of the paper.

5.2 Feature Selection

The goal in this section is to identify the right features to match word-images. The groundtruth dataset is divided into two sets: $Train$ and $Test$, with a random 50:50 split. The $Train$ set will be assumed to consist of labeled examples while the $Test$ set is considered to be unlabeled (the labels will only be used for evaluation in the latter case). The classifier is an NN classifier, so that for each instance in the $Test$ set, the label of the closest exemplar in the $Train$ set is assigned. Since the label for each $Test$ data point is present in the $Train$ set, we do not use the refuse-to-predict scheme in these experiments.

The summary of feature selection results is given in Table 1. The first feature we consider is the classical Profile features. These features are of unequal length across the dataset. The matching in such cases is performed using DTW (Dynamic Time Warping). The score from the DTW matrix is normalised by the length of the backtrack path of the DP array. The disadvantage of DTW is the high time complexity. Recognition of the $Test$ set alone takes around 75 hours. This expense can be avoided by using a fixed-length descrip-

Matching Scheme	Accuracy	Time
NN Classifier	79%	4 hours
HKM ($B = 8$)	76.5%	31 secs
HKM ($B = 32$)	75.1%	38 secs

Table 2: Word Recognition accuracy with Hierarchical K-Means based matching

tion for each image, so that the Euclidean distance may be used to compare the word-images.

A fixed-length representation can be obtained by scaling all word-images to the same size and extracting profile features from them. However, this changes the aspect ratio of words. Another method is to transform the features to the DFT domain and picking a fixed set of important coefficients from the transform domain. With either representation, the time for recognizing the $Test$ set is about 4 hours. We also look at SIFT and PHOG based representations for word-images.

As Table 1 shows the best performing feature was the Profile feature with DTW matching. This is closely followed by Profiles + DFT with Euclidean distance matching. Surprisingly, the gradient features - SIFT and PHOG - perform rather poorly inspite of their great success with generic vision tasks. Rodriguez and Perronnin [32] show that for word spotting in handwriting, gradient features work better than profile features. However, our results do not agree with their conclusion. We suspect that this is because of the large amount of degradations present in our document images which drastically affect the oriented gradients. The profile features are thus more robust to degradations. The DTW algorithm is too expensive to run and not practical and hence we perform the remaining experiments with the Profiles + DFT with Euclidean distance comparison.

5.3 Performance of HKM

Our next experiment evaluates the performance of Hierarchical K-Means based word recognition. We use the HKM version of the FLANN software [19]. The algorithm first traverses the HKM tree and add the unexplored branches in each node along the path to a priority queue. It then finds the closest center in the priority queue to the given query, and uses this node to restart the traversal. The process is stopped when a predetermined number of nodes are visited.

The result from this experiment is given in Table 2. As can be seen, HKM gives a tremendous speedup of over 500 times as compared to a brute-force NN classifier. This comes at very little loss in accuracy due to the indexing scheme.

5.4 Effect of Labeled Data Quantity

One of the important questions we address in this work is what amount of labeled word-images, is required to reliably recognize unlabeled word-images. Unlike a 50:50 split that was used in the previous experiments, we shall vary the $Train$, $Test$ proportions here. We stick to the Profiles + DFT features and the Euclidean distance based NN classifier as well as the HKM based approaches with the two parameters in Table 2. The recognition accuracy for different percentage of training data is shown in Figure 7.

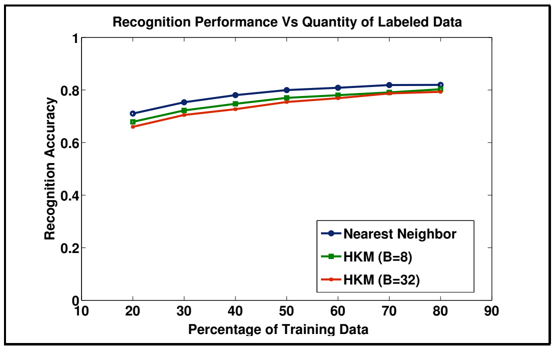

Figure 7: Graph of % Training data Vs Recognition performance

For the NN classifier, it can be seen that for a labeled dataset of as small as 20%, the recognition performance is a respectable 71%. The performance improves with additional data until it tapers around 76.5% (for the 50:50 split), after which there is no significant improvement. For the HKM approach with $B = 8$, we can achieve 70% accuracy with just 30% of the data.

6. RECOGNIZING IN BOOK COLLECTIONS

Our final system is built on all the words from the 33 book dataset. From the previous experiments, we infer that in the presence of 30% labeled words, we achieve about 70% recognition accuracy of unlabeled words. Keeping with this observation, we use labels for one-third of the pages in each book. It is too expensive to do a word-by-word transcription of this size (100K words). Instead, we use a transcription done at the page level and not at the word level. Thus, to obtain word level annotations, the transcribed text is automatically aligned with the word-images (which are segmented from the page-image). Such alignments can be done as in [9]. In cases where there are segmentation errors, the alignment between the word-images and their labels may be erroneous. However, we do not correct these mistakes as the effort is prohibitive.

An HKM is built over the 100K labeled images, for the Profiles + DFT features. The NN for each test point is done using a lookup in the HKM and the label propagated to the test image from the NN word-image. The time taken for building the HKM over the 100K labeled words was 93 seconds, while the recognition of the remaining 200K words took 112 seconds.

Example results from this recognition method are given in Figure 8. As we observe, a large number of words are correctly recognized inspite of heavy degradations. We benefit from the presence of similar looking labeled examples, which allows our *Collection OCR* to yield good results. In cases where our procedure fails, the features are unable to distinguish between different words. For example, consider the results for the word *man'chi* (row 2) of Figure 8. The first erroneous word is *man'ta*, which differs in the last (third) character. Notice the strong similarity between the first correct and the first erroneous word. In the second erroneous example, the second half of the word (*lin'chi*) matches with the label, and is hence misclassified. Besides errors caused

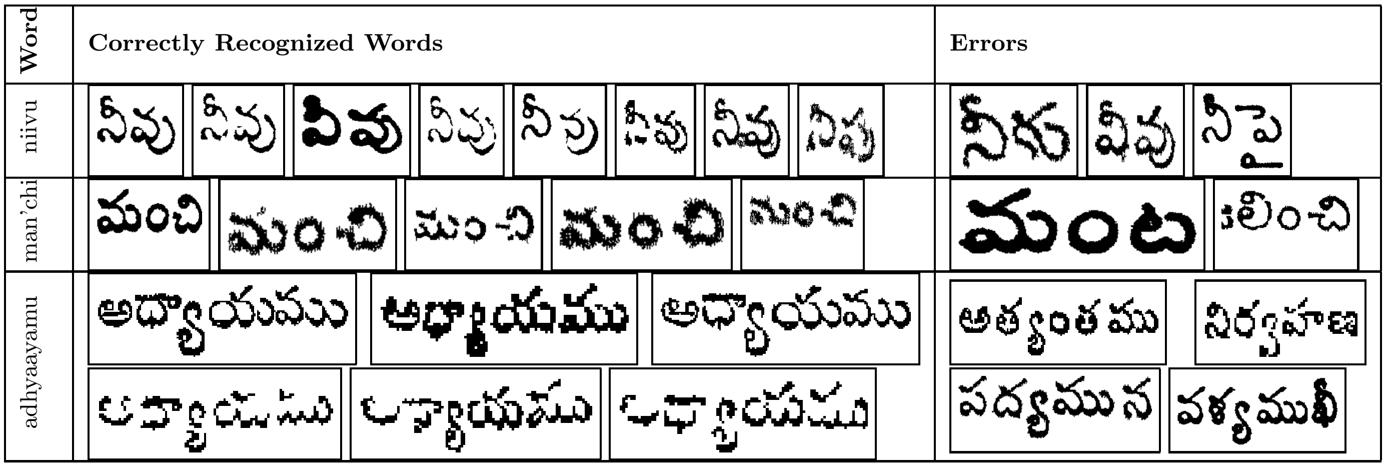
Word
Correctly Recognized Words
Errors
niivu
man'chi
adhyaayamu

Figure 8: Example recognition results from the book collection. Notice the severe degradations in some of the correctly recognized words. We recognize these words where OCR easily fails. Most of the errors are due to similar looking words (or part of), or due to erroneous labeled data.

by very similar looking words, we observe errors due to erroneously labeled training images.

We follow a refuse-to-predict scheme, to avoid wrongly labeling word-images which do not have any similar labeled exemplar. This is enforced by thresholding the NN distance. The threshold τ determines the quality of recognition. With a larger value of τ, a large number of words would be labeled which might result in many errors. With a smaller value of τ, it is more likely that a lesser number of words would be recognized, but their labels would be more accurate. By eyeballing the results for a handful examples, we set a threshold which gives recognition results of acceptable accuracy. At such a threshold, about 50% of the unlabeled data was recognised, the labels for the rest were not predicted. The reason for such a high percentage of rejection, is because the labels for a large number of words are not included in the 30% of labeled words. This would include a significant number of unique words in the collection. We discuss in the next subsection how such words may be handled.

6.1 Recognizing the Un-Labeled Images

Some fraction of word-images remain unrecognized at the end of the *Collection OCR* phase. These words do not contain a labeled exemplar in the training data. To address this issue, we could generate synthetic exemplars for a large vocabulary using text rendering. These exemplars can then be matched with the unlabeled images, to identify the NN exemplar and in turn the recognised text.

To evaluate this step, we generated exemplars for the 1000 word dataset (see Section 5.1), using a standard font for Telugu. The recognition accuracy over the 33K labeled dataset using these exemplars was about 33%. Our font (a modern Telugu font) is not a good match to the older fonts of the collection. We believe it is possible to improve on this substantially by choosing better fonts for rendering. We shall leave this exploration for future work.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a *Collection OCR* which recognizes words by exploiting their similarities in the collection. Given labels for a fraction of the words, word-images are matched using an efficient nearest neighbor algorithm (HKM). Labels are propagated from these nearest neighbors. It is also shown that profile features are more robust compared to gradient features for matching printed word-images. In future work, we shall look at recognizing the data that was not labeled in the first pass. Word models could also be improved using an iterative approach, that could lead to improving the labeling and cleaning up the initial labeling.

8. ACKNOWLEDGMENTS

Pramod Sankar K. and C. V. Jawahar were supported in part by the Ministry of Communication and Information Technology, Govt of India. R. Manmatha was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

9. REFERENCES

[1] Universal Library at: http://www.ulib.org.
[2] Digital Library of India at: http://dli.iiit.ac.in.
[3] E. Ataer and P. Duygulu. Matching Ottoman words: An image retrieval approach to historical document indexing. In *Proc. CIVR*, pages 341–347, 2007.
[4] A. Balasubramanian, M. Meshesha, and C. V. Jawahar. Retrieval from document image collections. In *Proc. DAS*, pages 1–12, 2006.
[5] A. Bharati, P. Rao, R. Sangal, and S. M. Bendre. Basic statistical analaysis of corpus and cross comparision. In *Proc. ICON*, 2002.
[6] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *Trans. PAMI*, 30(4):712–727, 2008.
[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, pages

886–893, 2005.

[8] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley, 2001.

[9] S. Feng and R. Manmatha. A hierarchical, hmm-based automatic evaluation of ocr accuracy for a digital library of books. In *JCDL*, pages 109–118, 2006.

[10] V. Govindaraju and S. Setlur, editors. *Guide to OCR for Indic Scripts*. Springer, Sep 2009.

[11] T. K. Ho and G. Nagy. Ocr with no shape training. In *Proc. ICPR*, pages 4027–4030, 2000.

[12] C. V. Jawahar, M. P. Kumar, and S. S. Ravikiran. A bilingual ocr system for hindi-telugu documents and its applications. In *Proc ICDAR*, pages 408–413, 2003.

[13] T. Konidaris, B. Gatos, K. Ntzios, I. E. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *IJDAR*, 9(2-4):167–177, 2007.

[14] A. Kumar, C. V. Jawahar, and R. Manmatha. Efficient search in document image collections. In *Proc. ACCV*, pages 586–595, 2007.

[15] V. Lavrenko, T. M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proc. DIAL*, pages 278–287, 2004.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[17] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. *Trans. PAMI*, 23(2):149–164, 2001.

[18] C. Mathis and T. Breuel. Classification using a hierarchical bayesian approach. In *Proc. ICPR*, pages IV: 103–106, 2002.

[19] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[20] P. Natarajan, E. MacRostie, and M. Decerbo. The BBN Byblos Hindi OCR System. *Guide to OCR for Indic Scripts*, pages 173–180, 2009.

[21] N. V. Neeba and C. V. Jawahar. Recognition of Books by Verification and Retraining. *Proc. ICPR*, pages 1–4, 2008.

[22] A. Negi, C. Bhagvati, and B. Krishna. An OCR System for Telugu. In *Proc. ICDAR*, pages 1110–1114, 2001.

[23] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, pages 2161–2168, 2006.

[24] U. Pal and B. Chaudhuri. Indian script character recognition: A survey. *Pattern Recognition*, 37(9):1887–1899, 2004.

[25] Pramod Sankar, K., Vamshi Ambati, Lakshmi Pratha, and C. V. Jawahar. Digitizing a million books: Challenges for document analysis. In *Proc. DAS*, pages 425–436, 2006.

[26] Pramod Sankar, K. and C. V. Jawahar. Probabilistic reverse annotation for large scale image retrieval. In *Proc. CVPR*, 2007.

[27] V. Rasagna, A. Kumar, C. V. Jawahar, and R. Manmatha. Robust recognition of documents by fusing results of word clusters. In *Proc. ICDAR*, pages 566–570, 2009.

[28] T. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proc. CVPR*, pages 521–527, 2003.

[29] T. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *Proc. SIGIR*, pages 369–376, 2004.

[30] T. M. Rath and R. Manmatha. Word spotting for historical documents. *IJDAR*, 9(2-4):139–152, 2007.

[31] S. V. Rice, G. Nagy, and T. A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer, 1999.

[32] J. A. Rodriguez and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *Proc. ICFHR*, 2008.

[33] P. Sarkar and G. Nagy. Style consistent classification of isogenous patterns. *Trans. PAMI*, 27(1):88–98, 2005.

[34] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, pages 1470–1477, 2003.

[35] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[36] P. Xiu and H. S. Baird. Scaling up whole-book recognition. In *Proc. ICDAR*, pages 698–702, 2009.