Recognition of Tibetan Wood Block Prints with Generalized Hidden Markov and Kernelized Modified Quadratic Distance Function

Fares Hedayati, Jike Chong, Kurt Keutzer

Abstract—Recognition of Tibetan wood block print is a difficult problem that has many challenging steps. We propose a two stage framework involving image preprocessing, which consists of noise removal and baseline detection, and simultaneous character segmentation and recognition by the aid of a generalized hidden Markov model (also known as gHMM). For the latter stage, we train a gHMM and run the generalized Viterbi algorithm on our image to decode observations. There are two major motivations for using gHMM. First, it incorporates a language model into our recognition system which in turn enforces grammar and disambiguates classification errors caused by printing errors and image noise. Second, gHMM solves the segmentation challenge. Simply put gHMM is an HMM where the emission model allows multiple consecutive observations to be mapped to the same state. For features of our emission model we apply line and circle Hough transform to stroke detection, and use class-specific scaling for feature weighing. With gHMM, we find KMQDF to be the most effective distance metric for discriminating character classes. The accuracy of our system is 90.03%.

Index Terms—Document analysis, handwriting analysis, optical character recognition, pattern Analysis

1 Introduction

There is a vast knowledge base of 14,000 volumes of Buddhist teaching stored in Tibetan wood-block print that is of great interest to scholars around the world. These ancient works are only available in physical reproductions and are difficult to access and search. There is great value and interest in bringing these ancient documents into searchable electronic form and making them accessible to everyone. The characters in wood blocks used for printing are individually carved, and contain large variations compared to modern machine prints. Neighboring characters, often merged, lines can be significantly skewed and sometimes touch one another. In many aspects, the recognition problem is as complex as handwriting recognition. There has been some previous work in optical character recognition of Tibetan machine print [1] [2] [3]. Masami et al [1] proposed a character categorization technique that separately recognizes basic consonants, combination characters and vowels. In a later work, Masami et al [1] used Euclidean distance with differential weights for the classification. In the work by Ding et al [3], a modified quadratic discriminant function is used for classification, which better shapes the region that defines a class in the feature space. We leverage the techniques presented in these previous works used in machine printed Tibetan, and extend them to wood-block print optical character recognition of Tibetan, which is yet an unsolved problem. We have encountered numerous cases in segmentation and classification where the higher level language model is required to resolve ambiguities. This motivated us to use generalized hidden Markov models, which both solves the segmentation problem and makes classification more accurate. Other approaches that are not based on gHMM are also presented here and with their overall accuracies. We tried two other approaches beside gHMM for our OCR system namely a segmentation-based approach and a thinning-based approached. The segmentation-based approach resulted in a 68.08% accuracy which is much lower than the 90.03% accuracy of the gHMM. For this approach, we used vertical histograms, DP space search, and local edges as segmentation points for segmentation. With the addition of a cost-based segmentation insertion for under segmentation cases, we were able to achieve 92% segmentation accuracy. This shows that segmentation is still a challenge in this approach and the final accuracy of the system is highly dependent on the initial segmentation. Finally the thinning based approach is based on extracting the skeleton of the text. We tried three different thinning algorithms. These methods are discussed in Section 5.2. The feature extraction of gHMM-based and segmentation-based OCRs and their distance functions are the same. This made testing the overall accuracy of these two approaches easier than that of the thinning-based approach. We did not research on ways to extract discriminative and robust features

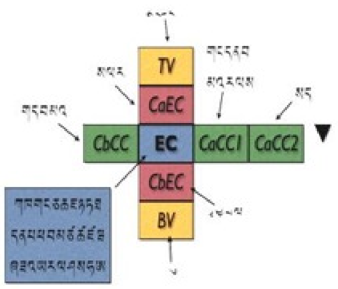

Fig. 1. General structure of Tibetan syllable (after Masami [1])

from the skeleton of the text. Researchers should come up with their own features. We only present the thinning algorithms here to give researchers insights into possible directions in Tibetan block print OCR.

2 SURVEY ON OCR AND OFF-LINE HANDWRITING RECOGNITION

Optical character recognition, abbreviated to OCR, is a field of research in computer vision, pattern recognition and digital image processing that converts images of typewritten, handwritten or printed text into machine-editable text. This process of conversion usually involves intelligent association of pieces of image with their corresponding characters; hence OCR is an artificial intelligence research field. Off-line handwriting recognition, a field of research in OCR, is the ability of computers to translate human handwriting to machine-editable text. Off-line handwriting recognition is different from on-line handwriting recognition, as the latter involves recognition of images of text as it is written on special digitizer or PDA by interpreting pen-tip movements of users. The former on the other hand gets static images of text as input. Off-line handwriting recognition is a difficult task due to variation in peoples handwriting and the amount of noise in handwritten documents. This made the researchers of the field focus on limited range of inputs. Consequently research in this field has been moving in different directions specified by range of inputs. The state of the art of this field should be studied separately for each of these ranges of inputs. Since Tibetan block prints are individually carved and printed, their recognition is an off-line handwriting recognition problem. As there has been no research on off-line recognition of Tibetan block prints (The only research that has ever been done on Tibetan OCR has been on printed Tibetan texts [1] [2] [3]), we state the state of the art of other somewhat-similar off-line handwriting recognition. One of the simplest forms of off-line handwriting recognitions is digit recognition. Handwritten digit recognition is a typical example of off-line handwriting recognition that has been extensively researched. Four major independent methods have been used by researchers of the field [12]: Shape context matching [13], pixel-to-pixel image matching with local contexts [15], invariant support vector machine [14], and convolutional neural net and virtual data [16]. The accuracy of these methods fall above 99%. In other words off-line digit recognition could be considered a solved problem, as misclassified digits of the aforementioned methods are hard for humans too, to recognize. The first two approaches of the four, use the idea of deformable templates and correspondence be-tween pixels of test and reference images. The other two methods use machine learning techniques of convolutional neural networks and SVM to classify digits. Off-line handwriting recognition of texts is a much harder problem than that of the digit recognition primarily because of segmentation problem and number of classes to be recognized. Large size of classes to be recognized is a substantial bottleneck in handwriting recognition. In most languages the number of characters is much larger than ten. For instance in Chinese, researchers are faced with thousands of characters to recognize. The other challenge is the segmentation problem. In contrast to digit recognition that classifies digits in isolation; handwritten text recognition should classify all characters of an image in context. Characters are in most languages not segmented due to image noise and writing style. Even though for most languages research on optical character recognition of printed text has been carried on, off-line handwriting recognition for many languages is a new field of research. Arabic, Chinese and scripts with Roman alphabets are among the languages with enough research on off-line handwriting recognition. We explain some of the common trends in these instances of handwriting recognition and the state of the art of each. For Chinese off-line recognition segmentation and normalization is a common approach to preprocess the image be-fore recognition [17]. Segmented characters are then recognized through stroke-based approaches, holistic approaches or radical approaches. The state of the art of Chinese off-line handwriting recognition is as following: an accuracy of 98.1%, for regular or hand-print scripts, 82.1%, for fluent scripts, and 70.1% for cursive scripts [17]. Surveys on off-line cursive word recognitions of Roman scripts show that three major approaches have been used in this field namely holistic approaches, segmentation-based approaches and HMM-based approaches. Normalization of image and some preprocessing are also common trends prior to the recognition step [18]. In segmentation-based approaches a word is first segmented to its constituent characters and each character is then recognized individually. Holistic approaches on the other hand recognize words as a whole without trying to break them into individual pieces. HMM-based approaches use hidden Markov models and their language model and emission model make them avoid segmentation. This method is adopted from speech recognition. This is despite the fact that in speech observations are one-dimensional and in text they are two-dimensional. There has been very little effort to develop two dimensional HMMs or two-dimensional stochastic models due to complexity of these models. However with increase in computation power these models could be of great use. HMM-based approaches and holistic approaches have also been a common trend in Arabic and Persian off-line handwriting recognition systems [11] [19]. Another trend in Arabic handwriting recognition has been using fuzzy constrained character graph models. These models are fuzzily labeled graphs that represent characters [20]. This method has resulted in an accuracy of 73.6%.

3 PROPERTIES OF THE TIBETAN SCRIPT

The Tibetan character set is composed of 30 consonants and 4 vowels. We define a character to be a stack of consonants with optional vowels. A syllable is a group of characters with one essential consonant (EC) and 7 optional parts, as shown in Figure 1. There can be up to one consonant before the combined character (CC) with EC, and up to two consonants after the CC. A stack with an EC is tightly packed together and some consonants change form as it is stacked.

4 OCR CHALLENGES

The characters in wood blocks used for printing are individually carved, and contain large variations compared to modern machine prints. Neighboring characters often merged, lines can be significantly skewed and sometimes touch one another. In many aspects, the recognition problem is as complex as handwriting recognition. Off-line recognition of wood block print Tibetan texts is a much more challenging task than that of the printed Tibetan texts. In single-font systems only bitmap font mask for exact match is required. In multi-font, on the other hand systems structure and contour based feature extraction is required. Since each character is well defined, with no interference from neighbors, segmentation is an easy task. This is in contrast to wood block prints where large width, height and proportion variation are common and stokes intrude from neighboring characters and lines. Character strokes merge with neighboring characters due to manual Rubbing printing techniques. In wood block prints characters merge with one another with no clear boundaries. Knowledge of character forms and language are required to disambiguate segmentation points. This problem defines the entire recognition framework. Tseks that are word delimiters in Tibetan are clearly distinguishable in machine printed texts and corner cases are resolved by rule based syntax correction. On the other hand majority of Tseks are not optically visible in wood block prints. There is significant blurring with baseline of neighboring characters and in-depth knowledge of Tibetan vocabulary is required for effective Tsek insertion. Figure 2 compares a line of wood block print with a line of machine printed Tibetan text.

Fig. 2. Printed versus wood block print

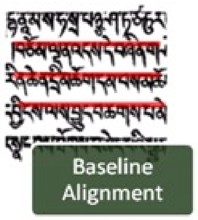

Fig. 3. Baselines aligned after recognition

4.1 BASELINE DETECTION CHALLENGES

Baselines are not always straight and parallel. Furthermore the distance between baselines is not uniform and varies from line to line and from page to page. Some-times due to noise or printing errors top of one line touches the bottom of another. All these make baseline detection hard. Figure 9 shows all these challenges that baseline detection is faced with. After baseline detection lines should be aligned equally (see Figure 3).

4.2 SEGMENTATION CHALLENGES

Segmentation of baselines into characters is one of the most challenging steps in off-line recognition of Tibetan block prints. Segmentation requires syntax knowledge. In a sense it is a chicken-and-egg problem. If we recognize the characters we can segment them easily, but we need to segment characters in order for them to be recognized (see Figure 2). Even though the majority of segmentation points could be detected by some heuristics (see Section 5.1), there are still cases that segmentation point detection is impossible without context knowledge (see Figure 4). Segmentation has direct effect on recognition. Those cropped images that have wrong segmentation points are not going be recognized correctly, no matter how accurate and well-trained the recognition system is. To segment or not to segment is a critical decision that we have to make for our OCR system. Using generalized hidden Markov models we can avoid independent segmentation and combine the step with recognition. See Section 5.3.

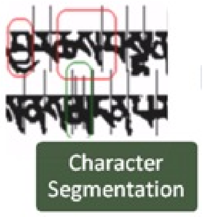

Fig. 4. Image in red box is under-segmented, the one in green is over-segmented

4.3 Recognition Challenges

As mentioned earlier at the beginning of Section 6, optical character recognition of Tibetan block prints is as hard as offline handwriting recognition. Characters that are separated by gaps in machine printed texts are not visually separable in Tibetan block prints. This makes recognition hard, because we have to know if the image we are trying to recognize corresponds to a character and is not noise or does not correspond to different parts of multiple characters. Another challenge is the amount of noise and variation in width, height, size and curvature of each character. Features that are invariant to noise, rotation, smear and other transformation are needed to over-come this challenge. In short segmentation and invariant features are needed for off-line recognition of Tibetan block prints.

5 OVERVIEW OF THE RECOGNITION APPROACHES

5.1 Segmentation-based approaches

There are two approaches for a Tibetan recognition framework based on segmentation. The first is segmenting lines into characters and recognizing the characters (see Figure 6) and the second is segmenting lines to characters, then its constituting consonants and vowels, then recognizing the consonants and vowels (see Figure 5).

The first approach involves discriminating approximately 500 common characters, whereas the second approach involves discriminating approximately 50 consonant and vowel symbols. Since characters, vowels and consonants are not completely separable in Tibetan block-prints, due to noise, line slant, smearing of the ink, etc, the first approach seems more reasonable. Here we present a pipeline of the first approach. After detecting baseline of an image we segment each line into characters. Then for each of these cropped images of characters some features are extracted which are in turn fed into a distance function. The distance function finds the closest class of characters to its input and output that class. Feature extraction and distance function of our segmentation-based approach is exactly the same as of those of the gHMM-based approach. For more information about

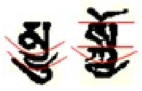

Fig. 5. Two stack characters segmented into vowels and consonants

them please refer to Sections 10 and 11. In this Section we present the segmentation algorithm of the method.

Segmenting a line into characters is a significant problem in Tibetan wood block prints. Unlike machine printed Tibetan, character width and spacing are highly variable. Searching for white spaces in the vertical histogram finds only 35% of segmentation points (see Figure 6). Top vowels and bottom vowels frequently extend into the neigh-boring characters and characters often merge into each other, sometimes at multiple points. By using dynamic programming to find points that connect the top and bot-tom white space, we obtain no more than 60% of expected segmentation points. By extracting Tibetan specific features such as tail-stroke at the right end of more than 80% of the characters, we capture up to 85% of the segmentation points (see Figure 9). The above heuristics does not find all segmentation points. By detecting gaps of segmentation points that are larger than the typical character width, we insert potential segmentation points not captured by previous heuristics. For wide gaps, multiple possible segmentation scenarios are possible. For example, a typical Tibetan wood block print character ranges from 15-25 pixels wide at 300 dpi, a gap in detected segmentation points of 70 pixels may contain 3 or 4 characters. We examine the potential segmentation sites for different segmentation scenarios, and choose the scenario with the lowest average cost. The insertion cost is calculated by a discrimination function over the vertical histogram:

$$g(m) = \frac{2V(m) - V(m-1)V(m+1)}{V(m)} + 1 \qquad (1)$$

Where $V(m)$ = the number of pixels at the vertical strip at position $x = m$ and $g(m)$. Where and is the tsek insertion cost; segmentation points are the local minima of this function. Applying this technique increases the segmentation point detection rate to 92%. There are many segmentation points that pose significant challenge for image analysis techniques to detect. They are: 1) closely linked ga and shey near the end of a sentence, 2) neighboring characters touching at more than one place, and 3) ambiguous segmentation points that provide multiple segmentation possibilities. These require some higher-level language model to help discriminate.

5.2 Thinning Based Approach

In the thinning-based approach, features are extracted from the skeleton of the image. Extracting features

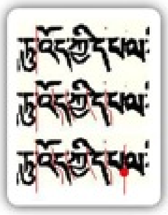

Fig. 6. Character segmentation

from the skeleton of an image has been a common approach for many OCR researchers. For instance, Khorsheed [11] uses a thinning algorithm similar to Stentiford [5] to extract the skeleton of images of Arabic manuscripts. He then traverses these skeletons and extracts different features from them. Curvature, orientation, and percentage of the skeleton above or below the baseline are some of these features. He finally feeds this features to a hidden Markov model trained to classify Arabic characters. Our initial goal was feature extraction from skeleton. However since the amount of noise in Tibetan block prints is very high we ended up using our thinning algorithms for noise removal and smoothing only. We developed three methods for thinning Tibetan block prints. Figure 7 shows outputs of these three methods and the original image at the bottom. Even though we only used one of these algorithms namely the third method, we pre-sent all three here for future researchers of the field. The first thinning algorithm, is the most sophisticated thinning algorithm of all. It is based on contour detection and iteratively shrinking the image, the pseudo-code of this algorithm is in algorithm 1.

In algorthim-1 after detecting the contour of an image, was mark its inner contour. Inner contour here is defined as the immediate black neighbors of a contour that are trapped inside the contour. If a contour pixel does not have any neighbors that is trapped in, that pixel is also included in the inner contour. After this step, the current contour is updated by the inner contour that was just marked, and the original image is shrunk by removing the old contour. This step is repeated iteratively until the skeleton of the image is found and the contour does not change anymore. In each step of the iteration we make sure that the image remains connected by identifying joint points on the contour and putting them back on the new contour. Joint points are contour pixels that are neighbors with no more than two inside pixels and have at least a contour pixel neighbor that does not have any inside neighbors. The output of this thinning algorithm is further smoothed and thinned by yet another skeletonizer called Stentiford [5]. The second thinning algorithm, is based on contour detection

Algorithm 1 Input: binary image β, Output: Thinned image γ

```
while true do
    $\delta \leftarrow$ ContourDetection($\beta$)
     only keeps black pixels that have
     at least 1 white neighbor
    $\eta \leftarrow$ InnerContourDetection($\delta, \beta$)
     only keeps black pixels in $\beta$
     that are adjacent to black pixels of $\delta$
    $\hat{\beta} \leftarrow \beta$ - RemoveJointPoints($\delta$)
     RemoveJointPoints removes black pixels
     that are in $\delta$ and are neighbors with no
     more than two pixels in $\beta - \delta$ and have
     at least a neighbor in $\delta$ that does not
     have any neighbors in $\beta - \delta$
    $\delta \leftarrow \eta$
    if $\hat{\beta} \neq \beta$ then
        Break
    else
        $\hat{\beta} \leftarrow \beta$
    end if
end while
$\gamma \leftarrow$ Stentiford ($\beta$)
return $\gamma$
```

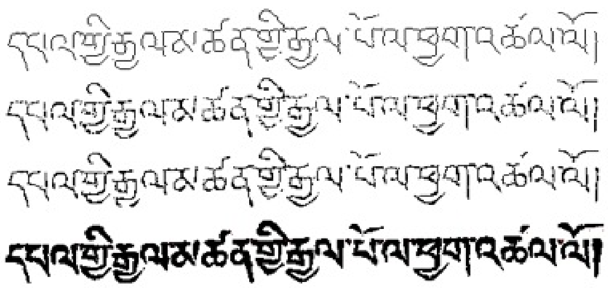

Fig. 7. The fourth line is the original image and the other three lines are outputs of our thinning algorithms.

and iteratively shrinking the image like the previous method. However it is much simpler and it does not use the skeletonization algorithm on the output. The third thinning method, corresponding to the third line in figure 6, is exactly like the second thinning method where we iteratively shrink the contour until it converges to the skeleton, with one exception. In this method when we are updating our contour by inner contour we keep any contour pixel that has more than four black neighbors. This way we keep the contour more connected compared to the other method. Algorithm 2 explains this algorithm:

5.3 gHMM Based Approach

Segmentation and classification are the two inter-dependent stages in our OCR project. Facing the same problem for optical character recognition of Latin manuscripts Jaety Edwards and et al used generalized

Algorithm 2 Input: binary image β, Output: Thinned image γ

```
while true do
  $\delta \leftarrow$ ContourDetection($\beta$)
   only keeps black pixels that have
   at least 1 white neighbor
  $\eta \leftarrow$ InnerContourDetection($\delta, \beta$)
   only keeps black pixels in $\beta$
   that are adjacent to black pixels of $\delta$
  $\hat{\beta} \leftarrow \beta$ - RemoveJointPoints($\delta$)
   RemoveJointPoints removes black
   pixels that are in $\delta$ and have more
   than four black neighbors
  $\delta \leftarrow \eta$
  if $\hat{\beta} \neq \beta$ then
    Break
  else
    $\hat{\beta} \leftarrow \beta$
  end if
end while
$\gamma \leftarrow \beta$
return $\gamma$
```

hidden Markov models [10]. We have encountered numerous cases in segmentation and classification where the higher level language model is required to resolve ambiguities. [3] proposed an iterative approach, where character recognition result is fed back to the character segmentation and classification stages. gHMM based model uses a structured approach for incorporating language model based on the generalized Hidden Markov Model (gHMM). Simply put gHMM is an HMM where the emission model allows multiple consecutive observations to be mapped to the same state. The problem to be solved is to distinguish good segmentation points and bad segmentation points based on character transition model and state emission model. gHMM is characterized by three constituents: State transition model, initial state, and state emission model. The language model is encoded in all three parameters. For OCR, the hidden states are the characters to be recognized labeled with their relative position in the word, and the observations are the images at the input. Character to character transition model is encoded in the state transition model and the initial state. The emission model then helps to compute the probability that an image is an instance of a state. Given a sequence of observations our gHMM uses a variation of the Viterbi algorithm to find the most probable sequence of characters that produced the image.

6 Our Approach: gHMM

We find the gHMM based approached most promising and most accurate. This is because it recognizes Tibetan characters in context not in isolation by us-ing a language model of Tibetan language that incorporates grammar into its recognition system. Furthermore it solves the segmentation and classification problem simultaneously. It finds the best way to segment characters and the most likely sequence of characters that could correspond to an image of a line of Tibetan block print. This method increases the accuracy of a segmentation-based model by 25%.

7 Components of gHMM

7.1 Emission Model: Feature Extraction and Observation Probability

Emission model is the common part of all classification algorithms in general and OCR systems in particular. Emission model consists of two parts: feature extraction and probability distribution of feature vectors for each class, these probability distributions are known as observation probabilities. Feature extraction is the process of converting our input data into a vector of features. All classification algorithms should eventually go through this process. The more descriptive the extracted features the more accurate the final system. We extract 157 features for our OCR system. These features are discussed in details in Section 9. The observation probability can be also used to define a distance function which measures the closeness of feature vectors to each other. Without a proper distance function that groups vectors of the same class together and separates them from vectors of other classes, classification would be impossible or very inaccurate. There are four different distance functions that we worked with. Section 10 explains them meticulously.

7.2 Language Model

Hidden Markov Models have this language model part which incorporates transitional and initial probabilities of a set of states, in our case characters, into their system. This basically helps the recognition system to avoid un-likely and ungrammatical sentences and take the gram-mar of the language into account. Section 11.1 explains this part in details.

7.3 Viterbi Decoder for Simultaneous Character Segmentation and Character Classification

Viterbi decoder is the most essential part of a gHMM system. It is the heart of the system. gHMM could be thought of a black box that has raw image of a line of Tibetan as its input and the most probably sequence of Tibetan characters with its corresponding segmentation points as its output . The algorithm that does this most favorable and simultaneous segmentation and classification is called Viterbi. Refer to Section 11.3 for a through explanation of the algorithm.

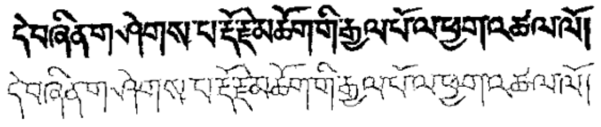

Fig. 8. Line of Tibetan block print with its skeleton

8 Preparing the Input to gHMM: Image Preprocessing

8.1 Noise Removal

One of the main challenges in recognition of Tibetan block prints is that they are very noisy. Noise in computer vision is defined as those pixels that are not part of the image. They make recognition hard. Noise removal is an essential step in our OCR pipeline. In this step all pixels are traversed one by one and those pixels that do not have more than 2 non-white neighbors are whitened, i.e. deleted. This way noisy pixels and dots are removed and the image is cleaned.

8.2 Thinning Thick and Noisy Characters

Noise is not restricted to random dots. Smeary characters are considered noise too. The characters in wood blocks used for printing are individually carved, and contain large variations compared to modern machine prints. Some of these variations are not tolerable. More specifically when a character is too smeary and thickened by printing errors, recognizing it becomes very hard. These characters are thinned by algorithm 2 in section 5.2 that extracts the skeleton of characters. After thinning we add pixels around the skeleton in two steps. In the first step direct neighbors of the skeleton are added to the thinned image which originally contained the skeleton only, and in the second step the immediate neighbors of the thinned image are added. This way the pixels that are either in skeleton or at most two pixels away from it are included in the final thinned image. Figure 8 shows a line of Tibetan block print with its skeleton extracted using our skeleton extraction algorithm. For more details of this thinning algorithm refer to Section 5.2.

8.3 Baseline Detection

The lines of text in Tibetan are aligned to a baseline near the top of the characters. This feature can be used to align Tibetan text lines in an image for effective character window search. However, in Tibetan texts are written in wide and short pages. The significant variations in line slant angle and line spacing across a page makes any global baseline search technique ineffective and error prone, see left of Figure 9. We use a horizontal histogram of a typical line of Tibetan text learnt from samples, to dynamically detect local line offset variations in short segments of 100-200 pixels across the page, see right of Figure 9. For each local window which is usually 60 x 100-200 pixels we build a horizontal histogram that contains the number of black pixels along each y position in the window. This histogram is then compared to our typical histogram learned from training samples and if they are close enough, the vertical offset of the window or its local baseline is detected. The vertical offset for each pixel is computed by interpolating neighboring measurements. Each line is separated to avoid over writing neighboring lines, and pixels are shifted rather than rotated to gain computation efficiency. This way, the lines are aligned for character segmentation.

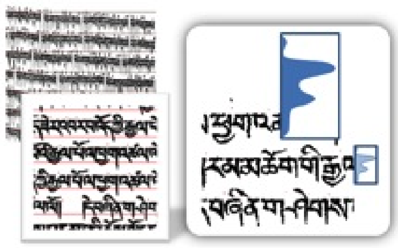

Fig. 9. Baseline detection using horizontal histograms

9 Emission Model of gHMM, Part I: Feature Extraction

The segmentation points found in the section 5.1 are used to divide each line of text into individual characters. For training the classifier, we call the characters class templates. Several classes of features are considered for wood-block print OCR.

1. Skeletonization and stroke detection (structure based approach) 2. Histograms and profiles (texture based approach) 3. Hough transform and variations (structure and texture based approach)

Skeletonization and stroke detection has been used successfully in many handwriting OCR systems [4]. The approach thins a stroke down to single pixel width, while keeping the connected components connected. We developed three thinning algorithms. While these approaches are suitable for handwriting with thin strokes, the strokes in Tibetan wood block print are dense and thick, and the Skeletonization process generates considerable systematic noise. These thinning algorithms produce split ends at the tip of a stroke, and have difficulties thinning dense junction points. Our experiments with vertical and horizontal histograms and pro-files showed that histograms are sensitive to stroke thick-ness and profiles are easily affected by cross-sections generated while segmenting merged characters. We use Hough transform as a basis for stroke detection. Sections 9.1 and 9.2 explain stroke detection in details. The Hough transform is an image space transformation that maps points in an image in x-y Euclidean plane into the Hough space, defined by θ and ρ. By checking for the strongest response in

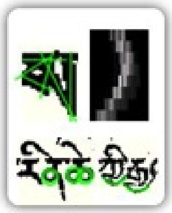

Fig. 10. Lines and circles detected using Hough Transform

Hough space, we can quickly find the most prominent line feature in the image, see Figure 10.

9.1 Hough Line Extraction

The Hough transform has been frequently used to ex-tract features in image outlines [6]. Tibetan wood block print is not suited for such an extraction approach. Firstly, a character outline may be badly distorted by noise and merged characters. Secondly, the thin outlines require precise matches with angles (θ) resulting in expensive computation time making features less robust to rotational variations. In our work, we compute the Hough transform from preprocessed image data, i.e. with noise removed and too thick characters thinned to normal size. Preprocessing keeps the strokes thick and only makes the smeary and too thick stokes look like other strokes. Extracting Hough response from images with thick strokes has a positive and a negative implication on feature ex-traction. The positive implication is that a thick stroke can respond strongly to lines of many angles, so only a few angles need to be tested to collect a significant response from a stroke. This makes the stroke more rotationally in-variant, as small amount of rotations will still yield a prominent response. The negative implication is that thick strokes add considerable noise for Hough transform responses, as two or three cross sections of thick strokes yield a response as high as an average length stroke. To mitigate this problem, we verify the high response in the Hough space by checking the collinear pixels in the x-y plane that produced the response. We define a validated prominent stroke to be a stroke that has a certain number of consecutive pixels. The Hough space efficiently stores a wealth of information about the character. As illustrated above, the strength of the response at a (θ ,ρ) position illustrate how many collinear points lie on that line in x-y plane. We can also examine the response along the ρ axis in Hough space for a particular orientation θ . This shows us the parallel strokes found in the orientation θ at various offsets. For example, long vertical strokes that appear at the right side of the image will have a significant response at $\theta = 0$ and ρ at a particular offset corresponding to the right side of the image. This information helps determine the location of the response. We use the Hough transform to extract a total of 90 features. Analysis shows that 6 orientation of equally spaced θ is enough to capture the strokes in Tibetan. For each stroke orientation, we bin the captured stroke by their verified consecutive length to long, medium, and short segments. For each bin, five features are extracted: number of verified prominent stroke, maximum and mini-mum offset ρ of verified prominent stroke, average offset ρ of prominent strokes, and average offset ρ of all responses, prominent and non-prominent. The non-prominent responses are those lines with enough collinear points to trigger a response but not enough points are found to be consecutive. These features help capture both structural and textural information in the character. The extraction of 6 orientations on thick strokes provides us with local rotational invariance. The binning of stroke length provides invariance in size of strokes, which vary significantly over characters, even on the same line. Small variations of the stroke location in the direction of the stroke do not affect the features. Small variations of the stroke location perpendicular to the stroke will only translate to small offset in offset ρ, thus providing local translational invariance.

9.2 Hough Circle Extraction

Tibetan scripts contain many loops curves that are not easily extracted with straight line strokes. We extend the Hough transform to produce 13 more features for detect-ing circular and half ellipsoid features to capture loops in cha, chha like characters, and curves in ya-ta. We found the circle response to be highly sensitive to the circle radius. Analysis shows that three bins of radii produced the most discriminating features: those that capture small circles as in cha, chha, tsha, zha, tha, a, ya, la, sha and finally those that capture larger circles as in ba, kha and ga, and those that capture half ellipsoids as in characters with ya-ta consonants. For each of the circles detected, we construct 3 bins for their locations: those that touch the top baseline, those that are one stroke away from the baseline, and those that are far below the bottom baseline. Each bin contains 2 features, number of circles detected above a response threshold and the maximum response produced in that bin. Since there are at most two circles in each bin this method of extracting features related to circles is invariant to shift along the x-axis. As the y locations of circles are binned to top, middle, or bottom, the method is also lo-cally invariant to shift along the y-axis. For the half ellipsoidal detected, one binary feature is constructed corresponding to the presence of a half ellipse of a certain size oriented downwards in the bottom part of the template.

9.3 Character Complexity Features

In addition to stroke base extraction, we also extract a complexity measure for characters by counting the number of stroke-crossings encountered in two orientations, the horizontal and the vertical orientations. Some characters like da have a simple structure and their maximum number of horizontal or vertical crossings is never more than three, whereas for more complex characters like wa the number of crossings in either direction is at least three. We have allocated ten features for each template to capture the complexity of their structure. These features are built as follows. We construct horizontal and vertical histograms of the number of crossings in the correspond-ing directions. The total number of crossings is recorded for vertical and horizontal orientations.

9.4 Directional features

We partition the image into eight horizontal and four vertical strips. For the horizontal strips we extract the following features: the number of black pixels across each column, the height of the highest pixel in each column, the number of horizontal crossings for each column, the mean grayscale of the strips, and the x-mean and the y-mean of each column. For the vertical strips we only extract the mean number of horizontal crossings for each row. In summary we extract 38 directional features from the image.

9.5 Additional Features

Finally we extract five basic features from the image: the grayscale of the image, the width of the image, the top vowel of the image if there is any, and the x-mean and the y-mean of the image. For the top-vowel feature, we used a simple decision tree algorithm to classify the top part of the image into five classes, one for each possible top vowel and one for the case when there is no vowel. The decision tree is very simple and works as following. First the top part of the baseline which contains vowels only is cut from the image and sent to a vowel classifier. This vowel classifier then detects intervals where there are contiguous chunk of black pixels. These chunks are vowels. Now on each of these vowels our classifier runs the following decision tree. If the number of horizontal cross-ing and vertical crossing is each at least two then the vowel is marked i, otherwise if the beginning and the ending part of the vowel are local maxima then the vowel is marked o otherwise if the slope of the vowel is close to negative one, then the vowel is marked e, otherwise if the vowel is less than fifteen pixels wide it is marked n otherwise p. e, o and i are top vowels and n is the Tibetan mark tsa-phru that is put on top of some characters like tsa. Finally p is the right part of the top vowel o. Since in some wood prints this vowel is dis-joint our classifier might recognize it as

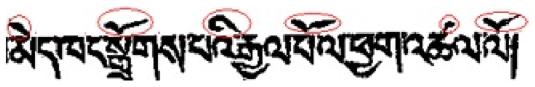
Fig. 11. Top vowels are marked with red ellipses, top vowels from left to right are, e, o , i, o, n and o

a two separate vowels, e or n as its left part and p as it right part. Based on this observation our classifier checks all vowels that are identified as p and if their previous vowels are e or n, it combines the two vowels into o. See Figure 11 to see how these vowels look like in Tibetan block prints.

9.6 Feature summary

In summary, we have a feature set of 157 features, with 90 features for 6 orientations of straight line strokes, 13 features for loops and curves, and 10 features as measure of character complexity, 5 basic features and 38 directional features.

10 Emission Model of gHMM, Part II: Distance Function

Tibetan requires a large number of classes to be discriminated. In such situations, previous works [7] [3] have proposed multilevel hierarchical classification. In this work, for simplicity, we start with a single level of classification based only on the centroid of classes. This approach requires an effective method to measure the close-ness of a sample point to a class. The set of features de-scribed above contains measures for feature counts, off-sets, and peak responses. The features of different numerical range are first normalized to the range (0,1). This normalized feature space, however, is highly inseparable with respect to the classes. Classification using Euclidean distance achieves only 48% accuracy for a correctly segmented image. The issue arises because we have a global uniform weighting on all the features. Certain characters such as ka and ja has prominent stroke features, but noisy circle features; whereas cha and chha has prominent circle features, but noisy stroke features. To increase the signal-to-noise ratio for measuring closeness to a class, we adopt a class-wise feature weighing approach, where features are weighed differently depending which class they are measuring distance with. In general, this approach shapes the high dimensional sphere in the feature space around a class centroid into an ellipsoid to better approximate the feature space belonging to a class. We experiment with four different distance metrics:

1. Weighted Euclidean Distance, 2. Weighted Euclidean Distance with Multi-cluster Classes, 3. Modified Quadratic Discrimination Function (MQDF) [8], 4. Kernelized Modified Quadratic Discrimination Function (KMQDF) [9].

The weighted Euclidean distance approach measures the variance in each feature for all training template within a class. The difference in each feature between a test sample and a class is then scaled by the inverse of the feature-wise variance in this class. This is a degenerate case of the Mahalonobis Distance where the covariance matrix is a diagonal matrix. There are two corner cases in this approach to handle. When a feature is very consistent in a class, it has a very low variance that can lead to numerical stability problems in the variance computation step, and can allow too much weight to be placed on a particular feature. We solve this by setting a lower bound on the variance, thus limiting the weight one can put on a particular feature. The second corner case occurs when we only have one sample for a class, in which case we set the variances uniformly across all features. A class-wise weighting for features helped increase prediction accuracy significantly to 67.5% for a correctly segmented image. The classification results for weighted Euclidean distance indicate that simpler characters are predicted with less accuracy, as some prominent features tends to have large variations depend-ing on the robustness of certain features. We mitigate this problem by splitting simpler, more populous classes into multiple clusters using k-means algorithm. The number of clusters in each class is proportional to the number of samples in that class. Those clusters that have population less than a certain percentage of the average population of a cluster in a class are treated as outliers, and not used in the classification phase. There are two parameters that should be determined, namely the threshold for outliers and the scaling factor for the number of clusters in each class. These two parameters are determined by an exhaustive search in the parameter space. The weighted Euclidean distance can only construct ellipsoid in the feature space in the direction of the axis. To better approximate the data variation for a class, we use Principle Component Analysis (PCA) to extract directions of most variance, and apply MQDF as a measure of the distance be-tween a sample and a class. Applying this metric improved the prediction accuracy to 75% for a correctly segmented image. Finally KMQDF is the Kernelized ver-sion of MQDF. It embeds the data in a higher dimensional Hilbert feature space which presumably makes it more separable. We use Kernel trick in KMQDF to avoid directly mapping data to its new feature space. The family of Kernel functions that we used was polynomial of the form $k(x, y) = (x.y + 1)^p$. We found to be the most accurate Kernel. For further details please refer to 11.2. Kernel trick only needs to know the dot product of the newly-mapped data which is easily computed as a function of the input in its original feature space. Apply-ing this metric improved the prediction accuracy to 80% for a correctly segmented image.

10.1 MQDF distance function

We can model each class by a multivariate Gaussian distribution, and use the negative of its log-probability as a distance function. This is exactly the aforementioned Mahalonobis distance function. The reason that we are deriving our distance function from a probability distribution is that we want to use it later in our generalized hidden Markov model as our emission model. Let w_i denote the ith class, μ_i denote the mean of the ith class and Σ_i its covariance, the probability that x is generated by class w_i is:

$$p(x|w_i) = (2\pi)^{\frac{-d}{2}} |\Sigma_i|^{\frac{-1}{2}} \exp \frac{-(x-\mu)^{\mathrm{T}} \Sigma_i^{-1} (x-\mu)}{2} \tag{2}$$

We denote the distance of x to w_i as $Q(x|w_i) = -\log p(x|w_i) =$

$$\frac{d}{2}(2\pi) + \frac{1}{2}|\Sigma_i| + \left(\frac{(x-\mu)^{\mathrm{T}} \Sigma_i^{-1} (x-\mu)}{2} \right) \tag{3}$$

We use negative of the log likelihood for distance function for two reasons. The first reason is that log likelihood has a nice form for normal distributions and the second reason is that likelihood and distance are reversely related. As the likelihood of a class w_i increases, the distance of x to that class decreases. If we diagonalize the covariance matrix by $\Sigma_i = \beta_i \Lambda_i \beta_i^{\mathrm{T}}$, where Λ_i is the diagonal matrix of eigenvalues of Σ_i and β_i is its corresponding orthonormal matrix of eigenvectors, the Mahalonobis distance function becomes

$$\frac{1}{2} \left(\sum_{j=1}^{d} \frac{\beta_{ij}^2 (x-\mu_i)^2}{\lambda_{ij}} + \sum_{j=1}^{d} \log \lambda_{ij} + d \log 2\pi \right) \tag{4}$$

Since the training of the Mahalonobis classifier always underestimate the patterns eigenvalues by limited sample set, the minor eigenvalues become some kind of un-stable noises and affect the classifiers robustness. By smoothing them in the MQDF classifier, not only the classification performance is improved, but also the computation time and storage for the parameters are saved. In MQDF we substitute minor eigenvalues including those that are zero with a small class-dependent constant δ_i and we only use K major eigenvectors, approximating the rest by the same δ_i. Putting it all together we get, distance from x to class w_i is $Q(x|w_i) = -\log p(x|w_i) =$

$$\frac{1}{2}(\sum_{j=1}^{k} \frac{(\beta_{ij}^{\mathrm{T}}(x-\mu_i))^2}{\lambda_{ij}} + \sum_{j=1}^{k} \log \lambda_{ij} +$$

$$\sum_{j=k+1}^{d} \frac{(\beta_{ij}^{\mathrm{T}}(x-\mu_i))^2}{\delta_i} + (d-k)\log \delta_i + d \log 2\pi) =$$

$$\frac{1}{2}(\sum_{j=1}^{k}\frac{(\beta_{ij}^{\mathrm{T}}(x-\mu_i))^2}{\lambda_{ij}}+\sum_{j=1}^{k}\log\lambda_{ij}+$$

$$\frac{r_i(x)}{\delta_i}+(d-k)\log\delta_i+d\log 2\pi) \quad (5)$$

Where $r_i(x)=||x-\mu_i||^2-\sum_{j=1}^{k}\left(\beta_{ij}^{\mathrm{T}}(x-\mu_i)\right)^2$. For more details refer to [8].

10.2 KMQDF distance function

Yang and et al first used KMQDF for facial expression recognition and showed that it outperforms MQDF [9]. KMQDF is the Kernelized version of MQDF. It embeds the data in a higher dimensional Hilbert feature space which presumably makes it more separable. We use Kernel trick in KMQDF to avoid directly mapping data to its new feature space. Kernel trick only needs to know the dot product of the newly-mapped data which is easily computed as a function of the input in its original feature space. Our experiments show that $k(x,y)=(x.y+1)^p$with $p=1$ is a good embedding kernel function for our 157 features. For mathematical details of deriving the KMQDF distance function refer to [9]. Here we state the final distance function and explain its parameters. KMQDF distance function looks like equation (5):

$$\frac{1}{2}(\sum_{j=1}^{k}\frac{M}{\lambda_{ij}^2}\left(r_{ij}^{\mathrm{T}}\left(R_{it}-1_M R_{it}-R_i 1_{M^{-1}}+1_M R_i 1_{M^{-1}}\right)\right)^2$$

$$+\sum_{j=1}^{k}\log\frac{\lambda_{ij}}{M}+\frac{r_i(x)}{\delta_i}+(d-k)\log\delta_i+d\log 2\pi) \quad (6)$$

Where

$$r_i(x)=||\phi(x)-\mu_i^{\phi}||^2-\sum_{j=1}^{k}\left(\beta_{ij}^{\mathrm{T}}\left(\phi(x)-\mu_i^{\phi}\right)\right)^2=$$

$$\phi(x).\phi(x)-2\times 1_{M^{-1}}R_{it}+1_M R_i 1_{M^{-1}}-$$

$$\sum_{j=1}^{k}\frac{1}{\lambda_{ij}}\left(r_{ij}^{\mathrm{T}}\left(R_{it}-1_M R_{it}-R_i 1_{M^{-1}}+1_M R_i 1_{M^{-1}}\right)\right)^2 \quad (7)$$

ϕ is the mapping function , we do not need to know this mapping because of the kernel trick, i.e.$ker(x,y)=\phi(x).\phi(y)$

M is the number of samples for our class.
$R_{it}$is a $1\times M$ matrix which contains the dot products of all the M samples of our class with the input x after being mapped to their new feature space, kernel trick is being used to fill in this matrix.
R_i is a $M\times M$ Gramm matrix which contains the dot products of all the M samples of our class after being mapped to their new feature space, kernel trick is being used to fill in this matrix.
1_M is a $M\times M$ matrix with all elements equal to $\frac{1}{M}$
$1_{M^{-1}}$ is a $M\times 1$ matrix with all elements equal to $\frac{1}{M}$
λ_{ij} and r_{ij} are the eigenvectors and eigenvalues of R_i

We used up to 125 samples for each class, i.e. our maximum equals 125. This is a tradeoff between accuracy, computation and memory. Storing all our samples, their eigenvalues and eigenvectors requires a lot of memory and is not practical. At the same time, computing dot products and matrix multiplications of size is very time consuming. Our experiments show that setting our maximum to 125 does not affect the accuracy of our model while saving us time and memory.

11 Putting All Components Of gHMM Together

Segmentation and classification are the two inter-dependent stages in our OCR project. We have encountered numerous cases in segmentation and classification where the higher level language model is required to re-solve ambiguities. [3] proposed an iterative approach, where character recognition result is fed back to the character segmentation and classification stages. We use a structured approach for incorporating language model based on the generalize Hidden Markov Model (gHMM). Simply put gHMM is an HMM where the emission model allows multiple consecutive observations to be mapped to the same state. The problem to be solved is to distinguish good segmentation points and bad segmentation points based on character transition model and state emission models. gHMM is characterized by three parameters: State transition model, initial state, and state emission model. The language model is encoded in all three parameters. For OCR, the hidden states are the characters to be recognized labeled with their relative position in the word, and the observations are the images at the input. Character to character transition model is encoded in the state transition model and the initial state. The emission model then helps to compute the probability that an image is an instance of a state. Given a sequence of observations our gHMM uses a variation of the Viterbi algorithm to find the most probable sequence of character that produced the image. Section 11.1 talks about the language model of our Tibetan gHMM, Section 11.2 gives an over-view of the emission model. In Section 11.3 we describe the decoding problem and the Viterbi algorithm in gHMM and in Section 11.4 we explain the training process and forced alignment.

11.1 Language Model

The language model consists of the transitional and initial probabilities of a gHMM. It is an essential element in gHMM in that it disambiguates segmentation and classification errors. We tried two different language models for our gHMM. In the first, we treated stack characters as states and in the second, we labeled each stack character with its relative position in the word and the size of the word. The second model resulted in a more accurate gHMM. The reason behind this is very intuitive; the second model enforces the grammar more and discards less grammatically correct outputs in the Viterbi algorithm. We label our states in the following way. A stack characters label contains the information about a characters relative position in a word and the size of the word. Possible labels are: 1-1, 1-2, 2-2, 1-3, 2-3, 3-3, 1-4, 2-4, 3-4, 4-4. As an example 2-3 means the character is the second character in a three-character word. Here we describe two of the major advantages of using labeled stack characters vs. unlabeled stack characters. Labeled stack characters constitute a more accurate language model; hence they do not allow certain paths in the decoding trellis that are otherwise allowed in the unlabeled model. At the same time certain paths are returned with more probability in this model than in the unlabeled model, hence the overall accuracy is increased. Usually tseks, the end-of-word delimiters in Tibetan, are blurred in the original image, which makes their recognition very hard. Tsek insertion is a free by-product of our language model: any state that is the first stack character of a word, namely stack characters with the following labels 1-1, 1-2, 1-3, and 1-4, has a tsek before it.

11.2 Emission Model

The emission model contains the observation and length probabilities of all states. We tried four different distance functions which are described in Section 10. MQDF and KMQDF were the most accurate of all four. KMQDF was the most accurate of all. For more information about KMQDF refer to 11.2. As KMQDF and MQDF are negative of log likelihoods, they can be used in our emission model. They are easily converted to observation probabilities by exponentiation of the negative of the distance function. Since for numerical stability, all calculations are done in logs, we only have to change the sign of our distance function to get the log of observation probability.

11.3 Decoding problem, the Viterbi algorithm

In gHMM the probability of observing a line of image that corresponds to a certain sequence of states is:

$$p(c_1|\alpha) = \prod_{t>1} p(c_t|c_{t-1})p(d_t|c_{t-1})p(\text{image}|d_t, c_t) \quad (8)$$

Where c_t is the current state and d_t is the number of observations in the current state. The decoding problem looks for a sequence of states that maximizes the posterior probability of that sequence, given the image. Since

$$p(c_1|, c_2, ..., c_m|\text{image}) = \frac{p(c_1|, c_2, ..., c_m, \text{image})}{p(\text{image})} \quad (9)$$

the sequence of states that maximizes the joint probability also maximizes the posterior probability. Exploit-ing this mathematical fact the Viterbi algorithm finds the maximizing sequence of states by maximizing the joint probability. As the number of sequences of a certain length is exponential in length and the length of the maximizing sequence is unknown due to lack of segmentation, the total number of possible sequences is the sum of exponentials over all possible lengths, which is still exponential. The Viterbi algorithm avoids this exponential set by the aid of dynamic programming in the following way:

$$\delta_t(i,l) = \max_{c_1, c_2, ..., c_m, m} p(\text{image}(0,t), S(t,l,i)) \quad (10)$$

Where $S(t,l,i) = c_1, c_2, ..., c_m$ means that observations $t - l$ to t of the image corresponds to the ith state. Let a_{ij} denote the transition probability from the ith state to the jth state, and$b_i(t,l)$ be the observation probability of observing observations $t - l$ to t of the image given we are in the ith state, and let $\pi(i,l)$ be the initial probability of the ith state with length l then it is trivial to derive the following recursive formula:

$$\delta_t(i,l) = b_i(t,l) \max_{j,w} \delta_{t-1}(j,w)a_{ij} \quad (11)$$

$$\delta_0(i,l) = b_i(t,l)\pi(i,l) \quad (12)$$

The Viterbi algorithm retrieves the maximizing sequence of states by looking for the maximum of over all possible states and possible lengths and tracing back from there. In this formula T is the length of the image, i.e. number of observations.

11.4 Training

Training is the most challenging task of our gHMM. Figure 12 shows a line of Tibetan block print with its segmentation points. Supervised and manual segmentation and labeling of observations is time consuming and unpractical. Having initialized our emission model with seven pages of manually-segmented block prints, we used the automatic and unsupervised technique of forced alignment to further train our model. Researchers use forced alignment algorithm, one type of a hard EM algorithm, when the corresponding hidden states are given and only segmentation points are missing. Our training data consists of images of lines of Tibetan block prints with their corresponding sequence of Tibetan characters, i.e. the ground truth. Since our training data lack segmentation points, we treat them as latent variables with indices of state sequence as our states. For instance if we know that our input image (which is one line of Tibetan block

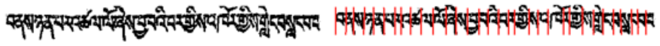

Fig. 12. A line of Tibetan block print, the red lines corresponds to the segmentation points.

print) corresponds to a 20-long sequence of Tibetan characters, then the states of our EM are the indices of this sequence, i.e. 0 to 19. Hard EM finds the most probable alignment, by running the Viterbi algorithm on our new set of states, with the following changes:

a. Each column of the trellis is initialized with our new set of states. b. For each state i only i-1 is considered as the possible previous state not other states. c. The transition probability and initial probability are not taken into consideration here, because the sequence of states is given and we are only maximizing over the observation probabilities, in other words we are looking for a segmentation that maximizes the product of observation probabilities, given the sequence of states.

Having found the most likely segmentation we can re-estimate our gHMM, and with our new gHMM we can re-segment our images. We should continue this cycle till our model becomes stable. We trained our model using fifty pages of Tibetan block prints. After twenty iterations of forced alignment our model output the same alignments meaning that it converged to a stable solution.

12 Results

We have constructed a database of 6555 characters from 7 pages of Tibetan wood-block print for our experiments. The document used is from the publisher Dege Kanjur, titled 400 Praises to the Buddha. Our sample had 168 classes of stack characters, with a maximum of 699 instances per class. We used cross validation to train and test our model. We manually segmented and labeled all pages of Dege Kanjur for training and initializing the gHMM. We ran forced alignment with a total of twenty iterations on yet another collection of training samples with a total number of 54 pages to further improve our gHMM. This was done for each cross validation step. Our measurement of accuracy was Levenstein distance function. Levenshtein distance is a measure of closeness of two sequences to each other. The Levenshtein distance between two strings is given by the minimum number of insertion, deletion, or substitution of characters to convert one string to another. Our gHMM resulted in 89.49% accuracy using the Levenshtein distance for comparing Viterbis output to the ground truth. Later on we manually segmented 104 pages and retrained our OCR on them to see how accurate our forced alignment EM was compared to the case when the latent variable, i.e. segmentation points, were available. Manual segmentation increased the accuracy to

TABLE 1
Error averages of four OCR systems

Thinned with 104 Manually segmented pages	8.712834
Thinned with 54 Manually segmented pages	8.862216
Not Thinned EM	12.40975
Thinned EM	10.5180

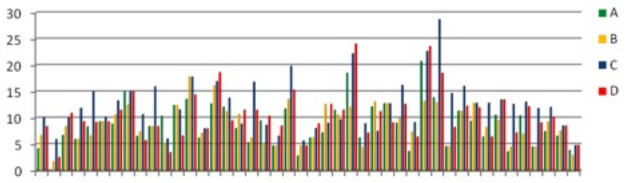

Fig. 13. Comparison of errors of four OCR systems, A (green) represents thinned with 104 manually segmented pages, B (yellow) represents thinned with 54 manually segmented pages, C (blue) represents not thinned pages with EM, D (red) represents thinned with EM.

91.29% which only boosted up the accuracy by 1.8%. This shows the effectiveness of our training with EM. Table-1 shows four error rates of four different OCR systems: two OCR systems with training EM, one without thinning and preprocessing and one without preprocessing, and two OCR systems with manually-segmented pages, one with 54 pages and one with 104 pages. Table 1 further shows that thinning and image preprocessing increases the overall accuracy, and increasing the training samples from 54 pages to 104 pages increases the accuracy by only by 0.15%, in other words more training does not have a noticeable effect on the overall accuracy.

13 Conclusion

In examination of approaches to solving the OCR problem for Tibetan block prints, one critical decision is to segment or not to segment the image. This problem is common to other off-handwriting recognitions, such as Arabic, Roman and Chinese OCR systems. Our results indicate that segmentation and recognition should not be too separate steps. One needs to recognize characters in image in order to segment them and characters are segmented in order to be recognized. This chicken-and-egg problem is solved simultaneously by the aid of gHMM. In other words gHMM carries segmentation and recognition at the same time. In conclusion gHMM-based approach outperforms segmentation-based approaches for optical character recognition of Tibetan block prints. Furthermore correct choices of language model and distance function with noise-invariant, discriminative features for gHMM and enough training of the gHMM has substantial effects on the final accuracy of the overall system. More specifically Kernelized MQDF distance function outperforms MQDF which was initially used in our system and per-formed better than Mahalonobis-based and Euclidean distance functions. For language model of gHMM, characters labeled with their relative distance to tseks and size of their word outperformed vanilla language model that lacked these labels. This is due to the fact that labels of relative distance to tseks and size of words enforce more grammar into language model than the vanilla model. Finally using Hard EM to train our gHMM, makes the emission model more robust and accurate.

14 Future Work

Our experiments show that good features could easily increase the accuracy of recognition even if the distance function is not discriminative enough. Hence one of the directions that future researchers should take is improving the qualities of features that would be eventually used in the GHMMs emission model. Good features should have two characteristics. One is that they should be discriminative, i.e. they should be able to separate classes of characters from each other. Secondly features should be invariant to different types of noise, rotation and smear. Good features however do not eliminate the need for generalized hidden Markov models, as they do not solve the segmentation problem. Higher levels of language model could also increase the accuracy of our approach. As in speech recognition instead of the most probable sequence of states, the Viterbi algorithm could output a certain number of them. A higher-level language model could further process these outputs using this model and filter the unlikely and ungrammatical sentences out.

References

[1] K Masami, N Chikako, K Takanobu, A Yoko, and K Yoshiyuki, Recognition of similar characters by using object oriented design printed Tibetan dictionary, Trans-actions of IPSJ, 36(11), pp.23042307, 1995.

[2] K Masami, K Yoshiyuki, and K Masayuki, Character recognition of wooden blocked Tibetan similar manuscripts by using Euclidean distance with deferential weight, IPSJ SIG Note, Information Processing Society of Japan (IPSJ), pp. 1318, 1996.

[3] Xiaoqing Ding and Hua Wang, Multi-Font Printed Tibetan OCR, Digital Document Processing, Springer Lon-don, Bidyut b. Chaudhuri edition, pp. 73-98, 2007.

[4] M Khorsheed, W Clocksin, "Structural Features of Cursive Arabic Script". The 10th British Machine Vision Conference, University of Nottingham, Nottingham-UK, pp.422-431, 1999.

[5] H Freeman, Machine vision for three, dimensional scenes, New York, Academic Press 1990.

[6] R Dua, P Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM, vol. 15, no. 1, pp. 11-15 January 1972.

[7] Y Tang, Lo-Ting Tu, Jiming Liu, Seong-Whan Lee Win-Win Lin, and Ing-Shyh Shyu, offline recognition of Chinese handwriting by multi-feature and multilevel classification, Pattern Analysis and Machine Intelligence, IEEE Transactions, vol. 20, pp. 556-561, 1998.

[8] F Kimura, K Takashina, and S Tsuruoka, Modified quadratic discriminant functions and the application to Chinese character recognition, IEEE Transactions on Pat-tern Analysis and Machine Intelligence, vol. 9. pp. 149153, 1987.

[9] D Yang, L Jin, J Yin, L Zhen, J Huang, Kernel Modified Quadratic Discriminant Function for Facial Expression Recognition, Lecture notes in computer science, Advances in machine vision, image processing, and pattern analysis. pp. 66-75, 2006.

[10] J Edwards, D. A Forsyth, Searching for character models, Advances in Neural Information Processing Systems, pp.331-338, 2006

[11] M.S Khorsheed, Recognizing Handwritten Arabic Manuscripts Using a Single Hidden Markov Model. Pattern Recognition Letters, 24, pp.2235-2242, 2003.

[12] D Keysers Comparison and Combination of State-of-the-art Techniques for Handwritten Character Recognition: Topping the MNIST Benchmark, Image Understanding and Pattern Recognition (IUPR) Group German Research Center for Artificial Intelligence (DFKI), pp/ 1-18, 2007

[13] S Belongie, J Malik, and J Puzicha. Shape Match-ing and Object Recognition Using Shape Contexts. IEEE Trans. Pattern Analysis and Machine Intelligence, 24(4). pp.509-522, April 2002.

[14] D DeCoste, B Schoelkopf. Training Invariant Support Vector Machines. Machine Learning, Machine Learning, vol. 46, Numbers 1-3, pp.161-190, 2002.

[15] D Keysers, C Gollan, H Ney. Local Context in Non-linear Deformation Models for Handwritten Character Recognition. 17th Int. Conference on Pattern Recognition, vol. 4, pp. 511-514, Cambridge, UK, August 2004.

[16] P Simard. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In 7th Int. Conf. Document Analysis and Recognition, pp. 958-962, Edinburgh, Scotland, August 2003.

[17] S. N Srihari, X. Yang, G. R. Ball. Offline Chinese Handwriting Recognition: A Survey. Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on, vol. 1, pp. 133-137, 2007

[18] A Vinciarelli A survey on off-line cursive word recognition, Pattern Recognition, vol. 35, pp.14331446, 2002.

[19] A Amin, Off-line Arabic character recognition: the state of the art, Pattern Recognition, vol. 31, Issue 5, 1, pp. 517-530, March 1998.

[20] I.S.I Abuhaiba, S.A Mahmoud, R.J Green, "Recognition of handwritten cursive Arabic characters," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.16, no.6, pp.664-672, Jun 1994.